

Métodos Numéricos

Aplicados a la Ingeniería



Métodos Numéricos Aplicados a la Ingeniería

Antonio Nieves Hurtado

Federico C. Domínguez Sánchez[†]

*Profesores de la Academia de Matemáticas Aplicadas
ESIQIE-IPN*

PRIMERA EDICIÓN EBOOK
MÉXICO, 2014

GRUPO EDITORIAL PATRIA

**Para establecer comunicación
con nosotros puede hacerlo por:**



correo:
Renacimiento 180, Col. San Juan
Tlihuaca, Azcapotzalco,
02400, México, D.F.



fax pedidos:
(01 55) 5354 9109 • 5354 9102



e-mail:
info@editorialpatria.com.mx



home page:
www.editorialpatria.com.mx

Dirección editorial: Javier Enrique Callejas
Coordinadora editorial: Estela Delfín Ramírez
Supervisor de pre prensa: Gerardo Briones González
Diseño de portada: Juan Bernardo Rosado Solís
Ilustraciones: Braulio Morales
Fotografías: © 2011, Thinkstockphoto/ Nemesis
Revisión técnica: Dr. José Job Flores Godoy
Departamento de Matemáticas
Universidad Iberoamericana

Métodos numéricos aplicados a la ingeniería, 4a. edición

Derechos reservados:

© 2014, Antonio Nieves Hurtado / Federico C. Domínguez Sánchez

© 2014, GRUPO EDITORIAL PATRIA, S.A. DE C.V.

Renacimiento 180, Colonia San Juan Tlihuaca

Delegación Azcapotzalco, Código Postal 02400, México, D.F.

Miembro de la Cámara Nacional de la Industria Editorial Mexicana
Registro Núm. 43

ISBN ebook: 978-607-438-926-5

Queda prohibida la reproducción o transmisión total o parcial del contenido de la presente obra en cualesquiera formas, sean electrónicas o mecánicas, sin el consentimiento previo y por escrito del editor.

Impreso en México
Printed in Mexico

Primera edición ebook: 2014

*A dos ángeles en el cielo:
a mi madre Isabel y un gran
amigo, Fred Eggli.*

*A tres ángeles en la Tierra: mi
padre, Antonio; Mom, Violet
Eggli y mi esposa, Alda María.*

Antonio

*A la memoria de mis padres,
Aurelia y Cliserio; y de mis
hermanas, Isabel y María Elma.*

*A mis hermanos, Susana y
Alejandro; a mis hijos, Alura
Lucia, Alejandra y Federico;
a mi nieto Osiel; y a mi esposa,
María Sara Araceli.*

Federico

Contenido

Prefacio	xi
1 Errores	1
1.1 Sistemas numéricos	3
1.2 Manejo de números en la computadora	9
1.3 Errores	13
1.4 Algoritmos y estabilidad	22
Ejercicios	23
Problemas propuestos	27
2 Solución de ecuaciones no lineales	31
2.1 Método de punto fijo	32
ALGORITMO 2.1 Método de punto fijo	38
2.2 Método de Newton-Raphson	48
ALGORITMO 2.2 Método de Newton-Raphson	51
2.3 Método de la secante	54
ALGORITMO 2.3 Método de la secante	56
2.4 Método de posición falsa	57
ALGORITMO 2.4 Método de posición falsa	61
2.5 Método de la bisección	61
2.6 Problemas de los métodos de dos puntos y orden de convergencia	63
2.7 Aceleración de convergencia	66
ALGORITMO 2.5 Método de Steffensen	70
2.8 Búsqueda de valores iniciales	71
2.9 Raíces complejas	77
ALGORITMO 2.6 Método de Müller	85
2.10 Polinomios y sus ecuaciones	86
ALGORITMO 2.7 Método de Horner	89
ALGORITMO 2.8 Método de Horner iterado	91
Ejercicios	100
Problemas propuestos	131

3	Matrices y sistemas de ecuaciones lineales	145
3.1	Matrices	146
	ALGORITMO 3.1 Multiplicación de matrices	153
3.2	Vectores	158
3.3	Independencia y ortogonalización de vectores	167
	ALGORITMO 3.2 Ortogonalización de Gram-Schmidt	179
3.4	Solución de sistemas de ecuaciones lineales	183
	ALGORITMO 3.3 Eliminación de Gauss	189
	ALGORITMO 3.4 Eliminación de Gauss con pivoteo	193
	ALGORITMO 3.5 Método de Thomas	204
	ALGORITMO 3.6 Factorización directa	210
	ALGORITMO 3.7 Factorización con pivoteo	211
	ALGORITMO 3.8 Método de Doolittle	214
	ALGORITMO 3.9 Factorización de matrices simétricas	217
	ALGORITMO 3.10 Método de Cholesky	220
3.5	Métodos iterativos	231
	ALGORITMO 3.11 Métodos de Jacobi y Gauss-Seidel	242
3.6	Valores y vectores propios	248
	Ejercicios	254
	Problemas propuestos	272
4	Sistemas de ecuaciones no lineales	289
4.1	Dificultades en la solución de sistemas de ecuaciones no lineales	291
4.2	Método de punto fijo multivariable	295
	ALGORITMO 4.1 Método de punto fijo multivariable	301
4.3	Método de Newton-Raphson	302
	ALGORITMO 4.2 Método de Newton-Raphson multivariable	310
4.4	Método de Newton-Raphson modificado	311
	ALGORITMO 4.3 Método de Newton-Raphson modificado	315
4.5	Método de Broyden	316
	ALGORITMO 4.4 Método de Broyden	320
4.6	Aceleración de convergencia	321
	ALGORITMO 4.5 Método del descenso de máxima pendiente	337
4.7	Método de Bairstow	339
	Ejercicios	345
	Problemas propuestos	359
5	Aproximación funcional e interpolación	367
5.1	Aproximación polinomial simple e interpolación	370
	ALGORITMO 5.1 Aproximación polinomial simple	373
5.2	Polinomios de Lagrange	373
	ALGORITMO 5.2 Interpolación con polinomios de Lagrange	379

5.3	Diferencias divididas	381
	ALGORITMO 5.3 Tabla de diferencias divididas	384
5.4	Aproximación polinomial de Newton	385
	ALGORITMO 5.4 Interpolación polinomial de Newton	389
5.5	Polinomio de Newton en diferencias finitas	390
5.6	Estimación de errores en la aproximación	399
5.7	Aproximación polinomial segmentaria	405
5.8	Aproximación polinomial con mínimos cuadrados	412
	ALGORITMO 5.5 Aproximación con mínimos cuadrados	420
5.9	Aproximación multilínea con mínimos cuadrados	420
	Ejercicios	424
	Problemas propuestos	438
6	Integración y diferenciación numérica	451
6.1	Métodos de Newton-Cotes	454
	ALGORITMO 6.1 Método trapezoidal compuesto	464
	ALGORITMO 6.2 Método de Simpson compuesto	468
6.2	Cuadratura de Gauss	478
	ALGORITMO 6.3 Cuadratura de Gauss-Legendre	486
6.3	Integrales múltiples	487
	ALGORITMO 6.4 Integración doble por Simpson 1/3	494
6.4	Diferenciación numérica	495
	ALGORITMO 6.5 Derivación de polinomios de Lagrange	505
	Ejercicios	505
	Problemas propuestos	521
7	Ecuaciones diferenciales ordinarias	535
7.1	Formulación del problema de valor inicial	538
7.2	Método de Euler	539
	ALGORITMO 7.1 Método de Euler	543
7.3	Método de Taylor	543
7.4	Método de Euler modificado	546
	ALGORITMO 7.2 Método de Euler modificado	549
7.5	Métodos de Runge-Kutta	549
	ALGORITMO 7.3 Método de Runge-Kutta de cuarto orden	554
7.6	Métodos de predicción-corrección	555
	ALGORITMO 7.4 Método predictor-corrector	568
7.7	Ecuaciones diferenciales ordinarias de orden superior y sistemas de ecuaciones diferenciales ordinarias	569
	ALGORITMO 7.5 Método de Runge-Kutta de cuarto orden para un sistema de dos ecuaciones diferenciales ordinarias	577

7.8	Formulación del problema de valores en la frontera	578
7.9	Ecuaciones diferenciales rígidas	582
	Ejercicios	586
	Problemas propuestos	608
8	Ecuaciones diferenciales parciales	621
8.1	Obtención de algunas ecuaciones diferenciales parciales a partir de la modelación de fenómenos físicos (ecuación de calor y ecuación de onda)	623
8.2	Aproximación de derivadas por diferencias finitas	627
8.3	Solución de la ecuación de calor unidimensional	632
	ALGORITMO 8.1 Método explícito	637
	ALGORITMO 8.2 Método implícito	648
8.4	Convergencia (método explícito), estabilidad y consistencia	651
8.5	Método de Crank-Nicholson	654
	ALGORITMO 8.3 Método de Crank-Nicholson	660
8.6	Otros métodos para resolver el problema de conducción de calor unidimensional	660
8.7	Solución de la ecuación de onda unidimensional	663
8.8	Tipos de condiciones frontera en procesos físicos y tratamientos de condiciones frontera irregulares	670
	Ejercicios	674
	Problemas propuestos	683
	Respuestas a problemas seleccionados	691
	Índice analítico	705

Prefacio

Objetivo del libro

El análisis numérico y sus métodos son una dialéctica entre el análisis matemático cualitativo y el análisis matemático cuantitativo. El primero nos dice, por ejemplo, que bajo ciertas condiciones algo existe, que es o no único, etc.; en tanto que el segundo complementa al primero, permitiendo calcular *aproximadamente* el valor de aquello que existe.

Así pues, el análisis numérico es una reflexión sobre los cursos tradicionales de cálculo, álgebra lineal y ecuaciones diferenciales, entre otros, que se concreta en una serie de *métodos* o *algoritmos*, cuya característica principal es la posibilidad de obtener resultados *numéricos* de problemas matemáticos de cualquier tipo a partir de números y de un número finito de operaciones aritméticas. La finalidad de este libro es el estudio y uso racional de dichos algoritmos en diferentes áreas de ingeniería y ciencias.

Enfoque del libro

La noción de algoritmo es un concepto clásico en las matemáticas, anterior a la aparición de las computadoras y las calculadoras. Por ejemplo, en el Papiro de Ahmes o de Rhind (de hacia el año 1650 a.C.) se encuentra la técnica de posición falsa aplicada a la solución de ecuaciones lineales y en el Jiu Zhang Suanshu (el libro más famoso de la matemática china, del año 200 a. C.) se resolvían sistemas de ecuaciones lineales con el método conocido hoy en día como eliminación de Gauss.

En realidad, en la enseñanza básica tradicional todos aprendimos algoritmos como el de la división, la multiplicación y la extracción de raíces cuadradas. Con el transcurso del tiempo, los dos primeros suelen convertirse en las operaciones más conocidas y practicadas (aunque quizás también, en las más incomprendidas) y, el tercero en la operación más fácilmente olvidada.

A fin de no caer en un curso más de recetas matemáticas desvinculadas y sin sentido, hemos desarrollado el material de este libro en torno a tres ideas fundamentales: el punto fijo, la eliminación de Gauss y la aproximación de funciones. Para instrumentarlas empleamos como recursos didácticos, en cada método o situación, diferentes sistemas de representación: el gráfico, el tabular y el algebraico, y promovemos el paso entre ellos. Con el fin de que el lector vea claramente la relación entre los métodos que estudia en el libro y su aplicación en el contexto real, se resuelven al final de cada capítulo alrededor de diez o más problemas de diferentes áreas de aplicación. De igual manera, hacemos énfasis en el uso de herramientas como la calculadora y la computadora, así como en la importancia de la visualización en los problemas. Dada la importancia de cada uno de estos aspectos, los trataremos con cierto detalle a continuación.

Los métodos numéricos y las herramientas computacionales

Computadora

Cada algoritmo implica numerosas operaciones lógicas, aritméticas y en múltiples casos graficaciones, por ello la computadora es fundamental para el estudio de éstos. El binomio computadora-lenguaje de alto nivel (Fortran, Basic, C y otros) ha sido utilizado durante muchos años para la enseñanza y el

aprendizaje de los métodos numéricos. Si bien esta fórmula ha sido exitosa y sigue aún vigente, también es cierto que la aparición de paquetes comerciales como Mathcad, Maple, Matlab (por citar algunos de los más conocidos) permite nuevos acercamientos al estudio de los métodos numéricos. Por ejemplo, ha permitido que la programación sea más sencilla y rápida y facilitado además la construcción directa de gráficas en dos y tres dimensiones, así como la exploración de conjeturas y la solución numérica directa de problemas matemáticos.

En respuesta a estas dos vertientes, se acompaña el libro con un CD donde se han mantenido los programas de la segunda edición (Fortran 90, Pascal, Visual Basic y C) y se han incorporado 34 nuevos programas en Visual Basic. En numerosos ejemplos, ejercicios y problemas utilizamos o sugerimos además el empleo de los paquetes matemáticos mencionados arriba.

Calculadoras graficadoras

Las calculadoras graficadoras (como la TI-89, TI-92 Plus, Voyage 200, HP-48 o HP-49) disponen hoy en día de poderosos elementos como:

- Un sistema algebraico computarizado (CAS por sus siglas en inglés) que permite manipulaciones simbólicas y soluciones analíticas de problemas matemáticos.
- La graficación en dos y tres dimensiones con facilidades como el *zoom* y el *trace*.
- La posibilidad de resolver numéricamente problemas matemáticos.
- La posibilidad de programar y utilizar a través de dicha programación los recursos mencionados en los incisos anteriores, convirtiéndose así el conjunto lenguaje-recursos en una herramienta aún más poderosa que un lenguaje procedural como Basic o C.

Finalmente, su bajo costo, portabilidad y posibilidades de comunicación con sitios Web donde es posible actualizar, intercambiar y comprar programas e información, permiten plantear un curso de métodos numéricos sustentado en la calculadora o una combinación de calculadora y computadora. A fin de apoyar esta acción hemos incorporado en muchos de los ejemplos y ejercicios programas que funcionan en las calculadoras TI-89, TI-92, TI-92 Plus, Voyage 200.

Visualización

A partir de las posibilidades gráficas que ofrecen las computadoras y las calculadoras, la visualización (un recurso natural del ser humano) ha tomado mayor importancia y se ha podido utilizar en las matemáticas de diferentes maneras, como la aprehensión de los conceptos, la solución de problemas, la ilustración de los métodos y, en general, para darle un aspecto dinámico a diversas situaciones físicas. Así, hemos intentado aprovechar cada uno de estos aspectos y aplicarlos a lo largo del libro siempre que fue posible. Por ejemplo, en el capítulo 4 se presentan ilustraciones novedosas de los métodos para resolver sistemas de ecuaciones no lineales (inclusive se han puesto en color varias de esas gráficas a fin de tener una mejor apreciación de las intersecciones de superficies y de las raíces), ilustraciones de conceptos abstractos como el criterio de convergencia del método de punto fijo univariable y la ponderación de pendientes en los métodos de Runge-Kutta. Además, se incluyen varios ejercicios en Visual Basic donde se simula algún fenómeno como el de crecimiento de poblaciones (ejercicio 7.13), amortiguación en choques (ejercicio 7.11) y el desplazamiento de una cuerda vibrante (ejemplo 8.5). En estos últimos se pueden observar los resultados numéricos en tiempo real y la gráfica que van generando e incluso modificar los parámetros para hacer exploraciones propias. Todos ellos aparecen en el CD y se identifican con el icono correspondiente.

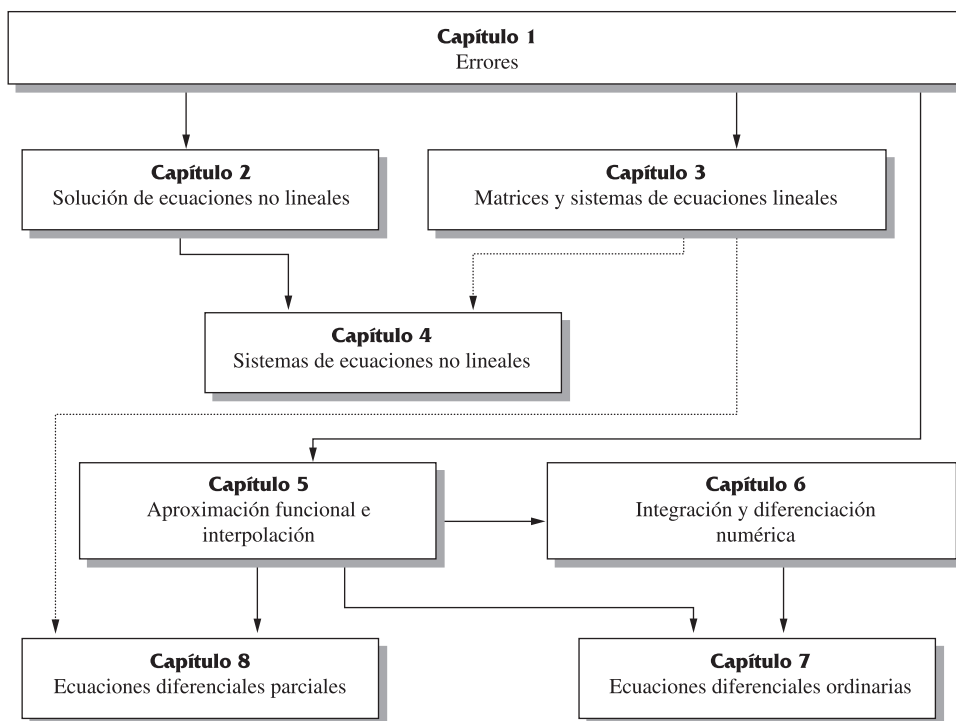
Prerrequisitos

Generalmente los cursos de métodos numéricos siguen a los de cálculo en una variable, el de ecuaciones diferenciales ordinarias y el de programación. No obstante, sólo consideramos los cursos de cálculo y de programación (en el modelo computadora-lenguaje de alto nivel) como prerrequisitos. Los conocimientos de álgebra lineal requeridos, así como los elementos básicos para estudiar las técnicas de las ecuaciones diferenciales parciales, se exponen en los capítulos correspondientes. Si bien los conceptos y técnicas analíticas de las ecuaciones diferenciales ordinarias serían benéficos y complementarios con los métodos de este curso, no son, sin embargo, material indispensable.

Secuencias sugeridas

Como se dijo antes, el libro se desarrolla alrededor de tres ideas matemáticas fundamentales: punto fijo, eliminación de Gauss y aproximación de funciones. Las dos primeras se estudian en los capítulos 2 y 3, respectivamente, y junto con el capítulo 4 constituyen la parte algebraica del libro. Un primer curso (semestral) de métodos numéricos podría organizarse con los primeros cuatro capítulos del libro, seleccionando las secciones que correspondan a su programa de estudios o a las necesidades específicas del curso.

Red de temas e interrelación



—> Dependencia en requisitos básicos y mecánica de cálculo de los algoritmos

.....> Dependencia solamente de la mecánica de cálculo de los algoritmos

La tercera idea matemática clave en el libro es la de aproximación de funciones, que se presenta en el capítulo 5 y sustenta el material de análisis: integración y derivación numérica (capítulo 6), y que más adelante será la base de la parte de dinámica: ecuaciones diferenciales ordinarias y parciales (capítulos 7 y 8, respectivamente). De este modo, un curso semestral podría configurarse con los capítulos 1, 5, 6, 7 o bien 1, 5, 6, 7 y 8 (ver red de temas e interrelación).

Debido a ello y al hecho de que algunos tecnológicos y universidades sólo tienen un curso de un semestre de métodos numéricos, podría elaborarse éste con una secuencia como capítulos 1, 2, 3, 5 o bien 1, 2, 5, 6, por ejemplo.

Finalmente, recomendamos al maestro que cualquiera que sea la secuencia y las secciones elegidas en cada una de ellas, se discuta y trabaje con los ejemplos resueltos de final de cada capítulo.

Novedades en esta edición

- *Entradas de capítulo*

Cada capítulo se ilustra con una situación proveniente de la cotidianeidad o del ámbito ingenieril, mostrando la necesidad de emplear métodos numéricos en el análisis de tales situaciones. La finalidad es que el lector vea los métodos numéricos como algo vinculado a su realidad circundante y futuro ejercicio profesional.

- *Proyectos*

Al final de cada capítulo se plantean uno o dos proyectos, cuya característica es una considerable demanda intelectual y de trabajo para los lectores. En éstos puede requerirse consultar bibliografía adicional, integrar conocimientos diversos y reflexionar sobre los conceptos matemáticos y de ingeniería involucrados. A cambio, los estudiantes y profesores podrán involucrarse en la explotación de ideas novedosas y enfrentar retos, consiguiendo con ello un mejor manejo de los métodos estudiados, pero sobre todo disfrutar del aspecto lúdico de la resolución de este tipo de problemas.

- *Programas*

A los 39 programas de la versión anterior se agregan 34 nuevos programas que, en su mayoría, permiten la visualización gráfica y el seguimiento numérico, paso a paso, de la evolución de los métodos al resolver algún problema planteado por el usuario. A continuación se describen por capítulo.

Capítulo 1. Conversión de números entre distintos sistemas numéricos.

Capítulo 2. Métodos: punto fijo, Newton-Raphson, secante, posición falsa y bisección; para todos ellos se cuenta con un capturador de funciones que permite al usuario proponer sus funciones.

Capítulo 3. Programas de multiplicación de matrices, ortogonalización de vectores, métodos de eliminación gaussiana, factorización LU y los métodos iterativos de Jacobi y Gauss-Seidel.

Capítulo 4. Programa para graficación de funciones de dos variables; métodos de punto fijo, Newton-Raphson y el de descenso de máxima pendiente.

Capítulo 5. Programa de interpolación con diferencias divididas. Se cuenta con un programa que permite al usuario, mediante visualizaciones y manipulaciones virtuales, captar la idea fundamental que subyace en la aproximación lineal por mínimos cuadrados; también con un programa que permite el ajuste a partir de un menú de modelos comunes.

Capítulo 6. Se presenta un programa para graficar una función analítica proporcionada por el usuario; puede observarse en éste la recta tangente en cada punto de la función y la gráfica de la derivada de dicha función. Asimismo, se dan programas para la derivación de una función dada tabularmente y para la integración de funciones analíticas por cuadratura de Gauss-Legendre con dos y tres puntos.

Capítulo 7. Programas para los métodos de Euler y Runge-Kutta (segundo, tercero y cuarto orden).

- *Nuevos ejercicios y problemas*

Se han eliminado algunos ejercicios y problemas e incorporado nuevos a lo largo del texto.

- *Íconos utilizados en la tercera edición*

El libro se rediseñó íntegramente para facilitar su lectura; en particular, se incluyeron los íconos que aparecen a continuación para permitir al lector identificar con rapidez los apoyos con los que cuenta:



Guiones de Matlab.



Programas para las calculadoras Voyage 200.



Indica un programa en Visual Basic que se ha incluido en el CD y que le ayuda en la solución de ese ejercicio o ejemplo.



La solución se incluye en el CD (en Mathcad, Matlab y Mathematica).

Materiales adicionales



CD-ROM diseñado especialmente para la cuarta edición, que contiene:

- Programas fuente en Visual Basic y sus respectivos ejecutables que corren en Windows 95 o posterior para la solución de ejemplos y ejercicios.
- Documentos de Mathcad y guiones de Matlab. Los documentos en Mathcad permiten dar un sentido exploratorio a los métodos numéricos y los guiones de Matlab, acceso a uno de los paquetes más poderosos para resolver problemas matemáticos.
- Algoritmos, descripción de los programas de cómputo y explicaciones detalladas de su uso.
- Ligas a sitios donde el lector encontrará tutoriales de Mathcad, Matlab y Mathematica, en los que podrá aprender a usar estos paquetes.
- Sugerencias de empleo de software comercial (Mathcad y Matlab, Mathematica) para resolver un gran número de ejemplos y ejercicios.

Agradecimientos

Esta obra tiene su origen en apuntes para los cursos de métodos numéricos en la carrera de Ingeniería Química Industrial del Instituto Politécnico Nacional, desarrollados durante una estancia de año sabático en el Instituto Tecnológico de Celaya y, posteriormente, a raíz de un certamen organizado por el propio IPN, se convirtieron en una propuesta de libro que ganó el primer lugar en el Primer Certamen Editorial Politécnico en 1984. Desde entonces, con actualizaciones continuas, ha sido utilizado como texto para estos cursos en diferentes instituciones del país y del extranjero. Los autores agradecen al

Instituto Politécnico Nacional la facilidad que otorgó para que la editorial CECSA, ahora Grupo Editorial Patria, lo publicara.

Agradecemos también a los investigadores y profesores que colaboraron para la realización de esta nueva edición y de las anteriores.

- Dr. Gustavo Iglesias Silva. Texas A & M University e Instituto Tecnológico de Celaya.
- Dr. Ramón Duarte Ramos. Universidad Autónoma de Sinaloa.
- Dr. Horacio Orozco Mendoza. Instituto Tecnológico de Celaya.
- Ing. Adriana Guzmán López. Instituto Tecnológico de Celaya.
- M. en C. Miguel Hesiquio Garduño. ESIQIE-IPN.
- Ing. Arturo Javier López García. ESIQIE-IPN.
- Ing. Rogelio Márquez Nuño. ESIQIE-IPN.
- Ing. César Gustavo Gómez Sierra. ESIQIE-IPN.
- Dr. Ricardo Macías Salinas. ESIQIE-IPN.
- Ing. Blanca Navarro Anaya. ESIQIE-IPN.
- Dr. César Cristóbal Escalante. Universidad de Quintana Roo.
- Dr. Carlos Angüis Terrazas†. ESIQIE-IPN.
- Dr. Daniel Gómez García. Universidad Autónoma de Saltillo.
- Ing. Manuel Guía Calderón. Universidad de Guanajuato, Campus Salamanca.
- Ing. José Luis Turrizza Pinto. ESIME ZAC.-IPN.
- Q. Amparo Romero Márquez. Instituto Tecnológico de Celaya.
- Dr. Guillermo Marroquín Suárez†. ESIQIE-IPN.
- Q. Gabriela Romero Márquez. Instituto Tecnológico de Celaya.
- Ing. Leslie Gómez Ortíz. ESIQIE-IPN.

Agradecemos al Dr. José Job Flores Godoy de la Universidad Iberoamericana por la completa y extraordinaria revisión técnica de la obra y al Profesor Raúl Guinovart Díaz del ITESM-CEM por sus comentarios para mejorar la obra.

Nuestro agradecimiento especial al personal del Grupo Editorial Patria, y en particular a la Ing. Estela Delfín Ramírez, nuestra editora, por su esmero y pulcritud, así como por la serie de ideas y sugerencias que permitieron enriquecer esta nueva edición.

Errores

El 4 de junio de 1996 el cohete no tripulado *Ariane 5*, lanzado por la Agencia Espacial Europea, explotó 40 segundos después de despegar. Después de una década de desarrollo con una inversión de 7 mil millones de dólares, el cohete hacía su primer viaje al espacio. El cohete y su carga estaban valuados en 500 millones de dólares. Un comité investigó las causas de la explosión y en dos semanas emitió un reporte. La causa de la falla fue un error de software en el sistema de referencia inercial. Específicamente un número de punto flotante de 64 bits, relativo a la velocidad horizontal del cohete con respecto a la plataforma, fue convertido a un entero con signo de 16 bits. El número entero resultó mayor que 32,768, el entero con signo más grande que puede almacenarse en una palabra de 16 bits, fallando por ello la conversión.*



Figura 1.1 *Ariane 5*.

El objetivo de este capítulo es conocer y analizar errores del tipo mencionado y prevenirlos en alguna medida.

* Fuente: <http://ta.twi.tudelft.nl/users/vuik/wi211/disasters.html>

A dónde nos dirigimos

En este capítulo revisaremos tres de los sistemas numéricos posicionales más relevantes en el estudio de los métodos numéricos: binario, octal y decimal. Analizaremos las conversiones entre ellos, la representación y el manejo del sistema binario en la computadora, así como los diversos errores que ello puede ocasionar y algunas formas de evitarlos. Dada la naturaleza electrónica de las calculadoras y las computadoras, los sistemas binario y octal resultan los más indicados para usarse en estos aparatos; a fin de tener una idea de los procesos numéricos internos en ellas, conviene hacer un estudio de tales sistemas y su conversión al decimal, ya que éste es finalmente nuestro medio de enlace con las máquinas.

Por un lado, dada la finitud de la palabra de memoria de las máquinas, es imposible representar todos los números reales en ella. Así, números como π , $\sqrt{2}$, $1/3 = 0.333\dots$, números muy pequeños* (o muy grandes) se manejan usando números que son aproximaciones de ellos, o simplemente no se manejan. Por otro lado, una de las características más sobresalientes de los métodos numéricos es el uso de los números reales en cálculos extensos. Cabe entonces preguntarse qué efecto tienen tales aproximaciones en los cálculos que hacemos con dichos números, en los resultados que obtenemos e incluso qué números reales pueden representarse con exactitud en la computadora.

El conocimiento de todo esto nos ayudará a evitar cierto tipo de errores, analizar su propagación e, incluso, interpretar mejor los resultados dados por una máquina.

Introducción

En la antigüedad, los números naturales se representaban con distintos tipos de símbolos o **numerales**. A continuación se presentan algunas muestras de numerales primitivos (figuras 1.2 y 1.3.).

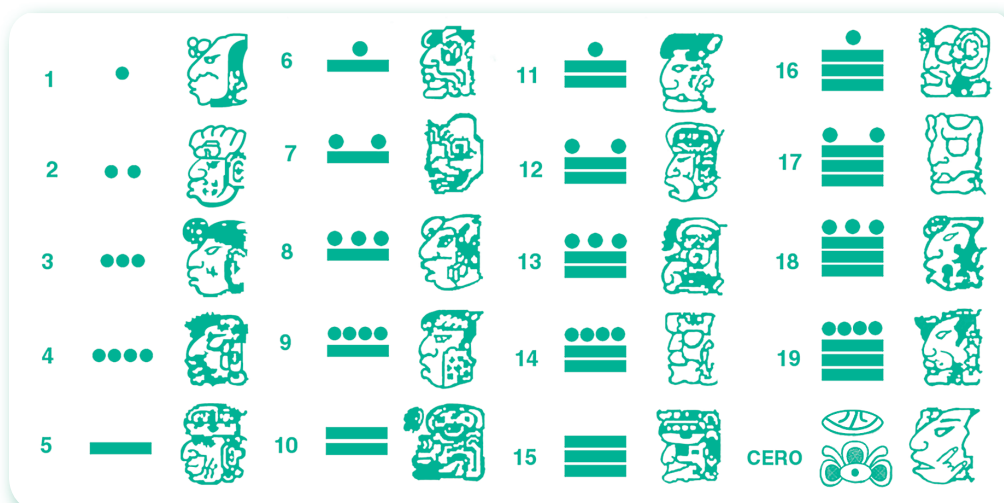


Figura 1.2 Numerales usados por los mayas.

* En estos casos la computadora envía un mensaje indicando que el número es muy pequeño (*underflow*) o muy grande (*overflow*) para su capacidad.

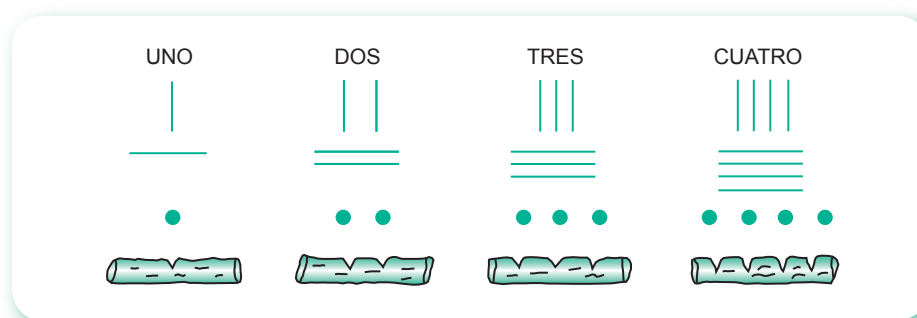


Figura 1.3 Numerales primitivos.

En la figura 1.3 se puede observar que cada numeral es un conjunto de marcas **sencillas** e iguales. ¡Imagínese si así se escribiera el número de páginas del directorio telefónico de la Ciudad de México! No sería práctico, por la enorme cantidad de tiempo y espacio que requeriría tal sucesión de marcas iguales. Más aún: nadie podría reconocer, a primera vista, el número representado. Por ejemplo, ¿podría identificar rápidamente el siguiente numeral?

||||| = ?

Los antiguos egipcios evitaron algunos de los inconvenientes de los numerales representados por medio de marcas iguales, usando un solo jeroglífico o figura. Por ejemplo, en lugar de |||||, usaron el símbolo , que representaba el hueso del talón. En la figura 1.4 se muestran otros numerales egipcios básicos relacionados con los del sistema decimal que les corresponden.

1	10	100	1 000	10 000	100 000	1 000 000
						
Raya	Hueso del talón	Cuerda enrollada	Flor de loto	Dedo señalando	Pez	Hombre sorprendido

Figura 1.4 Numerales egipcios antiguos.

1.1 Sistemas numéricos

Numeración con base dos (sistema binario)

Dado el siguiente conjunto de marcas simples e iguales:

|||||,

si se encierran en óvalos por parejas, a partir de la izquierda, se tiene



A continuación, también empezando por la izquierda, se encierra cada par de óvalos en otro mayor.



Finalmente, se encierra cada par de óvalos grandes en uno mayor todavía, comenzando también por la izquierda.



Nótese que el número de marcas dentro de cualquier óvalo es una potencia de 2.

El número representado por el numeral $\begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline \end{array}$ se

obtiene así $2^3 + 2^1 + 2^0$,

o también $(1 \times 2^3) + (1 \times 2^1) + (1 \times 2^0)$

Hay que observar que en esta suma no aparece 2^2 . Como $0 \times 2^2 = 0$, entonces la suma puede escribirse así:

$$(1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

Ahora puede formarse un nuevo símbolo para representar esta suma omitiendo los paréntesis, los signos de operación + y $\times$, y las potencias de 2, de la siguiente manera:

$$\begin{array}{cccc} (1 \times 2^3) & + & (0 \times 2^2) & + & (1 \times 2^1) & + & (1 \times 2^0) \\ \downarrow & & \downarrow & & \downarrow & & \downarrow \\ \text{Nuevo símbolo:} & 1 & 0 & 1 & 1 \end{array}$$

Ahora bien, ¿cómo interpretaremos este nuevo símbolo?

El significado de los números 1 en este nuevo símbolo depende del lugar que ocupan en el numeral. Así pues, el primero de derecha a izquierda representa una unidad; el segundo, un grupo de dos (o bien 2^1) y el cuarto, cuatro grupos de dos (8, o bien 2^3). El cero es el medio de asignarle a cada "1", su posición correcta. A los números o potencias de 2 que representan el "1", según su posición en el numeral, se les llama **valores de posición**; se dice que un sistema de numeración que emplea valores de posición es un **sistema posicional**.

El de este ejemplo es un **sistema de base dos**, o sistema binario, porque emplea un grupo básico de dos símbolos: 0 y 1. Los símbolos "1" y "0" utilizados para escribir los numerales se denominan **dígitos binarios** o **bits**.

¿Qué número representa el numeral 101010_{dos} ?

(Se lee: "uno, cero, uno, cero, uno, cero, base dos".)

Escríbanse los valores de posición debajo de los dígitos:

Dígitos binarios	1	0	1	0	1	0 _{dos}
Valores de posición	2^5	2^4	2^3	2^2	2^1	2^0

Al multiplicar los valores de posición por los dígitos binarios correspondientes y sumándolos todos, se obtiene el equivalente en decimal.

$$\begin{aligned} 101010_{\text{dos}} &= (1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) \\ &= 42_{\text{diez}} \text{ (se lee: "cuatro, dos, base diez").} \end{aligned}$$

El sistema de numeración más difundido en la actualidad es el **sistema decimal**, un sistema posicional que usa un grupo básico de diez símbolos (base diez).

Considérese por ejemplo el numeral 582_{diez}

Dígitos decimales	5	8	2
Valores de posición	10^2	10^1	10^0
Forma desarrollada	$(5 \times 10^2) + (8 \times 10^1) + (2 \times 10^0)$		

Al escribir números decimales se omite la palabra "diez" y se establece la convención de que un numeral con valor de posición es un número decimal, sin necesidad de indicar la base. De ahí que siempre se anote 582 en lugar de 582_{diez} .

El desarrollo y arraigo del sistema decimal quizá se deba al hecho de que siempre tenemos a la vista los diez dedos de las manos. El sistema binario se emplea en las computadoras digitales debido a que los alambres que forman los circuitos electrónicos presentan sólo dos estados: magnetizados o no magnetizados, ya sea que pase o no corriente por ellos.

Conversión de números enteros del sistema decimal a un sistema de base b y viceversa

Para convertir un número n del sistema decimal a un sistema con base b , se divide el número n entre la base b y se registran el cociente c_1 y el residuo r_1 resultantes; se divide c_1 entre la base b y se anotan el nuevo cociente c_2 y el nuevo residuo r_2 . Este procedimiento se repite hasta obtener un cociente c_i igual a cero con residuo r_i . El número equivalente a n en el sistema con base b queda formado así: $r_i r_{i-1} r_{i-2} \dots r_1$.

Ejemplo 1.1

Convierta el número 358_{10} al sistema octal.

Solución

La base del sistema octal* es 8, por tanto

$$\begin{array}{rclclcl} 358 & = & 8 & \times & 44 & + & 6 \\ & & & & c_1 & & r_1 \\ 44 & = & 8 & \times & 5 & + & 4 \\ & & & & c_2 & & r_2 \\ 5 & = & 8 & \times & 0 & + & 5 \\ & & & & c_3 & & r_3 \end{array}$$

Así que el número equivalente en el sistema octal es 546.

* El sistema octal usa un grupo básico de ocho símbolos: 0, 1, 2, 3, 4, 5, 6, 7.

Ejemplo 1.2

Convierta el número 358_{10} a binario (base 2).

Solución

358	=	2	×	179	+	0
179	=	2	×	89	+	1
89	=	2	×	44	+	1
44	=	2	×	22	+	0
22	=	2	×	11	+	0
11	=	2	×	5	+	1
5	=	2	×	2	+	1
2	=	2	×	1	+	0
1	=	2	×	0	+	1



Por tanto, $358_{10} = 101100110_2$

Para convertir un entero m de un sistema con base b al sistema decimal, se multiplica cada dígito de m por la base b elevada a una potencia igual a la posición del dígito, tomando como posición cero la del dígito situado más a la derecha. De la suma resulta el equivalente decimal. Así:

$$276_8 = 2 \times 8^2 + 7 \times 8^1 + 6 \times 8^0 = 190_{10}$$

$$1010001_2 = 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 81_{10}$$

Conversión de números enteros del sistema octal al binario y viceversa

Dado un número del sistema octal, su equivalente en binario se obtiene sustituyendo cada dígito del número octal con los tres dígitos equivalentes del sistema binario.

Base octal	Equivalente binario en tres dígitos
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Ejemplo 1.3

Convierta el número 546_8 a binario.

Solución

5	4	6
101	100	110

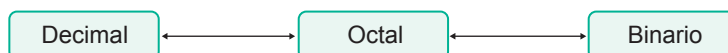
Así que, $546_8 = 101100110_2$

Dado un número en binario, su equivalente en octal se obtiene formando ternas de dígitos, contando de derecha a izquierda y sustituyendo cada terna por su equivalente en octal. Así:

Convertir 10011001_2 a octal

010	011	001 ₂	Por tanto, $10011001_2 = 231_8$
2	3	1	

Dado que la conversión de octal a binario es simple y la de decimal a binario resulta muy tediosa, se recomienda usar la conversión a octal como paso intermedio al convertir un número decimal a binario.



Las flechas señalan dos sentidos porque lo dicho es válido en ambas direcciones.

Ejemplo 1.4

Convierta 101100110_2 a decimal.

Solución

a) Conversión directa

$$101100110_2 = 1 \times 2^8 + 0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 358_{10}$$

b) Usando la conversión a octal como paso intermedio:

1) Conversión a octal	101	100	110
	5	4	6

Por tanto, $101100110_2 = 546_8$

2) Conversión de octal a decimal

$$546_8 = 5 \times 8^2 + 4 \times 8^1 + 6 \times 8^0 = 358_{10}$$

Conversión de números fraccionarios del sistema decimal a un sistema con base b

Para convertir un número x_{10} fraccionario a un número con base b , se multiplica dicho número por la base b ; el resultado tiene una parte entera e_1 y una parte fraccionaria f_1 . Ahora se multiplica f_1 por b y se obtiene un nuevo producto con parte entera e_2 y fraccionaria f_2 . Este procedimiento se repite indefinidamente o hasta que se presenta $f_i = 0$. El equivalente de x_{10} con base b queda así: $0. e_1 e_2 e_3 e_4 \dots$

Ejemplo 1.5

Convierta 0.2_{10} a octal y binario.

Solución

a) Conversión a octal

0.2	0.6	0.8	0.4	0.2
$\times 8$	$\times 8$	$\times 8$	$\times 8$	$\times 8$
1.6	4.8	6.4	3.2	1.6
$e_1 f_1$	$e_2 f_2$	$e_3 f_3$	$e_4 f_4$	$e_5 f_5$

Después de e_4 se repetirá la secuencia e_1, e_2, e_3, e_4 indefinidamente, de modo que $0.2_{10} = 0.14631463\dots_8$

b) Conversión a binario

0.2	0.4	0.8	0.6	0.2
$\times 2$	$\times 2$	$\times 2$	$\times 2$	$\times 2$
0.4	0.8	1.6	1.2	0.4
$e_1 f_1$	$e_2 f_2$	$e_3 f_3$	$e_4 f_4$	$e_5 f_5$

Igual que en el inciso a), después de e_4 se repite la secuencia e_1, e_2, e_3, e_4 indefinidamente, por lo que $0.2_{10} = 0.001100110011\dots_2$

Obsérvese que 0.2_{10} pudo convertirse en binario simplemente tomando su equivalente en octal y sustituyendo cada número por su terna equivalente en binario. Así:

$0.2_{10} =$	0.1	4	6	3	1	4	6	3
	0.001	100	110	011	001	100	110	011

y

$$0.2_{10} = 0.001100110011001100110011\dots_2$$

De lo anterior se puede observar que

$$358.2_{10} = 101100110.001100110011001100110011\dots_2$$

y cualquier número con parte entera y fraccionaria puede pasarse a otro sistema, cambiando su parte entera y fraccionaria de manera independiente, y al final concatenarlos.

Para convertir números decimales enteros y fraccionarios a base 2, 3, ..., 9 puede usar el **PROGRAMA 1.4** del CD-ROM.

Conversión de un número fraccionario en sistema binario a sistema decimal

El procedimiento es similar al que se aplica en el caso de números enteros, sólo hay que tomar en cuenta que la posición inicia con -1 , a partir del punto.

Ejemplo 1.6

Convierta 0.010101110_2 a octal y decimal.

Solución

a) Conversión a octal

$$\begin{array}{ccc} 0.010 & 101 & 110 \\ 2 & 5 & 6 \end{array}$$

y

$$0.010101110_2 = 0.256_8$$

b) Conversión a decimal

$$0.256_8 = 2 \times 8^{-1} + 5 \times 8^{-2} + 6 \times 8^{-3} = 0.33984375_{10}$$

Ejemplo 1.7

Convierta 0.010101110_2 a decimal.

Solución

$$\begin{aligned} 0.010101110 &= 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} \\ &+ 0 \times 2^{-5} + 1 \times 2^{-6} + 1 \times 2^{-7} + 1 \times 2^{-8} + 0 \times 2^{-9} \\ &= 0.33984375_{10} \end{aligned}$$

1.2 Manejo de números en la computadora

Por razones prácticas, en una computadora sólo puede manejarse una cantidad finita de bits para cada número; esta cantidad o longitud varía de una máquina a otra. Por ejemplo, cuando se realizan cálculos de ingeniería y ciencias, es mejor trabajar con una longitud grande; por otro lado, una longitud pequeña es más económica y útil para cálculos y procesamiento administrativos.

Para una computadora dada, el número de bits generalmente se llama *palabra*. Las palabras van desde ocho hasta 64 bits y, para facilitar su manejo, cada una se divide en partes más cortas denominadas bytes; por ejemplo, una palabra de 32 bits puede dividirse en cuatro bytes (ocho bits cada uno).



Figura 1.5 Una computadora sólo puede manejar una cantidad finita de bits.

Números enteros

Cada palabra, cualquiera que sea su longitud, almacena un número, aunque en ciertas circunstancias se usan varias palabras para contener un número. Por ejemplo, considérese una palabra de 16 bits para almacenar números enteros. De los 16 bits, el primero representa el signo del número; un cero es un signo más y un uno es un signo menos. Los 15 bits restantes pueden usarse para guardar números binarios desde 000000000000000 hasta 111111111111111 (véase figura 1.6). Al convertir este número en decimal se obtiene:

$$(1 \times 2^{14}) + (1 \times 2^{13}) + (1 \times 2^{12}) + \dots + (1 \times 2^1) + (1 \times 2^0)$$

que es igual a $2^{15} - 1 = 32767$. Por lo tanto, cada palabra de 16 bits puede contener un número cualquiera del intervalo -32768 a $+32767$ (véase el problema 1.10).

Números reales (punto flotante)

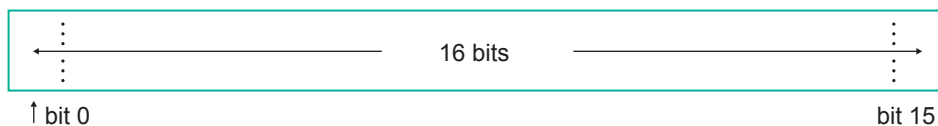


Figura 1.6 Esquema de una palabra de 16 bits para un número entero.

Ejemplo 1.8

Represente el número 525_{10} en una palabra de 16 bits.

Solución

$525_{10} = 1015_8 = 1000001101_2$, y su almacenamiento quedaría de la siguiente forma:

0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Ejemplo 1.9

Represente el número -26 en una palabra de 16 bits.

Solución

$-26_{10} = -11010_2$ y su almacenamiento en una palabra de 16 bits quedaría así:

1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Cuando se desea almacenar un número real, se emplea en su representación binaria, llamada de punto flotante, la notación

$$0.d_1d_2d_3d_4d_5d_6d_7d_8 \times 2^{d'_1d'_2d'_3d'_4d'_5d'_6d'_7}$$

donde $d_1 = 1$ y d_i y d'_j con $i = 2, \dots, 8$ y $j = 1, 2, \dots, 7$ pueden ser ceros o unos y se guarda en una palabra, como se muestra en la figura 1.7.

Igual que antes, el bit cero se usa para guardar el signo del número. En los bits del uno al siete se almacena el exponente de la base 2 y en los ocho restantes la fracción.* Según el lenguaje de los logaritmos,

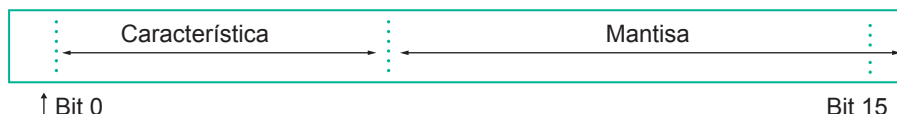


Figura 1.7 Esquema de una palabra de 16 bits para un número de punto flotante.

la fracción es llamada **mantisa** y el exponente **característica**. El número mayor que puede guardarse en una palabra de 16 bits, usando la notación de punto flotante, es

$$\begin{array}{ccc} & \text{Exponente} & \\ & \text{positivo} & \\ & \uparrow & \\ 0 & 011111 & 111111 \\ \uparrow & & \uparrow \\ \text{más} & 2^{63} & \text{Equivalente a } 0.99 \text{ en decimal} \end{array}$$

y los números que se pueden guardar en punto flotante binario van de alrededor de 2^{-64} (si la característica es negativa) a cerca de 2^{63} ; en decimal, de 10^{-19} a cerca de 10^{18} en magnitud (incluyendo números positivos, negativos y el cero).

Nótese que primero se normaliza el número, después se almacenan los primeros ocho bits y se truncan los restantes.

Ejemplo 1.10

El número decimal -125.32 que en binario es

$$-1111101.010100011110101,$$

Solución

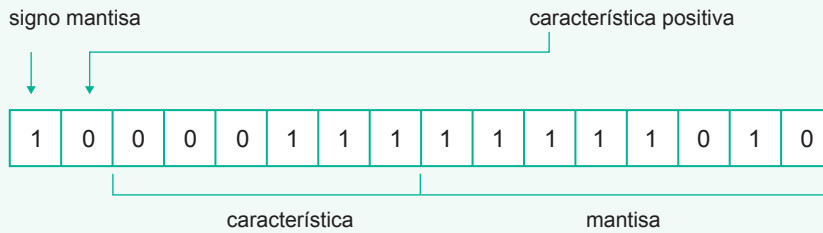
Normalizado queda así:

$$-.11111010100011110101 \times 2^{+111}$$

bits truncados en el almacenamiento

* El exponente es un número binario de seis dígitos, ya que el bit uno se emplea para su signo. En algunas computadoras el exponente se almacena en base ocho (octal) o base 16 (hexadecimal), en lugar de base 2.

y la palabra de memoria de 16 bits donde se almacena este valor quedaría como



El número decimal + 0.2, que en binario es

$$0.0011001100110011\dots$$

y que normalizado queda

$$.1100110011001100\dots \times 2^{-10}$$

bits truncados

se almacena así:



Doble precisión

La doble precisión es un esfuerzo para aumentar la exactitud de los cálculos adicionando más bits a la mantisa. Esto se hace utilizando dos palabras, la primera en la forma expuesta con anterioridad y los bits de la segunda para aumentar la mantisa de la primera. Entonces, con una palabra de 16 bits puede usarse en doble precisión una mantisa de $8 + 16 = 24$ bits, los cuales permiten expresar alrededor de 7 dígitos de exactitud en un número decimal, en lugar de 3 de la precisión sencilla.

La desventaja del uso de la doble precisión es que se emplean más palabras, con lo que aumenta el uso de memoria por un programa.

Error de redondeo

Para finalizar esta sección, se analizarán brevemente algunas consecuencias de utilizar el sistema binario y una longitud de palabra finita.

Como no es posible guardar un número binario de longitud infinita o un número de más dígitos de los que posee la mantisa de la computadora que se está empleando, se almacena sólo un número finito de estos dígitos; la consecuencia es que comete automáticamente un pequeño error, conocido como error de redondeo, que al repetirse muchas veces puede llegar a ser considerable. Por ejemplo, si se desea guardar la fracción decimal 0.0001, que en binario es la fracción infinita

$$0.0000000000000011010001101101110001011101011000\dots$$

quedaría, después de normalizarse, almacenada en una palabra de 16 bits como

$$.11010001 \times 2^{-1101}$$

Si se desea sumar el número 0.0001 con él mismo diez mil veces, usando una computadora, naturalmente que no se esperará obtener 1 como resultado, ya que los números que se adicionen no serán realmente 0.0001, sino valores aproximados a él (véase el problema 1.16).

1.3 Errores

Error absoluto, error relativo y error en por ciento

Si p^* es una aproximación a p , el error se define como

$$E = p^* - p$$

Sin embargo, para facilitar el manejo y el análisis se emplea el error absoluto definido como

$$EA = |p^* - p|$$

y el error relativo como

$$ER = \frac{|p^* - p|}{p}, \text{ si } p \neq 0$$

y como por ciento de error a

$$ERP = ER \times 100$$

En otros libros, las definiciones pueden ser diferentes; por ejemplo, algunos autores definen el error E como $p - p^*$; por tanto, sugerimos que al consultar distintas bibliografías se busquen las definiciones de error dadas.

Ejemplo 1.11

Suponga que el valor para un cálculo debiera ser

$p = 0.10 \times 10^2$, pero se obtuvo el resultado $p^* = 0.08 \times 10^2$, entonces

$$EA = |0.08 \times 10^2 - 0.10 \times 10^2| = 0.2 \times 10^1$$

$$ER = \frac{|0.08 \times 10^2 - 0.10 \times 10^2|}{0.10 \times 10^2} = 0.2 \times 10^0$$

$$ERP = ER \times 100 = 20\%$$

Por lo general, interesa el error absoluto y no el error relativo; pero, cuando el valor exacto de una cantidad es “muy pequeño” o “muy grande”, los errores relativos son más significativos.

Por ejemplo, si

$$p = 0.24 \times 10^{-4} \text{ y } p^* = 0.12 \times 10^{-4},$$

entonces:

$$EA = |0.12 \times 10^{-4} - 0.24 \times 10^{-4}| = 0.12 \times 10^{-4}$$

Sin reparar en las cantidades que se comparan, puede pensarse que el error absoluto es muy pequeño y, lo más grave, aceptar p^* como una buena aproximación a p .

Si, por otro lado, se calcula el error relativo

$$ER = \frac{|0.12 \times 10^{-4} - 0.24 \times 10^{-4}|}{0.24 \times 10^{-4}} = 0.5 \times 10^0$$

se observa que la “aproximación” es tan sólo la mitad del valor verdadero y, por lo tanto, está muy lejos de ser aceptable como aproximación a p . Finalmente,

$$ERP = 50\%$$

De igual manera, puede observarse que si

$$p = 0.46826564 \times 10^6 \quad \text{y} \quad p^* = 0.46830000 \times 10^6,$$

entonces:

$$EA = 0.3436 \times 10^2,$$

y si de nueva cuenta dejan de considerarse las cantidades en cuestión, puede creerse que el EA es muy grande y que se tiene una mala aproximación a p . Sin embargo, al calcular el error relativo

$$ER = 0.7337715404 \times 10^{-4},$$

se advierte que el error es muy pequeño, como en realidad ocurre.

Advertencia:

Cuando se manejan cantidades “muy grandes” o “muy pequeñas”, el error absoluto puede ser engañoso, mientras que el error relativo es más significativo en esos casos.

Definición

Se dice que el número p^* aproxima a p con t dígitos significativos si t es el entero más grande no negativo, para el cual se cumple

$$\frac{|p^* - p|}{p} < 5 \times 10^{-t}$$

Supóngase, por ejemplo, el número 10. Para que p^* aproxime a 10 con dos cifras significativas, usando la definición, p^* debe cumplir con

$$\frac{|p^* - 10|}{10} < 5 \times 10^{-2}$$

$$-5 \times 10^{-2} < \frac{p^* - 10}{10} < 5 \times 10^{-2}$$

$$10 - 5 \times 10^{-1} < p^* < 5 \times 10^{-1} + 10$$

$$9.5 < p^* < 10.5$$

esto es, cualquier valor de p^* en el intervalo (9.5, 10.5) cumple la condición.

En general, para t dígitos significativos

$$\frac{|p^* - p|}{p} < 5 \times 10^{-t} \quad \text{si } p > 0$$

$$p - 5 p \times 10^{-t} < p^* < p + 5 p \times 10^{-t}$$

Si, por ejemplo, $p = 1000$ y $t = 4$

$$1000 - 5 \times 1000 \times 10^{-4} < p^* < 1000 + 5 \times 1000 \times 10^{-4}$$

$$999.5 < p^* < 1000.5$$

Causas de errores graves en computación

Existen muchas causas de errores en la ejecución de un programa de cómputo, ahora discutiremos algunas de las más serias. Para esto, pensemos en una computadora imaginaria que trabaja con números en el sistema decimal, de tal forma que tiene una mantisa de cuatro dígitos decimales y una característica de dos dígitos decimales, el primero de los cuales es usado para el signo. Sumados estos seis al bit empleado para el signo del número, se tendrá una palabra de siete bits. Los números que se van a guardar deben normalizarse primero en la siguiente forma:

$$3.0 = .3000 \times 10^1$$

$$7956000 = .7956 \times 10^7$$

$$-0.0000025211 = -.2521 \times 10^{-5}$$

Empleando esta computadora imaginaria, podemos estudiar algunos de los errores más serios que se cometen en su empleo.

a) Suma de números muy distintos en magnitud

Vamos a suponer que se trata de sumar 0.002 a 600 en la computadora decimal imaginaria.

$$0.002 = .2000 \times 10^{-2}$$

$$600 = .6000 \times 10^3$$

Estos números *normalizados* no pueden sumarse directamente y, por lo tanto, la computadora debe desnormalizarlos antes de efectuar la suma.

$$\begin{array}{r} .000002 \times 10^3 \\ + .600000 \times 10^3 \\ \hline .600002 \times 10^3 \end{array}$$

Como sólo puede manejar cuatro dígitos, son eliminados los últimos dos y la respuesta es $.6000 \times 10^3$ o 600. Según el resultado, la suma nunca se realizó.

Este tipo de errores, cuyo origen es el redondeo, es muy común y se recomienda, de ser posible, no sumar o restar dos números muy diferentes (véase el ejercicio 1.12).

b) Resta de números casi iguales

Supóngase que la computadora decimal va a restar 0.2144 de 0.2145.

$$\begin{array}{r} .2145 \times 10^0 \\ - .2144 \times 10^0 \\ \hline .0001 \times 10^0 \end{array}$$

Como la mantisa de la respuesta está desnormalizada, la computadora automáticamente la normaliza y el resultado se almacena como $.1000 \times 10^{-3}$.

Hasta aquí no hay error, pero en la respuesta sólo hay un dígito significativo; por lo tanto, se sugiere no confiar en su exactitud, ya que un pequeño error en alguno de los números originales produciría un error relativo muy grande en la respuesta de un problema que involucrara este error, como se ve a continuación.

Supóngase que la siguiente expresión aritmética es parte de un programa:

$$X = (A - B) * C$$

Considérese ahora que los valores de A , B y C son

$$A = 0.2145 \times 10^0, \quad B = 0.2144 \times 10^0, \quad C = 0.1000 \times 10^5$$

Al efectuarse la operación se obtiene el valor de $X = 1$, que es correcto. Sin embargo, supóngase que A fue calculada en el programa con un valor de 0.2146×10^0 (error absoluto 0.0001, error relativo 0.00046 y $ERP = 0.046\%$). Usando este valor de A en el cálculo de X , se obtiene como respuesta $X = 2$. Un error de 0.046% de pronto provoca un error de 100% y, aún más, este error puede pasar desapercibido.

c) Overflow y Underflow

Con frecuencia una operación aritmética con dos números válidos da como resultado un número tan grande o tan pequeño que la computadora no puede representarlo; la consecuencia es que se produce un *overflow* o un *underflow*, respectivamente.

Por ejemplo, al multiplicar 0.5000×10^8 por 0.2000×10^9 , se tiene

$$\begin{array}{r} \times 0.5000 \times 10^8 \\ 0.2000 \times 10^9 \\ \hline 0.1000 \times 10^{17} \end{array}$$

Cada uno de los números que se multiplican puede guardarse en la palabra de la computadora imaginaria; sin embargo, su producto es muy grande y no puede almacenarse porque la característica requiere tres dígitos. Entonces se dice que ha sucedido un *overflow*.

Otro caso de *overflow* puede ocurrir en la división; por ejemplo

$$\frac{2000000}{0.000005} = \frac{0.2000 \times 10^7}{0.5000 \times 10^{-5}} = 0.4000 \times 10^{12}$$

Generalmente, las computadoras reportan esta circunstancia con un mensaje que varía dependiendo de cada máquina.

El *underflow* puede aparecer en la multiplicación o división, y por lo general no es tan serio como el *overflow*; las computadoras casi nunca envían mensaje de *underflow*. Por ejemplo:

$$(0.3000 \times 10^{-5}) \times (0.02000 \times 10^{-3}) = 0.006 \times 10^{-8} = 0.6000 \times 10^{-10}$$

Como el exponente -10 está excedido en un dígito, no puede guardarse en la computadora y este resultado se expresa como valor cero. Dicho error expresado como error relativo es muy pequeño, y a menudo no es serio. No obstante, puede ocurrir, por ejemplo:

$$A = 0.3000 \times 10^{-5}, \quad B = 0.0200 \times 10^{-3}, \quad C = 0.4000 \times 10^7,$$

y que se desee en algún punto del programa calcular el producto de A , B y C .

$$X = A * B * C$$

Se multiplican primero A y B . El resultado parcial es cero. La multiplicación de este resultado por C da también cero. Si, en cambio, se arregla la expresión como

$$X = A * C * B$$

se multiplica A por C y se obtiene 0.1200×10^2 . La multiplicación siguiente da la respuesta correcta: 0.2400×10^{-3} . De igual manera, un arreglo en una división puede evitar el *underflow*.

d) División entre un número muy pequeño

Como se dijo, la división entre un número muy pequeño puede causar un *overflow*.

Supóngase que se realiza en la computadora una división válida y que no se comete error alguno en la operación; pero considérese que previamente ocurrió un pequeño error de redondeo en el programa, cuando se calculó el denominador. Si el numerador es grande y el denominador pequeño, puede presentarse en el cociente un error absoluto considerable. Si éste se resta después de otro número del mismo tamaño relativo, puede presentarse un error mayor en la respuesta final.

Como ejemplo, considérese la siguiente instrucción en un programa

$$X = A - B / C$$

donde:

$$A = 0.1120 \times 10^9 = 112000000$$

$$B = 0.1000 \times 10^6 = 100000$$

$$C = 0.900 \times 10^{-3} = 0.0009$$

Si el cálculo se realiza en la computadora decimal de cuatro dígitos, el cociente B / C es 0.1111×10^9 , y X es 0.0009×10^9 o, después de ser normalizado, $X = 0.9000 \times 10^6$. Nótese que sólo hay un dígito significativo.

Vamos a imaginar ahora que se cometió un pequeño error de redondeo al calcular C en algún paso previo y resultó un valor $C^* = 0.9001 \times 10^{-3}$ ($EA = 0.0001 \times 10^{-3}$; $ER = 10^{-4}$ y $ERP = 0.01\%$).

Si se calcula B / C^* se obtiene como cociente 0.1110×10^9 y $X^* = 0.1000 \times 10^7$. El valor correcto de X es 0.9000×10^6 .

Entonces:

$$EA = |1000000 - 900000| = 100000$$

$$ER = \frac{|1000000 - 900000|}{900000} = 0.11$$

$$ERP = 0.11 \times 100 = 11\%$$

El error relativo se ha multiplicado cerca de 1100 veces. Como ya se dijo, estos cálculos pueden conducir a un resultado final carente de significado o sin relación con la respuesta verdadera.

e) Error de discretización

Dado que un número específico no se puede almacenar exactamente como número binario de punto flotante, el error generado se conoce como error de discretización (error de cuantificación), ya que los números expresados exactamente por la máquina (números máquina) no forman un conjunto continuo sino discreto.

Ejemplo 1.12

Cuando se suma 10000 veces 0.0001 con él mismo, debe resultar 1; sin embargo, el número 0.0001 en binario resulta en una sucesión infinita de ceros y unos, que se trunca al ser almacenada en una palabra de memoria. Con esto se perderá información, y el resultado de la suma ya no será 1. Se obtuvieron los siguientes resultados que corroboran lo anterior, utilizando una PC, precisión sencilla y Visual Basic.

Solución

$$a) \sum_{i=1}^{10000} 0.0001 = 1.000054$$

$$b) 1 + \sum_{i=1}^{10000} 0.0001 = 2.000166$$

$$c) 1000 + \sum_{i=1}^{10000} 0.0001 = 1001.221$$

$$d) 10000 + \sum_{i=1}^{10000} 0.0001 = 10000$$

Nótese que en los tres últimos incisos, además del error de discretización, se generó el error de sumar un número muy grande con un número muy pequeño (véanse los problemas 1.16 y 1.17). El programa se ejecutó iniciando primero a una variable con el valor entero 0, 1, 1000 y 10000; después se fue acumulando a esa variable 0.0001 diez mil veces.

f) Errores de salida

Aun cuando no se haya cometido error alguno durante la fase de cálculos de un programa, puede presentarse un error al imprimir resultados.

Por ejemplo, supóngase que la respuesta de un cálculo particular es exactamente 0.015625. Cuando este número se imprime con un formato tal como F10.6 o E14.6 (de FORTRAN), se obtiene la respuesta correcta. Si, por el contrario, se decide usar F8.3, se imprimirá el número 0.016 (si la computadora redondea), o bien 0.015 (si la computadora trunca), con lo cual se presenta un error.

Propagación de errores

Una vez que sabemos cómo se producen los errores en un programa de cómputo, podríamos pensar en tratar de determinar el error cometido en cada paso y conocer, de esa manera, el error total en la respuesta final. Sin embargo, esto no es práctico. Resulta más adecuado analizar las operaciones individuales realizadas por la computadora para ver cómo se propagan los errores de dichas operaciones.

a) Suma

Se espera que al sumar a y b , se obtenga el valor correcto de $c = a + b$; no obstante, se tiene en general un valor de c incorrecto debido a la longitud finita de palabra que se emplea. Puede considerarse que este error fue causado por una operación incorrecta de la computadora $\dot{+}$ (el punto indica que es suma con error). Entonces el error es:

$$\text{Error} = (a \dot{+} b) - (a + b)$$

La magnitud de este error depende de las magnitudes relativas, de los signos de a y b , y de la forma binaria en que a y b son almacenados en la computadora. Esto último varía dependiendo de la computadora. Por tanto, es un error muy difícil de analizar y no lo estudiaremos aquí.

Si por otro lado, de entrada a y b son inexactos, hay un segundo error posible. Por ejemplo, considérese que en lugar del valor verdadero de a , la computadora tiene el valor a^* , el cual presenta un error $\in_a$

$$a^* = a + \in_a$$

y de igual forma para b

$$b^* = b + \in_b$$

Como consecuencia de ello se tendría, incluso si no se cometiera error en la adición, un error en el resultado:

$$\begin{aligned} \text{Error} &= (a^* + b^*) - (a + b) \\ &= (a + \in_a + b + \in_b) - (a + b) = \in_a + \in_b = \in_c \end{aligned}$$

o sea $c^* = c + \in_c$

El error absoluto es:

$$| (a^* + b^*) - (a + b) | = | \in_a + \in_b | \leq | \in_a | + | \in_b |$$

o bien

$$| \in_c | \leq | \in_a | + | \in_b |$$

Se dice que los errores $\in_a$ y $\in_b$ se han propagado a c , y $\in_c$ se conoce como el error de propagación.

Dicho error es causado por valores inexactos de los valores iniciales y se propaga en los cálculos siguientes, con lo cual causa un error en el resultado final.

b) Resta

El error de propagación ocasionado por valores inexactos iniciales a^* y b^* , puede darse en la sustracción de manera similar que en la adición, con un simple cambio de signo (véase el problema 1.24).

c) Multiplicación

Si se multiplican los números a^* y b^* (ignorando el error causado por la operación misma), se obtiene:

$$\begin{aligned}(a^* \times b^*) &= (a + \epsilon_a) \times (b + \epsilon_b) \\ &= (a \times b) + (a \times \epsilon_b) + (b \times \epsilon_a) + (\epsilon_a \times \epsilon_b)\end{aligned}$$

Si ϵ_a y ϵ_b son suficientemente pequeños, puede considerarse que su producto es muy pequeño en comparación con los otros términos y, por lo tanto, despreciar el último término. Se obtiene, entonces, el error del resultado final:

$$(a^* \times b^*) - (a \times b) \approx (a \times \epsilon_b) + (b \times \epsilon_a)$$

Esto hace posible encontrar el valor absoluto del error relativo del resultado, dividiendo ambos lados entre $a \times b$.

$$\left| \frac{(a^* \times b^*) - (a \times b)}{(a \times b)} \right| \approx \left| \frac{\epsilon_b}{b} + \frac{\epsilon_a}{a} \right| \leq \left| \frac{\epsilon_b}{b} \right| + \left| \frac{\epsilon_a}{a} \right|$$

El error de propagación relativo en valor absoluto en la multiplicación es aproximadamente igual o menor a la suma de los errores relativos de a y b en valor absoluto.

Ejemplo 1.13

En los cursos tradicionales de álgebra lineal se enseña que al multiplicar una matriz por su inversa, se obtiene la matriz identidad (una matriz con unos en la diagonal principal y ceros como los demás elementos).

Solución

No obstante, al realizar este cálculo con un software matemático, éste cometerá pequeños errores, como puede verse en este caso.

Utilizando el programa Matlab generamos una matriz cuadrada con números aleatorios¹ con la instrucción

A = rand (3) y obtenemos:

A =	0.84622141782432	0.67213746847429	0.68127716128214
	0.52515249630517	0.83811844505239	0.37948101802800
	0.20264735765039	0.01963951386482	0.83179601760961

Si ahora invertimos esta matriz con **B = inv (A)**, resulta:

B =	2.95962951001411	-2.34173605180686	-1.35572133824039
	-1.54449898903352	2.42808986631982	0.15727157830512
	-0.68457636062692	0.51317884382928	1.52879381800410

Finalmente, al multiplicar ambas matrices con **A*B**, obtenemos:

ans =	1.000000000000000e+000	-2.220446049250313e-016	-2.220446049250313e-016
	-3.330669073875470e-016	1.000000000000000e+000	-1.110223024625157e-016
	0	-1.110223024625157e-016	1.000000000000000e+000

¹ Cada vez que se ejecute esta instrucción, el generador de números aleatorios proporcionará diferentes valores numéricos.

Moraleja: Siempre debemos tener precaución con los resultados que arroja una computadora.

El lector puede usar algún otro software matemático o la calculadora de que disponga y comparar.

d) División

Puede considerarse la división de a^* y b^* como sigue:

$$\begin{aligned}\frac{a^*}{b^*} &= \frac{(a + \epsilon_a)}{(b + \epsilon_b)} \\ &= (a + \epsilon_a) \frac{1}{(b + \epsilon_b)}\end{aligned}$$

Multiplicando numerador y denominador por $b - \epsilon_b$

$$\begin{aligned}\frac{a^*}{b^*} &= \frac{(a + \epsilon_a)(b - \epsilon_b)}{(b + \epsilon_b)(b - \epsilon_b)} \\ &= \frac{ab - a\epsilon_b + \epsilon_a b - \epsilon_a \epsilon_b}{b^2 - \epsilon_b^2}\end{aligned}$$

Si, como en la multiplicación, se considera el producto $\epsilon_a \epsilon_b$ muy pequeño y, por las mismas razones, $a \epsilon_b^2$ y se desprecian, se tiene:

$$\begin{aligned}\frac{a^*}{b^*} &\approx \frac{ab}{b^2} + \frac{\epsilon_a b}{b^2} - \frac{a\epsilon_b}{b^2} \\ &\approx \frac{a}{b} + \frac{\epsilon_a}{b} - \frac{a\epsilon_b}{b^2}\end{aligned}$$

El error es entonces:

$$\frac{a^*}{b^*} - \frac{a}{b} \approx \frac{\epsilon_a}{b} - \frac{a\epsilon_b}{b^2}$$

Dividiendo entre a/b se obtiene el error relativo. Al tomar el valor absoluto del error relativo, se tiene:

$$\left| \frac{\frac{a^*}{b^*} - \frac{a}{b}}{\frac{a}{b}} \right| \approx \left| \frac{\frac{\epsilon_a}{b} - \frac{a\epsilon_b}{b^2}}{\frac{a}{b}} \right| \approx \left| \frac{\epsilon_a}{a} - \frac{\epsilon_b}{b} \right| \leq \left| \frac{\epsilon_a}{a} \right| + \left| \frac{\epsilon_b}{b} \right|$$

Se concluye que el error de propagación relativo del cociente en valor absoluto es aproximadamente igual o menor a la suma de los errores relativos en valor absoluto de a y b .

e) Evaluación de funciones

Por último, se estudiará la propagación del error (asumiendo operaciones básicas $+$, $-$, $\times$ y $/$, ideales o sin errores), cuando se evalúa una función $f(x)$ en un punto $x = a$. En general, se dispone de un valor de a aproximado: a^* ; la intención es determinar el error resultante:

$$\epsilon_f = f(a^*) - f(a)$$

La figura 1.6 muestra la gráfica de la función $f(x)$ en las cercanías de $x = a$. A continuación se determina la relación entre ϵ_a y ϵ_f .

Si ϵ_a es pequeño, puede aproximarse la curva $f(x)$ por su tangente un entorno de $x = a$. Se sabe que la pendiente de esta tangente es $f'(a)$ o aproximadamente ϵ_f/ϵ_a ; esto es:

$$\frac{\epsilon_f}{\epsilon_a} \approx f'(a)$$

y

$$\epsilon_f \approx \epsilon_a f'(a) \approx \epsilon_a f'(a^*)$$

En valor absoluto:

$$|\epsilon_f| \approx |\epsilon_a f'(a^*)| \approx |\epsilon_a| |f'(a^*)|$$

El error al evaluar una función en un argumento inexacto es proporcional a la primera derivada de la función en el punto donde se ha evaluado.

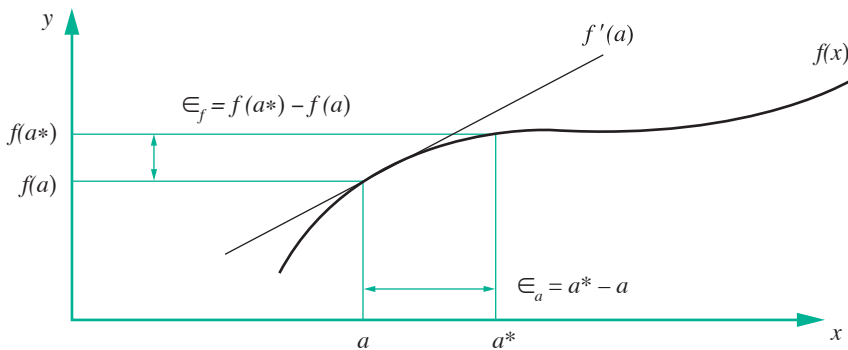


Figura 1.8 Gráfica de una función y su primera derivada en a .

1.4 Algoritmos y estabilidad

El tema fundamental de este libro es el estudio, selección y aplicación de algoritmos, que se definen como secuencias de operaciones algebraicas y lógicas para obtener la solución de un problema. Por lo general, se dispone de varios algoritmos para resolver un problema particular; uno de los criterios de selección es la estabilidad del algoritmo; esto es, que a pequeños errores de los valores manejados se obtengan pequeños errores en los resultados finales.

Supóngase que un error ϵ se introduce en algún paso en los cálculos y que el error de propagación de n operaciones subsiguientes se denota por ϵ_n . En la práctica, por lo general, se presentan dos casos.

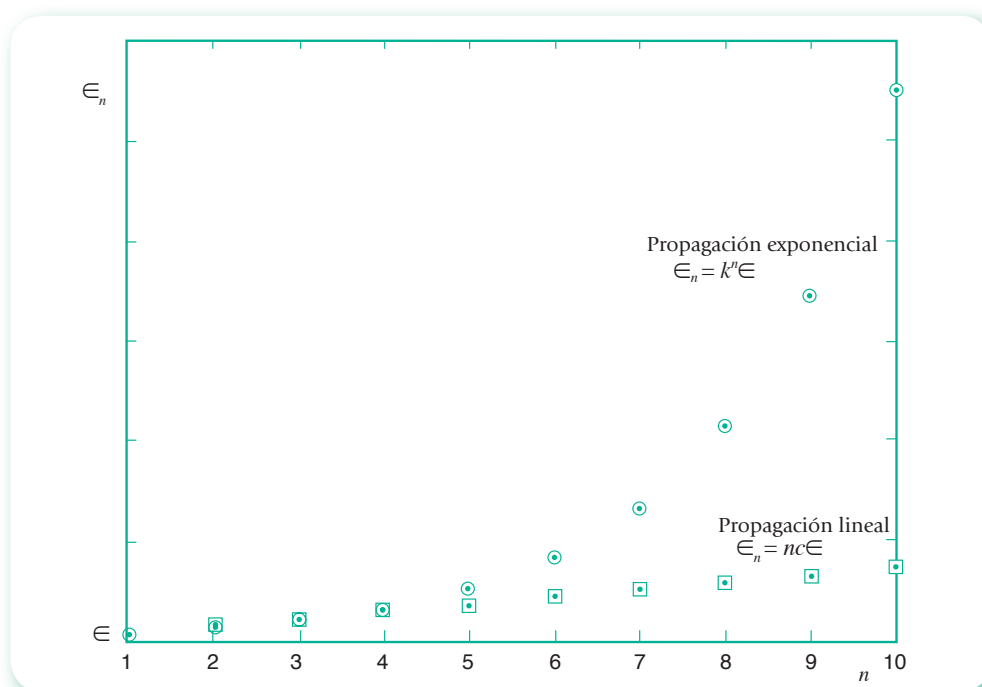


Figura 1.9 Propagación lineal y propagación exponencial de errores.

- a) $|\epsilon_n| \approx n c \epsilon$, donde c es una constante independiente de n ; se dice entonces que la propagación del error es lineal.
- b) $|\epsilon_n| \approx k^n \epsilon$, para $k > 1$; se dice entonces que la propagación del error es exponencial.

La propagación lineal de los errores suele ser inevitable; cuando c y ϵ son pequeños, los resultados finales normalmente son aceptables. Por otro lado, debe evitarse la propagación exponencial, ya que el término k^n crece con rapidez para valores relativamente pequeños de n . Esto conduce a resultados finales muy poco exactos, sea cual sea el tamaño de ϵ . Como consecuencia, se dice que un algoritmo con crecimiento lineal del error es estable, mientras que uno con una propagación exponencial es inestable (véase la figura 1.9).

Ejercicios

1.1 Error de redondeo al restar dos números casi iguales.

Vamos a considerar las ecuaciones



$$31.69 x + 14.31 y = 45.00 \quad (1)$$

$$13.05 x + 5.89 y = 18.53 \quad (2)$$

La única solución de este sistema de ecuaciones es (redondeando a cinco cifras decimales) $x = 1.25055$, $y = 0.37527$. Un método para resolver este tipo de problemas es multiplicar la ecuación (1) por el coeficiente de x de la ecuación (2), multiplicar la ecuación (2) por el coeficiente de x de la ecuación (1) y después restar

las ecuaciones resultantes. Para este sistema se obtendría (como los coeficientes tienen dos cifras decimales, todas las operaciones intermedias se efectúan redondeando a dos cifras decimales):

$$\begin{aligned} [13.05 (14.31) - 31.69 (5.89)] y &= 13.05 (45.00) - 31.69 (18.53) \\ (186.75 - 186.65) y &= 587.25 - 587.22 \\ 0.10 y &= 0.03 \end{aligned}$$

de donde $y = 0.3$, luego

$$x = \frac{(18.53) - 5.89 (0.3)}{13.05} = \frac{18.53 - 1.77}{13.05} = \frac{16.76}{13.05} = 1.28$$

Para la variable x

$$EA = |1.28 - 1.25| = 0.03; \quad ER = 0.03/1.25 = 0.024; \quad ERP = 2.4\%$$

Para la variable y

$$EA = |0.3 - 0.38| = 0.08; \quad ER = 0.08/0.38 = 0.21; \quad ERP = 21\%$$

1.2 Error de redondeo al sumar un número grande y uno pequeño.

Considere la sumatoria infinita

$$s = \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{1}{1} + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \frac{1}{25} + \dots + \frac{1}{100} + \dots$$

resulta (usando precisión sencilla y 5000 como valor final de n) 1.644725 si se suma de izquierda a derecha, pero el resultado es 1.644834 si se suma de derecha a izquierda, a partir de $n = 5000$.

Debe notarse que el resultado de sumar de derecha a izquierda es más exacto, ya que en todos los términos se suman valores de igual magnitud.

Por el contrario, al sumar de izquierda a derecha, una vez que se avanza en la sumatoria, se sumarán números cada vez más grandes con números más pequeños.

Lo anterior se corrobora si se realiza la suma en ambos sentidos, pero ahora con doble precisión. El resultado obtenido es 1.64473408684689 (estos resultados pueden variar de máquina a máquina).

1.3 Reducción de errores.

Para resolver la ecuación cuadrática

$$100 x^2 - 10011 x + 10.011 = 0,$$

el método común sería usar la fórmula

$$x = \frac{-b \pm \sqrt{b^2 - 4 a c}}{2a},$$

después de dividir la ecuación entre 100

$$\begin{aligned} x^2 - 100.11 x + 10.011 &= 0 \\ x &= \frac{100.11 \pm \sqrt{(-100.11)^2 - 4(0.10011)}}{2} \end{aligned}$$

Trabajando con aritmética de cinco dígitos

$$x = \frac{100.11 \pm \sqrt{10022 - 0.40044}}{2} = \frac{100.11 \pm \sqrt{10022}}{2}$$

$$= \frac{100.11 \pm 100.11}{2} = \begin{cases} \frac{200.22}{2} = 100.11 \\ 0 \end{cases}$$

Las soluciones verdaderas, redondeadas a cinco dígitos decimales, son 100.11 y 0.00100.

El método empleado fue adecuado para la solución mayor, pero no del todo para la solución menor. Si las soluciones fueran divisores de otras expresiones, la solución $x = 0$ hubiese causado problemas serios.

Se restaron dos números “casi iguales” (números iguales en aritmética de cinco dígitos) y sufrieron pérdida de exactitud.

¿Cómo evitar esto? Una forma sería reescribir la expresión para la solución de una ecuación cuadrática a fin de evitar la resta de números “casi iguales”.

El problema, en este caso, se da en el signo negativo asignado a la raíz cuadrada; esto es:

$$\frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Multiplicando numerador y denominador por $-b + \sqrt{b^2 - 4ac}$, queda

$$\frac{(-b - \sqrt{b^2 - 4ac})(-b + \sqrt{b^2 - 4ac})}{2a(-b + \sqrt{b^2 - 4ac})} = \frac{(-b)^2 - (b^2 - 4ac)}{2a(-b + \sqrt{b^2 - 4ac})}$$

$$\frac{4ac}{2a(-b + \sqrt{b^2 - 4ac})} = \frac{2c}{(-b + \sqrt{b^2 - 4ac})}$$

Usando esta expresión con $a = 1$, $b = -100.11$, y $c = 0.10011$, se obtiene

$$\frac{2(0.10011)}{100.11 + \sqrt{10022}} = \frac{0.20022}{200.22} = 0.001 \text{ (en aritmética de cinco dígitos)}$$

que es el valor verdadero, redondeado a cinco dígitos decimales.

Esta forma alternativa para calcular una raíz pequeña de una ecuación cuadrática, casi siempre produce una respuesta más exacta que la de la fórmula usual (véase el problema 2.31).

1.4 Más sobre reducción de errores.

Se desea evaluar la expresión $A / (1 - \sin x)$, en $x = 89^\circ 41'$. En tablas con cinco cifras decimales, $\sin 89^\circ 41' = 0.99998$. Con aritmética de cinco dígitos y redondeando se tiene:

$$\sin x = 0.99998 \quad \text{y} \quad 1 - \sin x = 0.00002$$

El valor de $\sin x$ sólo tiene cuatro dígitos exactos (confiables). Por otro lado, el único dígito que no es cero en $1 - \sin x$ se ha calculado con el dígito no confiable de $\sin x$, por lo que se pudo perder la exactitud en la resta.

Tal situación de arriba puede mejorarse observando que

$$1 - \sin x = \frac{(1 - \sin x)(1 + \sin x)}{1 + \sin x} = \frac{1 - \sin^2 x}{1 + \sin x} = \frac{\cos^2 x}{1 + \sin x}$$

Por esto, es posible escribir $1 - \sin x$ de una forma que no incluye la resta de dos números casi iguales.

1.5 Comparaciones seguras.

A menudo, en los métodos numéricos la comparación de igualdad de dos números en notación de punto flotante permitirá terminar la repetición de un conjunto de cálculos (proceso cíclico o iterativo). En vista de los errores observados, es recomendable comparar la diferencia de los dos números en valor absoluto contra una tolerancia ε apropiada, usando por ejemplo el operador de relación menor o igual ($\leq$). Esto se ilustra en seguida.

En lugar de:

SI $X = Y$ ALTO; en caso contrario REPETIR las instrucciones 5 a 9.

Deberá usarse:

SI $\text{ABS}(X - Y) \leq \varepsilon$ ALTO; en caso contrario REPETIR las instrucciones 5 a 9.

En lugar de:

REPETIR

{Pasos de un ciclo}

HASTA QUE $X = Y$

Deberá usarse:

REPETIR

{Pasos de un ciclo}

HASTA QUE $\text{ABS}(X - Y) \leq \varepsilon$

donde ε es un número pequeño (generalmente menor que uno, pero puede ser mayor, dependiendo del contexto en que se trabaje) e indicará la cercanía de X con Y , que se aceptará como "igualdad" de X y Y .

1.6 Análisis de resultados.

Al ejecutar las siguientes instrucciones en Visual Basic con doble precisión y en Matlab, se tiene, respectivamente:

```
Dim Y As Double, A as Double
Y=1000.2
A=Y-1000
Print A
```

Se obtiene:
0.2000000000000045

```
format long
Y=1000.2;
A=Y-1000
```

Se obtiene:
0.200000000000005

Ejecute las mismas instrucciones usando $Y = 1000.25$. Los resultados ahora son correctos. Explíquelo.

En doble precisión pueden manejarse alrededor de quince dígitos decimales de exactitud, de modo que la resta de arriba se representa

$$1000.200 - 1000.000$$

La computadora convierte Y a binario dando un número infinito de ceros y unos, y almacena un número distinto a 1000.2 (véase el problema 1.6 b).

Por otro lado, 1000 sí se puede almacenar o representar exactamente en la computadora en binario en punto flotante (los números con esta característica se llaman números de máquina). Al efectuarse la resta se obtiene un número diferente de 0.2. Esto muestra por qué deberá analizarse siempre un resultado de un dispositivo digital antes de aceptarlo.

1.7 Más sobre análisis de resultados.

El método de posición falsa (véase sección 2.4) obtiene su algoritmo al encontrar el punto de corte de la línea recta que pasa por los puntos (x_D, y_D) , (x_I, y_I) y el eje x . Pueden obtenerse dos expresiones para encontrar el punto de corte x_M :

$$\text{i) } x_M = \frac{x_I y_D - x_D y_I}{y_D - y_I} \quad \text{ii) } x_M = x_D - \frac{(x_D - x_I) y_D}{y_D - y_I}$$

Si $(x_D, y_D) = (2.13, 4.19)$ y $(x_I, y_I) = (1.96, 6.87)$ y usando aritmética de tres dígitos y redondeando, ¿cuál es la mejor expresión y por qué?

Solución

Sustituyendo en i) y en ii)

$$\text{i) } x_M = \frac{1.96 (4.19) - 2.13 (6.87)}{4.19 - 6.87} = 2.38$$

$$\text{ii) } x_M = 2.13 - \frac{(2.13 - 1.96) 4.19}{4.19 - 6.87} = 2.40$$

Al calcular los errores absoluto y relativo, y tomando como valor verdadero a 2.395783582, el cual se calculó con aritmética de 13 dígitos, se tiene:

$$\text{i) } EA = 2.395783582 - 2.38 = 0.015783582$$

$$ER = \frac{0.015783582}{2.395783582} = 0.006588066$$

$$\text{ii) } EA = 2.395783582 - 2.40 = 0.004216418$$

$$ER = \frac{0.004216418}{2.395783582} = 0.001759932$$

de donde es evidente que la forma ii) es mejor. Se sugiere al lector reflexionar sobre el porqué.

Problemas propuestos

1.1 Convierta* los siguientes números decimales a los sistemas con base 2 y base 8, y viceversa.

a) 536 b) 923 c) 1536 d) 8 e) 2 f) 10 g) 0

1.2 Escriba los símbolos o numerales romanos correspondientes a los siguientes símbolos arábigos.

10, 100, 1000, 10000, 100000, 1000000

* Puede usar el Programa 1.1 del CD del texto para comprobar sus resultados.

1.3 Convierta los siguientes números enteros del sistema octal a sistema binario y viceversa.

- a) 0 b) 573 c) 7 d) 777 e) 10 f) 2

1.4 Responda las siguientes preguntas.

- a) ¿El número 101121 pertenece al sistema binario?
 b) ¿El número 3852 pertenece al sistema octal?
 c) Si su respuesta es NO en alguno de los incisos, explique por qué; si es SÍ, conviértalo(s) a decimal.

1.5 Convierta los siguientes números fraccionarios dados en decimal a binario y octal.

- a) 0.8 b) 0.2 c) 0.973 d) 0.356 e) 0.713 f) 0.10

1.6 Convierta los siguientes números dados en binario a decimal y viceversa, usando la conversión a octal como paso intermedio.

- a) 1000 b) 001011 c) 01110 d) 10101 e) 11111 f) 10010
 g) 01100

1.7 Convierta los siguientes números fraccionarios dados en binario a decimal.

- a) 0.0011 b) 0.010101 c) 0.11 d) 0.11111 e) 0.00110011 f) 0.0110111
 g) 0.00001 h) 0.1

1.8 Repita los incisos a) a h) del problema 1.7, pero pasando a octal como paso intermedio.

1.9 Convierta los siguientes números en decimal a octal y binario.

- a) -0.9389 b) 977.93 c) 985.34 d) 0.357 e) 10.1 f) 0.9389
 g) 888.222 h) 3.57

1.10 En la sección 1.2 se dijo que cada palabra de 16 bits puede contener un número entero cualquiera del intervalo -32768 a $+32767$. Investigue por qué se incluye al -32768 , o bien por qué el intervalo no inicia en -32767 .

1.11 Represente el número -26 en una palabra de 8 bits.

1.12 Considere una computadora con una palabra de 8 bits. ¿Qué rango de números enteros puede contener dicha palabra?

1.13 Dados los siguientes números de máquina en una palabra de 16 bits:

- a)

1	0	0	0	1	0	1	1	0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 b)

0	0	0	1	1	0	0	0	1	0	0	0	1	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

 c)

0	1	0	0	0	0	1	0	1	1	0	0	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

¿Qué decimales representan?

1.14 Normalice los siguientes números.

- a) 723.5578 b) 8×10^3 c) 0.003485 d) -15.324

Sugerencia: Pasar los números a binario y después normalizarlos.

- 1.15** Represente en doble precisión el número decimal del ejemplo 1.10.
- 1.16** Elabore un programa para la calculadora o el dispositivo de cálculo con el que cuente, de modo que el número 0.0001 se sume diez mil veces consigo mismo.

$$\begin{array}{ccccccc} 0.0001 & + & 0.0001 & + & \dots & + & 0.0001 \\ 1 & & 2 & & & & 10000 \end{array}$$

El resultado deberá imprimirse. Interprete este resultado de acuerdo con los siguientes lineamientos:

- Si es 1, ¿cómo es posible si se sumaron diez mil valores que no son realmente 0.0001?
 - En caso de haber obtenido 1, explore con el valor 0.00001, 0.000001, etc., hasta obtener un resultado diferente de 1.
 - ¿Es posible obtener un resultado menor de 1? ¿Por qué?
- 1.17** Efectúe con el programa del problema 1.16 los cálculos de los incisos a) a d) del ejemplo 1.12 de la página 18 y obtenga los resultados de la siguiente manera:
- Inicialice la variable SUMA con 0, 1, 1000 y 10000 en los incisos a), b), c) y d), respectivamente, y luego en un ciclo sume a ese valor diez mil veces el 0.0001. Anote sus resultados.
 - Inicialice la variable SUMA con 0 para los cuatro incisos y al final del ciclo donde se habrá sumado 0.0001 consigo mismo 10000 veces, sume a ese resultado los números 0, 1, 1000 y 10000 e imprima los resultados.

Interprete las diferencias de los resultados.

- 1.18** La mayoría de las calculadoras científicas almacenan dos o tres dígitos de seguridad más de los que despliegan. Por ejemplo, una calculadora que despliega ocho dígitos puede almacenar realmente diez (dos dígitos de seguridad); por tanto, será un dispositivo de diez dígitos. Para encontrar la exactitud real de su calculadora, realice las siguientes operaciones:
- Divida 10 entre 3, al resultado réstele 3.
 - Divida 100 entre 3, al resultado réstele 33.
 - Divida 1000 entre 3, al resultado réstele 333.
 - Divida 10000 entre 3, al resultado réstele 3333.

Notará que la cantidad de los números 3 desplegados se va reduciendo.

La cantidad de números 3 desplegada en cualquiera de las operaciones anteriores, sumada a la cantidad de ceros utilizados con el 1, indica el número de cifras significativas que maneja su calculadora. Por ejemplo, si con la segunda operación despliega 0.3333333, la calculadora maneja nueve cifras significativas de exactitud (7 + 2 ceros que tiene 100).

ALERTA: Si su calculadora es del tipo intérprete BASIC, no realice las operaciones como 1000/3–333 porque obtendrá otros resultados.

- 1.19** Evalúe la expresión $A / (1 - \cos x)$, en un valor de x cercano a 0° . ¿Cómo podría evitar la resta de dos números casi iguales en el denominador?
- 1.20** Deduzca las expresiones para x_M dadas en el ejercicio 1.7.
- 1.21** Determine si en su calculadora o microcomputadora se muestra un mensaje de *overflow* o no.
- 1.22** Un número de máquina para una calculadora o computadora es un número real que se almacena exactamente (en forma binaria de punto flotante). El número -125.32 del ejemplo 1.10, evidentemente no es un número de máquina (si el dispositivo de cálculo tiene una palabra de 16 bits). Por otro lado, el número -26 del ejemplo 1.8 sí lo es, empleando una palabra de 16 bits. Determine 10 números de máquina en el intervalo $[10^{-19}, 10^{18}]$ cuando se emplea una palabra de 16 bits.

- 1.23** Investigue cuántos números de máquina positivos es posible representar en una palabra de 16 bits.
- 1.24** Haga el análisis de la propagación de errores para la resta (véase análisis de la suma, en la sección 1.3).
- 1.25** Resuelva el siguiente sistema de ecuaciones usando dos cifras decimales para guardar los resultados intermedios y finales:

$$21.76x + 24.34y = 1.24 \quad 14.16x + 15.84y = 1.15$$

y determine el error cometido. La solución exacta (redondeada a 5 cifras decimales) es $x = -347.89167$, $y = 311.06667$.

- 1.26** Se desea evaluar la función e^{5x} en el punto $x = 1.0$; sin embargo, si el valor de x se calculó en un paso previo con un pequeño error y se tiene $x^* = 1.01$; determine ϵ_f con las expresiones dadas en la evaluación de funciones de la sección 1.3. Luego, establezca ϵ_f como $f(1) - f(1.01)$ y compare los resultados.
- 1.27** Codifique el siguiente algoritmo en su microcomputadora (utilice precisión sencilla).
- PASO 1. Leer A.
- PASO 2. Mientras $A > 0$, repetir los pasos 3 y 4.
- PASO 3. IMPRIMIR $\text{Ln}(\text{Exp}(A)) - A$, $\text{Exp}(\text{Ln}(A)) - A$.
- PASO 4. Leer A.
- PASO 5. TERMINAR.

Ejécútelos con diferentes valores de A, por ejemplo 0.18, 0.21, 0.25, 1, 1.5, 1.8, 2.5, 3.14159, 0.008205, 0.3814 entre otros, y observe los resultados.

- 1.28** Modifique el programa del problema del ejemplo 1.27 usando doble precisión para A y compare los resultados.
- 1.29** Modifique el paso 3 del programa del problema 1.27 para que quede así:

$$\text{IMPRIMIR } \text{SQR}(A \wedge 2) - A, \text{SQR}(A) \wedge 2 - A$$

y vuelva a ejecutarlo con los mismos valores.

- 1.30** Realice la modificación indicada en el problema 1.29 al programa del problema 1.28. Compare los resultados.
- 1.31** Repita los problemas 1.27 a 1.30 con lenguaje Pascal (puede usar Delphi, por ejemplo), con lenguaje Visual C++ y compare los resultados con los obtenidos en Basic.

Proyecto

Un *número de máquina* en una calculadora o computadora es un número que está almacenado exactamente en forma normalizada de punto flotante (véase sección 1.2).

Todo dispositivo digital sólo puede almacenar un número finito de *números de máquina* N . Por tanto, la mayoría de los números reales no puede almacenarse exactamente en cualquier dispositivo. Calcule cuántos *números de máquina* pueden almacenarse en una palabra de 16 bits.