Universidad de Huelva

Departamento de Matemáticas

Análisis Numérico I

Ingeniería Técnica Informática.

Curso académico 2009/10

Tema 1

Análisis del error

Referencias básicas: Burden y Faires [1985, Cap. 1]; Conte y de Boor [1985, Cap.1]; Henrici [1972, Cap. 15 y 16]; Henrici [1982, Cap. 1]; Kincaid y Cheney [1994, Cap. 2]; Stoer y Bulirsch [1980, Cap. 1].

PROBLEMA 1-1: Se trata de calcular, para un $n \in \mathbb{N}$ dado,

$$S_n = 1 + \sum_{k=1}^{n} \frac{1}{k^2 + k},$$

(puesto que $\frac{1}{k^2+k} = \frac{1}{k} - \frac{1}{k+1}$, se sigue que $S_n = 2 - \frac{1}{n+1}$).
(1) Utilizar los algoritmos:

(Alg_1) $\quad S_0 := 1, \quad S_k := S_{k-1} + \frac{1}{k^2+k}, \quad 1 \le k \le n,$

(Alg_2) $\quad D_0 := 0, \quad D_k := D_{k-1} + \frac{1}{(n+2-k)(n+1-k)}, \quad 1 \le k \le n, \quad S_n = D_n + 1,$

para calcular $S_9, S_{99}, \cdots, S_{99999}$.
¿Qué algoritmo da mejores resultados?

PROBLEMA 1-2: Se trata de evaluar un polinomio. Sea $f(x) = 6x^4 - 2x^3 + x^2 - x + 2$, un polinomio que también se puede escribir de la forma, $f(x) = (((6x - 2)x + 1)x - 1)x + 2$, ¿ qué algoritmo de evaluación de f es más eficiente desde el punto de vista del número de operaciones? Para ello construye una tabla con el número de operaciones necesarias para calcular $f(x)$ mediante ambos algoritmos en $x = 0, 0.5, 1.0, 1.5, \ldots, 10$. Utilizar el algoritmo generado para evaluar $f'(x)$.

PROBLEMA 1-3: Se trata de hallar las soluciones de la ecuación: $x^2 - 168467x + 2 = 0$.
(1) Utilizar: $x_1 = \frac{168467+\sqrt{168467^2-8}}{2}$ y $x_2 = \frac{168467-\sqrt{168467^2-8}}{2}$. ¿ Se produce cancelación en x_2?
(2) Racionalizar para calcular x_2, y con ello calcular x_2 con más cifras significativas.
(3) Calcular x_2 de forma que $x_2 = \frac{2}{x_1}$. ¿Es éste cálculo más preciso que el de los apartados anteriores?

PROBLEMA 1-4: Sean $P_k(r_k, \phi_k)$ las coordenadas polares del punto P_k. Es concocido que la distancia del punto P_k al punto P_{k-1} viene dada por:

(Alg_3) $\quad d(P_{k-1}, P_k) = \sqrt{r_{k-1}^2 + r_k^2 - 2r_{k-1}r_k\, cos(\phi_k - \phi_{k-1})}$,

o de forma equivalente, utilizando la igualdad: $cos(\phi_k - \phi_{k-1}) = 1 - 2\left[sen\left(\frac{1}{2}(\phi_k - \phi_{k-1})\right)\right]^2$,

(Alg_4) $\quad d(P_{k-1}, P_k) = \sqrt{(r_{k-1} - r_k)^2 + 4r_{k-1}r_k\left[sen\left(\frac{1}{2}(\phi_k - \phi_{k-1})\right)\right]^2}$.

(1) El segmento AB, con $A(1,0)$ y $B(1,1)$, es la unión de los segmentos $P_{k-1}P_k$ con $P_k(r_k, \phi_k)$ donde $\phi_k = \frac{\pi k}{4n}$ y $r_k = \frac{1}{cos(\phi_k)}$ para $1 \leq k \leq n$. Por tanto

$$\sum_{k=1}^{n} d(P_{k-1}, P_k) = 1.$$

Usar los algoritmos (Alg_3) y (Alg_4) para calcular la suma anterior para $n = 2, 2^2, \cdots, 2^{12}$. ¿Se produce cancelación ? ¿Qué algoritmo es más estable numéricamente?

PROBLEMA 1-5: Se trata de calcular algunos elementos de la sucesión $p_n = (1/3)^n$, para ello definimos los siguientes esquemas:

(1)
$$\begin{cases} p_0 = 1, \\ p_n = \frac{1}{3}p_{n-1}, \quad n \geq 1. \end{cases}$$

(2)
$$\begin{cases} p_0 = 1,\ p_1 = \frac{1}{3}, \\ p_n = \frac{10}{3}p_{n-1} - p_{n-2}, \quad n \geq 2. \end{cases}$$

(3)
$$\begin{cases} p_0 = 1,\ p_1 = \frac{1}{3}, \\ p_n = \frac{5}{6}p_{n-1} - \frac{1}{6}p_{n-2}, \quad n \geq 2. \end{cases}$$

(4)
$$\begin{cases} p_0 = 1,\ p_1 = \frac{1}{3}, \\ p_n = \frac{5}{3}p_{n-1} - \frac{4}{9}p_{n-2}, \quad n \geq 2. \end{cases}$$

¿Qué esquema es numéricamente más estable?

PROBLEMA 1-6: Se trata de calcular $e^{-x} = \sum_{k=0}^{\infty}(-1)^k \frac{x^k}{k!}$ para $x \gg 0$ (*smearing*= error cometido al evaluar una suma con una suma parcial muy grande comparada con el resultado final). Llamando $a_k = \frac{x^k}{k!}$, se tiene que $a_0 = 1$ y $a_k = a_{k-1}\frac{x}{k}$ para $k \geq 1$.
(1) Calcular $e^{-x} = \sum_{k\geq 0}(-1)^k a_k$, para x = 5,10,15,20,25 y 30. Tomar como criterio de paro $|a_k| < 10^{-6}$.
(2) Comparar los resultados anteriores con los valores exactos. ¿Se produce *smearing* en el algoritmo anterior.?

SOLUCIÓN PROBLEMA 1-1:

Notemos que utilizando el algoritmo:

$$(Alg_1) \qquad S_n = 1 + \frac{1}{1(1+1)} + \frac{1}{2(2+1)} + \cdots + \frac{1}{k(k+1)} + \cdots + \frac{1}{n(n+1)},$$

$$(Alg_2) \qquad S_n = \frac{1}{(n+1)n} + \frac{1}{n(n-1)} + \cdots + \frac{1}{(n+2-k)(n+1-k)} + \cdots + \frac{1}{2} + 1.$$

El fichero problema1tema1.m nos permite responder a la pregunta planteada.

```
function problema1tema1(n)
%               Datos de entrada
% n es un vector con el numero de sumandos de la suma parcial
%
for h=1:length(n)
s=1; d=0;
for k=1:n(h)
d=d+1/((n(h)+2-k)*(n(h)+1-k));
s=s+1/(k*(k+1));
end
Sumamayormenor(h)=s; % s es el valor de la suma parcial por el algoritmo 1
Sumamenormayor(h)=d+1; % d es el valor de la suma parcial por el algoritmo 2
end
format long
disp('Sumando de mayor a menor se obtiene:')
Sumamayormenor'
disp('Sumando de menor a mayor se obtiene:')
Sumamenormayor'
```

RESULTADO

▷ (Alg_1) Sumando de mayor a menor:

n	S_n
9	1.90000000000000
99	1.99000000000000
999	1.99900000000000
9999	1.99990000000000
99999	1.99998999999998

(Alg_2) Sumando de menor a mayor:

n	S_n
9	1.90000000000000
99	1.99000000000000
999	1.99900000000000
9999	1.99990000000000
99999	1.99999000000000

◁

Los valores obtenidos mediante el algoritmo (Alg_2) son los mismos que resultan al calcular S_n mediante la fórmula $S_n = 2 - \frac{1}{n+1}$.

SOLUCIÓN PROBLEMA 1-2:

Generamos el fichero problema2tema1.m siguiente:

```
function s=problema2tema1(x,P)
%              Datos de entrada
% P es un vector que contiene los coeficientes del polinomio f(x) en
% sentido ascendente, es decir, P(1)=a_0, P(2)=a_1,...
% x es un vector que contiene los valores donde vamos a calcular f(x)
%              Datos de salida
%    En versiones anteriores de Matlab exitia la orden flops que contaba
%    el numero de operaciones aritmeticas.
%    Esta orden ya no existe por lo que utilizaremos la orden cputime que
%    cuenta los segundos de utilizacion de cpu.
% s es un vector con dos elementos, el primero es el tiempo necesario
%   para evaluar f(x) por el algoritmo clasico de sustituir x en el
%   polinomio f(x). El segundo dato es el tiempo necesario
%   para evaluar f(x) por el metodo de Horner


% n es el numero de puntos a evaluar
n=length(x);
% g es el numero de coeficientes de f(x)
g=length(P);
format long
t=cputime;
% metodo clasico el valor se almacena en el vector px1
    px1=P(1);
    for j=2:g
        px1=px1+P(j)*x.^(j-1);
    end
s(:,1)=cputime-t; t=cputime;
% Ahora vemos el tiempo con el metodo de Horner los valores se almacenan en
% px2
    px2=P(g);
    for j=g-1:-1:1
        px2=P(j)+x.*px2;
    end
s(:,2)=cputime-t;
% Ahora utilizamos el algoritmo para evaluar f'(x). Lo escribimos aparte
% para que no se incremente el tiempo de cpu
    px2=P(g);
    derpx2=P(g);
    for j=g-1:-1:2
        px2=P(j)+x.*px2;
        derpx2=px2+x.*derpx2;
    end
disp('Los valores que se obtienen son')
```

RESULTADO

Los resultados tomando los valores propuestos por el enunciado del problema son aproximadamente cero segundos para ambos métodos. Para que aparezcan valores significativos hemos tomado un millón de datos a calcular entre 0 y 10. Concretamente las ordenes de Matlab para las entradas son:

```
x=linspace(0,10,10^6);

P=[2,-1,1,-2,6];
```

▷ Los resultados aparecen en la tabla 1.1. ◁

tiempo por el método clásico	0.43750
tiempo por el método de Horner	0.09375

Cuadro 1.1: Resultados problema 1-2.

SOLUCIÓN PROBLEMA 1-3:

Generamos el fichero problema3tema1.m siguiente:

```
% Para nuestro proposito solo necesitamos un fichero de comandos
% Primero calculamos los valores de las raices tal y como aparecen en el
% apartado 1 del problema 1.3
x1=(168467+sqrt(168467^2-8))/2
x21=(168467-sqrt(168467^2-8))/2
% Ahora calculamos x2 racionalizando, es lo que se pide en el apartado 2
x22=4/(168467+sqrt(168467^2-8))
% Por ultimo el valor de x2=2/x1 pedido en el apartado 3
x23=2/x1
```

RESULTADO

▷ Como se puede apreciar por los resultados que aparecen en la tabla 1.2, existe cancelación al calcular el valor de x_2 por el método directo del apartado (1), no ocurre lo mismo para los valores hallados racionalizando o por el propuesto en el apartado 3. ◁

valor de x_1	1.684669999881283e+005
valor de x_2 calculado en el apartado 1	1.187175803352147e-005
valor de x_2 calculado en el apartado 2	1.187176123597463e-005
valor de x_2 calculado en el apartado 3	1.187176123597463e-005

Cuadro 1.2: Resultados problema 1-3.

SOLUCIÓN PROBLEMA 1-4:

Generamos el fichero problema4tema1.m siguiente:

```
function s=problema4tema1nuevo(m)
%              Datos de entrada
% m es el exponente de la potencia de 2
%              Datos de salida
% s es un vector cuya primera columna es el valor de la suma calculada por el algoritmo
% tres y la segunda columna es el resultado de la suma por el algoritmo cuatro.
%
for h=1:m
n=2^h;
s=0; d=0;
for k=1:n
    fia=pi*(k-1)/(4*n);
    ra=1/cos(fia);
    fi=pi*k/(4*n);
    r=1/cos(fi);
    s=s+sqrt(ra^2+r^2-2*ra*r*cos(fi-fia));
    d=d+sqrt((r-ra)^2+4*ra*r*(sin(1/2*(fi-fia)))^2);
end
z(h)=s;
zz(h)=d;
end
format long
s=[z' zz'];
```

RESULTADO

▷ Los resultados, para $m = 12$, aparecen en la tabla 1.3. En ella se observa que se produce cancelación en el algoritmo (Alg_3) y que el algoritmo (Alg_4) es estable. ◁

n	(Alg_3)	(Alg_4)
2^1	1.00000000000000	1.00000000000000
2^2	1.00000000000000	1.00000000000000
2^3	1.00000000000000	1.00000000000000
2^4	1.00000000000000	1.00000000000000
2^5	1.00000000000001	1.00000000000000
2^6	1.00000000000026	1.00000000000000
2^7	1.00000000000058	1.00000000000000
2^8	0.99999999999753	1.00000000000000
2^9	1.00000000001007	1.00000000000000
2^{10}	0.99999999998506	1.00000000000000
2^{11}	0.99999999978241	1.00000000000000
2^{12}	0.99999999917042	1.00000000000000

Cuadro 1.3: Resultados problema 1-4.

SOLUCIÓN PROBLEMA 1-5:

Generamos el fichero problema5tema1.m siguiente:

```
function p=problema5tema1(n)
%              Datos de entrada
% n es el numero de elementos calculados en cada esquema
%              Datos de salida
% p es un vector cuya primera columna son los valores de la sucesion
% para el primer esquema, la segunda para el segundo esquema y asi
% sucesivamente.
%
% valores iniciales para el primer valor p0 y cualquiera de los 4 metodos
p(1,1:4)=1;
% bucle para el calculo de valores en el primer apartado
for k=2:n
    p(k,1)=p(k-1,1)/3;
end
% valores iniciales p1 en los tres ultimos apartados
p(2,2:4)=1/3;
% bucle para el calculo de valores en el segundo apartado
for k=3:n
    p(k,2)=10*p(k-1,2)/3-p(k-2,2);
end
% bucle para el calculo de valores en el tercer apartado
for k=3:n
    p(k,3)=5*p(k-1,3)/6-p(k-2,3)/6;
end
% bucle para el calculo de valores en el cuarto apartado
for k=3:n
    p(k,4)=5*p(k-1,4)/3-4*p(k-2,4)/9;
end
```

RESULTADO

▷ De los resultados, para $n = 19$, que aparecen en la tabla 1.4 se deduce que los esquemas 1, 3 y 4 son estables y el esquema 2 no lo es. Para un estudio más profundo de la estabilidad de dichos esquemas hemos de hallar el máximo en valor absoluto de los autovalores de los sistemas en diferencias que resultan en cada uno de los esquemas. ◁

	Esquema 1	Esquema 2	Esquema 3	Esquema 4
p_0	1.00000000000000	1.00000000000000	1.00000000000000	1.00000000000000
p_1	0.33333333333333	0.33333333333333	0.33333333333333	0.33333333333333
p_2	0.11111111111111	0.11111111111111	0.11111111111111	0.11111111111111
p_3	0.03703703703704	0.03703703703704	0.03703703703704	0.03703703703704
p_4	0.01234567901235	0.01234567901234	0.01234567901235	0.01234567901235
p_5	0.00411522633745	0.00411522633744	0.00411522633745	0.00411522633745
p_6	0.00137174211248	0.00137174211247	0.00137174211248	0.00137174211248
p_7	0.00045724737083	0.00045724737078	0.00045724737083	0.00045724737083
p_8	0.00015241579028	0.00015241579014	0.00015241579028	0.00015241579028
p_9	0.00005080526343	0.00005080526302	0.00005080526343	0.00005080526342
p_{10}	0.00001693508781	0.00001693508658	0.00001693508781	0.00001693508781
p_{11}	0.00000564502927	0.00000564502559	0.00000564502927	0.00000564502927
p_{12}	0.00000188167642	0.00000188166539	0.00000188167642	0.00000188167642
p_{13}	0.00000062722547	0.00000062719239	0.00000062722547	0.00000062722547
p_{14}	0.00000020907516	0.00000020897590	0.00000020907516	0.00000020907516
p_{15}	0.00000006969172	0.00000006939394	0.00000006969172	0.00000006969172
p_{16}	0.00000002323057	0.00000002233725	0.00000002323057	0.00000002323057
p_{17}	0.00000000774352	0.00000000506355	0.00000000774352	0.00000000774352
p_{18}	0.00000000258117	-0.00000000545874	0.00000000258117	0.00000000258117
p_{19}	0.00000000086039	-0.00000002325934	0.00000000086039	0.00000000086038

Cuadro 1.4: Resultados problema 1-5.

SOLUCIÓN PROBLEMA 1-6:

Para resolverlo creamos el fichero problema6tema1.m siguiente:

```
function problema6tema1
%
for x=5:5:30
    a=1;s=1;
    for k=1:100000
        a=a*x/k;
        s=s+(-1)^k*a;
        if abs(a)<10^(-6)
           format long e
           sol=[x',s',exp(-x)',abs(s-exp(-x)')]
           break
        end
    end
end
```

RESULTADO

▷ Los resultados obtenidos al ejecutar dicho programa aparecen recogidos en la tabla 1.5. De ella se deduce, observando la columna del error, que se produce *smearing* en el algoritmo propuesto ◁

x	Suma serie	e^{-x}	Error
5	6.737867030659049e-03	6.737946999085467e-03	7.996842641810192e-08
10	4.524791416064885e-05	4.539992976248485e-05	1.520156018359933e-07
15	1.674038770543654e-07	3.059023205018258e-07	1.384984434474604e-07
20	-9.868659851254508e-08	2.061153622438558e-09	1.007477521349836e-07
25	-4.800527568587555e-07	1.388794386496402e-11	4.800666448026205e-07
30	-3.051378836242318e-05	9.357622968840175e-14	3.051378845599941e-05

Cuadro 1.5: Resultados problema 1-6.