N1

Szymon Pach

Program był pisany w pythonie 2.7.14 na systemie Windows 10

Program liczy sumę szeregu podanego w treści zadania.

Wyniki:

**I[0] = 3.9195505378**

**I[1] = -6.9388939039e-18**

**I[2] = -2.6331145808**

**I[3] = -4.1633363423e-17**

**I[4] = 1.7679769567**

**I[5] = -6.9388939039e-18**

**I[6] = -1.1857156457**

**I[7] = -2.2204460493e-16**

**I[8] = 0.79316883809**

**I[9] = 1.734723476e-17**

**I[10] = -0.52752904453**

**I[11] = 3.469446952e-18**

**I[12] = 0.34629389809**

**I[13] = 1.5959455979e-16**

**I[14] = -0.22046577534**

**I[15] = 7.9797279895e-17**

**I[16] = 0.12991217665**

**I[17] = 2.8102520311e-16**

**I[18] = -0.060144526229**

**I[19] = -4.1893571945e-16**

Dla N = 20, liczba obliczeń to **3 543 600**Dla N = 1 to **14**

Podczas liczenia zaimplementowałem obie wersje liczenia wielomianów Chebysheva, to jest wersję używającą rekurencji i wersję używającą funkcji matematycznych. Wersja rekurencyjna była szybsza dla początkowych wartości N, ale potem funkcje matematyczne (cos, acos) okazują się szybsze.

Program ma złożoność obliczeniowa **O(n).**

1. import math
2. pi = 3.14159265358979323846
3. N = 20
4. aritmCounter = 0
5. mathFuncCounter = 0
6. operatorCounter = 0.5
7. def x\_equation(n):
8. global aritmCounter
9. aritmCounter += 3
10. return (pi / N) \* (n + 0.5)
11. def t\_equation(k, x):
12. global aritmCounter
13. global mathFuncCounter
14. aritmCounter += 2
15. mathFuncCounter += 2\*30
16. return math.cos(k \* math.acos(x))
17. def t\_equation\_recur(k, x):
18. global aritmCounter
19. aritmCounter += 5
20. if k == 0:
21. return 1.0
22. elif k == 1:
23. return x
24. else:
25. return 2.0 \* x \* t\_equation\_recur(k-1, x) - t\_equation\_recur(k-2, x)
26. def sum\_equation(N = 20):
27. global aritmCounter
28. global mathFuncCounter
29. global operatorCounter
31. I\_k = []
32. sum = 0.0
33. for k in range(N):
34. for i in range(N):
35. aritmCounter += 3
36. mathFuncCounter += 90
37. operatorCounter += 1
38. temp = math.cos(x\_equation(i))
39. sum = sum + t\_equation\_recur(k, math.cos(x\_equation(i))) / (1.0 + 25.0 \* temp \* temp)
41. I\_k.append(sum)
42. sum = 0.0
43. for i in range(N):
44. print "I[{}] = {:.11}".format(i, I\_k[i])
46. sum\_equation()
47. print "\nLiczba obliczen: ", aritmCounter
48. print "Liczba funkcji: ", mathFuncCounter
49. print "Liczna obliczen w sumie: ", aritmCounter + mathFuncCounter + operatorCounter