EJEMPLO C.3 **Ilustración del error relativo**

Sea $x = 2$ y $x^* = 2.1$. Entonces $\varepsilon_a = 0.1$ y $\varepsilon_r = \dfrac{0.1}{2} = 0.05$. Si $x_1 = 2\,000$ y $x_1{}^* = 2\,000.1$, entonces de nuevo $\varepsilon_a = 0.1$. Pero ahora $\varepsilon_r = \dfrac{0.1}{2.000} = 0.00005$. Muchas personas estarán de acuerdo en que el error de 0.1 en el primer caso es más significativo que el error de 0.1 en el segundo.

Una parte importante del análisis numérico se refiere a preguntas sobre **convergencia** y **estabilidad**. Si x es la solución exacta al problema y el método computacional da valores aproximados x_n, entonces el método converge si, teóricamente, x_n tiende a x cuando n crece. Más aún, si se puede demostrar que los errores de redondeo no se acumularán de forma que la respuesta sea muy poco exacta, entonces se dice que el **método** es **estable**.

Convergencia
Estabilidad

Método estable

Es sencillo proporcionar un ejemplo de un procedimiento en el que el error de redondeo sea bastante grande. Suponga que se quiere calcular $y = \dfrac{1}{(x - 0.66666665)}$. Para $x = \dfrac{2}{3}$ si la computadora trunca, entonces $x = 0.66666666$ y $y = \dfrac{1}{0.00000001} = 10^8 = 10 \times 10^7$. Si la computadora redondea, entonces $x = 0.66666667$ y $y = \dfrac{1}{0.00000002} = 5 \times 10^7$. La diferencia en este caso es enorme. La solución exacta es

$$\frac{1}{\left(\frac{2}{3} - \frac{66666665}{100\,000\,000}\right)} = \frac{1}{\left(\frac{200\,000\,000}{300\,000\,000} - \frac{199\,999\,995}{300\,000\,000}\right)} = \frac{1}{\frac{5}{300\,000\,000}} = \frac{300\,000\,000}{5} = 60\,000\,000 = 6 \times 10^7.$$

Nota. La estabilidad aquí no es causa de preocupación. Sin embargo, las personas que diseñan el software sí se preocupan mucho por este factor. El lector debe saber que quien se dedica a análisis numérico y diseña software elige los algoritmos (o desarrolla nuevos) que tienden a minimizar las consecuencias adversas. En particular, MATLAB utiliza programas de muy alta calidad. En la actualidad, ningún principiante bien informado desarrolla su propio software. Se usan subrutinas de diseños probados.

Complejidad computacional

Al resolver problemas en una computadora surgen dos preguntas naturales:

¿Qué tan exactas son mis respuestas?
¿Cuánto tiempo llevará hacerlo?

Trataremos de dar respuesta a la primera pregunta en la parte inicial de esta sección. Para contestar la segunda, debe estimarse el número de pasos requeridos para llevar a cabo cierto cálculo. La **complejidad computacional** de un problema es una medida del número de operaciones aritméticas necesarias para resolver el problema y el tiempo necesario para llevar a cabo todas las operaciones requeridas.

Complejidad computacional

Existen dos tipos de operaciones básicas que se llevan a cabo en una computadora: suma o resta, y multiplicación o división.

De esta forma, con el fin de estimar el tiempo necesario para resolver un problema en una computadora, primero deben contarse las sumas, restas, multiplicaciones y divisiones involucradas en la solución.

Contar las operaciones necesarias para resolver un problema con frecuencia es difícil. Se ilustra cómo se puede hacer en el caso de eliminación de Gauss-Jordan. Para simplificar, la suma y la resta se manejarán como la misma operación y la multiplicación y la división igual.

EJEMPLO C.4 **Cuenta de sumas y multiplicaciones en la eliminación de Gauss-Jordan**

Sea A una matriz invertible de $n \times n$. Estime el número de sumas y multiplicaciones necesarias para resolver el sistema $A\mathbf{x} = \mathbf{b}$ mediante eliminación de Gauss-Jordan.