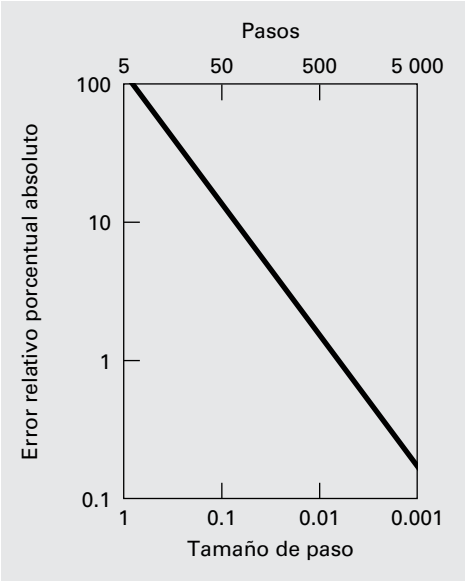

Figura 25.5
Efecto del tamaño de paso sobre el error de truncamiento global en el método de Euler para la integral de $y' = -2x^3 + 12x^2 - 20x + 8.5$. La gráfica muestra el error global relativo porcentual absoluto en $x = 5$ en función del tamaño de paso.

El efecto de algunas reducciones del tamaño de paso sobre el error de truncamiento global del método de Euler se ilustra en la figura 25.5, esta gráfica muestra el error relativo porcentual absoluto en $x = 5$ en función del tamaño de paso para el problema que se estudió en los ejemplos 25.1 a 25.3. Observe que aun cuando h se reduce a 0.001, el error todavía es mayor de 0.1%. Ya que este tamaño de paso corresponde a 5 000 pasos para ir desde $x = 0$ hasta $x = 5$, la gráfica sugiere que una técnica de primer orden, como el método de Euler, requiere de muchos cálculos para obtener niveles de error aceptables. Más adelante en este capítulo, se presentarán técnicas de orden superior que dan mucha mayor exactitud con el mismo trabajo de cálculo. Sin embargo, deberá observarse que, a pesar de su ineficiencia, la simplicidad del método de Euler lo hace una opción extremadamente atractiva para muchos problemas de ingeniería. Puesto que es muy fácil de programar, en particular la técnica es útil para llevar a cabo análisis rápidos. En la próxima sección se desarrolla un algoritmo computacional para el método de Euler.

25.1.2 Algoritmo para el método de Euler

Los algoritmos para las técnicas de un paso como el método de Euler son muy simples de programar. Como se especificó al inicio de este capítulo, todos los métodos de un paso tienen la forma general

$$\text{Nuevo valor} = \text{valor anterior} + \text{pendiente} \times \text{tamaño de paso} \tag{25.10}$$

En lo único que difieren los métodos es en el cálculo de la pendiente.

Suponga que usted quiere realizar el cálculo simple expuesto en la tabla 25.1. Es decir, a usted le gustaría utilizar el método de Euler para integrar $y' = -2x^3 + 12x^2 - 20x + 8.5$, con la condición inicial de que $y = 1$ en $x = 0$. Usted quiere integrarla hasta $x = 4$ usando un tamaño de paso de 0.5, y desplegar todos los resultados. Un pseudocódigo simple para realizar esto será como el de la figura 25.6.

Aunque este programa "hará el trabajo" de duplicar los resultados de la tabla 25.1 no está muy bien diseñado. Primero, y ante todo, no es muy modular. Aunque esto no es muy importante para un programa así de pequeño, podría resultar crítico si deseamos modificar y mejorar el algoritmo.

Además, existen algunos detalles relacionados con la forma en que se establecen las iteraciones. Por ejemplo, suponga que el tamaño de paso se volviese muy pequeño para obtener mayor exactitud. En tales casos, debido a que se despliega cada valor calculado, la cantidad de valores de salida podría ser muy grande. Asimismo, el algoritmo supone que el intervalo de cálculo es divisible entre el tamaño de paso. Por último, la acumulación de x en la línea $x = x + dx$ puede estar sujeta a la cuantificación de errores analizada en la sección 3.4.1. Por ejemplo, si dx se cambiara a 0.01 y se usara la representación estándar IEEE de punto flotante (cerca de siete cifras significativas), el resultado al final del cálculo sería 3.999997 en lugar de 4. Para $dx = 0.001$, ¡sería 3.999892!

En la figura 25.7 se muestra un algoritmo mucho más modular que evita esas dificultades. El algoritmo no despliega todos los valores calculados. En lugar de eso, el usuario especifica un intervalo de salida, *xout*, que indica el intervalo en el cual los resultados calculados se guardan en arreglos, xp_m y yp_m. Dichos valores se guardan en arreglos, de tal modo que se puedan desplegar de diferentes formas una vez que termine el cálculo (por ejemplo, impresos graficados, escritos en un archivo).

```
'intervalo de integración
xi = 0
xf = 4
'variables iniciales
x = xi
y = 1
'establece el tamaño de paso y determina el
'número de pasos de cálculo
dx = 0.5
nc = (xf - xi)/dx
'condiciones de salida inicial
PRINT x, y
'ciclo para implementar el método de Euler
'y despliegue de resultados
DOFOR i = 1, nc
  dydx = -2x³ + 12x² - 20x + 8.5
  y = y + dydx · dx
  x = x + dx
  PRINT x, y
END DO
```

Figura 25.6
Pseudocódigo para una primera versión del método de Euler.

***a*) Programa principal o "manejador"**

```
Asigna valores para
y = valor inicial variable dependiente
xi = valor inicial variable independiente
xf = valor final variable independiente
dx = cálculo del tamaño de paso
xout = intervalo de salida

x = xi
m = 0
xp_m = x
yp_m = y
DO
  xend = x + xout
  IF (xend > xf) THEN xend = xf
  h = dx
  CALL Integrator (x, y, h, xend)
  m = m + 1
  xp_m = x
  yp_m = y
  IF (x ≥ xf) EXIT
END DO
DISPLAY RESULTS
END
```

***b*) Rutina para tomar un paso de salida**

```
SUB Integrator (x, y, h, xend)
  DO
    IF (xend - x < h) THEN h = xend - x
    CALL Euler (x, y, h, ynew)
    y = ynew
    IF (x ≥ xend) EXIT
  END DO
END SUB
```

***c*) Método de Euler para una EDO sola**

```
SUB Euler (x, y, h, ynew)
  CALL Derivs(x, y, dydx)
  ynew = y + dydx * h
  x = x + h
END SUB
```

***d*) Rutina para determinar la derivada**

```
SUB Derivs (x, y, dydx)
  dydx = ...
END SUB
```

Figura 25.7
Pseudocódigo para una versión modular "mejorada" del método de Euler.

El programa principal toma grandes pasos de salida y llama a una rutina denominada Integrator que hace los pasos de cálculo más pequeños. Observe que los ciclos o *loops* que controlan tanto los pasos grandes como los pequeños terminan basándose en condiciones lógicas. Así, los intervalos no tienen que ser divisibles entre los tamaños de paso.

La rutina Integrator llama después a la rutina Euler que realiza un solo paso con el método de Euler. La rutina Euler llama a una rutina Derivate que calcula el valor de la derivada.

Aunque parecería que tal forma modular es demasiada para el presente caso, facilitará en gran medida la modificación del programa posteriormente. Por ejemplo, aunque el programa de la figura 25.7 está diseñado específicamente para implementar el método de Euler, el módulo de Euler es la única parte que es específica para el método. Así, todo lo que se requiere para aplicar este algoritmo a otros métodos de un paso es modificar esta rutina.

EJEMPLO 25.4 Solución de una EDO con la computadora

Planteamiento del problema Es factible desarrollar un programa computacional a partir del pseudocódigo de la figura 25.7. Este software también sirve para resolver otro problema relacionado con la caída del paracaidista. Usted recordará de la parte uno que nuestro modelo matemático para la velocidad se basaba en la segunda ley de Newton, escrita en la forma:

$$\frac{dv}{dt} = g - \frac{c}{m}v \tag{E25.4.1}$$

Esta ecuación diferencial se resolvió tanto de manera analítica (ejemplo 1.1) como numérica usando el método de Euler (ejemplo 1.2). Las soluciones fueron para el caso donde $g = 9.81$, $c = 12.5$, $m = 68.1$ y $v = 0$ en $t = 0$.

El objetivo del presente ejemplo es repetir esos cálculos numéricos empleando un modelo más complicado para la velocidad con base en una descripción matemática más completa de la fuerza de arrastre causada por la resistencia del viento. Este modelo se obtiene como: