Uma série de Taylor chata de somar

Nelson L. Dias

3 de fevereiro de 2018

Trabalho Computacional 3 para a Disciplina TEA-010 “Matemática Aplicada I”

Curso de Graduação de Engenharia Ambiental

Prof. Nelson Luís Dias

Atenção: este trabalho não deve ser entregue: ele faz parte da matéria, e seu conteúdo será cobrado nas provas parciais e na prova final.

Dada a função

$$f(x) = \mathrm{e}^{-x}\operatorname{sen}(x),$$

os 20 primeiros termos de sua série de Taylor em torno de $x = 0$ são

$$f(x) = x - x^2 + \frac{x^3}{3} - \frac{x^5}{30} + \frac{x^6}{90} - \frac{x^7}{630} + \frac{x^9}{22680} - \frac{x^{10}}{113400} + \frac{x^{11}}{1247400} - \frac{x^{13}}{97297200} + \frac{x^{14}}{681080400} - \frac{x^{15}}{10216206000} + \frac{x^{17}}{1389404016000} - \frac{x^{18}}{12504636144000} + \frac{x^{19}}{237588086736000} + \cdots$$

É difícil identificar uma lei de formação. Por outro lado,

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!} x^n,$$

e as derivadas $f^{(n)}(0)$ elas mesmas talvez sejam um pouco melhores. Uma lista das derivadas é a seguinte:

n	$n \mod 4$	$f^{(n)}(0)$
0	0	0
1	1	1
2	2	−2
3	3	2
4	0	0
5	1	−4
6	2	8
7	3	−8
8	0	0
9	1	16
10	2	−32
11	3	32
12	0	0
13	1	−64
14	2	128
15	3	−128

Note que os 0's se repetem a cada 4 linhas. O resto da divisão de n por 4, $n \mod 4$, parece ser a chave de um algoritmo para calcular $f^{(n)}(0)$:

$n \mod 4 = 0$:

$$f^{(n)}(0) = 0.$$

$n \mod 4 = 1$: $f^{(n)}(0) = 1, 4, 16, 64$ para $n = 1, 5, 9, 13$. Uma "lei" de formação parece ser a seguinte: seja $p = n$ div 4, *onde* div *significa divisão inteira*; então, $p = 0, 1, 2, 3$ e

$$f^{(n)}(0) = 4^p.$$

$n \mod 4 = 2$: $f^{(n)}(0) = -2, 8, -32, 128$ para $n = 2, 6, 10, 14$. A lei de formação é a seguinte: $p = n$ div 4; $p = 0, 1, 2, 3$; $q = n$ div 2; então, $q = 1, 3, 5, 7$, e

$$f^{(n)}(0) = (-1)^{p+1} 2^q.$$

$n \mod 4 = 3$: $f^{(n)}(0) = 2, -8, 32, -128$ para $n = 3, 7, 11, 15$. A lei de formação é a seguinte: $p = n$ div 4; $p = 0, 1, 2, 3$; $q = n$ div 2; então, $q = 1, 3, 5, 7$, e

$$f^{(n)}(0) = (-1)^p 2^q.$$

Note que isso não é o mais eficiente computacionalmente que se pode fazer. A "melhor" alternativa computacional parece ser a seguinte:

```
der[0] ← 0 ;
der[1] ← 1 ;
for n = 2 to 15 do
    r = n mod 4 ;
    switch r do
        case r = 0 do
            der[n] ← 0
        end
        case r = 1 do
            der[n] = - 2*der[n-2] ;
        end
        case r = 2 do
            der[n] = -2*der[n-1]
        end
        case r = 3 do
            der[n] = -der[n-1]
        end
    end
end
```

É relativamente fácil e rápido implementar o algoritimo acima para uma lista de 16 elementos em Python, só para "ver" que o algoritmo funciona.

Agora, implemente um algoritmo *realmente* eficiente para calcular o valor de $f(x) = e^{-x}\operatorname{sen}(x)$ para um valor qualquer de x, da seguinte maneira:

1. Inicialize os dois primeiros valores da soma (0 e 1), para $n = 0, 1$.
2. Crie e atualize uma variável para o fatorial de n; a cada novo n, $n! = n \times (n-1)!$.
3. Crie e atualize uma variável para $f^{(n)}(0)$; como vimos, a atualização depende dos *dois últimos valores* de $f^{(n)}(0)$ (por exemplo, eles podem ser chamados de `der0` e `der1`).
4. Continue somando até que o próximo termo tenha um valor absoluto menor que uma determinada tolerância δ.

Obtenha os valores de $f(1)$, $f(2)$, $f(3)$ e $f(10)$ e compare com os retornados utilizando as funções `exp` e `sin`. Observe que você precisa de tolerâncias extremamente pequenas para que a série convirja! Que valores de δ você precisa impor para que a soma seja bem sucedida até $x = 3$, e $x = 10$?