Práctica voluntaria (1er. parcial) Técnicas Numéricas

Francisco R. Villatoro y Carmen M. García 6 de Octubre de 2000

PRIMER TEMA: Operaciones en coma flotante en Matlab

1. En clase hemos visto el formato de representación de números IEEE de simple precisión. Sin embargo, Matlab utiliza siempre doble precisión (8 bytes o 64 bits). Para la mantisa se utilizan 53 bits (incluido el bit de signo) y el exponente está entre $-1022 \leq e \leq 1023$.

   a) ¿Cuántos dígitos binarios o bits requiere el exponente?

   b) ¿Cómo se representa el 0?

   c) ¿Cómo se representan los números $\pm$`Inf` (infinito positivo y negativo) y `NaN` (el resultado no es un número)?

   d) ¿Cuál es el resultado de las operaciones `1/0`, `log(1/0)` y `1/0-log(1/0)`.

   e) ¿Cuántos números con exponente $e = 0$ puede representar Matlab?

   f) ¿Cuál es el número positivo más pequeño representable (para $e = 0$ y para $e \neq 0$)?, y, ¿cuál es el más grande?

2. Calcule el épsilon de la máquina para Matlab (comando `eps`). ¿Qué es el épsilon de la máquina? Escribe en representación flotante IEEE de doble precisión (de Matlab) el épsilon de la máquina (ε) y el número $1 + \varepsilon$. El siguiente programa

   ```
   e = 1;  while (1+e > 1), e=e/2; end,
   ```

   ¿calcula el epsilon de la máquina? Si no lo hace, cómo lo arreglarías y por qué.

3. ¿Escribe una cota superior del error de normalización de un número flotante cuando se utiliza redondeo y otra cuando se utiliza truncado? Justifica tu respuesta. Calcula estas cotas para Matlab.

4. Para calcular las integrales

$$E_n = \int_0^1 x^n \, e^{x-1} \, dx, \qquad n = 1, 2, \ldots,$$

   podemos usar integración por partes

$$\int_0^1 x^n \, e^{x-1} \, dx = x^n \, e^{x-1} \Big|_0^1 - \int_0^1 n \, x^{n-1} \, e^{x-1} \, dx,$$

con lo que obtenemos el siguiente algoritmo numérico recursivo

$$E_n = 1 - n\,E_{n-1}, \qquad n = 2, \ldots,$$

donde $E_1 = 1/e$. Implemente dicho algoritmo de forma recursiva en Matlab (utilice funciones `function`). Implemente dicho algoritmo de forma iterativa (no recursiva). ¿Los resultados de ambos algoritmos son exactamente los mismos? ¿Por qué? Dibuje la gráfica de E_n para $n = 1 \ldots 100$. ¿Obtiene algún valor negativo? ¿Por qué dicho resultado sería sorprendente?

Este algoritmo es inestable numéricamente. En su lugar, podemos utilizar la iteración hacia atrás

$$E_{n-1} = \frac{1 - E_n}{n}, \qquad n = \ldots, 3, 2,$$

tiene la ventaja que el error de redondeo original decrece conforme se itera. Para obtener un valor inicial para esta iteración se puede usar,

$$E_n = \int_0^1 x^n\, e^{x-1}\, dx \leq \int_0^1 x^n\, dx = \frac{1}{n+1}.$$

Implemente en Matlab este algoritmo de forma recursiva y luego de forma iterativa. ¿Obtiene los mismos resultados en ambos casos? Dibuje la gráfica de los valores de E_n para $n = 1 \ldots 100$. Compare la gráfica con la que obtuvo antes.

Finalmente, calcule la integral E_n utilizando la función `quad` de Matlab. Dibuje la gráfica de los resultados y compare éstos con los obtenidos en apartados anteriores.

5. El ejemplo clásico de Wilkinson (1963) sobre la inestabilidad de métodos numéricos y la propagación de errores es el polinomio

$$p(x) = (x-1)\,(x-2)\,\cdots\,(x-19)\,(x-20) = x^{20} - 210\,x^{19} + \ldots,$$

cuyas raíces son reales y están bien separadas. En Maltab:

```
lista = 1:20; factores = [lista*0+1; -lista]';
polinomio = [1];
for ind=1:20,
   polinomio = conv (polinomio, factores(ind,:));
```

```
end
roots (polinomio),
```

Ahora cambiaremos el coeficiente $-210\,x^19$ por $(-210 + 2^{-23})\,x^{19}$, y calcularemos de nuevo los ceros

```
polinomio(2) = polinomio(2) +2^(-23);
roots(polinomio),
```

¿cómo son los ceros del polinomio ahora? Comenta las diferencias entre los ceros de ambos polinomios.

¿Cómo estudiarías la sensibilidad del polinomio

$$p(x, \alpha) = x^{20} - \alpha\, x^{19} + \ldots,$$

ante cambios en el valor de α? Es decir, cómo determinarías la influencia de un pequeño cambio en α sobre cada una de las raíces del polinomio.

6. Considere el sistema de ecuaciones diferenciales no lineales (que Lorenz introdujo como simplificación del flujo de fluido en la atmósfera en 1963)

$$\begin{aligned} \dot{x} &= -10x + 10y, \\ \dot{y} &= 28x - y - xz, \\ \dot{z} &= -\frac{8}{3}z + xy, \end{aligned}$$

donde el punto indica derivada respecto del tiempo y aplique el método de Euler hacia adelante para resolverlo (sustituya $\dot{x}$ por $(x_{n+1} - x_n)/\delta t$. Dibuje la solución ($x(t)$, $y(t)$ y $z(t)$) que obtiene para dos pasos de tiempo $\delta t = 0{,}1$ y $\delta t = 0{,}2$ y las mismas condiciones iniciales (las que quiera). ¿Por qué los resultados son distintos? Linearice el problema y determine si es estable, como ecuación diferencial o no. Si el problema es inestable, ¿es inestable el método numérico? Para un paso temporal (δt) suficientemente pequeño, ¿las soluciones de la ecuación diferencial son estables y dependen continuamente de las condiciones iniciales? Dibuje la curva de la solución en el espacio xyz (utilice funciones de Matlab para dibujar curvas en 3D). El resultado que observa se denomina atractor extraño de Lorenz. ¿Cómo cambia dicha figura cuando cambia las condiciones iniciales?