|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_Информатика и системы управления\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_Программное обеспечение ЭВМ и информационные технологии\_\_\_\_\_\_\_\_

**Лабораторная работа № 2**

**Тема** Построение и программная реализация алгоритма многомерной интерполяции табличных функций

**Студент** Серова М. Н.

**Группа** ИУ7-45Б

**Оценка (баллы)** \_\_\_\_\_\_\_\_

**Преподаватель** Градов В. М.

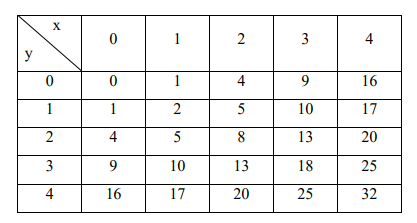
*Москва,*

*2021 г.*

**Цель работы.** Получение навыков построения алгоритма интерполяции таблично заданных функций двух переменных.

**Исходные данные.**

1. Таблица функции c количеством узлов 5х5



1. Степень аппроксимирующих полиномов – nx и ny.
2. Значение аргументов x, y, для которого выполняется интерполяция.

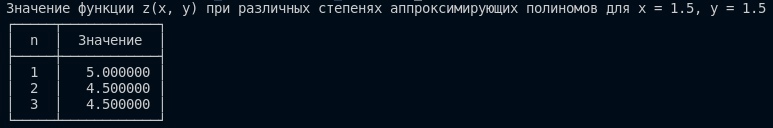
**Код программы**

Код программы представлен на листингах 1-3

|  |
| --- |
| Листинг 1. src/polynoms.c  #include "../inc/polynoms.h"  void **free\_table**(**table\_t** \*table)  {      if (table->x)  **free**(table->x);      if (table->y)  **free**(table->y);      if (table->z)  **free**(table->z);  }  **error\_code** **memory\_allocation**(**table\_t** \*table)  {  **error\_code** result = no\_errors;      table->x = **calloc**(**LENGTH**, sizeof(double));      if (!table->x)      {          result = error\_memory;      }      else      {          table->y = **calloc**(**LENGTH**, sizeof(double));          if (!table->y)              result = error\_memory;      }      if (!result)      {          table->z = **malloc**(**LENGTH** \* sizeof(double \*) + **LENGTH** \* **LENGTH** \* sizeof(double));          if (!table->z)              result = error\_memory;          else          {              for (int i = 0; i < **LENGTH**; i++)                  table->z[i] = (double \*) (((char \*) table->z) + **LENGTH** \* sizeof(double \*) + **LENGTH** \* i \* sizeof(double));          }      }      return result;  }  **error\_code** **get\_table**(**table\_t** \*table)  {  **error\_code** result = no\_errors;      result = **memory\_allocation**(table);      if (!result)      {          for (int i = 0; i < **LENGTH**; i++)          {              table->x[i] = i;              table->y[i] = i;          }          for (int i = 0; i < **LENGTH**; i++)          {              for (int j = i; j < **LENGTH**; j++)              {                  table->z[i][j] = i \* i + j \* j;                  table->z[j][i] = i \* i + j \* j;              }          }      }      return result;  }  int **get\_start\_index**(**table\_t** \*table, double x, int n)  {      int index = 0;      for (int i = 0; i < **LENGTH** - 1; i++)      {          if ((table->x[i] <= x && table->x[i + 1] >= x) || (table->x[i] >= x && table->x[i + 1] <= x))              index = i;      }      index -= n / 2 - 1;      if (index + n > **LENGTH**)          index = **LENGTH** - n - 1;      return (index < 0) ? 0 : index;  }  void **swap**(double \*\*first, double \*\*second)  {      double \*tmp = \*first;      \*first = \*second;      \*second = tmp;  }  double **linear\_interpolation**(int start, double \*tmp, double \*x\_line, double \*y\_line, int n, double x)  {      double p\_x = y\_line[start];  **memmove**(tmp, y\_line, **LENGTH** \* sizeof(double));      for (int i = start + 1; i < n + start; i++)      {          for (int j = start; j < n - i + start + start; j++)          {              tmp[j] = (tmp[j] - tmp[j + 1]) / (x\_line[j] - x\_line[j + i - start]);          }          double prom = tmp[start];          for (int j = start; j < i; j++)          {              prom \*= (x - x\_line[j]);          }          p\_x += prom;      }      return p\_x;  }  double **find\_value\_polynomial**(double x, int nx, double y, int ny, **error\_code** \*result)  {      double p\_x = 0;      nx++;      ny++;      int start\_x = 0;      int start\_y = 0;      double \*tmp = NULL;      double \*f = NULL;  **table\_t** table =      {          .x = NULL,          .y = NULL,          .z = NULL,      };      \*result = **get\_table**(&table);      if (!\*result)      {          start\_x = **get\_start\_index**(&table, x, nx);  **swap**(&table.x, &table.y);          start\_y = **get\_start\_index**(&table, y, ny);  **swap**(&table.x, &table.y);          tmp = **calloc**(**LENGTH**, sizeof(double));          if (!tmp)              \*result = error\_memory;          else          {              f = **calloc**(**LENGTH**, sizeof(double));              if (!f)                  \*result = error\_memory;          }      }      if (!\*result)      {          for (int i = start\_y; i < start\_y + ny; i++)          {              f[i] = **linear\_interpolation**(start\_x, tmp, table.x, table.z[i], nx, x);          }          p\_x = **linear\_interpolation**(start\_y, tmp, table.y, f, ny, y);      }  **free\_table**(&table);      if (tmp)  **free**(tmp);      if (f)  **free**(f);      return p\_x;  } |
| Листинг 2. polynoms.h  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <math.h>  #define **LENGTH** 5  #define **EPS** 1e-8  #define **START\_POW** 1  #define **END\_POW** 3  typedef struct  {      double \*x;      double \*y;      double \*\*z;  } **table\_t**;  typedef enum  {      no\_errors,      error\_memory,      error\_input,  } **error\_code**;  double **find\_value\_polynomial**(double x, int nx, double y, int ny, **error\_code** \*result); |
| Листинг 3. main.c  #include "../inc/polynoms.h"  int **main**(void)  {  **error\_code** result = no\_errors;      double x = 1.5;      double y = 1.5;  **printf**("Значение функции z(x, y) при различных степенях аппроксимирующих полиномов для x = %3.1f, y = %3.1f\n"              "┌─────┬────────────┐\n"              "│  n  │  Значение  │\n"              "├─────┼────────────┤\n", x, y);      for (int i = **START\_POW**; i <= **END\_POW** && !result; i++)      {  **printf**("│  %d  │ %10.6f │\n", i, \  **find\_value\_polynomial**(x, i, y, i, &result));      }  **printf**("└─────┴────────────┘\n");        return result;  } |

**Результаты работы.**

Результат интерполяции z(x,y) при степенях полиномов 1,2,3 для x=1.5, y=1.5



**Вопросы при защите лабораторной работы.**

1. Пусть производящая функция таблицы суть z(x,y)=x2 +y 2 . Область определения по x и y 0-5 и 0-5. Шаги по переменным равны 1. Степени nx = ny =1, x=y=1.5. Приведите по шагам те значения функции, которые получаются в ходе последовательных интерполяций по строкам и столбцу.

Для начала выберем отрезки x и y для интерполяции. Поскольку степень полинома равна 1, выберем отрезок [1;2], середине которой принадлежат данные значения аргументов. Первым действием рассмотрим интерполяцию функции для x = {1, 2}, y1 = {2, 5}. Тогда полученное в результате интерполяции значение: 2 + \* (1.5 – 1) = 3.5.

Далее интерполируем функцию x = {1, 2}, y2 = {5, 8}. Результат интерполяции: 5 + \* (1.5 – 1) = 6.5.

Последний шаг – интерполяция по столбцу для значений y = {1, 2}, f = {3.5, 6.5}. Результат двумерной интерполяции: 3.5 + \* (1.5 – 1) = 5.

1. Какова минимальная степень двумерного полинома, построенного на четырех узлах? На шести узлах?

Минимальная степень двумерного полинома, построенного на четырех узлах, равна двум. На шести – также двум. В случае использования четырех узлов для двумерного полинома граничные условия полинома – 4 уравнения от двух переменных f(xi, yi) = zi (i = 1, …, 4), а в полиноме первой степени (вида z = a + bx + cy) лишь 3 неизвестных параметра, что меньше количества граничных уравнений, следовательно, минимально возможная степень для такого случая – 2. В случае двумерного полинома, построенного на 6 узлах, также минимальным будет полином второй степени, поскольку количество неизвестных в нем совпадает с количеством граничных уравнений (z = dx2 + gy2 + hxy + bx + cy + a).

1. Предложите алгоритм двумерной интерполяции при хаотичном расположении узлов, т.е. когда таблицы функции на регулярной сетке нет, и метод последовательной интерполяции не работает. Какие имеются ограничения на расположение узлов при разных степенях полинома?

Алгоритм:

1. Выделить из множества точек группы по 6 точек таких, что все они не лежат на одной прямой.
2. По имеющимся 6 узлам для каждой группы построить полином второй степени (z = dx2 + gy2 + hxy + bx + cy + a) для интерполяции.

Главное ограничение на расположение узлов – точки в выделенных группах не должны лежать на одной прямой.

1. Пусть на каком-либо языке программирования написана функция, выполняющая интерполяцию по двум переменным. Опишите алгоритм использования этой функции для интерполяции по трем переменным.

Для осуществления трехмерной интерполяции (по трем точкам) достаточно использовать подход, аналогичный использованному для двумерной интерполяции. Трехмерную интерполяцию нужно привести к серии последовательных двумерных: сначала двумерно интерполировать по двум фиксированным переменным (в зависимости от степени используемых полиномов), результаты сохранить в отдельной таблице и, используя ее и третью переменную, провести итоговую двумерную интерполяцию.

1. Можно ли при последовательной интерполяции по разным направлениям использовать полиномы несовпадающих степеней или даже разные методы одномерной интерполяции, например, полином Ньютона и сплайн?

Можно использовать полиномы несовпадающих степеней по разным направлениям, от этого будет меняться лишь точность ответа. При последовательной интерполяции метод одномерного интерполирования не играет значительную роль, ведь ответ все равно получится приближенным к реальному значению. Однако вычислить погрешность последовательной интерполяции при использовании различных степеней полиномов по разным направлениям и методов одномерной интерполяции будет сложнее, чем при одинаковых степенях и методах.

1. Опишите алгоритм двумерной интерполяции на треугольной конфигурации узлов.

При треугольной конфигурации узлов степень многочлена минимальна, при этом многочлен n-ной степени в форме Ньютона для двумерной интерполяции можно представить в виде обобщения одномерного варианта:



На такой конфигурации можно использовать полином либо первой, либо второй степеней. В первом случае нужно выбрать 3 узла в окрестности точки интерполяции и построить по ним полином. Во втором случае на основе выбранных аналогично описанному ранее алгоритму точек и их производных (значение которых нужно вычислить на основе окружающих их узлов) построить полином второй степени.