**Токаева Александра, 409 группа**

**Отчет по практикуму на ЭВМ**

**Задание**: загадана некоторая функция f; дан отрезок [a, b],

даны n точек на этом отрезке и даны значения

загаданной функции в этих точках. Нужно по этим данным построить для загаданной функции на данном отрезке интерполяционный многочлен двумя способами:

1) в форме многочлена и найти его коэффициенты как решение СЛУ используя какой-нибудь точный метод решения системы (я буду использовать метод отражений).

2) в форме где – базисные функции, которые равны 1 в своем узле и равны 0 в остальных узлах.

Требуется построить этот многочлен по равноотстоящим узлам (то есть ) и по системе чебышевских узлов (то есть ).

Построить графики функции и ее многочлена для своей функции, функции Рунге и функции модуль.

**Решение:** сначала опишем интерфейс программы на си, затем способ вычисления , и наконец, суть метода отражений для вычисления .

Программа называется mnog\_lagranzh.cpp, никаких файлов на вход не требует, зато требует аргументы командной строки:

Usage: ./a.out a b n ravnom/chebushov

Здесь a,b – концы отрезка, n-число узлов ( в случае равноотстоящих узлов оба конца отрезка являются первым и последним узлом соответственно, в случае чебышевских узлов – все n узлов лежат внутри отрезка, концы отрезка не являются узлами), ravnom/chebushov отвечает за вид узлов; вид загаданной функции регулируется функцией double func(double x).

Функция int generate\_input(double a, double b, int n, char\* tip\_uzlov, const char\* filename) записывает входные данные (узлы и значения функции func) в файл filename, потом функция main открывает этот файл, считывает массив точек в mas1, массив значений в mas2, вычисляет значения обоих многочленов и в исходных узлах и в добавочных узлах (по два добавочных узла между каждыми старыми узлами) и записывает их в файл output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt или output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt ( в зависимости от типа точек) в 4 столбика в формате x\_i, f(x\_i), P\_n\_1(x\_i), L\_n(x\_i).

Значения многочлена в любой точке x вычислить легко, имея mas1 и mas2 – просто честно посчитать значение каждой из n базисных функций в этой точке, умножить на и просуммировать по всем i. Этим и занимаются функции

double F\_i(int n, int i, double x, double\* mas1)

и

double L\_n(int n, double x, double\* mas1, double\* mas2).

Для вычисления значения многочлена в любой точке x нужно сначала вычислить его коэффициенты . Для этого надо составить и решить систему

Это делает функция

int algoritm(int n, double\* A, double\* X, double\* B).

Она приводит систему методом отражений к верхнетреугольному виду, а потом применяет обратный ход метода Гаусса. То есть она на k-м шаге для вектора y:= k-го стобца матрицы ( у которого сверху обрубили n-k элементов) по формуле строит единичный вектор x ( которому соответствует n\*n-матрица отражения U(x)= E-xx\*) такой, U(x)y = ||y||e\_k, то есть после применения матрицы U(x) к k-му столбцу получится на диагонали число ||y||, а ниже диагонали – нули. После этого последовательно применяем матрицу U(x) ко всем обрубкам столбцов с номерами k+1, …, n и еще к вектору B.

Матрица U(x) хороша по трем причинам:

1)она ортогональная

2)результат ее применения к вектору y считается не за n\*3 (как если бы мы честно перемножали две матрицы), а за 2n, за счет вида самой матрицы: U(x)y = (E-xx\*)y=y - x<x,y>.

3)Физический смысл применения матрицы U(x) к вектору y состоит в том, что мы вектор y отражаем относительно плоскости с нормалью x, поэтому используя этот факт и то, что U(x)y = ||y||e\_1 получаем, что y-||y||e\_1 параллелено вектору x, то есть y-||y||e\_1=x. Из условия единичной длины вектора x находим явную формулу для него:

Теперь построим графики функций y=x\*x + cos(5\*x\*x\*x - 2), y=1/(25x\*x+1) и y=|x| .

**1)Моя функция**

Сначала мы откомментируем строку

return x\*x + cos(5\*x\*x\*x - 2);

в функции double func(double x). , тем самым сообщив программе о том, какую функцию мы хотим приближать

Потом запускам в терминале g++ mnog\_lagranzh.cpp

Теперь у нас есть файл a.out, который можно (см. Usage: ./a.out a b n ravnom/chebushov) запускать, задавая тип узлов либо равномерный, либо чебышевский, но по обеим системам узлов строиться будет функция x\*x + cos(5\*x\*x\*x - 2)

Пусть мы хотим построить нашу функцию по n=10 узлам

**a)Пусть сначала узлы равномерные**

Запускаем в терминале ./a.out -1 1 10 ravnom

В текущей директории появился файл output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt, в котором 4 столбика:

x\_i, f(x\_i), P\_n\_1(x\_i), L\_n(x\_i)

Посмотрим на них

-1.000000 1.753902 1.753902 1.753902

-0.925926 1.808437 1.808553 1.808553

-0.851852 1.095039 1.051079 1.051079

-0.777778 0.252803 0.252801 0.252801

-0.703704 -0.329700 -0.301780 -0.301780

-0.629630 -0.597907 -0.579334 -0.579334

-0.555556 -0.651229 -0.651229 -0.651229

-0.481481 -0.602717 -0.610767 -0.610767

-0.407407 -0.528223 -0.533615 -0.533615

-0.333333 -0.465348 -0.465348 -0.465348

-0.259259 -0.426481 -0.423979 -0.423979

-0.185185 -0.410512 -0.408823 -0.408823

-0.111111 -0.410028 -0.410028 -0.410028

-0.037037 -0.415006 -0.415586 -0.415586

0.037037 -0.414544 -0.414656 -0.414656

0.111111 -0.397555 -0.397555 -0.397555

0.185185 -0.352775 -0.353801 -0.353801

0.259259 -0.268225 -0.270130 -0.270130

0.333333 -0.130493 -0.130492 -0.130492

0.407407 0.075012 0.080418 0.080418

0.481481 0.360360 0.369445 0.369445

0.555556 0.723817 0.723815 0.723815

0.629630 1.126779 1.099188 1.099188

0.703704 1.462195 1.411924 1.411924

0.777778 1.543438 1.543438 1.543438

0.851852 1.187482 1.367886 1.367886

0.925926 0.469427 0.818218 0.818218

1.000000 0.010008 0.010008 0.010008

Мы видим, что 3 и 4 столбики абсолютно одинаковые, то есть значения интерполяционного многочлена, вычисленные двумя способами совпадают. Это говорит о том, что используемая в первом способе СЛУ получилась хорошо обусловленная и что мы ее хорошо решили (найдя тем самым коэффициенты интерполяционного многочлена), не накопив огромную погрешность. Далее, 2 и 3(4) столбики совпадают не во всех строках, а с шагом три: 1,4,7,10… Это связано с тем, что исходная функция и интерполяционный многочлен обязаны совпадать только в узлах, а между узлами мы добавили по две точки просто чтобы посмотреть, сильно ли многочлен от исходной функции отличается в точках, отличных от узлов.

Построим график по этим точкам, используя gnuplot. Запускаем в терминале gnuplot, консоль стала выдавать приглашения на команды.

gnuplot> set terminal png size 640,480

gnuplot> set output "plot\_ravnom\_myfunc.png"

gnuplot> plot 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:2 with lines, 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:3 with lines, 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:4 with lines, 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:2 with points

gnuplot> q

Изображение выглядит как карта

Автоматически созданное описание

Смотрим на график – видим, что функция хорошо приблизилась. (После этого для удобства хранения файлов переименуем файл 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' в 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n\_myfunc.txt')

**б)Теперь узлы равномерные, функция по-прежнему моя**

Запускаем в терминале ./a.out -1 1 10 chebushov

В текущей директории появился файл output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt, в котором 4 столбика:

x\_i, f(x\_i), P\_n\_1(x\_i), L\_n(x\_i)

Посмотрим на них

0.987688 0.027561 0.027560 0.027560

0.955461 0.202248 0.228412 0.228412

0.923234 0.496489 0.520464 0.520464

0.891007 0.827860 0.827865 0.827865

0.829707 1.343947 1.297512 1.297512

0.768407 1.554612 1.512266 1.512266

0.707107 1.473156 1.473155 1.473155

0.622735 1.089850 1.136352 1.136352

0.538362 0.633647 0.671896 0.671896

0.453990 0.244746 0.244748 0.244748

0.354805 -0.078541 -0.108022 -0.108022

0.255619 -0.273506 -0.296192 -0.296192

0.156434 -0.374195 -0.374195 -0.374195

0.052145 -0.412783 -0.399821 -0.399821

-0.052145 -0.414072 -0.405558 -0.405558

-0.156434 -0.409003 -0.409003 -0.409003

-0.255619 -0.425205 -0.426650 -0.426650

-0.354805 -0.481311 -0.480483 -0.480483

-0.453990 -0.575387 -0.575387 -0.575387

-0.538362 -0.645564 -0.646364 -0.646364

-0.622735 -0.610032 -0.607355 -0.607355

-0.707107 -0.310274 -0.310275 -0.310275

-0.768407 0.161019 0.144897 0.144897

-0.829707 0.831441 0.800496 0.800496

-0.891007 1.528057 1.528053 1.528053

-0.923234 1.792233 1.832626 1.832626

-0.955461 1.909862 1.976825 1.976825

-0.987688 1.836102 1.836101 1.836101

Опять 3 и 4 столбики полностью совпадают, а 2 и 3(4) совпадают с шагом три. Строим график в gnuplot, читая данные из файла 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' и сохраняя полученный график в файл "plot\_chebushov\_myfunc.png".

gnuplot> set terminal png size 640,480

gnuplot> set output "plot\_chebushov\_myfunc.png"

gnuplot> plot 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:2 with lines, 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:3 with lines, 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:4 with lines, 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:2 with points

gnuplot> q

Изображение выглядит как карта

Автоматически созданное описание

Смотрим на график – тоже хорошо приблизилась. (После этого для удобства хранения файлов переименуем файл 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' в 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n\_myfunc.txt')

**2)Функция Рунге**

Сначала мы откомментируем строку

return 1/(25\*x\*x + 1);

в функции double func(double x). , тем самым сообщив программе о том, что мы хотим приближать функцию Рунге

Потом запускам в терминале g++ mnog\_lagranzh.cpp

Теперь у нас есть файл a.out, который можно (см. Usage: ./a.out a b n ravnom/chebushov) запускать, задавая тип узлов либо равномерный, либо чебышевский, но по обеим системам узлов строиться будет функция 1/(25\*x\*x + 1);

Пусть мы хотим построить нашу функцию по n=12 узлам

**a)Пусть сначала узлы равномерные**

Запускаем в терминале ./a.out -1 1 12 ravnom

В текущей директории появился файл output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt, в котором 4 столбика:

x\_i, f(x\_i), P\_n\_1(x\_i), L\_n(x\_i)

Посмотрим на них

-1.000000 0.038462 0.038462 0.038462

-0.939394 0.043362 0.597214 0.597214

-0.878788 0.049245 0.335129 0.335129

-0.818182 0.056384 0.056383 0.056383

-0.757576 0.065155 -0.038870 -0.038870

-0.696970 0.076079 0.004843 0.004843

-0.636364 0.089896 0.089896 0.089896

-0.575758 0.107673 0.149168 0.149168

-0.515152 0.130984 0.166026 0.166026

-0.454545 0.162198 0.162198 0.162198

-0.393939 0.204930 0.174554 0.174554

-0.333333 0.264706 0.233546 0.233546

-0.272727 0.349711 0.349710 0.349710

-0.212121 0.470614 0.510041 0.510041

-0.151515 0.635356 0.682949 0.682949

-0.090909 0.828767 0.828767 0.828767

-0.030303 0.977558 0.912039 0.912039

0.030303 0.977558 0.912039 0.912039

0.090909 0.828767 0.828767 0.828767

0.151515 0.635356 0.682949 0.682949

0.212121 0.470614 0.510041 0.510041

0.272727 0.349711 0.349710 0.349710

0.333333 0.264706 0.233546 0.233546

0.393939 0.204930 0.174554 0.174554

0.454545 0.162198 0.162198 0.162198

0.515152 0.130984 0.166026 0.166026

0.575758 0.107673 0.149168 0.149168

0.636364 0.089896 0.089896 0.089896

0.696970 0.076079 0.004843 0.004843

0.757576 0.065155 -0.038870 -0.038870

0.818182 0.056384 0.056383 0.056383

0.878788 0.049245 0.335129 0.335129

0.939394 0.043362 0.597214 0.597214

1.000000 0.038462 0.038462 0.038462

Опять 3 и 4 столбики полностью совпадают, а 2 и 3(4) совпадают с шагом три. Строим график в gnuplot, читая данные из файла 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' и сохраняя полученный график в файл "plot\_ravnom\_runge.png".

gnuplot> set terminal png size 640,480

gnuplot> set output "plot\_ravnom\_runge.png"

gnuplot> plot 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:2 with lines, 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:3 with lines, 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:4 with lines, 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:2 with points

gnuplot> q

Изображение выглядит как текст, карта

Автоматически созданное описание

Смотрим на график – видим, что к концам отрезка приближающий многочлен разбалтывается, причем при увеличении n размах и частота колебаний будет увеличиваться. При этом, к сожалению, область колебаний (то есть где многочлен очень плохо похож на исходную функцию) не будет приближаться к концам отрезка, а так и останется примерно [-1,-0.86] и [0.86,1]. То есть увеличение n не улучшает качество приближения. Однако окажется, что если узлы не равномерные, а Чебышевские (то есть более скучены к концам отрезка), то разбалтывания не будет. Проверим это, построив график для чебышевских узлов. (После этого для удобства хранения файлов переименуем файл 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' в 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n\_runge.txt')

**б)Теперь узлы равномерные, функция по-прежнему Рунге**

Запускаем в терминале ./a.out -1 1 12 chebushov

В текущей директории появился файл output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt, в котором 4 столбика:

x\_i, f(x\_i), P\_n\_1(x\_i), L\_n(x\_i)

0.991445 0.039102 0.039102 0.039102

0.968923 0.040866 0.048259 0.048259

0.946402 0.042750 0.048165 0.048165

0.923880 0.044765 0.044765 0.044765

0.880371 0.049077 0.040661 0.040661

0.836862 0.054029 0.046277 0.046277

0.793353 0.059754 0.059754 0.059754

0.731822 0.069497 0.081057 0.081057

0.670292 0.081751 0.094001 0.094001

0.608761 0.097421 0.097421 0.097421

0.533402 0.123260 0.103158 0.103158

0.458042 0.160126 0.135547 0.135547

0.382683 0.214539 0.214539 0.214539

0.298631 0.309645 0.359375 0.359375

0.214578 0.464879 0.537466 0.537466

0.130526 0.701299 0.701298 0.701298

0.043509 0.954813 0.803576 0.803576

-0.043509 0.954813 0.803576 0.803576

-0.130526 0.701299 0.701298 0.701298

-0.214578 0.464879 0.537466 0.537466

-0.298631 0.309645 0.359375 0.359375

-0.382683 0.214539 0.214539 0.214539

-0.458042 0.160126 0.135547 0.135547

-0.533402 0.123260 0.103158 0.103158

-0.608761 0.097421 0.097421 0.097421

-0.670292 0.081751 0.094001 0.094001

-0.731822 0.069497 0.081057 0.081057

-0.793353 0.059754 0.059754 0.059754

-0.836862 0.054029 0.046277 0.046277

-0.880371 0.049077 0.040661 0.040661

-0.923880 0.044765 0.044765 0.044765

-0.946402 0.042750 0.048165 0.048165

-0.968923 0.040866 0.048259 0.048259

-0.991445 0.039102 0.039102 0.039102

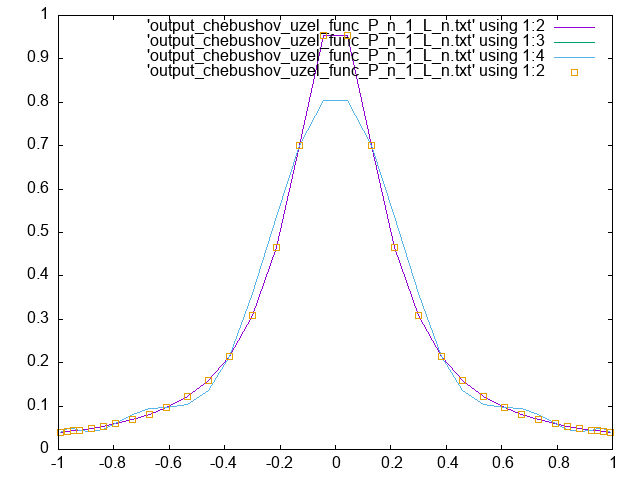
Опять 3 и 4 столбики полностью совпадают, а 2 и 3(4) совпадают с шагом три. Строим график в gnuplot, читая данные из файла 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' и сохраняя полученный график в файл "plot\_chebushov\_runge.png".

gnuplot> set terminal png size 640,480

gnuplot> set output "plot\_chebushov\_runge.png"

gnuplot> plot 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:2 with lines, 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:3 with lines, 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:4 with lines, 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:2 with points

gnuplot> q



Смотрим на график – видим, что к концам отрезка разбалтывание есть, то намного меньше, чем при интерполяции с равномерными узлами. (После этого для удобства хранения файлов переименуем файл 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' в 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n\_runge.txt')

**3)Функция модуль**

Сначала мы откомментируем строку

return (x>0)?x:-x;

в функции double func(double x). , тем самым сообщив программе о том, что мы хотим приближать функцию модуль

Потом запускам в терминале g++ mnog\_lagranzh.cpp

Теперь у нас есть файл a.out, который можно (см. Usage: ./a.out a b n ravnom/chebushov) запускать, задавая тип узлов либо равномерный, либо чебышевский, но по обеим системам узлов строиться будет функция y=|x|

Пусть мы хотим построить нашу функцию по n=15 узлам

**a)Пусть сначала узлы равномерные**

Запускаем в терминале ./a.out -1 1 15 ravnom

В текущей директории появился файл output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt, в котором 4 столбика:

x\_i, f(x\_i), P\_n\_1(x\_i), L\_n(x\_i)

Посмотрим на них

-1.000000 1.000000 1.000000 1.000000

-0.952381 0.952381 -3.006273 -3.006273

-0.904762 0.904762 -0.840552 -0.840552

-0.857143 0.857143 0.857146 0.857146

-0.809524 0.809524 1.261678 1.261678

-0.761905 0.761905 1.019501 1.019501

-0.714286 0.714286 0.714285 0.714285

-0.666667 0.666667 0.566332 0.566332

-0.619048 0.619048 0.551154 0.551154

-0.571429 0.571429 0.571429 0.571429

-0.523810 0.523810 0.559774 0.559774

-0.476190 0.476190 0.504238 0.504238

-0.428571 0.428571 0.428572 0.428572

-0.380952 0.380952 0.361450 0.361450

-0.333333 0.333333 0.315937 0.315937

-0.285714 0.285714 0.285714 0.285714

-0.238095 0.238095 0.254049 0.254049

-0.190476 0.190476 0.206985 0.206985

-0.142857 0.142857 0.142857 0.142857

-0.095238 0.095238 0.073910 0.073910

-0.047619 0.047619 0.020272 0.020272

0.000000 -0.000000 -0.000000 0.000000

0.047619 0.047619 0.020272 0.020272

0.095238 0.095238 0.073910 0.073910

0.142857 0.142857 0.142857 0.142857

0.190476 0.190476 0.206985 0.206985

0.238095 0.238095 0.254049 0.254049

0.285714 0.285714 0.285714 0.285714

0.333333 0.333333 0.315937 0.315937

0.380952 0.380952 0.361450 0.361450

0.428571 0.428571 0.428572 0.428572

0.476190 0.476190 0.504238 0.504238

0.523810 0.523810 0.559774 0.559774

0.571429 0.571429 0.571429 0.571429

0.619048 0.619048 0.551154 0.551154

0.666667 0.666667 0.566332 0.566332

0.714286 0.714286 0.714285 0.714285

0.761905 0.761905 1.019501 1.019501

0.809524 0.809524 1.261678 1.261678

0.857143 0.857143 0.857146 0.857146

0.904762 0.904762 -0.840552 -0.840552

0.952381 0.952381 -3.006273 -3.006273

1.000000 1.000000 1.000000 1.000000

Опять 3 и 4 столбики полностью совпадают, а 2 и 3(4) совпадают с шагом три. Строим график в gnuplot, читая данные из файла 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' и сохраняя полученный график в файл "plot\_ravnom\_modul.png".

gnuplot> set terminal png size 640,480

gnuplot> set output "plot\_ravnom\_modul.png"

gnuplot> plot 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:2 with lines, 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:3 with lines, 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:4 with lines, 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:2 with points

gnuplot> q

Изображение выглядит как текст, карта

Автоматически созданное описание

Смотрим на график – видим, что к концам отрезка приближающий многочлен разбалтывается. (После этого для удобства хранения файлов переименуем файл 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n.txt' в 'output\_ravnom\_uzel\_func\_P\_n\_1\_L\_n\_module.txt')

**б)Теперь узлы равномерные, функция по-прежнему модуль**

Запускаем в терминале ./a.out -1 1 15 chebushov

В текущей директории появился файл output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt, в котором 4 столбика:

x\_i, f(x\_i), P\_n\_1(x\_i), L\_n(x\_i)

0.994522 0.994522 0.994522 0.994522

0.980034 0.980034 0.973035 0.973035

0.965545 0.965545 0.960589 0.960589

0.951057 0.951057 0.951057 0.951057

0.922713 0.922713 0.929753 0.929753

0.894369 0.894369 0.900397 0.900397

0.866025 0.866025 0.866025 0.866025

0.825065 0.825065 0.817456 0.817456

0.784105 0.784105 0.776990 0.776990

0.743145 0.743145 0.743145 0.743145

0.691358 0.691358 0.700179 0.700179

0.639572 0.639572 0.648357 0.648357

0.587785 0.587785 0.587785 0.587785

0.527436 0.527436 0.516304 0.516304

0.467086 0.467086 0.455292 0.455292

0.406737 0.406737 0.406737 0.406737

0.340462 0.340462 0.356408 0.356408

0.274187 0.274187 0.292516 0.292516

0.207912 0.207912 0.207912 0.207912

0.138608 0.138608 0.109682 0.109682

0.069304 0.069304 0.030505 0.030505

0.000000 -0.000000 0.000000 0.000000

-0.069304 0.069304 0.030505 0.030505

-0.138608 0.138608 0.109682 0.109682

-0.207912 0.207912 0.207912 0.207912

-0.274187 0.274187 0.292516 0.292516

-0.340462 0.340462 0.356408 0.356408

-0.406737 0.406737 0.406737 0.406737

-0.467086 0.467086 0.455292 0.455292

-0.527436 0.527436 0.516304 0.516304

-0.587785 0.587785 0.587785 0.587785

-0.639572 0.639572 0.648357 0.648357

-0.691358 0.691358 0.700179 0.700179

-0.743145 0.743145 0.743145 0.743145

-0.784105 0.784105 0.776990 0.776990

-0.825065 0.825065 0.817456 0.817456

-0.866025 0.866025 0.866025 0.866025

-0.894369 0.894369 0.900397 0.900397

-0.922713 0.922713 0.929753 0.929753

-0.951057 0.951057 0.951057 0.951057

-0.965545 0.965545 0.960589 0.960589

-0.980034 0.980034 0.973035 0.973035

-0.994522 0.994522 0.994522 0.994522

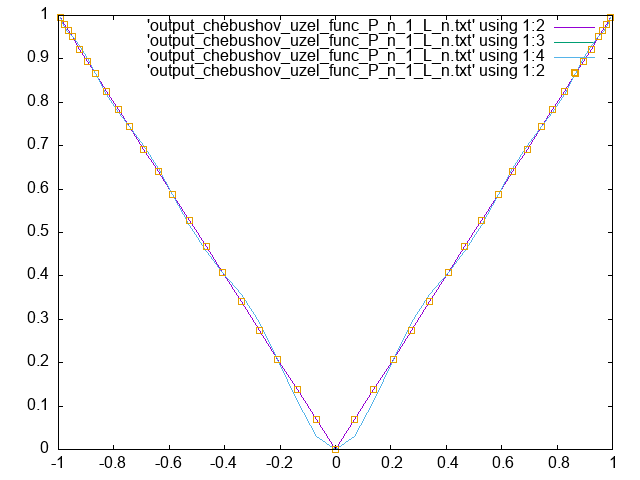
Опять 3 и 4 столбики полностью совпадают, а 2 и 3(4) совпадают с шагом три. Строим график в gnuplot, читая данные из файла 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' и сохраняя полученный график в файл "plot\_chebushov\_modul.png".

gnuplot> set terminal png size 640,480

gnuplot> set output "plot\_chebushov\_modul.png"

gnuplot> plot 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:2 with lines, 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:3 with lines, 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:4 with lines, 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' using 1:2 with points

gnuplot> q



Смотрим на график – видим, что функция хорошо приблизилась. (После этого для удобства хранения файлов переименуем файл 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n.txt' в 'output\_chebushov\_uzel\_func\_P\_n\_1\_L\_n\_modul.txt')