UNIVERSIDAD AUTÓNOMA DE CHIAPAS

Facultad de Ciencias en Física y Matemáticas

Apuntes del curso de
MÉTODOS NUMÉRICOS

realizados por el

Dr. Roberto Arceo Reyes

Agosto - Diciembre 2017

Tuxtla Gutiérrez, Chiapas, Mexico

Objetivo

El análisis numérico trata de la obtención, descripción y análisis de algoritmos para el estudio y solución de problemas matemáticos. El desarrollo continuo de las máquinas computadoras y su cada vez más fácil accesibilidad, aumenta a igual velocidad como la importancia de los métodos numéricos en la solución de problemas en las ciencias y la ingeniería. Gran parte de los egresados de una licenciatura en Ciencias Físico-Matemáticas tratan con problemas que requieren del uso de estos métodos. El presente curso pretende dar un panorama amplio de la gama de problemas matemáticos que se pueden resolver usando los métodos numéricos y se obtienen los algoritmos correspondientes. El curso proporciona material que debe ser conocido por todo científico o ingeniero.

El curso requiere de los conocimientos de la línea de cálculo, el manejo de un lenguaje de programación y el conocimiento de conceptos de álgebra lineal como espacios vectoriales, normas y matrices. Al término del curso el estudiante dominará los puntos esenciales del mismo, a saber: estudio y clasificación de errores, solución de ecuaciones no lineales, aproximación e interpolación, derivación e integración numérica, solución numérica de ecuaciones diferenciales ordinarias y solución de sistemas de ecuaciones lineales y habrá realizado programas de computadora de los principales métodos estudiados.

Propósito

Conocerá la importancia del análisis numérico por su aplicación a problemas clásicos: solución de ecuaciones, aproximación de funciones, solución numérica de ecuaciones diferenciales ordinarias y solución de sistemas de ecuaciones lineales; y los conceptos de algoritmo, convergencia y estabilidad de un algoritmo.

Calcular los errores absoluto y relativo de una aproximación;

· Apreciar la utilidad del valor relativo;

· Entender los conceptos de cifras significativas; programación del error en operaciones aritméticas; programación del error en la evaluación de funciones y condicionamiento de un algoritmo.

Conocer y diferenciar los métodos de bisección, secante, falsa posición e iteración de punto fijo;

· Comprender los conceptos de aceleración de la convergencia; convergencia lineal y cuadrática;

· Aplicar los métodos estudiados al cálculo de raíces de polinomios.

· Aplicar el método de Newton a sistemas de ecuaciones no lineales.

Entender los conceptos básicos de aproximación de funciones y será capaz de aproximarlas usando los diferentes criterios estudiados: a saber, mínimos cuadrados por polinomios; polinomios ortogonales; funciones splines; e interpolación polinomial.

Conocer y aplicar los métodos de derivación y sus errores de redondeo y truncamiento; y los de integración, sus fórmulas de error y la aplicación de éstas para la integración compuesta.

Dominar los métodos de Euler, Taylor; Runge-Kutta, los basados en integración numérica, Adams-Bashfort para solucionar ecuaciones diferenciales ordinarias;

· Comprender los conceptos de estabilidad numérica.

Conocer y ser capaz de aplicar los métodos directos para la solución de sistemas de ecuaciones lineales; las estrategias de pivoteo; y los métodos iterativos. Así también los métodos correspondientes para el cálculo de valores propios.

TABLA DE CONTENIDOS

Lista de Tablas

Lista de Figuras

Bibliografía

[1] Atkinson, K. (1989). An Introduction to Numerical Analysis. Wiley, 2nd edition.

[2] Hamming, R.W. (1987). Numerical Methods for Scientists and Engineers. Dover Publications, 2nd edition.

[3] Burden, R.L., Faires, J. D. (2002). Anlisis Numrico. Mxico: International Thomson Editores, S.A. De C.V.

[4] Henrici, P. (1980). Elementos de Anlisis Numrico. Mxico: Editorial Trillas.

[5] Carnahan, B., Luther, H. A., Wilkes, J.O. (1969). Applied Numerical Methods. Wiley, 1st edition.

[6] Kincaid, D.R., Cheney, E.W. (2001). Numerical Analysis: Mathematics of Scientific Computing. Brooks Cole, 3rd edition.

[7] Gibbs, W.R. (2001). Computation in Modern Physics. USA: World Scientific Publishing Company, 2nd edidition.

Capítulo 1

Introducción

El presente trabajo es un compilación de los apuntes dados en el curso de Métodos Númericos impartido en el semestre Agosto - Diciembre del 2017 en la Licenciatura en Física de la Facultad de Ciencias en Física y Matemáticas de la Universidad Autónoma de Chiapas.

1.1. Problemas clásicos del Analísis Numérico

Recordando la segunda ley de Newton,

$$F = ma \tag{1.1}$$

donde F es la fuerza neta que actúa sobre el cuerpo, m es la masa del objeto, y a es su aceleración.

La ecuación (1.1) tiene varias características habituales de los modelos matemáticos del mundo físico.

1. Describe un sistema o proceso natural en términos matemáticos.

2. Representa una idealización y una simplificación de la realidad.

3. Conduce a resultados predecibles, y en consecuencia puede emplearse para propósitos de predicción.

La aceleración de la ecuación (1.1) puede despejarse,

$$a = F/m \tag{1.2}$$

esta ecuación puede usarse para calcular la velocidad final de un cuerpo en caída libre cerca de la superficie terrestre.

El cuerpo en descenso será un paracaidista.

Figura 1.1: Representación de las fuerzas que actúan sobre un paracaídista en descenso. F_D es la fuerza hacia abajo debido a la atracción de la gravedad. F_U es la fuerza hacia arriba debido a la resistencia del aire.

Podemos crear un modelo matemático al expresar la aceleración como la razón de cambio de la velocidad con respecto al tiempo ($\frac{dv}{dt}$) y sustituir en la ecuación (1.1) para dar

$$m \cdot \frac{dv}{dt} = F \tag{1.3}$$

donde v es la velocidad.

Para un cuerpo que cae dentro del perímetro de la Tierra, la fuerza total está compuesta por dos fuerzas contrarias. La atracción hacia abajo debida a la gravedad F_D y la fuerza hacia arriba debida a la resistencia del aire F_U.

$$F = F_D + F_U \tag{1.4}$$

Si a la fuerza hacia abajo se les asiga un signo positivo, se puede usar la segunda ley para formular la fuerza debida a la gravedad como

$$F_D = mg \tag{1.5}$$

donde g es la constante de gravitacón, o la aceleración debida a la gravedad, con un valor de 9.8 m/s^2 (mks) o 980 cm/s^2 (cgs).

La resistencia del aire puede formularse de varias maneras, una es suponer que es linealmente proporcional a la velocidad,

$$F_U = -cv \tag{1.6}$$

donde c es una constante llamada el coeficiente de arrastre (en gramos por segundo o kg/s).

Combinando las ecuaciones (1.3) a (1.6),

$$m\frac{dv}{dt} = mg - cv \tag{1.7}$$

$$\frac{dv}{dt} = g - \frac{c}{m}v \tag{1.8}$$

$$\frac{dv}{(g - \frac{c}{m}v)} = dt,$$

haciendo

$$u = g - \frac{c}{m}v$$

$$du = -\frac{c}{m}dv$$

$$dv = -\tfrac{m}{c}du \qquad y \qquad -\tfrac{m}{c}\tfrac{du}{u} = dt$$

$$\frac{du}{u} = -\frac{c}{m}dt$$

Integrando

$$\int \frac{du}{u} = -\frac{c}{m}\int dt$$

(en reposo $v = 0$ en $t = 0$)

$$\ln(u)|_0^v = -\frac{c}{m}t\Big|_0^t$$

$$\ln(g - \frac{c}{m}v)\Big|_0^v = -\frac{c}{m}t$$

$$\ln(g - \frac{c}{m}v) - \ln(g) = -\frac{c}{m}t$$

$$\ln(\frac{g - \frac{c}{m}v}{g}) = -\frac{c}{m}t$$

$$\ln(1 - \frac{c}{mg}v) = -\frac{c}{m}t$$

$$(1 - \frac{c}{mg}v) = e^{-\frac{c}{m}t}$$

$$\frac{c}{mg}v = (1 - e^{(\frac{c}{m})t})$$

$$\longrightarrow v(t) = \frac{mg}{c}[1 - e^{-\frac{c}{m}t}](1.9)$$

Ejemplo: Solución analítica al problema del paracaidista que cae.

Un paracaidista con una masa de $68,100$ gramos salta de un aeroplano. Aplíquese la ecuación (1.9) para calcular la velocidad antes de abrir el paracaidas. El coeficiente de arrastre c es aproximadamente igual a 12,500 g/s.

Solución: Al sustituir los valores en la ecuación (1.9) se tiene,

$$v(t) = \frac{(980cm/s^2)(68,100g)}{12,500g/s}[1 - exp(-\frac{12500g/s}{68,100g}t)]$$

$$v(t) = (5339.04cm/s)[1 - e^{-0.18355t/s}]$$

Al dar valores a t se obtiene,

t(s)	**v(cm/s)**
0	0
2	1640.5
4	2776.9
6	3564.2
8	4109.5

10	4487.3
12	4749.0
∞	5339.0

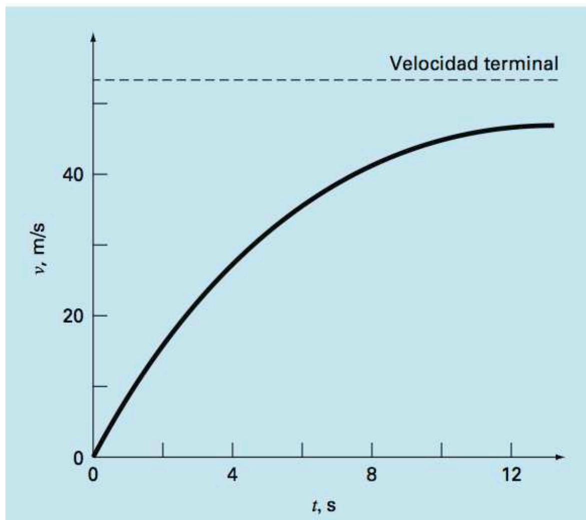


Figura 1.2: Solución analítica al problema del paracaidista.

A la ecuación (1.9) se le llama una solución analítica exacta. Si no es posible solucionar la ecuación se tiene que usar una solución numérica que se aproxime a la solución exacta, y emplearemos un método numérico.

La segunda ley de Newton en la aceleración se puede aproximar,

$$\frac{dv}{dt} \approx \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} \tag{1.10}$$

donde Δv y Δt son diferencias en la velocidad y el tiempo calculadas sobre intervalos

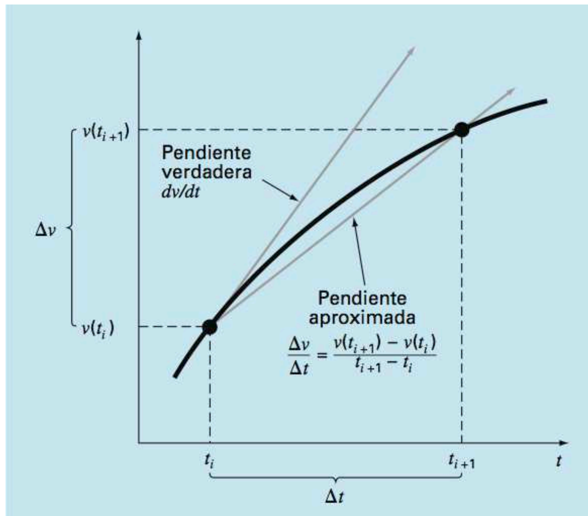


Figura 1.3: Uso de una diferencia finita para aproximar la primera derivada de V con respecto a t.

finitos, $v(t_i)$ es la velocidad en el tiempo inicial t_i, y $v(t_{i+1})$ es la velocidad en un tiempo más tarde t_{i+1}. La ecuación (1.10) es una diferencia finita dividida en el tiempo t_i. Al sustituir en la ecuación (1.8) se da,

$$\frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} = g - \frac{c}{m}v(t_i) \tag{1.11}$$

al ordenar esta ecuación dá,

$$v(t_{i+1}) = v(t_i) + [g - \frac{c}{m}v(t_i)](t_{i+1} - t_i) \tag{1.12}$$

(nuevo valor de v)= (valor anterior de v)+ (valor estimado de la pendendiente) x (incremento del tiempo)

Ejemplo 2: Solución numérica al problema del paracaidista que cae.

Al hacer lo mismo pero con un incremento de 2 s. En $t_1 = 0$ la $v(t_i)$=0 y se puede estimar $v(t_{i+1})$ en t_{i+1}=2 s.

$$i = 1 : t_1 = 0 \qquad y \qquad v(t_1) = 0$$

$$v(t_2) = v(t_1) + [980 - \frac{12500}{68100} v(t_1)(t_2 - t_1)$$

$$t_2 = 2s$$

$$v(t_2) = (980)(2 - 0) = 1960 \frac{cm}{s}$$

$$i = 2 :$$

$$v(t_3) = v(t_2) + [980 - \frac{12500}{65100} v(t_2)](t_3 - t_2)$$

$$t_3 = 4s$$

$$v(4) = 1960 + [980 - \frac{12500}{68100}(1960)](4 - 2)$$

$$v(4) = 3200.5 \frac{cm}{s}$$

$$i = 3 :$$

$$v(t_4) = V(t_3) + [980 - \frac{12500}{68100} v(t_3)](t_4 - t_3)$$

$$t_4 = 6s$$

$$v(6) = 3200.5 + [980 - \frac{12500}{68100} 3200.5](6 - 4)$$

$$v(6) = 3985.6 \frac{cm}{s}$$

$$v(8) = 4482.5$$

$$v(10) = 4796.9$$

$$v(12) = 4995.9$$

$$\infty = 5339.0$$

t(s)	**v(cm/s)**
0	0
2	1960.0
4	3200.5
6	3985.6
8	4482.5
10	4796.9

12	4995.9
∞	5339.0

1.2. Descripción de un Algoritmo

1.2.1. Diseño de algoritmos

Un algoritmo es una secuencia lógica de pasos necesarios para ejecutar una tarea específica tal como la solución de un problema. Los buenos algoritmos tienen ciertas características. Siempre deben terminar después de una cantidad finita de pasos y deben ser lo más general posible para tratar cualquier caso particular.

1.2.2. Composición de un programa

Después de confeccionar un algoritmo, el paso siguiente es expresarlo como una secuenca de declaraciones de programación llamado código. Lo realizaremos en FORTRAN.

Programa de computadora en $FORTRAN$ para el problema de la suma simple.

```
c    Version en FORTRAN
      A=25
      B=15
      C=A+B
      WRITE(6,1) C
1    FORMAT(' ',F10.3)
      STOP
      END
```

$\uparrow$ *Columna* 7

$\uparrow$ *Columna* 5

$\uparrow$ *Columna* 1

1.2.3. FORTRAN

Es la contrucción de fórmula translation (traducción de fórmulas), y se desarrolló en la decáda de 1950. Debido a que fue expresamente diseñado para cálculos, ha sido el lenguaje más usado en la ingeniería y la ciencia.

1.2.4. Rastreo y prueba

Después de escribir el código del programa, se debe probar para buscar los errores, a los que se les llama bugs. Al proceso de localizar y corregir los errores se les conoce como rastreo.

1.2.5. Documentación

Después del rastreo y probado del programa, éste se debe documentar. La documentación es la inclusión de comentarios que le permiten al usario implementar el programa más fácilmente.

1.2.6. Almacenamiento y mantenimiento

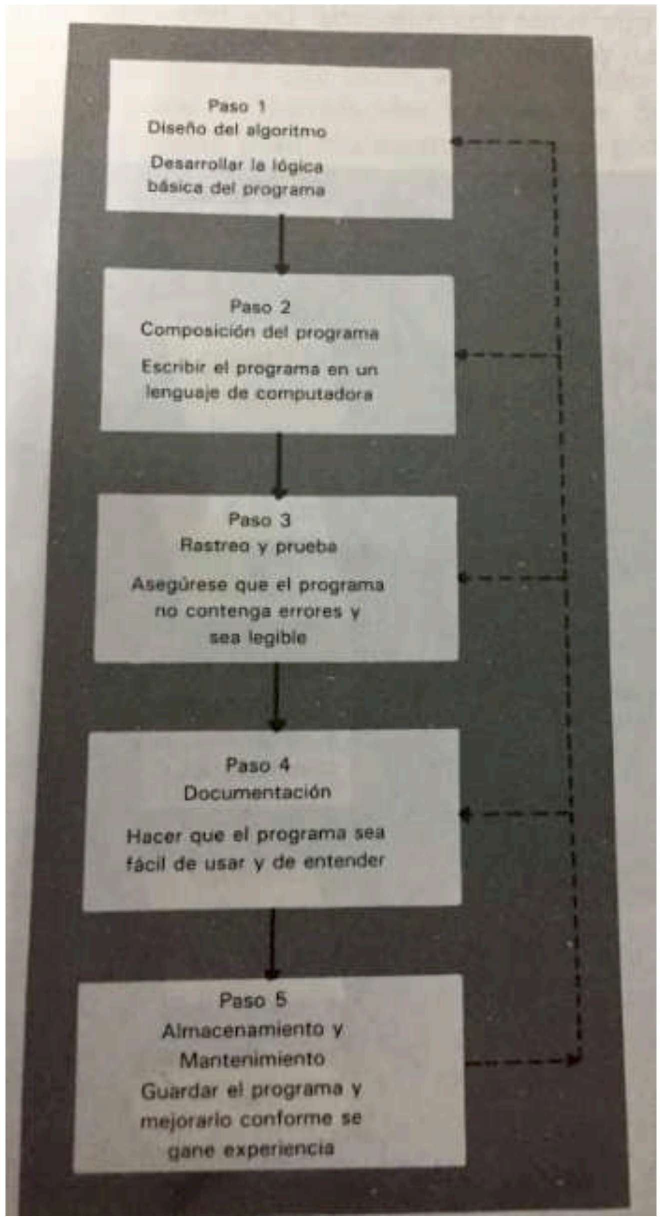
Los pasos finales del programa son el almacenamiento y mantenimiento del mismo. El mantenimiento involucra acondicionar el programa e incluso hacerle cambios que lo hagan accesible a problemas reales.

El almacenamiento se refiere a la manera en que los programas se guardan para uso posterior.

Compilación en fortran:

$gfortran \qquad -o \qquad nombre:del:ejecutable \qquad codigo:fuente.for$

Tabla 1.1: Cinco pasos necesarios para producir y dar soporte a programas de alta calidad.



Capítulo 2

Estudio general del error en un proceso numérico

2.0.1. Exactitud y precisión

Los errores asociados con los cálculos y medidas se pueden caracterizar observando su precisión y exactitud. La precisión se refiere a:

1. El número de cifras significativas en una cantidad. La extensión en las lecturas repetidas.
2. La exactitud se refiere a la aproximación de un número o de una medida al valor verdadero que se supone representa.

Los errores númericos se originan con el uso de aproximaciones. Estos incluyen errores de truncamiento y los errores de redondeo, el primer caso se generan por representar aproximadamente un procedimiento matemático exacto y el segundo por representar números exactos.

Valor verdadero = Valor aproximado + error

Error numérico igual a la diferencia del valor verdadero y el valor aproximado:

E_v= **valor verdadero - valor aproximado**

E_v**: valor exacto del error**

Una manera de medir las magnitudes de las cantidades que se están evaluando es normalizar el error respecto al valor verdadero, como en:

$$Error : relativo : funcional = \frac{error}{valor : verdadero}$$

El error relativo también se puede multiplicar por el 100 % para expresarlo como:

$$\epsilon_v = \frac{error : verdadero}{valor : verdadero} \times 100\,\%$$

donde ϵ_v denota el error relativo porcentual (real).

Normalizar el error es una alternativa, usando la mejor estimación posible del valor verdadero, esto es, a la aproximación misma, como:

$$\epsilon_a = \frac{error : aproximado}{valor : aproximado} \times 100\,\%$$

donde el subíndice a significa que el error está normalizado a un valor aproximado.

ϵ_a: error relativo porcentual aproximado.

Ciertos métodos numéricos usan un esquema iterativo para calcular resultados. Se hace una aproximación anterior, esto se repite varias veces, en tales casos, el error a menudo se calcula como la diferencia entre la aproximación previa y la actual. Por lo tanto, el error relativo porcentual está dado por:

$$\epsilon_a = \frac{aproximación : actual - aproximación : previa}{aproximación : actual} \times 100\,\%$$

a menudo no importa el signo del error, si no que su valor absoluto sea menor a una tolerancia prefijada ϵ_s.

$\mid \epsilon_a \mid < \epsilon_s$

ϵ_s=$(0.5\times10^{2-n})\,\%$

si se cumple este último criterio el resultado es correcto en al menos n cifras significativas.

Ejemplo: Estimación del error para métodos iterativos

A menudo se pueden representar las funciones mediante una serie infinita. La función exponencial se puede representar como,

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x3}{3} + \frac{x4}{4} + ...$$

(Expansión en series de Maclaurin)

Empezando con el primer término, $e^x = 1$, y agregando más términos, estímese el valor de $e^{0.5}$. Después que se agregue cada término, calcúlesen los errores relativos porcentuales real y aproximado. El valor real de $e^{0.5} = 1.648721271$. Agregar términos hasta que el valor absoluto del error aproximado e_a sea menor al criterio preestablecido, es que contempla tres cifras significativas.

Solución:

ϵ_s=(0.5×10$^{2-3}$) % = 0.005 %

(primer estimación)

$$e^x \approx 1$$

(segunda estimación)

$$e^x \approx 1 + x$$

para $x = 0.5$

$$e^{0.5} \approx 1 + 0.5 = 1.5$$

error relativo porcentual real,

$$\epsilon_v = \frac{1.648721271 - 1.5}{1.648721271} \times 100\,\% = 9.02\,\%$$

$$\epsilon_a = \frac{1.5 - 1}{1.5} \times 100\,\% = 33.3\,\%$$

ϵ_a no es menor al valor prefijado, ϵ_s; así que se continúa agregando más términos y calculando los errores ($\epsilon_a < \epsilon_s$).

(tercera estimación)

$$e^x \approx 1 + 0.5 + (\frac{0.51^2}{2} = 1.625$$

$$\epsilon_v = \frac{1.648721271 - 1.6525}{1.648721271} \times 100\,\% = 1.438\,\%$$

$$\epsilon_a = \frac{1.625 - 1.5}{1.625} \times 100\,\% = 7.69\,\%$$

2.1. Reglas de redondeo

1. En el redondeo, se conservan las cifras significativas y el resto se descarta (ver figura). El último dígito que se conserva se aumenta en uno si el primer dígito descartado es mayor de 5. De otra manera se deja igual. Si el primer dígito descartado es 5 o 5 seguido de ceros, entonces el último dígito retenido se incrementa en 1, sólo si es impar.

2. En la suma y en las restas, el redondeo se lleva a cabo de forma tal que el último dígito retenido en la respuesta corresponda al último dígito más significativo de los números que están sumando o restando. Nótese que un dígito en la columna de las centésimas es más significativo que uno de la columna de las milésimas.

3. Para la multiplicación y para la división el redondedo es tal que la cantidad de cifras significativas del resultado es igual al número más pequeño de cifras significativas que contiene la cantidad en la operación.

4. Para combinaciones de las operaciones aritméticas, existen dos casos generales. Se puede sumar o restar el resultado de las multiplicaciones o de las divisiones.

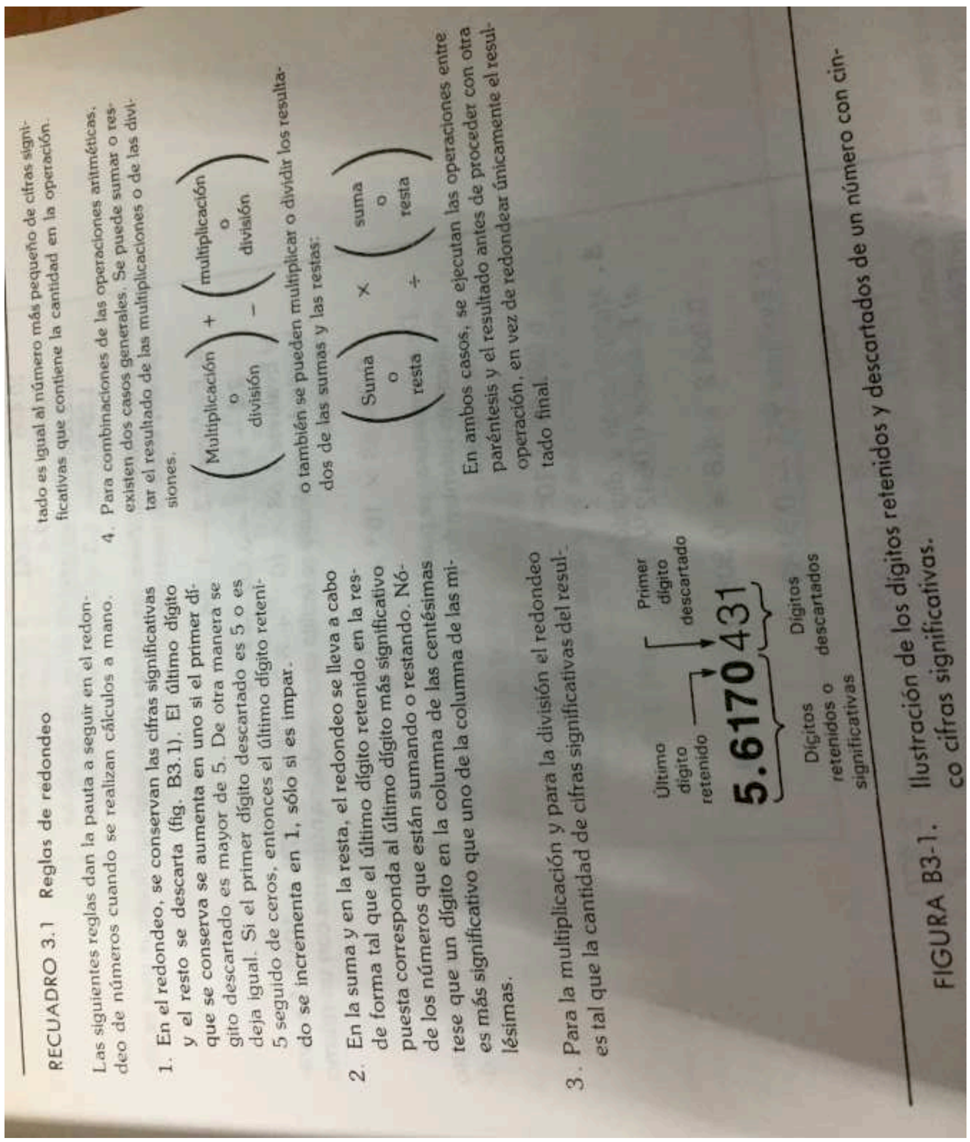
   (Multiplicación o división)$\pm$(Multiplicación o división)

   o también se pueden multiplicar o dividir los resultados de las sumas y las restas:

   (Suma o resta)$\times\div$(Suma o resta)

En ambos casos, se ejecutan las operaciones entre paréntesis y el resultado antes de proceder con otra operación, en vez de redondear únicamente el resultado final.

Tabla 2.1: Ilustración de los dígitos retenidos y descartados de un número con cinco cifras significativas.



RECUADRO 3.1 Reglas de redondeo

Las siguientes reglas dan la pauta a seguir en el redondeo de números cuando se realizan cálculos a mano.

1. En el redondeo, se conservan las cifras significativas y el resto se descarta (fig. B3.1). El último dígito que se conserva se aumenta en uno si el primer dígito descartado es mayor de 5. De otra manera se deja igual. Si el primer dígito descartado es 5 o es 5 seguido de ceros, entonces el último dígito retenido se incrementa en 1, sólo si es impar.
2. En la suma y en la resta, el redondeo se lleva a cabo de forma tal que el último dígito retenido en la respuesta corresponda al último dígito más significativo de los números que están sumando o restando. Nótese que un dígito en la columna de las centésimas es más significativo que uno de la columna de las milésimas.
3. Para la multiplicación y para la división el redondeo es tal que la cantidad de cifras significativas del resultado es igual al número más pequeño de cifras significativas que contiene la cantidad en la operación.
4. Para combinaciones de las operaciones aritméticas, existen dos casos generales. Se puede sumar o restar el resultado de las multiplicaciones o de las divisiones.

$$\begin{pmatrix}\text{Multiplicación}\\ \text{o}\\ \text{división}\end{pmatrix} \begin{matrix}+\\ -\end{matrix} \begin{pmatrix}\text{multiplicación}\\ \text{o}\\ \text{división}\end{pmatrix}$$

o también se pueden multiplicar o dividir los resultados de las sumas y las restas:

$$\begin{pmatrix}\text{Suma}\\ \text{o}\\ \text{resta}\end{pmatrix} \begin{matrix}\times\\ \div\end{matrix} \begin{pmatrix}\text{suma}\\ \text{o}\\ \text{resta}\end{pmatrix}$$

En ambos casos, se ejecutan las operaciones entre paréntesis y el resultado antes de proceder con otra operación, en vez de redondear únicamente el resultado final.

FIGURA B3-1. Ilustración de los dígitos retenidos y descartados de un número con cinco cifras significativas.

Ejemplo: Ilustraciones de las reglas de redondeo

1. Errores de redondeo

   $5.6723 \longrightarrow 5.67$ 3 cifras significativas

   $10.406 \longrightarrow 10.41$ 4 cifras significativas

   $7.3500 \longrightarrow 7.4$ 2 cifras significativas

   $88.21650 \longrightarrow 88.216$ 5 cifras significativas

   $1.25001 \longrightarrow 1.3$ 2 cifras significativas

2. Sumas y restas

   $2.2 - 1.768 = 0.432 \longrightarrow 0.4$

3. Multiplicación y división

   a) $0.0642 \times 4.8 = 0.30816 \longrightarrow 0.4$

   b) $\frac{945}{0.3185} = 2967.032967 \longrightarrow 2970$

4. Combinaciones

Capítulo 3

Resolución de ecuaciones no lineales

3.1. La serie de Taylor

La serie de Taylor da una formulación para predecir el valor de la función en x_{i+1} en términos de la función y de sus derivadas en una vecindad al punto x_i.
Se construirá término a término, el primer término de la serie es:

$$f(x_{i+1}) \approx f(x_i) \qquad (\textit{aproximación} \quad \textit{de} \quad \textit{orden} \quad \textit{cero}) \tag{3.1}$$

La ecuación (3.1) da una buena aproximación si la función que se va a aproximar es una constante. La aproximación a primer orden se obtiene sumando otro término al anterior para obtener:

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i) \tag{3.2}$$

$f'(x_i)$ es la pendiente multiplicada por la distancia entre x_{i+1} y x_i.

Agregando a la serie un término de 2^{do} orden para obtener algo sobre la curvatura de la función si es que la tiene:

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2 \tag{3.3}$$

agregando más términos para desarrollar la expansión completa de la serie de Taylor:

$$f(x_{i+1}) \approx f(x_i)+f'(x_i)(x_{i+1}-x_i)+\frac{f''(x_i)}{2!}(x_{i+1}-x_i)^2+\frac{f'''(x_i)}{3!}(x_{i+1}-x_i)^3+...+\frac{f^n(x_i)}{n!}(x_{i+1}-x_i)^n+R_n \tag{3.4}$$

se incluye un término residual para considerar todos los términos desde $(n+1)$ hasta el infinito:

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x_{i+1} - x_i)^{n+1} \tag{3.5}$$

donde el subíndice n indica que el residuo es de la aproximación a n-ésimo orden y ξ es un valor cualquiera de x que se encuentra en x_i y x_{i+1}.

R_n da una estimación exacta del error.

$$h = x_{i+1} - x_i$$

y la ecuación (3.4) toma la forma:

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f'''(x_i)}{3!}h^3 + ... + \frac{f^{(n)}(x_i)}{n!}h^n + R_n$$

en donde

$$R_n = \frac{f^{n+1}(\xi)}{(n+1)!}h^{n+1}$$

Ejemplo:

Usense términos en la serie de Taylor de cero a cuarto orden para aproximar la función:

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

desde el punto $x_i = 0$ y con $h = 1$. Esto es, predecir el valor de la función en $x_{i+1} = 1$.

3.2. El método de bisección

Si una función $f(x)$ es real y continua en el intervalo de x_l a x_u y $f(x_l)$ y $f(x_u)$ tienen signos opuestos, esto es,

$$f(x_l)f(x_u) < 0$$

entonces hay, al menos una raíz real entre x_l y x_u.

El método de bisección, conocido también como de corte binario, de partición en dos intervalos iguales o método de Bolzano, es un método de búsqueda incremental donde el intervalo se divide siempre en dos. Si la función cambia de signo sobre un intervalo, se evalúa el valor de la función en el punto medio. La posición de la raíz se determina sitúandola en el punto medio del subintervalo dentro del cual ocurre un cambio de signo. El proceso se repite hasta obtener una mejor aproximación.

Pasos:

1. Escójanse los valores iniciales x_l y x_u de forma tal que la función cambie de signo sobre el intervalo. Esto se puede verificar asegurándose de que $f(x_l)f(x_u) < 0$.

2. La primera aproximación a la raíz x_l se determina como: $x_r = \frac{x_l+x_u}{2}$

3. Realícense las siguientes evaluaciones y determínense en que subintervalo cae la raíz:
   a) Si $f(x_l)f(x_r) < 0$, entonces la raíz se encuentra dentro del primer subintervalo.

Por lo tanto, resuélvanse $x_u = x_r$ y continúese en el paso 4.

b) Si $f(x_l)f(x_r) > 0$, entonces la raíz se encuentra dentro del segundo subintervalo.

Por lo tanto, resuélvanse $x_l = x_r$ y continúese en el paso 4.

c) Si $f(x_l)f(x_r) = 0$, entonces la raíz es igual a x_r y se terminan los cálculos.

4. Calcúlese una nueva aproximación a la raíz mediante:

$$x_r = \frac{x_l + x_u}{2}$$

5. Decídase si la nueva aproximación es tan exacta como se desee. Si es así, entonces los cálculos terminan, de otra manera, regrésese al paso 3.

Ejemplo (bisección):

Usese el método de la bisección para determinar la raíz de $f(x) = e^{-x} - x$.

$$x_l = 0 \qquad y \qquad x_u = 1$$

$$x_r = \frac{x_l + x_u}{2} = \frac{0+1}{2} = \frac{1}{2} = 0.5$$

y la cuarta iteración es:

$$f(0.5)f(0.625) = -0.010 < 0$$

la raíz está entre 0.5 y 0.625:

$$x_u = 0.625$$

$$x_r = \frac{0.5 + 0.625}{2} = 0.5625 \quad y \quad \mid \epsilon_v \mid = 0.819\,\%$$

El método se puede repetir para obtener mejores estimaciones.

El error relativo aproximado ϵ_a

$$\mid \epsilon_a \mid = \mid \frac{x_r^{nueva} - x_r^{anterior}}{x_r^{nueva}} \mid \times 100\,\%$$

x_r^{nueva} : es la raíz de la iteración actual.

$x_r^{anterior}$: es el valor de la raíz de la iteración anterior.

cuando $\mid \epsilon_a \mid$ es menor que un valor previamente fijado, que define el criterio de paro, ϵ_s, el programa se detiene.

Programación del método de bisección (*biseccion.for*).

```
implicit real*8 (a-h,o-z)
F(X)=EXP(-X)-X
OPEN(5,FILE='biseccion.dat')
READ(5,*)XL,XU,ES,IM
1 FORMAT(3F10.0,I5)

AA=F(XL)*F(XU)

IF(AA.GE.0.0) GOTO 310
XR=(XL+XU)/2
DO 240 NI=2,IM

AA=F(XL)+f(XR)
IF (ABS(AA).EQ.0.01) GOTO 300
IF (AA.LT.0.0)XU=XR
IF (AA.GT.0.0)XL=XR
```

```
      XN=(XL+XU)/2
      IF (XN.EQ.0.0) GOTO 230
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LT.ES) GOTO 280
230 XR=XN
240 CONTINUE
      OPEN(6,FILE='biseccion.out')
      WRITE(6,2)
2 FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      WRITE(6,3)XR,EA
3 FORMAT(' ',2F10.5)
      GOTO 310
280 WRITE(6,4)XN,EA,NI
4 FORMAT(' ',2F10.5,I5)
      GOTO 310
300 WRITE(6,5)XR
5 FORMAT(' ','LA RAIZ EXACTA ES =',F10.5)
310 STOP
      END
```

3.3. Método de la regla falsa

Un defecto del método de bisección es que al dividir el intervalo x_l a x_u en mitades iguales, no se toma en consideración la magnitud de $f(x_l)$ y de $f(x_u)$. Por ejemplo, si $f(x_l)$ está mucho más cerca de cero que $f(x_u)$, es lógico que la raíz se encuentra más cerca de $f(x_l)$ que de $f(x_u)$. Este método alternativo aprovecha la idea de unir los puntos con una línea recta. La intersección de esta línea con el eje x proporciona una mejor estimación de la raíz. El reemplazamiento de la curva por una línea recta da una $posición\ falsa$ de la raíz, de aquí el nombre de método de la regla falsa o en latín, Regla falsi. Tambíen se le conoce como método de interpolación lineal.

Con el uso de triángulos semejantes, la intersección de la línea recta y el eje x se puede calcular de la siguiente manera:

$$\frac{f(x_l)}{x_r - x_l} = \frac{f(x_u)}{x_r - x_u}$$

Resolviéndose para x_r

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$

Está es la fórmula de la regla falsa.

Programación del método de la regla falsa ($rfalsa.for$).

```
implicit real*8 (a-h,o-z)
F(X)=EXP(-X)-X
OPEN(5,FILE='rfalsa.dat')
READ(5,*)XL,XU,ES,IM
1 FORMAT(3F10.0,I5)

AA=F(XL)*F(XU)

IF(AA.GE.0.0) GOTO 310
```

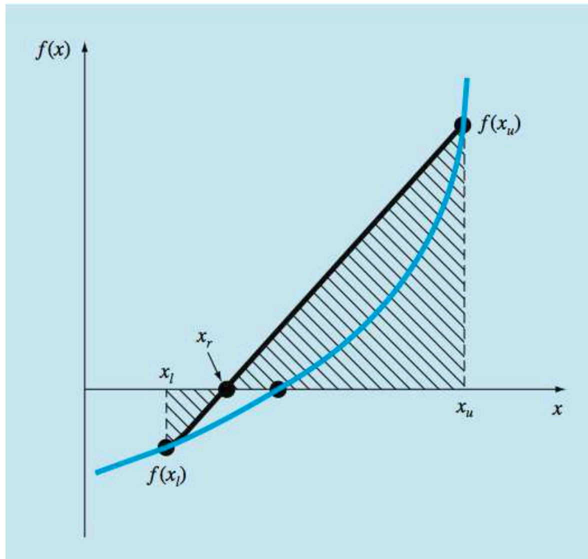


Figura 3.1: Esquema gráfico del método de la regla falsa. La fórmula se deriva de los triángulos semejantes (áreas sombreadas).

```
XR=(XL+XU)/2
      XR=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
      DO 240 NI=2,IM


      AA=F(XL)+F(XR)
      IF (ABS(AA).EQ.0.01) GOTO 300
      IF (AA.LT.0.0)XU=XR
      IF (AA.GT.0.0)XL=XR


      XN=(XL+XU)/2
      XN=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
```

```
    IF (XN.EQ.0.0) GOTO 230
    EA=ABS((XN-XR)/XN)*100
    IF (EA.LT.ES) GOTO 280
230 XR=XN
240 CONTINUE
    OPEN(6,FILE='rfalsa.out')
    WRITE(6,2)
2 FORMAT(' ','NO SE ENCONTRO LA RAIZ')
    WRITE(6,3)XR,EA
3 FORMAT(' ',2F10.5)
    GOTO 310
280 WRITE(6,4)XN,EA,NI
4 FORMAT(' ',2F10.5,I5)
    GOTO 310
300 WRITE(6,5)XR
5 FORMAT(' ','LA RAIZ EXACTA ES =',F10.5)
310 STOP
    END
```

Los métodos abiertos se basan en fórmulas que requieren de un solo valor de x o un par de ellos pero no necesariamente encierran la raíz, algunas veces divergen o se alejan de la raíz a medida que crece el número de iteraciones.

3.4. Iteración de punto fijo

Se emplea una fórmula que se puede desarrollar para la iteración de punto fijo o real usando la ecuación $f(x) = 0$ de tal forma que x quede del lado requerido de la ecuación:

$$x = g(x)$$

Dada una aproximación inicial de la raíz, x_i, se puede obtener una nueva aproximación x_{i+1}, expresada por la fórmula iterativa:

$$x_{i+1} = g(x_i)$$

Se puede estimar el error:

$$\mid \epsilon_a \mid = \mid \frac{x_{i+1} - x_i}{x_{i+1}} \mid \times 100\,\%$$

Ejemplo: Localizar la raíz de $f(x) = e^{-x} - x$.

Programación del método de Iteración de punto fijo (*iteracion.for*).

```
    implicit real*8(a-h,o-z)
    F(X)=EXP(-X)-X
    OPEN(5,FILE='iteracion.dat')
    READ(5,*) XR,ES,IM
    PRINT *, XR,ES,IM
1 FORMAT(2F10.0,I5)
    DO 180 NI=1,IM
    XN=F(XR)
```

```
    IF(XN.EQ.0.0)GOTO 170
    EA=ABS((XN-XR)/XN)*100
    IF (EA.LE.ES) GOTO 210
170 XR=XN
180 CONTINUE
    OPEN(6,FILE='iteracion.out')
    WRITE(6,2)
2 FORMAT(' ','NO SE ENCONTRO LA RAIZ')
    NI=NI-1
210 WRITE(6,3) XN,EA,NI
3 FORMAT(' ',2F15.5,I5)
    STOP
    END
```

3.5. Método de Newton-Raphson

La fórmula de Newton-Raphson es una de las más ampliamente usadas. Si el valor inicial de la raíz es x_i, entonces se puede extender una tangente desde el punto $[x_i, f(x_i)]$. El punto donde esta tangente cruza al eje x representa una aproximación mejorada a la raíz.
La primera derivada en x es equivalente a la pendiente:

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

ordenando se obtiene:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Conocida como la fórmula de Newton-Raphson.

3.5.1. Criterios de paro y estimación de errores

La derivación del método con la serie de Taylor proporciona un conocimiento teórico relacionado con la velocidad de convergencia expresado como:
$E_{i+1} = 0(E_i)^2$. De esta forma, el error debe ser casi proporcional al cuadrado del error anterior.

La serie de Taylor se puede representar como:

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(\xi)}{2}(x_{i+1} - x_i)^2$$

$\xi \ \epsilon \ [x_i, x_{i+1}]$

truncado a la primera derivada

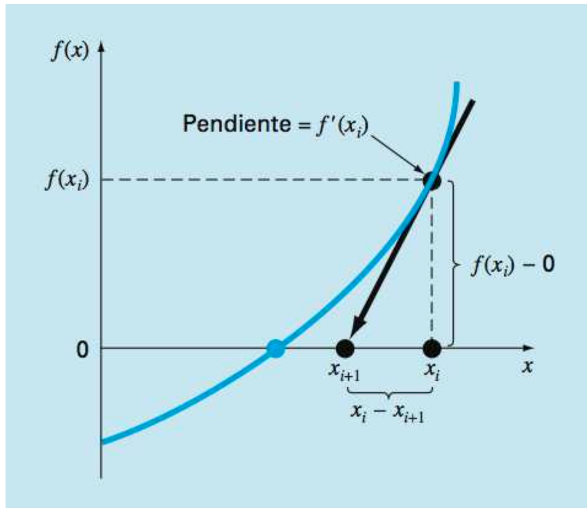


Figura 3.2: Esquema gráfico del método de Newton-Raphson.

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

en x intersectado

$f(x_{i+1}) = 0,\qquad$ y

$$0 \approx f(x_i) + f'(x_i)(x_{i+1} - x_i) \qquad (*)$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

idéntica a la ecuación de Newton-Raphson.

$x_{i+1} = x_r$ en donde x_r es el valor exacto de la raíz $f(x_r) = 0$ y

$$f(x_r) = f(x_i) + f'(x_i)(x_r - x_i) + \frac{f''(\xi)}{2}(x_\tau - x_i)^2 \qquad (**)$$

restando (*) y (**), se obtiene:

$$0 = f'(x_i)(x_r - x_{i+1}) + \frac{f''(\xi)}{2}(x_r - x_i)^2 \qquad (***)$$

El error es igual a la diferencia entre x_{i+1} y el valor real, x_r como en:

$$E_{v,i+1} = x_r - x_{i+1}$$

y la ecuación (***) se expresa como:

$$0 = f'(x_i)E_{v.i+1} + \frac{f''(\xi)}{2}E_{v,i}^2$$

y si hay convergencia, x_i y ξ se aproximan a la raíz x_r y podemos expresar:

$$E_{v,i+1} \approx -\frac{f''(x_r)}{2f'(x_r)}E_{v,i}^2$$

Programación del método Newton-Raphson (*nraphson.for*).

```
      implicit real*8(a-h,o-z)
      F(X)=EXP(-X)-X
      FD(X)=-EXP(-X)-1
      OPEN(5,FILE='nraphson.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
```

```
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170 XR=XN
180 CONTINUE
      OPEN(6,FILE='nraphson.out')
      WRITE(6,2)
2 FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210 WRITE(6,3) XN,EA,NI
3 FORMAT(' ',2F12.5,I5)
      STOP
      END
```

3.6. Método de la secante

La derivada se puede aproximar mediante una diferencia dividida, como

$$f'(x_i) \approx \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i}$$

Se puede sustituir en la ecuación de Newton-Raphson obteniendo la ecuación iterativa:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{\frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i}} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$

Fórmula para el método de la secante:

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$

No requiere que $f(x)$ cambie de signo entre estos valores, a este método no se le clasifica como aquellos que usan intervalos.

Ejemplo: Úsese el método de la secante para calcular la raíz de $f(x) = e^{-x} - x$. Empiécese con los valores iniciales de $x_{-1} = 0$ y $x_0 = 1.0$.

Programación del método de la secante (*secante.for*).

```
      implicit real*8(a-h,o-z)
      F(X)=EXP(-X)-X
      OPEN(5,FILE='secante.dat')
      READ(5,*) XR,XI,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(F(XR)*(XI-XR))/(F(XI)-F(XR))
      IF(XN.EQ.0.0)GOTO 170
```

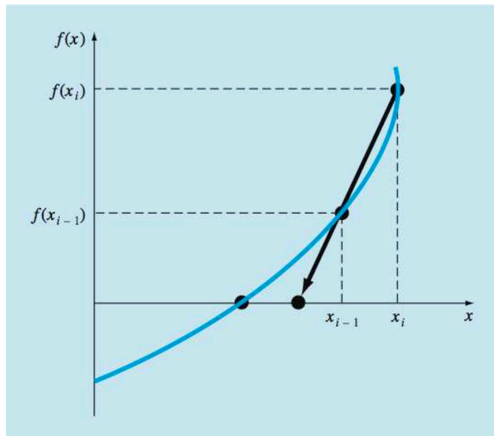


Figura 3.3: Esquema gráfico del método de la secante.

```
    EA=ABS((XN-XR)/XN)*100
    IF (EA.LE.ES) GOTO 210
170 XR=XN
180 CONTINUE
    OPEN(6,FILE='secante.out')
    WRITE(6,2)
2 FORMAT(' ','NO SE ENCONTRO LA RAIZ')
    NI=NI-1
210 WRITE(6,3) XN,F(XR),F(XI)
3 FORMAT(' ',3F12.5)
    STOP
    END
```

3.7. Raíces Múltiples

Una raíz multiple corresponde a un punto donde una función es tangencial al eje x. Por ejemplo, dos raíces repetidas resultan de:

$$f(x) = (x-3)(x-1)(x-1) \qquad (multiplicando \quad términos)$$

$$f(x) = x^3 - 5x^2 + 7x - 3 \qquad (esta \quad función \quad tiene \quad una \quad raíz \quad doble)$$

La raíces mútiples ofrecen ciertas dificultades a los métodos numéricos expuestos.

1. El hecho de que la función no cambia de signo en una raíz de multiplicidad par impide el uso de los métodos confiables que usan intervalos.

2. Otro posible problema se relaciona con el hecho de que no solo $f(x)$ se aproxima a cero. Estos problemas afectan los métodos de Newton-Raphson y al de la secante, los que contienen derivadas (o aproximaciones a ella) en el denominador de sus respectivas fórmulas.

3. Se puede demostrar que el método de Newton-Raphson y el de la secante convergen en forma lineal, en vez de manera cuadrática, cuando hay raíces múltiples. Se ha propuesto,

$$x_{i+1} = x_i - m\frac{f(x_i)}{f'(x_i)}$$

   en donde m es la multiplicidad de la raíz ($m = 2$ para una raíz doble, $m = 3$ para una raíz triple, etc...)

Otra alternativa, sugerida por Ralston y Rabinowitz (1978), es proponer una función

$$u(x) = \frac{f(x)}{f'(x)}$$

Sustituyendo esta función en la ecuación de Newton-Raphson:

$$x_{i+1} = x_i - \frac{u(x_i)}{u'(x_i)} \qquad\qquad (*)$$

derivando $u'(x)$:

$$u(x) = f(x)f'(x)^{-1}$$

$$u'(x) = f'(x)f'(x)^{-1} - f(x)f'(x)^{-2}f''(x) = \frac{f'(x)}{f'(x)} - \frac{f(x)f''(x)}{[f'(x)]^2}$$

$$u'(x) = \frac{f'(x)f'(x) - f(x)f''(x)}{[f'(x)]^2}$$

sustituyendo en (*) obtenemos:

$$x_{i+1} = x_i - \frac{\frac{f(x_i)}{f'(x_i)}}{\frac{f'(x_i)f'(x_i)-f(x_i)f''(x_i)}{[f'(x_i)]^2}}$$

Método de Newton-Raphson modificado,

$$x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i)]^2 - f(x_i)f''(x_i)}$$

Capítulo 4

Aproximación e interpolación

4.1. Mínimos cuadrados

Se ajustará una línea recta a un conjunto de datos observados $(x_1, y_1), ..., (x_n, y_n)$. La expresión matemática de una línea recta es,

$$y = a_0 + a_1 x + E$$

a_0: intersección con el eje de las abscisas.

a_1: es la pendiente.

E: es el error o residuo entre el modelo y las observaciones.

$$E = y - a_0 - a_1 x$$

y: valor real.

$(-a_0 - a_1 x)$: valor aproximado.

4.1.1. Críterios para un mejor ajuste

a) Minimizar la suma de los errores residuales,

$$(inadecuado) \qquad \sum_{i=1}^{n} E_i = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)$$

b) Minimizar la suma de los valores absolutos de las diferencias,

$$(inadecuado) \qquad \sum_{i=1}^{n} \mid E_i \mid = \sum_{i=1}^{n} \mid y_i - a_0 - a_1 x_i \mid$$

c) Minimizar la suma de los cuadrados de los residuos, S_r, de la siguiente manera:

$$S_r = \sum_{i=1}^{n} E_i^2 = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^2$$

Ajuste de una recta utilizando mínimos cuadrados

Para determinar las constantes a_0 y a_1 se deriva la ecuación S_r con respecto a sus coeficientes:

$$\frac{\partial S_r}{\partial a_o} = -2 \sum (y_i - a_0 - a_1 x_i)$$

$$\frac{\partial S_r}{\partial a_1} = -2\sum[(y_i - a_0 - a_1x_i)x_i]$$

Igualando estas derivadas a cero, se genera un mínimo S_r:

$$0 = \sum y_i - \sum a_0 - \sum a_1x_i$$

$$0 = \sum y_ix_i - \sum a_0x_i - \sum a_1x_i^2$$

$$\sum a_0 = na_0$$

Ecuaciones normales:

$$na_0 + \sum a_1x_i = \sum y_i$$

$$\sum a_0x_i + \sum x_i^2a_1 = \sum x_iy_i$$

Resolviendo para $a_0 \quad y \quad a_1$:

$$a_1 = \frac{n\sum x_iy_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

$$a_0 = \bar{y} - a_1\bar{x}$$

$\bar{x}$: medida de $x \quad (\sum \frac{x_i}{n})$

$\bar{y}$: medida de $y \quad (\sum \frac{y_i}{n})$

Una *desviación estándar* de la línea de regresión se puede determinar como

$$S_{\frac{y}{x}} = \sqrt{\frac{S_r}{n-2}}$$

en donde $S_{\frac{y}{x}}$ se llama error estándar de la aproximación.

$\frac{y}{x}$: indica que el error es para un valor predicho de y correspondiente a un valor particular de x.

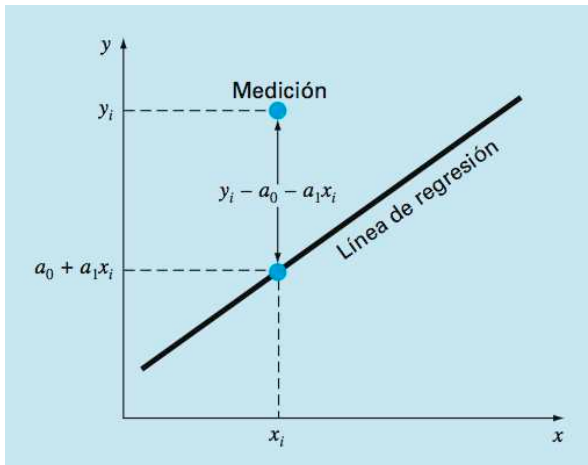


Figura 4.1: El residuo en la regresión lineal representa el cuadrado de la distancia vertical entre un punto y la línea recta.

Programa de mínimos cuadrados ajustando una línea recta (*mcuadrados.for*).

```
implicit real*8(a-h,o-z)
DATA SX/0./, SY/0./, X2/0./, XY/0./
OPEN(5,FILE="mcuadrados.dat")
READ(5,*) N
1 FORMAT(I5)
DO 170 I=1,N
READ(5,*) X,Y
2 FORMAT(2F10.0)
```

```
      SX=SX+X
      SY=SY+Y
      X2=X2+X*X
      XY=XY+X*Y
170 CONTINUE
      XM=SX/N
      YM=SY/N
      A1=(N*XY-SX*SY)/(N*X2-SX*SX)
      A0=YM-A1*XM
      OPEN(6,FILE="mcuadrados.out")
      WRITE(6,3) A0, A1
3 FORMAT(' ',2F10.7)
      STOP
      END
```

4.2. Interpolación

$f(x) = a_0 + a_{1x} + a_2x2 + ... + a_nx^n$

(polinomio de n-ésimo orden para $(n+1)$ puntos)

4.2.1. Interpolación lineal

La forma más simple de interpolación es la de conectar dos puntos con una línea recta, esto es la interpolación lineal.

$$\frac{f_1(x) - f(x_0)}{x - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

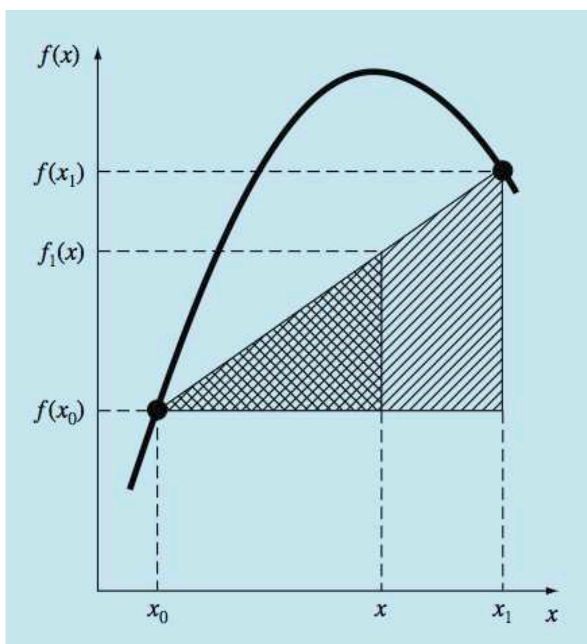


Figura 4.2: Las áreas sombreadas muestran triángulos semejantes.

Reordenando:

$$f_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) \quad \longrightarrow Interpolación \quad lineal$$

4.2.2. Polinomios de interpolación de Newton

Ajuste de un polinomio de n-ésimo orden a los $(n+1)$ puntos. El polinomio de n-ésimo orden es:

$$f_n(x) = b_0 + b_1(x - x_0) + ... + b_n(x - x_0)(x - x_1)...(x - x_{n-1})$$

se requiere encontrar los $b_0, ..., b_n$; $(n+1)$ puntos se requieren para obtener un polinomio de n-ésimo orden: $x_0, x_1, ..., x_n$. Usando estos datos, con las ecuaciones siguientes se evalúan los coeficientes:

$b_0 = f(x_0)$

$b_1 = f[x_1, x_0]$

$b_2 = f[x_2, x_1, x_0]$

.

.

.

$\text{b}_n = f[x_n, x_{n-1}, ..., x_1, x_0]$

las evaluaciones entre corchetes son diferencias divididas finitas. La primera diferencia dividida finita se represena como:

$$f[x_i, x_j] = \frac{f(x_i) - f(x_j)}{x_i - x_j}$$

La segunda diferencia dividida finita es,

$$f[x_i, x_j, x_k] = \frac{f[x_i, x_j] - f[x_j, x_k]}{x_i - x_k}$$

La n-ésima diferencia dividida finita es,

$$f[x_n, x_{n-1}, ..., x_1, x_0] = \frac{f[x_n, x_{n-1}, ..., x_1] - f[x_{n-1}, x_{n-2}, ..., x_0]}{x_n - x_0}$$

El polinomio de interpolación es,

$$f_n(x) = f(x_0) + (x - x_0)f[x_1, x_0] + (x - x_0)(x - x_1)f[x_2, x_1, x_0]$$

$$+... + (x - x_0)(x - x_1)...(x - x_{n-1})f[x_n, x_{n-1}, ..., x_0]$$

(**Polinomio de interpolación con diferencias divididas de Newton**)

Ejemplo: Úsese la siguiente información para evaluar $f(x) = ln(x)$ en $x = 2$.

x	**f(x)=ln(x)**
0	0
2	1960.0
4	3200.5
6	3985.6
8	4482.5
10	4796.9
12	4995.9
∞	5339.0

$Aproximación del error: R_n = F[x_{n+1}, x_n, x_{n-1}, ..., x_0](x - x_0)(x - x_1)...(x - x_n)$

4.2.3. Polinomios de interpolación de Lagrange

Este polinomio es una reformulación del polinomio de Newton que evita los cálculos de las diferencias divididas. Esto es,

$$f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$$

en donde

$$L_i(x) = \prod_{j=0, j=i}^{n} \frac{x - x_i}{x_i - x_j}$$

en donde Π denota el *producto de*. Por ejemplo, la versión lineal ($n = 1$) es:

$$f_1(x) = \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1)$$

y la versión de segundo orden es:

$$f(x_2) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1) + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2)$$

Error aproximado de Lagrange es:

$$R_n = f[x, x_n, x_{n-1}, ..., x_0] \prod_{i=0}^{n} (x - x_i)$$

4.3. Fórmulas de integración de Newton-Cotes

Las fórmulas de integración de Newton-Cotes son los esquemas más comunes dentro de la integración numérica. Se basan en la estrategia de reemplazar una función complicada o un conjunto de datos tabulares con una función aproximada que sea más fácil de integrar:

$$I = \int_a^b f(x)dx \approx \int_a^b f_n(x)dx \qquad (*)$$

en donde $f_n(x)$ es un polinomio de la forma:

$$f_n(x) = a_0 + a_1x + ... + a_{n-1}x^{n-1} + a_nx^n$$

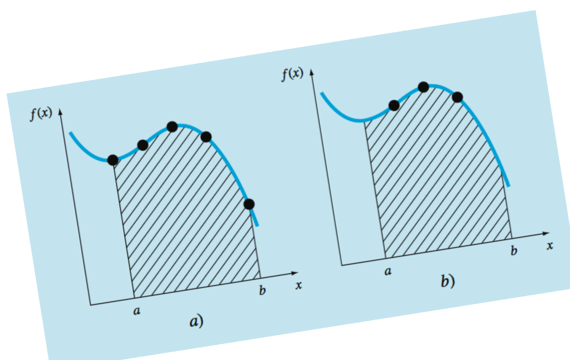


Figura 4.3: Diferencia entre fórmulas de integración.
a) **cerradas:** los puntos al principio y al final de los límites de integración.
b) **abiertas:** tienen los límites de integración extendidos más allá del rango de los datos.

4.4. Regla del trapecio

La regla del trapecio o regla trapezoidal es la primera de las fórmulas cerradas de Newton-Cotes. Corresponde al caso en donde el polinomio de la ecuación $(*)$ es de primer orden.

$$I = \int_a^b f(x)dx \approx \int_a^b f_1(x)dx$$

$$f_1(x) = f(a) + \frac{f(b) - f(a)}{b - a}(x - a)$$

El área bajo la línea recta es una aproximación de la integral de $f(x)$ entre los límites a y b:

$$I \approx \int_a^b [f(a) + \frac{f(b) - f(a)}{b - a}(x - a)]dx$$

El resultado de la integración es,

$$I \approx (b - a)\frac{f(a) + f(b)}{2}$$

al que se le llama regla trapezoidal.

Error de truncamiento en la regla trapezoidal:

$$E_t = -\frac{1}{12}f''(\xi)(b - a)^3$$

Programa de la regla trapezoidal (*trapezoidal.for*).

```
    implicit real*8(a-h,o-z)
    DIMENSION F(10), Y(20)
    REAL IN
    COMMON N,A,B
    READ (5,1) N
1 FORMAT(I5)
```

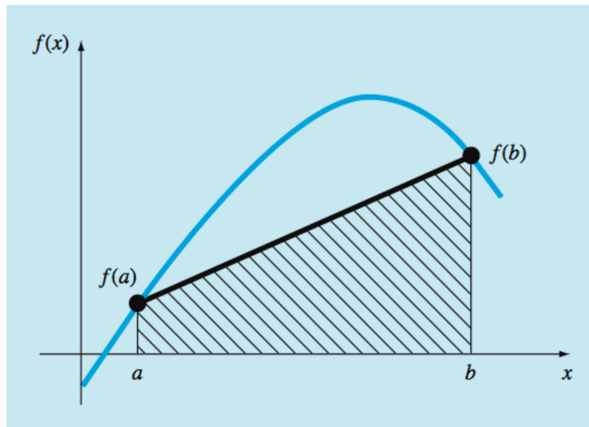


Figura 4.4: Esquema gráfico de la regla trapezoidal.

```
      NI=N-1
      READ(5,2) A,B
2 FORMAT(2F10.0)
      H=(B-A)/NI
      DO 170 I=1,N
      READ(5,3) Y(I)
3 FORMAT(F10.0)
170 CONTINUE
      CALL TRAP(Y,IN)
      WRITE(6,4)IN
4 FORMAT(' ',F10.3)
      STOP
```

```
      END

      SUBROUTINE TRAP(Y,IN)
      DIMENSION Y(20)
      REAL IN
      COMMON N,A,B
      NI=N-1
      SU=Y(1)
      DO 1030 I=2,NI
      SU=SU+2*Y(1)
 1030 CONTINUE
      HT=(SU+Y(N))/(2*NI)
      IN=(B-A)/HT
      RETURN
      END
```

4.5. Regla de Simpson

Si podemos conectar más de 3 o 4 puntos (una parábola o polinomio de 3^{er} orden). A éstas fórmulas se les conoce como reglas de Simpson.

4.5.1. Regla de Simpson de $\frac{1}{3}$

Esta fórmula resulta de sustituir un polinomio de 2^{do} orden en la ecuación $(*)$:

$$I \approx \int_a^b f(x)dx \approx \int_a^b f_2(x)dx$$

Si $a = x_0$ y $b = x_2$; $f_2(x)$ polinomio de Lagrange de segundo orden, la integral es:

$$I \approx \int_{x_0}^{x_2} [\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}f(x_0)+\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}f(x_1)+\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}f(x_2)]dx$$

Integrando,

$$I \approx \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)]$$

Regla de Simpson de $\frac{1}{3}$

donde $h = \frac{(b-a)}{2}$.

$$E_v = -\frac{1}{90}h^5 f^4(\xi) \qquad Error : de : truncamiento.$$

Programa de la regla de simpson.

Capítulo 5

Solución numérica de ecuaciones diferenciales ordinarias: problema a valores iniciales

Solución de ecuaciones diferenciales ordinarias de la forma,

$\frac{dy}{dx} = f(x,y)$ (problema del paracaidista)

Valor actual = valor anterior + pendiente x tamaño del paso

o en términos matemáticos:

$$y_{i+1} = y_i + \phi h \qquad (**)$$

5.1. La serie de Taylor

La serie de Taylor da una formulación para predecir el valor de la función en x_{i+1} en términos de la función y de sus derivadas en una vecindad al punto x_i.

Se construirá término a término, el primer término de la serie es:

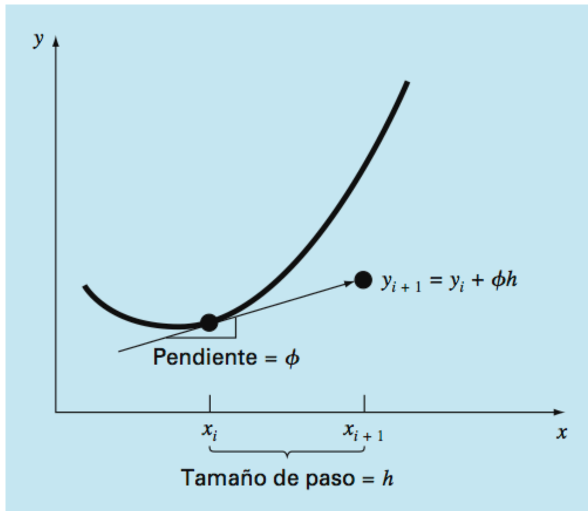


Figura 5.1: Esquema gráfico del método de un paso.

$$f(x_{i+1}) \approx f(x_i) \qquad (aproximación \quad de \quad orden \quad cero) \tag{5.1}$$

La ecuación (5.1) da una buena aproximación si la función que se va a aproximar es una constante. La aproximación a primer orden se obtiene sumando otro término al anterior para obtener:

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i) \tag{5.2}$$

$f'(x_i)$ es la pendiente multiplicada por la distancia entre x_{i+1} y x_i.

Agregando a la serie un término de 2^{do} orden para obtener algo sobre la curvatura de la función si es que la tiene:

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2 \tag{5.3}$$

agregando más términos para desarrollar la expansión completa de la serie de Taylor:

$$f(x_{i+1}) \approx f(x_i)+f'(x_i)(x_{i+1}-x_i)+\frac{f''(x_i)}{2!}(x_{i+1}-x_i)^2+\frac{f'''(x_i)}{3!}(x_{i+1}-x_i)^3+...+\frac{f^n(x_i)}{n!}(x_{i+1}-x_i)^n+R_n \tag{5.4}$$

se incluye un término residual para considerar todos los términos desde $(n+1)$ hasta el infinito:

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x_{i+1} - x_i)^{n+1} \tag{5.5}$$

donde el subíndice n indica que el residuo es de la aproximación a n-ésimo orden y ξ es un valor cualquiera de x que se encuentra en x_i y x_{i+1}.

R_n da una estimación exacta del error.

$$h = x_{i+1} - x_i$$

y la ecuación (5.4) toma la forma:

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f'''(x_i)}{3!}h^3 + ... + \frac{f^{(n)}(x_i)}{n!}h^n + R_n$$

en donde

$$R_n = \frac{f^{n+1}(\xi)}{(n+1)!}h^{n+1}$$

Ejemplo:

Usense términos en la serie de Taylor de cero a cuarto orden para aproximar la función:

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

desde el punto $x_i = 0$ y con $h = 1$. Esto es, predecir el valor de la función en $x_{i+1} = 1$.

5.2. Método de Euler

La primera derivada proporciona una aproximación directa a la pendiente en x_i:

$$0 = f(x_i, y_i)$$

donde $f(x_i, y_i)$ es la ecuación diferencial evaluada en x_i y y_i. Esta aproximación se sustituye en la ecuación $(**)$:

$$y_{i+1} = y_i + f(x_i, y_i)h$$

Método de Euler (o método de Euler-Cauchy o de pendiente puntual).

$$R_n = \frac{y^{(n+1)}(\xi)}{(n+1)!}h^{n+1} \qquad ”término \quad residual”$$

Ejemplo: Utilícese el método de Euler para integrar numéricamente la ecuación

$$f(x, y) = -2x^3 + 12x^2 - 20x + 8.5$$

de $x = 0$ hasta $x = 4$ con un tamaño de paso de 0.5. La condición inicial en $x = 0$ es $y = 1$. La solución exacta esta dada por la ecuación:

$$y = -0.5x^4 + 4x^3 - 10x^2 + 8.5x + 1$$

Solución: Implementando el método de Euler:

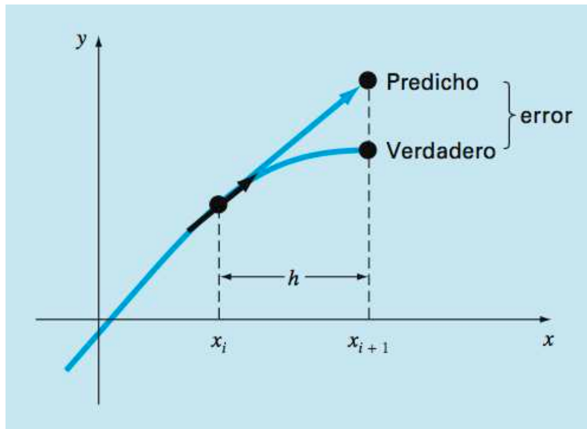


Figura 5.2: Método de Euler (o método de Euler-Cauchy o de pendiente puntual).

$$y(0.5) = y(0) + f(0,1)0.5$$

$$y(0) = 1$$

y la aproximación a la pendiente en $x = 0$ es:

$$f(0,1) = -2(0)^3 + 12(0)^2 - 20(0) + 8.5 = 8.5$$

Por lo tanto:

$$y(0.5) = 1.0 + 8.5(0.5) = 5.25$$

La solución verdadera en $x = 0.5$ es:

$$y(0.5) = -0.5(0.5)^4 + 4(0.5)^3 - 10(0.5)^2 + 8.5(0.5) + 1 = 3.21875$$

El error es:

$$E_v\text{= verdadero - aproximado= } 3.21875 - 5.25 = -2.03125$$

o, como error relativo porcentual, $\epsilon_v = -63.1\,\%$

En el segundo paso:

$$y(1.0) = y(0.5) + f(0.5, 5.25)0.5$$

$$= 5.25 + [-2(0.5)^3 + 12(0.5)^2 - 20(0.5) + 8.5]0.5 = 5.875$$

Programa del método de Euler (*meuler.for*).

```
      IMPLICIT REAL*8(A-H,O-Z)
      F(X,Y)=4*EXP(0.8*X)-0.5*Y
      OPEN(5,FILE='meuler.dat')
      READ(5,*) X0,X1
1     FORMAT(2F10.0)
      READ(5,*) Y0
2     FORMAT(F10.0)
      READ(5,*) H
      READ(5,*) PI
      NP=(X1-X0)/PI
      NC=PI/H
      X=X0
```

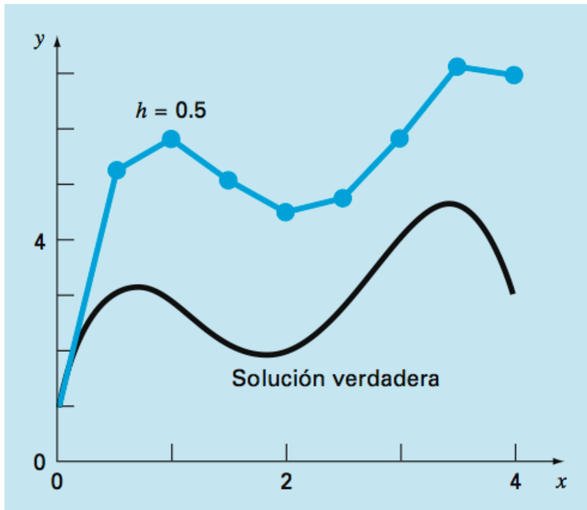


Figura 5.3: Solución del método de Euler.

```
    Y=Y0
    OPEN(6,FILE='meuler.out')
    WRITE(6,3) X,Y
3 FORMAT(' ',2F20.3)
    DO 270 I=1,NP
    DO 250 J=1,NC
    CALL EUL(SL)
    Y=Y+SL*H
    X=X+H
250 CONTINUE
    WRITE(6,3) X,Y
270 CONTINUE
```

```
STOP
END

SUBROUTINE EUL(SL)
COMMON X,Y
F(X,Y)=4*EXP(0.8*X)-0.5*Y
SL=F(X,Y)
RETURN
END
```

5.3. Método de Runge-Kutta

Los métodos de Runge-Kutta tienen la exactitud del esquema de la serie de Taylor sin necesitar del cálculo de derivadas superiores. Existen muchas variaciones pero todas ellas se pueden ajustar a la forma general de la ecuación,

$$y_{i+1} = y_i + \phi(x_i, y_i, h)h$$

donde

$\phi(x_i, y_i, h)$: función de incremento y puede interpretarse como el promedio de la pendiente sobre el intervalo.

ϕ se puede escribir como,

$\phi = a_1k_1 + a_2k_2 + ... + a_nk_n$

$a's$: son constantes.

$k's$: se pueden escribir como,

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + p_1h, y_i + q_{11}k_1h)$$

$$k_3 = f(x_i + p_2h, y_i + q_{21}k_1h + q_{22}k_2h)$$

.

.

.

$$k_n = f(x_i + p_{n-1}h, y_i + q_{n-1,1}k_1h + q_{n-1,2}k_2h + ... + q_{n-1,n-1}k_{n-1}h)$$

Para $n = 2$ se tiene el método de Runge-kutta de segundo orden.

5.3.1. Método de Runge-Kutta de segundo orden

La versión de segundo orden es ($n = 2$),

$$y_{i+1} = y_i + \phi(x_i, y_i, h)h$$

$$\phi = a_1k_1 + a_2k_2$$

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + p_1h, y_i + q_{11}k_1h)$$

Recordando que la serie de Taylor de segundo orden para y_i y $f(x_i, y_i)$ se escribe como,

$$y_{i+1} = y_i + f(x_i, y_i)h + f'(x_i, y_i)\frac{h^2}{2}$$

$$f'(x_i, y_i) = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx}$$

$$y_{i+1} = y_i + f(x_i, y_i)h + (\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx})\frac{h^2}{2}$$

$$y_{i+1} = y_i + [a_1k_1 + a_2k_2]h$$

expandiendo

$$k_2 = f(x_i + p_i h, y_i + q_{11} k_1 h)$$

en la serie de Taylor a segundo orden se obtiene,

$$f(x_i + p_1 h, y_i + q_{11} k_1 h) = f(x_i, y_i) + p_1 h \frac{\partial f}{\partial x} + q_{11} k_1 h \frac{\partial f}{\partial y} + 0(h^2)$$

sustituyendo este resultado y el valor de $k_1 = f(x_i, y_i)$ en

$$y_{i+1} = y_i + [a_1 k_1 + a_2 k_2] h \qquad se \quad tiene,$$

$$y_{i+1} = y_i + a_1 f(x_i, y_i) h + a_2 h [f(x_i, y_i) + p_1 h \frac{\partial f}{\partial x} + q_{11} k_1 h \frac{\partial f}{\partial y}]$$

$$y_{i+1} = y_i + [a_1 f(x_i, y_i) + a_2 f(x_i, y_i)] h + [a_2 p_1 \frac{\partial f}{\partial x} + a_2 q_{11} f(x_i, y_i) \frac{\partial f}{\partial y}] h^2 + 0(h^3)$$

Comparando los términos con la serie de Taylor,

$$a_1 + a_2 = 1$$

$$a_2 p_1 = \frac{1}{2}$$

$$a_2 q_{11} = \frac{1}{2}$$

Por lo que tenemos 4 incóngnitas y 3 ecuaciones, se tiene que tener el valor de una incógnita para determinar el valor de las otras tres.

Supóngase que conocemos a_2:

$$a_1 = 1 - a_2$$

$$p_1 = \frac{1}{2a_2}$$

$$q_{11} = \frac{1}{2a_2}$$

5.4. Eliminación Gaussiana

Programa de eliminación Gaussiana ($egaussiana.for$).

c Eliminacion de Gauss

dimension a(5,6), b(5,6), y(5), x(5)

y(1) = 0.468

y(2) = 0.695

y(3) = 0.398

y(4) = 0.913

y(5) = 0.483

a(1,1) = 0.546

a(1,2) = 0.447

a(1,3) = 0.242

a(1,4) = 0.194

a(1,5) = 0.795

a(2,1) = 0.380

a(2,2) = 0.276

a(2,3) = 0.581

a(2,4) = 0.108

a(2,5) = 0.416

a(3,1) = 0.721

a(3,2) = 0.022

a(3,3) = 0.853

a(3,4) = 0.068

a(3,5) = 0.312

a(4,1) = 0.151

a(4,2) = 0.759

a(4,3) = 0.186

a(4,4) = 0.597

a(4,5) = 0.757

a(5,1) = 0.192

a(5,2) = 0.509

a(5,3) = 0.041

a(5,4) = 0.411

a(5,5) = 0.632

a(1,6) = y(1)

a(2,6) = y(2)

a(3,6) = y(3)

a(4,6) = y(4)

a(5,6) = y(5)

open(55,file='egauss.out',status='unknown')

call gauss(a, 5, 6, 5)

call m$_display(a,5,6)$

stop

end

```
      subroutine gauss(a,n,m,ldim)
      dimension t(1000),a(ldim,1)
      do 10 i=1,n-1
      con=1./a(i,i)
      do 5 k=1,m+1-i
5     t(k)=-con*a(i,k+i-1)
      call elim (a(i,i),t,n-i,m+1-i,ldim,con)
10    continue
      return
      end

      subroutine elim(a,t,nn,m,ldim,con1)
      dimension a(ldim,1),t(1)
      do 4 i=2,nn+1
      con=a(i,1)
      do 3 j=1,m
3     a(i,j)=a(i,j)+con*t(j)
      !a(i,1)=con1*con
4     continue
      return
      end
```

subroutine m$_d isplay(a, l_r ow, l_c ol)$

dimension a(l$_r ow, l_c ol)$

$doi = 1, l_r ow$

$print 10, (a(i, j), j = 1, l_c ol)$

$write(55, 10)(a(i, j), j = 1, l_c ol)$

10 $format(6F10.3)$

$enddo$

print *,

return

end

Capítulo 6

Ejercicios resueltos de los diferentes métodos usados en el curso de Métodos Numéricos

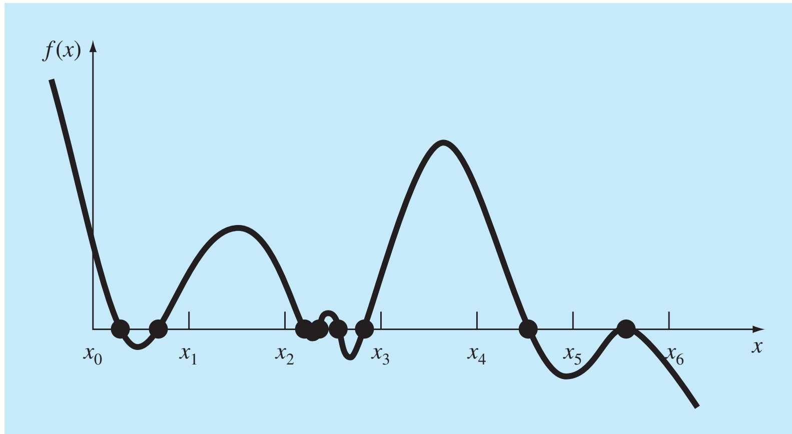


FIGURA 5.16
Casos donde las raíces pueden pasar inadvertidas debido a que la longitud del incremento en el método de búsqueda incremental es demasiado grande. Observe que la última raíz a la derecha es múltiple y podría dejar de considerarse independientemente de la longitud del incremento.

Un problema potencial en los métodos de búsqueda por incremento es el de escoger la longitud del incremento. Si la longitud es muy pequeña, la búsqueda llega a consumir demasiado tiempo. Por otro lado, si la longitud es demasiado grande, existe la posibilidad de que raíces muy cercanas entre sí pasen inadvertidas (figura 5.16). El problema se complica con la posible existencia de raíces múltiples. Un remedio parcial para estos casos consiste en calcular la primera derivada de la función $f'(x)$ al inicio y al final de cada intervalo. Cuando la derivada cambia de signo, puede existir un máximo o un mínimo en ese intervalo, lo que sugiere una búsqueda más minuciosa para detectar la posibilidad de una raíz.

Aunque estas modificaciones o el empleo de un incremento muy fino ayudan a resolver el problema, se debe aclarar que métodos tales como el de la búsqueda incremental no siempre resultan sencillos. Será prudente complementar dichas técnicas automáticas con cualquier otra información que dé idea de la localización de las raíces. Esta información se puede encontrar graficando la función y entendiendo el problema físico de donde proviene la ecuación.

PROBLEMAS

5.1 Determine las raíces reales de $f(x) = -0.5x^2 + 2.5x + 4.5$:

a) Gráficamente
b) Empleando la fórmula cuadrática
c) Usando el método de bisección con tres iteraciones para determinar la raíz más grande. Emplee como valores iniciales $x_l = 5$ y $x_u = 10$. Calcule el error estimado ε_a y el error verdadero ε_t para cada iteración.

5.2 Determine las raíces reales de $f(x) = 5x^3 - 5x^2 + 6x - 2$:

a) Gráficamente
b) Utilizando el método de bisección para localizar la raíz más pequeña. Use los valores iniciales $x_l = 0$ y $x_u = 1$ iterando hasta que el error estimado ε_a se encuentre debajo de $\varepsilon_s = 10\%$.

5.3 Determine las raíces reales de $f(x) = -25\ 1\ 82x - 90x^2 + 44x^3 - 8x^4 + 0.7x^5$:

a) Gráficamente
b) Usando el método de bisección para localizar la raíz más grande con $\varepsilon_s = 10\%$. Utilice como valores iniciales $x_l = 0.5$ y $x_u = 1.0$.
c) Realice el mismo cálculo que en *b*), pero con el método de la falsa posición y $\varepsilon_s = 0.2\%$.

5.4 Calcule las raíces reales de $f(x) = -12 - 21x + 18x^2 - 2.75x^3$:

a) Gráficamente
b) Empleando el método de la falsa posición con un valor ε_s correspondiente a tres cifras significativas para determinar la raíz más pequeña.

5.5 Localice la primera raíz no trivial de sen $x = x^2$, donde x está en radianes. Use una técnica gráfica y bisección con un intervalo inicial de 0.5 a 1. Haga el cálculo hasta que ε_a sea menor que $\varepsilon_s = 2\%$. Realice también una prueba de error sustituyendo la respuesta final en la ecuación original.
5.6 Determine la raíz real de $\ln x^2 = 0.7$:

a) Gráficamente
b) Empleando tres iteraciones en el método de bisección con los valores iniciales $x_l = 0.5$ y $x_u = 2$.
c) Usando tres iteraciones del método de la falsa posición, con los mismos valores iniciales de *b*).

5.7 Determine la raíz real de $f(x) = (0.8 - 0.3x)/x$:

a) Analíticamente
b) Gráficamente
c) Empleando tres iteraciones en el método de la falsa posición, con valores iniciales de 1 a 3. Calcule el error aproximado ε_a y el error verdadero ε_t en cada iteración.

5.8 Calcule la raíz cuadrada positiva de 18 usando el método de la falsa posición con $\varepsilon_s = 0.5\%$. Emplee como valores iniciales $x_l = 4$ y $x_u = 5$.
5.9 Encuentre la raíz positiva más pequeña de la función (x está en radianes) $x^2|\cos\sqrt{x}| = 5$ usando el método de la falsa posición. Para localizar el intervalo en donde se encuentra la raíz, grafique primero esta función para valores de x entre 0 y 5. Realice el cálculo hasta que ε_a sea menor que $\varepsilon_s = 1\%$. Compruebe su respuesta final sustituyéndola en la función original.
5.10 Encuentre la raíz positiva de $f(x) = x^4 - 8x^3 - 35x^2 + 450x - 1001$, utilizando el método de la falsa posición. Tome como valores iniciales a $x_l = 4.5$ y $x_u = 6$, y ejecute cinco iteraciones. Calcule los errores tanto aproximado como verdadero, con base en el hecho de que la raíz es 5.60979. Emplee una gráfica para explicar sus resultados y hacer el cálculo dentro de un $\varepsilon_s = 1.0\%$.
5.11 Determine la raíz real de $x^{3.5} = 80$:

a) En forma analítica.
b) Con el método de la falsa posición dentro de $\varepsilon_s = 2.5\%$. Haga elecciones iniciales de 2.0 a 5.0.

5.12 Dada

$$f(x) = -2x^6 - 1.5x^4 + 10x + 2$$

Use el método de la bisección para determinar el *máximo* de esta función. Haga elecciones iniciales de $x_l = 0$ y $x_u = 1$, y realice iteraciones hasta que el error relativo aproximado sea menor que 5%.
5.13 La velocidad v de un paracaidista que cae está dada por

$$v = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right)$$

donde g = 9.8 m/s^2. Para un paracaidista con coeficiente de arrastre de c = 15 kg/s, calcule la masa m de modo que la velocidad sea v = 35 m/s en t = 9s. Utilice el método de la falsa posición para determinar m a un nivel de $\varepsilon_s = 0.1\%$.
5.14 Se carga una viga de la manera que se aprecia en la figura P5.14. Emplee el método de bisección para resolver la posición dentro de la viga donde no hay momento.

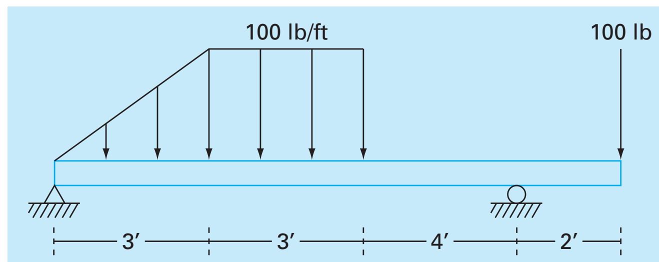


Figura P5.14

5.15 Por un canal trapezoidal fluye agua a una tasa de Q = 20 m^3/s. La profundidad crítica y para dicho canal satisface la ecuación

$$0 = 1 - \frac{Q^2}{gA_c^3}B$$

donde g = 9.81m/s^2, A_c = área de la sección transversal (m^2), y B = ancho del canal en la superficie (m). Para este caso, el ancho y el área de la sección transversal se relacionan con la profundidad y por medio de

$$B = 3 + y \qquad \text{y} \qquad A_c = 3y + \frac{y^2}{2}$$

Resuelva para la profundidad crítica con el uso de los métodos *a*) gráfico, *b*) bisección, y *c*) falsa posición. En los incisos *b*) y *c*), haga elecciones iniciales de $x_l = 0.5$ y $x_u = 2.5$, y ejecute iteraciones hasta que el error aproximado caiga por debajo del 1% o el número de interaciones supere a 10. Analice sus resultados.
5.16 Suponga el lector que está diseñando un tanque esférico (véase la figura P5.16) para almacenar agua para un poblado pequeño en un país en desarrollo. El volumen de líquido que puede contener se calcula con

$$V = \pi h^2 \frac{[3R - h]}{3}$$

donde V = volumen [m^3], h = profundidad del agua en el tanque [m], y R = radio del tanque [m].

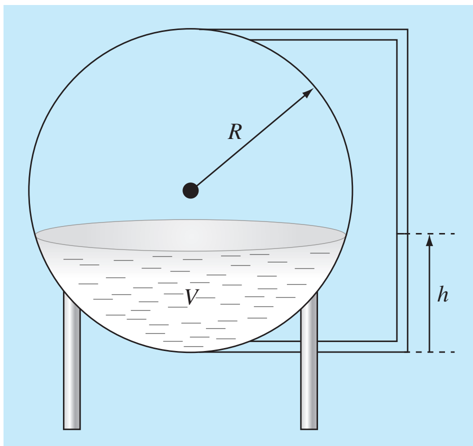


Figura P5.16

Si R = 3m, ¿a qué profundidad debe llenarse el tanque de modo que contenga 30 m^3? Haga tres iteraciones con el método de la falsa posición a fin de obtener la respuesta. Determine el error relativo aproximado después de cada iteración.

5.17 La concentración de saturación de oxígeno disuelto en agua dulce se calcula con la ecuación (APHA, 1992)

$$\ln o_{sf} = -139.34411 + \frac{1.575701\times 10^5}{T_a} - \frac{6.642308\times 10^7}{T_a^2} + \frac{1.243800\times 10^{10}}{T_a^3} - \frac{8.621949\times 10^{11}}{T_a^4}$$

donde o_{sf} = concentración de saturación de oxígeno disuelto en agua dulce a 1 atm (mg/L) y T_a = temperatura absoluta (K). Recuerde el lector que $T_a = T + 273.15$, donde T = temperatura (ºC). De acuerdo con esta ecuación, la saturación disminuye con el incremento de la temperatura. Para aguas naturales comunes en climas templados, la ecuación se usa para determinar que la concentración de oxígeno varía de 14.621 mg/L a 0ºC a 6.413 mg/L a 40ºC. Dado un valor de concentración de oxígeno, puede emplearse esta fórmula y el método de bisección para resolver para la termperatura en ºC.

a) Si los valores iniciales son de 0 y 40ºC, con el método de la bisección, ¿cuántas iteraciones se requerirían para determinar la temperatura con un error absoluto de 0.05ºC.
b) Desarrolle y pruebe un programa para el método de bisección a fin de determinar T como función de una concentración dada de oxígeno, con un error absoluto preespecificado como en el inciso *a*). Dadas elecciones iniciales de 0 y 40ºC, pruebe su programa para un error absoluto de 0.05ºC para los casos siguientes: o_{sf} = 8, 10 y 12 mg/L. Compruebe sus resultados.

5.18 Integre el algoritmo que se bosquejó en la figura 5.10, en forma de subprograma completo para el método de bisección amigable para el usuario. Entre otras cosas:

a) Construya enunciados de documentación en el subprograma a fin de identificar lo que se pretende que realice cada sección.
b) Etiquete la entrada y la salida.
c) Agregue una comprobación de la respuesta, en la que se sustituya la estimación de la raíz en la función original para verificar si el resultado final se acerca a cero.
d) Pruebe el subprograma por medio de repetir los cálculos de los ejemplos 5.3 y 5.4.

5.19 Desarrolle un subprograma para el método de bisección que minimice las evaluaciones de la función, con base en el seudocódigo que se presenta en la figura 5.11. Determine el número de evaluaciones de la función (n) para el total de iteraciones. Pruebe el programa con la repetición del ejemplo 5.6.

5.20 Desarrolle un programa amigable para el usuario para el método de la falsa posición. La estructura del programa debe ser similar al algoritmo de la bisección que se bosquejó en la figura 5.10. Pruebe el programa con la repetición del ejemplo 5.5.

5.21 Desarrolle un subprograma para el método de la falsa posición que minimice las evaluaciones de la función en forma similar a la figura 5.11. Determine el número de evaluaciones de la función (n) para el total de iteraciones. Pruebe el programa por medio de la duplicación del ejemplo 5.6.

5.22 Desarrolle un subprograma amigable para el usuario para el método de la falsa posición modificado, con base en la figura 5.15. Pruebe el programa con la determinación de la raíz de la función del ejemplo 5.6. Ejecute corridas hasta que el error relativo porcentual verdadero esté por debajo de 0.01%. Elabore una gráfica en papel semilogarítmico de los errores relativo, porcentual, aproximado y verdadero, *versus* el número de iteraciones. Interprete los resultados.

verdadera. Mientras que para el caso de una sola ecuación los métodos gráficos son útiles para obtener un buen valor inicial, ningún procedimiento tan simple está disponible para el caso de múltiples ecuaciones. Aunque existen algunos métodos avanzados para obtener una primer aproximación aceptable, los valores iniciales a menudo deben obtenerse mediante prueba y error, con el conocimiento del sistema físico que se está modelando.

El método de Newton-Raphson para dos ecuaciones puede generalizarse para resolver n ecuaciones simultáneas. Debido a que el camino más eficiente para esto implica el álgebra matricial y la solución de ecuaciones lineales simultáneas, se pospondrá su estudio para la parte tres.

PROBLEMAS

6.1 Utilice la iteración simple de punto fijo para localizar la raíz de

$$f(x) = 2 \text{ sen}\left(\sqrt{x}\right) - x$$

Haga una elección inicial de $x_0 = 0.5$ e itere hasta que $\varepsilon_a \leq 0.001\%$. Compruebe que el proceso converge en forma lineal según se describió en el recuadro 6.1.

6.2 Determine la raíz real más grande de

$$f(x) = 2x^3 - 11.7x^2 + 17.7x - 5$$

a) En forma gráfica.
b) Con el método de iteración simple de punto fijo (tres iteraciones, $x_0 = 3$). Nota: asegúrese de haber desarrollado una solución que converja a la raíz.
c) Con el método de Newton-Raphson (tres iteraciones, $x_0 = 3$, $\delta = 0.001$).
d) Con el método de la secante (tres iteraciones $x_{-1} = 3$, $x_0 = 4$).
e) Con el método de la secante modificado (tres iteraciones, $x_0 = 3$, $\delta = 0.01$). Calcule el porcentaje aproximado de errores relativos para sus soluciones.

6.3 Utilice los métodos de *a*) iteración de punto fijo, y *b*) Newton-Raphson, para determinar una raíz de $f(x) = -x^2 + 1.8x + 2.5$ con el uso de $x_0 = 5$. Haga el cálculo hasta que ε_a sea menor que $\varepsilon_s = 0.05\%$. Asimismo, realice una comprobación del error de su respuesta final.

6.4 Determine las raíces reales de $f(x) = -1 + 5.5x - 4x^2 + 0.5x^3$: *a*) en forma gráfica, y *b*) con el método de Newton-Raphson dentro de $\varepsilon_s = 0.01\%$.

6.5 Emplee el método de Newton-Raphson para determinar una raíz real de $f(x) = -1 + 5.5x - 4x^2 + 0.5x^3$ con el uso de elecciones iniciales de *a*) 4.52, y *b*) 4.54. Estudie y use métodos gráficos y analíticos para explicar cualquier peculiaridad en sus resultados.

6.6 Determine la raíz real más pequeña de $f(x) = -12 - 21x + 18x^2 - 2.4x^3$: *a*) en forma gráfica, y *b*) con el empleo del método de la secante para un valor de ε_s que corresponda a tres cifras significativas.

6.7 Localice la primera raíz positiva de

$$f(x) = \text{sen } x + \cos(1 + x^2) - 1$$

donde x está en radianes. Para localizar la raíz, use cuatro iteraciones del método de la secante con valores iniciales de *a*) $x_{i-1} = 1.0$ y $x_i = 3.0$; y *b*) $x_{i-1} = 1.5$ y $x_i = 2.5$, y *c*) $x_{i-1} = 1.5$ y $x_i = 2.25$.

6.8 Determine la raíz real de $x^{3.5} = 80$, con el método de la secante modificado dentro de $\varepsilon_s = 0.1\%$, con el uso de una elección inicial de $x_0 = 3.5$ y $\delta = 0.01$.

6.9 Determine la raíz real más grande de $f(x) = 0.95x^3 - 5.9x^2 + 10.9x - 6$:

a) En forma gráfica.
b) Con el uso del método de Newton-Raphson (tres iteraciones, $x_i = 3.5$).
c) Con el método de la secante (tres iteraciones, $x_{i-1} = 2.5$ y $x_i = 3.5$).
d) Por medio del método de la secante modificado (tres iteraciones, $x_i = 3.5$, $\delta = 0.01$).

6.10 Determine la menor raíz positiva de $f(x) = 8 \text{ sen}(x)e^{-x} - 1$:

a) En forma gráfica.
b) Con el uso del método de Newton-Raphson (tres iteraciones, $x_i = 0.3$).

c) Con el método de la secante (tres iteraciones, $x_{i-1} = 0.5$ y $x_i = 0.3$).

d) Por medio del método de la secante modificado (cinco iteraciones $x_i = 0.3$, $\delta = 0.01$).

6.11 La función $x^3 + 2x^2 - 4x + 8$ tiene una raíz doble en $x = 2$. Emplee *a*) el método estándar de Newton-Raphson [ec. (6.6)], *b*) el método de Newton-Raphson modificado [ec. (6.9*a*)], y *c*) el método de Newton-Raphson modificado [ec. (6.13)] para resolver para la raíz en $x = 2$. Compare y analice la tasa de convergencia con un valor inicial $x_0 = 1.2$.

6.12 Determine las raíces de las siguientes ecuaciones no lineales simultáneas, por medio de los métodos de *a*) iteración de punto fijo, y *b*) Newton-Raphson:

$$y = -x^2 + x + 0.75$$

$$y + 5xy = x^2$$

Utilice valores iniciales de $x = y = 1.2$, y analice los resultados.

6.13 Encuentre las raíces de las ecuaciones simultáneas que siguen:

$$(x - 4)^2 + (y - 4)^2 = 5$$

$$x^2 + y^2 = 16$$

Use un enfoque gráfico para obtener los valores iniciales. Encuentre estimaciones refinadas con el método de Newton-Raphson para dos ecuaciones, que se describe en la sección 6.5.2.

6.14 Repita el problema 6.13, excepto que

$$y = x^2 + 1$$

$$y = 2 \cos x$$

6.15 El balance de masa de un contaminante en un lago bien mezclado se expresa así:

$$V\frac{dc}{dt} = W - Qc - kV\sqrt{c}$$

Dados los valores de parámetros $V = 1 \times 10^6$ m^3, $Q = 1 \times 10^5$ m^3/año y $W = 1 \times 10^6$ g/año, y $k = 0.25$ m$^{0.5}$/año, use el método de la secante modificado para resolver para la concentración de estado estable. Emplee un valor inicial $c = 4$ g/m^3 y $\delta = 0.5$. Realice tres iteraciones y determine el error relativo porcentual después de la tercera iteración.

6.16 Para el problema 6.15, la raíz puede localizarse con iteración de punto fijo como

$$c = \left(\frac{W - Qc}{kV}\right)^2$$

o bien como

$$c = \frac{W - kV\sqrt{c}}{Q}$$

De las que solo una convergerá para valores iniciales de $2 < c < 6$. Seleccione la que sea correcta y demuestre por qué siempre lo será.

6.17 Desarrolle un programa amigable para el usuario para el método de Newton-Raphson, con base en la figura 6.4 y la sección 6.2.3. Pruébelo por medio de repetir el cálculo del ejemplo 6.3.

6.18 Desarrolle un programa amigable para el usuario para el método de la secante, con base en la figura 6.4 y la sección 6.3.2. Pruébelo con la repetición de los cálculos del ejemplo 6.6.

6.19 Haga un programa amigable para el usuario para el método de la secante modificado, con base en la figura 6.4 y la sección 6.3.2. Pruébelo con la repetición del cálculo del ejemplo 6.8.

6.20 Desarrolle un programa amigable para el usuario para el método de Newton-Raphson para dos ecuaciones, con base en la sección 6.5. Pruébelo con la solución del ejemplo 6.10.

6.21 Use el programa que desarrolló en el problema 6.20 para resolver los problemas 6.12 y 6.13, con una tolerancia de $\varepsilon_s = 0.01\%$.

6.22 El antiguo método de *dividir y promediar*, para obtener una apoximación de la raíz cuadrada de cualquier número positivo, *a*, se formula del modo siguiente:

$$x = \frac{x + a/x}{2}$$

Demuestre que éste es equivalente al algoritmo de Newton-Raphson.

6.23 *a*) Aplique el método de Newton-Raphson a la función $f(x) = \tanh(x^2 - 9)$ para evaluar su raíz real conocida en $x = 3$. Use un valor inicial de $x_0 = 3.2$ y haga un mínimo de cuatro iteraciones. *b*) ¿Converge el método a su raíz real? Bosqueja la gráfica con los resultados para cada iteración que obtenga.

6.24 El polinomio $f(x) = 0.0074x^4 - 0.284x^3 + 3.355x^2 - 12.183x + 5$ tiene una raíz real entre 15 y 20. Aplique el método de Newton-Raphson a dicha función con valor inicial $x_0 = 16.15$. Explique sus resultados.

6.25 Emplee el método de la secante con la función del círculo $(x + 1)^2 + (y - 2)^2 = 16$, a fin de encontrar una raíz real positiva. Haga que el valor inicial sea $x_i = 3$ y $x_{i-1} = 0.5$. Aproxímese a la solución del primer y cuarto cuadrantes. Cuando resuelva para

$f(x)$ en el cuarto cuadrante, asegúrese de tomar el valor negativo de la raíz cuadrada. ¿Por qué diverge la solución?

6.26 Suponga el lector que está diseñando un tanque esférico (véase la figura P6.26) de almacenamiento de agua para un poblado pequeño de un país en desarrollo. El volumen del líquido que puede contener se calcula con

$$V = \pi h^2 \frac{[3R - h]}{3}$$

donde V = volumen [pie^3], h = profundidad del agua en el tanque [pies], y R = radio del tanque [pies].

Si $R = 3$ m, ¿a qué profundidad debe llenarse el tanque de modo que contenga 30 m^3? Haga tres iteraciones del método de Newton-Raphson para determinar la respuesta. Encuentre el error relativo aproximado después de cada iteración. Observe que el valor inicial de R convergerá siempre.

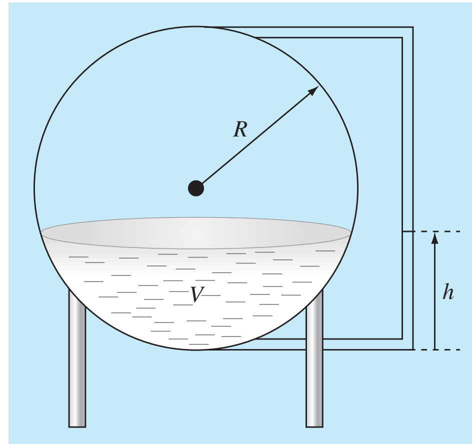


Figura P6.26

```
c       version fortran
        implicit real*8(a-h,o-z)

        A= -0.5
        B=2.5
        C=4.5
        R=(B*B)-(4*A*C)
        if(R .le. 0) then
        write(*,*)"elrecultado es real"
        else
        D=(sqrt(R))

   35   X1=(-B+D)/(2*A)
        X2=(-B-D)/(2*A)
   40   print*,X1
        print*,X2
        end if
        stop
        end
```

```
      implicit real*8 (a-h,o-z)
      f(x)=-0.5*x**2+2.5*x+4.5
      open(5,file='programa1c.dat')
      read(5,1)XL,XU,ES,IM
    1 format(3f10.0,I5)
      AA=f(XL)*f(XU)

      if(AA.GE.0.0) GOTO 310
      XR=(XL+XU)/2
      DO 240 NI=2,IM

      AA=f(XL)+f(XR)
      if(AA.EQ.0.0) GOTO 300
      if(AA.LT.0.0)XU=XR
      if(AA.Gt.0.0) XL=XR

      XN=(XL+XU)/2
      if(XN.EQ.0.0) GOTO 230
      EA=ABS((XN-XR)/XN)*100
      if(EA.LT.ES) GOTO 280
  230 XR=XN
  240 continue
      open(6,FILE='programa1c.out')
      write(6,2)
    2 format(' ','no se encontro la raiz')
      write(6,3)XR,EA
    3 format(' ',2f10.3)
      GOTO 310
  280 write(6,4)XN,EA,NI
    4 format(' ',2f10.3,I5)
      GOTO 310
  300 write(6,5)XR
    5 format(' ','la raiz exacta es=',f10.3)
  310 stop
      end
```

```
      implicit real*8 (a-h,o-z)
      f(x)=5*x**3-5*x**2+6*x-2
      open(5,file='programa2b.dat')
      read(5,*)XL,XU,ES,IM
    1 format(3f10.0,I5)
      AA=f(XL)*f(XU)

      if(AA.GE.0.0) GOTO 310
      XR=(XL+XU)/2
      DO 240 NI=2,IM

      AA=f(XL)+f(XR)
      if(AA.EQ.0.0) GOTO 300
      if(AA.LT.0.0)XU=XR
      if(AA.Gt.0.0) XL=XR

      XN=(XL+XU)/2
      if(XN.EQ.0.0) GOTO 230
      EA=ABS((XN-XR)/XN)*100
      if(EA.LT.ES) GOTO 280
  230 XR=XN
  240 continue
      open(6,FILE='programa2b.out')
      write(6,2)
    2 format(' ','no se encontro la raiz')
      write(6,3)XR,EA
    3 format(' ',2f10.3)
      GOTO 310
  280 write(6,4)XN,EA,NI
    4 format(' ',2f10.3,I5)
      GOTO 310
  300 write(6,5)XR
    5 format(' ','la raiz exacta es=',f10.3)
  310 stop
      end
```

problema5.3b

```
      implicit real*8 (a-h,o-z)
      f(x)=-25+82*x-90*x**2+44*x**3
      open(5,file='programa3b.dat')
      read(5,*)XL,XU,ES,IM
   1  format(3f10.0,I5)
      AA=f(XL)*f(XU)

      if(AA.GE.0.0) GOTO 310
      XR=(XL+XU)/2
      DO 240 NI=2,IM

      AA=f(XL)+f(XR)
      if(AA.EQ.0.0) GOTO 300
      if(AA.LT.0.0)XU=XR
      if(AA.Gt.0.0) XL=XR

      XN=(XL+XU)/2
      if(XN.EQ.0.0) GOTO 230
      EA=ABS((XN-XR)/XN)*100
      if(EA.LT.ES) GOTO 280
 230  XR=XN
 240  continue
      open(6,FILE='programa3b.out')
      write(6,2)
   2  format(' ','no se encontro la raiz')
      write(6,3)XR,EA
   3  format(' ',2f10.3)
      GOTO 310
 280  write(6,4)XN,EA,NI
   4  format(' ',2f10.3,I5)
      GOTO 310
 300  write(6,5)XR
   5  format(' ','la raiz exacta es=',f10.3)
 310  stop
      end
```

```
      implicit real*8 (a-h,o-z)
      F(X)=-25+82*x-90*x**2+44*x**3
      OPEN(5,FILE='programa3c.dat')
      READ(5,*)XL,XU,ES,IM
 1    FORMAT(3F10.0,I5)

      AA=F(XL)*F(XU)

      IF(AA.GE.0.0) GOTO 310
c     XR=(XL+XU)/2
      XR=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
      DO 240 NI=2,IM

      AA=F(XL)+F(XR)
      IF (ABS(AA).EQ.0.01) GOTO 300
      IF (AA.LT.0.0)XU=XR
      IF (AA.GT.0.0)XL=XR

C     XN=(XL+XU)/2
      XN=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
      IF (XN.EQ.0.0) GOTO 230
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LT.ES) GOTO 280
 230  XR=XN
 240  CONTINUE
      OPEN(6,FILE='programa3c.out')
      WRITE(6,2)
 2    FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      WRITE(6,3)XR,EA
 3    FORMAT(' ',2F10.5)
      GOTO 310
 280  WRITE(6,4)XN,EA,NI
 4    FORMAT(' ',2F10.5,I5)
      GOTO 310
 300  WRITE(6,5)XR
 5    FORMAT(' ','LA RAIZ EXACTA ES =',F10.5)
 310  STOP
      END
```

```
      implicit real*8 (a-h,o-z)
      F(X)=-12-21*x+18*x**2-2.75*x**3
      OPEN(5,FILE='programa4b.dat')
      READ(5,*)XL,XU,ES,IM
 1    FORMAT(3F10.0,I5)

      AA=F(XL)*F(XU)

      IF(AA.GE.0.0) GOTO 310
c     XR=(XL+XU)/2
      XR=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
      DO 240 NI=2,IM

      AA=F(XL)+F(XR)
      IF (ABS(AA).EQ.0.01) GOTO 300
      IF (AA.LT.0.0)XU=XR
      IF (AA.GT.0.0)XL=XR

C     XN=(XL+XU)/2
      XN=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
      IF (XN.EQ.0.0) GOTO 230
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LT.ES) GOTO 280
 230  XR=XN
 240  CONTINUE
      OPEN(6,FILE='programa4b.out')
      WRITE(6,2)
 2    FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      WRITE(6,3)XR,EA
 3    FORMAT(' ',2F10.5)
      GOTO 310
 280  WRITE(6,4)XN,EA,NI
 4    FORMAT(' ',2F10.5,I5)
      GOTO 310
 300  WRITE(6,5)XR
 5    FORMAT(' ','LA RAIZ EXACTA ES =',F10.5)
 310  STOP
      END
```

```fortran
      implicit real*8 (a-h,o-z)
      f(x)=sin(x)-x**2
      open(5,file='programa5.dat')
      read(5,*)XL,XU,ES,IM
 1    format(3f10.0,I5)
      AA=f(XL)*f(XU)

      if(AA.GE.0.0) GOTO 310
      XR=(XL+XU)/2
      DO 240 NI=2,IM

      AA=f(XL)+f(XR)
      if(AA.EQ.0.0) GOTO 300
      if(AA.LT.0.0)XU=XR
      if(AA.Gt.0.0) XL=XR

      XN=(XL+XU)/2
      if(XN.EQ.0.0) GOTO 230
      EA=ABS((XN-XR)/XN)*100
      if(EA.LT.ES) GOTO 280
230   XR=XN
240   continue
      open(6,FILE='programa5.out')
      write(6,2)
 2    format(' ','no se encontro la raiz')
      write(6,3)XR,EA
 3    format(' ',2f10.3)
      GOTO 310
280   write(6,4)XN,EA,NI
 4    format(' ',2f10.3,I5)
      GOTO 310
300   write(6,5)XR
 5    format(' ','la raiz exacta es=',f10.3)
310   stop
      end
```

```
      implicit real*8 (a-h,o-z)
      f(x)=log(x**2) -0.7
      open(5,file='programa6b.dat')
      read(5,*)XL,XU,ES,IM
    1 format(3f10.0,I5)
      AA=f(XL)*f(XU)

      if(AA.GE.0.0) GOTO 310
      XR=(XL+XU)/2
      DO 240 NI=2,IM

      AA=f(XL)+f(XR)
      if(AA.EQ.0.0) GOTO 300
      if(AA.LT.0.0)XU=XR
      if(AA.Gt.0.0) XL=XR

      XN=(XL+XU)/2
      if(XN.EQ.0.0) GOTO 230
      EA=ABS((XN-XR)/XN)*100
      if(EA.LT.ES) GOTO 280
  230 XR=XN
  240 continue
      open(6,FILE='programa6b.out')
      write(6,2)
    2 format(' ','no se encontro la raiz')
      write(6,3)XR,EA
    3 format(' ',2f10.3)
      GOTO 310
  280 write(6,4)XN,EA,NI
    4 format(' ',2f10.3,I5)
      GOTO 310
  300 write(6,5)XR
    5 format(' ','la raiz exacta es=',f10.3)
  310 stop
      end
```

```
      implicit real*8 (a-h,o-z)
      F(X)=log(x**2) -0.7
      OPEN(5,FILE='programa6c.dat')
      READ(5,*)XL,XU,ES,IM
  1   FORMAT(3F10.0,I5)

      AA=F(XL)*F(XU)

      IF(AA.GE.0.0) GOTO 310
c     XR=(XL+XU)/2
      XR=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
      DO 240 NI=2,IM

      AA=F(XL)+F(XR)
      IF (ABS(AA).EQ.0.01) GOTO 300
      IF (AA.LT.0.0)XU=XR
      IF (AA.GT.0.0)XL=XR

C     XN=(XL+XU)/2
      XN=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
      IF (XN.EQ.0.0) GOTO 230
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LT.ES) GOTO 280
  230 XR=XN
  240 CONTINUE
      OPEN(6,FILE='programa6c.out')
      WRITE(6,2)
  2   FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      WRITE(6,3)XR,EA
  3   FORMAT(' ',2F10.5)
      GOTO 310
  280 WRITE(6,4)XN,EA,NI
  4   FORMAT(' ',2F10.5,I5)
      GOTO 310
  300 WRITE(6,5)XR
  5   FORMAT(' ','LA RAIZ EXACTA ES =',F10.5)
  310 STOP
      END
```

```
      implicit real*8 (a-h,o-z)
      F(X)=(0.8-0.3*x)/x
      OPEN(5,FILE='programa7c.dat')
      READ(5,*)XL,XU,ES,IM
  1   FORMAT(3F10.0,I5)

      AA=F(XL)*F(XU)

      IF(AA.GE.0.0) GOTO 310
c     XR=(XL+XU)/2
      XR=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
      DO 240 NI=2,IM

      AA=F(XL)+F(XR)
      IF (ABS(AA).EQ.0.01) GOTO 300
      IF (AA.LT.0.0)XU=XR
      IF (AA.GT.0.0)XL=XR

C     XN=(XL+XU)/2
      XN=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
      IF (XN.EQ.0.0) GOTO 230
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LT.ES) GOTO 280
  230 XR=XN
  240 CONTINUE
      OPEN(6,FILE='programa7c.out')
      WRITE(6,2)
  2   FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      WRITE(6,3)XR,EA
  3   FORMAT(' ',2F10.5)
      GOTO 310
  280 WRITE(6,4)XN,EA,NI
  4   FORMAT(' ',2F10.5,I5)
      GOTO 310
  300 WRITE(6,5)XR
  5   FORMAT(' ','LA RAIZ EXACTA ES =',F10.5)
  310 STOP
      END
```

```
      implicit real*8 (a-h,o-z)
      F(X)=X**4-8*X**3-35*X**2+450*X-1001
      OPEN(5,FILE='programa8.dat')
      READ(5,*)XL,XU,ES,IM
  1   FORMAT(3F10.0,I5)

      AA=F(XL)*F(XU)

      IF(AA.GE.0.0) GOTO 310
c     XR=(XL+XU)/2
      XR=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
      DO 240 NI=2,IM

      AA=F(XL)+F(XR)
      IF (ABS(AA).EQ.0.01) GOTO 300
      IF (AA.LT.0.0)XU=XR
      IF (AA.GT.0.0)XL=XR

C     XN=(XL+XU)/2
      XN=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
      IF (XN.EQ.0.0) GOTO 230
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LT.ES) GOTO 280
  230 XR=XN
  240 CONTINUE
      OPEN(6,FILE='programa8.out')
      WRITE(6,2)
  2   FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      WRITE(6,3)XR,EA
  3   FORMAT(' ',2F10.5)
      GOTO 310
  280 WRITE(6,4)XN,EA,NI
  4   FORMAT(' ',2F10.5,I5)
      GOTO 310
  300 WRITE(6,5)XR
  5   FORMAT(' ','LA RAIZ EXACTA ES =',F10.5)
  310 STOP
      END
```

```
      implicit real*8 (a-h,o-z)
      F(X)=X**3.5 -80
      OPEN(5,FILE='programa9.dat')
      READ(5,*)XL,XU,ES,IM
  1   FORMAT(3F10.0,I5)

      AA=F(XL)*F(XU)

      IF(AA.GE.0.0) GOTO 310
c     XR=(XL+XU)/2
      XR=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
      DO 240 NI=2,IM

      AA=F(XL)+F(XR)
      IF (ABS(AA).EQ.0.01) GOTO 300
      IF (AA.LT.0.0)XU=XR
      IF (AA.GT.0.0)XL=XR

C     XN=(XL+XU)/2
      XN=XU-(F(XU)*(XL-XU))/(F(XL)-F(XU))
      IF (XN.EQ.0.0) GOTO 230
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LT.ES) GOTO 280
  230 XR=XN
  240 CONTINUE
      OPEN(6,FILE='programa9.out')
      WRITE(6,2)
  2   FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      WRITE(6,3)XR,EA
  3   FORMAT(' ',2F10.5)
      GOTO 310
  280 WRITE(6,4)XN,EA,NI
  4   FORMAT(' ',2F10.5,I5)
      GOTO 310
  300 WRITE(6,5)XR
  5   FORMAT(' ','LA RAIZ EXACTA ES =',F10.5)
  310 STOP
      END
```

```
      implicit real*8 (a-h,o-z)
      f(x)=-12*x**5-6*x**3+10
      open(5,file='programa10.dat')
      read(5,*)XL,XU,ES,IM
    1 format(3f10.0,I5)
      AA=f(XL)*f(XU)

      if(AA.GE.0.0) GOTO 310
      XR=(XL+XU)/2
      DO 240 NI=2,IM

      AA=f(XL)+f(XR)
      if(AA.EQ.0.0) GOTO 300
      if(AA.LT.0.0)XU=XR
      if(AA.Gt.0.0) XL=XR

      XN=(XL+XU)/2
      if(XN.EQ.0.0) GOTO 230
      EA=ABS((XN-XR)/XN)*100
      if(EA.LT.ES) GOTO 280
  230 XR=XN
  240 continue
      open(6,FILE='programa10.out')
      write(6,2)
    2 format(' ','no se encontro la raiz')
      write(6,3)XR,EA
    3 format(' ',2f10.3)
      GOTO 310
  280 write(6,4)XN,EA,NI
    4 format(' ',2f10.3,I5)
      GOTO 310
  300 write(6,5)XR
    5 format(' ','la raiz exacta es=',f10.3)
  310 stop
      end
```

```
      implicit real*8(a-h,o-z)
      F(X)=2*sin(sqrt(x)) -x
      OPEN(5,FILE='programa11.dat')
      READ(5,*) XR,ES,IM
      PRINT *, XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=F(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa11.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F15.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=2*x**3-11.7*x**2+17.7*x-5
      OPEN(5,FILE='programa12b.dat')
      READ(5,*) XR,ES,IM
      PRINT *, XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=F(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa12b.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F15.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=x**3+2*x**2-4*x+8
      FD(X)=3*x**2+4*x-4
      FD2(X)=6*X+4
      OPEN(5,FILE='programa21b.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(F(XR)*FD(XR))/(FD(XR)*FD(XR)-F(XR)*FD2(XR))
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa21b.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=2*x**3-11.7*x**2+17.7*x-5
      OPEN(5,FILE='programa12d.dat')
      READ(5,*) XR,XI,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(F(XR)*(XI-XR))/(F(XI)-F(XR))
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa12d.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,F(XR),F(XI)
3     FORMAT(' ',3F12.5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=2*x**3-11.7*x**2+17.7*x-5
      OPEN(5,FILE='programa12e.dat')
      READ(5,*) A,XR,XI,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(A*XR*F(XR)/(F(XR+A*XR)-F(XR)))
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa12e.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,F(XR),F(XI)
3     FORMAT(' ',3F12.5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=-x**2+1.8*x+2.5
      FD(X)=-2*x+1.8
      OPEN(5,FILE='programa13b.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa13b.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=-1+5.5*x-4*x**2+0.5*x**3
      FD(X)=5.5-8*x+3*0.5*x**2
      OPEN(5,FILE='programa14b.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa14b.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=-1+5.5*x-4*x**2+0.5*x**3
      FD(X)=5.5-8*x+3*0.5*x**2
      OPEN(5,FILE='programa15a.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa15a.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=-1+5.5*x-4*x**2+0.5*x**3
      FD(X)=5.5-8*x+3*0.5*x**2
      OPEN(5,FILE='programa15b.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa15b.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=-12-21*x+18*x**2-2.4*x**3
      OPEN(5,FILE='programa16b.dat')
      READ(5,*) XR,XI,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(F(XR)*(XI-XR))/(F(XI)-F(XR))
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa16b.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,F(XR),F(XI)
3     FORMAT(' ',3F12.5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=sin(x)+cos(1+x**2)-1
      OPEN(5,FILE='programa17a.dat')
      READ(5,*) XR,XI,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(F(XR)*(XI-XR))/(F(XI)-F(XR))
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa17a.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,F(XR),F(XI)
3     FORMAT(' ',3F12.5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=sin(x)+cos(1+x**2)-1
      OPEN(5,FILE='programa17b.dat')
      READ(5,*) XR,XI,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(F(XR)*(XI-XR))/(F(XI)-F(XR))
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa17b.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,F(XR),F(XI)
3     FORMAT(' ',3F12.5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=sin(x)+cos(1+x**2)-1
      OPEN(5,FILE='programa17c.dat')
      READ(5,*) XR,XI,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(F(XR)*(XI-XR))/(F(XI)-F(XR))
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa17c.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,F(XR),F(XI)
3     FORMAT(' ',3F12.5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=x**(7/2) -80
      OPEN(5,FILE='programa18.dat')
      READ(5,*) A,XR,XI,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(A*XR*F(XR)/(F(XR+A*XR)-F(XR)))
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa18.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,F(XR),F(XI)
3     FORMAT(' ',3F12.5)
      STOP
      END
```

problema6.9b

```
      implicit real*8(a-h,o-z)
      F(X)=0.95*x**3-5.9*x**2+10.9*x-6
      FD(X)=3*0.95*x**2-5.9*2*x+10.9
      OPEN(5,FILE='programa19b.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa19b.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=0.95*x**3-5.9*x**2+10.9*x-6
      OPEN(5,FILE='programa19c.dat')
      READ(5,*) XR,XI,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(F(XR)*(XI-XR))/(F(XI)-F(XR))
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa19c.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,F(XR),F(XI)
3     FORMAT(' ',3F12.5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=0.95*x**3-5.9*x**2+10.9*x-6
      OPEN(5,FILE='programa19d.dat')
      READ(5,*) A,XR,XI,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(A*XR*F(XR)/(F(XR+A*XR)-F(XR)))
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa19d.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,F(XR),F(XI)
3     FORMAT(' ',3F12.5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=8*sin(x)*exp(-x)-1
      FD(X)=8*(-1*sin(x)*exp(-x)+cos(x)*exp(-x))
      OPEN(5,FILE='programa20b.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa20b.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

problema6.10c

```
      implicit real*8(a-h,o-z)
      F(X)=8*sin(x)*exp(-x)-1
      OPEN(5,FILE='programa20c.dat')
      READ(5,*) XR,XI,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(F(XR)*(XI-XR))/(F(XI)-F(XR))
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa20c.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,F(XR),F(XI)
3     FORMAT(' ',3F12.5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=8*sin(x)*exp(-x)-1
      OPEN(5,FILE='programa20d.dat')
      READ(5,*) A,XR,XI,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(A*XR*F(XR)/(F(XR+A*XR)-F(XR)))
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa20d.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,F(XR),F(XI)
3     FORMAT(' ',3F12.5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=x**3+2*x**2-4*x+8
      FD(X)=3*x**2+4*x-4
      OPEN(5,FILE='programa21a.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa21a.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=x**3+2*x**2-4*x+8
      FD(X)=3*x**2+4*x-4
      FD2(X)=6*X+4
      OPEN(5,FILE='programa21b.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-(F(XR)*FD(XR))/(FD(XR)*FD(XR)-F(XR)*FD2(XR))
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa21b.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=-1*x**2+x+0.75
      OPEN(5,FILE='programa22a1.dat')
      READ(5,*) XR,ES,IM
      PRINT *, XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=F(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa22a1.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F15.5,I5)
      STOP
      END
```

problema6.12b

```
      implicit real*8(a-h,o-z)
      F(X)=(x**2)/(5*x+1)
      OPEN(5,FILE='programa22a1.dat')
      READ(5,*) XR,ES,IM
      PRINT *, XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=F(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa22a2.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F15.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=-1*x**2+x+0.75
      FD(X)=-2*x+1
      OPEN(5,FILE='programa22a1.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa22b1.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=(x**2)/(5*x+1)
      FD(X)=(5*x**2+2*x)/(25*x**2+10*x+1)
      OPEN(5,FILE='programa22a1.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa22b2.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=sqrt(5-(x-4)**2)+4
      FD(X)=-(x-4)/sqrt(5-(x-4)**2)
      OPEN(5,FILE='programa23a.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa23a.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=sqrt(16-x**2)
      FD(X)=-1*x/sqrt(16-x**2)
      OPEN(5,FILE='programa23b.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa23b.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

```
      implicit real*8(a-h,o-z)
      F(X)=tanh(x**2-9)
      FD(X)=-1*tanh(x**2-9)/x
      OPEN(5,FILE='programa24a.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa24a.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```

problema6.24

```fortran
      implicit real*8(a-h,o-z)
      F(X)=0.0074*x**4-0.284*x**3+3.355*x**2-12.183*x+5
      FD(X)=4*0.0074*x**3-3*0.284*x**2+2*3.355*x-12.183

      OPEN(5,FILE='programa25.dat')
      READ(5,*) XR,ES,IM
1     FORMAT(2F10.0,I5)
      DO 180 NI=1,IM
      XN=XR-F(XR)/FD(XR)
      IF(XN.EQ.0.0)GOTO 170
      EA=ABS((XN-XR)/XN)*100
      IF (EA.LE.ES) GOTO 210
170   XR=XN
180   CONTINUE
      OPEN(6,FILE='programa25.out')
      WRITE(6,2)
2     FORMAT(' ','NO SE ENCONTRO LA RAIZ')
      NI=NI-1
210   WRITE(6,3) XN,EA,NI
3     FORMAT(' ',2F12.5,I5)
      STOP
      END
```