Национальный исследовательский университет «МИЭТ»

Отчет о проделанной лабораторной работе №1

По предмету: Численные методы

На тему: Распространение ошибок в вычислительных процедурах

Выполнила Марина Алина

Группа ПИН-24

21.02.2021

Задание №1

Рассмотрим многочлен Уилкинсона



Очевидно, корнями его являются x1 = 1, x2 = 2, . . ., x20 = 20. Выполним в Matlab команду p=poly(1:20), которая позволяет получить коэффициенты полинома, корнями которого является аргумент функции. А затем наберем roots(p) и убедимся, что корни полинома найдены верно.

**clear**

**clc**

**disp('---task 1.1---')**

**p=poly(1:20);**

**roots(p)**

**Command window**

**---task 1.1---**

**ans =**

**19.9999**

**19.0013**

**17.9937**

**17.0185**

**15.9597**

**15.0593**

**13.9302**

**13.0627**

**11.9589**

**11.0225**

**9.9912**

**9.0027**

**7.9994**

**7.0001**

**6.0000**

**5.0000**

**4.0000**

**3.0000**

**2.0000**

**1.0000**

……………………………………………………………………………………..

Теперь изменим значение, например, второго коэффициента на малую величину 10−7 (составляет ≈ 5 · 10−8% от числа) и снова выполним эту команду - половина корней стала комплексными числами! Исходная задача оказалась неустойчивой к входным данным (их малое изменение ведет к сильному изменению решения), в результате чего получился абсурдный результат.

**clear**

**clc**

**disp('---task 1.2---')**

**p(2) = p(2) + 1e-7;**

**roots(p)**

**Command window**

**---task 1.2---**

**ans =**

**20.4220 + 0.9992i**

**20.4220 - 0.9992i**

**18.1572 + 2.4702i**

**18.1572 - 2.4702i**

**15.3147 + 2.6986i**

**15.3147 - 2.6986i**

**12.8459 + 2.0620i**

**12.8459 - 2.0620i**

**10.9207 + 1.1010i**

**10.9207 - 1.1010i**

**9.5762 + 0.0000i**

**9.1081 + 0.0000i**

**7.9946 + 0.0000i**

**7.0002 + 0.0000i**

**6.0000 + 0.0000i**

**5.0000 + 0.0000i**

**4.0000 + 0.0000i**

**3.0000 + 0.0000i**

**2.0000 + 0.0000i**

**1.0000 + 0.0000i**

**Вывод:** Уилкинсон выбрал возмущение 2 -23 , потому что его пилот ACE компьютер имел 30-бит с плавающей точкой significands . Мы же взяли возмущение в 10−7, неудивительно, что половина корней стала комплексными числами, ведь анализ возмущений показывает, что 30-битная точность коэффициента недостаточна для разделения корней полинома Уилкинсона.

……………………………………………………………………………………….

Задание №2

Рассмотрим представление числа в компьютере. Чаще всего в Matlab производятся операции над числами с двойной точностью. Число формата double — 64-разрядное число (8 байт), в котором 1 бит — знаковый, 52 отводятся под мантиссу и 11 под порядок числа, m = 0, m1m2 . . . mk, m1 6= 0 — мантисса числа, n — порядок). Так как в порядке тоже есть знаковый бит, то множество его значений лежит от . В командном окне Matlab наберём 2ˆ1023.

>>2^1023

ans =

8.9885e+307

Получим число с десятичным порядком +308 (его легко оценить аналитически:).

……………………………………………………………………………………..

Теперь выполним 2ˆ1024 — получили Inf т.е. машинную бесконечность. Команда realmax как раз и выдаёт максимальное число, которое можно представить в Matlab, команда realmin — минимальное (по абсолютной величине).

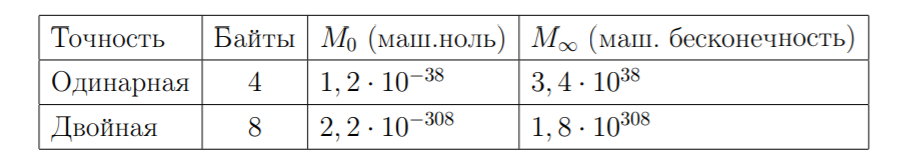
>> 2^1024

ans =

Inf

……………………………………………………………………………………….

Теперь посмотрим на мантиссу: , таким образом, мантисса содержит 15-16 десятичных знаков; все, что вылезет за эти пределы, будет отброшено. Установим формат отображения с плавающей точкой: format long e, выполним sqrt(2) (квадратный корень из 2) и посчитаем число выданных цифр.



>> format long e

>> sqrt(2)

ans =

1.414213562373095

>> % 16 цифр

Казалось бы, такая точность представления способна удовлетворить любые нужды исследователя. Однако, необходимо помнить, что, например, перед тем как два числа будут сложены они должны быть приведены к единому порядку, а то, что при сложении выйдет за мантиссу будет отброшено! Поэтому, если сложить 10ˆ8+10ˆ-7 в мантиссу уложатся 15 десятичных знаков и будет получен верный результат, а если выполнить 10ˆ8+10ˆ-8, то малое число выйдет за разрядную сетку, и получим те же 10ˆ8 . И мы сейчас это проверим

>> 100000000 + 1e-7

ans =

100000000.0000001

>> 100000000 + 1e-8

ans =

100000000

Получается парадоксальная ситуация: если прибавим к 1 малое число 10ˆ-16, но последовательно 10ˆ17 раз. Легко найти ответ 11, однако, машинная арифметика даёт 1 — Что является ошибкой.

a=1;

syms n

n = 10^17;

for i = 0:1:n

result = a+ 10^(-16);

end

result

Command window

result=1

Если бы мы начали с конца, т.е. сначала нашли сумму малых, а затем прибавили к единице, то получили бы верный результат. Таким образом, машинное сложение, в общем случае, не коммутативно. Знание этих фактов позволяет протестировать компьютер на погрешность вычислении.

……………………………………………………………………………………….

Задание №3

Найдем значение выражения , n = 0, 1, 2, . . .. С точки зрения аналитической математики значение этого выражения постоянно и равно единице для любых n. Попробуем сделать предположения, при каком значении n это выражение перестанет отличаться от единицы, чему будет равно; а также напишем программу, реализующую этот алгоритм с пошаговой выдачей номера шага, значения ε и результата выражения.

Подставим значение 2^(-10) в выражение , оно будет =1

Подставим значение 2^(-20) в выражение , оно будет =1

Подставим значение 2^(-30) в выражение , оно будет =1

Подставим значение 2^(-40) в выражение , оно будет =1

Подставим значение 2^(-50) в выражение , оно будет =1

Подставим значение 2^(-60) в выражение , оно будет уже = 0

Значит это произойдет в промежутке между 50 и 60

**syms n**

**n = 60;**

**for i = 0:1:n**

**eps = 2^-i;**

**result = (1 + eps - 1) / eps;**

**fprintf('%d: eps=%d; result=%d\n',i, eps, result);**

**end**

**Command window**

**0: eps=1; result=1**

**1: eps=5.000000e-01; result=1**

**2: eps=2.500000e-01; result=1**

**3: eps=1.250000e-01; result=1**

**4: eps=6.250000e-02; result=1**

**5: eps=3.125000e-02; result=1**

**6: eps=1.562500e-02; result=1**

**7: eps=7.812500e-03; result=1**

**8: eps=3.906250e-03; result=1**

**9: eps=1.953125e-03; result=1**

**10: eps=9.765625e-04; result=1**

**11: eps=4.882813e-04; result=1**

**12: eps=2.441406e-04; result=1**

**13: eps=1.220703e-04; result=1**

**14: eps=6.103516e-05; result=1**

**15: eps=3.051758e-05; result=1**

**16: eps=1.525879e-05; result=1**

**17: eps=7.629395e-06; result=1**

**18: eps=3.814697e-06; result=1**

**19: eps=1.907349e-06; result=1**

**20: eps=9.536743e-07; result=1**

**21: eps=4.768372e-07; result=1**

**22: eps=2.384186e-07; result=1**

**23: eps=1.192093e-07; result=1**

**24: eps=5.960464e-08; result=1**

**25: eps=2.980232e-08; result=1**

**26: eps=1.490116e-08; result=1**

**27: eps=7.450581e-09; result=1**

**28: eps=3.725290e-09; result=1**

**29: eps=1.862645e-09; result=1**

**30: eps=9.313226e-10; result=1**

**31: eps=4.656613e-10; result=1**

**32: eps=2.328306e-10; result=1**

**33: eps=1.164153e-10; result=1**

**34: eps=5.820766e-11; result=1**

**35: eps=2.910383e-11; result=1**

**36: eps=1.455192e-11; result=1**

**37: eps=7.275958e-12; result=1**

**38: eps=3.637979e-12; result=1**

**39: eps=1.818989e-12; result=1**

**40: eps=9.094947e-13; result=1**

**41: eps=4.547474e-13; result=1**

**42: eps=2.273737e-13; result=1**

**43: eps=1.136868e-13; result=1**

**44: eps=5.684342e-14; result=1**

**45: eps=2.842171e-14; result=1**

**46: eps=1.421085e-14; result=1**

**47: eps=7.105427e-15; result=1**

**48: eps=3.552714e-15; result=1**

**49: eps=1.776357e-15; result=1**

**50: eps=8.881784e-16; result=1**

**51: eps=4.440892e-16; result=1**

**52: eps=2.220446e-16; result=1**

**53: eps=1.110223e-16; result=0**

**54: eps=5.551115e-17; result=0**

**55: eps=2.775558e-17; result=0**

**56: eps=1.387779e-17; result=0**

**57: eps=6.938894e-18; result=0**

**58: eps=3.469447e-18; result=0**

**59: eps=1.734723e-18; result=0**

**60: eps=8.673617e-19; result=0**

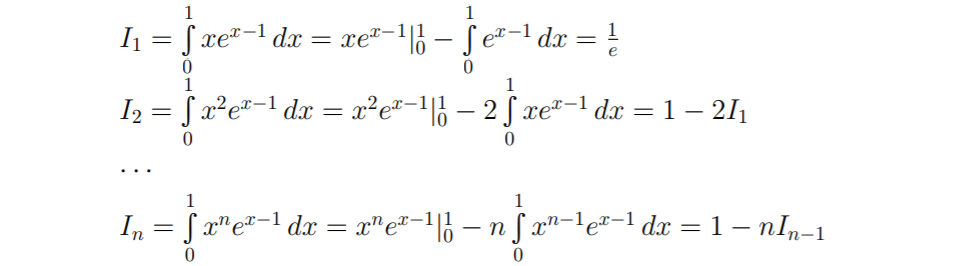
Действительно, это произошло на 53 итерации

……………………………………………………………………………………

Задание №4

Рассмотрим погрешности численных методов. Построим алгоритм вычисления интеграла

Интегрируя по частям, находим :



Вычислим значения интегралов, до n = 30.

**disp('---task 3---')**

**syms n;**

**n=30;**

**I = exp(-1);**

**for i = 2:1:n**

**I = 1 - i\*I;**

**fprintf('I%d = %d\n',i,I);**

**end**

**Command window**

**I2 = 2.642411e-01**

**I3 = 2.072766e-01**

**I4 = 1.708934e-01**

**I5 = 1.455329e-01**

**I6 = 1.268024e-01**

**I7 = 1.123835e-01**

**I8 = 1.009320e-01**

**I9 = 9.161229e-02**

**I10 = 8.387707e-02**

**I11 = 7.735223e-02**

**I12 = 7.177325e-02**

**I13 = 6.694778e-02**

**I14 = 6.273108e-02**

**I15 = 5.903379e-02**

**I16 = 5.545930e-02**

**I17 = 5.719187e-02**

**I18 = -2.945367e-02**

**I19 = 1.559620e+00**

**I20 = -3.019239e+01**

**I21 = 6.350403e+02**

**I22 = -1.396989e+04**

**I23 = 3.213084e+05**

**I24 = -7.711400e+06**

**I25 = 1.927850e+08**

**I26 = -5.012410e+09**

**I27 = 1.353351e+11**

**I28 = -3.789382e+12**

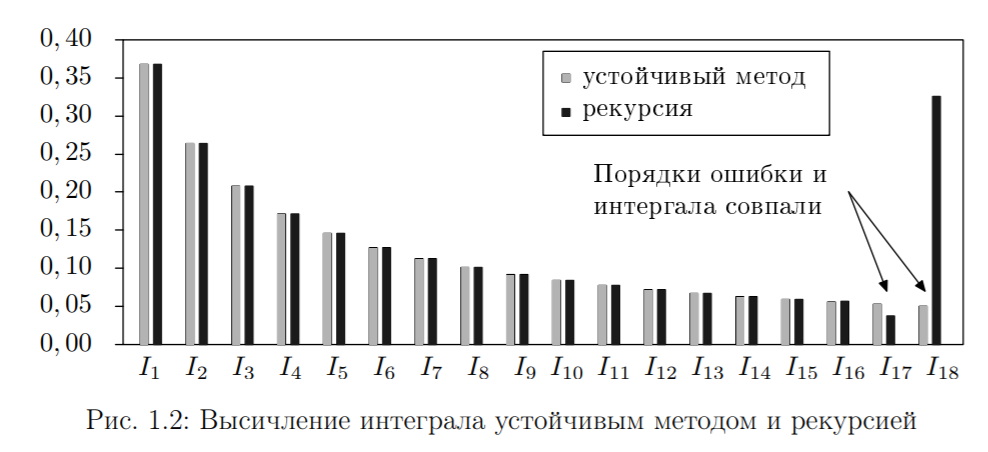
**I29 = 1.098921e+14**

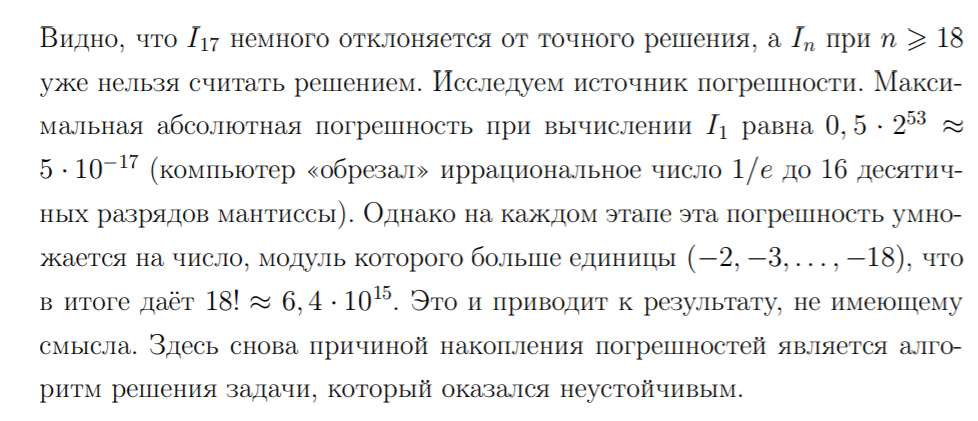
**I30 = -3.296762e+15**

Заметим , что подынтегральная функция на всем отрезке интегрирования неотрицательна, следовательно, и значение интеграла — положительное число. Более того, подынтегральная функция на данном интервале ограничена функцией y = 1, т.е. значение интеграла не может превышать единицы.

При рекуррентных операциях одновременного сложения и умножения возникают погрешности машинного вычисления, которые накапливаются с каждой итерацией и с определенного шага становятся заметны.

Вывод: На рис. 1.2 n-ый столбик обозначает In. Причём чёрный столбик — это In, вычисленный по только что изложенному рекурсивному методу, а серый — это In, вычисленный по более устойчивому методу конечных сумм.





………………………………………………………………………………………

Задание №5

Напишем функцию вычисления значения синуса в виде конечной суммы ряда . По признаку Лейбница погрешность вычисления сходящегося знакопеременного ряда не превышает по абсолютной величине первого из отброшенных членов. Вычисления членов ряда проводите до вычисления члена по модулю не превышающего 10^−17. Вычислим значение синуса в точках 0, π /3 , π /2 , π, 2π, убедимся, что полученные значения с высокой степенью точности совпадают с действительным значением синуса в этих точках. Далее проведём вычисления в точках 12π, 13π, 14π, выводя результаты по шагам (посчитанный член и частичную сумму ряда). Полученные результаты объясняются погрешностями округления, в реальных программах значение аргумента приводится к отрезку [0; π /2 ].

**function[result] = sin\_series(x)**

**eps = 1e-17;**

**result = 0;**

**i = 0;**

**f = @(x,k)(-1)^k \* x^(2\*k + 1) / factorial(2\*k + 1);**

**while(true)**

**a = f(x,i);**

**result = result + a;**

**if (abs(a) < eps)**

**break;**

**end**

**i = i + 1;**

**end**

**end**

**Command window**

**>> sin\_series(0)**

**ans =**

**0**

**>> sin\_series(pi / 3)**

**ans =**

**0.8660254037844385**

**>> sin\_series(pi / 2)**

**ans =**

**1**

**>> sin\_series(pi)**

**ans =**

**3.328027611925569e-16**

**>> sin\_series(2 \* pi)**

**ans =**

**3.300897561490086e-15**

**>> sin\_series2(12 \* pi)**

**0: member = 3.769911e+001; sum = 37.69911**

**1: member = -8.929808e+003; sum = -8892.109**

**2: member = 6.345624e+005; sum = 625670.3**

**3: member = -2.147273e+007; sum = -20847060**

**4: member = 4.238547e+008; sum = 403007700**

**5: member = -5.476292e+009; sum = -5073284000**

**6: member = 4.989123e+010; sum = 44817950000**

**7: member = -3.376503e+011; sum = -292832400000**

**8: member = 1.764252e+012; sum = 1471419000000**

**9: member = -7.331564e+012; sum = -5860145000000**

**10: member = 2.480902e+013; sum = 18948870000000**

**11: member = -6.968211e+013; sum = -50733240000000**

**12: member = 1.650564e+014; sum = 114323100000000**

**13: member = -3.341623e+014; sum = -219839100000000**

**14: member = 5.848758e+014; sum = 365036600000000**

**15: member = -8.938053e+014; sum = -528768700000000**

**16: member = 1.202932e+015; sum = 674163800000000**

**17: member = -1.436668e+015; sum = -762504500000000**

**18: member = 1.532902e+015; sum = 770397900000000**

**19: member = -1.470038e+015; sum = -699640000000000**

**20: member = 1.273934e+015; sum = 574294000000000**

**21: member = -1.002516e+015; sum = -428222300000000**

**22: member = 7.195956e+014; sum = 291373300000000**

**23: member = -4.730369e+014; sum = -181663600000000**

**24: member = 2.858380e+014; sum = 104174400000000**

**25: member = -1.593096e+014; sum = -55135250000000**

**26: member = 8.215330e+013; sum = 27018050000000**

**27: member = -3.931251e+013; sum = -12294460000000**

**28: member = 1.750371e+013; sum = 5209252000000**

**29: member = -7.269632e+012; sum = -2060379000000**

**30: member = 2.822887e+012; sum = 762508100000**

**31: member = -1.027126e+012; sum = -264617500000**

**32: member = 3.509073e+011; sum = 86289820000**

**33: member = -1.127810e+011; sum = -26491190000**

**34: member = 3.416176e+010; sum = 7670570000**

**35: member = -9.768909e+009; sum = -2098338000**

**36: member = 2.641514e+009; sum = 543175600**

**37: member = -6.764289e+008; sum = -133253400**

**38: member = 1.642783e+008; sum = 31024900**

**39: member = -3.788965e+007; sum = -6864756**

**40: member = 8.310131e+006; sum = 1445375**

**41: member = -1.735314e+006; sum = -289939.2**

**42: member = 3.454158e+005; sum = 55476.62**

**43: member = -6.561252e+004; sum = -10135.9**

**44: member = 1.190629e+004; sum = 1770.385**

**45: member = -2.066116e+003; sum = -295.7308**

**46: member = 3.431991e+002; sum = 47.46822**

**47: member = -5.462065e+001; sum = -7.152431**

**48: member = 8.336354e+000; sum = 1.183923**

**49: member = -1.221173e+000; sum = -0.03724956**

**50: member = 1.718375e-001; sum = 0.134588**

**51: member = -2.324571e-002; sum = 0.1113422**

**52: member = 3.025397e-003; sum = 0.1143676**

**53: member = -3.791011e-004; sum = 0.1139885**

**54: member = 4.576853e-005; sum = 0.1140343**

**55: member = -5.327379e-006; sum = 0.114029**

**56: member = 5.982453e-007; sum = 0.1140296**

**57: member = -6.485431e-008; sum = 0.1140295**

**58: member = 6.791368e-009; sum = 0.1140295**

**59: member = -6.873699e-010; sum = 0.1140295**

**60: member = 6.728003e-011; sum = 0.1140295**

**61: member = -6.372113e-012; sum = 0.1140295**

**62: member = 5.842705e-013; sum = 0.1140295**

**63: member = -5.189218e-014; sum = 0.1140295**

**64: member = 4.466471e-015; sum = 0.1140295**

**65: member = -3.727452e-016; sum = 0.1140295**

**66: member = 3.017510e-017; sum = 0.1140295**

**67: member = -2.370677e-018; sum = 0.1140295**

**ans =**

**0.1140295178893492**

**>> sin\_series2(13 \* pi)**

**0: member = 4.084070e+001; sum = 40.8407**

**1: member = -1.135346e+004; sum = -11312.62**

**2: member = 9.468581e+005; sum = 935545.4**

**3: member = -3.760296e+007; sum = -36667420**

**4: member = 8.711160e+008; sum = 834448600**

**5: member = -1.320899e+010; sum = -12374550000**

**6: member = 1.412315e+011; sum = 128857000000**

**7: member = -1.121757e+012; sum = -992900000000**

**8: member = 6.878857e+012; sum = 5885957000000**

**9: member = -3.354877e+013; sum = -27662820000000**

**10: member = 1.332336e+014; sum = 105570800000000**

**11: member = -4.391873e+014; sum = -333616500000000**

**12: member = 1.220914e+015; sum = 887297200000000**

**13: member = -2.900910e+015; sum = -2013613000000000**

**14: member = 5.958881e+015; sum = 3945268000000000**

**15: member = -1.068731e+016; sum = -6742037000000000**

**16: member = 1.688071e+016; sum = 10138670000000000**

**17: member = -2.366084e+016; sum = -13522170000000000**

**18: member = 2.962869e+016; sum = 16106520000000000**

**19: member = -3.334653e+016; sum = -17240010000000000**

**20: member = 3.391512e+016; sum = 16675100000000000**

**21: member = -3.132290e+016; sum = -14647800000000000**

**22: member = 2.638659e+016; sum = 11738790000000000**

**23: member = -2.035701e+016; sum = -8618223000000000**

**24: member = 1.443654e+016; sum = 5818318000000000**

**25: member = -9.442988e+015; sum = -3624669000000000**

**26: member = 5.715006e+015; sum = 2090336000000000**

**27: member = -3.209569e+015; sum = -1119232000000000**

**28: member = 1.677144e+015; sum = 557911300000000**

**29: member = -8.174791e+014; sum = -259567900000000**

**30: member = 3.725478e+014; sum = 112980000000000**

**31: member = -1.590876e+014; sum = -46107590000000**

**32: member = 6.378659e+013; sum = 17678990000000**

**33: member = -2.406008e+013; sum = -6381086000000**

**34: member = 8.553139e+012; sum = 2172053000000**

**35: member = -2.870487e+012; sum = -698434100000**

**36: member = 9.109335e+011; sum = 212499400000**

**37: member = -2.737664e+011; sum = -61266990000**

**38: member = 7.803012e+010; sum = 16763140000**

**39: member = -2.112161e+010; sum = -4358475000**

**40: member = 5.436739e+009; sum = 1078264000**

**41: member = -1.332395e+009; sum = -254131500**

**42: member = 3.112585e+008; sum = 57127040**

**43: member = -6.938890e+007; sum = -12261860**

**44: member = 1.477759e+007; sum = 2515739**

**45: member = -3.009583e+006; sum = -493843.4**

**46: member = 5.867079e+005; sum = 92864.58**

**47: member = -1.095865e+005; sum = -16721.89**

**48: member = 1.962910e+004; sum = 2907.208**

**49: member = -3.374626e+003; sum = -467.4174**

**50: member = 5.573021e+002; sum = 89.88471**

**51: member = -8.847891e+001; sum = 1.405804**

**52: member = 1.351461e+001; sum = 14.92042**

**53: member = -1.987469e+000; sum = 12.93295**

**54: member = 2.816025e-001; sum = 13.21455**

**55: member = -3.846869e-002; sum = 13.17608**

**56: member = 5.069876e-003; sum = 13.18115**

**57: member = -6.450318e-004; sum = 13.1805**

**58: member = 7.927271e-005; sum = 13.18058**

**59: member = -9.416319e-006; sum = 13.18057**

**60: member = 1.081685e-006; sum = 13.18058**

**61: member = -1.202327e-007; sum = 13.18058**

**62: member = 1.293830e-008; sum = 13.18058**

**63: member = -1.348620e-009; sum = 13.18058**

**64: member = 1.362311e-010; sum = 13.18058**

**65: member = -1.334283e-011; sum = 13.18058**

**66: member = 1.267678e-012; sum = 13.18058**

**67: member = -1.168845e-013; sum = 13.18058**

**68: member = 1.046366e-014; sum = 13.18058**

**69: member = -9.098637e-016; sum = 13.18058**

**70: member = 7.688040e-017; sum = 13.18058**

**71: member = -6.315063e-018; sum = 13.18058**

**ans =**

**13.18057553492365**

**>> sin\_series2(14 \* pi)**

**0: member = 4.398230e+001; sum = 43.9823**

**1: member = -1.418020e+004; sum = -14136.22**

**2: member = 1.371539e+006; sum = 1357403**

**3: member = -6.317057e+007; sum = -61813170**

**4: member = 1.697220e+009; sum = 1635407000**

**5: member = -2.984704e+010; sum = -28211630000**

**6: member = 3.701114e+011; sum = 341899800000**

**7: member = -3.409330e+012; sum = -3067430000000**

**8: member = 2.424688e+013; sum = 21179450000000**

**9: member = -1.371468e+014; sum = -115967300000000**

**10: member = 6.316728e+014; sum = 515705400000000**

**11: member = -2.414891e+015; sum = -1899185000000000**

**12: member = 7.785778e+015; sum = 5886593000000000**

**13: member = -2.145462e+016; sum = -15568020000000000**

**14: member = 5.111172e+016; sum = 35543700000000000**

**15: member = -1.063147e+017; sum = -70771020000000000**

**16: member = 1.947535e+017; sum = 123982500000000000**

**17: member = -3.165878e+017; sum = -192605300000000000**

**18: member = 4.597754e+017; sum = 267170100000000000**

**19: member = -6.001411e+017; sum = -332971000000000000**

**20: member = 7.078893e+017; sum = 374918300000000000**

**21: member = -7.582342e+017; sum = -383315900000000000**

**22: member = 7.407881e+017; sum = 357472200000000000**

**23: member = -6.628178e+017; sum = -305345600000000000**

**24: member = 5.451458e+017; sum = 239800200000000000**

**25: member = -4.135502e+017; sum = -173750100000000000**

**26: member = 2.902718e+017; sum = 116521800000000000**

**27: member = -1.890620e+017; sum = -72540230000000000**

**28: member = 1.145769e+017; sum = 42036700000000000**

**29: member = -6.476987e+016; sum = -22733160000000000**

**30: member = 3.423322e+016; sum = 11500060000000000**

**31: member = -1.695397e+016; sum = -5453910000000000**

**32: member = 7.883767e+015; sum = 2429857000000000**

**33: member = -3.448823e+015; sum = -1018965000000000**

**34: member = 1.421899e+015; sum = 402933500000000**

**35: member = -5.534369e+014; sum = -150503400000000**

**36: member = 2.036895e+014; sum = 53186110000000**

**37: member = -7.099560e+013; sum = -17809480000000**

**38: member = 2.346837e+013; sum = 5658887000000**

**39: member = -7.367447e+012; sum = -1708560000000**

**40: member = 2.199368e+012; sum = 490807600000**

**41: member = -6.251176e+011; sum = -134310000000**

**42: member = 1.693633e+011; sum = 35053330000**

**43: member = -4.378823e+010; sum = -8734904000**

**44: member = 1.081535e+010; sum = 2080446000**

**45: member = -2.554539e+009; sum = -474093000**

**46: member = 5.775605e+008; sum = 103467500**

**47: member = -1.251128e+008; sum = -21645330**

**48: member = 2.599051e+007; sum = 4345173**

**49: member = -5.182142e+006; sum = -836968.2**

**50: member = 9.925302e+005; sum = 155562**

**51: member = -1.827520e+005; sum = -27189.98**

**52: member = 3.237392e+004; sum = 5183.942**

**53: member = -5.521556e+003; sum = -337.6141**

**54: member = 9.073337e+002; sum = 569.7197**

**55: member = -1.437498e+002; sum = 425.9699**

**56: member = 2.197185e+001; sum = 447.9417**

**57: member = -3.242050e+000; sum = 444.6997**

**58: member = 4.620954e-001; sum = 445.1618**

**59: member = -6.365881e-002; sum = 445.0981**

**60: member = 8.481012e-003; sum = 445.1066**

**61: member = -1.093298e-003; sum = 445.1055**

**62: member = 1.364466e-004; sum = 445.1056**

**63: member = -1.649469e-005; sum = 445.1056**

**64: member = 1.932415e-006; sum = 445.1056**

**65: member = -2.195035e-007; sum = 445.1056**

**66: member = 2.418643e-008; sum = 445.1056**

**67: member = -2.586360e-009; sum = 445.1056**

**68: member = 2.685254e-010; sum = 445.1056**

**69: member = -2.707992e-011; sum = 445.1056**

**70: member = 2.653725e-012; sum = 445.1056**

**71: member = -2.528060e-013; sum = 445.1056**

**72: member = 2.342139e-014; sum = 445.1056**

**73: member = -2.111049e-015; sum = 445.1056**

**74: member = 1.851852e-016; sum = 445.1056**

**75: member = -1.581590e-017; sum = 445.1056**

**76: member = 1.315572e-018; sum = 445.1056**

**ans =**

**445.1056211091088**