Universidad de Puerto Rico
Departamento de Matemáticas
Humacao, Puerto Rico 00791

MATE 4061 Análisis Numérico Prof. Pablo Negrón

Laboratorio 3: Representación y Aritmética de Punto Flotante

Si $\bar{x}$ representa una aproximación al número real x, entonces $|\hat{x} - x|$ se llama el *error absoluto* y $|\hat{x} - x| \,/\, |x|$ se conoce como el *error relativo*. Cuando el error relativo es del orden de 10^{-t}, entonces $\hat{x}$ tiene aproximadamente t cifras significativas correctas como aproximación a x.

Al representar un número en la computadora, si este no tiene representación exacta en el sistema de punto flotante, se incurre en un pequeño error. Al efectuar una operación aritmética en la computadora, si el resultado no tiene representación exacta en la computadora, también se induce un pequeño error. Estos errores aunque pequeños, pueden acumularse y ser significativos. Esto no necesariamente implica un problema con el sistema de computación sino que es consecuencia natural del uso de un numero finito de cifras en los cómputos. Veamos un ejemplo que ilustra estas ideas.

Ejemplo 1. El polinomio de Taylor de grado n para la función $f(x) = \mathrm{e}^x$ alrededor de $x = 0$ esta dado por:

$$p_n(x) = \sum_{k=0}^{n} \frac{x^k}{k!},$$

con un residual:

$$R_n(x) = \frac{\mathrm{e}^{c_x}}{(n+1)!} x^{n+1},$$

donde c_x esta entre 0 y x. Se puede verificar que para una x fija el residual $R_n(x)$ tiende a cero según $n \to \infty$. Este análisis del error asume que la aritmética es exacta. Veamos que sucede con $R_n(x)$ cuando dejamos que la n aumente en la computadora. El siguiente programa en MATLAB calcula la función e^x para $x = -10, -5, -1, 1, 5, 10$, los primeros 50 polinomios de Taylor alrededor de cero evaluados en estas x, y los 50 residuales en cada valor de x.

```
nTerms=50;
error=zeros(nTerms,1);
x=[-10 -5 -1 1 5 10];
for j=1:6
   exact=exp(x(j));
   pn=1;
   termk=1;
   for k=1:nTerms
      termk=x(j)*termk/k;
      pn=pn+termk;
      error(k)=abs(exact-pn);
   end
   errrel=error/exact;
   subplot(2,3,j)
   semilogy(1:nTerms,errrel)
   xlabel('Grado del Polinomio')
   ylabel('Error Relativo')
   title(sprintf('x= %5.2f',x(j)))
end
```

Podemos observar en la Figura (1) que los residuales no tienden a cero como predice la teoría. De hecho se estancan de un n en adelante. Esto se debe a la aritmética de punto flotante ya que para k suficientemente grande (depende de la x), el termino $x^k/k!$ deja de contribuir a la sumatoria. Note que el punto donde el error se estanca y el tamaño del error cuando esto ocurre, depende del valor de x.

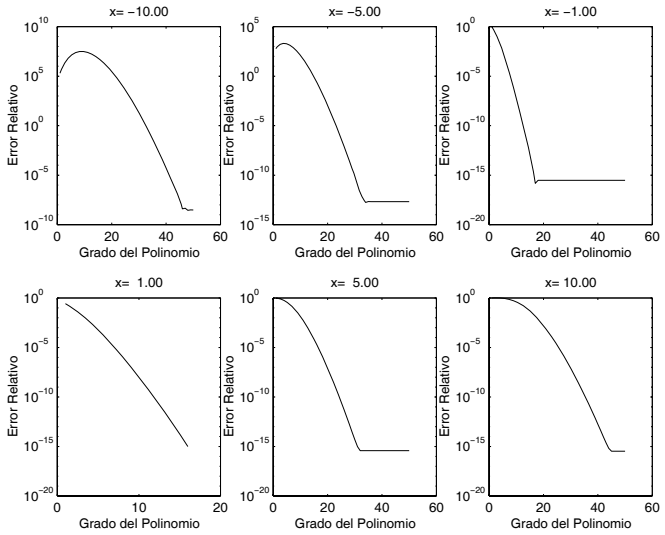

Figure 1: Comportamiento del error relativo en los primeros 50 polinomios de Taylor de e^x para distintos valores de x.

Un *sistema de punto flotante* se especifica por la *base* β, el *largo de mantisa* t, y limites para los exponentes de $-N, M$. Un numero de punto flotante tiene la forma

$$x = \pm 0.b_1 b_2 \dots b_t \times \beta^e,$$

donde $0.b_1 b_2 \dots b_t$ es la *mantisa*, $b_1 \neq 0$ (para $x \neq 0$), $0 \leq b_i \leq \beta - 1$ para $2 \leq i \leq t$, y e el *exponente* el cual satisface $-N \leq e \leq M$. El *cero* se representa con mantisa cero y exponente cero. El sistema de punto flotante se representa por $F(\beta, t, N, M)$.

Ejemplo 2. Tomando $(\beta, t, N, M) = (10, 2, 1, 2)$, tenemos 90 posibles mantisas, y 4 exponentes, i.e., $-1, 0, 1, 2$. Como hay dos posibles signos, tenemos un total de $2(90)(4) + 1 = 721$ numeros en el sistema. Note que el sistema de punto flotante es finito. Estos numeros se distribuyen en forma no uniforme en la recta real. (Ver Figura (2)). De hecho se distribuyen en grupos de numeros donde los numeros en cada grupo difieren por $\beta^{e-t} = 10^{e-2}$ donde e asume los valores permitidos de los exponentes. Con el siguiente programa podemos trazar los numeros positivos del sistema de punto flotante:

```
x=zeros(4*90,1);
y=ones(4*90,1);
i=1;
for e=-1:2
   for f=0.10:0.01:0.99
      x(i)=f*10^e;
      i=i+1;
   end
end
semilogx(x,y,'+')
```

En este sistema, el numero positivo mas pequeño es $0.10 \times 10^{-1} = 0.01$. El numero positivo más grande es $0.99 \times 10^2 = 99$.

Si x es un numero real, entonces $fl(x)$ denota la representación de punto flotante de x en el sistema $F(\beta, t, N, M)$. El siguiente resultado nos da un estimado del error en dicha representación:

Teorema 1. *Sea $x \in \mathbb{R}$ y $fl(x)$ su representación de punto flotante en el sistema $F(\beta, t, N, M)$. Si $\beta^{-N} < |x| < \beta^M$, entonces el error relativo en $fl(x)$ como aproximación de x es a lo más $c\beta^{1-t}$ donde $c = 1$ en truncación y $c = 1/2$ en redondeo.*

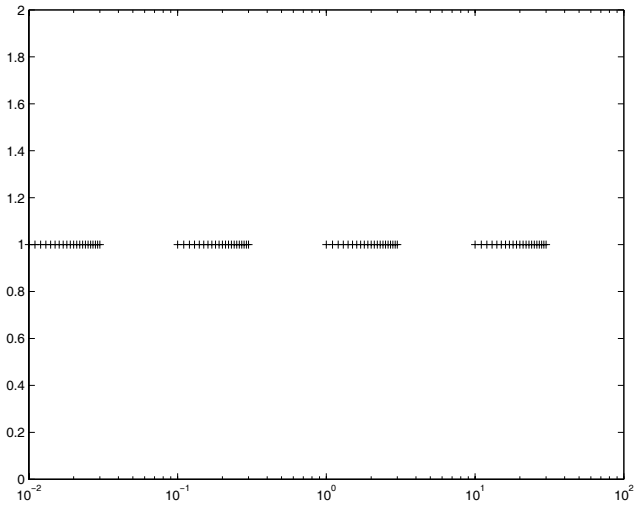

Figure 2: Gráfica en escala logarítmica para los numeros positivos del sistema de punto flotante $(\beta, t, N, M) = (10, 2, 1, 2)$.

Demostración: Veamos la demostración de este resultado para el caso de truncación. En este caso sabemos que si $x = \sigma(0.a_1a_2\ldots)_\beta \times \beta^e$, entonces $fl(x) = \sigma(0.a_1a_2\ldots a_t)_\beta \times \beta^e$. De modo que

$$|x - fl(x)| = (0.a_{t+1}a_{t+2}\ldots)_\beta \times \beta^{e-t}.$$

Como $(0.a_1a_2\ldots)_\beta$ esta entre $(0.1)_\beta = \beta^{-1}$ y uno, tenemos que $|x| \geq \beta^{e-1}$. Así que

$$\frac{|x - fl(x)|}{|x|} \leq \frac{(0.a_{t+1}a_{t+2}\ldots)_\beta \times \beta^{e-t}}{\beta^{e-1}} \leq \frac{\beta^{e-t}}{\beta^{e-1}} = \beta^{1-t}.$$

□

Las operaciones de suma, resta, multiplicación, y división en la computadora se pueden modelar en forma razonable como sigue:

- la operación aritmética se lleva a cabo en forma exacta.
- El resultado se escribe en su forma normalizada, i.e., en notación científica.
- Ahora se redondea la mantisa al numero de cifras del sistema de punto flotante.

Ejemplo 3. En el sistema de punto flotante (10,3,9,9) considere los numeros 12.3 y 5.27 los cuales tienen representación exacta en este sistema. El resultado exacto de la suma de estos dos numeros es 17.57. La representación de este resultado en el sistema de punto flotante usando redondeo es 0.176×10^2, lo cual se toma como el resultado de la suma en el sistema de punto flotante. Note que esta operación indujo un error, en este caso de 0.03.

Ejercicio 1. *Ejecute el siguiente programa en MATLAB, y describa en palabras lo que hace cada ciclo "while":*

```
x=1; p=0; y=1; z=x+y;
while x~=z
   y=y/2;
   p=p+1;
   z=x+y;
end;
echo off;
```

```
disp(' ')
disp(sprintf('> p = %2.0f',p))
disp(' ')
disp('Strike any key to continue.')
pause
%
x=1; q=0;
while x>0
   x=x/2;
   q=q+1;
end;
echo off
disp(' ')
disp(sprintf('> q = %2.0f',q))
disp(' ')
disp('Strike any key to continue.')
pause
%
x=1; r=0;
while x~=inf
   x=2*x;
   r=r+1;
end
echo off;
disp(' ')
disp(sprintf('> r = %2.0f',r))
disp(' ')
```

Con el propósito de poder experimentar y estudiar mejor la propagación de errores en un sistema de punto flotante, vamos a describir un sistema $(\beta, t, N, M) = (10, 3, 9, 9)$ implementado en MATLAB. La implementación de este sistema utiliza un vector (v_1, v_2, v_3, v_4) para describir el numero de punto flotante

$$0.v_1v_2v_3 \times 10^{v_4}.$$

El signo de la mantisa y el del exponente son los de v_1 y v_4 respectivamente. El cero se representa con el vector $(0, 0, 0, 0)$ y el "overflow" con `inf`. Con esta representación, la implantación del sistema consiste de cuatro subrutinas que tienen las siguientes funciones:

`represent(x).` Esta función recibe un numero real `x` y devuelve su representación (v_1, v_2, v_3, v_4) en el sistema descrito arriba.

`convert(v).` Esta función convierte un numero en el sistema (v_1, v_2, v_3, v_4) al formato regular de punto flotante.

`float(x,y,op).` Dados `x`, `y` en el sistema (v_1, v_2, v_3, v_4), y la cadena `op` que puede ser $+, -, \times, \div$, se calcula `x op y`.

`pretty(v).` Convierte el numero `v` en el sistema (v_1, v_2, v_3, v_4) a una cadena apropiada para imprimir.

Ejemplo 4. Vamos a ilustrar el uso de estas subrutinas calculando la sumatoria

$$\sum_{k=1}^{\infty} \frac{1}{k}.$$

Usando aritmética exacta, esta sumatoria es ∞. Veamos que sucede en el sistema $(\beta, t, N, M) = (10, 3, 9, 9)$. El siguiente programa en MATLAB calcula la sumatoria de arriba en este sistema:

```
oldsum = represent(0);
one = represent(1);
sum = one;
k = 1;
while convert(sum) ~= convert(oldsum)
   k = k+1;
   kay  = represent(k);
   term = float(one,kay,'/');
   oldsum = sum;
   sum  = float(sum,term,'+');
   disp(['Suma hasta ' num2str(k) ' terminos = ' pretty(sum)])
end
```

El resultado es que el ciclo termina después de 200 términos con un valor de 6.14 para la suma. Los términos después del 200, no aportan a la sumatoria en este sistema.

Ejercicio 2. *Modifique los programas del Ejercicio (1) para el sistema de punto flotante* $(\beta, t, N, M) = (10, 3, 9, 9)$. *Ejecute el programa y compare los resultados obtenidos con los del Ejercicio (1).*

Ejercicio 3. *Usando los programas* `convert`, `represent`, `float`, *y* `pretty` *que emulan el sistema de punto flotante* $(\beta, t, N, M) = (10, 3, 9, 9)$, *escriba un programa en MATLAB que calcule las raíces de* $x^2 - 110x + 1 = 0$ *lo más preciso posible en dicho sistema.*

En clase estudiamos el problema de sumar muchos números en la computadora. Esto es, cálcular la sumatoria

$$S = a_1 + a_2 + \cdots + a_n = \sum_{i=1}^{n} a_i,$$

usando aritmética de punto flotante. Para esto definimos

$$\begin{cases} S_1 &= a_1, \\ S_j &= fl(S_{j-1} + a_j)\ ,\ 2 \leq j \leq n. \end{cases}$$

Usando que

$$S_j = (S_{j-1} + a_j)(1 + \varepsilon_j) \quad , \quad 2 \leq j \leq n,$$

donde $|\varepsilon_j| \leq c\beta^{1-t}$ con $c \leq 1$, vimos que

$$\begin{aligned} S - S_n \quad \approx \quad & -a_1(\varepsilon_2 + \varepsilon_3 + \cdots + \varepsilon_n) - a_2(\varepsilon_2 + \varepsilon_3 + \cdots + \varepsilon_n), \\ & -a_3(\varepsilon_3 + \varepsilon_4 + \cdots + \varepsilon_n) - \cdots - a_n\varepsilon_n. \end{aligned}$$

De aquí se deduce que para minimizar el error en la sumatoria se deben ordenar los numeros del mas pequeño (en valor absoluto) al mas grande.

Ejercicio 4. *Escriba un programa en MATLAB similar al del Ejemplo (4) para calcular en el sistema* $(10, 3, 9, 9)$ *la suma de los primeros* 100 *términos de la sumatoria*

$$\sum_{k=1}^{\infty} \frac{1}{k^2},$$

cuyo valor exacto es $\pi^2/6$. *Calcule la suma de dos formas: en el orden* $1 + 1/2^2 + 1/3^2 + \cdots + 1/100^2$ *y como* $1/100^2 + 1/99^2 + \cdots + 1/2^2 + 1$. *Compare ambos resultados con la representación en el sistema del valor exacto de la suma.*

Los programas a usarse en este laboratorio son: `texp.m`, `euler.m`, `floatnum.m`, `fpfacts.m`, `represent.m`, `convert.m`, `float.m`, y `pretty.m`.