Curso de Métodos Numéricos

DEMAT, Universidad de Guanajuato

Clase 3: Propagación de errores

- Error relativo en las operaciones aritméticas
- Ejemplos

MAT–251

Dr. Joaquín Peña Acevedo
CIMAT A.C.
e-mail: joaquin@cimat.mx

Propagación del error en el producto

Si $fl(x) = x(1+\delta_x)$, $\quad fl(y) = y(1+\delta_y)$, donde $|\delta_x|, |\delta_y| \leq u$, entonces

$$\delta_{xy} = \frac{fl(x)fl(y) - xy}{xy} = \frac{xy(1+\delta_x)(1+\delta_y) - xy}{xy} = \delta_x + \delta_y + \delta_x\delta_y$$

Entonces el error relativo asociado al producto es

$$|\delta_{xy}| = |\delta_x + \delta_y + \delta_x\delta_y| \leq |\delta_x| + |\delta_y| + |\delta_x|\,|\delta_y| \leq 2u + u^2 \leq 3u$$

Propagación del error en la división

Si $fl(x) = x(1+\delta_x)$, $fl(y) = y(1+\delta_y)$, donde $|\delta_x|, |\delta_y| \leq u$, y $y \neq 0$, se tiene que:

$$\delta_{\frac{x}{y}} = \frac{\frac{fl(x)}{fl(y)} - \frac{x}{y}}{\frac{x}{y}} = \frac{\frac{x(1+\delta_x)}{y(1+\delta_y)} - \frac{x}{y}}{\frac{x}{y}} = \frac{1+\delta_x}{1+\delta_y} - 1 = \frac{\delta_x - \delta_y}{1+\delta_y}$$

Entonces el error relativo asociado a la división es

$$|\delta_{\frac{x}{y}}| \leq \frac{|\delta_x| + |\delta_y|}{1-u} \leq \frac{2u}{1-u} \leq \frac{2u}{0.5} = 4u$$

Resumen de errores relativos asociados a las operaciones aritméticas

- El error relativo que se comete al realizar la suma, el producto o el cociente de las representaciones de dos números puede acotarse por la unidad de redondeo.

$$\delta_{x \circ y} \leq cu, \quad \text{donde} \quad \circ = +, \times, /, \quad \text{y } c \text{ es una constante}$$

- El error relativo que se comete al realizar una resta de las representaciones de dos números queda acotado por la unidad de redondeo y por el recíproco del valor absoluto de la resta de los números:

$$|\delta_{x-y}| \leq u \frac{|x| + |y|}{|x - y|}$$

- De este modo, los errores relativos debidos a las operaciones de suma, producto y división mejoran al aumentar la precisión de los números, pero en la resta el error relativo puede ser que no mejore tanto.

Error al combinar operaciones aritméticas

- Se puede hacer un análisis de la propagación del error cuando se realizan varias operaciones aritméticas, pero los desarrollos se van complicando de acuerdo a la cantidad de operaciones.

- Por esta razón, no lo haremos y sólo nos quedamos con la información anterior, teniendo cuidado con la resta, y si es posible, tratar de realizar el menor número de operaciones.

Ejemplos (I)

Ejemplo 1. Consideremos la función

$$f(x) = x\left(\sqrt{x+1} - \sqrt{x}\right)$$

Se puede ver que la función es creciente para $x \geq 0$.
Si queremos evaluarla en la computadora cuando x va aumentado de valor, ¿qué resultados obtendremos?
En lugar de evaluar $f(x)$ podemos utilizar

$$g(x) = \frac{x}{\sqrt{x+1} + \sqrt{x}}.$$

y comparar los resultados con $h(x) = \frac{\sqrt{x}}{2}$:

$$2\sqrt{x} \leq \sqrt{x+1} + \sqrt{x} \quad \Longrightarrow \quad h(x) \geq g(x).$$

Ejemplos (II)

Usando variables de 64 bits:

x	f(x)	g(x)	h(x)
1.0e+05	1.581134877e+02	1.581134877e+02	1.581138830e+02
1.0e+06	4.999998750e+02	4.999998750e+02	5.000000000e+02
1.0e+07	1.581138790e+03	1.581138791e+03	1.581138830e+03
1.0e+08	5.000000056e+03	4.999999988e+03	5.000000000e+03
1.0e+09	1.581139077e+04	1.581138830e+04	1.581138830e+04
1.0e+10	4.999994417e+04	5.000000000e+04	5.000000000e+04
1.0e+11	1.581152901e+05	1.581138830e+05	1.581138830e+05
1.0e+12	5.000038072e+05	5.000000000e+05	5.000000000e+05
1.0e+13	1.578591764e+06	1.581138830e+06	1.581138830e+06
1.0e+14	5.029141903e+06	5.000000000e+06	5.000000000e+06
1.0e+15	1.862645149e+07	1.581138830e+07	1.581138830e+07
1.0e+16	0.000000000e+00	5.000000000e+07	5.000000000e+07
1.0e+17	0.000000000e+00	1.581138830e+08	1.581138830e+08
1.0e+18	0.000000000e+00	5.000000000e+08	5.000000000e+08
1.0e+19	0.000000000e+00	1.581138830e+09	1.581138830e+09

Ejemplos (III)

Y si en vez de 64 bits usamos variables tipo 32 bits se obtiene:

x	f(x)	g(x)	h(x)
1.0e+00	4.142135382e-01	4.142135680e-01	5.000000000e-01
1.0e+01	1.543471813e+00	1.543471217e+00	1.581138849e+00
1.0e+02	4.987525940e+00	4.987562180e+00	5.000000000e+00
1.0e+03	1.580810547e+01	1.580743790e+01	1.581138802e+01
1.0e+04	4.997253418e+01	4.999875259e+01	5.000000000e+01
1.0e+05	1.586914062e+02	1.581134949e+02	1.581138763e+02
1.0e+06	4.882812500e+02	4.999998779e+02	5.000000000e+02
1.0e+07	2.441406250e+03	1.581138794e+03	1.581138794e+03
1.0e+08	0.000000000e+00	5.000000000e+03	5.000000000e+03
1.0e+09	0.000000000e+00	1.581138770e+04	1.581138867e+04
1.0e+10	0.000000000e+00	5.000000000e+04	5.000000000e+04
1.0e+11	0.000000000e+00	1.581138906e+05	1.581138750e+05
1.0e+12	0.000000000e+00	5.000000000e+05	5.000000000e+05
1.0e+13	0.000000000e+00	1.581138750e+06	1.581138875e+06
1.0e+14	0.000000000e+00	5.000000000e+06	5.000000000e+06
1.0e+15	0.000000000e+00	1.581138800e+07	1.581138800e+07
1.0e+16	0.000000000e+00	5.000000000e+07	5.000000000e+07

Ejemplos (IV)

Ejemplo 2. Considere las siguientes expresiones:

$$
\begin{array}{rcrcrcrcr}
s_1 & = & 10^{22} & + & 17 & - & 10 & + & 130 & - & 10^{22} \\
s_2 & = & 10^{22} & - & 10 & + & 130 & - & 10^{22} & + & 17 \\
s_3 & = & 10^{22} & + & 17 & - & 10^{22} & - & 10 & + & 130 \\
s_4 & = & 10^{22} & - & 10 & - & 10^{22} & + & 130 & + & 17 \\
s_5 & = & 10^{22} & - & 10^{22} & + & 17 & - & 10 & + & 130 \\
s_6 & = & 10^{22} & + & 17 & + & 130 & - & 10^{22} & - & 10
\end{array}
$$

En teoría deberían dar el mismo resultado, pero si no se tiene una buena precisión,
¿cuales son los valores que se obtienen en la computadora?

Evaluación de polinomios (I)

Evaluamos el polinomio cúbico

$$p(x) = ax^3 + bx^2 + cx + d$$

donde

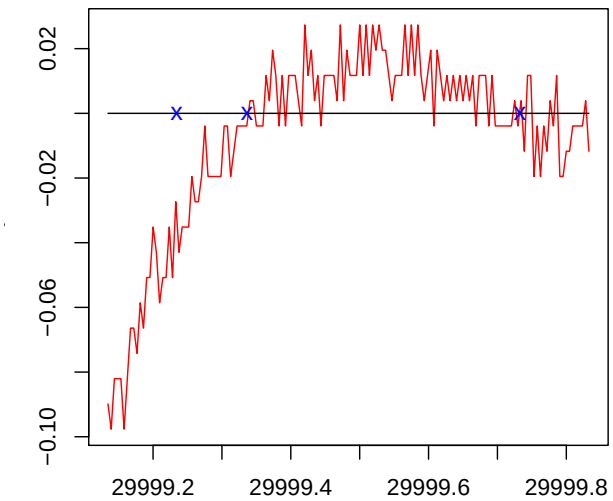
$$\begin{aligned} a &= 1.000, \\ b &= -89998.304, \\ c &= 2699898236.405, \\ d &= -26998473559412.543, \end{aligned} \tag{1}$$

Evaluación de polinomios (I)

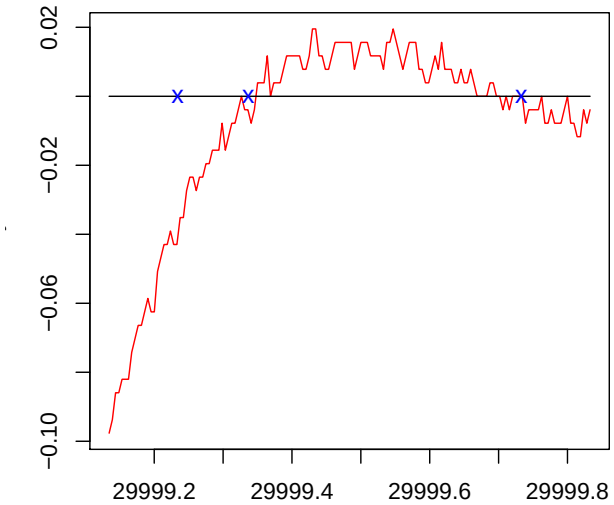
Evaluamos el polinomio cúbico

$$p(x) = ax^3 + bx^2 + cx + d$$

donde
$$\begin{aligned} a &= 1.000, \\ b &= -89998.304, \\ c &= 2699898236.405, \\ d &= -26998473559412.543, \end{aligned} \tag{1}$$



$((ax^3 + bx^2) + cx) + d$



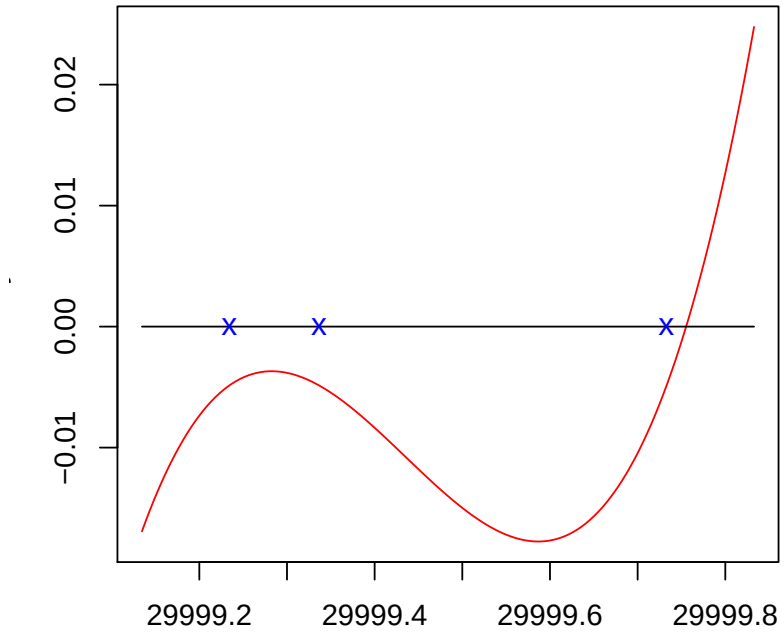
$((ax + b)x + c)x + d$

Evaluación de polinomios (II)

Usando algunas estrategias, como las que se describen en

> *S. Graillat, P. Langlois and N. Louvet. Algorithms for accurate, validated and fast polynomial evaluation. Japan J. Indust. Appl. Math., vol. 26, pp. 191–214, 2009*

se obtiene lo siguiente:



Evaluación de polinomios (III)

El polinomio $p(x)$ se obtuvo al desarrollar la expresión

$$p(x) = (x - s_1)(x - s_2)(x - s_3)$$

donde

$$s_1 = 29999.234288122, \quad s_2 = 29999.336581462, \quad s_3 = 29999.733055670,$$

de modo que

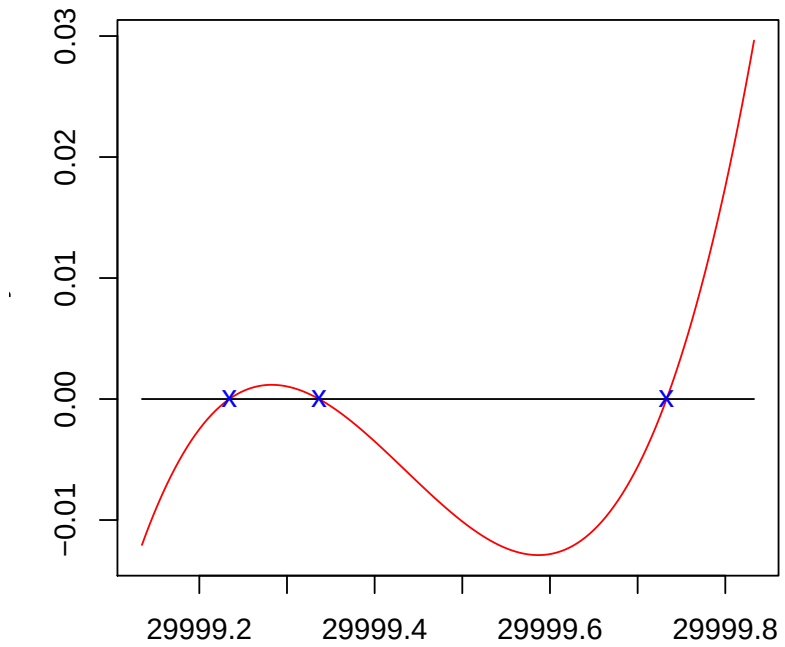
$$\begin{aligned} a &= 1 \\ b &= -s_1 - s_2 - s_3 \\ c &= s_2 s_3 + s_1 s_3 + s_1 s_2 \\ d &= -s_1 s_2 s_3 \end{aligned}$$

Los puntos indicados con 'x' en la gráfica anterior son las raíces de $p(x)$. Es posible estimar los valores y_i que corrigen el cálculo de los coeficientes del polinomio

$$\begin{aligned} a &= fl(a) + y_1 \qquad b = fl(b) + y_2 \\ c &= fl(c) + y_3 \qquad d = fl(d) + y_4 \end{aligned}$$

Evaluación de polinomios (IV)

Entonces $p(x) = Horner(p, x) + y(x)$ y al evaluarlo se obtiene



En general, se puedan estrategias para calcular los coeficientes de un polinomio dadas sus raíces:

> *Calvetti, D. and Reichel, Lothar. On the evaluation of polynomial coefficients. Numerical Algorithms 33, pp. 153-161, 2003.*

Ejemplo de evaluación de polinomios

El polinomio de Rump se define como

$$R(x,y) = \frac{33375}{100}y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + \frac{55}{10}y^8 + \frac{x}{2y}$$

Si evaluamos este polinomio usando ’double’ y ’long double’ se tiene que

$$\text{(float)} \quad R(77617,\ 33096) \quad = \quad 1.1726039648056030$$

Ejemplo de evaluación de polinomios

El polinomio de Rump se define como

$$R(x,y) = \frac{33375}{100}y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + \frac{55}{10}y^8 + \frac{x}{2y}$$

Si evaluamos este polinomio usando 'double' y 'long double' se tiene que

$$\text{(float)} \quad R(77617,\ 33096) \quad = \quad 1.172603964805603 0$$

$$\text{(double)} \quad R(77617,\ 33096) \quad = \quad -1.1805916207 \times 10^{21}$$

Ejemplo de evaluación de polinomios

El polinomio de Rump se define como

$$R(x,y) = \frac{33375}{100}y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + \frac{55}{10}y^8 + \frac{x}{2y}$$

Si evaluamos este polinomio usando 'double' y 'long double' se tiene que

$$\text{(float)} \quad R(77617,\ 33096) \quad = \quad 1.172603964805603 0$$

$$\text{(double)} \quad R(77617,\ 33096) \quad = \quad -1.1805916207 \times 10^{21}$$

$$\text{(long double)} \quad R(77617,\ 33096) \quad = \quad 5.7646075230 \times 10^{17}$$

Ejemplo de evaluación de polinomios

El polinomio de Rump se define como

$$R(x,y) = \frac{33375}{100}y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + \frac{55}{10}y^8 + \frac{x}{2y}$$

Si evaluamos este polinomio usando 'double' y 'long double' se tiene que

$$\text{(float)} \quad R(77617,\ 33096) \quad = \quad 1.172603964805603 0$$

$$\text{(double)} \quad R(77617,\ 33096) \quad = \quad -1.1805916207 \times 10^{21}$$

$$\text{(long double)} \quad R(77617,\ 33096) \quad = \quad 5.7646075230 \times 10^{17}$$

Realizando las operaciones con fracciones, obtenemos

$$R(77617, 33096) = -\frac{54767}{66192} \approx -0.8273960599$$

Ejemplo de evaluación de polinomios

Si

$$\begin{aligned} R_1(x,y) &= \frac{33375}{100}y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) \\ R_2(x,y) &= \frac{55}{10}y^8 \\ R_3(x,y) &= \frac{x}{2y} \end{aligned}$$

Entonces

$$\begin{aligned} R_1(77617,\ 33096) &= -79171113406689613611011347015249428 50, \\ R_2(77617,\ 33096) &= \ \ 79171113406689613611011347015249428 48, \\ R_3(77617,\ 33096) &= \frac{77617}{66192} \end{aligned}$$

Lectura de datos desde la línea de comandos en C (I)

- En cada programa queremos realizar un conjunto de pruebas, lo que implica cambiar los valores de entrada de los algoritmos y no queremos recompilarlos en cada prueba.
- Eso requiere poder leer los datos de entrada de manera independiente.
- Usamos los argumentos de la función `main()` para obtener la información. Ejemplo (código *argumentos.cpp*)

```
int main(int argc, char **argv)    {
    int     i, n;
    double  dval;

    printf("\nLista de argumentos:\n");
    for(i=0; i<argc; ++i) {
        printf("\tArgumento %d: %s\n", i, argv[i]);
    }
    if(argc>0) n    = atoi(argv[1]);
    if(argc>1) dval = atof(argv[2]);

    printf("\nEntero    = %d\n", n);
    printf("Flotante  = %lf\n", dval);
```

Lectura de datos desde la línea de comandos en C (II)

```
    return(0);
}
```

Ejemplo de una llamada del programa desde la línea de comandos:

```
./argumentos -5 0.12345 y el resto de argumentos
```

Otro ejemplo se muestra en el código `evaluacionFnc.cpp`, el cual evalua tres funciones en intervalo definido por los valores que lee desde la línea de comandos:

```
./evaluacionFnc 10 1e11
```

En el código se puede cambiar tipo de las variables para ver como se modifican los resultados.

Lectura de datos desde la línea de comandos en Python

En Python hay que usar la librería sys para leer los argumentos desde la línea de comandos. Por ejemplo:

```
import sys

nargs =  len(sys.argv)
print("Numero de argumentos: ", nargs )
for i, str_arg in enumerate(sys.argv):
    print("\tArgumento %d: %s" % (i, sys.argv[i]))

n    = 1
dval = 3.1416

if nargs>1:
    n    = int(sys.argv[1])
if nargs>2:
    dval = float(sys.argv[2])

print("\nEntero    = ", n, type(n));
print("Flotante  = ", dval, type(dval));
```

Ejemplo de evaluación de funciones

Ejemplo:

```
double x = 2.0;

for(int i=1; i<=60; ++i) {
    x = sqrt(x);
}
for(int i=1; i<=60; ++i) {
    x = x*x;
}
```

¿Qué valor se obtiene para $x = 2$? ¿ Para $x = 10^{10}$?

Orden de los términos en sumas

Consideremos las sumas

$$S_1(n) = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n-1} + \frac{1}{n},$$

$$S_2(n) = \frac{1}{n} + \frac{1}{n-1} + \ldots + \frac{1}{3} + \frac{1}{2} + 1.$$

Vale la pena comparar los resultados entre $S_1(n)$ y $S_2(n)$ para diferentes valores de $n \geq 1$

Algoritmo de Kahan (I)

Cuando no se ordenan las cantidades que se van a sumar conviene usar un algoritmo que reduzca el efecto de los errores por redondeo. Uno de estos algoritmos es el de Kahan:

Algoritmo 1: Algoritmo de Kahan

Entrada: Lista de valores $a_0, a_1, \ldots, a_n$.
Salida: El resultado de la suma de los numeros dados
Inicializar: $S = a_0$, $c = 0$.
Para $k = 1, 2, \ldots, n$:

1. $y = a_k - c$
2. $t = S + y$
3. $c = (t - S) - y$
4. $S = t$

Devolver el valor de S.

FLOPS

Para evaluar un algoritmo se necesita evaluar la cantidad de memoria que requiere, la precisión de sus resultados y la cantidad de tiempo que requiere para realizar los cálculos.

Para determinar el tiempo, se puede contar el número de *flops*.
flop es un acrónimo de "floating-point operation". Un flop denota una suma, resta, multiplicación o división de números de punto flotante. Por ejemplo:

- Si $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, el producto punto $\boldsymbol{a}^\top \boldsymbol{b} = \sum_{i=1}^n a_i b_i$ requiere aproximadamente $2n$ flops (n productos y $n-1$ sumas)
- Si $\boldsymbol{M} \in \mathbb{R}^{m \times n}$, el producto $\boldsymbol{M}\boldsymbol{a}$ requiere $2nm$ flops.

Tiempo de cómputo

Contar el número de flops da una medida burda del costo computacional de un algoritmo, pero se requiere considerar otros aspectos para estimar el tiempo de ejecución.

Para relacionar el número de operaciones con el tiempo, conviene saber el tiempo promedio que cada operación aritmética o evaluación de funciones elementales toma.