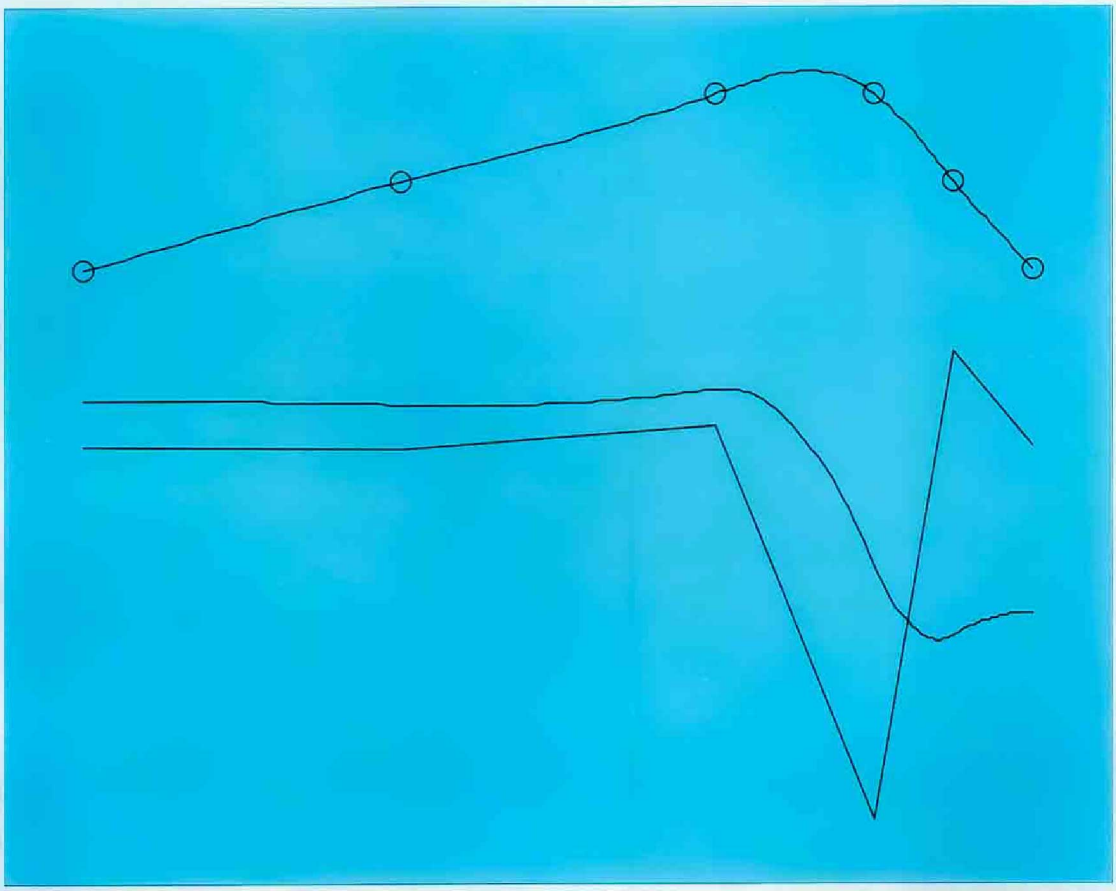
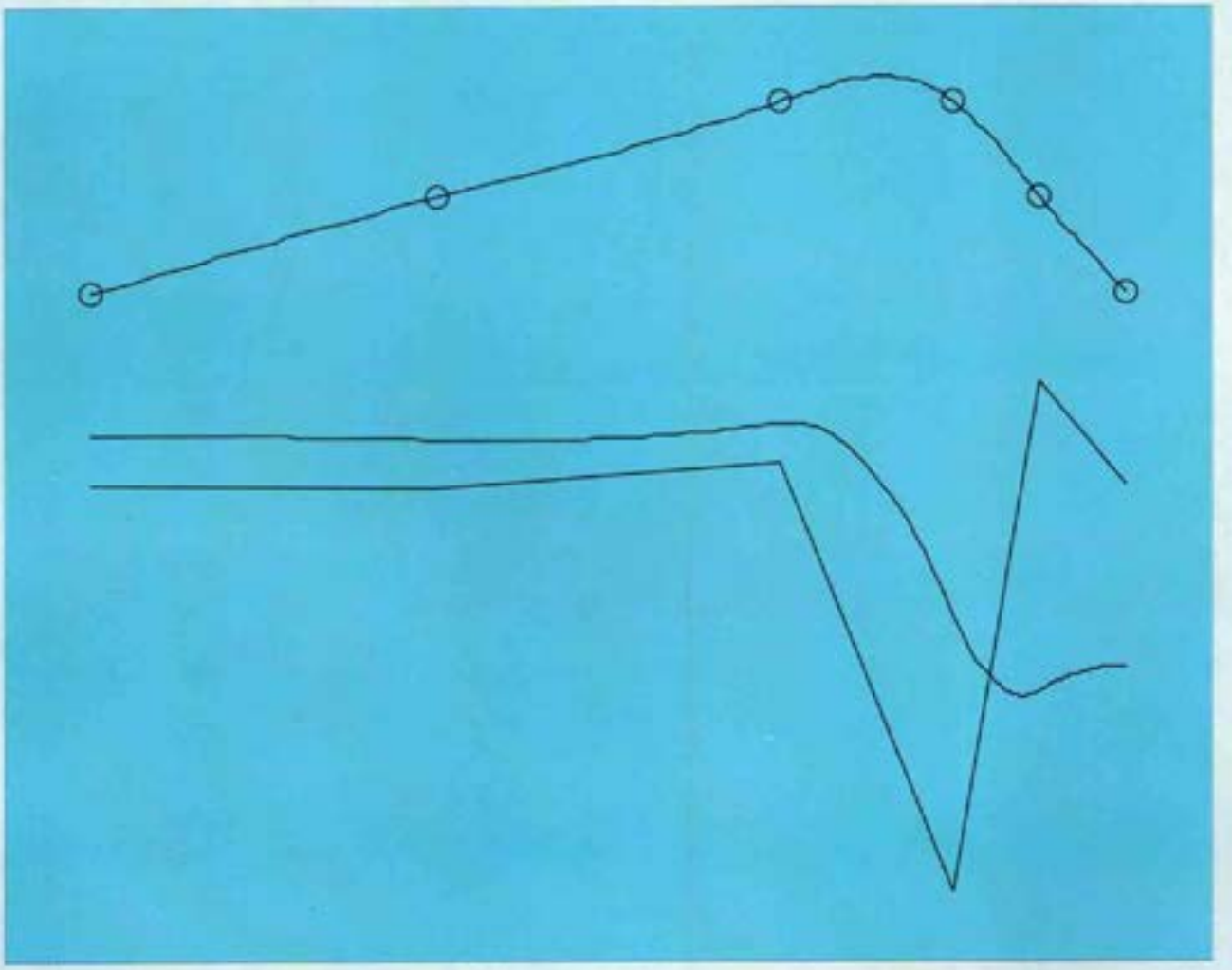
Métodos Numéricos I



Métodos Numéricos I



ANÁLISIS DE ERROR, ARITMÉTICA DE PUNTO FLOTANTE E INTERPOLACIÓN

Justino Alavez Ramírez

Métodos numéricos I

FEUHD
FONDO EDITORIAL
UNIVERSITARIO
HISTORICO DIGITALIZADO

Guillermo Narváez Osorio
Rector

Métodos numéricos I

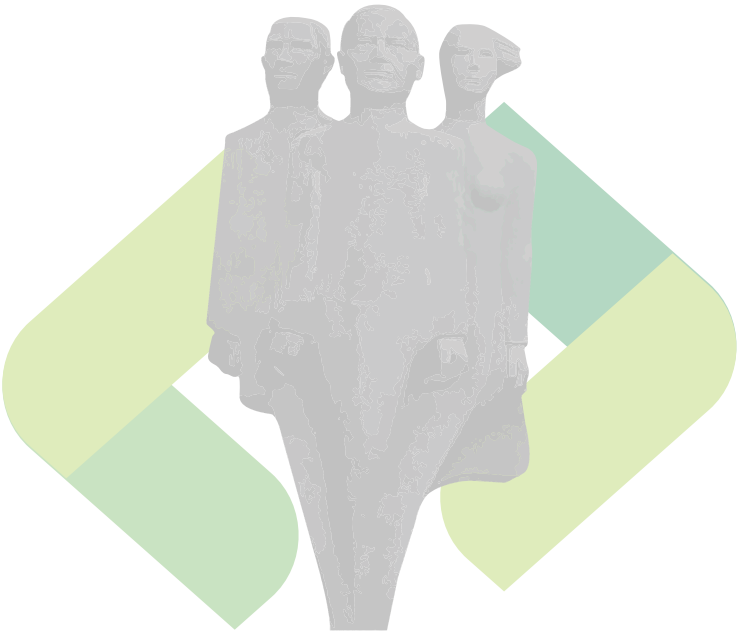
Justino Alavez Ramírez

Universidad Juárez
Autónoma de Tabasco

Edición digitalizada, 2021

Serie: FEU-HD / 2021 / 028

Hecho en Villahermosa, Tabasco, México.



JUSTINO ALAVEZ RAMÍREZ

Nació en Santa Catarina Estetla, Etla, Oaxaca, el 26 de Septiembre de 1959.

Egresado de la Licenciatura en Física y Matemáticas, en la Escuela Superior de Física y Matemáticas del IPN en 1984; y de la Maestría en Ciencias (Matemáticas) en la Facultad de Ciencias de la UNAM en 1994.
Candidato a Doctor en Ciencias por la UNAM en Marzo del 2006.

Profesor Investigador de Tiempo Completo de la División Académica de Ciencias Básicas de la UJAT desde 1986.

Miembro del Cuerpo Académico de Matemáticas Aplicadas.

La primera edición rústica de este libro se terminó de imprimir en marzo de 2002, bajo el título de *APUNTES DE MÉTODOS NUMÉRICOS I (Análisis de Error, Aritmética de Punto Flotante e Interpolación).*

Los objetivos originales planteados en la presentación de aquellos apuntes, siguen vigentes en este libro, es decir, el libro tiene como objetivo servir de guía para los alumnos y maestros del curso de Métodos Numéricos I, que se imparte como asignatura obligatoria en el cuarto semestre de la Licenciatura en computación, y como asignatura optativa para quinto o sexto semestres de las Licenciaturas en Matemáticas y Física.

Se agradecen de antemano todas las críticas y comentarios que se hagan del libro con el fin de mejorarlo.

Métodos Numéricos I

ANÁLISIS DE ERROR, ARITMÉTICA DE PUNTO FLOTANTE E INTERPOLACIÓN

COLECCIÓN
HÉCTOR OCHOA BACELIS
TEXTOS DE ENSEÑANZA DE CIENCIAS BÁSICAS

Candita V. Gil Jiménez
Rectora

Fernando García Lucas
Director de la División Académica de Ciencias Básicas

Métodos Numéricos I

ANÁLISIS DE ERROR, ARITMÉTICA DE PUNTO FLOTANTE E INTERPOLACIÓN

Justino Alavez Ramírez

Universidad Juárez Autónoma de Tabasco
División Académica de Ciencias Básicas

Primera edición, 2006

División Académica de Ciencias Básicas
Km 1.5 carretera Cunduacán Jalpa de Méndez
C.P. 86690
Cunduacán, Tabasco, México

Portada: spline cúbico natural que suavizan un conjunto discreto de datos y las derivadas del spline.

ISBN: 968-9024-03-5

Impreso y hecho en México

Í N D I C E

Página

PRESENTACIÓN

La primera edición rústica de este libro se terminó de imprimir en marzo de 2002, bajo el título de *APUNTES DE MÉTODOS NUMÉRICOS I (Análisis de Error, Aritmética de Punto Flotante e Interpolación)*. Los objetivos originales planteados en la presentación de aquellos apuntes, siguen vigentes en este libro, es decir, el libro tiene como objetivo servir de guía para los alumnos y maestros del curso de Métodos Numéricos I, que se imparte como asignatura obligatoria en el cuarto semestre de la Licenciatura en computación, y como asignatura optativa para quinto o sexto semestres de las Licenciaturas en Matemáticas y Física.

Consta de dos capítulos que cubren casi un sesenta por ciento del curso. En el Capítulo I se comienza resaltando la importancia que tienen los modelos matemáticos para resolver problemas de ciencia y tecnología, y se desarrollan ampliamente los conceptos de error y de los números punto flotante. Se presentan varios ejemplos para resaltar la importancia de los errores por redondeo y de la cancelación numérica. Se hace énfasis sobre la importancia que tienen los algoritmos para resolver problemas, y aunque no se programan específicamente en algún lenguaje de programación, se recomienda programar algunos en la clase. También se recomienda a los alumnos que investiguen cómo se programa en Matlab, y se dejan de ejercicio para que programen algunos algoritmos sencillos.

En el Capítulo II se aborda el problema de interpolación en una variable a través de los polinomios de Lagrange y Newton, y del spline cúbico natural. Se desarrollan y se programan los algoritmos que evalúan tanto los polinomios como el spline en Matlab. Aquí se resalta la importancia del objetivo principal de los métodos numéricos, "que consiste en encontrar soluciones aproximadas a problemas complejos, utilizando sólo las operaciones más simples de la aritmética como sumar, restar, multiplicar y dividir, es decir, resolver problemas difíciles mediante muchos pasos fáciles. Ello significa identificar los procedimientos por medio de los cuales las computadoras pueden hacer ese trabajo por nosotros". Se presentan ejemplos de estimación del error de truncamiento en la aproximación polinomial.

Gran parte de la teoría básica que se requiere para este estudiar este libro provienen de Álgebra Superior I, Álgebra Lineal I y Cálculo Diferencial e Integral, por lo que se recomienda a los alumnos hacer un repaso previo de los temas que se cubren en dichos cursos. Asimismo es conveniente tener conocimientos básicos en algún lenguaje de programación como Turbo Pascal, Fortran, Matlab o algún otro lenguaje de alto nivel. En particular, los programas que se presentan están hechos en Matlab V.5.3.

Los Apuntes fueron bien recibidos por los alumnos y profesores que imparten la asignatura de Métodos Numéricos I, y a sugerencia de algunos de ellos de publicarlo como libro, se optó por modificar el nombre original por el de *MÉTODOS NUMÉRICOS I (Análisis de Error, Aritmética de Punto Flotante e Interpolación)*.

Se hizo una revisión exhaustiva de todo el material presentado en los apuntes, por ejemplo, en el Capítulo I el Ejemplo 1.6 fue cambiado totalmente. Y se desarrolló más a detalle el algoritmo para calcular la suma de N números positivos, que se presenta en la Sección 1.10. En el Capítulo II, la prueba de existencia del Teorema 2.1 se modificó y se presenta ahora de dos maneras diferentes; la primera prueba se da en la Sección 2.2.1 en la Fórmula de Lagrange, y la segunda prueba basada en el método de inducción matemática se presenta en la Sección 2.2.3 en la Fórmula de Newton. Se incorporaron más ejemplos, más programas hechos en Matlab y más gráficas de polinomios de interpolación.

Al final de la prueba de algún teorema, así como al final de la solución de algunos ejemplos, se indica con el símbolo ■.

Se agradecen de antemano todas las críticas y comentarios que se hagan del libro con el fin de mejorarla.

Justino Alavez Ramírez

Julio de 2006

AGRADECIMIENTOS

Gracias al M. en C. Edilberto Nájera Rangel y al Ing. José Luis Meza Godínez, por la paciente revisión que hicieron de los Apuntes. Muchas gracias al M. en C. Francisco Alberto Hernández de la Rosa, por la revisión que hizo de la versión previa de este libro y por sus valiosos comentarios para mejorarlo. Un agradecimiento muy especial al M. en C. Robert Jeffrey Flowers Jarvis, por la revisión exhaustiva que hizo de la versión previa del libro, y por las recomendaciones y sugerencias que me hizo llegar, y que fueron incorporados en la versión final.

Muchas gracias a todos los que de alguna u otra forma contribuyeron en el proceso de la publicación del libro.

Una nota muy personal a la memoria de mi padre Seferino, cuyos consejos me han servido de guía e inspiración a lo largo de mi vida. A mi madre Paula que a sus 77 años, me sigue dando ejemplos invaluables de cómo se debe enfrentar la vida. También doy gracias a tres mujeres maravillosas que han compartido su vida conmigo, y que me han brindado todo el apoyo y comprensión que necesito para mi vida profesional, y en particular para escribir este libro: mi compañera Juana y mis hijas Beatriz y Fabiola.

CAPÍTULO I

ANÁLISIS DE ERROR Y ARITMÉTICA DE PUNTO FLOTANTE

1.1 MODELOS MATEMÁTICOS, APROXIMACIONES NUMÉRICAS Y ALGORITMOS

Los modelos matemáticos son herramientas básicas para solucionar problemas científicos que se presentan en las ciencias naturales y sociales, en el gobierno y en la industria. Típicamente, algunas leyes naturales fundamentales son usadas para derivar una o varias ecuaciones que modelan el problema que se estudia. Con el tratamiento matemático de las ecuaciones, es posible encontrar respuestas a los problemas planteados por los físicos, biólogos, economistas, etc. El esquema con que se puede representar el papel de las matemáticas y en consecuencia de los matemáticos, en las aplicaciones tiene el aspecto que se muestra en la Figura 1.1:

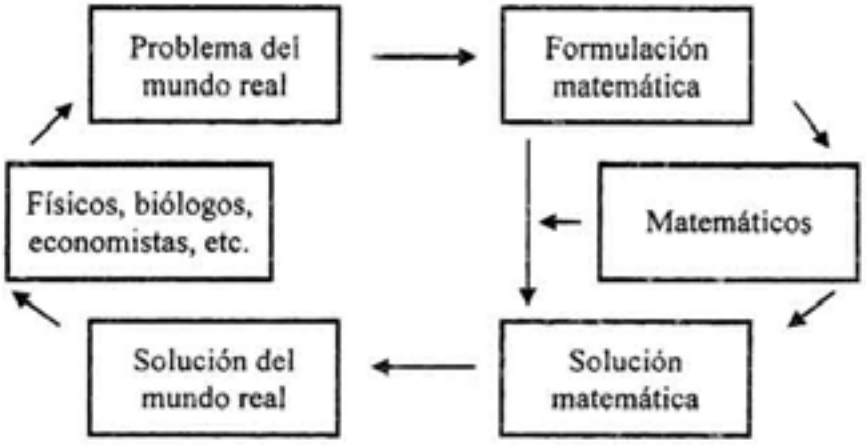


Figura 1.1. Ubicación de las matemáticas y matemáticos en las aplicaciones.

Es importante señalar que para resolver un problema formulado (modelado) matemáticamente, contribuyen muchos especialistas, sobre todo expertos programadores, especialistas en análisis numérico, ecuaciones diferenciales, entre otros.

Para modelar un problema científico se hacen muchas "simplificaciones", por lo que el modelo matemático que se obtiene no describe exactamente la reali-

dad del problema. Las respuestas que produzca el modelo deben ser verificados y comparados con los resultados experimentales.

Algunos ejemplos:

Ejemplo 1.1: ¿Cuál fue la población de Tabasco en 1975?

Solución: Según datos del INEGI, la población del estado de Tabasco entre 1940 y 1995 se muestra en la siguiente Tabla 1.1. Podemos usarlos para estimar razonablemente el volumen de la población de 1975 o incluso del año 2000.

Año	1940	1950	1960	1970	1980	1990	1995
Población	285630	362716	496340	768327	1062961	1501744	1748664

Tabla 1.1. Población del estado de Tabasco (INEGI).

Hay varias formas de resolver el problema, el más sencillo es usar únicamente los datos de 1970 y 1980 para estimar la población en 1975. En este caso, la recta que pasa por estos dos puntos es

$$y = 29463.4(x - 1970) + 768327$$

y es un modelo matemático sencillo del problema. Para $x = 1975$, $y = 915\,644$, que nos dice que en 1975 hubo aproximadamente 915 644 habitantes en Tabasco. Es posible que se pueda mejorar la aproximación si utilizamos más información, por ejemplos los datos entre 1960 y 1980, o bien usar todos los datos disponibles. El modelo matemático para estos casos será más complicado. Lo abordaremos en el Capítulo II. ■

Ejemplo 1.2: Se va a fabricar una lámina corrugada para techos usando una máquina que prensa una hoja plana de aluminio, para convertirla en una hoja metálica con perfil en forma de onda senoidal, ver Figura 1.2.

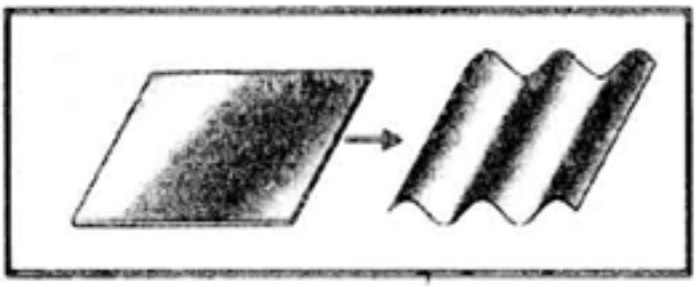


Figura 1.2. Conversión de lámina plana en corrugada.

Se requiere una lámina corrugada de 4 pies de largo, donde la altura de cada onda es de 1 pulgada desde la línea central, y cada onda tiene un periodo de aproximadamente 2π pulgadas. Determinar la longitud de la hoja plana inicial.

Solución: El problema de encontrar la longitud de la hoja plana inicial, equivale a determinar la longitud del arco de la curva dada por $f(x) = \operatorname{sen} x$, desde $x = 0$ pulgadas hasta $x = 48$ pulgadas. Del cálculo integral sabemos que esa longitud es

$$L = \int_0^{48} \sqrt{1 + \left(f'(x)\right)^2}\, dx = \int_0^{48} \sqrt{1 + \cos^2 x}\, dx$$

que es el modelo matemático del problema. Aunque la función seno es una de las funciones más comunes, el cálculo de la longitud de su arco da lugar a una integral elíptica de segunda clase, que no es fácil de evaluar. ■

Ejemplo 1.3: ¿Cómo medir la respuesta cardiaca del corazón?

Solución: En fisiología, la respuesta cardiaca R del corazón se define como el volumen de sangre que el corazón impele por unidad de tiempo. Una respuesta cardiaca anormal es indicativo de una posible enfermedad. Una manera de medir esta respuesta es el llamado *método de dilución por tinción*. Una cantidad D de tinte, medida en miligramos, se inyecta en la arteria pulmonar cerca del corazón. El tinte circula a través de los pulmones y de las venas pulmonares hacia la aurícula izquierda del corazón, y finalmente a través de la aorta. Una sonda insertada en esta última comprueba la cantidad de tinte que sale del corazón, en valores del tiempo igualmente espaciados dentro de un intervalo $[0, T]$ (por ejemplo, cada segundo, hasta los 30 segundos). Para determinar R, se hace la siguiente simplificación: *se supone que la concentración de tinte en cualquier instante t está dada por una función continua $c(t)$*. A partir de este supuesto, se determina el modelo matemático que calcula la respuesta cardiaca R del corazón. Está dada por la ecuación:

$$R = \frac{D}{\int_0^T c(t)\, dt}$$

Conocida o determinada la función $c(t)$ "podemos calcular" la integral, y determinar el valor de R que es la respuesta cardiaca del corazón de la persona que se trate. Por ejemplo, si se emplean 5 mg de tinte con una concentración en miligramos por litro de $c(t) = 0$ entre 0 y 2 segundos y $c(t) = -\frac{1}{40}\left(t^2 - 26t + 48\right)$

entre 2 y 24 segundos, entonces la respuesta cardiaca de un corazón durante 24 segundos, será de 0.1127 litros por segundo o bien de 6.76 litros por minuto. ■

Ejemplo 1.4: Modelo predador–presa: Dadas dos poblaciones que interactúan a través de predador–presa, ¿Cómo pronosticar el desarrollo de las poblaciones?

Solución: Sean $x(t)$ la población de presas en el tiempo t, y
$y(t)$ la población de predadores en el tiempo t.

Supuestos:

a) Si no existen predadores, la especie presa crece a una tasa proporcional a la población de la especie presa.

b) Si no existen presas, la especie predadores disminuye a una tasa proporcional a la población de la especie predadores.

c) La presencia de ambas especies predador–presa, beneficia el crecimiento de la población de la especie predadores y declina a la población de la especie presa a una tasa proporcional al producto de las dos poblaciones.

Estos supuestos conducen al modelo matemático:

$$\frac{dx}{dt} = ax - bxy, \qquad a,b > 0$$

$$\frac{dy}{dt} = -py + qxy, \qquad p,q > 0$$

$$C.I.: \quad x(0) = x_0 \quad \text{y} \quad y(0) = y_0$$

Que se pueden usar para pronosticar el desarrollo de ambas poblaciones. Estas ecuaciones fueron formuladas por primera vez en 1925 y son conocidas como las ecuaciones de Lotka-Volterra. ■

Los modelos matemáticos dan una buena descripción del problema que se estudia, pero no siempre dan respuestas directas a nuestras preguntas, tal es el caso del modelo predador–presa. Hay dos alternativas:

1. Simplificar el modelo de tal manera que permita obtener expresiones analíticas para las variables que nos interesan, como en los Ejemplos 1.1 y 1.3.

2. Para el modelo predador–presa no es posible obtener expresiones analíticas para $x(t)$ y $y(t)$. En este caso, se hacen *aproximaciones* (también se dice *aproximaciones numéricas*) a las derivadas, para transformar el modelo en un nuevo problema que se llama *problema numérico*.

En muchos casos, la primera alternativa es inadecuada, pues el modelo puede resultar tan sencillo que las respuestas que dé no sean confiables, es decir, no tengan nada que ver con la "realidad" del problema. En la segunda alternativa también pueden ocurrir errores, pero aquí las respuestas son más confiables, pues es posible estimar cómo afectan las aproximaciones a la solución exacta del problema. Un esquema para la segunda alternativa se muestra en la Figura 1.3.

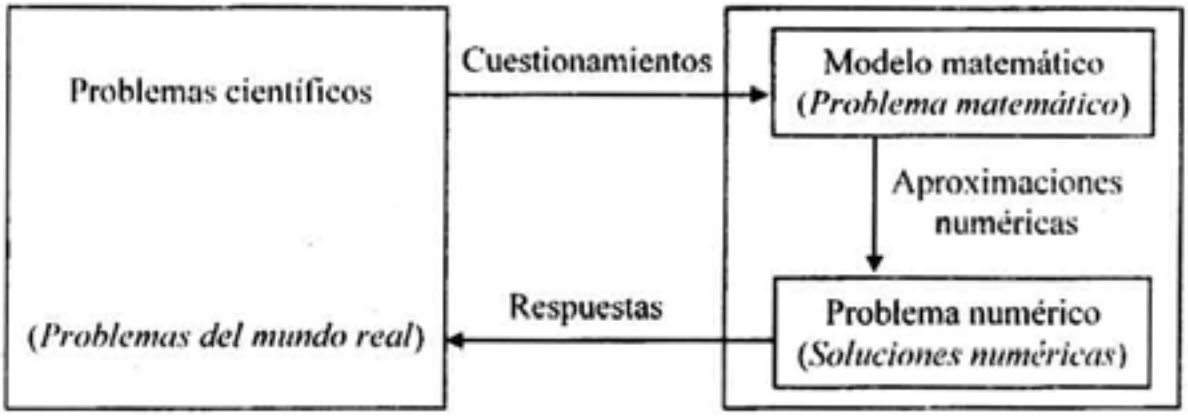


Figura 1.3. Ilustración de la segunda alternativa.

Las aproximaciones o aproximaciones numéricas que se introducen en el modelo matemático, reemplazan el problema matemático por un *problema numérico*.

Definición de problema numérico: Un problema numérico es una descripción clara y sin ambigüedades de las conexiones funcionales entre *datos de entrada*, es decir "variables independientes" del problema, y *datos de salida*, es decir "resultados deseados". Datos de entrada y datos de salida consisten de un *número finito* de cantidades reales.

El objetivo del análisis numérico es resolver problemas numéricos complejos utilizando sólo operaciones simples de la aritmética, con el fin de desarrollar y evaluar métodos para calcular resultados numéricos a partir de los datos de entrada. Los métodos de cálculo se denominan *algoritmo*.

Definición de algoritmo: Un algoritmo para un problema numérico es una descripción completa de un número finito de operaciones bien definidas, a través de los cuales cada vector de datos de entrada permisible es transformado en un vector de datos de salida.

Algoritmo, voz de origen árabe que significa *procedimiento matemático para la solución de un problema.* Nuestras tareas se centrarán en la búsqueda de algoritmos. Para algunos problemas aún no se ha encontrado un algoritmo satisfactorio, mientras que para otros hay varios, por lo que deberemos elegir de entre ellos. Son varias las razones para elegir un algoritmo en vez de otro; dos criterios evidentes son la *rapidez* y la *precisión*. La rapidez es una ventaja evidente, aunque en el caso de problemas pequeños dicha ventaja se ve casi eliminada por la capacidad de la computadora. En problemas de gran escala, la rapidez es aún un factor principal y un algoritmo lento tiene que rechazarse por no ser práctico. Así, siendo otros factores iguales, es seguro que el método más rápido será el elegido.

Una esquematización del problema numérico se muestra en la Figura 1.4.

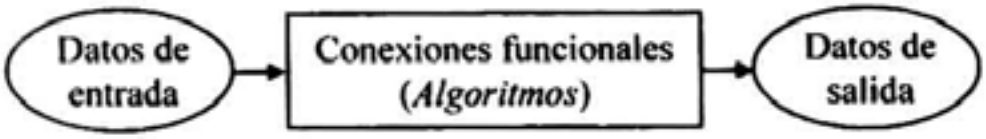


Figura 1.4. Esquema de un problema numérico.

Un algoritmo puede ser formulado usando algún lenguaje de programación como Fortran, C++, Matlab o cualquier otro y procesarlo en una computadora. Dado que una computadora está compuesta de dispositivos que realizan operaciones lógicas y aritméticas; *los procedimientos matemáticos deben de simplificarse a tal grado que sean accesibles para procesarse en una computadora.* Éste es uno de los objetivos principales del estudio de los métodos numéricos.

Los métodos que vamos a estudiar nos permitirán simplificar los procedimientos matemáticos de manera que podamos auxiliarnos con una computadora o una calculadora, para obtener resultados; como ejemplos de los procedimientos que podemos desarrollar, se encuentran: *interpolaciones, ajustes de curvas, raíces de ecuaciones de segundo grado y ceros de polinomios, cálculo de derivadas, integrales, ecuaciones diferenciales, operaciones con matrices, etc.*

Las aplicaciones de los métodos numéricos son prácticamente ilimitadas, y se requieren conocimientos de la materia en disciplinas tan variadas como: eco-

nomía, contabilidad, mercadotecnia, física e ingeniería industrial, civil, eléctrica, mecánica, química, etc. Asimismo, *propicia la formación de criterios de decisión para la elección del método adecuado*, dependiendo del equipo de cómputo con el que nos estemos auxiliando, pudiendo ser éste desde una gran computadora hasta una calculadora de bolsillo (programable o no), pasando por equipos compartidos por varios usuarios.

Características de un algoritmo:

1. FINITO: Siempre debe de terminar en un número determinado de pasos.
2. DEFINIDO: Las acciones deben definirse sin ambigüedad.
3. ENTRADA: Puede tener una o varias entradas, o bien, valores iniciales.
4. SALIDA: Debe tener una o más salidas.
5. EFECTIVIDAD: Todas las operaciones deben ser lo suficientemente básicas para que puedan hacerse exactamente en un determinado tiempo, no mayor que el que tome una persona empleando lápiz y papel.

Ejemplo 1.5: Encontrar la raíz cuadrada de 2 hasta cuatro decimales.

Solución: Existen varios métodos numéricos para aproximar la raíz cuadrada de 2, que sólo utilizan las cuatro operaciones básicas de la aritmética. El favorito es sin duda

$$x_1 = 1, \quad x_{n+1} = \frac{1}{2}\left(x_n + \frac{2}{x_n}\right), \quad n \geq 1.$$

Un algoritmo que se deriva de este método es:

Entrada: $x_1 = 1$.

Salida: x_2 como raíz cuadrada aproximada de 2.

Paso 1: Calcule $x_2 = \frac{1}{2}\left(x_1 + \frac{2}{x_1}\right)$.

Paso 2: Si $|x_2 - x_1| \leq 5 \times 10^{-5}$, termina.
En caso contrario, hacer $x_1 = x_2$ y repetir el paso 1.

Paso 3: Salida x_2.

Paso 4: Alto.

El último valor de x_2 es el resultado buscado. La elección de la cota del error en el paso 2, se justificará en la Observación 3 del Teorema 1.1 de la Sección 1.8.

Los resultados parciales son: $x_2 = 1.500000$ y $|x_2 - x_1| = 5.0 \times 10^{-1}$.

$x_1 = 1.500000$, $x_2 = 1.416667$ y $|x_2 - x_1| = 8.3 \times 10^{-2}$.

$x_1 = 1.416667$, $x_2 = 1.414216$ y $|x_2 - x_1| = 2.45 \times 10^{-3}$.

$x_1 = 1.414216$, $x_2 = 1.414214$ y $|x_2 - x_1| = 2.0 \times 10^{-6}$.

Por lo tanto, $x_2 = 1.414214$ tiene los primeros 4 decimales de la raíz cuadrada del número 2. ■

Un problema importante en estadística consiste en calcular la media aritmética o el promedio de un conjunto de N observaciones. El siguiente algoritmo trata este problema.

Ejemplo 1.6: Un algoritmo simple para calcular el promedio de N observaciones numéricas $x_1, x_2, ..., x_N$, es el siguiente:

Entrada: $N, x_1, x_2, ..., x_N$.
Salida: *PROMEDIO.*
Paso 1: Hacer $PROMEDIO = 0$. (Se inicializa un acumulador).
Paso 2: Para $i = 1, 2, ..., N$.
Hacer $PROMEDIO = PROMEDIO + x_i$. (Se suman todos los números).
Paso 3: $PROMEDIO = \dfrac{PROMEDIO}{N}$
Paso 4: Alto.

Este algoritmo se puede aplicar para obtener información general sobre el aprendizaje de un grupo de alumnos de algún curso. Por ejemplo, las calificaciones del primer examen parcial del grupo de Métodos Numéricos I, del semestre Agosto–Diciembre de 2002 fueron los que se muestran en la Tabla 1.2:

Alumno:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Calif.:	3.6	5.7	6.9	5.7	7.0	2.5	3.0	4.3	9.0	7.7	5.9	5.5	1.4	2.7	4.7	2.4	4.2	5.1	2.2

Tabla 1.2. Calificaciones del primer examen parcial de Métodos Numéricos I.

Aplicando el algoritmo anterior se obtiene *PROMEDIO* = **4.7** correctamente redondeado. El resultado muestra que el aprendizaje del grupo fue muy bajo,

requiere mayor atención de parte del profesor hacia los alumnos y, mayor esfuerzo de parte de los estudiantes en estudiar las notas del curso y resolver los ejercicios que se dejan de tarea. ■

Estudiaremos métodos numéricos con los que podamos obtener resultados con una exactitud satisfactoria en una amplia gama de problemas. Un criterio que fijaremos al algoritmo, en la medida de lo posible, es que los cambios pequeños de los datos iniciales generen cambios igualmente pequeños en el resultado final. Un algoritmo que cumpla con él recibe el nombre de *algoritmo estable*, en caso contrario se llama *algoritmo inestable*. El algoritmo del Ejemplo 1.5 es estable porque x_1 puede tomar un valor positivo diferente de 1 y la salida siempre será la raiz cuadrada de 2. No todos los algoritmos son estables, por lo que se caracterizarán en la medida de lo posible las propiedades de estabilidad de los algoritmos que estudiaremos en el libro.

1.2 CALCULANDO CON NÚMEROS REALES

Muchos números reales se pueden representar con una cadena finita de dígitos, como $\frac{1}{2} = 0.5$, $\frac{8}{5} = 1.6$, sin embargo, muchos otros requieren de una cadena infinita de dígitos, como:

$$\pi = 3.14159265358979\cdots$$
$$\sqrt{2} = 1.41421356237309\cdots$$
$$e = 2.71828182845904\cdots$$

y son muy comunes en las aplicaciones.

Las computadoras, por otro lado, son máquinas finitas, es decir, únicamente procesan números con una cantidad fija de dígitos en su representación. Así que en un problema que se resuelve con la ayuda de una computadora, tanto los datos del problema, resultados intermedios y respuestas finales deben ser aproximados por estos "números especiales" que se representan en la computadora. Es importante tener en cuenta que estas aproximaciones pueden tener serios efectos sobre los resultados del problema. Por ejemplo, algunas subrutinas para evaluar funciones trigonométricas, como $\operatorname{sen}(\theta)$ y $\cos(\theta)$, primero reemplazan θ por $\theta \pm 2k\pi$ para obtener un argumento entre $-\pi$ y π. Este cambio en el argumento no debería cambiar los resultados, porque, por ejemplo $\operatorname{sen}(\theta) = \operatorname{sen}(\theta \pm 2k\pi)$, $k = 1, 2, \ldots$. Pero, π *no puede ser representado exactamente en la computadora*, y entonces el

proceso de substracción altera el resultado. De hecho, por esta razón, es extremadamente difícil calcular $\text{sen}(\theta)$ exactamente para valores grandes de θ. Por ejemplo, tomando $\pi = 3.14159$

$$\text{sen}(10000) = \text{sen}(10000 - 2*1592*3.14159) = \text{sen}(-2.82256) = -0.3136482$$

Calculando directamente con una calculadora, $\text{sen}(10000) = -0.3056144$.
Es importante notar que a pesar de haber usado 5 cifras decimales correctas del número π, la resta en el proceso altera el resultado, y apenas se obtiene una cifra decimal correcta de $\text{sen}(10000)$.

Más generalmente, cualquier proceso numérico que involucra números reales que no pueden ser representados exactamente en la computadora debe ser usado con cuidado. Aún cuando el problema y su solución involucra únicamente números que puedan representarse en la computadora, los cálculos pueden introducir números que no pueden ser representados en la computadora. *En la práctica rara vez se puede calcular la solución exacta de un problema, aunque la solución pueda salir de números que se puedan representar exactamente en la computadora.*

1.3 CONCEPTOS BÁSICOS DE ERROR

Sólo en raras ocasiones los datos proporcionados de un problema serán exactos, puesto que suelen originarse en procesos de medida, de modo que hay un error probable en la información de entrada. Además, el propio algoritmo genera error, quizá redondeos inevitables. La información de salida contendrá entonces error generado por ambas fuentes.

Algunos términos importantes en el análisis de error:

Exactitud: Se refiere a la cercanía de un número o de una medida del valor verdadero que representa.

Precisión: Se refiere al número de cifras significativas que representan una cantidad. Esto se refiere cuando se habla de *simple precisión* o *doble precisión*, dependiendo de la máquina que estemos utilizando.

Dígitos significativos: Son aquellos números diferentes de cero, en una cifra o guarismo, leyendo de izquierda a derecha; empiezan con el primer dígito

diferente de cero y terminan con el tamaño que permitan las celdas que guardan la mantisa. El número 0.0001001234 tiene 7 dígitos significativos. Lo mismo que los números 1001.234 y 1001.234000 .

Existen principalmente tres tipos de errores que afectan los resultados de un cálculo numérico.

a) **Errores en los datos de entrada o heredados:** Son errores contenidos en los valores numéricos de los datos de entrada, y son prácticamente inevitables para el calculista. Esto es, porque generalmente los datos de problemas del mundo real provienen de algún experimento, como la población de Tabasco de la Tabla 1.1 del Ejemplo 1.1. Se clasifican en *sistemáticos* y *accidentales*.
Errores sistemáticos: Debidos a la imprecisión de los aparatos de medición.
Errores accidentales: Debidos a la apreciación del observador y a otras causas.

b) **Error por truncamiento o aproximación:** Se debe a la interrupción de un proceso matemático antes de su terminación. Por ejemplo, cuando la suma de una serie infinita es aproximada por una suma parcial, cuando una derivada es aproximada por un cociente de diferencias. En general, el error por truncamiento ocurre cuando un problema matemático es transformado en un problema numérico. Un caso adicional de error de truncamiento, ocurre cuando una calculadora sólo toma en cuenta los dígitos que aparecen en la pantalla y no analiza el primer dígito perdido.

c) **Error por redondeo:** Se presenta debido a las limitaciones propias de la máquina para representar cantidades que requieren un gran número de dígitos. Existen dos clasificaciones:

Error por redondeo inferior: Se ignoran los dígitos que no pueden conservarse dentro de la localización de memoria correspondiente (es un caso particular del error por truncamiento). El redondeo de 0.10666 hasta cuatro decimales es 0.1066.

Error por redondeo superior: El último dígito que puede considerarse en la localización de memoria se incrementa en una unidad si el primer dígito del resto que se pierde es ≥ 5. En el caso, de que el resto que se pierde sea exactamente igual a cinco, se pide que el último dígito que puede considerarse en la localización de memoria sea impar.

Formalizaremos la definición del error por redondeo en la Sección 1.6.
Una esquematización de los tres tipos de error se muestra en la Figura 1.5:

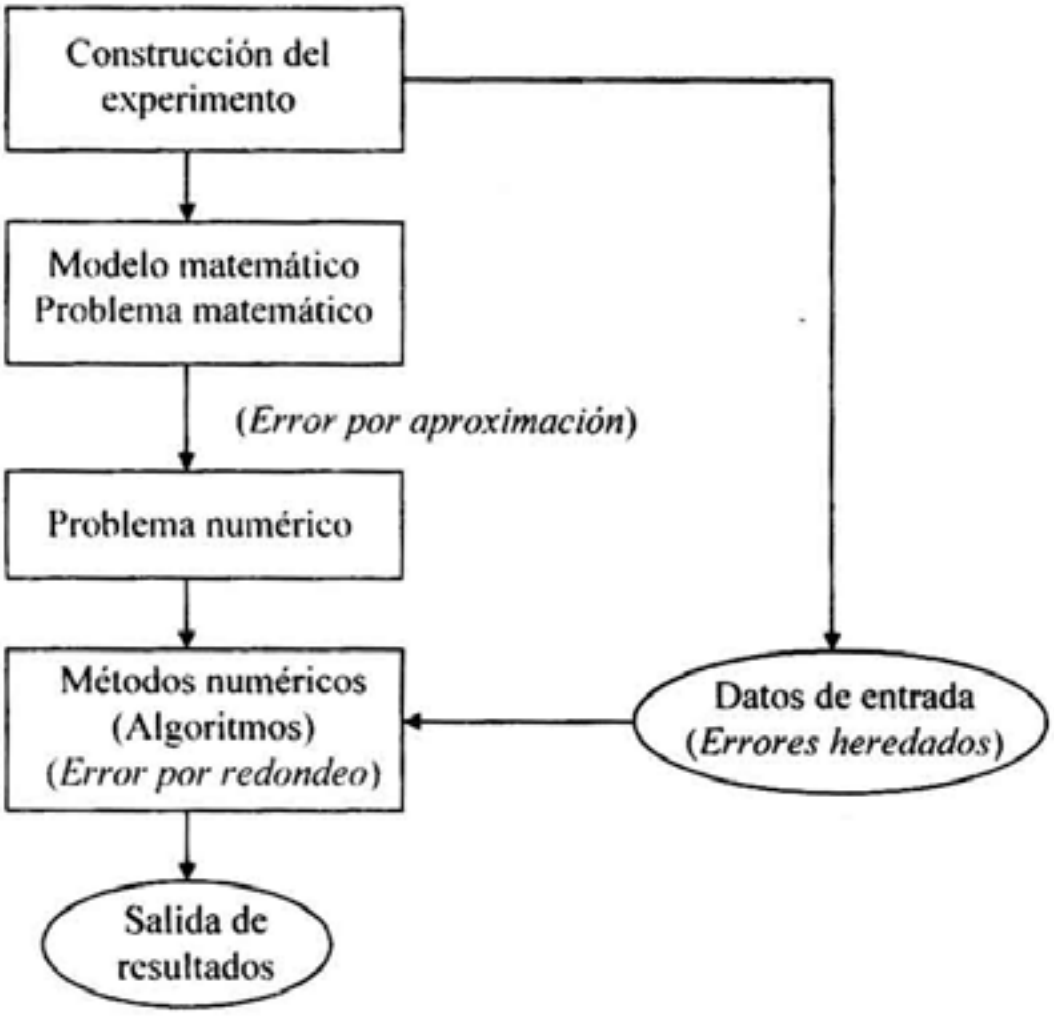


Figura 1.5. Ubicación de las fuentes de error en un proceso numérico.

El error total en un proceso numérico es la suma de todos los errores que ocurren en cada etapa del proceso: *errores heredados + error por aproximación + error por redondeo.*

Ejemplo 1.7: Algunos ejemplos de redondeo superior hasta cuatro lugares decimales, se muestran en la Tabla 1.3.

Número	Redondeo del número
± 0.10666121	± 0.1067
± 0.10664	± 0.1066
± 0.10665	± 0.1066
± 0.10615	± 0.1062
± 1.69999	± 1.7000
± 1.60015001	± 1.6002

Tabla 1.3. Ejemplos de redondeo superior hasta cuatro lugares decimales.

Ejemplo 1.8: Calcular el coseno de x para valores pequeños de x.

Solución: La aproximación que se le aplica al $\cos(x)$ es la serie clásica de Maclaurin, dado por

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots$$

Para resolver el problema con $x = 0.5$, podemos tomar los primeros 4 términos de la serie.

$$\cos(0.5) \equiv 1 - 0.125 + 0.0026042 - 0.000021701 = 0.877582499,$$

en este caso

$$\cos(0.5) = 0.877582499 + \textit{error por aproximación} + \textit{error por redondeo.}$$

El error por aproximación se comete al considerar únicamente los primeros 4 términos de la serie, y el error por redondeo se comete al considerar solamente 5 dígitos significativos en el tercero y cuarto sumandos. El resultado será más exacto mientras más términos tomemos de la serie. Se presentarán más ejemplos a lo largo del curso. ■

Para medir la exactitud con que se determina un número o una cantidad, se usan las medidas conocidas como *error absoluto* y *error relativo.*

Error absoluto: Es la diferencia entre el valor real (también se dice valor exacto o valor verdadero) de un número y su valor aproximado. Si x es el valor real, x^* es el valor aproximado y Δx su error absoluto, entonces:

$$\Delta x = x^* - x.$$

Error relativo: Es el cociente del error absoluto entre el valor real del número.

$$R_x = \frac{\Delta x}{x}, \quad \text{si} \quad x \neq 0.$$

En el caso de que el valor verdadero x se desconozca o sea difícil de manejar, la aproximación x^* se sustituye por x y al resultado se le sigue llamando error relativo. En muchos casos, únicamente se conocerá una cota de la magnitud del error absoluto de una aproximación, por lo que es práctico dar una estimación de la magnitud de los errores absoluto y relativo. Veamos el siguiente ejemplo:

Ejemplo 1.9: Determinar los errores absoluto y relativo, para $x = \sqrt{2}$ y $x^* = 1.414$, así como algunas cotas para dichos errores.

Solución: El error absoluto es, $\Delta x = x^* - x = -0.00021356\ldots$,
y el error relativo

$$R_x = \frac{-0.00021356\ldots}{\sqrt{2}} \cong \frac{-0.00021356\ldots}{1.414} = -0.00015\ldots.$$

Algunas cotas para los errores absoluto y relativo podrían ser por ejemplo:

$$|\Delta x| \leq 0.00022 \quad \text{y} \quad |R_x| \leq 0.00016, \quad \text{ó bien} \quad |\Delta x| \leq 0.0003 \quad \text{y} \quad |R_x| \leq 0.0002. \blacksquare$$

Los errores relativos se presentan con frecuencia como porcentajes, que se pueden estimar como:

$$\text{Porcentaje de error} = \frac{|x - x^*|}{|x|} \times 100, \quad \text{si} \quad x \neq 0.$$

En el Ejemplo 1.9, el porcentaje de error es a lo más de 0.02 %.

Ejemplo 1.10: Cuando 999 999 se multiplica por 8 888 888, el producto es 8 888 879 111 112. Las computadoras trabajan de acuerdo con una "longitud de palabra" establecida, redondeando todos los números según esa longitud. Si realiza la misma multiplicación con una calculadora que visualiza menos de 13 dígitos, obtendrá un resultado diferente. Tales errores de redondeo son errores de algoritmo y se cometen inevitablemente por millones en las computadoras modernas.

Overflow: En el lenguaje técnico de computación se acostumbra emplear este anglicismo, ya que las traducciones posibles no proporcionan una idea clara

de su significado. Se dice que existe un overflow cuando dentro de una localización de almacenamiento no cabe un número, debido a que éste es mayor que la capacidad de la mencionada localización de almacenamiento.

Underflow: Similarmente, se dice que existe un underflow cuando dentro de una localización de almacenamiento no se puede representar un número positivo muy pequeño, debido a que éste es menor que la capacidad de la mencionada localización de almacenamiento.

1.4 PROPAGACIÓN DEL ERROR

Cuando valores aproximados son usados para calcular nuevas cantidades, los errores de cada valor influirán en los resultados. Se derivarán métodos para estimar cómo se propagan los errores contenidos en los datos en un cálculo dado.

Propagación del error para la multiplicación: Sea $z = x\,y$ con $x \neq 0$ y $y \neq 0$.

$$\Delta z = x^* y^* - xy = (x+\Delta x)(y+\Delta y) - xy = x\Delta y + y\Delta x + \Delta x \Delta y .$$

Si suponemos que Δx y Δy son arbitrariamente pequeños, entonces $\Delta x \Delta y$ será más pequeño todavía y se puede despreciar, por lo tanto, el error absoluto para el producto resulta:

$$\Delta z \cong x\Delta y + y\Delta x \qquad \text{..... (1.1)}$$

Es más conveniente considerar el error relativo y su cota:

$$\frac{\Delta z}{z} \cong \frac{\Delta x}{x} + \frac{\Delta y}{y}; \qquad \left|\frac{\Delta z}{z}\right| \lesssim \left|\frac{\Delta x}{x}\right| + \left|\frac{\Delta y}{y}\right| . \qquad \text{..... (1.2)}$$

($\lesssim$ se lee "menor que o aproximadamente igual a").

Propagación del error para la división: Sea $z = \frac{x}{y}$, con $x \neq 0$ y $y \neq 0$.

$$\Delta z = \frac{x^*}{y^*} - \frac{x}{y} = \frac{x+\Delta x}{y+\Delta y} - \frac{x}{y} = \frac{xy + y\Delta x - xy - x\Delta y}{y(y+\Delta y)} = \frac{y\Delta x - x\Delta y}{y(y+\Delta y)} .$$

dividiendo entre de z tenemos:

$$\frac{\Delta z}{z}=\frac{y}{x}\frac{y\Delta x-x\Delta y}{y(y+\Delta y)}=\frac{\Delta x}{x}\frac{y}{y+\Delta y}-\frac{\Delta y}{y+\Delta y}=\left(\frac{\Delta x}{x}-\frac{\Delta y}{y}\right)\frac{y}{y+\Delta y}.$$

Si Δy es suficientemente pequeño, entonces $\dfrac{y}{y+\Delta y}\cong 1$, por lo tanto, el error relativo para la división y su cota son:

$$\frac{\Delta z}{z}\cong\frac{\Delta x}{x}-\frac{\Delta y}{y};\qquad \left|\frac{\Delta z}{z}\right|\lesssim\left|\frac{\Delta x}{x}\right|+\left|\frac{\Delta y}{y}\right|. \qquad \text{..... (1.3)}$$

La propagación del error para la suma y la resta, así como sus respectivas cotas, aparecen como ejercicio número 19 en la Sección 1.11 de Problemas.

Ejemplo 1.11: a) Estimar el valor de $y=x^3$ si $x^*=2\pm 0.001$.

b) Estimar el valor de $z=\dfrac{x}{y}$, y calcular una cota para el error relativo si $x^*=2\pm 0.1$ y $y^*=3\pm 0.2$.

Solución: a) De acuerdo con la fórmula (1.1) tenemos

$$\Delta y\cong x^2\Delta x+x\Delta x^2\cong x^2\Delta x+x(x\Delta x+x\Delta x)\ \Rightarrow\ \Delta y\cong 3x^2\Delta x.$$

Compare el resultado con la derivada de y, esto no es casualidad, la propagación del error en una función diferenciable depende del Teorema del Valor Medio. Sustituyendo valores tenemos

$$\Delta y\cong 3(2^2)(0.001)=0.012,\ \text{ por lo tanto, }\ y=8\pm 0.012.$$

b) La propagación del error lo podemos calcular como

$$\Delta z=\frac{y\Delta x-x\Delta y}{y(y+\Delta y)}=\frac{3(0.1)-2(0.2)}{3(3+0.2)}=-\frac{0.1}{9.6}=-\frac{1}{96}\ \ \therefore\ \ z=\frac{2}{3}\pm\frac{1}{96}.$$

Una cota para el error relativo será:

$$\left|\frac{\Delta z}{z}\right|\lesssim\left|\frac{\Delta x}{x}\right|+\left|\frac{\Delta y}{y}\right|=\frac{0.1}{2}+\frac{0.2}{3}\le 0.1167.$$

Observe que la cota se amplifica, pues

$$\frac{\Delta z}{z}=\frac{1/96}{2/3}=\frac{1}{64}=0.015625<0.1167.\ \blacksquare$$

Investigaremos la propagación del error cuando se evalúa una función diferenciable f de n variables $x_1, x_2, ..., x_n$. Usaremos la siguiente generalización del **Teorema del Valor Medio**: "Si una función real valuada f es diferenciable en una vecindad del punto $x = (x_1, x_2, ..., x_n)$, y $x + \Delta x$ es un punto en la vecindad, entonces existe un número θ, con $0 < \theta < 1$, tal que

$$\Delta f = f(x + \Delta x) - f(x) = \sum_{k=1}^{n} \frac{\partial f}{\partial x_k}(x + \theta \Delta x)\Delta x_k ."$$

Cuando usamos este teorema en la práctica, usualmente se evalúan las derivadas parciales para $x = x^*$ (la aproximación de x). Cuando se conocen únicamente las cotas de los errores para el argumento x, se puede encontrar una cota para Δf usando la desigualdad del triángulo.

Fórmula general para la propagación del error: $\Delta f \cong \sum_{k=1}^{n} \frac{\partial f}{\partial x_k} \Delta x_k$.

Cota de error máximo: $|\Delta f| \lesssim \sum_{k=1}^{n} \left| \frac{\partial f}{\partial x_k} \Delta x_k \right|$ ..... (1.4)

Ejemplo 1.12: Obtener una estimación del valor de $y = \text{sen}(x_1^2 x_2)$ si $x_1 = 0.75 \pm 10^{-2}$ y $x_2 = 0.413 \pm 3 \times 10^{-3}$.

Solución: El valor aproximado de y es

$$y^* = \text{sen}((0.75^2)(0.413)) = 0.23022851... = 0.23 + 0.00022851... = 0.23 \pm 0.03 \times 10^{-2},$$

donde $y^* = 0.23$ con un margen de error de redondeo aproximado por $\pm 0.03 \times 10^{-2}$.

Aplicamos la fórmula (1.4) para estimar una cota de error máximo:

$$\begin{aligned}
|\Delta y| &\lesssim \left|\cos(x_1^2 x_2) 2 x_1 x_2 \Delta x_1\right| + \left|\cos(x_1^2 x_2) x_1^2 \Delta x_2\right| \\
&= \left|(0.97313659...)(2)(0.75)(0.413)(10^{-2})\right| + \left|(0.97313659...)(0.5625)(3 \times 10^{-3})\right| \\
&\leq \left|(0.974)(2)(0.75)(0.413)(10^{-2})\right| + \left|(0.974)(0.5625)(3 \times 10^{-3})\right| \\
&= 0.603393 \times 10^{-2} + 1.643625 \times 10^{-3} = 0.7677555 \times 10^{-2} \leq 0.77 \times 10^{-2},
\end{aligned}$$

de donde $|\Delta y| \leq 0.77\times10^{-2}$. "Es importante observar cómo se fueron obteniendo las cotas superiores en cada paso."

Finalmente para obtener el valor aproximado de y, añadimos la cota de error máximo que acabamos de calcular al error de redondeo de y^*:

$$y = 0.23 \pm \left(0.77\times10^{-2} + 0.03\times10^{-2}\right)$$

Por lo tanto, $y = 0.23 \pm 0.8\times10^{-2}$. ■

La cota de error máximo es muy pesimista si el número de variables es grande. Si f es una función vectorial de n variables, es decir, $f = (f_1, f_2, \ldots, f_m)$ el método anterior se puede usar para estimar el error para cada componente por separado.

1.5 REPRESENTACIÓN DE NÚMEROS REALES EN BASE β

Comenzamos analizando el siguiente ejemplo, donde se presenta la Tabla 1.4 con algunos números reales escritos en base 10.

Ejemplo 1.13:

Número real *a* en forma reducida.	**Valor del número real *a* en forma expandida**
230789	$230789 = 2\times10^5 + 3\times10^4 + 0\times10^3 + 7\times10^2 + 8\times10 + 9$.
$\frac{1}{8} = 0.125$	$\frac{1}{8} = \frac{1}{10} + \frac{2}{10^2} + \frac{5}{10^3}$.
$\sqrt{2} = 1.41421\cdots$	$\sqrt{2} = 1 + \frac{4}{10} + \frac{1}{10^2} + \frac{4}{10^3} + \frac{2}{10^4} + \frac{1}{10^5} + \cdots$
$\pi = 3.14159\cdots$	$\pi = 3 + \frac{1}{10} + \frac{4}{10^2} + \frac{1}{10^3} + \frac{5}{10^4} + \frac{9}{10^5} + \cdots$

Tabla 1.4. Algunos números reales escritos en base 10.

En general, si β es un número entero ≥ 2, entonces cualquier número real a puede escribirse en la forma reducida:

$$a = \pm \underbrace{a_1 a_2 \cdots a_n}_{\text{Parte entera.}} \cdot \underbrace{a_{n+1} a_{n+2} \cdots a_{n+m} \cdots}_{\text{Parte fraccionaria.}} \qquad \text{..... (1.5)}$$

donde $0 \leq a_i \leq \beta - 1$, $i = 1, 2, \ldots$

El valor de a en forma expandida es dado por la serie infinita:

$$a = \pm \left(a_1 \beta^{n-1} + a_2 \beta^{n-2} + \cdots + a_{n-1}\beta + a_n + \frac{a_{n+1}}{\beta} + \frac{a_{n+2}}{\beta^2} + \cdots + \frac{a_{n+m}}{\beta^m} + \cdots \right) \qquad (1.6)$$

Las bases más comúnmente usados son $\beta = 10, 2, 8$ y 16. A los números en estas bases se les llama decimales, binarios, octales y hexadecimales, respectivamente. Para $\beta = 16$, los dígitos son 1, 2, ... , 9, A, B, C, D, E, F, 0. La mayoría de las computadoras usan el sistema binario que únicamente necesita dos dígitos, 0 y 1 para representar los números reales.

Ejemplo 1.14: $(23)_{10} = 2 \times 10 + 3 = 2^4 + 2^2 + 2 + 1 = (10111)_2$

$= 2 \times 8 + 7 = (27)_8$

$= 16 + 7 = (17)_{16}$.

Por lo tanto, $(23)_{10} = (10111)_2 = (27)_8 = (17)_{16}$.

Si un switch cerrado (o polaridad positiva) denota el 1 y un switch abierto (o polaridad negativa) denota el 0, entonces cualquier número binario puede ser representado por una serie de switchs abiertos y cerrados (o polaridades negativas y positivas), como se muestra en la Figura 1.6.

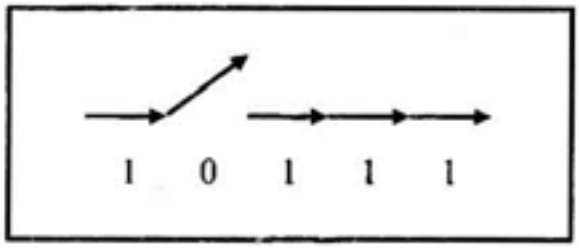


Figura 1.6. Representación del número binario 10111.

Un problema con el sistema binario es que requiere de muchos dígitos binarios para representar números decimales (aunque sean pequeños), el número 23 necesita 5 dígitos binarios 10111 y un número de 8 dígitos decimales requiere alrededor de 27 dígitos binarios.

Las personas usamos números decimales, las computadoras usan números binarios y octales. Ciertas computadoras modernas (la serie IBM 360) usan números hexadecimales.

Ejemplo 1.15: Convertir en binario los números decimales 101, 0.1, 101.1, y en decimal el número binario 0.1011.

Solución: Usaremos las propiedades de las series infinitas para convertir números decimales en binarios.

a) $(101)_{10} = 10^2 + 1 = (2^3 + 2)^2 + 1 = 2^6 + 2^5 + 2^2 + 1 = (1100101)_2$.

b) $(0.1)_{10} = \frac{1}{10} = \frac{3}{30} = \frac{1}{30} + \frac{2}{30} = \frac{1}{30} + \frac{1}{15}$,

donde

$$\frac{1}{30} = \frac{1}{2^5 - 2} = \frac{1}{2^5} \frac{1}{1 - \frac{1}{2^4}} = \frac{1}{2^5} \sum_{n=0}^{\infty} \frac{1}{2^{4n}} = \sum_{n=0}^{\infty} \frac{1}{2^{4n+5}}, \quad \text{y}$$

$$\frac{1}{15} = \frac{1}{2^4 - 1} = \frac{1}{2^4} \frac{1}{1 - \frac{1}{2^4}} = \frac{1}{2^4} \sum_{n=0}^{\infty} \frac{1}{2^{4n}} = \sum_{n=0}^{\infty} \frac{1}{2^{4n+4}}$$

Por lo tanto,

$$(0.1)_{10} = \sum_{n=0}^{\infty} \left(\frac{1}{2^{4n+4}} + \frac{1}{2^{4n+5}} \right) = \left(\frac{1}{2^4} + \frac{1}{2^5} \right) + \left(\frac{1}{2^8} + \frac{1}{2^9} \right) + \left(\frac{1}{2^{12}} + \frac{1}{2^{13}} \right) + \cdots$$
$$= (0.0001100110011\cdots)_2.$$

c) $(101.1)_{10} = (101)_{10} + (0.1)_{10} = (1100101.0001100110011\cdots)_2$.

d) $(0.1011)_2 = \frac{1}{2} + \frac{1}{2^3} + \frac{1}{2^4} = \frac{11}{16} = (0.6875)_{10}$. ■

Otra alternativa para transformar el número $(101.1)_{10}$ a binario, son los siguientes algoritmos que se aplican en la parte entera y fraccionaria del número respectivamente. Ver Figura 1.7.

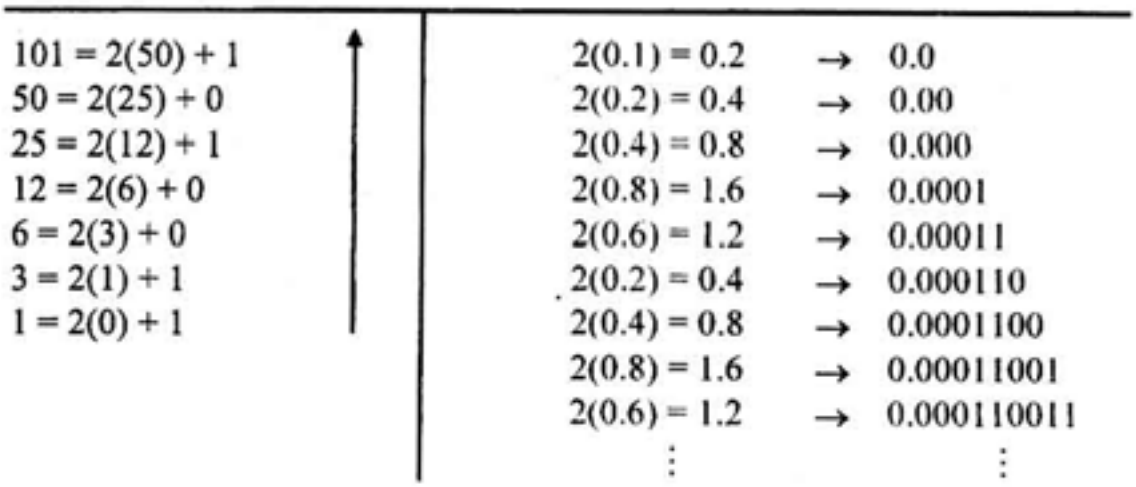


Figura 1.7. Algoritmo para transformar un número decimal a binario.

Los residuos en la parte entera se leen de abajo hacia arriba como indica la flecha en la Figura 1.7, para obtener de nuevo el número binario 1100101 que es el equivalente binario del número decimal 101. En este caso, se está aplicando sucesivamente el Algoritmo de la División, dividiendo entre 2 hasta obtener un cociente cero. Para la parte fraccionaria se comienza multiplicando por 2 al decimal que se quiere convertir a binario (en este caso al 0.1), la parte entera del resultado es el primer dígito de la parte fraccionaria del binario correspondiente. Se multiplica de nuevo por 2 a la parte fraccionaria del resultado anterior, y la parte entera del nuevo resultado será el siguiente dígito de la parte fraccionaria del binario correspondiente. Se continúa el proceso y se detendrá cuando al multiplicar por 2 una fracción dé como resultado 1.0, en caso contrario, el proceso será infinito como en el ejemplo, pues la sucesión 0011 se repite indefinidamente. Sumando los equivalentes binarios de las partes entera y fraccionaria, obtenemos de nuevo:

$$1100101 + 0.000110011\ldots = 1100101.0001100110011\ldots .$$

El número decimal 0.1 que tiene representación binaria infinita no podrá ser representado exactamente en una computadora binaria. Tampoco podrán ser representados exactamente los números decimales con representación infinita.

La arquitectura de la mayoría de las computadoras está basada sobre el principio de que los datos son procesados con una cantidad fija de información como unidad. Tal unidad de información es llamada una *palabra* y el número de dígitos (usualmente binarios) en una palabra es llamado *longitud de la palabra* de

la computadora. Las longitudes de palabras más comunes son 16 (para la mayoría de los microprocesadores), 32 (IBM 3090, VAX) y 64 en las supercomputadoras. La mayoría de las microcomputadoras usan 32 bits (*dígitos binarios*) para representar los números reales. Los números enteros pueden representarse exactamente en una computadora si la longitud de la palabra es lo suficientemente grande para almacenar todos los dígitos.

¿Cómo se hace para representar los números en la computadora?.

1.6 NÚMEROS PUNTO FLOTANTE

La escritura (1.5) de la sección anterior para el número real $a \neq 0$, se puede escribir también como:

$$a = M \times \beta^e = \pm\, a_0 . a_1 a_2 \cdots \times \beta^e \qquad \text{..... (1.7)}$$

donde β es la base del sistema numérico, e entero es el exponente (*orden o característica de a*), y cada cifra es tal que $0 < a_0 < \beta$ y $0 \le a_i < \beta$ para $i = 1, 2, \ldots$. Esta forma de escribir el número a es conocida como la *notación científica*. $M = \pm a_0 . a_1 a_2 \ldots$ puede ser un número con una cantidad infinita de dígitos.

Ejemplo 1.16: En la Tabla 1.5, presentamos algunos números reales representados en la notación científica.

$230789 = 2.30789 \times 10^5$	$(0.000110011\cdots)_2 = (1.10011\cdots)_2 \times 2^{-4}$	$(0.1)_8 = 8^{-1}$
$0.1 = 1.0 \times 10^{-1}$	$(10)_2 = 2^1$	$(0.01)_8 = 8^{-2}$
$\sqrt{2} = 1.4142\cdots \times 10^0$	$(100)_2 = 2^2$	$(10)_{16} = 16^1$
$\pi = 3.1415\cdots \times 10^0$	$(0.1)_2 = 2^{-1}$	$(100)_{16} = 16^2$
$0.00005001 = 5.001 \times 10^{-5}$	$(0.01)_2 = 2^{-2}$	$(0.1)_{16} = 16^{-1}$
$(0.1011)_2 = (1.011)_2 \times 2^{-1}$	$(10)_8 = 8^1$	$(0.01)_{16} = 16^{-2}$
$(0.00000AB8)_{16} = (A.B8)_{16} \times 16^{-6}$	$(100)_8 = 8^1$	$(10000)_{16} = 16^4$

Tabla 1.5. Algunos números reales representados en la notación científica.

Con la notación científica se evitan escribir los ceros de la derecha del punto decimal. Una computadora o calculadora no puede leer el número real a representado en la forma científica (1.7), sino que lee el número:

$$a^* = m \times \beta^e = \pm \alpha_0.\alpha_1\alpha_2 \cdots \alpha_{t-1}\alpha_t^* \times \beta^e \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (1.8)$$

llamado *representación de punto flotante* para el número real a. En otras literaturas, se prefiere elegir $\alpha_0 = 0$ y $0 < \alpha_1 < \beta$ lo que conduce a otra notación de punto flotante.

$m = \pm \alpha_0.\alpha_1\alpha_2 \cdots \alpha_{t-1}\alpha_t^*$, donde $\alpha_i = a_i$ para $i = 0, 1, 2, \ldots, t\text{-}1$ y α_t^* es leída como:

1. *Si hay truncamiento (redondeo inferior)*: $\alpha_t^* = a_t$,
2. *Si hay redondeo (redondeo superior)*: $\alpha_t^* = a_t$ si $a_{t+1} < \beta/2$ ó si $a_{t+1} = \beta/2$ y $a_{t+k} = 0 \; \forall \, k \geq 2$ y a_t es par. O bien, $\alpha_t^* = a_t + 1$ si $a_{t+1} \geq \beta/2$ ó si $a_{t+1} = \beta/2$ y $a_{t+k} = 0 \; \forall \, k \geq 2$ y a_t es impar. (En este caso, α_i no siempre será igual a a_i para $i = 0, 1, 2, \ldots, t\text{-}1$; ya que pueden ser modificados de acuerdo con el resultado de $a_t + 1$).

El número t en (1.8), es llamado *precisión de la máquina* (en una computadora de 32 bits en precisión simple, $\beta = 2$ y $t = 23$, lo que equivale a $t = 7$ para $\beta = 10$). El número $m = \pm \alpha_0.\alpha_1\alpha_2 \cdots \alpha_{t-1}\alpha_t^*$, se llama *mantisa* o *significando* del número a y es el redondeo de M a $t+1$ dígitos. El número e en (1.8) es llamado el *exponente* y es un número entero decimal. Los dígitos a la derecha del punto en m se llaman *fracción*. Así que el número punto flotante a^* en (1.8) tiene $t+1$ dígitos en la mantisa y t dígitos en la fracción. Podemos decir que a^* es igual al número real a redondeado a $t+1$ dígitos. Además

$$0 < a_0 < \beta \quad \Rightarrow \quad 1 \leq |m| < \beta.$$

Lo anterior significa que los números punto flotante están normalizados. No se almacenan los ceros que están al comienzo del número. La parte entera de m tiene solamente un dígito. El tamaño del almacenamiento que se reserva para el exponente, determina el rango del número que puede ser representado. Los límites de e se pueden escribir como $L \leq e \leq U$, donde L y U son enteros negativo y positivo respectivamente. Si el resultado de un cálculo es un número punto flotante con $e > U$, la máquina da como salida una señal de error. Este tipo de error es llamado overflow. El error correspondiente con $e < L$ es el underflow. Este último error no es tan grave como el overflow.

Ejemplo 1.17: Presentamos en la Tabla 1.6, algunos números reales con su representación científica. y su representación de punto flotante con precisión cuatro y redondeo (redondeo superior). Se ha elegido la representación de punto flotante con precisión cuatro a manera de ejemplo, porque en la práctica las computadoras son de mayor precisión.

Número *a*.	**Representación Científica del número *a*.**	**Representación del Punto Flotante a^*. Precisión t=4 y redondeo.**
23078900	2.3078900×10^7	2.3079×10^7
23078500	2.3078500×10^7	2.3078×10^7
23077500	2.3077500×10^7	2.3078×10^7
23079500	2.3079500×10^7	2.3080×10^7
23075501	2.3075501×10^7	2.3076×10^7
$\frac{1}{8} = 0.125$	$\frac{1}{8} = 1.25 \times 10^{-1}$	$\left(\frac{1}{8}\right)^* = 1.2500 \times 10^{-1}$
$\sqrt{2} = 1.41421\cdots$	$\sqrt{2} = 1.41421\cdots \times 10^0$	$\left(\sqrt{2}\right)^* = 1.4142 \times 10^0$
$\pi = 3.14159\cdots$	$\pi = 3.14159\cdots \times 10^0$	$\pi^* = 3.1416 \times 10^0$
0.00005001	$0.00005001 = 5.001 \times 10^{-5}$	$(0.00005001)^* = 5.0010 \times 10^{-5}$
$(0.1011)_2$	$(0.1011)_2 = (1.011)_2 \times 2^{-1}$	$(0.1011)_2^* = (1.0110)_2 \times 2^{-1}$
$(0.00000AB8)_{16}$	$(0.00000AB8)_{16} = (A.B8)_{16} \times 16^{-6}$	$(0.00000AB8)_{16}^* = (A.B800)_{16} \times 16^{-6}$

Tabla 1.6. Ejemplos de representación científica y punto flotante.

Es importante observar que en cada caso en el Ejemplo 1.17 . se está aplicando el criterio de redondeo superior. ■

Un conjunto de números punto flotante está únicamente caracterizado por la base β. la precisión t. y los enteros L y U (que determinan los límites para los exponentes). En lo sucesivo denotaremos este conjunto por $Fl(\beta, t, L, U)$. De esta

manera se ha establecido un sistema numérico, con el que la computadora puede operar y con los cuales hace aritmética.

Definimos un **Sistema de Punto Flotante,** como el conjunto de números de punto flotante normalizados, en el sistema de números con base β con t dígitos para la fracción (equivalentemente $t+1$ dígitos en la mantisa), es decir, el conjunto de números de la forma:

$$Fl(\beta, t, L, U) = \left\{ a = \pm\, a_0.a_1a_2\ldots a_t \times \beta^e \left| \begin{array}{l} 0 < a_0 < \beta, \quad L \le e \le U, \\ 0 \le a_i < \beta, \quad i = 1,2,\ldots,t. \end{array} \right. \right\} \cup \left\{ 0.\underbrace{00\ldots0}_{t\text{-veces}} \right\}$$

La cantidad de números de punto flotante, que admite el sistema numérico $Fl(\beta, t, L, U)$ es $2(\beta-1)\beta^{t}(U-L+1)+1$.

Es importante enfatizar, que en el "sistema de números de punto flotante $Fl(\beta, t, L, U)$", significa que la fracción tiene t dígitos. En otras literaturas, los números de punto flotante son normalizados, de modo que, $\beta^{-1} \le |m| < 1$ y aquí t dígitos son usados para la mantisa. Nuestra notación es consistente con el IEEE* Estándar, para la aritmética de punto flotante.

Diferentes computadoras usan distintas técnicas para compactar los números, pero el procedimiento general es el mismo. En las computadoras, los números punto flotante tienen tres partes importantes: el *signo* (requiere un bit), la *mantisa* mejor conocido como *significando*, y la *parte exponencial* conocido como la *característica*. Las tres partes de los números tienen una longitud total fija que con frecuencia es de 32 o 64 bits (algunas veces más). La mantisa ocupa la mayor parte de estos bits, 23 y muchas veces 52 bits, y este número de bits determina la precisión de la representación. La parte exponencial usa 7 y muchas veces 11 bits, y este número determina el rango de los valores. Un esquema estándar es la que se muestra en la Figura 1.8.

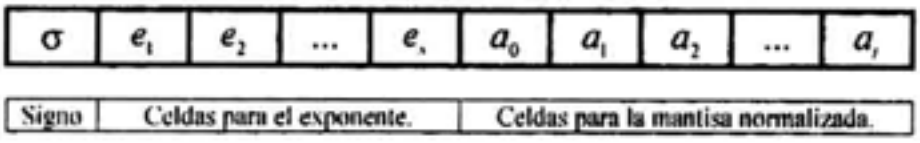


Figura 1.8. Esquema de representación de números punto flotante en una computadora.

* The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, New York 10017.

Donde σ es el signo de la mantisa. En el caso del sistema binario, $a_0 = 1$ siempre, por lo que no es necesario almacenarlo, y esto permite ganar una celda más para la fracción, almacenado $t+1$ dígitos binarios en la fracción.

La mayoría de las computadoras registran dos y hasta tres tipos de números: *precisión simple*, que equivale de 6 a 7 dígitos decimales significativos; *precisión doble*, equivalente de 13 a 14 dígitos decimales significativos; y *precisión extendida*, que puede ser equivalente de 19 a 20 dígitos decimales significativos. Una tabla comparativa de los métodos de almacenamiento de números punto flotante aparece en Gerald, C.F. y Wheatley, P.O. (1999) [7] . Parte de ella es la Tabla 1.7 siguiente:

Método.	Longitud total.	Bits en la fracción.	Bits en el exponente.	Valor sesgado***.	Base.	Exponente máximo.	Exponente mínimo.	Dígitos decimales significativos.
IEEE:								
Simple	32	23*	8	127	2	127	– 126	7
Doble	64	52*	11	1023	2	1023	– 1022	16
Extendido	80	64	15	16383	2	16383	– 16382	19
VAX:								
Simple	32	23	8	127	2	127	–127	7
Doble 1	64	55	8	127	2	127	–127	16
Doble 2	64	52	11	1023	2	1023	–1023	15
Extendido	128	112	15	16383	2	16383	–16383	33
IBM:								
Simple	32	24	7	63	16	63	– 64	7
Doble	64	56	7	63	16	63	– 64	16
Extendido	128**	112	7	63	16	63	– 64	33

Tabla 1.7. Métodos comparativos de almacenamiento de números punto flotante.

La cantidad de números punto flotante que admite el IEEE Estándar es de 4 261 412 865 (cuatro mil doscientos sesenta y un millones cuatrocientos doce mil ochocientos sesenta y cinco) ¿Cómo se puede calcular dicha cantidad?.

*** El valor sesgado se añade al exponente para almacenar (e+127 por ejemplo), así que los exponentes negativos son almacenados como enteros positivos.

* Más 1 "bit oculto"

** 8 bits no usados.

Cuatro formatos para la aritmética de punto flotante binario, fueron propuestos por el IEEE en 1979, y adoptados como formato estándar en 1985 con algunos cambios por otras compañías que fabrican computadoras, como el *American National Standards Institute.* Adoptaron el *ANSI/IEEE Standard* 754-1985 para la Aritmética de Punto Flotante Binario. Esta fue la culminación de casi una década de trabajo de un grupo de 92 personas, entre matemáticos, científicos e ingenieros de universidades, fabricantes de computadoras y compañías de microprocesadores.

A pesar de todo, fue el desarrollo de los microcomputadores lo que hizo necesario estandarizar la aritmética de punto flotante. Una ventaja fue facilitar la portabilidad, es decir, que fuera posible correr un mismo programa sin cambios en diferentes computadoras. Si dos computadoras conforman el estándar, entonces la ejecución del mismo programa en las dos computadoras daría resultados idénticos. Esto no sería posible para las computadoras que no tienen aritmética estándar.

Todas las computadoras diseñadas en los últimos 15 años usan la aritmética de punto flotante IEEE (los microcomputadores INTEL 8087 y Motorola 68881 fueron los primeros en implementarlo). Esto no significa que todas ellas encuentran exactamente los mismos resultados, porque existe algo de flexibilidad en el estándar. Pero esto significa que nosotros tenemos un modelo de máquina independiente de cómo se comporta la aritmética de punto flotante. MATLAB usa el formato IEEE *de precisión doble*. Existe también un formato de *precisión simple* que ahorra espacio pero no es muy rápido en máquinas modernas. Y existe un formato de *precisión extendida*, que es opcional y por lo tanto es una de las razones de la falta de uniformidad entre máquinas diferentes, Moler, C.B. (1996) [8].

El formato básico de precisión simple (32 bits), está implementado en casi todas las máquinas. Está diseñado dentro del sistema operativo. Para el formato estándar del IEEE, el número positivo más grande que se puede almacenar o representar es el número:

$$\underbrace{1.11\ldots1}_{23\text{ - veces.}} \times 2^{127} = 2^{127}\left(2 - 2^{-23}\right) \cong 3.4028235 \times 10^{38}$$

Y el número positivo más pequeño que se puede almacenar es:

$$\underbrace{1.00\ldots0}_{23\text{ - veces.}} \times 2^{-126} = 2^{-126} \cong 1.1754944 \times 10^{-38}.$$

1.7 DISTRIBUCIÓN DE NÚMEROS PUNTO FLOTANTE

Los números punto flotante no son uniformemente distribuidos sobre el eje real, como en el caso continuo. Veamos para el sistema $Fl(2, 2, -2, 1)$. Los números positivos binarios de este sistema, convertidos a decimales se muestran en la Tabla 1.8.

$(0.00)_2 \times 2^0 = 0$	$(1.01)_2 \times 2^{-1} = \left(1+\frac{1}{4}\right)\times\frac{1}{2} = \frac{5}{8}$	$(1.11)_2 \times 2^0 = \frac{7}{4}$
$(1.00)_2 \times 2^{-2} = 1 \times \frac{1}{4} = \frac{1}{4}$	$(1.10)_2 \times 2^{-1} = \left(1+\frac{1}{2}\right)\times\frac{1}{2} = \frac{3}{4}$	$(1.00)_2 \times 2^1 = 2$
$(1.01)_2 \times 2^{-2} = \left(1+\frac{1}{4}\right)\times\frac{1}{4} = \frac{5}{16}$	$(1.11)_2 \times 2^{-1} = \left(1+\frac{1}{2}+\frac{1}{4}\right)\times\frac{1}{2} = \frac{7}{8}$	$(1.10)_2 \times 2^1 = 3$
$(1.10)_2 \times 2^{-2} = \left(1+\frac{1}{2}\right)\times\frac{1}{4} = \frac{3}{8}$	$(1.00)_2 \times 2^0 = 1$	$(1.11)_2 \times 2^1 = \frac{7}{2}$
$(1.11)_2 \times 2^{-2} = \left(1+\frac{1}{2}+\frac{1}{4}\right)\times\frac{1}{4} = \frac{7}{16}$	$(1.01)_2 \times 2^0 = \frac{5}{4}$	$(1.01)_2 \times 2^1 = \frac{5}{2}$
$(1.00)_2 \times 2^{-1} = 1 \times \frac{1}{2} = \frac{1}{2}$	$(1.10)_2 \times 2^0 = \frac{3}{2}$	

Tabla 1.8. Números positivos binarios del sistema $Fl(2, 2, -2, 1)$ convertidos a decimales.

Hay 16 números positivos. En total hay 33 números punto flotante en este sistema incluyendo los números negativos y el cero. La distribución de los números positivos es como se muestra en la Figura 1.9. Entre los números flotantes 2^e y 2^{e+1}, los números punto flotante están uniformemente distribuidos con una separación de 2^{e-2}. Cuando e se incrementa, las separaciones también se incrementan como se observa en la Figura 1.9. La separación entre los números punto flotante que están entre 1 y 2 en nuestro sistema es 2^{-2} o 1/4. En el sistema completo IEEE Estándar, esta separación es 2^{-52}. Matlab llama a esta cantidad *eps*, el cual es conocido como el *épsilon de la máquina* ($eps = 2^{-52}$). Antes del IEEE es-

tándar, máquinas diferentes tenían diferentes valores de *eps*. El valor decimal aproximado de *eps* es 2.2204×10^{-16}. Uno u otro, $\frac{eps}{2}$ o eps pueden ser llamados *nivel de redondeo.*

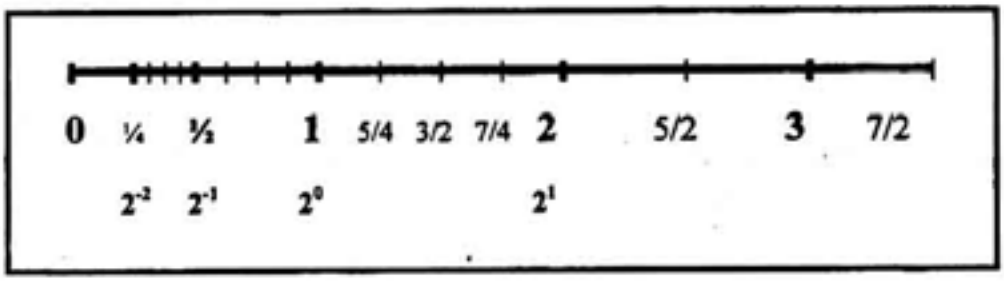


Figura 1.9. Distribución de números punto flotante positivos del sistema $Fl(2, 2, -2, 1)$.

1.8 ERRORES POR REDONDEO EN ARITMÉTICA DE PUNTO FLOTANTE

Teorema 1.1: Sea $Fl(\beta, t, L, U)$ un sistema de punto flotante.

i. Las cotas para el error absoluto y relativo en el caso de redondeo son:

$|\Delta a| \le \frac{1}{2}\beta^{e-t}$, donde *e* depende de *a*.

$\left|\frac{\Delta a}{a}\right| \le \mu = \frac{1}{2}\beta^{-t}$, μ se llama unidad o nivel de redondeo de la máquina.

ii. Si hay truncamiento: $|\Delta a| \le \beta^{e-t}$ y $\left|\frac{\Delta a}{a}\right| \le \mu_t = \beta^{-t}$.

Observaciones

1. La unidad de redondeo para el caso de truncamiento es mayor que en el caso redondeo, por lo que una máquina que trunca es menos precisa.
2. La unidad de redondeo μ es independiente del número *a*, por lo que números grandes y pequeños son representados con la misma exactitud relativa.
3. Si se quiere aproximar un número hasta $t+1$ dígitos significativos correctos, es suficiente con pedir que la magnitud del error relativo sea $\le$

$\frac{1}{2}\times 10^{-t}\left(=5\times 10^{-t-1}\right)$, o que el tamaño del error absoluto sea $\leq \frac{1}{2}\times 10^{e-t}$ si se conoce el orden *e* del número.

4. En Matlab el épsilon de la máquina es $eps = 2^{-52} \cong 2.2204\times 10^{-16}$ y la unidad de redondeo es $\mu = \frac{eps}{2} = \frac{1}{2}\times 2^{-52} \cong 0.11102\times 10^{-15}$, lo que significa que despliega 15 decimales correctamente redondeados en la fracción.

Demostración del Teorema 1.1: Sea a un número real distinto de cero. Entonces $a = m\beta^e$, con $1 \leq |m| < \beta$, $m = \pm a_0.a_1a_2\cdots a_{t-1}a_ta_{t+1}\cdots$, $a_0 \neq 0$ & $0 \leq a_i < \beta$. Como $a^* = m^*\beta^e$, con m^* igual a m redondeado a $t+1$ dígitos, entonces $|m^* - m| \leq \frac{1}{2}\beta^{-t}$. (Para el caso truncamiento $|m^* - m| \leq \beta^{-t}$), por lo tanto,

$$|\Delta a| = |a^* - a| = |m^* - m|\beta^e \leq \frac{1}{2}\beta^{-t}\beta^e = \frac{1}{2}\beta^{e-t},$$

de donde, $|R_a| = \frac{|\Delta a|}{|a|} \leq \frac{\frac{1}{2}\beta^{-t}\beta^e}{|m|\beta^e} \leq \frac{1}{2}\beta^{-t}$, $\left(\text{pues, } \frac{1}{|m|} \leq 1\right)$. ■

Ejemplo 1.18: Para el sistema punto flotante $FI(2, 23, -126, 127)$ del IEEE estándar, la unidad de redondeo es $\mu = \frac{1}{2}\times 2^{-23} = 2^{-24} \cong 0.596\times 10^{-7}$ (esto significa que los resultados en este sistema, tienen 7 dígitos decimales correctamente redondeados en la fracción). El épsilon de la máquina es:

$$eps = 2^{-23} \cong 1.1920929\times 10^{-7},$$

que representa la separación entre los números punto flotante que están entre los números flotantes 1 y 2 del sistema. Recuerde que el número positivo más pequeño que se puede representar en este sistema es el:

$$1.\underbrace{00\cdots 0}_{23\text{ veces}}\times 2^{-126} = 2^{-126} \cong 1.1754944\times 10^{-38},$$

y el número positivo más grande es el:

$$1.\underbrace{11\ldots 1}_{23\text{ veces}}\times 2^{127} = 2^{127}\left(2 - 2^{-23}\right) \cong 3.4028235\times 10^{38}.$$

La representación geométrica de este sistema se muestra en la Figura 1.10.

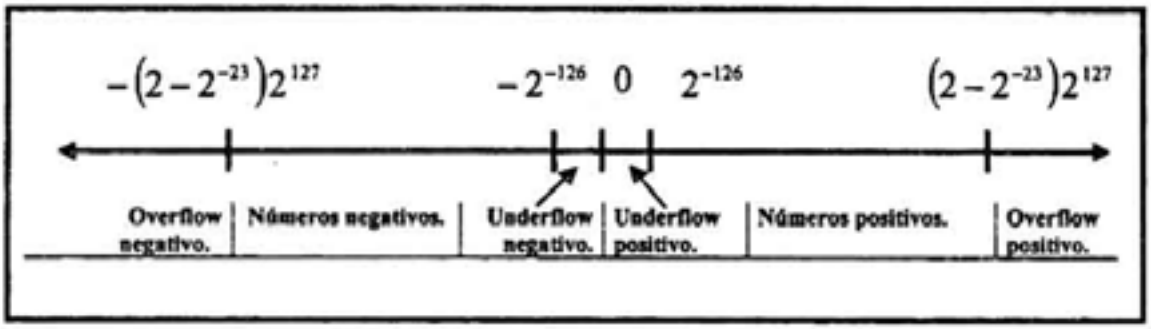


Figura 1.10. Números punto flotante del formato estándar IEEE $Fl(2, 23, -126, 127)$.

¿Cuántos dígitos decimales corresponden a $t+1$ dígitos binarios?
Para responder a la pregunta, resolvemos la ecuación:

$$\frac{1}{2}\times 2^{-t} = \frac{1}{2}\times 10^{-s} \quad \Rightarrow \quad s = t\log_{10}(2) \cong 0.3t.$$

De donde obtenemos la siguiente regla: t dígitos binarios en la fracción corresponden a $0.3t$ dígitos decimales en la fracción, y s dígitos decimales en la fracción corresponden a $3.3s$ dígitos binarios en la fracción.

Para $t = 23$, $s = (0.3)(23) = 6.9 \cong 7$ dígitos decimales en la fracción. Para el caso de Matlab, $t = 52$ por lo que $s = (0.3)(52) = 15.6 \cong 15$ dígitos decimales en la fracción.

1.9 CANCELACIÓN NUMÉRICA EN LAS OPERACIONES ARITMÉTICAS DE PUNTO FLOTANTE

Consideramos un número real a distinto de cero, y a^* o $fl(a)$ su correspondiente número flotante, es decir, a^* o $fl(a)$ es el redondeo de a en el sistema $Fl(\beta, t, L, U)$. El error absoluto de a es $\Delta a = fl(a) - a$. Se analizarán con detalle, ejemplos de operaciones aritméticas donde se presenta la cancelación numérica. Para un estudio más riguroso sobre las operaciones aritméticas de punto flotante, se puede ver en Eldén, L. and Wittmeyer-Koch, L. (1990) [6]. Un enunciado equivalente al Teorema 1.1 y que es especialmente adecuado para el análisis de acumulación de los errores de redondeo es el siguiente:

Teorema 1.2: $\left|\frac{\Delta a}{a}\right| \le \mu = \frac{1}{2}\beta^{-t} \quad \Leftrightarrow \quad \exists\ \delta \ni fl(a) = a(1+\delta)$ con $|\delta| \le \mu$.

Para ver cómo se comportan los errores por redondeo en las cuatro operaciones aritméticas básicas, denotemos por $\otimes$ a cualquiera de ellas (es decir, $\otimes = +, -, *, /$) y por el Teorema 1.2, la aritmética de punto flotante se considera buena si se tiene:

$$fl(a \otimes b) = (a \otimes b)(1+\delta) \quad \text{con} \quad |\delta| \le \mu .$$

Observe que sucede lo mismo para el resultado de una operación que lo que ocurre para un número, sin embargo este resultado puede ser una buena aproximación del número exacto (así como los números de punto flotante son una buena aproximación del número exacto cuando este es introducido en la computadora). Y decimos que "puede" porque con este tipo de aritmética se presenta un fenómeno que en análisis numérico se denomina cancelación numérica; ésta se presenta específicamente en la resta. Veamos más de cerca. Sean $a, b \in Fl(\beta, t, L, U)$ tales que:

$$a = a_0.a_1a_2\cdots a_r a_{r+1}\cdots a_t \times \beta^e \quad \text{y} \quad b = a_0.a_1a_2\cdots a_r b_{r+1}\cdots b_t \times \beta^e ,$$

entonces,

$$fl(a-b) = 0.00\cdots 0c_{r+1}\cdots c_t \times \beta^e = c_{r+1}.c_{r+2}\cdots c_t \times \beta^{e-(r+1)} .$$

Esto quiere decir que al hacer la resta de a y b se pierden las cifras más significativas (las que aseguraban la buena aproximación del número), y el resultado no necesariamente tiene que ser una buena aproximación.

Ejemplo 1.19: Sean a = 1.99999935 y b = 1.99999923. Calcular $a - b$ en una aritmética de punto flotante con β = 10, t = 7 *y redondeo*, y compararlo con el resultado exacto.

Solución: En el sistema de precisión 7 y redondeo tenemos que $a^* = fl(a) = 1.9999994$,

$b^* = fl(b) = 1.9999992$ y $fl(a^* - b^*) = 0.0000002 = 2.0 \times 10^{-7}$.

El resultado exacto es $a - b = 0.00000012 = 1.2 \times 10^{-7}$. (No son iguales ni por redondeo en el primer dígito de la fracción.) ■

El error relativo de un valor aproximado es una medida de la información contenida en la aproximación. Medir una cantidad, por ejemplo, la distancia entre dos puntos, con seis dígitos significativos requiere equipos más avanzados que para medir la misma cantidad con dos dígitos significativos. Cuando la aproxima-

ción es usada en un cálculo numérico, es importante no destruir información por usar algoritmos malos.

Ejemplo 1.20: Calcular las soluciones de la ecuación cuadrática $ax^2+bx+c=0$, con una aritmética de punto flotante: $\beta=10$, $t=7$ y *redondeo*, y con los coeficientes $a=1$, $b=10^5$ y $c=1$.

Solución: Primero usamos la fórmula general $x=\dfrac{-b\pm\sqrt{\Delta}}{2a}$, con $\Delta=b^2-4ac$, $a\neq 0$.

$$\Delta^* = fl(\Delta)= fl\left(\left(10^5\right)^2-4(1)(1)\right)= fl\left(10^{10}-4\right)=10^{10}.$$

$$x_1^* = fl(x_1)= fl\left(\frac{-10^5-fl\left(\sqrt{10^{10}}\right)}{2}\right)= fl\left(\frac{-10^5-10^5}{2}\right)=-10^5.$$

$$x_2^* = fl(x_2)= fl\left(\frac{-10^5+fl\left(\sqrt{10^{10}}\right)}{2}\right)= fl\left(\frac{-10^5+10^5}{2}\right)=0.$$

Las soluciones exactas son $x_1=-9.999999999\times10^4$ y $x_2=-1.000000338\times10^{-5}$. x_2 no coincide ni por redondeo con x_2^*. Esto quiere decir que algo anda mal. Además, se sabe que las raíces y los coeficientes de la ecuación tienen las siguientes relaciones, conocidas como Fórmulas de Vieta:

$$x_1+x_2=-\frac{b}{a}=-10^5 \quad \text{y} \quad x_1x_2=\frac{c}{a}=1.$$

Pero, $x_1^*x_2^*=0\neq 1=\dfrac{c^*}{a^*}$, por lo que los resultados son inaceptables. La razón es que en la fórmula general para obtener las raíces hay cancelación numérica, ya que si $b>0$, al calcular $x_2=\dfrac{-b+\sqrt{\Delta}}{2a}$ hay una resta, y si $b<0$, al calcular $x_1=\dfrac{-b-\sqrt{\Delta}}{2a}$ también se hace una resta ¿Qué hacer en estos casos?. Quitar la resta que causa problemas (no todas las restan presentan esta dificultad).
Otra forma de calcular x_1^* y x_2^* es el siguiente: Como $b=10^5>0$, así que primero calculamos x_1 (que no tiene resta al usar la fórmula general): $x_1^*=fl(x_1)=-10^5$. Ahora, calculamos x_2 utilizando la segunda fórmula de la relación con los coeficientes:

$$x_2^* = fl(x_2) = fl\left(\frac{c^*}{x_1^* a^*}\right) = fl\left(\frac{1}{-10^5 \times 1}\right) = -10^{-5} \quad \text{(es diferente de cero).}$$

Observamos que

$$x_1^* + x_2^* = -10^5 - 10^{-5} = -10^5 = -\frac{b^*}{a^*} \quad \text{y} \quad x_1^* x_2^* = \left(-10^5\right)\left(-10^{-5}\right) = 1 = \frac{c^*}{a^*}.$$

Así que las soluciones $x_1^* = -10^5$ y $x_2^* = -10^{-5}$ están correctamente calculados. ■

Ejemplo 1.21: Evaluar la función $f(x) = \sqrt{x^2+1} - x$, para $x \geq 1$. Por ejemplo, $x = 10^5$, en una aritmética de punto flotante con $\beta = 10$, $t = 7$ y *redondeo*.

Solución: Primero calculamos $fl(x^2+1) = fl\left((10^5)^2 + 1\right) = fl(10^{10}+1) = 10^{10}$, entonces.

$$f^*(10^5) = fl\left(\sqrt{10^{10}} - 10^5\right) = fl(10^5 - 10^5) = 0.$$

De nuevo la cancelación numérica es por la resta, y se puede evitar aplicando la siguiente transformación:

$$f(x) = \left(\sqrt{x^2+1} - x\right)\frac{\sqrt{x^2+1}+x}{\sqrt{x^2+1}+x} = \frac{1}{\sqrt{x^2+1}+x} \quad \Rightarrow \quad f(x) = \frac{1}{\sqrt{x^2+1}+x}.$$

Se ha eliminado la resta en la última expresión. Evaluamos de nuevo,

$$f^*(10^5) = fl\left(\frac{1}{\sqrt{10^{10}}+10^5}\right) = fl\left(\frac{1}{2\times 10^5}\right) = fl\left(0.5\times 10^{-5}\right) = 5.0\times 10^{-6},$$

es distinto de cero, y es el redondeo correcto del resultado exacto. ■

Ejemplo 1.22: Matlab 5.3 que opera con precisión doble, dibuja la gráfica del polinomio $y = x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1$ sobre el intervalo $0.988 \leq x \leq 1.012$, como se muestra en la Figura 1.11.

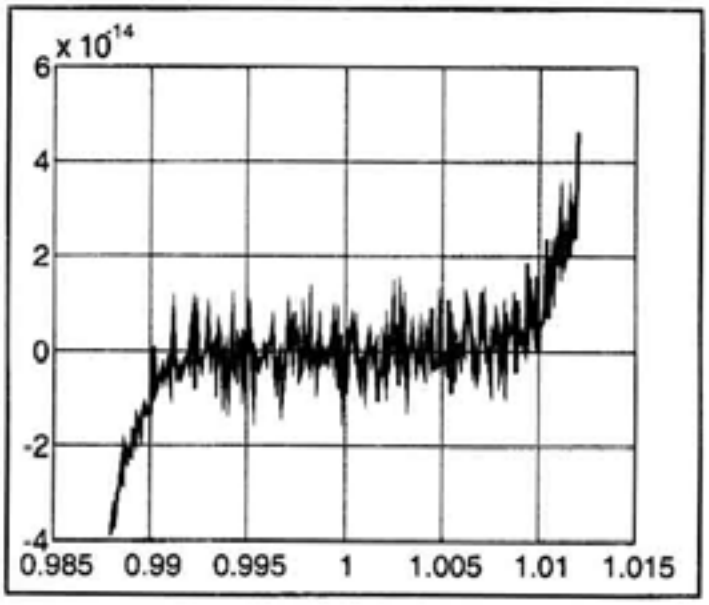


Figura 1.11. Visualización gráfica de la acción del error por redondeo.

La gráfica resultante no se parece nada a la de un polinomio. No está liso. *Estás viendo error por redondeo en acción.* El factor de escala en el eje y es pequeño, 10^{-14}. Los valores pequeños de y están siendo calculados por sumas y diferencias de números grandes como -35×1.012^4. Existe una cancelación substractiva severa. Si los valores de y son calculados por la expresión equivalente $y=(x-1)^7$, entonces, resulta una gráfica lisa (pero muy aplanada) como se muestra en la Figura 1.12.

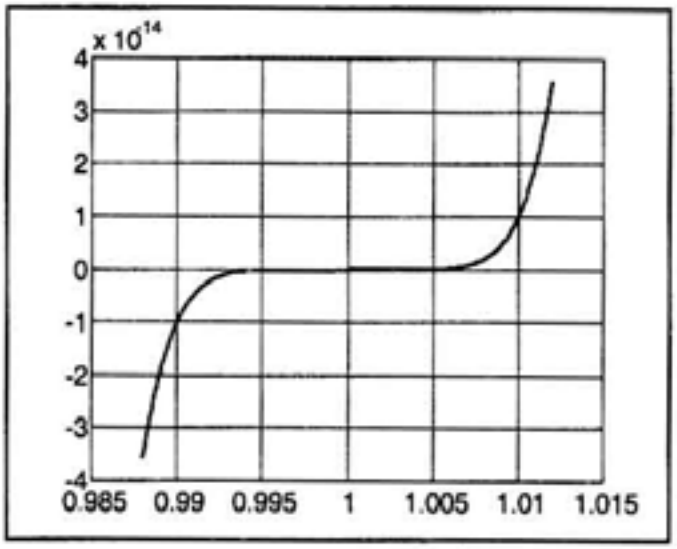


Figura 1.12. Gráfica del polinomio $y=(x-1)^7$.

Otras observaciones

1. Rara vez se da una prueba para la igualdad entre números punto flotante. Si en el programa se empieza declarando que x y y son reales,

$$if\ x = y\ \ then\ ....$$

y los resultados son de un cálculo primitivo, entonces la probabilidad de que la expresión boleana sea verdadera es muy pequeña. Se podría escribir la expresión como:

$$if\ abs(x - y) < eps\ \ then$$

donde *eps* puede ser el épsilon de la máquina o algún épsilon constante.

2. A consecuencia de los errores en la aritmética de punto flotante, las leyes de la aritmética usual no se cumplen en general. Por ejemplo, las leyes asociativas para la adición y la multiplicación no se cumplen. Consideramos los números:

$$a = 9.876 \times 10^4,\ \ b = -9.880 \times 10^4 \quad y \quad c = 3.456 \times 10^1,$$

en el sistema flotante $Fl(10, 3, -9, 9)$.

$$fl(a+b) = -4.000 \times 10^1,\ \ fl(b+c) = -9.877 \times 10^4,$$

$$fl\big(fl(a+b)+c\big) = -5.440 \times 10^0 = -5.440,$$

mientras que, $fl\big(a + fl(b+c)\big) = -10 \quad \therefore\ fl(fl(a+b)+c) \neq fl(a + fl(b+c))$.

1.10 ACUMULACIÓN DE ERRORES

En esta sección calcularemos la suma $S_n = \sum_{k=1}^{n} x_k$ de números positivos, para ver un ejemplo de acumulación del error en las operaciones repetidas de punto flotante. De acuerdo con el Teorema 1.2 de la Sección 1.9, es práctico usar la fórmula siguiente para la estimación del error:

$$fl\,(a + b) = (a + b)\,(1 + \varepsilon),\ \ \text{para algún}\ \ |\varepsilon| \leq \mu\,.$$

Si calculamos la suma en el orden natural ($\overline{S}_i$ denota la suma parcial calculada), entonces

$$\overline{S}_1 = x_1 \ \ y\ \ \overline{S}_i = fl\big(\overline{S}_{i-1} + x_i\big),\ \ \ i = 2,3,....,n\,;$$

se encuentra inmediatamente que: $\bar{S}_i = (\bar{S}_{i-1} + x_i)(1+\varepsilon_i)$, $i = 2,3.....n$.

Usando un argumento de inducción obtenemos: $\bar{S}_n = \bar{x}_1 + \bar{x}_2 + ... + \bar{x}_{n-1} + \bar{x}_n$, donde:

$$\bar{x}_1 = x_1(1+\varepsilon_2)(1+\varepsilon_3)...(1+\varepsilon_n),$$

$$\bar{x}_2 = x_2(1+\varepsilon_2)(1+\varepsilon_3)...(1+\varepsilon_n), \text{ en general,}$$

$$\bar{x}_i = x_i(1+\varepsilon_i)(1+\varepsilon_{i+1})...(1+\varepsilon_n), \quad i = 3,4,....,n.$$

Por ejemplo. $\bar{S}_2 = (\bar{S}_1 + x_2)(1+\varepsilon_2) = x_1(1+\varepsilon_2) + x_2(1+\varepsilon_2) = \bar{x}_1 + \bar{x}_2$, y

$$\bar{S}_3 = (\bar{S}_2 + x_3)(1+\varepsilon_3) = x_1(1+\varepsilon_2)(1+\varepsilon_3) + x_2(1+\varepsilon_2)(1+\varepsilon_3) + x_3(1+\varepsilon_3) = \bar{x}_1 + \bar{x}_2 + \bar{x}_3.$$

Para obtener una estimación del error que sea práctico, necesitamos el siguiente lema cuya prueba es inmediata.

Lema: Sean $\varepsilon_1.\varepsilon_2,....,\varepsilon_n$ números que satisfacen $|\varepsilon_i| \le \mu$, $i = 1,2,....,n$, y suponemos que $n\mu < 0.1$. Entonces, existe un número δ_n tal que

$$(1+\varepsilon_1)(1+\varepsilon_2)\cdots(1+\varepsilon_n) = 1+\delta_n \text{ y } |\delta_n| \le 1.06n\mu.$$

De aquí se derivan dos tipos de resultados, que dan la estimación del error para la suma en aritmética de punto flotante.

Teorema 1.3: Análisis progresivo. Si $n\mu < 0.1$, entonces el error en la suma calculada puede ser estimada como

$$|\bar{S}_n - S_n| \le |x_1||\delta_{n-1}| + |x_2||\delta_{n-1}| + |x_3||\delta_{n-2}| + \cdots + |x_n||\delta_1|,$$

donde, $|\delta_i| \le (1.06)i\mu$. $i = 1,2,....,n-1$.

Demostración: De acuerdo con el Lema anterior, tenemos que

$$\bar{S}_n = x_1(1+\delta_{n-1}) + x_2(1+\delta_{n-1}) + x_3(1+\delta_{n-2}) + \cdots + x_n(1+\delta_1),$$

donde cada δ_i satisface la desigualdad que indica el teorema. Sustrayendo S_n y aplicando la desigualdad triangular se tiene el resultado. ■

El análisis de error progresivo o hacia adelante, es el tipo de análisis de error que hemos usado desde el comienzo del curso. Sin embargo, hay dificultades al usar este método para analizar un algoritmo fundamental como la eliminación Gaussiana para la solución de un sistema lineal de ecuaciones. En los años de 1950, J. H. Wilkinson fue el primero en hacer un análisis de error correcto de este algoritmo, usando el análisis de error regresivo o hacia atrás.

En el análisis de error regresivo, uno muestra que la solución aproximada $\bar{S}$ que ha sido calculada para el problema P, es la solución exacta de un problema perturbado $\bar{P}$. Citamos la siguiente descripción del propósito del análisis regresivo, de un libro de álgebra lineal numérica:

"El objetivo del análisis del error regresivo consiste en hacer un inquietante alto, cuando uno tiene la respuesta "exacta", porque no existe objeto bien definido en la mayoría de las situaciones del mundo real. Lo que uno quiere es encontrar una respuesta que sea la solución matemática verdadera de un problema que está dentro del dominio de incertidumbre del problema original. Cualquier resultado que hace esto debe ser aceptado como una respuesta del problema, al menos con la filosofía del análisis de error regresivo." (J. R. Rice, Matrix Computations and Mathematical Software, MacGraw-Hill, New York, 1981).

En el ejemplo de la suma, se formula el siguiente resultado

Teorema 1.4: Análisis regresivo. Suponemos que $n\mu < 0.1$, entonces tenemos que

$$\bar{S}_n = \bar{x}_1 + \bar{x}_2 + \ldots + \bar{x}_{n-1} + \bar{x}_n,$$

donde, $\bar{x}_1 = x_1(1+\delta_{n-1})$,

$$\bar{x}_i = x_i(1+\delta_{n-i-1}), \quad i = 2,3,\ldots,n$$

$$|\delta_i| \le (1.06)i\mu, \quad i = 1,2,\ldots,n-1.$$

Aplicamos los dos teoremas anteriores (Teoremas 1.3 y 1.4), a la estimación del error en la suma, podemos reacomodar de tal manera que

$$|\bar{S}_n - S_n| \le \left[(n-1)|x_1| + (n-1)|x_2| + (n-2)|x_3| + \cdots + 2|x_{n-1}| + |x_n|\right] 1.06\mu.$$

Inmediatamente vemos que para minimizar la cota del error, añadiremos los términos en la sumatoria en orden creciente, puesto que los primeros términos en la suma tienen multiplicadores grandes en la estimación del error. Veamos el siguiente ejemplo:

Ejemplo 1.23: Sean los números: $x_1 = 1.234 \times 10^1$, $x_2 = 3.453 \times 10^0$, $x_3 = 3.441 \times 10^{-2}$, $x_4 = 4.667 \times 10^{-3}$, y $x_5 = 9.876 \times 10^{-4}$.

Si realizamos la suma en orden decreciente en el sistema de punto flotante $Fl(10, 3, -9, 9)$, con redondeo (redondeo superior), tendremos:

$$x_1 + x_2 = (1.234 + 0.3453) \times 10^1 = 1.5793 \times 10^1 \doteq 1.579 \times 10^1,$$

$$(x_1 + x_2) + x_3 = (1.579 + 0.003441) \times 10^1 = 1.582441 \times 10^1 \doteq 1.582 \times 10^1,$$

$$((x_1 + x_2) + x_3) + x_4 = (1.582 + 0.0004667) \times 10^1 = 1.5824667 \times 10^1 \doteq 1.582 \times 10^1,$$

$$(((x_1 + x_2) + x_3) + x_4) + x_5 = (1.582 + 0.00009876) \times 10^1 = 1.58209876 \times 10^1 \doteq 1.582 \times 10^1.$$

Y si sumamos en orden creciente, tendremos:

$$x_4 + x_5 = (4.667 + 0.9876) \times 10^{-3} = 5.6546 \times 10^{-3} \doteq 5.655 \times 10^{-3},$$

$$x_3 + (x_4 + x_5) = (3.441 + 0.5655) \times 10^{-2} = 4.0065 \times 10^{-2} \doteq 4.006 \times 10^{-2},$$

$$x_2 + (x_3 + (x_4 + x_5)) = (3.453 + 0.04006) \times 10^0 = 3.49306 \times 10^0 \doteq 3.493 \times 10^0,$$

$$x_1 + (x_2 + (x_3 + (x_4 + x_5))) = (1.234 + 0.3493) \times 10^1 = 1.5833 \times 10^1 \doteq 1.583 \times 10^1.$$

El resultado correcto redondeado a seis decimales es $S_5 = 1.583306 \times 10^1$. Por lo que la suma en orden creciente resulta más exacta, pues primero se acumulan los números más pequeños; mientras que la suma en orden decreciente al acumularse las cantidades más grandes, ya no contribuyen las cantidades pequeñas y pierde exactitud la suma acumulada. Similarmente, un error relativo grande aparece cuando una serie que converge lentamente se suma en orden decreciente. ■

Como conclusión presentamos el siguiente **algoritmo para calcular la suma de N números positivos.**

Entrada: $N, x_1, x_2, ..., x_N$.
Salida: *SUMA.*

Paso 1: Ordenamos los números $x_1, x_2, ..., x_N$ de menor a mayor.

Para $i = 1, 2, \ldots, N-1$;
Pongamos $z = x_i$;

Para $j = i+1, \ldots, N$;

Si $x_i > x_j$, entonces intercambiamos posiciones: $x_i = x_j$ y $x_j = z$;

Paso 2: Sumamos ahora los números ya ordenados de menor a mayor.

$SUMA = 0$;
Para $i = 1, 2, \ldots, N$;

$SUMA = SUMA + x_i$

Paso 3: Salida *SUMA*.
Paso 4: Alto.

Ejemplo 1.24: Calculamos la suma $S = \sum_{n=1}^{N} \frac{1}{n}$ para varios valores de N.

Los resultados que se presentan para esta suma en la Tabla 1.9, se calcularon programando en Matlab 5.3 el algoritmo anterior. Los resultados son en precisión doble.

N	Decreciente	Creciente
1000	7.48547086055034	7.48547086055034
10 000	9.78760603604435	9.78760603604439
30 000	10.88618499211993	10.88618499211991
100 000	12.09014612986334	12.09014612986341
200 000	12.78329081042982	12.78329081042961

Tabla 1.9. Resultados de la suma $S = \sum_{n=1}^{N} \frac{1}{n}$ en orden creciente y decreciente.

Note la diferencia entre los resultados en los últimos dígitos para $N \geq 10000$.

1.11 PROBLEMAS

Instrucciones: Lea con toda atención cada uno de los problemas antes de empezar a resolverlos.

1. De la Tabla 1.1 que se da para la población de Tabasco, estime la población de 1985 usando solamente los datos de 1980 y 1990.
2. Las ecuaciones paramétricas $x = 3\cos(t)$ y $y = 2\text{sen}(t)$ con $0 \le t \le 2\pi$ representan una elipse. Dibuje la elipse y demuestre que su longitud total s está dada por la integral elíptica $s = \int_0^{2\pi} \sqrt{4 + 5\text{sen}^2(t)}\, dt$. ¿Podrías calcular el valor de s?
3. La gráfica de la ecuación polar $r = 2\cos(2\theta)$ se llama rosa de cuatro ramas o de cuatro pétalos. Dibuje la rosa y demuestre que su longitud total s está dada por la integral elíptica $s = 16 \int_0^{\pi/4} \sqrt{4 - 3\cos^2(2\theta)}\, d\theta$. ¿Cómo se podría determinar el valor de s?
4. Desarrolle paso a paso el Ejemplo 1.3 sobre ¿Cómo medir la respuesta cardiaca del corazón?
5. Investigue las ecuaciones que representan el modelo matemático para predecir la dinámica demográfica de dos especies que compiten por la misma comida. Consulte la Sección 5.9 del libro de Burden, R.L. y Faires, J.D. (1986) [4].
6. En el año 1225, Leonardo de Pisa estudió la ecuación $x^3+2x^2+10x-20=0$ y obtuvo la raíz x=1.368808107. Nadie sabe qué método utilizó Leonardo para encontrar este valor, aunque fue un resultado notable en ese tiempo. Dibuje la gráfica de la ecuación y observe que efectivamente tiene una raíz real simple entre 1 y 2. ¿Cómo le harías para verificar que el resultado de Leonardo es correcto en todas sus cifras?
7. Otro método numérico para calcular la raíz cúbica de 2 es

$$x_1 = 1, \quad x_{n+1} = \frac{2}{3}\left(x_n + \frac{1}{x_n^2}\right), \quad n \ge 1.$$

Escriba un algoritmo a partir de este método para estimar la raíz cúbica de 2 con 4 decimales correctos. Obtenga dicha raíz aplicando su algoritmo. Inicie cualquier otro valor para x_1, realice de nuevo los cálculos y observe el comportamiento del método.

8. Investigue un algoritmo que calcula el mínimo de un conjunto de N números.
9. Investigue un algoritmo que calcula el máximo de un conjunto de N números.
10. Investigue un algoritmo más eficiente que el presentado en el paso 1 del algoritmo de la Sección 1.10, para ordenar los elementos de un conjunto de N números de menor a mayor.
11. Programe los algoritmos investigados en los ejercicios 8, 9 y 10 en Matlab y pruébelo con varios ejemplos.
12. Investigue el algoritmo de la "Criba de Eratóstenes" que se usa para calcular números primos $\leq N$. Prográmelo en Matlab para determinar todos los números primos entre 1 y 10 000.
13. A partir de la "Criba de Eratóstenes", diseñe un algoritmo que determine cuándo un número entero positivo N es primo. Prográmelo en Matlab y determine el número primo más grande que pueda.
14. Elabore un algoritmo para calcular el factorial de un número entero positivo, prográmelo en Matlab y úselo para encontrar exactamente el factorial del mayor entero posible.
15. Elabore un algoritmo para la "división sintética" que evalúa un polinomio en un punto dado. Prográmelo en Matlab y pruebe su programa con varios ejemplos.
16. ¿Cuáles son los errores absolutos y relativos de la aproximación familiar 3.1416 para π?. Dar una cota adecuada para el error absoluto y relativo de la aproximación anterior.
17. Los números son exactos hasta dos lugares decimales cuando su error no excede de 0.005. Las siguientes raíces cuadradas se toman de una tabla. Redondee cada una hasta dos lugares decimales y anote el error de redondeo. ¿Cómo se comparan estos errores de redondeo con el máximo 0.005?

n	11	12	13	14	15	16	17	18	19	20
$\sqrt{n}$ tres decimales	3.317	3.464	3.606	3.742	3.873	4.000	4.123	4.243	4.359	4.472
$\sqrt{n}$ dos decimales	3.32	3.46								
Error de redondeo	+.003	−.004								

18. Determine el polinomio de Taylor de tercer grado para la función $f(x)=\sqrt{x+1}$ alrededor de $x_0=0$. Úselo para aproximar $\sqrt{0.5}$ y $\sqrt{1.5}$. Use 5 dígitos significativos en los cálculos. Explique dónde se presentan los

errores de aproximación y de redondeo. Estime el tamaño del error en cada caso.

19. a) Demuestre que la propagación del error para la adición y la resta $y = x_1 \pm x_2$ es

$$\Delta y = \Delta x_1 \pm \Delta x_2 \quad \text{y} \quad |\Delta y| \le |\Delta x_1| + |\Delta x_2|.$$

b) Demuestre la generalización de (*a*) para *n* de datos en la adición, es decir, si

$$y = \sum_{k=1}^{n} x_k, \quad \text{entonces} \quad \Delta y = \sum_{k=1}^{n} \Delta x_k \quad \text{y} \quad |\Delta y| \le \sum_{k=1}^{n} |\Delta x_k|.$$

20. Estime el tamaño de los errores absoluto y relativo en la aproximación $y = x_1 - x_2$, si $x_1 = 10.123455 \pm 0.5 \times 10^{-6}$ y $x_2 = 10.123789 \pm 0.5 \times 10^{-6}$. ¿Cuántos dígitos significativos tiene la aproximación *y*?

21. Estime el valor de $f(a) = \sqrt{a}$, para $a = 2.05 \pm 0.01$. Sugerencia: aplique el Teorema del Valor Medio.

22. Estime el valor de la distancia focal f de una lente usando la fórmula

$$\frac{1}{f} = \frac{1}{a} + \frac{1}{b}$$

donde $a = 32 \pm 1\ mm$ y $b = 46 \pm 1\ mm$. Dar una estimación para el error.

23. Deduzca una fórmula de propagación del error en el cálculo de la longitud

$$|x| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$$

de un vector $x = (x_1, x_2, \ldots, x_n)$.
Estime la longitud del vector cuyas componentes están dadas por:

$$x = -3.75 \pm 10^{-2}, \quad y = 2.413 \pm 3 \times 10^{-3} \quad \text{y} \quad z = 5.341 \pm 2 \times 10^{-3}.$$

24. Sea una función $f : \Omega \subseteq R^n \to R^m$ diferenciable sobre un conjunto abierto convexo Ω y suponemos que queremos calcular $f(a^*)$, donde el vector a^* es una aproximación del vector $a \in \Omega$. Aplique la fórmula general de propagación de error a cada componente de *f* para demostrar que $\Delta f \cong J\,\Delta a$, donde *J* es una matriz *m*×*n* con entradas $(J)_{ij} = \dfrac{\partial f_i}{\partial a_j}$.

25. Algoritmo de conversión de un número decimal a representación en base β.

    a) Parte entera: Divida la parte entera del número y cada cociente sucesivo por β hasta obtener un cociente cero. La sucesión de residuos, en orden inverso, da la representación en base β de la parte entera del número.

    b) Parte fraccionaria: Multiplique la parte fraccionaria del número y la parte fraccionaria de cada producto sucesivo, por β, hasta obtener una parte fraccionaria cero o una parte fraccionaria duplicada. Entonces la sucesión finita o la sucesión infinita periódica de partes enteras del producto da la representación en base β de la parte fraccionaria del número.

    Note que cada residuo en (a) es menor que β y por lo tanto, es un dígito en base β, y que cada parte entera en (b) también es menor que β y por lo tanto, es un dígito en base β. Convierta cada uno de los números decimales 684, 0.4704, 684.4704, 0.4703, 10.01 en bases 2, 3, 4, 5, 8, y 16.

26. Algoritmo de conversión de un número de base β a representación decimal.

    a) Parte entera: Multiplique el dígito que está más hacia a la izquierda por la base β y sume el siguiente dígito de la derecha. Multiplique la suma por la base β y súmesela al siguiente dígito. Repita el proceso hasta que el dígito que está más hacia la derecha sea sumado.

    b) Parte fraccionaria: Multiplique el dígito que está más hacia la derecha por 1/β y sume el siguiente dígito de la izquierda. Multiplique la suma por 1/β y sume el siguiente dígito. Repita el proceso hasta que el dígito que está más hacia la izquierda sea sumado y la suma multiplicada por 1/β. El producto final es el equivalente decimal que se quiere.

    La parte (b) presupone que la parte fraccionaria del número en base β termina. En (a) el proceso termina cuando el último dígito sea sumado, pero en (b) el proceso termina cuando el último dígito sea sumado y la suma sea multiplicada por 1/β. Transforme en decimal los siguientes números: $(2401.2314)_5$, $(17.632)_8$, $(21.3)_{16}$ y $(1011.1011)_2$.

27. Se ha sugerido que el siguiente mensaje se transmita al espacio exterior como una señal de que en el planeta hay vida inteligente. La idea es que cualquier forma de vida inteligente en cualquier parte comprenda con seguridad su contenido intelectual y con ello traduzca nuestra presencia inteligente aquí en la Tierra. ¿Cuál es el significado del siguiente mensaje?:

    11.00100100001111110110

Sugerencia. transforme el número binario en número decimal y deduzca el mensaje.

28. Escriba un programa en Matlab para convertir números decimales en binarios y viceversa. Pruebe su programa con varios ejemplos, en particular, el mensaje del problema anterior.

29. Muestre que si u es correctamente redondeado a s dígitos significativos, entonces

$$\frac{|\Delta u|}{|u|} \leq \frac{1}{2}\beta^{-s+1}$$

donde β es la base del sistema numérico.

30. Demuestre que la cardinalidad del sistema flotante $Fl(\beta, t, L, U)$ es

$$2(\beta-1)\beta^{t}(U-L+1)+1.$$

31. Verifique que el mayor número positivo que se puede representar en el sistema IEEE Estándar $Fl(2, 23, -126, 127)$ es $2^{127}(2-2^{-23})$ y que es aproximadamente igual a 3.4028235×10^{38}.

32. Calcule el número positivo más pequeño y más grande que se pueden representar en el sistema IEEE de precisión doble. En cada caso, calcule su equivalente decimal. Dibuje en la recta numérica los números punto flotante del formato IEEE de precisión doble, es decir, intervalos de overflow negativo, números negativos, underflow negativo, underflow positivo, números positivos y overflow positivo.

33. Haga lo mismo como en el ejercicio anterior para el sistema IEEE de precisión extendida.

34. Calcule la unidad de redondeo y el épsilon del sistema flotante IEEE de precisión doble y de precisión extendida. Deduzca en cada caso, el número de decimales que redondea correctamente en la fracción de un número dado.

35. Si es dueño o dueña de una calculadora y/o de una computadora, es bueno que lo conozca mejor, para ello investigue la base β del sistema numérico con que opera, su precisión t, los límites L y U de los exponentes. Verifique cómo redondea e investigue su unidad de redondeo y su épsilon correspondiente. Dibuje en la recta numérica los números punto flotante de su máquina, similar como lo realizado en los problemas anteriores.

36. Divida 128 entre 5040 paso a paso en un sistema flotante de precisión 2 y redondeo. Calcule el tamaño de los errores absoluto y relativo que se cometen.

37. Resuelva la ecuación cuadrática $x^2 + 1000.01x - 2.5245315 = 0$ con la fórmula general en una aritmética flotante con precisión 8 y redondeo. Note que una de sus raíces pierde precisión y resulta $2.52500000 \times 10^{-3}$. Explique porqué sucede esto. Calcule de nuevo dicha raíz eliminando la resta en la fórmula general, y en el mismo sistema flotante para mejorar su precisión. Debe resultar $2.52449988 \times 10^{-3}$.

38. Resuelva la ecuación $x^2 + 111.11x + 1.2121 = 0$ como en el problema anterior en una aritmética flotante con precisión 8 y redondeo.

39. Es posible evaluar la función e^x para cualquier valor de x, mediante la serie de Taylor

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \cdots$$

pero cuando $x<0$, la suma anterior se convierte en una serie alternante de términos positivos y negativos, lo que propicia en la computadora el efecto de la cancelación numérica.

a) ¿De qué manera se podría evitar la cancelación sin usar mayor precisión en los cálculos?

En una aritmética flotante de precisión 2 y redondeo, calcule e^{-2} con los siguientes procedimientos:

b) Use los primeros 6 términos de la serie de Taylor respetando el orden de los sumandos. ¿Cuál es el error de aproximación?.

c) Use los primeros 6 términos de la serie de Taylor pero sume primero todos los términos positivos, luego todos los negativos y finalmente reste. ¿Cuál es el error de aproximación para este caso?.

d) Evite la cancelación aplicando la respuesta que dio en (a). Sugerencia: use también los primeros 6 términos de una serie y ordene de menor a mayor. Calcule el error de aproximación. Compare las respuestas de todos los incisos.

40. La serie $\sum_{n=1}^{\infty} \frac{1}{n^2}$ converge a $\frac{\pi^2}{6}$. Realice el siguiente experimento numérico en la aproximación del valor de la serie:

a) Aproxime el valor de la serie en orden decreciente en una aritmética flotante con precisión 1 y redondeo. ¿Cuántos términos contribuyen en la suma? ¿Cuál es el error de aproximación?

b) Aproxime ahora el valor de la serie en orden creciente en la misma aritmética flotante con precisión 1 y redondeo, con los términos de la serie que contribuyen en el inciso (a). Repita los cálculos con los primeros 5 términos de la serie, luego con los primeros 10 términos. ¿Cuál es el error de aproximación en cada caso?

c) Calcule de nuevo el valor de la serie en orden decreciente en una aritmética flotante con precisión 2 y redondeo. ¿Cuántos términos contribuyen ahora en la suma? ¿Cuál es el érror de aproximación?

d) Ahora el valor de la serie en orden creciente en la misma aritmética flotante con precisión 2 y redondeo, con los términos de la serie que contribuyen en el inciso (a). Repita los cálculos con los primeros 15 términos de la serie, luego con los primeros 20 términos. ¿Cuál es el error de aproximación en cada caso?

41. La serie $\sum_{n=1}^{\infty} \frac{1}{2^n}$ converge a 1. Aproxime el valor de la serie como en el ejercicio anterior, pero en los casos del orden creciente use solamente los términos que contribuyen en los casos decrecientes.

42. Suponemos que $n\mu < 0.1$ y $|\varepsilon_i| \le \mu$, $i = 1, 2, ..., n$. Muestre que

$$|(1+\varepsilon_1)(1+\varepsilon_2)...(1+\varepsilon_n)| \le 1 + 1.06n\mu .$$

Sugerencia: Use $(1+x)^n \le e^{nx}$ y realice una expansión en series.

43. Sea $S_n = \sum_{i=1}^{n} x_i y_i$. Muestre que $|\overline{S}_n - S_n| \le \sum_{i=1}^{n} |(n-i+2)x_i y_i| 1.06\mu$. (Como en el Teorema 1.3).

CAPÍTULO II

INTERPOLACIÓN

2.1 PLANTEAMIENTO DEL PROBLEMA

Problema: Dado un conjunto de $n+1$ puntos $(x_1, f_1), (x_2, f_2), \ldots, (x_{n+1}, f_{n+1})$ que llamaremos datos, con la condición de que $x_i \neq x_j$ para $i \neq j$, encontrar una función P tal que $P(x_i) = f_i, \quad i = 1, 2, \ldots, n+1$.

P se llama función de interpolación de los puntos $(x_1, f_1), (x_2, f_2), \ldots, (x_{n+1}, f_{n+1})$. Hay varios tipos de funciones de interpolación: polinomios, funciones racionales, funciones trigonométricas, splines, etc. En este curso se estudiarán las interpolaciones por polinomios y splines cúbicos. Cuando los datos corresponden a una función complicada o desconocida f, es decir, $f_i = f(x_i), \quad 1 \leq i \leq n+1$, a P se le llama función de interpolación de f en dichos puntos.

Algunos usos importantes de la función de interpolación son

1. Se usa para estimar el valor de f en puntos x de interés entre $x_1, x_2, \ldots, x_{n+1}$.
2. Se usa para aproximar las derivadas de f: f', f'', etc., en puntos x de interés.
3. Se usa para aproximar la integral de f en intervalos $[a, b]$ entre $x_1, x_2, \ldots, x_{n+1}$.

2.2 INTERPOLACIÓN POR POLINOMIOS

Teorema 2.1: Sean $x_1, x_2, \ldots, x_{n+1}$ puntos distintos. Para valores arbitrarios $f_1, f_2, \ldots, f_{n+1}$, existe un único polinomio P de grado $\leq n$ tal que $P(x_i) = f_i$, para toda $i = 1, 2, \ldots, n+1$.

Demostración: La prueba de la existencia del polinomio de interpolación lo daremos de dos maneras diferentes, una conocida como la *fórmula de Lagrange* que lo haremos en la Sección 2.2.1, y la otra usando el método de inducción matemática sobre n, que lo presentaremos en la Sección 2.2.3. Este último da origen a la llamada *fórmula de Newton* para el polinomio de interpolación.

Prueba de unicidad del polinomio P: Suponemos que hay dos polinomios distintos P y Q que resuelven el problema, es decir, $P(x_i) = f_i = Q(x_i)$, para toda

$i = 1, 2, \ldots, n+1$. El polinomio $R(x) = P(x) - Q(x)$ tiene grado $\leq n$ y satisface que $R(x_i) = 0$, $i = 1, 2, \ldots, n+1$. Por el Teorema Fundamental del Álgebra, $R(x) = 0$ para toda x. Así, los polinomios P y Q son iguales. ■

El error de aproximación se estima en el siguiente teorema:

Teorema 2.2: Si f es una función con derivadas continuas hasta de orden $n + 1$ en algún intervalo que contiene los puntos $x_1, x_2, \ldots, x_{n+1}$, y si P es el polinomio de grado $\leq n$ que interpola a f en dichos puntos, entonces el error de aproximación se puede estimar como

$$f(x) - P(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}(x - x_1)(x - x_2)\cdots(x - x_{n+1}), \qquad \ldots\ldots\ldots \qquad (2.1)$$

para toda x, y para algún punto ξ_x en el intervalo formado por los puntos $x_1, x_2, \ldots, x_{n+1}$.

Demostración: Sea $G(x) = (x - x_1)(x - x_2)\cdots(x - x_{n+1})$ y $E_T(x) = f(x) - P(x)$.
Definimos $H(t) = E_T(x^*)G(t) - E_T(t)G(x^*)$, para algún x^* fijo.
$H(x_i) = E_T(x^*)G(x_i) - E_T(x_i)G(x^*) = 0$, $i = 1, 2, \cdots, n+1$, y
$H(x^*) = E_T(x^*)G(x^*) - E_T(x^*)G(x^*) = 0$.
Por lo tanto, H es una función que tiene $n + 2$ ceros que son $x_1, x_2, \ldots, x_{n+1}, x^*$. Como H también tiene derivadas continuas hasta de orden $n + 1$, entonces, H' tiene $n + 1$ ceros, H'' tiene n ceros, así sucesivamente, $H^{(n+1)}$ tiene un cero que lo denotamos por ξ_{x^*}. Pero,

$$H^{(n+1)}(t) = E_T(x^*)G^{(n+1)}(t) - E_T^{(n+1)}(t)G(x^*) = E_T(x^*)(n+1)! - f^{(n+1)}(t)G(x^*),$$

sustituyendo $t = \xi_{x^*}$, resulta

$$E_T(x^*)(n+1)! - f^{(n+1)}(\xi_{x^*})G(x^*) = H^{(n+1)}(\xi_{x^*}) = 0,$$

por lo tanto,

$$E_T(x^*) = \frac{f^{(n+1)}(\xi_{x^*})}{(n+1)!}(x^* - x_1)(x^* - x_2)\cdots(x^* - x_{n+1}). \blacksquare$$

Observaciones y Comentarios

a) $E_T(x) = f(x) - P(x)$ dado en (2.1), es la función error de truncamiento o de aproximación, y mide el error que se comete al considerar el polinomio P una aproximación de la función f en cada punto x.

b) El error de aproximación en cada nodo $x_1, x_2, \ldots, x_{n+1}$ es cero.

c) El factor $(x^*-x_1)(x^*-x_2)\cdots(x^*-x_{n+1})$ depende de la elección de los nodos $x_1, x_2, \ldots, x_{n+1}$, y es minimizado en valor absoluto cuando x^* está próximo a todos los x_i's; de hecho cuando x^* es el promedio de los x_i's.

d) La observación (c) es importante, porque al aproximar el valor de $f(x^*)$ no es necesario usar todos los x_i's, sino únicamente "aquellos puntos" que están cerca de x^*, y es posible que el error de aproximación sea más pequeño.

e) La elección de n (cantidad de nodos de interpolación) es algunas veces más complicada, porque no sólo influye en el error de truncamiento sino también en el error de redondeo. El análisis del error de redondeo de los algoritmos que calculan el polinomio de interpolación no se hace en este libro, pero se puede consultar por ejemplo, en Vandergraft (1983) [11]. Vandergraft recomienda tomar n pequeño, digamos 5 o 6 a lo más. En caso de disponer de muchos datos, recomienda agruparlos, pero cuidando que no estén tan cercanos, digamos $|x_i - x_j| \geq 0.1$ para $i \neq j$.

f) Las observaciones (c) y (d) muestran que el error no disminuye necesariamente al aumentar los nodos de interpolación.

g) Si la derivada de orden $n+1$ de f no es acotada, el error de aproximación puede crecer indefinidamente. El ejemplo clásico que muestra este hecho es la función de Runge

$$f(x) = \frac{1}{1+25x^2}, \quad -1 \leq x \leq 1.$$

Lo analizaremos con detalle en el Ejemplo 2.5.

h) Si f es desconocida, estimar $f^{(n+1)}(x^*)$ es otro problema más complicado.

2.2.1 Fórmula de Lagrange

Presentaremos aquí la primera prueba de existencia del polinomio de interpolación. Como en el Teorema 2.1, sean $x_1, x_2, \ldots, x_{n+1}$ puntos distintos y $f_1, f_2, \ldots, f_{n+1}$ valores arbitrarios. El objetivo es construir $n+1$ polinomios de grado n: $p_1, p_2, \ldots, p_{n+1}$ de tal manera que p_1 tome el valor 1 en x_1 y 0 en los demás puntos; p_2 tome el valor 1 en x_2 y 0 en los demás puntos; y así sucesivamente hasta llegar a p_{n+1} que tome el valor 1 en x_{n+1} y 0 en los demás puntos. Con esto es claro que el polinomio:

$$\boxed{P(x) = f_1 p_1(x) + f_2 p_2(x) + \cdots + f_{n+1} p_{n+1}(x).} \quad \ldots\ldots\ldots (2.2)$$

será el polinomio que interpola los puntos (x_1, f_1), (x_2, f_2), ..., (x_{n+1}, f_{n+1}); pues es de grado $\leq n$, y $P(x_i) = f_i$ para $1 \leq i \leq n+1$. ■

La forma que tiene el polinomio $P(x)$ es conocido como la *fórmula de Lagrange*, y los polinomios $p_i(x)$ se les llama *polinomios de Lagrange*. Los polinomios p_i son fáciles de construir, pues por ejemplo:

$$p_1(x) = \frac{(x-x_2)(x-x_3)\ldots(x-x_{n+1})}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_{n+1})}.$$

En general,

$$p_i(x) = \frac{(x-x_1)(x-x_2)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_{n+1})}{(x_i-x_1)(x_i-x_2)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_{n+1})}, \quad 1 \leq i \leq n+1. \qquad (2.3)$$

Es importante notar que el polinomio dado por (2.2), no está expresado en la forma "estándar" $a_n x^n + a_{n-1}x^{n-1} + \ldots + a_0$. Se puede mostrar, ver Vandergraft (1983) [11], que la cantidad de operaciones necesarias que se requiere para evaluar un polinomio dado en forma estándar en un punto x^* es $nA + nM$ (*n adiciones + n multiplicaciones*). En la Sección 2.2.2, veremos que desafortunadamente la fórmula (2.2), requiere considerablemente más de estas operaciones. En la Sección 2.2.3, derivaremos la fórmula de Newton que disminuirá considerablemente el número de operaciones, que se requiere para evaluar el polinomio de interpolación en puntos de interés. La razón principal de introducir primero la fórmula de Lagrange es sobre todo por su simplicidad.

Ejemplo 2.1: Use la fórmula de Lagrange para calcular el polinomio que interpola el siguiente conjunto de datos

x	−1	0	1	2
f	−3	−2	−1	6

En efecto:
$$p_1(x) = \frac{x(x-1)(x-2)}{(-1)(-2)(-3)} = -\frac{1}{6}(x^3 - 3x^2 + 2x),$$

$$p_2(x) = \frac{(x+1)(x-1)(x-2)}{(1)(-1)(-2)} = \frac{1}{2}(x^3 - 2x^2 - x + 2),$$

$$p_3(x) = \frac{(x+1)(x)(x-2)}{(2)(1)(-1)} = -\frac{1}{2}(x^3 - x^2 - 2x),$$

$$p_4(x) = \frac{(x+1)(x)(x-1)}{(3)(2)(1)} = \frac{1}{6}(x^3 - x).$$

Sustituimos en la fórmula de Lagrange (2.2) y simplificamos para obtener

$$P(x) = -3p_1(x) - 2p_2(x) - p_3(x) + 6p_4(x) = x^3 - 2.$$

Por lo tanto, $P(x) = x^3 - 2$ es el polinomio de grado 3 que interpola los datos. ■

Una visualización gráfico de los datos y del polinomio de interpolación lo podemos realizar en Matlab, con las siguientes instrucciones:

```
xd=[-1,0,1,2];
fd=[-3,-2,-1,6];
% Evaluación del polimonio de interpolación.
x=[xd(1)-0.5:0.1:xd(end)+0.2];
p=x.^3-2;
% Graficación de los datos y el polimonio de interpolación.
plot(xd,fd,'o',x,p); grid
```

La gráfica que obtenemos se muestra en la Figura 2.1.

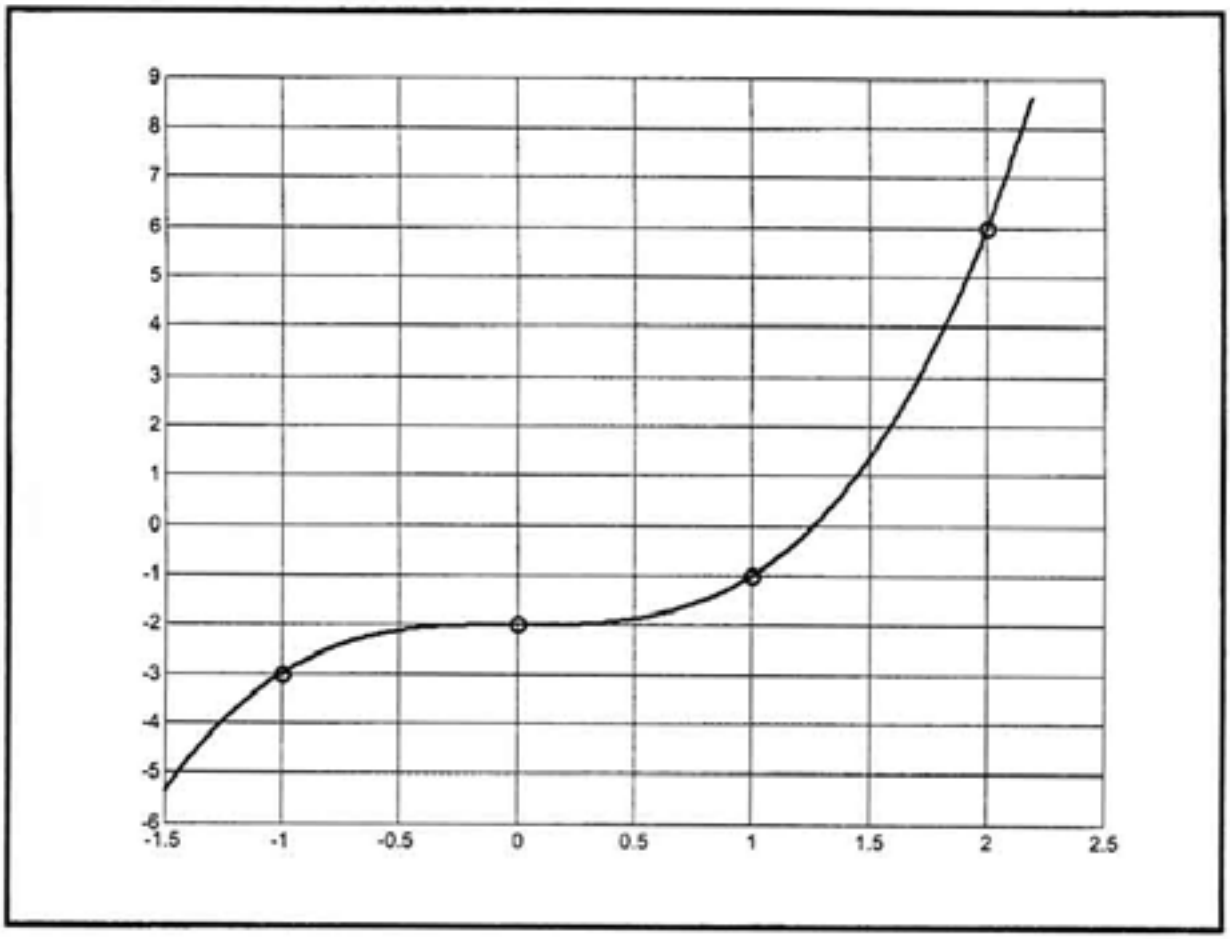


Figura 2.1. Gráfica del polinomio que interpola los datos del Ejemplo 2.1.

A continuación presentamos un algoritmo para evaluar el polinomio de interpolación en puntos de interés, usando la fórmula de Lagrange. El algoritmo es bastante útil para visualizar gráficamente el polinomio de interpolación, por lo que se requiere programarlo en algún lenguaje de programación. No se recomienda usar el algoritmo para realizar cálculos manuales, es mejor recordar la fórmula de Lagrange tal como es y usarlo cuando se requiera.

Algoritmo 1 para evaluar el polinomio $P(x)$ usando la fórmula de Lagrange en un punto de interés x^*.

Entrada: $x_1, x_2, \ldots, x_{n+1}, f_1, f_2, \ldots, f_{n+1}$ y x^*.
Salida: x^* y $p^* = P(x^*)$

1. $p^* = 0$
2. Para $i = 1, 2, \ldots, n+1$
   - 2.1. $p = 1$
   - 2.2. Para $j = 1, 2, \ldots, n+1$
     - 2.2.1. Para $j \neq i$, $p = p(x^* - x_j) / (x_i - x_j)$
   - 2.3. $p^* = p^* + f_i p$
3. Alto.

Observamos que en el paso 2, para $i = 1$, se comienza con $p = 1$. En el paso 2.2, al iterar sobre j desde 1, 2 hasta $n + 1$, obtenemos:

$$p = \frac{x^* - x_2}{x_1 - x_2} \frac{x^* - x_3}{x_1 - x_3} \cdots \frac{x^* - x_{n+1}}{x_1 - x_{n+1}},$$

que es el primer polinomio de Lagrange evaluado en x^*, es decir, es el valor de $p_1(x^*)$. En el paso 2.3, p^* toma el valor de $f_1 p_1(x^*)$, que es el primer sumando de (2.2). Al aumentar el valor de i a 2, se comienza de nuevo con $p = 1$, y se itera otra vez sobre j desde 1, 2 hasta $n + 1$. El valor de p se actualiza como $p_2(x^*)$ de acuerdo con (2.3), y p^* toma el valor de $f_1 p_1(x^*) + f_2 p_2(x^*)$ que son los primeros dos sumandos de (2.2). Al agotar el valor de i por $n + 1$, y agotar también los valores de j, p^* toma el valor de $P(x^*)$ de acuerdo con la fórmula (2.2).

Ejemplo 2.2: Dado el siguiente conjunto de datos:

x	−3.0	−1.0	1.0	2.0	2.5	3.0
f	1.0	1.5	2.0	2.0	1.5	1.0

Estimar el valor de f en el punto $x^* = 0.3$ usando la fórmula de Lagrange, en una aritmética flotante de precisión 7 y redondeo.

Solución: De acuerdo con la idea de la observación (d) del Teorema 2.2, es suficiente con elegir los puntos más cercanos a x^* para calcular el valor aproximado de $f(x^*)$. Sin embargo, no hay un criterio definido para saber cuántos puntos se deben elegir exactamente. Se recomienda aproximar el valor de $f(x^*)$ usando polinomios de grado k, para $k \leq n$ (donde $n+1$ es el número total de datos), y elegir la aproximación que mejor responda a la tendencia de los datos del problema. Con esta idea vamos a aproximar tres posibles valores para $f(x^*)$.

a) Primero usaremos un polinomio de interpolación de primer grado. Basta elegir dos puntos cuyas abscisas son los dos vecinos más cercanos de x^* que son $(-1.0, 1.5)$ y $(1.0, 2.0)$. Como ilustración, aplicaremos únicamente para este caso el algoritmo 1 manualmente.

Entrada: $x_1 = -1.0$, $x_2 = 1.0$, $f_1 = 1.5$, $f_2 = 2.0$ y $x^* = 0.3$
Salida: $x^* = 0.3$ y $p^* = P(0.3)$

1. $p^* = 0$
2. $i = 1, 2$.
   $i = 1$, $p = 1$, $j = 1, 2$.
   $j = 2$, $p = 1 * (0.3 - 1.0) / (-1.0 - 1.0) = 0.35$
   $p^* = p^* + f_1 p = 0 + 1.5*0.35 = 0.525$
   $i = 2$, $p = 1$, $j = 1, 2$.
   $j = 1$, $p = 1 * (0.3 + 1.0) / (1.0 + 1.0) = 0.65$
   $p^* = p^* + f_2 p = 0.525 + 2.0*0.65 = 1.825$
3. Alto.

El resultado para este caso es $f(0.3) = 1.825$, y su visualización gráfico se muestra en la Figura 2.2.

b) Para usar un polinomio de interpolación de segundo grado, se eligen tres puntos cuyas abscisas son los más cercanos a x^*. Ellos son $(-1.0, 1.5)$, $(1.0, 2.0)$ y $(2.0, 2.0)$. En este caso se obtiene como resultado $f(0.3) = 1.9008333$. La visualización gráfico se presenta en la Figura 2.3, con la ayuda de un programa hecho en Matlab y basado en el Algoritmo 1.

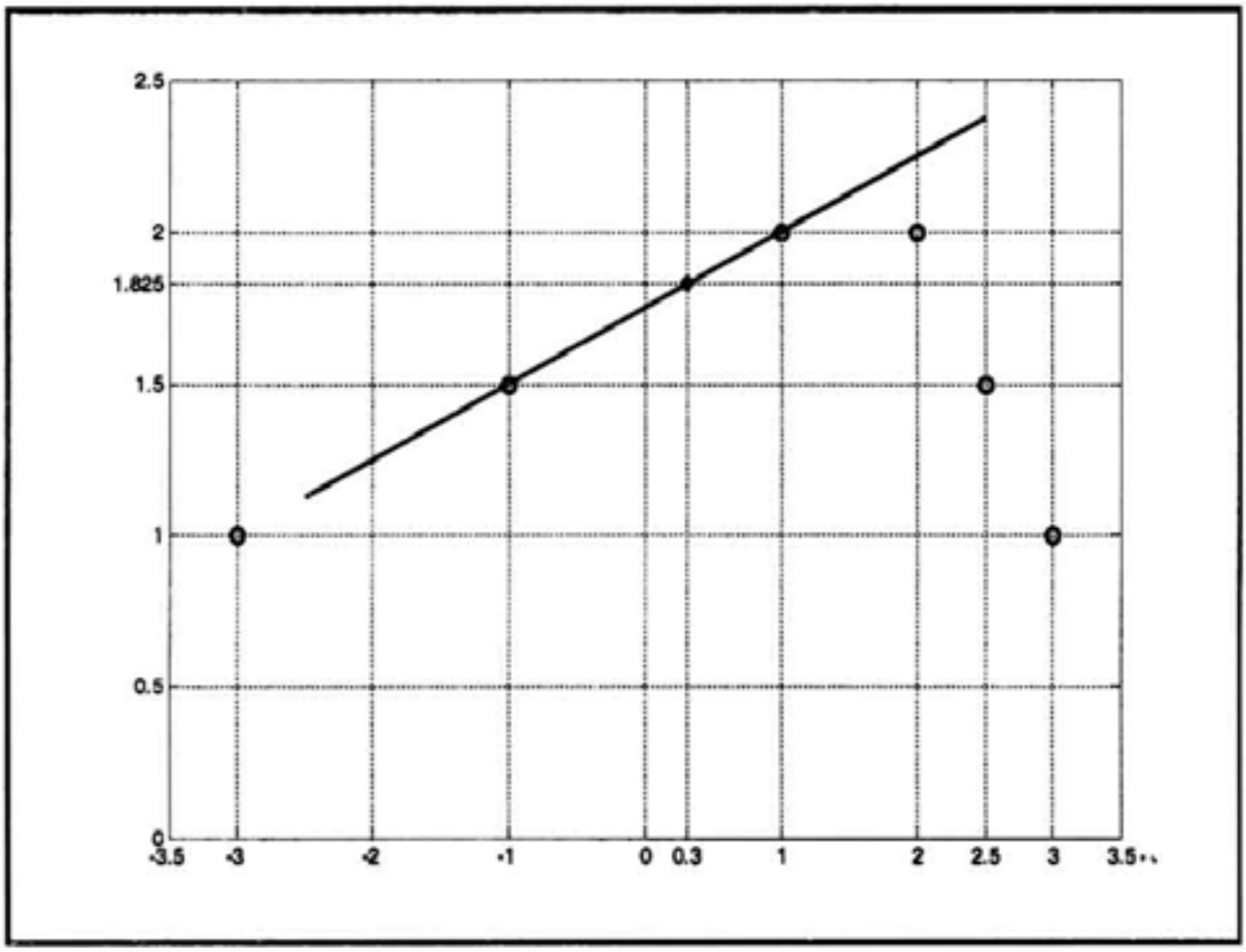


Figura 2.2. Interpolación lineal para estimar $f(0.3)$.

c) Para usar un polinomio de interpolación de tercer grado, se eligen cuatro puntos cuyas abscisas son los más cercanos a x^*. Ellos son (−1.0, 1.5), (1.0, 2.0), (2.0, 2.0) y (2.5, 1.5). En este caso, usaremos directamente los polinomios de Lagrange dados en (2.3), para resolver el problema. Los datos son: $x_1 = -1.0$, $x_2 = 1.0$, $x_3 = 2.0$, $x_4 = 2.5$, $f_1 = 1.5$, $f_2 = 2.0$, $f_3 = 2.0$, $f_4 = 1.5$ y $x^* = 0.3$.

$$p_1(0.3) = \frac{(0.3-1.0)(0.3-2.0)(0.3-2.5)}{(-1.0-1.0)(-1.0-2.0)(-1.0-2.5)} = 0.12466667,$$

$$p_2(0.3) = \frac{(0.3+1.0)(0.3-2.0)(0.3-2.5)}{(1.0+1.0)(1.0-2.0)(1.0-2.5)} = 1.6206667,$$

$$p_3(0.3) = \frac{(0.3+1.0)(0.3-1.0)(0.3-2.5)}{(2.0+1.0)(2.0-1.0)(2.0-2.5)} = -1.3346667,$$

$$p_4(0.3) = \frac{(0.3+1.0)(0.3-1.0)(0.3-2)}{(2.5+1.0)(2.5-1.0)(2.5-2)} = 0.58933333,$$

$$P(0.3) = 1.5(0.12466667) + 2.0(1.6206667) + 2.0(-1.3346667) + 1.5(0.58933333) = 1.6430000.$$

Por lo tanto, el resultado para este caso es $f(0.3) = 1.643$. La visualización gráfico se presenta en la Figura 2.4.

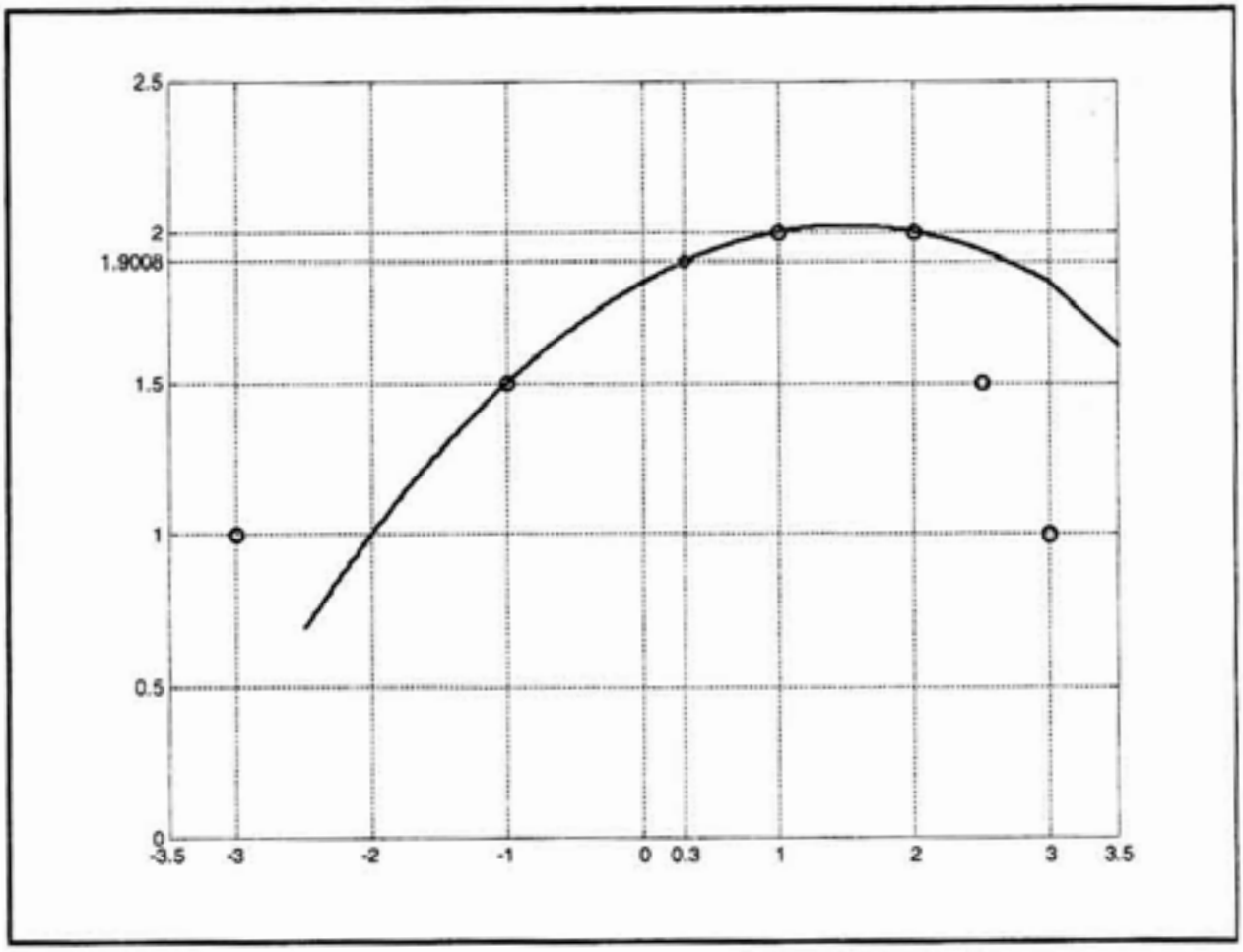


Figura 2.3. Interpolación cuadrática para estimar $f(0.3)$.

Si los datos provienen de alguna función f, ¿Cuál de los tres resultados obtenidos p^*=1.825, p^*=1.9008333 ó $P(0.3) = 1.643$ será más exacto como aproximación de $f(0.3)$?. Si no se dispone de más información sobre tal función f, es imposible determinar cuál de estos tres valores es más exacto como aproximación de $f(0.3)$. ■

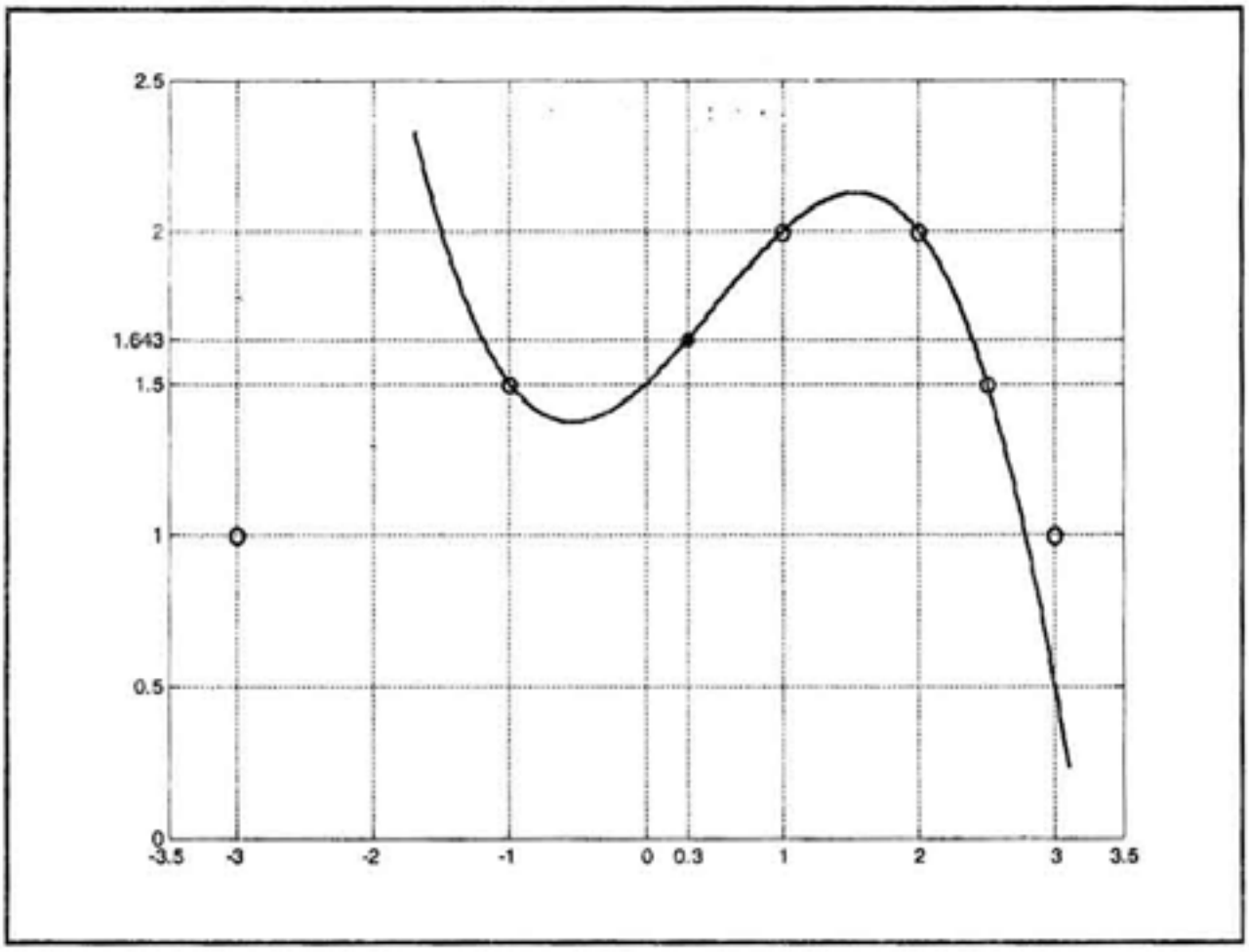


Figura 2.4. Interpolación cúbica para estimar $f(0.3)$.

Todavía es posible calcular dos aproximaciones más para f en x^*, usando un polinomio de interpolación de grado 4 con los últimos cinco puntos de la tabla, y por último usando el polinomio de interpolación de grado 5 con todos los puntos de la tabla. Para calcular estas dos aproximaciones manualmente se recomienda proceder como en el inciso (c). Trabajaremos de nuevo este caso en el Ejemplo 2.4 de la Sección 2.2.3.

2.2.2 Eficiencia del Algoritmo

La eficiencia de un algoritmo se define como la cantidad total de operaciones aritméticas de punto flotante que tiene que realizar para obtener un resultado.

En el caso del Algoritmo 1, las operaciones aritméticas de punto flotante se realizan en el paso 2 para obtener p^*.

En el paso 2.2 requiere: n multiplicaciones, n divisiones y $2n$ substracciones.
En el paso 2.3 requiere: 1 multiplicación y 1 adición.
Como el paso 2 se ejecuta $n + 1$ veces, resultan

$$(n+1)(2n+1)A+(n+1)(n+1)M+(n+1)nD\,.$$

donde A denota una adición o una substracción (una suma y una resta tardan en realizarse aproximadamente el mismo tiempo), M denota una multiplicación, y D una división. Simplificando la suma anterior, obtenemos la medida de eficiencia para el Algoritmo 1:

$$(2n^2+3n+1)A+(n^2+2n+1)M+(n^2+n)D\,.$$

Para $n = 3$, resultan $28\,A + 16\,M + 12\,D$.

La eficiencia computacional de un algoritmo, es el tiempo total que tarda una computadora en realizar todas las operaciones aritméticas de punto flotante, para obtener un resultado. Como ejemplo, en la Tabla 2.1 citamos el tiempo promedio en micro segundos (μseg) $\left(1\,\mu seg = 10^{-6}\,seg\right)$, que tardaban algunas computadoras antiguas en realizar operaciones aritméticas de punto flotante.

	A	M	D
Univac 1100/40	0.76	1.65	5.3
Univac 1108	2.2	3.0	8.63
IBM 360/50	6.13	20.75	21.25
IBM 360/85	0.38	1.36	1.64
CDC 6600	0.4	1.0	2.9

Tabla 2.1. Tiempo promedio en operaciones aritméticas.

Por ejemplo, una IBM 360/85 se tardaría 28(0.38)+16(1.36)+12(1.64) = 52.08 μseg o bien aproximadamente 5 cien milésimos de segundos, para hacer el Ejemplo 2.2(c) con el Algoritmo 1.

2.2.3 Fórmula de Newton

Presentaremos ahora una segunda prueba de la existencia del polinomio de interpolación del Teorema 2.1. La prueba se hará usando el método de inducción

matemática sobre n. Para $n = 1$, se tienen solamente dos puntos (x_1, f_1) y (x_2, f_2), y la recta que pasa entre ellos es de grado 0 ó 1, y está dado por:

$$P_1(x) = f_1 + \frac{f_2 - f_1}{x_2 - x_1}(x - x_1)$$

Por lo tanto es el polinomio de grado ≤ 1 que los interpola.

Para fijar ideas en el paso de inducción de $k-1$ a k, suponemos que tenemos un tercer punto (x_3, f_3). Si este punto fuera colineal con los dos anteriores, entonces la misma recta $P_1(x) = f_1 + \frac{f_2 - f_1}{x_2 - x_1}(x - x_1)$ interpolaría a los tres puntos. Si (x_3, f_3) no es colineal con los dos anteriores, entonces por los tres puntos debe pasar una única parábola, es decir existe un polinomio de grado 2 que los interpola. Como esta parábola pasa por los puntos (x_1, f_1) y (x_2, f_2), entonces para conocerlo basta con agregar un término cuadrático de la forma $c(x - x_1)(x - x_2)$ al polinomio $P_1(x)$. Denotaremos por $P_2(x)$ a la parábola, así que

$$P_2(x) = P_1(x) + c(x - x_1)(x - x_2) = f_1 + \frac{f_2 - f_1}{x_2 - x_1}(x - x_1) + c(x - x_1)(x - x_2),$$

donde c es una constante que se puede calcular, pues $P_2(x_3) = f_3$. Sustituyendo resulta que

$$c = \frac{f_3 - f_1 - \frac{x_3 - x_1}{x_2 - x_1}(f_2 - f_1)}{(x_3 - x_1)(x_3 - x_2)}$$

Con estas ideas avanzamos con la prueba: Suponemos que para $n = k - 1$, $k > 2$, existe un polinomio $P_{k-1}(x)$ de grado $\leq k - 1$ tal que $P_{k-1}(x_i) = f_i$, $\forall\, i = 1, 2, \ldots, k$. Mostraremos que para $n = k$, existe un polinomio $P_k(x)$ de grado $\leq k$ tal que $P_k(x_i) = f_i$, $\forall\, i = 1, 2, \ldots, k+1$.

En efecto: Con las ideas analizadas anteriormente, definimos el polinomio $P_k(x) = P_{k-1}(x) + c(x - x_1)(x - x_2)\cdots(x - x_k)$ para algún parámetro c por determinar. Este polinomio $P_k(x)$ tiene grado $\leq k$, y por hipótesis de inducción, satisface $P_k(x_i) = f_i$, $i = 1, 2, \ldots, k$. Veamos que para algún valor adecuado del parámetro c, también satisface $P_k(x_{k+1}) = f_{k+1}$. Para ello observamos que

$$P_k(x_{k+1}) = P_{k-1}(x_{k+1}) + c(x_{k+1} - x_1)(x_{k+1} - x_2)\cdots(x_{k+1} - x_k),$$

Por lo tanto, $P_k(x_{k+1}) = f_{k+1} \Leftrightarrow c = \dfrac{f_{k+1} - P_{k-1}(x_{k+1})}{(x_{k+1} - x_1)(x_{k+1} - x_2)\cdots(x_{k+1} - x_k)}$.

El denominador no se anula por ser puntos distintos, y es el valor del parámetro c que determina el polinomio $P_k(x)$ que se requiere. Con esto termina esta prueba de existencia. ■

La construcción del polinomio $P(x)$ que interpola el conjunto de datos $(x_1, f_1), (x_2, f_2), \ldots, (x_{n+1}, f_{n+1})$, dado en la prueba anterior es conocida como la *fórmula de Newton*. De dicha prueba observamos que $P(x)$ es igual al polinomio $P_n(x)$ que se obtiene inductivamente por

$$i)\ P_0(x) = c_0$$

$$ii)\ P_k(x) = P_{k-1}(x) + c_k(x - x_1)(x - x_2)\cdots(x - x_k),\ 1 \le k \le n. \quad \ldots. \quad (2.4)$$

donde $P_{k-1}(x)$ es el polinomio de grado $\le k-1$ que interpola los puntos $(x_1, f_1), (x_2, f_2), \ldots, (x_k, f_k)$, y las c_k's son constantes que se pueden calcular como antes.

Así que la fórmula de Newton para el polinomio $P(x)$ que interpola los puntos $(x_1, f_1), (x_2, f_2), \ldots, (x_{n+1}, f_{n+1})$ está dado por

$$\boxed{\begin{aligned} P(x) = P_n(x) = c_0 + c_1(x - x_1) + c_2(x - x_1)(x - x_2) + \cdots + \\ c_n(x - x_1)(x - x_2)\cdots(x - x_n) \end{aligned}} \qquad (2.5)$$

De hecho el polinomio dado por (2.5) se le llama *polinomio de Newton*.
Notemos que c_k es el coeficiente del término x^k en el polinomio $P_k(x)$, que

$$c_0 = f_1\ ,\ c_1 = \frac{f_2 - f_1}{x_2 - x_1},\quad c_2 = \frac{f_3 - f_1 - \dfrac{x_3 - x_1}{x_2 - x_1}(f_2 - f_1)}{(x_3 - x_1)(x_3 - x_2)},$$

y que en general, el parámetro c_k depende de los puntos $(x_1, f_1), (x_2, f_2), \ldots, (x_{n+1}, f_{n+1})$. Por lo tanto, si f_i proviene de alguna función f, es decir, $f_i = f(x_i)$ $\forall\ i = 1, 2, \ldots, n+1$, entonces, es natural introducir la notación

$$c_k = f[x_1, x_2, \ldots, x_{k+1}].$$

Obtendremos una fórmula de recurrencia para calcular c_k, mediante la siguiente caracterización del polinomio $P_k(x)$.

Teorema 2.3: $P_k(x) = P_{k-1}^{(1)}(x) + \dfrac{x - x_1}{x_{k+1} - x_1}\left(P_{k-1}^{(2)}(x) - P_{k-1}^{(1)}(x)\right)$. ..... (2.6)

donde $P_{k-1}^{(1)}(x)$ y $P_{k-1}^{(2)}(x)$ son polinomios de grado $\leq k-1$. El polinomio $P_{k-1}^{(1)}(x)$ interpola los puntos (x_1, f_1), (x_2, f_2), ... (x_k, f_k), y el polinomio $P_{k-1}^{(2)}(x)$ interpola los puntos (x_2, f_2), (x_3, f_3), ... (x_{k+1}, f_{k+1}).

Demostración: Es claro que el polinomio $P_k(x)$ es de grado $\leq k$, y

$$P_k(x_i) = P_{k-1}^{(1)}(x_i) + \frac{x_i - x_1}{x_{k+1} - x_1}\left(P_{k-1}^{(2)}(x_i) - P_{k-1}^{(1)}(x_i)\right) = f_i + \frac{x_i - x_1}{x_{k+1} - x_1}(f_i - f_i) = f_i,$$

$2 \leq i \leq k$.

$$P_k(x_1) = P_{k-1}^{(1)}(x_1) + \frac{x_1 - x_1}{x_{k+1} - x_1}\left(P_{k-1}^{(2)}(x_1) - f_1\right) = f_1,$$

$$P_k(x_{k+1}) = P_{k-1}^{(1)}(x_{k+1}) + \frac{x_{k+1} - x_1}{x_{k+1} - x_1}\left(f_{k+1} - P_{k-1}^{(1)}(x_{k+1})\right) = f_{k+1}.$$

Por lo tanto, $P_k(x)$ es el mismo polinomio de grado $\leq k$ que interpola a f en los puntos (x_1, f_1), (x_2, f_2), ... (x_{k+1}, f_{k+1}). ■

El coeficiente de x^{k-1} en el polinomio $P_{k-1}^{(1)}(x)$ es $c_{k-1} = f[x_1, x_2, \ldots, x_k]$, y el coeficiente de x^{k-1} en el polinomio $P_{k-1}^{(2)}(x)$ es $c_{k-1}^{*} = f[x_2, x_3, \ldots, x_{k+1}]$. Por lo tanto, el coeficiente de x^k en el polinomio $P_k(x)$ es $\dfrac{1}{x_{k+1} - x_1}\left(c_{k-1}^{*} - c_{k-1}\right)$.

Comparando los coeficientes de x^k en el polinomio $P_k(x)$ representados por (2.4) y (2.6) resulta:

$$c_k = f[x_1, x_2, \ldots, x_{k+1}] = \frac{1}{x_{k+1} - x_1}\left(f[x_2, x_3, \ldots, x_{k+1}] - f[x_1, x_2, \ldots, x_k]\right). \quad \ldots (2.7)$$

La fórmula (2.7) es una fórmula de recurrencia, conocida como *diferencias divididas de Newton*, y hace posible calcular las constantes $c_0, c_1, \ldots, c_n$.

En efecto, tomando $f[x_i] = f(x_i) = f_i$, $1 \leq i \leq n+1$; y aplicando la fórmula (2.7) resultan:

$$c_0 = f[x_1] = f_1,$$

$$c_1 = f[x_1, x_2] = \frac{f[x_2] - f[x_1]}{x_2 - x_1} = \frac{f_2 - f_1}{x_2 - x_1},$$

$$c_2 = f[x_1, x_2, x_3] = \frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1},$$

$$c_3 = f[x_1, x_2, x_3, x_4] = \frac{f[x_2, x_3, x_4] - f[x_1, x_2, x_3]}{x_4 - x_1},$$

$$\vdots$$

Así sucesivamente.

El arreglo que se muestra en la Figura 2.5, facilita los cálculos de dichas constantes.

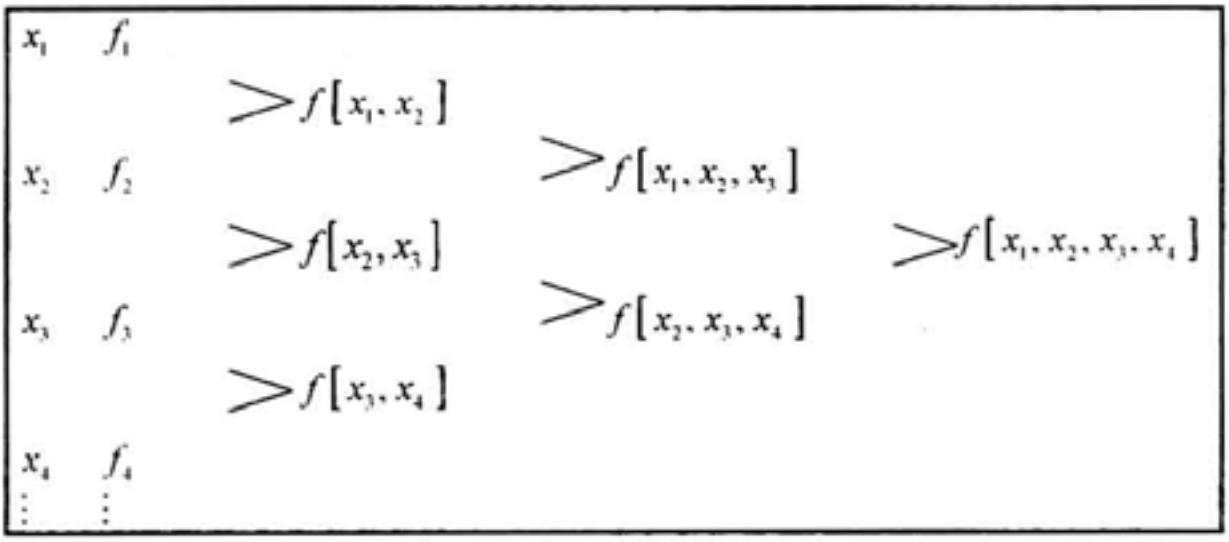


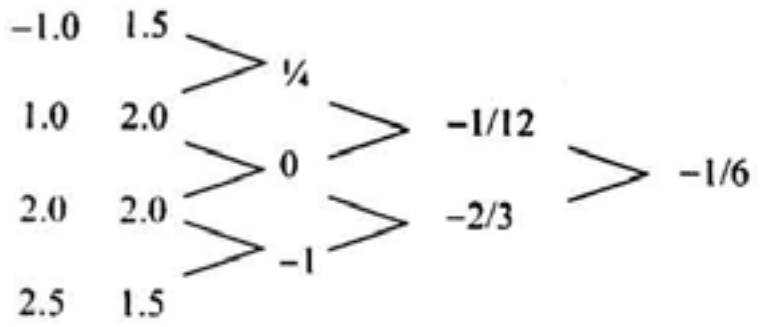
Figura 2.5. Esquema para calcular las diferencias divididas de Newton.

Ejemplo 2.3: Para el siguiente conjunto de datos

x	−1.0	1.0	2.0	2.5
f	1.5	2.0	2.0	1.5

a) Determine las diferencias divididas de Newton c_0, c_1, c_2 y c_3.
b) Obtenga el polinomio que interpola los datos usando la fórmula del polinomio de Newton.
c) Calcule el valor del polinomio de interpolación en el punto $x^* = 0.3$.

Solución: a) Usaremos el arreglo de la Figura 2.5, para obtener las diferencias divididas de Newton.



Así que las diferencias divididas son: $c_0 = 1.5$, $c_1 = 1/4$, $c_2 = -1/12$ y $c_3 = -1/6$.

b) El polinomio que interpola los datos es de grado 3 y está dado por

$$P(x) = \frac{3}{2} + \frac{1}{4}(x+1) - \frac{1}{12}(x+1)(x-1) - \frac{1}{6}(x+1)(x-1)(x-2)$$

Simplificando resulta: $P(x) = -\frac{1}{12}\left(2x^3 - 3x^2 - 5x - 18\right)$.

c) Evaluamos ahora el polinomio en $x^* = 0.3$

$$P(0.3) = -\frac{1}{12}\left(2(0.3)^3 - 3(0.3)^2 - 5(0.3) - 18\right) = 1.643$$

Por lo tanto, $P(0.3) = 1.643$. Este mismo resultado se obtuvo en el Ejemplo 2.2(c) con la fórmula de Lagrange, como debe de ser. ■

Es mucho mejor disponer de un algoritmo para programar el esquema de la Figura 2.5. Puesto que un problema importante en análisis numérico es la limitación de memoria de la máquina, es recomendable no crear demasiadas variables auxiliares en el proceso de cálculo. El arreglo de la Figura 2.5, sugiere que es suficiente con 3 variables aparte de los datos, es decir, n variables aparte de los $n + 1$ parejas de datos. Por cuestiones técnicas se crearán $n + 1$ variables. El siguiente algoritmo está basado en estas ideas.

Algoritmo 2 para determinar las diferencias divididas de Newton: $c_0, c_1, ..., c_n$.

[*La entrada consta de los datos* $x_1, x_2, ..., x_{n+1}$ *y los valores* $f_1, f_2, ..., f_{n+1}$. *La salida son las diferencias divididas de Newton:* $d_k = f[x_1, x_2, ..., x_k]$, $1 \le k \le n+1$. *Note que* $c_k = d_{k+1}$, $0 \le k \le n$.]

Entrada: $x_1, x_2, ..., x_{n+1}, f_1, f_2, ..., f_{n+1}$.
Salida: $d_1, d_2, ..., d_{n+1}$.

1. Para $j = 1, 2, ..., n + 1$
   1.1. $d_j = f_j$
2. Para $k = 1, 2, ..., n$
   2.1. Para $j = n + 1, n, ..., k + 1$
      2.1.1. $d_j = (d_j - d_{j-1}) / (x_j - x_{j-k})$
3. Alto.

Observamos que en el paso 1, se crean las variables de salida $d_1, d_2, ..., d_{n+1}$ cuyos valores iniciales son las ordenadas $f_1, f_2, ..., f_{n+1}$ de los datos, pero al final contendrán las diferencias divididas de Newton. En el paso 2, para $k = 1$ se modifican los valores de $d_{n+1}, d_n, ..., d_2$, tomando los valores de $f[x_n, x_{n+1}]$, $f[x_{n-1}, x_n]$, ..., $f[x_1, x_2]$ respectivamente. Estos son justamente las primeras diferencias divididas que aparecen en la 3ª columna, de "abajo hacia arriba" en el diagrama de la Figura 2.5. En este proceso, no se modifica el valor de d_1 que se mantiene como f_1. Para $k = 2$, se modifican los valores de $d_{n+1}, d_n, ..., d_3$, tomando los valores de $f[x_{n-1}, x_n, x_{n+1}]$, $f[x_{n-2}, x_{n-1}, x_n]$, ..., $f[x_1, x_2, x_3]$ respectivamente. Estos son las segundas diferencias divididas de Newton, que aparecen en la 4ª columna de "abajo hacia arriba" en el diagrama de la Figura 2.5. En este proceso, no se modifican los valores de d_1 y d_2 que se mantiene como f_1 y $f[x_1, x_2]$ respectivamente. Al continuar con el proceso se mantienen los valores de d_1, d_2, d_3, ..., hasta llegar a $k = n$ donde se modifica solamente el valor de d_{n+1} que toma el valor de la última diferencia dividida $f[x_1, x_2, ..., x_{n+1}]$.

Estamos interesados en evaluar el polinomio de Newton en puntos de interés. También evaluarla en varios puntos de interés para su visualización gráfico. Daremos un algoritmo que lo evalúa en varios puntos de interés: $w_1, w_2, ..., w_m$.

Algoritmo 3 para evaluar el polinomio $P(x)$ usando la fórmula de Newton en m puntos de interés: $w_1, w_2, ..., w_m$.

[*La entrada son los datos $x_1, x_2, ..., x_{n+1}$, la salida $d_1, d_2, ..., d_{n+1}$ del algoritmo 2 y los puntos de interés $w_1, w_2, ..., w_m$. La salida son los puntos de interés $w_1, w_2, ..., w_m$ y los valores del polinomio en dichos puntos de interés: $pw_1 = P(w_1), pw_2 = P(w_2), ..., pw_m = P(w_m)$.*]

Entrada: $x_1, x_2, ..., x_{n+1}, d_1, d_2, ..., d_{n+1}, w_1, w_2, ..., w_m$.
Salida: $w_1, w_2, ..., w_m, pw_1 = P(w_1), pw_2 = P(w_2), ..., pw_m = P(w_m)$.

1. Para $j = 1, 2, ..., m$
   1.1. $pw_j = d_{n+1}$
   1.2. Para $i = n, n-1, ..., 1$
      1.2.1. $pw_j = pw_j (w_j - x_i) + d_i$
2. Alto.

Observamos que para j=1, se calcula el valor del polinomio de interpolación en w_1, es decir, el valor de $P(w_1)$ que lo denotaremos por pw_1. Comenzamos en el paso 1.1, asignando el valor de d_{n+1} a pw_1. En el paso 1.2, iteramos sobre i desde n, $n-1$ hasta 1. Para $i = n$, pw_1 toma el valor de $pw_1(w_1 - x_n) + d_n$, que es igual a $d_{n+1}(w_1 - x_n) + d_n$. Para $i = n-1$, pw_1 toma el valor de $pw_1(w_1 - x_{n-1}) + d_{n-1}$, que es ahora igual a $d_{n+1}(w_1 - x_n)(w_1 - x_{n-1}) + d_n(w_1 - x_{n-1}) + d_{n-1}$. Al llegar hasta $i = 1$, pw_1 tomará el valor de $pw_1(w_1 - x_1) + d_1$, que resulta ser igual a

$$d_{n+1}(w_1 - x_n)(w_1 - x_{n-1}) \ldots (w_1 - x_1) + d_n(w_1 - x_{n-1}) \ldots (w_1 - x_1) + \ldots + d_1,$$

que salvo el orden de los sumandos y de los factores, es exactamente el polinomio de Newton evaluado en w_1 como en (2.5), es decir, el último valor que toma pw_1 es justamente $P(w_1)$. Se repite el proceso para obtener los valores de $P(w_2), ..., P(w_m)$.

A continuación presentamos el programa **Polinewton.m** hecho en Matlab que integra los algoritmos 2 y 3. Es decir, calcula las diferencias divididas de Newton, presenta la visualización gráfico del polinomio de interpolación sobre el intervalo $[x_1, x_{n+1}]$, y evalúa el polinomio de interpolación en m puntos de interés: $w_1, w_2, ..., w_m$.

```
% Programa Polinewton.m que grafica el polinomio de interpolación de un
% conjunto de datos.
%
      clear; clc
      format long
%
   xd=input('Dame las abscisas de los datos como [x1,x2,...,xn]: ');
   fd=input('Dame las ordenadas de los datos como [f1,f2,...,fn]: ');
      N=length(xd);
%
% Cálculo de las diferencias divididas d1, d2, ..., dN.
      d=fd;
      for k=1:N-1;
            for j=N:-1:k+1;
                  d(j)=(d(j)-d(j-1))/(xd(j)-xd(j-k)); end; end;
%
% Impresión de las diferencias divididas.
%
      d=d'
%
% Evaluación del polinomio de interpolación para su visualización
% gráfico.
%
      x=[xd(1)-0.4:0.1:xd(end)+0.5];
%
      for j=1:length(x);
            p(j)=d(N);
                  for i=N-1:-1:1;
                        p(j)=p(j)*(x(j)-xd(i))+d(i); end; end
%
% Evaluación del polinomio de interpolación en puntos de interés.
%
      w=input('Dame los puntos a interpolar como [w1,w2,...,wm]: ');
%
      for j=1:length(w);
            pw(j)=d(N);
                  for i=N-1:-1:1;
                        pw(j)=pw(j)*(w(j)-xd(i))+d(i); end; end
%
% Visualización gráfico del polinomio de interpolación, los datos y
% los puntos de interés.
%
      plot(xd,fd,'Ok',x,p,'k',w,pw,'Ok');grid
      pause
%
% Impresión de los resultados en los puntos de interés.
%
ww=[w',pw']
```

Ejemplo 2.4: Consideramos el conjunto de datos del Ejemplo 2.2

x	−3.0	−1.0	1.0	2.0	2.5	3.0
f	1.0	1.5	2.0	2.0	1.5	1.0

Presentamos las gráficas de los polinomios de grados 4 y 5 que interpolan los datos anteriores, y que se usaron para aproximar el valor de $f(0.3)$. Los polinomios de grados 1, 2 y 3 se trabajaron en el Ejemplo 2.2 con la fórmula de Lagrange. En este ejemplo los polinomios de grado 4 y 5 se evaluaron con el programa **Polinewton.m.** Los resultados se muestran en la Tabla 2.2 y en la Figura 2.6. Veremos en la siguiente Sección que la fórmula de Newton es más eficiente que la fórmula de Lagrange, por lo que su uso es más común en la práctica.

Los valores de los polinomios de interpolación en $x^* = 0.3$ lo resumimos en la Tabla 2.2.

Grado del polinomio de interpolación	1	2	3	4	5
Valor del polinomio en $x^* = 0.3$	1.825	1.901	1.643	1.218	1.451

Tabla 2.2. Valores de los polinomios de interpolación de los datos del Ejemplo 2.4 en x^*=0.3.

De las Figuras 2.2, 2.3, 2.4 y 2.6 se observa que los valores obtenidos por los polinomios de grado 1 y 2 se ajustan mejor a la tendencia de los datos, mientras que los valores obtenidos por los polinomios de grados 3, 4 y 5 se alejan de la tendencia global de los datos. El polinomio de grado 4 tiene una oscilación grande y su valor es el que más se aleja de la tendencia de los datos. De hecho los polinomios de alto grado tienden a ser mal condicionados, es decir, tienden a ser altamente sensibles a los errores de redondeo, como se mostró en el Ejemplo 1.22 del Capitulo I.

2.2.4 Comparación de los Algoritmos

La diferencia entre las fórmulas de Lagrange y de Newton, radica en la cantidad de operaciones aritméticas que requieren, para evaluar el polinomio de interpolación $P(x)$ en puntos de interés. Para hacer una comparación entre las dos fórmulas, veamos las eficiencias de los Algoritmos 2 y 3. La eficiencia del Algoritmo 2 que calcula las diferencias divididas de Newton, está determinada en el paso 2.

2. *Para* $k = 1, 2, ..., n$

2.1. Para $j = n + 1, n, ..., k + 1$

2.1.1. $d_j = (d_j - d_{j-1}) / (x_j - x_{j-k})$

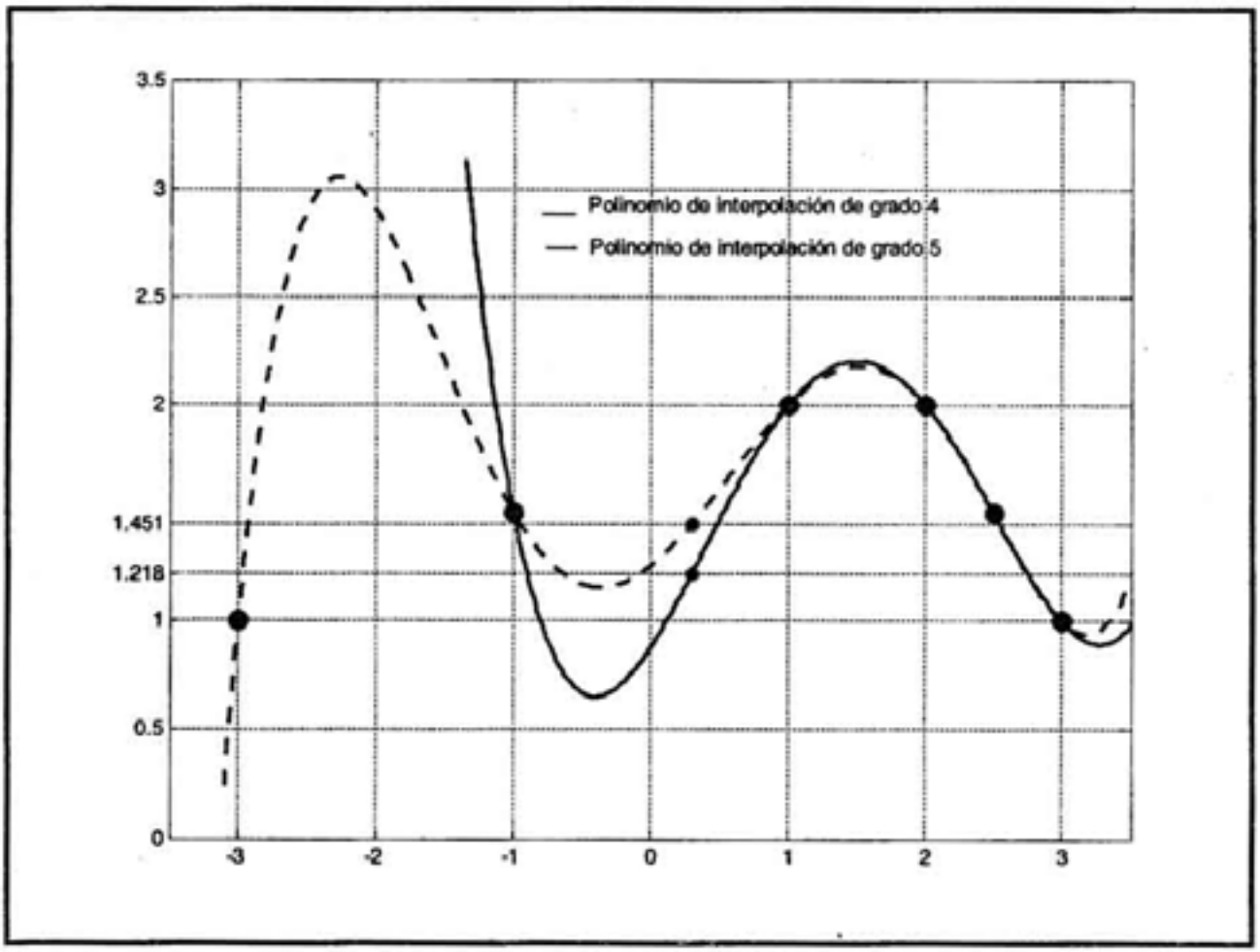


Figura 2.6. Gráfica de los polinomios de interpolación de grados 4 y 5 de los datos del Ejemplo 2.4.

La cantidad de operaciones que realiza es

$$(2A+1D)(n+(n-1)+...+1) = (2A+D)\frac{n(n+1)}{2} = (n^2+n)A + \frac{n^2+n}{2}D.$$

La eficiencia del Algoritmo 3 que evalúa la fórmula de Newton en un solo punto x^*, está determinado por el paso 1.2 (es el caso, $m = 1$ y tomando $w = w_1 = x^*$)

1.2. Para $i = n, n-1, \ldots, 1$

1.2.1. $pw = pw(x^* - x_i) + d_i$

que realiza $(2A + 1M)n = 2nA + nM$ operaciones.

La eficiencia en la evaluación de la fórmula de Newton es la suma de las eficiencias de los Algoritmos 2 y 3, que resulta

$$(n^2 + 3n)A + nM + \frac{n^2 + n}{2} D.$$

Recordemos que la eficiencia de la fórmula de Lagrange es

$$(2n^2 + 3n + 1)A + (n^2 + 2n + 1)M + (n^2 + n)D.$$

Así que podemos comparar los algoritmos que evalúan el polinomio de interpolación $P(x)$ usando la fórmula de Lagrange y la fórmula de Newton. Algunos valores de la cantidad de operaciones aritméticas de punto flotante que necesita realizar cada fórmula, para evaluar P en un solo punto de interés x^* se muestran en la Tabla 2.3 siguiente

n	$n+1$	Fórmula de Lagrange	Fórmula de Newton
3	4	$28A + 16M + 12D$	$18A + 3M + 6D$
5	6	$66A + 36M + 30D$	$40A + 5M + 15D$
10	11	$231A + 121M + 110D$	$130A + 10M + 55D$

Tabla 2.3. Comparación de las Fórmulas de Lagrange y Newton al evaluar en un punto x^*.

La fórmula de Newton es mucho más eficiente que la fórmula de Lagrange, cuando se requiere evaluar en varios puntos de interés $w_1, w_2, \ldots, w_m$, pues, para la fórmula de Newton se ejecuta el Algoritmo 2 solamente una vez. En este caso tenemos:

Eficiencia de la fórmula de Lagrange:

$$m(2n^2 + 3n + 1)A + m(n^2 + 2n + 1)M + m(n^2 + n)D$$

Eficiencia de la fórmula de Newton: $(n^2 + n + 2mn)A + mnM + \frac{n^2 + n}{2} D.$

Algunos valores comparativos se muestran en la Tabla 2.4.

m	n	$n+1$	Fórmula de Lagrange	Fórmula de Newton
30	3	4	$840\,A + 480\,M + 360\,D$	$192\,A + 90\,M + 6\,D$
50	5	6	$3\,300\,A + 1\,800\,M + 1\,500\,D$	$530\,A + 250\,M + 15\,D$
50	10	11	$11\,550\,A + 6\,050\,M + 5\,500\,D$	$1\,110\,A + 500\,M + 55\,D$

Tabla 2.4. Comparación de las fórmulas de Lagrange y Newton al evaluar en m puntos de interés.

2.2.5 Estimación del Error de Truncamiento

Recordamos el enunciado del Teorema 2.2: Si f es una función con derivadas continuas hasta de orden $n + 1$, en algún intervalo que contiene los puntos $x_1, x_2, \ldots, x_{n+1}$, y si P es el polinomio de grado $\leq n$ que interpola a f en dichos puntos, entonces, la función de error de aproximación está dado por la fórmula (2.1)

$$E_t(x) = f(x) - P(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}(x - x_1)(x - x_2)\cdots(x - x_{n+1}),$$

para toda x, y para algún punto ξ_x en el intervalo que contiene los puntos $x_1, x_2, \ldots, x_{n+1}$.

Es importante notar que hay dos formas de estimar el error de truncamiento en el Teorema 2.2. La primera es usando la función f y el polinomio P cuando ambos son conocidos. La segunda forma es usando la derivada de orden $n + 1$ de f cuando se conoce, y los datos del problema.

Veamos dos ejemplos.

Ejemplo 2.5: Con la función de Runge $f(x) = \dfrac{1}{1+25x^2}$, $-1 \leq x \leq 1$, generamos la siguiente tabla

x	-1.0	-0.8	-0.6	-0.4	-0.2	0
f	3.846×10^{-2}	5.882×10^{-2}	0.1000	0.2000	0.5000	1.000

Aplicando la fórmula de Newton con $n = 5$ y para $x^* = -0.5$, obtenemos

$$p = P(-0.5) = 0{,}13338$$

Por otro lado, el valor exacto es $f^* = f(-0.5) = \dfrac{1}{1+25(-0.5)^2} = 0.13793$.

a) Calculemos el error de aproximación de f^* por p, usando la función de Runge y el polinomio $P(x)$. En este caso

$$E_T(-0.5) = f(-0.5) - P(-0.5) \cong 0.13793 - 0.13338 = 0.00455 < 0.005,$$

lo que indica que hay dos lugares decimales correctos en la aproximación, sin embargo es poca exactitud. En la Figura 2.7, se muestra la situación geométrica de la aproximación de la función de Runge por el polinomio de interpolación de grado 5. Al ampliar una vecindad del punto (–0.5, 0.13338) se muestra la poca exactitud al aproximar el valor de $f(-0.5)$.

b) Estimamos ahora el error de truncamiento usando el Teorema 2.2, con la derivada de orden 6 de la función de Runge y los datos de la tabla.

$$f^{(6)}(x) = 3750000\frac{-3+5325x^2-353125x^4+546875x^6}{\left(1+25x^2\right)^7}.$$

Elegimos $\xi_{1*} = -0.5$,

$f^{(6)}(-0.5) = -43443.345$ es demasiado grande y es la causante de la poca exactitud en la aproximación.

$$\begin{aligned}\text{prod} &= (-0.5+1.0)(-0.5+0.8)(-0.5+0.6)(-0.5+0.4)(-0.5+0.2)(-0.5)\\ &= -2.25\times10^{-4},\end{aligned}$$

es pequeño, pero

$$E_T(-0.5) \cong \frac{-43443.345}{6!}\left(-2.25\times10^{-4}\right) \cong 0.01358 < 0.05.$$

La estimación es muy grande comparada con el valor exacto del error obtenido en el inciso (a). Pronostica solamente un decimal correcto en la aproximación, el cual es incorrecto. ■

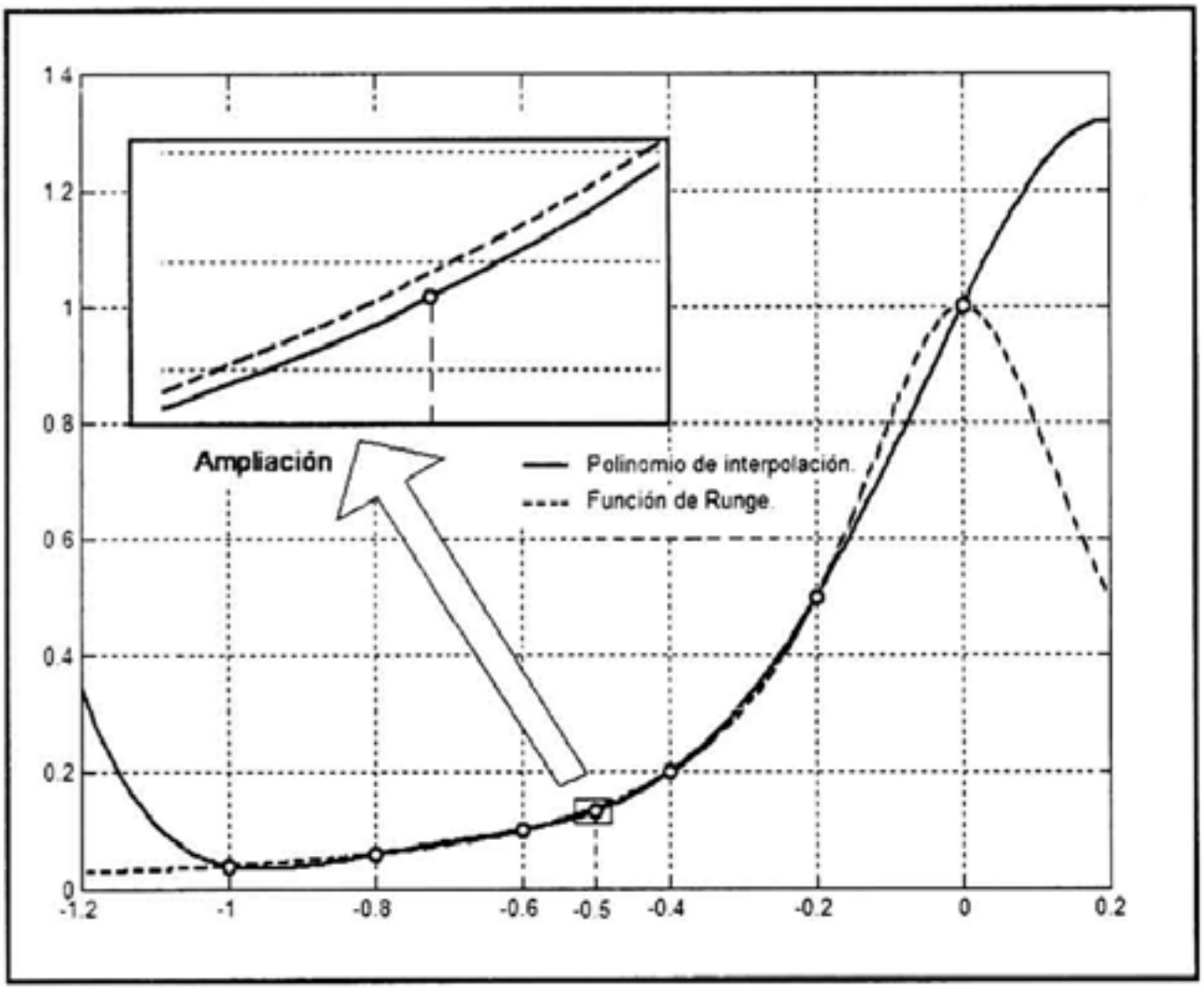


Figura 2.7. Gráfica del polinomio de interpolación de grado 5 con la función de Runge.

Ejemplo 2.6: En este caso se trabaja con la función $f(x) = \cos x$, los cálculos se realizan con Matlab. Consideramos los datos:

$x = [0.2 : 0.2 : 1.8] = [0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8]$; $n = 8$.

$f = \cos(x)$;

Al evaluar el polinomio $P(x)$ de grado 8 que interpola el conjunto de datos, en $x^* = 0.9$ con la fórmula de Newton da $P(0.9) = 0.621609968044 56$

El valor exacto en $x^* = 0.9$ es $\cos(0.9) = 0.621609968270 66$.

a) El error de truncamiento calculado directamente es,

$$E_T(0.9) = \cos(0.9) - P(0.9) \cong 2.2610 \times 10^{-10} < 5 \times 10^{-10}.$$

b) Estimamos ahora el error de truncamiento según el Teorema 2.2.

$f^{(9)}(0.9) = -\text{sen}\ (0.9) \cong -0.783$

$$\text{prod} = (0.9-0.2)(0.9-0.4)(0.9-0.6)(0.9-0.8)(0.9-1.0)(0.9-1.2)(0.9-1.4)$$

$$(0.9-1.6)(0.9-1.8) = -9.92 \times 10^{-5}.$$

Por lo tanto,

$$E_T(0.9) \cong \frac{-0.783}{9!}\left(-9.92 \times 10^{-5}\right) \cong \left(-2.16 \times 10^{-6}\right)\left(-9.92 \times 10^{-5}\right)$$

$$\cong 2.14 \times 10^{-10} < 5 \times 10^{-10}.$$

c) Otra forma de estimar el error de truncamiento es acotando primero la derivada de orden $n + 1$ de f. En este caso,

$\left| f^{(9)}(\xi_{x^*}) \right| = \left| -sen(\xi_{x^*}) \right| \leq 1$, y

$$\left| E_T(0.9) \right| \leq \frac{1}{9!}\left| -9.92 \times 10^{-5} \right| \cong 2.73 \times 10^{-10} < 5 \times 10^{-10}.$$

En cualesquiera de los tres casos se pronostican 9 lugares decimales correctos de $P(0.9)$ como aproximación del cos(0.9). Este resultado es muy bueno, porque el Polinomio de Taylor que aproxima a cos(x) para $|x| \leq 1$, que también calcula 9 decimales correctamente redondeados del cos(0.9) es el de grado ≥ 10, y tiene por lo menos 2 grados más que nuestro polinomio de interpolación. ■

2.3 SPLINE CÚBICO

El objetivo es mostrar una técnica para resolver el problema de interpolación planteado en la Sección 2.1: Conocidos los valores $f_k = f(x_k)$ de alguna función "complicada" $f(x)$ en n puntos distintos $x_1 < x_2 < \ldots < x_n$, determinar una función simple $s(x)$ tal que $s(x_k) = f_k$, para $k = 1, 2, \ldots, n$.

Entre los diferentes tipos de interpolación (polinomios, funciones racionales, trigonométricas, polinomios por tramos, etc.), quizás los más populares debido a sus propiedades son los polinomios cúbicos por tramos y en especial los splines cúbicos (naturales), por las siguientes razones:

1. Son fáciles de calcular y evaluar.
2. Son fáciles de derivar y sus derivadas aproximan a las derivadas de $f(x)$.
3. Son fáciles de integrar y se usan para aproximar la integral de $f(x)$.
4. Modelan con suavidad la tendencia de un conjunto de datos.
5. Finalmente, uno de los más importante desarrollos en Análisis Numérico durante los últimos 30 años, ha sido el uso de los Splines a través del Método de Elemento Finito, para resolver ecuaciones diferenciales parciales.

2.3.1 Historia de los Splines

Tomado de NA. Digest, V.98, # 26. 19 de Julio de 1998. Mail to na.digest@na-net.ornl.gov
Information about NA-NET: Mail to na.help@na-net.ornl.gov
URL for the World Wide Web: http://www.netlib.org/na-net/na_home.html

Hace dos semanas puse aquí una pregunta acerca de las conexiones entre el desarrollo de aproximaciones de splines y el diseño de cuerpos automotores, y recibí cerca de 30 respuestas, todas ellas muy informativas. Dado que muchos de ellos me pidieron que mostrara lo que había aprendido, decidí escribir un breve resumen y subirlo al compendio. Esta es la razón de estas notas.

Se acepta comúnmente que la primera referencia matemática de los splines es el trabajo de Schoenberg [S], donde probablemente fue el primer lugar donde el término "spline" se usó en conexión con la aproximación polinomial tenue por piezas. Sin embargo, estas ideas tienen sus raíces en las industrias de aviación y construcción de barcos. En el reenvío a [BBB], Robin Forrest describe el "localizar", una técnica en la industria británica de aviación usada durante la Segunda Guerra Mundial, para construir plantillas para aviones, pasando tablones de madera delgadas a través de puntos en el suelo de un local de diseño grande. Las plantillas estarían puestas en puntos discretos (llamados "patos" por Forrest; Schoenberg usa "perros" o "ratas") y entre estos puntos asumiría formas de un mínimo de energía de tensión. De acuerdo a Forrest, una motivación posible para un modelo matemático de este proceso fue la pérdida potencial de los componentes del diseño críticos en toda una nave si el local fuera bombardeado por el enemigo. Esto promovió el "localizamiento cónico", el cual usaba secciones cónicas para modelar la posición de la curva entre los "patos". El "localizamiento cónico"

fue reemplazado por lo que llamaríamos splines a principios de los 60's basados en el trabajo de J. C. Ferguson de Boeing y (tiempo después) por M. A. Sabin de British Aircraft. Lo que considero muy interesante es que Forrest dice que la palabra "spline" viene de un dialecto Anglicano del Este.

El uso de splines para modelar cuerpos automotores parece tener un sinfín de comienzos independientes. El crédito lo piden a nombre de De Casteljau de Citroen, Bezier de Renault, and Birkhoff, Garabedian, y de Boor de General Motors (GM), todos ellos por sus trabajos realizados a finales de la década de 1950's o principios de los años de 1960's. Al menos uno de los trabajos de De Casteljau fue publicado, pero no ampliamente, en 1959. La obra de De Boor de GM resultó en un conjunto de escritos siendo publicados a principios de los 60's, incluyendo algo del trabajo fundamental sobre los B-splines. El trabajo también fue hecho en Pratt y Whitney Aircraft, donde dos de los autores de [ANW] (el primer libro en sí acerca de splines) fueron contratados, y el Modelo Basin de David Taylor, hecho por Feodor Theilheimer. El trabajo en GM está perfectamente detallado en el artículo [B] y la retrospectiva [Y]. También se me indicó el artículo [BdB] por mucha gente, pero nuestra biblioteca no tiene ese volumen, así que no he podido verlo por mi cuenta. Paul Davis resumió algo de este material en SIAM News en 1996: ver [D].

De nuevo, muchas gracias a todos los que me mandaron mensajes y sugerencias.

Referencias

[ANW] Ahlberg, Nielson, and Walsh, The Theory of Splines and Their Applications, 1967.

[B] Birkhoff, "Fluid dynamics, reactor computations, and surface representation," in A History of Scientific Computation (Steve Nash, editor), 1990.

[BBB] Bartels, Beatty, and Barsky, An Introduction to Splines for Use in Computer Graphics and Geometric Modeling, 1987.

[BdB] Birkhoff and de Boor, "Piecewise polynomial interpolation and approximation," Proc. General Motors Symposium of 1964, H.L. Garabedian, ed., Elsevier, New York and Amsterdam, 1965, pp. 164-190.

[D] Davis, "B-splines and Geometric design," SIAM News, vol. 29, no. 5; available at http://www.wpi.edu/~pwdavis/sinews/spline17.htm.

[S] Schoenberg, "Contributions to the problem of approximation of equidistant data by analytic functions," Quart. Appl. Math., vol. 4, pp. 45-99 and 112-141.

[Y] Young, "Garrett Birkhoff and applied mathematics," Notices of the AMS, vol. 44, no. 11, pp. 1446-1449.

Tradujo Saúl Suárez Chávez, alumno del 4o. semestre de la Licenciatura en Computación (1998).

2.3.2 Spline Cúbico Natural

Sean $a = x_1 < x_2 < ... < x_n = b$ puntos dados y $f_1, f_2, ..., f_n$, valores de una función $f(x)$ en dichos puntos. Un spline cúbico natural que interpola a estos datos es una función $s(x)$, definido sobre el segmento $[a, b]$, con las propiedades siguientes:

I. $s(x_k) = f_k$, $k = 1, 2, ..., n$.

II. $s(x)$, $s'(x)$ y $s''(x)$ son continuas sobre el segmento $[a, b]$.

III. $s(x)$ es un polinomio cúbico diferente sobre cada uno de los segmentos $[x_k, x_{k+1}]$.

IV. Si $g(x)$ es cualquier otra función que satisface las propiedades (I) y (II), entonces el spline cúbico natural s es la función más "suave", en el sentido de que satisface la desigualdad

$$\int_a^b [s''(x)]^2 dx \leq \int_a^b [g''(x)]^2 dx = I(g) \quad \text{"Propiedad de norma mínima".}$$

Precisamente debido a esta última propiedad las funciones spline recibieron tal nombre, ya que durante mucho tiempo, para dibujar una curva suave que pasara por un conjunto de puntos prefijados, los ingenieros utilizaban un instrumento llamado spline. Este instrumento mecánico estaba formado por una varilla flexible de acero a la cual se le sujetaban ciertos pesos que hacían que la varilla se flexionara y la curva resultante s pasara por los puntos. De este modo, el spline mecánico s minimizaba la energía de tensión, que aproximadamente es proporcional a $I(g)$. De ahí que por analogía, la función que minimiza $I(g)$ recibiera el nombre de spline. Ver Figura 2.8.

La propiedad IV es más difícil de formular, y lo reemplazaremos por otra más simple. Para ello, observamos que cada polinomio cúbico es de la forma

$$a x^3 + b x^2 + c x + d$$

con 4 coeficientes libres en cada uno, y si ponemos:

$$s(x) = s_k(x), \quad x_k \leq x \leq x_{k+1}, \quad k = 1, 2, ..., n-1,$$

resultan $n-1$ polinomios cúbicos. Por lo tanto, para determinar $s(x)$ se necesitan calcular $4(n-1) = 4n-4$ coeficientes. Necesitamos igual número de condiciones

para resolver el problema. Como cada polinomio interpola a la función $f(x)$ en los dos puntos extremos de cada segmento $[x_i, x_{i+1}]$, de aquí resultan $2(n-1)$ condiciones.

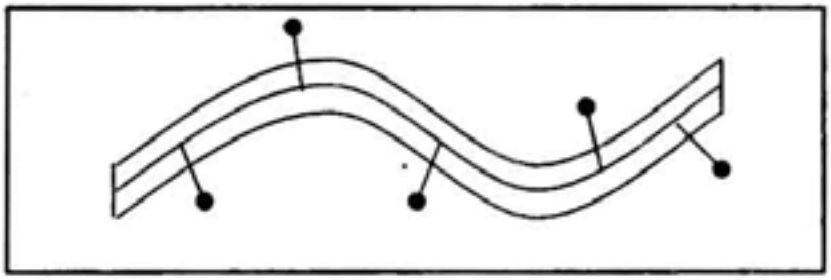


Figura 2.8. Spline mecánico.

Además $s'(x)$ y $s''(x)$ deben ser continuas en los nodos $x_2, x_3, \ldots, x_{n-1}$, por lo que se cumplen las condiciones

$$s'_i(x_{i+1}) = s'_{i+1}(x_{i+1}) \quad \text{y} \quad s''_i(x_{i+1}) = s''_{i+1}(x_{i+1}), \quad k = 1, 2, \ldots, n-2,$$

esto da $2(n-2)$ condiciones.

En total se tienen $2(n-1)+2(n-2) = 4n-6$ condiciones para determinar $4n-4$ coeficientes. Se necesitan 2 condiciones más, para que se pueda determinar de manera única el spline. Podemos pedir

IV'. $s''(x_1) = 0$ y $s''(x_n) = 0$, ó V. $s''(x_1) = f'(x_1)$ y $s''(x_n) = f'(x_n)$.

La propiedad IV' implica la propiedad de norma mínima IV y lo usaremos para determinar el spline $s(x)$. Es mejor el uso del spline con condiciones dadas en V, pues da una mejor aproximación de $f(x)$ cerca de x_1 y de x_n (a menos que $f''(x) \cong 0$ cerca de x_1 y de x_n). Sin embargo, en la práctica es difícil conocer la derivada de $f(x)$, cuando $f(x)$ misma es desconocida.

2.3.3 Cálculo del Spline Cúbico Natural

El spline $s(x)$ es un polinomio cúbico sobre $[x_i, x_{i+1}]$, lo que implica que $s''(x)$ es una línea recta sobre $[x_i, x_{i+1}]$. Por lo tanto, $s''(x)$ es la recta que pasa a través de los puntos $(x_i, s''(x_i))$ y $(x_{i+1}, s''(x_{i+1}))$, y está dada por

$$s''(x) = s''(x_k) + \frac{s''(x_{k+1}) - s''(x_k)}{x_{k+1} - x_k}(x - x_k), \qquad \text{..... (2.8)}$$

Aplicando el Teorema Fundamental del Cálculo, al integrar $s''(t)$ sobre el segmento $[x_k, x]$, con $x \le x_{k+1}$, resulta

$$s'(x) = s'(x_k) + s''(x_k)(x - x_k) + \frac{s''(x_{k+1}) - s''(x_k)}{2(x_{k+1} - x_k)}(x - x_k)^2. \qquad \text{..... (2.9)}$$

Similarmente, integrando $s'(t)$ sobre el segmento $[x_k, x]$ con $x \le x_{k+1}$, resulta

$$s(x) = f_k + s'(x_k)(x - x_k) + \frac{s''(x_k)}{2}(x - x_k)^2 + \frac{s''(x_{k+1}) - s''(x_k)}{6(x_{k+1} - x_k)}(x - x_k)^3. \qquad \text{..... (2.10)}$$

Para conocer los valores del spline en cualquier punto x, se necesita conocer los valores de las derivadas $s'(x_k)$, $s''(x_k)$ y $s''(x_{k+1})$. Pongamos

$$h_k = x_{k+1} - x_k, \text{ para } k = 1, 2, ..., n-1,$$
$$s'_k = s'(x_k) \text{ y } s''_k = s''(x_k), \text{ para } k = 1, 2, ..., n.$$

Sustituyendo $x = x_{k+1}$ en la ecuación (2.10), obtenemos

$$f_{k+1} = f_k + s'_k h_k + \frac{s''_k}{2} h_k^2 + \frac{s''_{k+1} - s''_k}{6h_k} h_k^3$$

de donde,

$$s'_k = \frac{f_{k+1} - f_k}{h_k} - \frac{s''_k}{3} h_k - \frac{s''_{k+1}}{6} h_k. \qquad \text{..... (2.11)}$$

La fórmula (2.11) calcula la primera derivada del spline en los nodos x_k para $1 \le k \le n-1$. En la ecuación (2.9), reemplazamos primero k por $k-1$ y luego x por x_k para obtener

$$s'_k = s'_{k-1} + s''_{k-1} h_{k-1} + \frac{s''_k - s''_{k-1}}{2h_{k-1}} h_{k-1}^2$$

de donde,

$$s'_k = s'_{k-1} + \frac{1}{2}(s''_k + s''_{k-1}) h_{k-1}. \qquad \text{..... (2.12)}$$

Igualando las ecuaciones (2.11) y (2.12), y sustituyendo s'_{k-1} con su equivalente en (2.11), tenemos

$$\frac{f_{k+1}-f_k}{h_k}-\frac{s''_k}{3}h_k-\frac{s''_{k+1}}{6}h_k=\frac{f_k-f_{k-1}}{h_{k-1}}-\frac{s''_{k-1}}{3}h_{k-1}-\frac{s''_k}{6}h_{k-1}+\frac{1}{2}\left(s''_k+s''_{k-1}\right)h_{k-1}$$

de donde,

$$\left.\begin{aligned} &h_{k-1}\,s''_{k-1}+2\left(h_{k-1}+h_k\right)s''_k+h_k\,s''_{k+1}=d_k. \\ &d_k=6\left(\frac{f_{k+1}-f_k}{h_k}-\frac{f_k-f_{k-1}}{h_{k-1}}\right), \quad k=2,\ldots,n-1. \end{aligned}\right\} \quad \ldots\ldots \quad (2.13)$$

El sistema (2.13) es un sistema lineal de $n-2$ ecuaciones con $n-2$ incógnitas:

$$s''_2,\ s''_3,\ \ldots,\ s''_{n-1},$$

con las condiciones adicionales: $s''_1=s''(x_1)=0$ y $s''_n=s''(x_n)=0$.

Si ponemos A como la matriz de coeficientes del sistema y los vectores s y d como:

$$A=\begin{bmatrix} 2(h_1+h_2) & h_2 & 0 & \cdots & 0 \\ h_2 & 2(h_2+h_3) & h_3 & \cdots & 0 \\ 0 & h_3 & 2(h_3+h_4) & \cdots & 0 \\ \cdots & \cdots & \cdots & \ddots & \cdots \\ 0 & 0 & 0 & \cdots & 2(h_{n-2}+h_{n-1}) \end{bmatrix},$$

$$s=\begin{bmatrix} s''_2 \\ s''_3 \\ s''_4 \\ \vdots \\ s''_{n-1} \end{bmatrix} \qquad \text{y} \qquad d=\begin{bmatrix} d_2 \\ d_3 \\ d_4 \\ \vdots \\ d_{n-1} \end{bmatrix}$$

el sistema (2.13) se puede escribir como

$$A\,s = d \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.14)$$

La matriz A del sistema (2.14) es tridiagonal y de diagonal dominante por filas, por lo tanto, el sistema tiene exactamente una solución. El algoritmo de eliminación Gaussiana sin pivoteo, genera las siguientes fórmulas recurrentes:

$$\left.\begin{aligned} a_2 &= 2(h_1 + h_2) \\ a_k &= 2(h_{k-1} + h_k) - \frac{h_{k-1}^2}{a_{k-1}}; \quad k = 3,4,\ldots,n-1. \end{aligned}\right\} \qquad \ldots\ldots\ldots \quad (2.15)$$

$$\left.\begin{aligned} b_2 &= d_2 \\ b_k &= d_k - \frac{h_{k-1} b_{k-1}}{a_{k-1}}; \quad k = 3,4,\ldots,n-1. \end{aligned}\right\} \qquad \ldots\ldots\ldots \quad (2.16)$$

que nos permiten transformar el sistema tridiagonal (2.14) en un sistema triangular superior

$$\begin{bmatrix} a_2 & h_2 & 0 & \cdots & 0 & 0 \\ 0 & a_3 & h_3 & \cdots & 0 & 0 \\ 0 & 0 & a_4 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \ddots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & a_{n-2} & h_{n-2} \\ 0 & 0 & 0 & \cdots & 0 & a_{n-1} \end{bmatrix} \bullet \begin{bmatrix} s''_2 \\ s''_3 \\ s''_4 \\ \vdots \\ s''_{n-2} \\ s''_{n-1} \end{bmatrix} = \begin{bmatrix} b_2 \\ b_3 \\ b_4 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix}. \qquad \ldots\ldots\ldots \quad (2.17)$$

El sistema (2.17) se puede resolver en forma recurrente hacia atrás, mediante las fórmulas

$$s''_{n-1} = \frac{b_{n-1}}{a_{n-1}} \quad \text{y} \quad s''_k = \frac{b_k - h_k\, s''_{k+1}}{a_k}; \quad k = n-2, n-3, \ldots, 2. \qquad \ldots\ldots\ldots \quad (2.18)$$

Observación: La fórmula recurrente (2.15) únicamente depende de los pasos h_k, $k = 1, 2, \ldots, n-1$, y es independiente de los valores f_k de la función. Así que esta parte del cálculo se puede realizar independientemente de la función particular que se esté aproximando.

Es claro que una vez calculado los valores de $s''_1, s''_2, \ldots, s''_{n-1}, s''_n$, la fórmula (2.8) nos ayuda a calcular la segunda derivada $s''(x)$ en cualquier punto x. La pendiente de la recta dada en (2.8) es la tercera derivada del spline en cada

intervalo. La fórmula (2.11) nos ayuda a calcular las derivadas s'_i en los nodos, la fórmula (2.9) calcula la primera derivada $s'(x)$ en cualquier punto x, y la fórmula (2.10) calcula el spline cúbico $s(x)$ en cualquier punto x. De hecho podemos graficar las tres funciones $s(x)$, $s'(x)$ y $s''(x)$ simultáneamente en un mismo sistema de coordenadas.

Ejemplo 2.7: De nuestro familiar conjunto de datos

x	−3.0	−1.0	1.0	2.0	2.5	3.0
f	1.0	1.5	2.0	2.0	1.5	1.0

a) Calcular el valor de $f(0.3)$ mediante un spline cúbico.
b) Aproximar la primera, segunda y tercera derivada de f en $x^* = 0.3$.
c) Obtener explícitamente el spline que suaviza el conjunto de datos.

Solución: Primero calcularemos los valores de $s''_1, s''_2, \ldots, s''_{n-1}, s''_n$, manualmente paso a paso.

Paso 1: $h_i = x_{i-1} - x_i, \quad k = 1, 2, 3, 4, 5. \quad \therefore \quad h = (2.0, 2.0, 1.0, 0.5, 0.5).$

Paso 2: $d_i = 6\left(\dfrac{f_{i+1} - f_i}{h_i} - \dfrac{f_i - f_{i-1}}{h_{i-1}}\right), \quad k = 2, 3, 4, 5.$

$$d_2 = 6\left(\frac{2.0-1.5}{2.0} - \frac{1.5-1.0}{2.0}\right) = 0, \qquad d_3 = 6\left(\frac{2.0-2.0}{1.0} - \frac{2.0-1.5}{2.0}\right) = -1.5,$$

$$d_4 = 6\left(\frac{1.5-2.0}{0.5} - \frac{2.0-2.0}{1.0}\right) = -6.0, \qquad d_5 = 6\left(\frac{1.0-1.5}{0.5} - \frac{1.5-2.0}{0.5}\right) = 0.$$

Por lo tanto, $d = (0, -1.5, -6.0, 0)$.

Paso 3: El sistema tridiagonal y de diagonal dominante por filas (2.14), de ecuaciones lineales que resulta en este caso es

$$\begin{aligned}
8s''_2 + 2s''_3 &= 0 \\
2s''_2 + 6s''_3 + s''_4 &= -1.5 \\
s''_3 + 3s''_4 + 0.5s''_5 &= -6.0 \\
0.5s''_4 + 2s''_5 &= 0
\end{aligned}$$

Para resolver este sistema, lo transformamos en un sistema triangular superior, trabajando con la matriz aumentada del sistema.

$$\begin{bmatrix} 8 & 2 & 0 & 0 & \vdots & 0 \\ 2 & 6 & 1 & 0 & \vdots & -1.5 \\ 0 & 1 & 3 & 0.5 & \vdots & -6.0 \\ 0 & 0 & 0.5 & 2 & \vdots & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 8 & 2 & 0 & 0 & \vdots & 0 \\ 0 & -22 & -4 & 0 & \vdots & 6 \\ 0 & 1 & 3 & 0.5 & \vdots & -6.0 \\ 0 & 0 & 0.5 & 2 & \vdots & 0 \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} 8 & 2 & 0 & 0 & \vdots & 0 \\ 0 & 22 & 4 & 0 & \vdots & -6 \\ 0 & 0 & 62 & 11 & \vdots & -126 \\ 0 & 0 & 0.5 & 2 & \vdots & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 8 & 2 & 0 & 0 & \vdots & 0 \\ 0 & 22 & 4 & 0 & \vdots & -6 \\ 0 & 0 & 62 & 11 & \vdots & -126 \\ 0 & 0 & 0 & 237 & \vdots & 126 \end{bmatrix}.$$

Calculamos ahora los valores de s''_2, s''_3, s''_4 y s''_5 con la sustitución hacia atrás dado en (2.18):

$$s''_5 = \frac{126}{237} = 5.3166 \times 10^{-1} \qquad s''_4 = \frac{-126 - 11 s''_5}{62} = -2.1266$$

$$s''_3 = \frac{-6 - 4 s''_4}{22} = 1.1393 \times 10^{-1} \qquad s''_2 = -\frac{2 s''_3}{8} = -2.8482 \times 10^{-2}$$

$$s''_1 = 0 \qquad s''_6 = 0$$

Estamos listos para responder a cada uno de los incisos del Ejemplo 2.7.

a) Para calcular el valor de $f(0.3)$, notamos que $x^* \in [-1.0, 1.0]$ que es el segundo intervalo determinado por los datos. Le corresponde $k = 2$, por lo tanto, de (2.10) y (2.11) tenemos:

$$f(0.3) = f_2 + s'_2 (0.3 - x_2) + \frac{1}{2} s''_2 (0.3 - x_2)^2 + \frac{1}{6} \frac{s''_3 - s''_2}{h_2} (0.3 - x_2)^3,$$

$$s'_2 = \frac{2.0 - 1.5}{2.0} + \frac{2.8482 \times 10^{-2}}{3} 2.0 - \frac{1.1393 \times 10^{-1}}{6} 2.0 = 2.3101 \times 10^{-1},$$

Sustituyendo valores tenemos

$$f(0.3) = 1.5 + 2.3101 \times 10^{-1} (1.3) - 1.4241 \times 10^{-2} (1.3)^2 + 1.1868 \times 10^{-2} (1.3)^3$$

$$= 1.8023,$$

de donde $f(0.3) = 1.8023$.

b) Aproximamos ahora $f'(0.3)$, $f''(0.3)$ y $f'''(0.3)$.

$$f'(0.3) = s'_2 + s''_2(0.3 - x_2) + \frac{1}{2}\frac{s''_3 - s''_2}{h_2}(0.3 - x_2)^2 = 0.25415,$$

por lo tanto $f'(0.3) = 0.25415$.

$$f''(0.3) = s''_2 + \frac{s''_3 - s''_2}{h_2}(0.3 - x_2) = 0.064086, \quad \text{por lo tanto } f''(0.3) = 0.064086.$$

$$f'''(0.3) = \frac{s''_3 - s''_2}{h_2} = 0.071206, \text{ por lo tanto } f'''(0.3) = 0.071206.$$

c) Para obtener explícitamente el spline $s(x)$, calculamos la primera derivada en los demás nodos. En el nodo $k = 2$ ya fue calculado. Para ello recordemos de (2.11) que

$$s'_k = \frac{f_{k+1} - f_k}{h_k} - \frac{s''_k}{3}h_k - \frac{s''_{k+1}}{6}h_k, \quad 1 \le k \le 5.$$

$$s'_1 = \frac{1.5 - 1.0}{2.0} - \frac{0}{3}2.0 + \frac{2.8482 \times 10^{-2}}{6}2.0 = 2.5949 \times 10^{-1},$$

$$s'_3 = \frac{2.0 - 2.0}{1.0} - \frac{1.1393 \times 10^{-1}}{3}1.0 + \frac{2.1266}{6}1.0 = 3.1646 \times 10^{-1},$$

$$s'_4 = \frac{1.5 - 2.0}{0.5} + \frac{2.1266}{3}0.5 - \frac{5.3165 \times 10^{-1}}{6}0.5 = -6.8987 \times 10^{-1},$$

$$s'_5 = \frac{1.0 - 1.5}{0.5} - \frac{5.3165 \times 10^{-1}}{3}0.5 - \frac{0}{6}0.5 = -1.0886.$$

Como: $s_k(x) = f_k + s'_k(x - x_k) + \frac{s''_k}{2}(x - x_k)^2 + \frac{s''_{k+1} - s''_k}{6h_k}(x - x_k)^3, \quad 1 \le k \le 5,$

tenemos que:

$$s_1(x) = 1.0 + 2.5949 \times 10^{-1}(x + 3.0) - 2.3735 \times 10^{-3}(x + 3.0)^3,$$

$$s_2(x) = 1.5 + 2.3101 \times 10^{-1}(x + 1.0) - 1.4241 \times 10^{-2}(x + 1.0)^2 + 1.1868 \times 10^{-2}(x + 1.0)^3,$$

$$s_3(x) = 2.0 + 3.1646 \times 10^{-1}(x - 1.0) + 5.6965 \times 10^{-2}(x - 1.0)^2 - 3.7342 \times 10^{-1}(x - 1.0)^3,$$

$$s_4(x) = 2.0 - 6.897 \times 10^{-1}(x - 2.0) - 1.0633(x - 2.0)^2 + 8.8608 \times 10^{-1}(x - 2.0)^3,$$

$$s_5(x) = 1.5 - 1.0886(x - 2.5) + 2.6582 \times 10^{-1}(x - 2.5)^2 - 1.7722 \times 10^{-1}(x - 2.5)^3.$$

Por lo tanto, el spline cúbico natural que suaviza nuestro conjunto de datos es:

$$s(x) = \begin{cases} s_1(x) & si \quad -3.0 \le x \le -1.0 \\ s_2(x) & si \quad -1.0 \le x \le 1.0 \\ s_3(x) & si \quad 1.0 \le x \le 2.0 \\ s_4(x) & si \quad 2.0 \le x \le 2.5 \\ s_5(x) & si \quad 2.5 \le x \le 3.0 \end{cases}$$

Con la ayuda de Matlab podemos dibujar la gráfica del spline $s(x)$, aunque es un poco más complicado. En la siguiente sección, daremos un programa en Matlab que dibuje tanto la gráfica del spline, como la de su primera y segunda derivada. ■

En la Tabla 2.5 se resumen los valores obtenidos para $f(0.3)$ del conjunto de datos del Ejemplo 2.7. Presentamos los valores obtenidos por los polinomios de interpolación de distintos grados y del spline cúbico natural.

Polinomios de Interpolación					**Spline Cúbico Natural**
Grados del Polinomio					
1	2	3	4	5	
1.825	1.901	1.643	1.218	1.451	1.8023

Tabla 2.5. Comparación de resultados entre polinomios de interpolación y spline cúbico natural.

El valor obtenido por el spline es casi igual al valor obtenido por el polinomio de interpolación de primer grado. Se manifiesta la suavidad del spline, pues los polinomios de grado ≥ 2 tienen oscilaciones grandes como se mostró en la Figura 2.6.

2.3.4 Algoritmo Para el Cálculo y Evaluación del Spline Cúbico

Presentaremos un algoritmo que calcula paso a paso las cantidades que se necesitan para calcular los valores de $s''_1, s''_2, \ldots, s''_{n-1}, s''_n$, así como la cantidad de operaciones aritméticas de punto flotante que se requieren, adiciones (A), multiplicaciones (M) y divisiones (D). Después presentamos un programa en Matlab basado en este algoritmo, otro que evalúa el spline $s(x)$ y su primera y se-

gunda derivada $s'(x)$ y $s''(x)$ en puntos del intervalo $[x_1, x_n]$ para su visualización gráfico, y un programa que evalúa $s(x)$, $s'(x)$ y $s''(x)$ en puntos de interés $w_1, w_2, \ldots, w_m$. Finalmente un programa que integra los tres.

Algoritmo 4

Entrada de datos: $x = [x_1, x_2, \ldots, x_n]$ y $f = [f_1, f_2, \ldots, f_n]$.

Salida: $s''_1, s''_2, \ldots, s''_{n-1}, s''_n$.

Cálculo de los pasos h_k, $k = 1, 2, \ldots, n-1$.

1. Para $k = 1, 2, \ldots, n-1$.

   1.1. $h_k = x_{k+1} - x_k$ ..... $(n-1)A$.

Cálculo de los coeficientes $a_2, a_3, \ldots, a_{n-1}$ **del sistema triangular superior (2.17).**

2. $a_2 = 2(h_1 + h_2)$ ..... $1A + 1M$.

3. Para $k = 3, 4, \ldots, n-1$.

   3.1. $a_k = 2(h_{k-1} + h_k) - \dfrac{h_{k-1}^2}{a_{k-1}}$ ..... $2(n-3)A + 2(n-3)M + (n-3)D$.

Cálculo de las componentes $d_2, d_3, \ldots, d_{n-1}$ **del vector** d **del sistema tridiagonal (2.14).**

4. Para $k = 2, 3, \ldots, n$.

   4.1. $c_k = \dfrac{f_k - f_{k-1}}{h_{k-1}}$ ..... $(n-1)A + (n-1)D$.

5. Para $k = 2, 3, \ldots, n-1$.

   5.1. $d_k = 6(c_{k+1} - c_k)$ ..... $(n-2)A + (n-2)M$.

Cálculo de las componentes $b_2, b_3, \ldots, b_{n-1}$ **del vector** b **del sistema triangular superior (2.17), de acuerdo con la fórmula (2.16).**

6. $b_2 = d_2$
7. Para $k = 3, 4, \ldots, n-1$.

   7.1. $b_k = d_k - \dfrac{h_{k-1} b_{k-1}}{a_{k-1}}$ ..... $(n-3)A + (n-3)M + (n-3)D$.

Cálculo de las incógnitas (salida de resultados): $s''_2, s''_3, \ldots, s''_{n-1}$ **del sistema tridiagonal (2.14).**

8. $s''_{n-1} = \dfrac{b_{n-1}}{a_{n-1}}$ ..... $1D$.

9. Para $k = n-2, n-3, \ldots, 2$.

   9.1. $s''_k = \dfrac{b_k - h_k s''_{k-1}}{a_k}$ ..... $(n-3)A + (n-3)M + (n-3)D$.

10. $s''_1 = 0, \quad s''_n = 0$.

Alto.

A continuación presentamos un programa en Matlab basado en el Algoritmo 4.

```
%  Este programa se llama biprima.m y determina las segundas derivadas
%  s''(1), s''(2),...,s''(n) del spline cúbico natural que interpola un
%  conjunto de n datos en el plano.
%
   format long e
   n=length(x);
%
%  Cálculo de los espaciamientos h(k) con k=1,2,...,n-1.
%
   for k=1:n-1;
       h(k)=x(k+1)-x(k); end
%
%  Cálculo de los coeficientes de la matriz triangular superior.
%
   a(2)=2*(h(1)+h(2));
   for k=3:n-1;
       a(k)=2*(h(k-1)+h(k))-(h(k-1)^2)/a(k-1); end
%
%  Cálculo del vector no homogéneo de la matriz tridiagonal.
%
   for k=2:n;
       c(k)=(f(k)-f(k-1))/h(k-1); end
%
   for k=2:n-1;
       d(k)=6*(c(k+1)-c(k)); end
%
%  Cálculo del vector no homogéneo de la matriz triangular superior.
%
   b(2)=d(2);
   for k=3:n-1;
       b(k)=d(k)-h(k-1)*b(k-1)/a(k-1); end
%
%  Solución para s''(1), s''(2),..., s''(n-1), s''(n).
%
   sbn(1)=0; sbn(n)=0;
```

```
sbn(n-1)=b(n-1)/a(n-1);
for k=n-2:-1:2
    sbn(k)=(b(k)-h(k)*sbn(k+1))/a(k); end
```

Presentamos ahora un programa en Matlab que evalúa el spline $s(x)$ y sus derivadas $s'(x)$ y $s''(x)$ en puntos del segmento $[x_1, x_n]$ para su visualización gráfico.

```
%  Este programa se llama sgrafica.m y presenta la visualización
%  gráfico de un spline cúbico natural s(x), de su primera derivada
%  sp(x) y de su segunda derivada sb(x) en el intervalo [x(1), x(n)].
%
   clc,
%
   plot(x,f,'*'); grid; % Aquí se visualizan los datos con '*'.
   hold on;             % Aquí se abre la ventana gráfica.
%
%  Evaluación del spline y sus derivadas en [x(k), x(k+1)].
%
   for k=1:n-1;
   p=h(k)/30; % Paso p para discretizar [x(k), x(k+1)].
   xx=[x(k):p:x(k+1)]; % xx es el vector [x(k),x(k)+p,...,x(k+1)].
%
%  Cálculo de la primera derivada s' del spline s en el nodo
%  x(k), de
%  acuerdo con la fórmula (2.11). Aquí s'(x(k))=spn(k) y es un
%  escalar.
%
   spn(k)=c(k+1)-sbn(k)*h(k)/3-sbn(k+1)*h(k)/6;
%
%  Cálculo de la tercera derivada s''' del spline s sobre el
%  intervalo
%  [x(k), x(k+1)]. Aquí s'''=stn(k) y es constante sobre el
%  intervalo.
%
   stn(k)=(sbn(k+1)-sbn(k))/h(k);
%
   bin1=xx-x(k);     % Aquí bin1 es el vector [0,p,...,x(k+1)-x(k)].
   bin2=bin1.*bin1; % Producto componente por componente de bin1.
%
%  Cálculo de la primera derivada s' en todas las componentes
%  del vector xx. Aquí s'=sp y es un vector del mismo tamaño de xx.
%  El cálculo se realiza de acuerdo con la fórmula (2.9).
%
   sp=spn(k)+sbn(k)*bin1+stn(k)*bin2/2;
%
%  Evaluación del spline cúbico s en las componentes del vector xx,
%  de acuerdo con la fórmula (2.10). s es del mismo tamaño de xx.
%
   s=f(k)+spn(k)*bin1+sbn(k)*bin2/2+stn(k)*(bin1.*bin2)/6;
%
%  Como la segunda derivada s''(x) es una linea recta sobre el
%  intervalo [x(k), x(k+1)], son suficientes dos puntos para
%  graficarlo.
```

```
    %
       xb=[x(k),x(k+1)]; sb=[sbn(k),sbn(k+1)];
    %
    %  Visualización gráfica del spline y sus derivadas en el intervalo
    %  [x(k), x(k+1)]. Se dibujan las gráficas en diferentes colores
    %  para su mejor identificación.
    %
       plot(xx,s,'r',xx,sp,'b',xb,sb,'g')
   end
   hold off
%
%  Aqui se cierra la ventana gráfica y termina la visualización.
```

El siguiente programa en Matlab evalúa el spline s(x) y sus derivadas $s'(x)$ y $s''(x)$ en puntos de interés $w_1, w_2, ..., w_m$.

```
%  Este programa se llama sevalua.m y evalúa el spline s(x) y sus
%  derivadas s'(x) y s''(x) en puntos de interés.
%
  clc; clear bin1 bin2 s sp sb;
  w=input('Dame el vector de puntos de interés como [w1,w2,...,wm]: ');
  m=length(w);
%
%  Localización de la componente w(j) en el intervalo [x(k),x(k+1)].
%
   for j=1:m;
       if w(j)<x(1)
       bin1=w(j)-x(1);
       bin2=bin1^2;
       sb(j)=sbn(1)+stn(1)*bin1;
       sp(j)=spn(1)+sbn(1)*bin1+stn(1)*bin2/2;
       s(j)=f(1)+spn(1)*bin1+stn(1)*bin1*bin2/6;
%
   elseif w(j)>x(n);
       bin1=w(j)-x(n-1);
       bin2=bin1^2;
       sb(j)=sbn(n-1)+stn(n-1)*bin1;
       sp(j)=spn(n-1)+sbn(n-1)*bin1+stn(n-1)*bin2/2;
       s(j)=f(n-1)+spn(n-1)*bin1+sbn(n-1)*bin2/2+stn(n-1)*bin1*bin2/6;
%
   else for k=1:n-1;
       if w(j)>=x(k) & w(j)<=x(k+1);
          bin1=w(j)-x(k);
          bin2=bin1^2;
          sb(j)=sbn(k)+stn(k)*bin1;
          sp(j)=spn(k)+sbn(k)*bin1+stn(k)*bin2/2;
          s(j)=f(k)+spn(k)*bin1+sbn(k)*bin2/2+stn(k)*bin1*bin2/6;
       end; end; end; end
```

Finalmente integramos los tres programas anteriores en un programa principal.

```
%  Este programa se llama scubico.m y presenta el cálculo de un spline
%  cúbico natural y sus derivas, visualización gráfico y la evaluación
%  en puntos de interés.
%
   clear; clc
%
%  Entrada de datos:
%
 x=input('Dame el vector de nodos como [x1,x2,...,xn]: ');
 f=input('Dame los valores del vector de nodos como [f1,f2,...,fn]: ');
%
%  Cálculo de s''(1), s''(2),...,s''(n) en los nodos x(1), x(2),...,
%  x(n).
%
   biprima
%
%  Visualización gráfica del spline s(x) y sus derivadas s'(x) y s''(x)
%  sobre el intervalo [x(1), x(n)].
%
   sgrafica
%
%  Evaluación del spline s(x) y sus derivadas s'(x) y s''(x) en puntos
%  de interés.
%
   sevalua
%
%  Salida de resultados.
%
   Resultados=[w',s',sp',sb']
```

Ejemplo 2.8: Usamos nuestro programa **scubico.m** para graficar el spline $s(x)$ que obtuvimos en el Ejemplo 2.7. Graficamos también sus derivadas $s'(x)$ y $s''(x)$. Al ejecutar el programa scubico.m nos solicitará los datos y automáticamente generará las gráficas pedidas. El resultado se presenta en la Figura 2.9, los letreros se añadieron posteriormente. Se puede volver a evaluar el spline y sus derivadas en $x^* = 0.3$.

2.3.5 Eficiencia y Error de Aproximación

La eficiencia del algoritmo para $n \geq 3$ es la cantidad de trabajo (es decir, operaciones aritméticas: adición, multiplicación y división) que se necesitan realizar para calcular las incógnitas s''_k, con $k = 2, 3, ..., n-1$. Esta se obtiene al sumar todas las operaciones aritméticas que se realizan en el Algoritmo 4, y está dada por:

$$(7n-15)A+(5n-13)M+(4n-9)D.$$

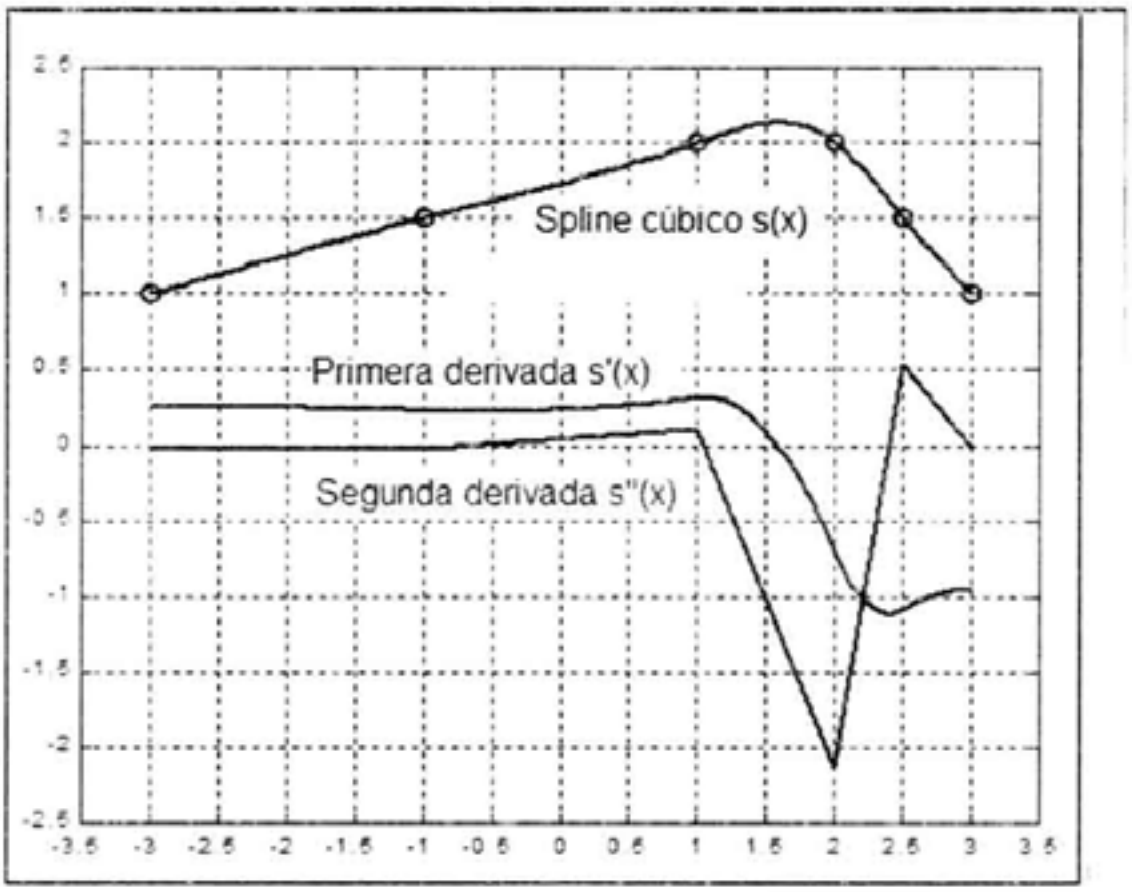


Figura 2.9. Gráfica del spline cúbico que suaviza el conjunto de datos del Ejemplo 2.7.

Para calcular $s(x^*)$, $s'(x^*)$ y $s''(x^*)$ una vez localizado x^* en el segmento $[x_i, x_{i+1}]$ se necesitan

$$8A+8M+4D.$$

Como la medida de la eficiencia del algoritmo es lineal en n, por lo tanto,

a) n pequeño implica substancialmente poca cantidad de trabajo para calcular $s(x)$, y
b) pequeñas variaciones de n implica pequeñas variaciones en la cantidad de trabajo.

El error de aproximación está dada por el siguiente teorema, cuya prueba está en las referencias.

Teorema 2.4: Si f tiene derivadas continuas hasta de orden 2 sobre $[a,b]$ y $M = \max_{a \le x \le b} |f''(x)|$, entonces, para cualquier $x^* \in [a,b]$:

$$|f(x^*) - s(x^*)| \le \frac{7}{8} h^2 M,$$

si los nodos son uniformemente espaciados por h. Si los nodos no son uniformemente espaciados, pero $2h_{min} > h_{máx}$, con $0 < h_{mín} \le h_k \le h_{máx}$, $k = 1, 2, \ldots, n-1$, entonces

$$|f(x^*) - s(x^*)| \le \frac{1}{8} h_{máx}^2 \left(\frac{2h_{min}}{h_{máx}} - 1 \right)^{-1} M.$$

En general, el error de aproximación puede ser tan pequeño como se quiera al aumentar el número de nodos, y disminuir $h_{máx}$, pero cuidando que $\frac{h_{máx}}{h_{min}} \le \alpha < 2$. Estas estimaciones muestran la exactitud de la aproximación por spline.

Ejemplo 2.9: En la referencia [2] aparece en la pág. 59 el conjunto de datos dados en la Tabla 2.6. El objetivo es obtener una gráfica suave que interpole los datos (0, 2024), (1, 2031), (3, 2320), (5, 2063), (7, 1860), (9, 1937) y (12, 2006) que se obtienen de dicha tabla, mediante un spline cúbico.

MUESTRA	I* a (c/s)
ZEOLON: 900-H	2024
Mo-H1	2031
Mo-H3	2320
Mo-H5	2063
Mo-H7	1860
Mo-H9	1937
Mo-H12	2006

Tabla 2.6. Conjunto de datos de la referencia [2].

Solución: Podemos aplicar el programa scubico.m a este conjunto de datos para obtener un spline cúbico natural que los suaviza. Como únicamente nos interesa el spline cúbico, hacemos una modificación a sgrafica.m, en lugar de:

```
plot(xx,s,'r',xx,sp,'b',xb,sb,'g')
```

escribimos `plot(xx,s)`.

Después ejecutamos el programa scubico.m en Matlab con los siguientes datos

```
x=[0,1,3,5,7,9,12];
f=[2024,2031,2320,2063,1860,1937,2006];
```

y enseguida obtenemos la gráfica del spline cúbico que suaviza los datos, como se muestra en la Figura 2.10. Podemos obtener explícitamente el spline cúbico, si seleccionamos los resultados que se generan en el programa de la siguiente manera:

```
d=f(1:n-1)';  c=spn(1:n-1)';  b=(1/2)*sbn(1:n-1);
 a=(1/6)*stn(1:n-1);  F=[d,c,b,a]
```

La matriz F contiene los coeficientes del spline cúbico, en este caso es la matriz

```
2024     -31.410       0          38.410
2031      83.820       115.23    -42.445
2320      35.398      -139.44     28.747
2063    -177.41        33.037     2.4593
1860     -15.751       47.793    -10.334
1937      51.417      -14.209     1.5787
```

En cada fila están los coeficientes de los polinomios cúbicos, así que el spline cúbico que suaviza los datos es:

$$s(x)=\begin{cases}2024-31.410x+38.410x^3, & \text{si } 0\le x\le 1\\ 2031+83.820(x-1)+115.23(x-1)^2-42.445(x-1)^3, & \text{si } 1\le x\le 3\\ 2320+35.398(x-3)-139.44(x-3)^2+28.747(x-3)^3, & \text{si } 3\le x\le 5\\ 2063-177.41(x-5)+33.037(x-5)^2+2.4593(x-5)^3, & \text{si } 5\le x\le 7\\ 1860-15.751(x-7)+47.793(x-7)^2-10.334(x-7)^3, & \text{si } 7\le x\le 9\\ 1937+51.417(x-9)-14.209(x-9)^2+1.5787(x-9)^3, & \text{si } 9\le x\le 12\end{cases}$$

y su gráfica es la que tenemos en la Figura 2.10.

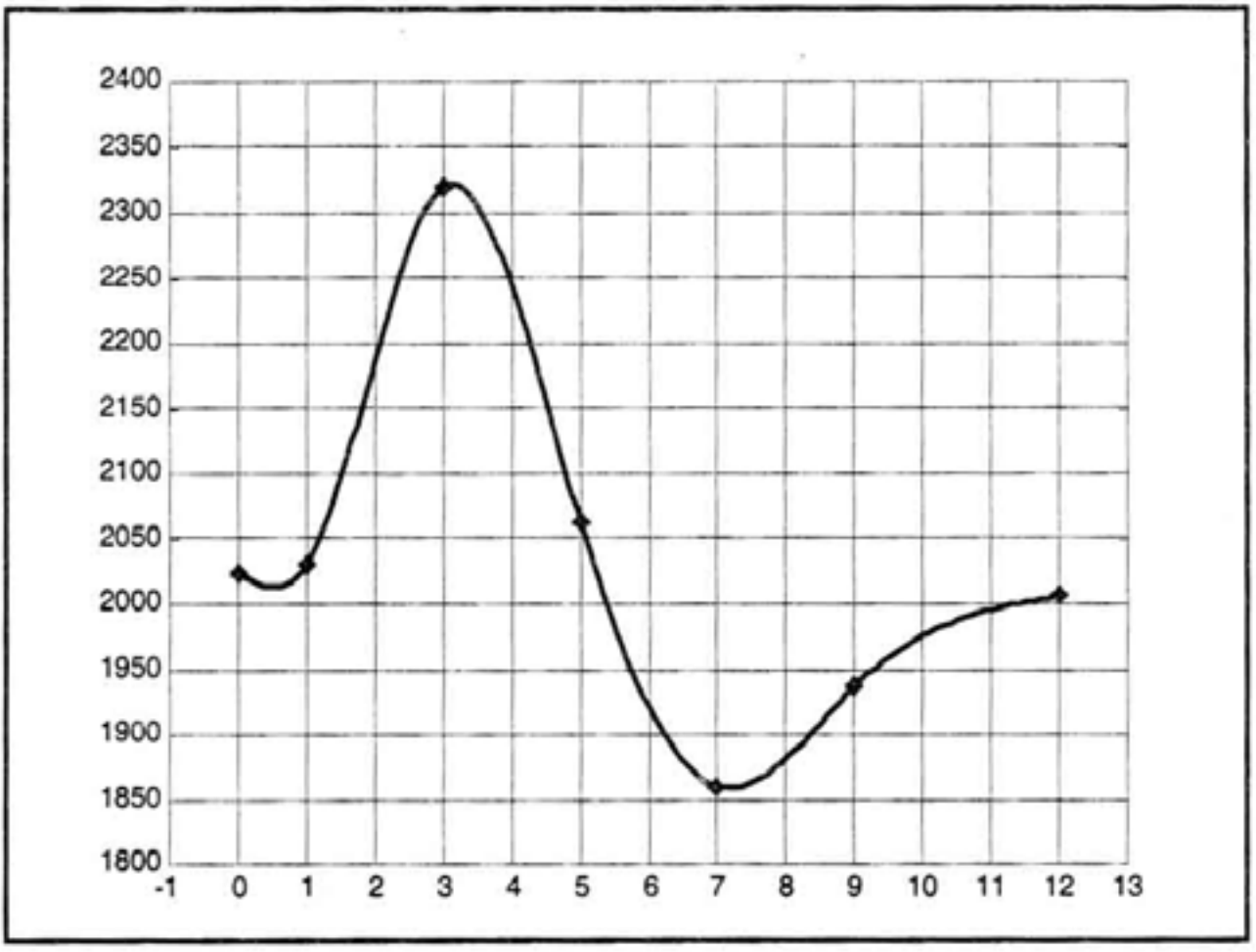


Figura 2.10. Gráfica del spline cúbico natural calculado.

Conclusiones: En el artículo de la referencia antes citado el autor no menciona cómo obtiene su gráfica. Con el algoritmo que se discutió fue posible calcular explícitamente los polinomios cúbicos, los que trazan con una "buena" exactitud dicha gráfica, como se muestra en la gráfica de la Figura 2.10.

Otra ventaja del algoritmo es que si nosotros tuviéramos la necesidad de calcular el número de cuentas por segundo para Mo-H2, por ejemplo, sólo hay que evaluar el número 2 en el polinomio correspondiente o evaluarla en puntos de interés de acuerdo con el programa sevalua.m, en el primer caso se tenemos:

$$s(2) = 2031 + 83.820(2-1) + 115.23(2-1)^2 - 42.445(2-1)^3,$$
$$s(2) = 2031 + 83.820 + 115.23 - 42.445,$$

de donde, $s(2) = 2187.605$, que se redondea a 2188 cuentas por segundo para Mo-H2. ■

2.4 PROBLEMAS

Instrucciones: Lea con toda atención cada uno de los problemas antes de empezar a resolverlos.

1. Simplifique el polinomio de Lagrange que interpola cada uno de los siguientes conjuntos de puntos. Grafique los puntos y el polinomio.

   a) (–2.3, 2.1), (0.5, –1.3) y (3.1, 4.2). b) (1, 2), (3, 4), (5, 3) y (9, 8).

2. Evalúe el polinomio de Lagrange que interpola los datos (–1.0, 1.5), (1.0, 2.0) y (2.0, 2.0) en $x^* = 0.3$, con una aritmética flotante $\beta=10$, $t=7$ y redondeo. Vea el Ejemplo 2.2(b).

3. Programe el Algoritmo 1 en Matlab o en lenguaje de su preferencia, y úselo para aproximar las siguientes funciones en los puntos que se especifican.

   a) $f(8.4)$ si $f(8) = 16.63553$, $f(8.1) = 17.61549$, $f(8.3) = 17.56492$ y $f(8.6) = 18.50515$.

   b) $f(\pi)$ si $f(3.0) = -4.24$, $f(3.1) = -3.50$ y $f(3.2) = -2.50$.

4. Evalúe el polinomio de Lagrange que interpola los datos (– 3.0, 1.0), (–1.0, 1.5), (1.0, 2.0), (2.0, 2.0), (2.5, 1.5) y (3.0, 1.0) en $x^* = 0.3$. Puede hacerlo manualmente o usar su programa basado en el Algoritmo 1. Compare su resultado con los tres resultados obtenidos en el Ejemplo 2.2 y escriba sus conclusiones.

5. Use el polinomio de Newton para aproximar $f(0.5)$ mediante los siguientes datos:

x	0.0	0.2	0.4	0.6	0.8
$f(x)$	1.00000	1.22140	1.49182	1.82212	2.22554

6. Use de nuevo el polinomio de Newton para aproximar $f(0.3)$ usando polinomios de grados 4 y 5, que interpolan los datos:

x	–3.0	–1.0	1.0	2.0	2.5	3.0
$f(x)$	1.0	1.5	2.0	2.0	1.5	1.0

   Compare los resultados con los que se muestran en la Tabla 2.2 del Ejemplo 2.4.

7. Use el programa Polinewton.m para aproximar las siguientes funciones en los puntos especificados. Debe obtener los mismos resultados del Problema 3 ¿Por qué?.

   a) $f(8.4)$ si $f(8) = 16.63553$, $f(8.1) = 17.61549$, $f(8.3) = 17.56492$ y $f(8.6) = 18.50515$.

   b) $f(\pi)$ si $f(3.0) = -4.24$, $f(3.1) = -3.50$ y $f(3.2) = -2.50$.

8. Si los datos se obtienen de una función f, modifique el programa Polinewton.m para que grafique la función f y el polinomio de interpolación.

9. Con el objetivo de comparar una función y su polinomio de interpolación, aplique su programa modificado del problema anterior para que grafique los datos, el polinomio de interpolación y la función donde se obtienen dichos datos en un mismo sistema de coordenadas, para cada uno de los siguientes casos:

   a) $f(x) = e^{2x}\cos 3x$, con $x_1 = 0$, $x_2 = 0.3$ y $x_3 = 0.6$.

   b) $f(x) = \text{sen}(\ln x)$, con $x_1 = 2.0$, $x_2 = 2.4$ y $x_3 = 2.6$.

   c) $f(x) = \ln x$, con $x_1 = 1.0$, $x_2 = 1.1$, $x_3 = 1.3$ y $x_4 = 1.4$.

   d) $f(x) = \cos x + \text{sen}\, x$, con $x_1 = 0$, $x_2 = 0.25$, $x_3 = 0.5$ y $x_3 = 1.0$.

10. Calcule una cota para la función del error de aproximación $E_1(x)$ que se comete al tomar el polinomio de interpolación $P(x)$, como una aproximación de la función $f(x)$ sobre el intervalo $[x_1, x_{n+1}]$, para cada una de las funciones del problema anterior. Sugerencia: este problema es fácil si usa los resultados del problema anterior.

11. Dibuje las gráficas de los polinomios de interpolación $P_n(x)$ de grados $\le n$ de la función de Runge, juntos con la función de Runge, con los datos obtenidos por los puntos equidistantes:

$$x_i = -1 + (i-1)\frac{2}{n}, \quad i = 1, 2, \ldots, n+1,$$

    para $n = 4$, 8 y 12. Las 4 gráficas se deben dibujar en un mismo sistema de coordenadas. Escriba sus conclusiones.

12. Dibuje la gráfica del polinomio que interpola a $f(x) = \cos x$ de acuerdo con los datos del Ejemplo 2.6 sobre el intervalo [0, 2], junto con la gráfica de $f(x)$. Compare las gráficas y escriba sus conclusiones.

13. La siguiente tabla se obtiene de la función de error normal $y(x) = e^{x^2/2} / \sqrt{2\pi}$.

x	1.00	1.20	1.40	1.60	1.80	2.00
y	0.2420	0.1942	0.1497	0.1109	0.0790	0.0540

Calcule el valor del polinomio de interpolación en x^*=1.50. Estime el error de aproximación en dos formas diferentes como en los Ejemplos 2.5 y 2.6, e interprete los resultados. Dibuje la gráfica del polinomio de interpolación y la de la función de error normal en un mismo sistema de coordenadas.

14. La población del estado de Tabasco entre 1900 y 1995 según datos de la INEGI, se muestra en la siguiente tabla:

Año	1940	1950	1960	1970	1980	1990	1995
Población	285 630	362 716	496 340	768 327	1 062 961	1 501 744	1 748 664

a) Investigue en la INEGI la población del estado de Tabasco en los años 1955, 1975, 1985 y 2000.

b) Use polinomios de interpolación de grados 1, 2, 3, 4, 5 y 6 para estimar la población en el año 1975. ¿Cuál de ellos se parece más al dato de la INEGI?. Vea el Ejemplo 1.1 del Capítulo I. Escoja los puntos base adecuadamente para obtener mayor exactitud en cada polinomio.

c) Use polinomios de interpolación de grados 1, 2 y 3 para predecir la población del año 2000. ¿Cuál de ellos se parece más al dato de la INEGI?.

15. Se sospecha que grandes cantidades de tanino en las hojas de robles maduros, inhibe el crecimiento de la polilla de invierno (*Operophteca bromata L., Geometridae*) larva que daña extremadamente estos árboles en ciertos años. La siguiente tabla muestra el peso promedio de dos muestras de larvas en los primeros 28 días de haber nacido. La primera muestra fue tomada de las hojas jóvenes de un roble, mientras que la segunda muestra fue tomada de las hojas maduras del mismo árbol.

Día	0	6	10	13	17	20	28
Muestra 1, peso promedio (mg)	6.67	17.33	42.67	37.33	30.10	29.31	28.74
Muestra 2, peso promedio (mg)	6.67	16.11	18.89	15.00	10.56	9.44	8.89

a) Use interpolación polinomial para aproximar la curva del peso promedio para cada muestra.

b) Encuentre una aproximación al valor máximo del peso promedio de cada muestra determinando el máximo del polinomio de interpolación.

16. El volumen específico v de un vapor sobrecalentado es enlistado en tablas de vapor para diferentes temperaturas. Por ejemplo, a una presión de 2 950 $lb/pulg^2$ absolutas:

T, °F	700	720	740	760	780
v	0.1058	0.1280	0.1462	0.1603	0.1703

Determine v para $T = 750$ °F.

17. Usted mide la caída de voltaje V a través de una resistencia para un número de diferentes valores de corriente i. Los resultados son

i	0.25	0.75	1.25	1.5	2.0
V	−0.45	−0.60	0.70	1.88	6.0

Use interpolación polinomial para estimar la caída de voltaje para $i = 1.1$. Interprete sus resultados.

18. La ley de Ohm establece que la caída de voltaje V a través de un resistor ideal es linealmente proporcional a la corriente i que fluye a través del resistor como $V=iR$, donde R es la resistencia. Sin embargo, resistencias reales podrían no siempre cumplir la ley de Ohm. Suponga que usted realiza varios experimentos muy precisos para medir la caída de voltaje y la corriente correspondiente para cada resistencia. Los resultados, como se enlistan en la tabla, sugieren una relación curvilínea más que una línea recta representada por la ley de Ohm.

i	−1.00	−0.50	−0.25	0.25	0.50	1.00
V	−193	−41	−13.5625	13.5625	41	193

Para cuantificar esta relación, se debe ajustar una curva a los datos. Debido al error de medición, el método preferido para el ajuste de la curva podría ser el de regresión para el análisis de tales datos experimentales. Sin embargo, la suavidad de esta relación, así como la precisión de los métodos experimentales, sugieren que la interpolación podría ser la apropiada. Use una interpolación polinomial de quinto grado para ajustar los datos y calcule V para $i = 0.10$.

19. La aceleración debido a la gravedad a una altitud y por arriba de la superficie de la tierra está dada por

y, m	0	20 000	40 000	60 000	80 000
g, m/s^2	9.8100	9.7487	9.6879	9.6278	9.5682

Calcule g para $y = 55\,000$ m.

20. Capture los programas del Algoritmo 4 hechos en Matlab, y úselos para obtener las gráficas del spline cúbico de los Ejemplos 2.8 y 2.9. Use dichos programas para resolver los problemas que siguen.

21. Dibuje las gráficas de los splines cúbicos naturales que suavizan la función de Runge, juntos con la función de Runge, con los datos obtenidos por los puntos equidistantes:

$$x_i = -1 + (i-1)\frac{2}{n}, \quad i = 1, 2, ..., n+1,$$

para $n = 4$, 8 y 12. Las 4 gráficas se deben dibujar en un mismo sistema de coordenadas. Compare sus gráficas con las gráficas del problema 12 y escriba sus conclusiones.

22. Consideramos nuevamente los datos de la población de Tabasco del problema 14. Use spline cúbico natural para:

a) Estimar la población en el año 1975. Compare su resultado con lo obtenido en el problema 14.
b) Estimar la población en los años 1955 y 1985. Compare sus resultados con lo investigado en el problema 14.
c) Pronostique la población del año 2000. Compare su resultado con lo obtenido en el problema 14.

23. Consideramos los datos dados en el problema 15.

a) Use spline cúbico natural para suavizar la curva del peso promedio para cada muestra.
b) Encuentre una aproximación al valor máximo del peso promedio de cada muestra determinando el máximo del spline cúbico correspondiente.
c) Compare sus resultados con los resultados obtenidos en el problema 15.

24. La distancia requerida para detener un automóvil es una función de su velocidad. Los siguientes datos experimentales fueron colectados para cuantificar esta relación:

Velocidad, mi/h	15	20	25	30	40	50	60
Distancia de frenados, pies	16	20	34	40	60	90	120

Estime la distancia de frenado para un carro que circula a 45 mi/h,

a) usando primero interpolación polinomial y
b) usando un spline cúbico natural.
c) ¿Cuál de los dos resultados es más exacto?. Justifique su respuesta.

25. Un problema de ingeniería ambiental: Los lagos en la zona templada pueden dividirse en estratos térmicos durante el verano. Cerca de la superficie, el agua es tibia y liviana, y en el fondo es más fría y densa. Tal estratificación divide efectivamente el lago de manera vertical en dos capas: el *epilimnio* y el *hipolimnio* separados por un plano conocido como el *thermocline*. La estratificación térmica tiene gran importancia para los ingenieros ambientales que estudian la contaminación de tales sistemas. En particular, la thermocline disminuye en gran medida el mezclado entre las dos capas. Como resultado, la descomposición de materia orgánica puede acarrear reducción de oxígeno en el fondo aislado de las aguas. La ubicación de la thermocline se puede definir como el punto de inflexión de la curva temperatura-profundidad: es decir, el punto en el cual $\frac{d^2T}{dz^2} = 0$. Es también el punto en el cual el valor absoluto de la primera derivada o gradiente es un máximo. Use spline cúbico para determinar la profundidad de la thermocline para el lago Platte, y el valor del gradiente en la thermocline. Los datos de la temperatura contra la profundidad durante el verano para el lago Platte en Michigan está dado en la siguiente tabla:

z, m	0	2.3	4.9	9.1	13.7	18.3	22.9	27.2
T, °C	28.8	28.8	22.8	20.6	13.9	11.7	11.1	11.1

26. Un reactor está térmicamente estratificado con los valores de la siguiente tabla:

Profundidad z, m	0	0.5	1.0	1.5	2.0	2.5	3.0
Temperatura T, °C	70	68	55	22	13	11	10

Se puede idealizar el tanque del reactor como dos zonas separadas por un fuerte gradiente de temperatura o *thermocline*. La profundidad de este gradiente se puede definir como el punto de inflexión de la curva temperatura-profundidad;

es decir, el punto para el cual $\frac{d^2T}{dz^2} = 0$. A esta profundidad, el flujo de calor de la superficie al fondo de la capa se puede calcular con la ley de Fourier, $J = -k\frac{dT}{dz}$. Use spline cúbico para el ajuste de estos datos con el fin de determinar la profundidad de la thermocline. Si $k = 0.01$ $^{cal}/_{(s.cm.°C)}$, calcule el flujo a través de esta interface.

27. El esfuerzo cortante, en kips por pie cuadrado (kpc) de nueve muestras tomadas a distintas profundidades en un estrato de arcilla son:

Profundidad z, m	1.9	3.1	4.2	5.1	5.8	6.9	8.1	9.3	10.0
Esfuerzo, kpc	0.3	0.6	0.4	0.9	0.7	1.1	1.5	1.3	1.6

Estime el esfuerzo cortante a una profundidad de 4.5 m.

28. Se realiza un experimento para determinar el porcentaje de alargamiento de un material conductor eléctrico como una función de la temperatura. Los datos resultantes son:

Temperatura, °C	200	250	300	375	425	475	525	600
Porcentaje de alargamiento	11	13	13	15	17	19	20	23

Prediga el porcentaje de alargamiento para una temperatura de 400 °C.

29. Las funciones de Bessel surgen con frecuencia en análisis avanzado de ingeniería, como en el estudio de los campos eléctricos por ejemplo. Esas funciones son usualmente no sujetas a una evaluación directa y, por tanto, a menudo se compilan en tablas matemáticas estándar. Por ejemplo

x	1.8	2.0	2.2	2.4	2.6
$J_0(x)$	0.3400	0.2239	0.1104	0.0025	0.0968

Estime $J_0(2.1)$.

a) Mediante una interpolación polinomial, y
b) Usando spline cúbico natural.
c) El valor real es 0.1666. ¿Cuál es el error de aproximación en los incisos anteriores?

ALFABETO GRIEGO

Mayúscula	Minúscula	Nombre	Abecedario equivalente
Α	α	Alfa	a
Β	β	Beta	b
Γ	γ	Gamma	g
Δ	δ	Delta	d
Ε	ε	Epsilon	e
Ζ	ζ	Zeta	z
Η	η	Eta	$\bar{e}$
Θ	θ	Theta	th
Ι	ι	Iota	i
Κ	κ	Kappa	k
Λ	λ	Lambda	l
Μ	μ	My o Mu	m
Ν	ν	Ny o Nu	n
Ξ	ξ	Xi	x
Ο	ο	Omicron	o
Π	π	Pi	p
Ρ	ρ	Rho	r
Σ	σ, ς	Sigma	s
Τ	τ	Tau	t
Υ, ϒ	υ	Ipsilon	y
Φ	ϕ, φ	Phi o Fi	ph
Χ	χ	Chi o Ji	kh
Ψ	ψ	Psi	ps
Ω	ω	Omega	w

Bibliografía

[1] Atkinson, K.E.: *Elementary Numerical Analysis*; John Wiley & Sons, Inc., USA, 1993.

[2] Barrera, *et al.*: *Determinación de la Relación SiO_2 / Al_2O_3 en la Desalumination de Zeolitas*. Revista del Instituto Mexicano del Petróleo. Vol. XVI, Núm. 1, pp. 54 – 63. Enero / 1984.

[3] Barrera–Sánchez, Pablo, *et al.*; *EL ABC de los Splines*. Aportaciones Matemáticas de la S.M.M. 1996.

[4] Burden, R.L., y Faires, J.D.: *Análisis Numérico*. 2ª. edición; G.E.I.: México, 1996.

[5] Chapra, S.C. y Canale, R.P.: *Métodos Numéricos para ingenieros*. 3ª. edición; McGraw-Hill, México 1999.

[6] Eldén, L. and Wittmeyer–Koch, L.; *Numerical Analysis, An Introduction*; Academic Press, Inc., San Diego, USA, 1990.

[7] Gerald, C.F. y Wheatley, P.O.: *Applied Numerical Analysis*, 6ª Edition; Addison – Wesley Longman; USA, 1999.

[8] Moler, C.B.: *Floating points*. Matlab News & Notes, pag. 11–13, USA, 1996.

[9] Rivera-Ramírez y López-Estrada, J.; *Fracaso en el Cálculo Numérico a la Arquimedes del Número π*. Eureka No. 9, pág. 14–22. UAQ. Méx. Marzo de 1997.

[10] Scheid y Di Constanzo; *Métodos Numéricos*; McGraw–Hill; México, 1991.

[11] Vandergraft, J. S.: *Introduction to Numerical Computtios*, 2nd, edition; Academic Press, Inc., USA, 1983.

[12] Zill, Dennis G.: *Cálculo con Geometria Analitica*; G.E.I. S.A. México, 1987.

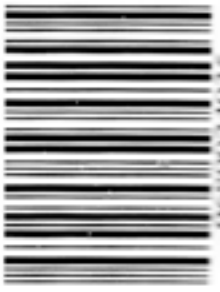
METODOS NUMERICOS I SE TERMINO DE IMPRIMIR EN EL MES DE
OCTUBRE DE 2006 EN LOS TALLERES ARTES-GRAFICOS

IMPRESION-ES
LITOMAR

UBICADO EN LA CALLE E. ZAPATA 103 COL. ATASTA
TELS.: (01-993) 315 20 97 Y 313 91 06
VILLAHERMOSA, TABASCO; MEXICO.

SU EDICION CONSTO DE 150 EJEMPLARES.

UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO



COLECCIÓN
HÉCTOR OCHOA BACELIS
TEXTOS DE ENSEÑANZA DE CIENCIAS BÁSICAS

DEPARTAMENTO
editorialcultural

L. D. Guillermo Narváez Osorio
Rector

Dra. Dora María Frías Márquez
Secretaria de Servicios Académicos

Mtro. Miguel Ángel Ruiz Magdónel
Director de Difusión Cultural

Mtro. Fredys Pérez Ruiz
Jefe del Departamento Editorial Cultural

• *Digitalización:* Olga Patricia Carrasco García
• *Edición, maquetación y diseño:* Yohana Noriega Alcudia, Fernando Ramos Bedoy y Fredys Pérez Ruiz.

Esta obra se terminó de digitalizar el año 2021. En Villahermosa, Tabasco, México. El cuidado de la edición estuvo a cargo del Departamento Editorial Cultural de la Dirección de Difusión Cultural y el Fondo Editorial Universitario.

Universidad Juárez Autónoma de Tabasco

"Estudio en la duda. Acción en la fe"