EJERCICIO COMPUTACIONAL Nº 1. PRELIMINARES

Ángel Durán

Departamento de Matemática Aplicada

Universidad de Valladolid

12 de febrero de 2011

Contenidos

Contenidos

Contenidos

Números enteros

Tipos de variables enteras: en función del número de bits usados para su almacenamiento

- Un entero con signo y n bits toma valores en $[-2^{n-1}, 2^{n-1} - 1]$.
- Un entero sin signo y n bits toma valores en $[0, 2^n - 1]$.

`y=int8(x)`	a 8-bit signed integer in $[-2^7, 2^7 - 1] = [-128, 127]$
`y=uint8(x)`	a 8-bit unsigned integer in $[0, 2^8 - 1] = [0, 255]$
`y=int16(x)`	a 16-bit signed integer in $[-2^{15}, 2^{15} - 1] = [-32768, 32767]$
`y=uint16(x)`	a 16-bit unsigned integer in $[0, 2^{16} - 1] = [0, 65535]$
`y=int32(x)`	a 32-bit signed integer in $[-2^{31}, 2^{31} - 1] = [-2147483648, 2147483647]$
`y=uint32(x)`	a 32-bit unsigned integer in $[0, 2^{32} - 1] = [0, 4294967295]$

Figure 19: Scilab integer functions.

Números enteros

$--> \text{format}(25)$

$--> n = 32; 2 ** n - 1$

ans

$= 4294967295.$

$--> i = uint32(0)$

$i = 0$

$--> j = i - 1$

$j = 4294967295$

$--> k = j + 1$

$k = 0$

Números enteros

Conversión entre números enteros

iconvert	conversion to integer representation
inttype	type of integers

Figure 20: Scilab integer conversion functions.

$$
\begin{array}{c|c}
-->i=int8(1) & -->inttype(j) \\
i=1 & ans=2. \\
-->inttype(i) & k=i+j \\
ans=1. & k=3 \\
-->j=int16(2) & -->inttype(k) \\
j=2 & ans=2.
\end{array}
$$

Números enteros

`inttype(x)`	Type
1	8-bit signed integer
2	16-bit signed integer
4	32-bit signed integer
11	8-bit unsigned integer
12	16-bit unsigned integer
14	32-bit unsigned integer

Figure 21: Types of integers returned by the `inttype` function.

$$-->b = int32([1 \quad -120 \quad 127 \quad 312])$$
$$-->y = iconvert(b, 1)$$
$$-->inttype(y)$$
$$ans = 1$$

Números enteros

Bases de representación

base2dec	bin2dec	dec2bin
dec2hex	dec2oct	hex2dec
oct2dec		

```
--> str('1010110')
  y = bin2dec(str)
--> y =
  86
```

```
--> base2dec(['ABC',' 0'], 16)
  ans = .
--> 2748.  0.
```

```
--> dec2oct([2748  10; 11  3])
  ans = .
--> 5274  12
  13  3
```

Punto flotante

ceil	floor	nearfloat
double	frexp	round
fix	int	

- Por defecto, las variables vienen representadas por números en punto flotante de 64 bits (doble precisión)
- Puede haber problemas si se alamcenan números enteros como de punto flotante

Punto flotante

$$-->A=testmatrix('' hilb'' ,2)$$
$$A=$$
$$4.\ -\ 6.$$
$$-6{,}12.$$
$$-->i=2$$
$$i=2.$$
$$-->j=1$$
$$j=1.$$
$$-->A(i,j)$$
$$ans=-6.$$
$$-->A(2,[1{,}01{,}11{,}51{,}9])$$
$$ans=$$
$$-6.\ -\ 6.\ -\ 6.\ -\ 6.$$

Punto flotante

double convierte un entero en un número en coma flotante

$-->x = int8([0\ \ 12\ \ 140])$

$x =$

$0\ \ 12\ \ -116$

$-->double(x)$

$ans = 0.\ \ 12.\ \ -116.$

comandos de redondeo

$$\begin{array}{rl} floor(x) = n & \text{si } n \leq x < n+1 \\ ceil(x) = n & \text{si } n-1 < x \leq n \\ int(x) = & floor(x) \text{ si } x \geq 0 \\ & ceil(x) \text{ si } x < 0 \end{array}$$

$-->floor(10,6)$ $ans = 10.$

$-->floor(10,6)$ $ans = -11.$

$-->ceil(10,6)$ $ans = 11.$

$-->ceil(-10,6)$ $ans = -10.$

$-->int(10,6)$ $ans = 10.$

$-->int(-10,6)$ $ans = -10.$

Algoritmo de Horner

```
function [nop,y]=horner1(a,c)
// Implementación del algoritmo de Horner
// Variables de entrada
// a = (a(1), ..., a(M)): coeficientes del polinomio
// c: punto de evaluación
// Variables de salida
// y: Valor del polinomio evaluado en x = c
// nop: contador de número de operaciones
M = length(a); nop = 0;
for k = M - 1 : -1 : 1
      a_k = a_k + c * a_{k+1};
      nop = nop + 1;
end
y = a(1);
endfunction
```

Algoritmo de Horner

$-->exec('F:\ldots\horner1.sci',-1)$
$-->a=[1\ \ 1\ \ 1\ \ 1];c=1;$
$-->[nop,y]=horner1(a,c)$
$y=$
4.
$nop=$
3.

Variante (como ejercicio): evaluación en $p \geq 1$ puntos
$c=(c(1),\ldots,c(p))$

Algoritmo de Horner

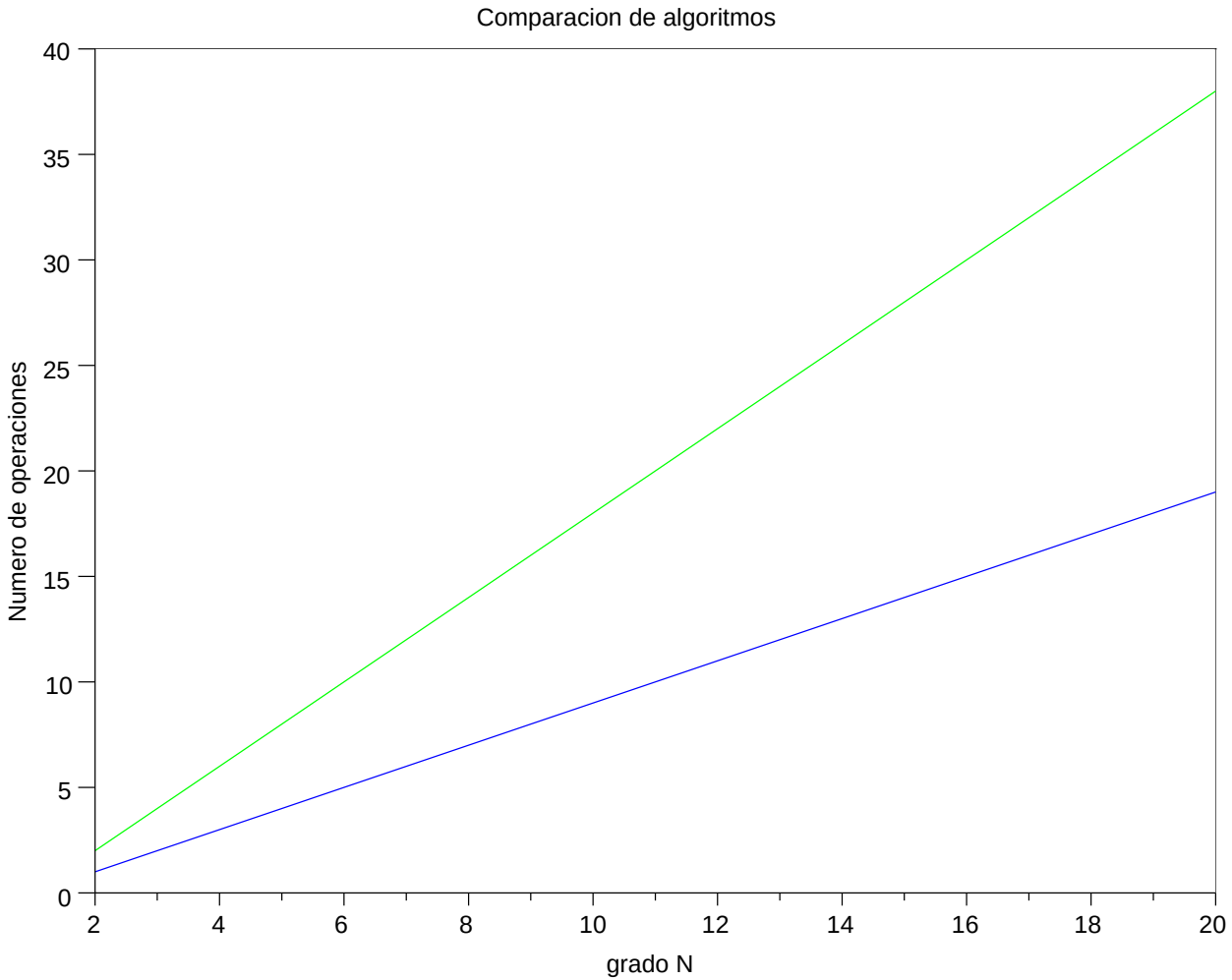
Comparación de algoritmos

```
function [nop2,y2]=evaluacion(a,c)
//Evaluación elemental de un polinomio
s = 1; y2 = a(1); nop2 = 0;
M = length(a);
for k = 2 : M
      s = s * c;
      y2 = y2 + a(k) * s;
      nop2 = nop2 + 2;
end
endfunction
```

Algoritmo de Horner

$$-->exec('F:\ldots\evaluacion.sci',-1)$$
$$-->a=[1\ \ 1\ \ 1\ \ 1];c=1;$$
$$-->[nop,y]=evaluacion(a,c)$$
$$y=$$
4.
$$nop=$$
6.

Algoritmo de Horner

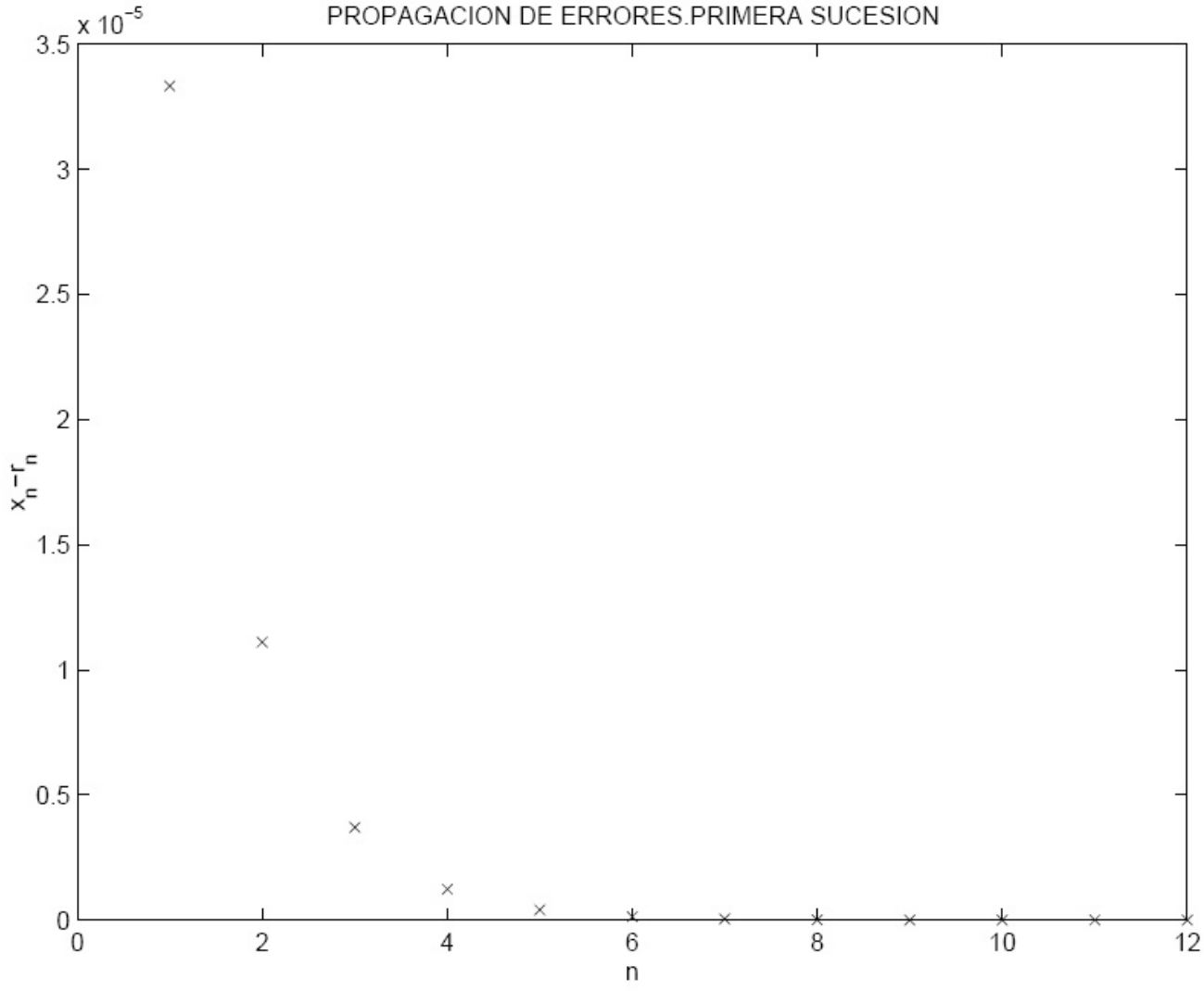


Errores y propagación

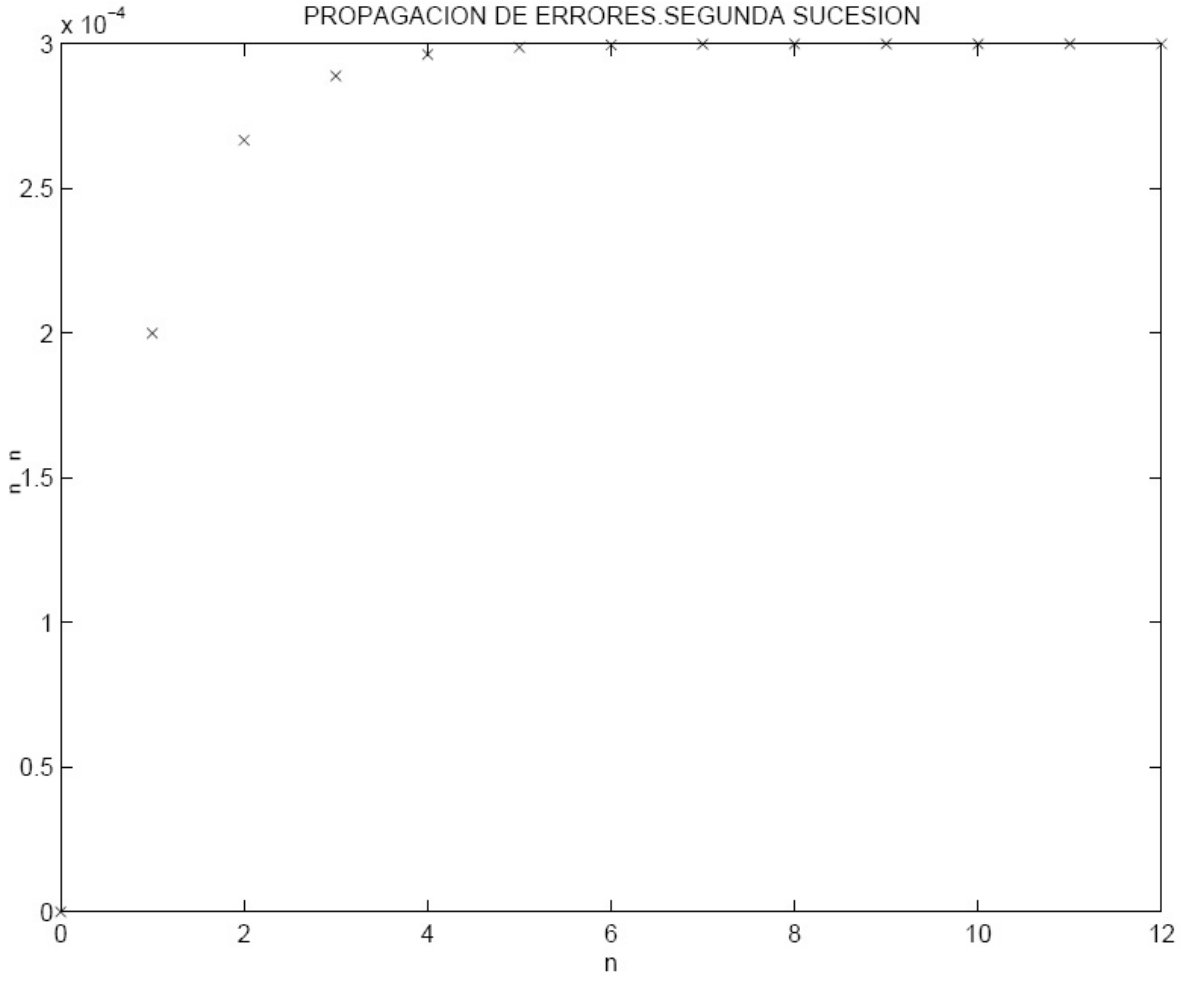
Ejemplo 1.8: Generación de la sucesión $\{\frac{1}{3^n}\}$

```
//Generacion de la sucesión {1/3^n}
//Inicialización
r(1) = 1; p(1) = 1; q(1) = 1;
p(2) = 1/3; q(2) = 1/3; r(2) = r(1)/3;
N = 20;
for k = 3 : N
      r(k) = r(k − 1)/3;
      p(k) = 4 ∗ p(k − 1)/3 − p(k − 2)/3;
      q(k) = 10 ∗ q(k − 1)/3 − q(k − 2);
end
for k = 1 : N
      x(k) = (1/3) ∗ ∗(k − 1);
end
```

Errores y propagación



Errores y propagación



Errores y propagación