1. Crear una función en Scilab que calcule en forma robusta las raíces de una ecuación cuadrática con discriminante positivo. Usar dicha función para evaluar la raíz positiva de la ecuación cuadrática con $\epsilon = 0,0001$ y estimar su error.

Solución

```
function r = raices(polinomio)
    a = coeff(p, 2)
    b = coeff(p, 1)
    c = coeff(p, 0)
    delta = b^2 - 4*a*c
    if b < 0 then
        x1 = (2*c) / (-b + sqrt(delta))
        x2 = (-b + sqrt(delta)) / (2*a)
    else
        x1 = (-b - sqrt(delta)) / (2*a)
        x2 = (2*c) / (-b - sqrt(delta))
    end
    r = [x1; x2]
endfunction
function r = resolvente(p)
    a = coeff(p, 2)
    b = coeff(p, 1)
    c = coeff(p, 0)
    delta = b^2 - 4*a*c
    x1 = (-b - sqrt(delta)) / 2*a
    x2 = (-b + sqrt(delta)) / 2*a
    r = [x1; x2]
endfunction
epsilon = 0.0001;
a = epsilon;
b = 1 / epsilon;
c = -epsilon;
p = poly([c b a], "x", "coeff");
assert_checkequal(raices(p), roots(p));
error1 = abs(raices(p)(1) - resolvente(p)(1))
error2 = abs(raices(p)(1) - resolvente(p)(1))
```

2. Usando aritmética de cuatro dígitos de precisión (mantiza decimal de 4 dígitos con redondeo), sume la siguiente expresión

$$0,9222 \cdot 10^4 + 0,9123 \cdot 10^3 + 0,3244 \cdot 10^3 + 0,2849 \cdot 10^3$$

tanto ordenando los números de mayor a menor (en valor absoluto), como de menor a mayor. Realiza cada operación de forma separada, primero igualando exponentes y luego normalizando el resultado en cada paso. ¿Cuál de las dos posibilidades es más exacta? Justifique los resultados que encuentre.

Solución

(a) $0,9222 \cdot 10^4 + 0,9123 \cdot 10^3 + 0,3244 \cdot 10^3 + 0,2849 \cdot 10^3$:

- $0,9123 \cdot 10^3 = 0,09123 \cdot 10^4 \rightarrow 0,0912 \cdot 10^4$
- $0,9222 \cdot 10^4 + 0,0912 \cdot 10^4 = 1,0134 \cdot 10^4 = 0,10134 \cdot 10^5 \rightarrow 0,1013 \cdot 10^5$
- $0,3244 \cdot 10^3 = 0,03244 \cdot 10^4 \rightarrow 0,0324 \cdot 10^4 = 0,00324 \cdot 10^5 \rightarrow 0,0032 \cdot 10^5$
- $0,1013 \cdot 10^5 + 0,0032 \cdot 10^5 = 0,1045 \cdot 10^5$
- $0,2849 \cdot 10^3 = 0,02849 \cdot 10^4 \rightarrow 0,0285 \cdot 10^4 = 0,00285 \cdot 10^5 \rightarrow 0,0029 \cdot 10^5$
- $0,1045 \cdot 10^5 + 0,0029 \cdot 10^5 = 0,1074 \cdot 10^5 = 10740$

(b) $0,2849 \cdot 10^3 + 0,3244 \cdot 10^3 + 0,9123 \cdot 10^3 + 0,9222 \cdot 10^4$:

- $0,2849 \cdot 10^3 + 0,3244 \cdot 10^3 = 0,6093 \cdot 10^3$
- $0,6093 \cdot 10^3 + 0,9123 \cdot 10^3 = 1,5216 \cdot 10^3 = 0,15216 \cdot 10^4 \rightarrow 0,1522 \cdot 10^4$
- $0,1522 \cdot 10^4 + 0,9222 \cdot 10^4 = 1,0744 \cdot 10^4 = 0,10744 \cdot 10^5 \rightarrow 0,1074 \cdot 10^5 = 10740$

(c)

- $0,9222 \cdot 10^4 + 0,9123 \cdot 10^3 + 0,3244 \cdot 10^3 + 0,2849 \cdot 10^3 = 10743,6$
- $|10743,6 - 10740| = 3,6 = |10743,6 - 10740|$

3. El algoritmo de Horner se usa para evaluar de forma eficiente funciones polinómicas. Dado un polinomio $p(x) = a_0 + a_1 x + \cdots + a_n x^n$ a coeficientes reales, se genera una secuencia de constantes dadas por:

$$\begin{aligned} b_n &= a_n \\ b_i &= a_i + x_0 b_{i+1} \ (i = n-1, \ldots, 1, 0) \end{aligned}$$

donde $b_0 = p(x_0)$.

(a) Mostrar que dado un x_0, $b_0 = p(x_0)$.

(b) Implementar el algoritmo de Horner para calcular $p(x_0)$ en Scilab.

(c) Dado $q(x) = b_1 + b_2 x + \cdots + b_n x^{n-1}$, mostrar que $p'(x_0) = q(x_0)$. Verificar que $p(x) = b_0 + (x - x_0) q(x)$.

(d) Implementar una generalización del algoritmo horner dado en Scilab, de tal forma que se pueda calcular $p(x_0)$ y $p'(x_0)$ al mismo tiempo.

Soluciones

(a) Observemos que $b_i = a_i + x_0 b_{i+1} \iff a_i = b_i - x_0 b_{i+1}$, luego:

$$\begin{aligned} p(x_0) &= \sum_{i=0}^{n} a_i x_0^i = a_n x_0^n + \sum_{i=0}^{n-1} a_i x_0^i = b_n x_0^n + \sum_{i=0}^{n-1} (b_i - x_0 b_{i+1}) x_0^i = \\ &= b_n x_0^n + \sum_{i=0}^{n-1} \left(b_i x_0^i - x_0 b_{i+1} x_0^i \right) = b_n x_0^n + \sum_{i=0}^{n-1} \left(b_i x_0^i - b_{i+1} x_0^{i+1} \right) = \\ &= b_n x_0^n + b_0 - b_n x_0^n \text{ (propiedad telescópica)} \\ &= b_0 \end{aligned}$$

(b)

```
function bi = aux(p, x0, i)
    n = degree(p)
    if i == n then
        bi = coeff(p, n)
    else
        ai = coeff(p, i)
        bi = ai + x0 * aux(p, x0, i + 1)
    end
endfunction

function b0 = calcular(p, x0)
    b0 = aux(p, x0, 0)
endfunction

p = poly([3 2 1], "x", "coeff");
assert_checkequal(calcular(p, 4), horner(p, 4));
```

(c) COMPLETAR.

(d) COMPLETAR.

4. Implementar en Scilab la función `derivar` que toma una función f, un valor v, un orden n y un paso h y retorna el valor de evaluar la derivada de f de orden n en el punto v. Implementar usando cociente incremental y luego usando el comando numderivative implementado en Scilab.

   (a) ¿Cómo son los errores cometidos en cada caso?

   (b) ¿Qué hace que el error en la implementación por cociente incremental crezca?

Soluciones

```
function y = derivar(f, v, n, h)
    if n == 0 then
        y = f(v)
    else
        d0 = f
        for i = 1:n
            argumento = "(x)"
            incremento = "(x + " + string(h) + ")"
            nombre = "d" + string(i)
            anterior = "d" + string(i - 1)
            cuerpo = "y = (" + anterior + incremento
            cuerpo = cuerpo + "-" + anterior + argumento
            cuerpo = cuerpo + ") /" + string(h)
            deff("y = " + nombre + "(x)", cuerpo)
        end
        deff("y = dn" + "(x)", cuerpo)
        y = dn(v);
    end
endfunction
```

(a) COMPLETAR.

(b) COMPLETAR.

5. Implementar en Scilab una función taylor que calcule el valor de un polinomio de Taylor de grado n de una función f en un punto dado v.

Solución COMPLETAR.

6. COMPLETAR.

(a) COMPLETAR.

(b) COMPLETAR.

Soluciones

(a) COMPLETAR.

(b) COMPLETAR.