| 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 (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ

**ОТЧЕТ**

| **по лабораторной работе №** | 1 |
| --- | --- |



Построение и программная реализация алгоритма полиномиальной интерполяции табличных функций

**Дисциплина:** Вычислительные алгоритмы

| Студент | ИУ7И - 46Б |  |  | Андрич К. |
| --- | --- | --- | --- | --- |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | В.М. Градов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

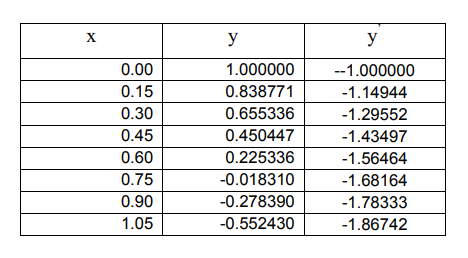
Москва, 2021

Цель работы

Получение навыков построения алгоритма интерполяции таблично заданных функций полиномами Ньютона и Эрмита.

Исходные данные

1. Таблица функции и её производных



1. Степень аппроксимирующего полинома - n
2. Значение аргумента, для которого выполняется интерполяция

Описание алгоритма

1. Алгоритм Ньютона
   1. Вводятся n и x. Потом строится компоновка из n + 1 узлов
   2. Потом строим таблицу которая содержит разделенние разности
   3. Затем строим полином пользуя верхную строку таблици
2. Алгоритм Эрмита
   1. Вводятся n и x. Потом строится компоновка из n + 1 узлов. У нас есть условие что кратност узлов не может быть больше двух
   2. Потом строим таблицу которая содержит разделенние разности. Есть условие что y(x0, x0) = y0'
   3. Затем строим полином пользуя верхную строку таблици
3. Алгоритм для нахождения корня пользуя метод обратной интерполяции
   1. Проверим если корень существует
   2. Поменяем места столбцам x и y
   3. Строим полином Ньютона, с тем что n равно 0

Код программы

В программе есть 5 файлов: 2 заголовочных файла (functions.h и errors.h) и 3 файла кода в СИ (main.c, in\_out.c и methods.c)

Файл: errors.h

| #ifndef ERRORS\_H #define ERRORS\_H  #define OK 0 #define ERR\_IO 1 #define ERR\_ARGS 2 #define ERR\_FILE 3 #define NO\_ROOT 4   #endif //ERRORS\_H |
| --- |

Файл: functions.h

| #ifndef FUNCTIONS\_H #define FUNCTIONS\_H  #include <stdio.h> #include <stdlib.h>  #define N 8  typedef struct {  float x;  float y;  float dy; }data\_t;  int read\_obj(FILE \*f, data\_t \*object); int read\_file(FILE \*f, data\_t \*arr); int input\_exponent(int \*n); int chosen\_x(float \*n, data\_t \*arr); void print\_obj(const data\_t object); void print\_table(data\_t \*arr); int comparator(const void \*q, const void \*p); void check\_beg\_end(int \*b, int \*e, int n); int get\_cur\_pos(data\_t \*arr, float x); float newton(data\_t \*arr, float x, int n); float hermit(data\_t \*arr, float x, int n); int check\_for\_root(data\_t \*arr); void swap\_columns(data\_t \*arr);  #endif //FUNCTIONS\_H |
| --- |
|  |

Файл: in\_out.c

| #include "functions.h" #include "errors.h"  int read\_obj(FILE \*f, data\_t \*object) {  if (fscanf(f, "%f", &object->x) != 1)  return ERR\_IO;  if (fscanf(f, "%f", &object->y) != 1)  return ERR\_IO;  if (fscanf(f, "%f", &object->dy) != 1)  return ERR\_IO;  return OK; }  int read\_file(FILE \*f, data\_t \*arr) {  data\_t cur;  int k = 0;  int rc = OK;  while (k < N)  {  rc = read\_obj(f, &cur);  if (rc == OK)  {  arr[k] = cur;  k++;  }  else  break;  }  return rc; }  int input\_exponent(int \*n) {  printf("\nСтепень плинома: ");  if (scanf("%d", n) != 1)  return ERR\_IO;  if (\*n < 0 || \*n > 4)  return ERR\_IO;  return OK; }  int chosen\_x(float \*n, data\_t \*arr) {  printf("\nВыбарный аргумент для интерполации: ");  if (scanf("%f", n) != 1)  return ERR\_IO;  if (\*n < arr[0].x || \*n > arr[7].x)  return ERR\_IO;  return OK; }  void print\_obj(const data\_t object) {  printf("| %10f ", object.x);  printf("| %9f ", object.y);  printf("| %10f | \n", object.dy); }  void print\_table(data\_t \*arr) {   printf("| x | y | y' | \n");  for(long i = 0; i < N; i++)  print\_obj(arr[i]); } |
| --- |

Файл: methods.c

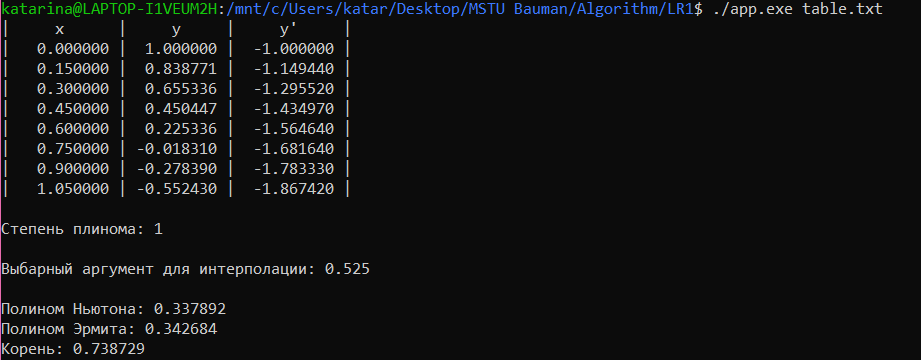
| #include "functions.h" #include "errors.h"  int comparator(const void \*q, const void \*p) {  const data\_t \*a = q;  const data\_t \*b = p;  return a->x - b->x; }  void check\_beg\_end(int \*b, int \*e, int n) {  if (n % 2 == 0)  \*b += 1;  if (\*b < 0)  {  \*b += 1;  \*e += 1;  } }  int get\_cur\_pos(data\_t \*arr, float x) {  int i = 0;  while (i < N)  {  if (arr[i].x > x)  break;  i++;  }  return i; }  float newton(data\_t \*arr, float x, int n) {  float table[N - 2][N - 2], rem, result;  int beginning, end;  int i, k, j = 0;  qsort(arr, N, sizeof(data\_t), comparator);  i = get\_cur\_pos(arr, x);  beginning = i - ((n + 1) / 2) - 1;  end = i + ((n + 1) / 2) - 1;  check\_beg\_end(&beginning, &end, n + 1);  for (i = beginning; i <= end; i++, j++)  table[j][0] = arr[i].y;  for (i = 1; i <= n; i++)  for (j = 0, k = beginning; k < (end - i + 1); j++, k++)  table[j][i] = (table[j][i - 1] - table[j + 1][i - 1]) / (arr[k].x - arr[k + i].x);  result = table[0][0];  rem = x - arr[beginning].x;  beginning++;  for (i = 1; i <= n; i++, beginning++)  {  result += rem \* table[0][i];  rem = rem \* (x - arr[beginning].x);  }  return result; }  float hermit(data\_t \*arr, float x, int n) {  float table[N - 2][N - 2], rem, result;  int beginning, end;  int i, j;  i = get\_cur\_pos(arr, x);  beginning = i - (n + 1) / 4 - 1;  end = i + (n + 1) / 4 - 1;  check\_beg\_end(&beginning, &end, n + 1);  for (i = 0, j = beginning; i <= n; i++)  {  table[i][0] = arr[j].x;  if (i < n)  {  i++;  table[i][0] = arr[j].x;  j++;  }  }  for (i = 0, j = beginning; i < n; i++)  {  if (table[i][0] != table[i + 1][0])  table[i][1] = (arr[j - 1].y - arr[j].y) / (arr[j - 1].x - arr[j].x);  else  {  table[i][1] = arr[j].dy;  j++;  }  }  for (i = 2; i < n + 1; i++)  for (int j = 0; j < n - i + 1; j++)  table[j][i] = (table[j][i - 1] - table[j + 1][i - 1]) / (table[j][0] - table[j + i][0]);  result = arr[beginning].y;  rem = x - table[0][0];  for (i = 1; i < n + 1; i++)  {  result += rem \* table[0][i];  rem = rem \* (rem - table[i][0]);  }  return result; }  int check\_for\_root(data\_t \*arr) {  int i, rc = OK;  for (i = 0; i < N; i++)  if ((arr[i].y \* arr[i + 1].y) < 0)  break;  if (i == N)  rc = NO\_ROOT;  return rc; }  void swap\_columns(data\_t \*arr) {  float buffer;  for (int i = 0; i < N; i++)  {  buffer = arr[i].x;  arr[i].x = arr[i].y;  arr[i].y = buffer;  } } |
| --- |

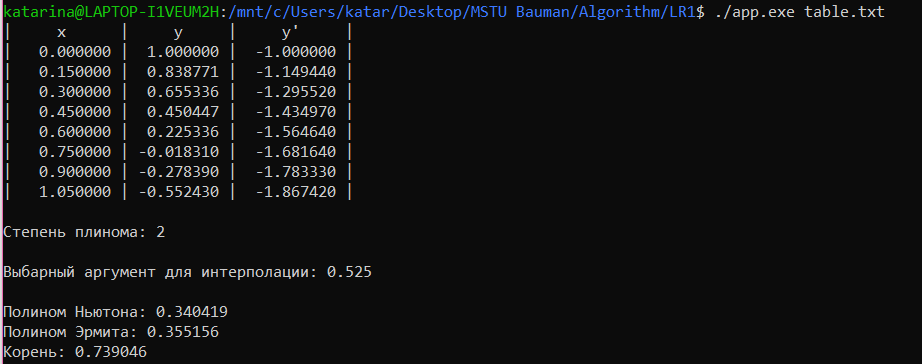
Файл: main.c

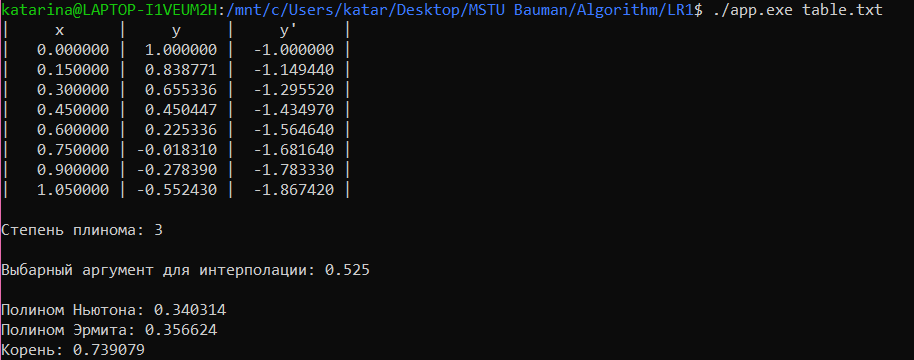
| #include "functions.h" #include "errors.h"  int main(**int** argc, char \*argv[]) {  int rc = OK;  FILE \*f;  data\_t data[N];  int n;  float x, new, her, root;  if (**argc** < 2)  {  printf("Недостаточно количество аргументов\n");  rc = ERR\_ARGS;  }  else   {  f = fopen(**argv**[1], "r+");  if (**f** == NULL)  {  printf("Не возможно октрыть файл\n");  return ERR\_FILE;  }  else  {  rc = read\_file(**f**, data);   if (!rc)  {  print\_table(**data**);  rc = input\_exponent(**&n**);  if (!rc)  {  rc = chosen\_x(**&x**, data);  if (!rc)  {  new = newton(**data**, x, n);  her = hermit(**data**, x, n);  printf("\nПолином Ньютона: %f \n", new);   printf("Полином Эрмита: %f \n", her);  rc = check\_for\_root(**data**);  if (!rc)  {  swap\_columns(**data**);  root = newton(**data**, 0, n);  printf("Корень: %f \n",root);  }  else  printf("Не возможно найти корень\n");  }  else  printf("Неправильный ввод\n");  }  else  printf("Неправильный ввод\n");  }  else  printf("Невозможно прочитать файл\n");  fclose(**f**);  }  }  return rc; } |
| --- |

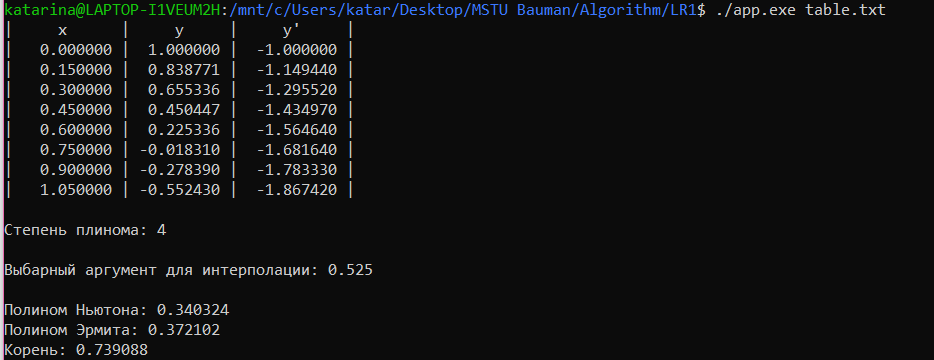
Результаты работы

| Степень | полином Ньютона | полином Эпмита | корень |
| --- | --- | --- | --- |
| 1 | 0.337892 | 0.342684 | 0.738729 |
| 2 | 0.340419 | 0.355156 | 0.739046 |
| 3 | 0.340314 | 0.35624 | 0.739079 |
| 4 | 0.340324 | 0.372102 | 0.739088 |



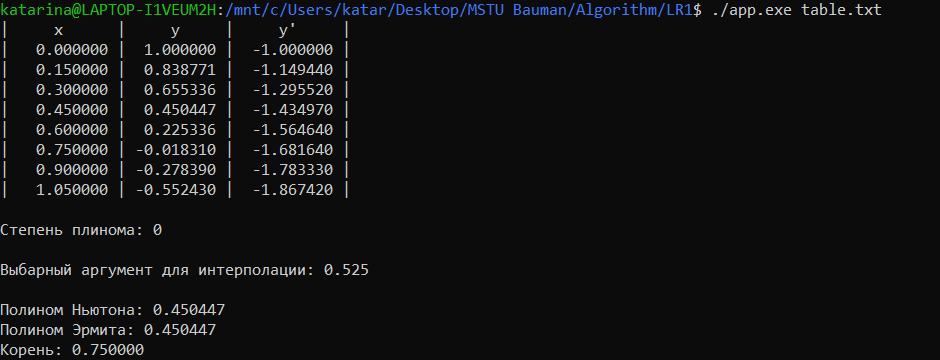






Вопросы при защите лабораторной работы

1. *Будет ли работать программа при степени полинома n=0?*



Видно что программа работает. Результатом стало ближайшее значение функции.

1. *Как практически оценить погрешность интерполяции? Почему сложно применить для этих целей теоретическую оценку?*

Практически можем оценить погрешность интерполяции так что оценим первый отброшенный член. Для этих целей теоретическую оценку сложно применить потому что нам нужно знать производные функции, но они неизвестны.

1. *Если в двух точках заданы значения функции и ее первых производных, то полином какой минимальной степени может быть построен на этих точках?*

Полином Ньютона: 0 и 1 степень

Полином Эрмита: 0, 1, 2 и 3 степень

⇒ минимальная допущена степень полинома - 0

1. *В каком месте алгоритма построения полинома существенна информация об упорядоченности аргумента функции (возрастает, убывает)?*

Информация об упорядоченности аргумента функции существенна в месте построения компоновки узлов. Из них строится таблица разделенных разностей. Если аргументы упорядочены, тогда легко можем найти соседние аргументы к нашему аргументу. Но, если они не упорядочены, тогда надо пробегать по таблице несколько раз.

1. *Что такое выравнивающие переменные и как их применить для повышения точности интерполяции?*

Выравнивающими переменными являются переменные η = η (i), ξ = ξ (k), с помощью которых можно добиться, чтобы график быстро меняющейся функции в новых переменных η (ξ) был близок к прямой, по крайней мере, в некоторых областях. В этом случае интерполяция выполняется по переменным (η, ξ), а затем yi = y (ηi) находится обратной интерполяцией. Однако преобразования η (i), ξ (k) должны быть достаточно простыми, например, логарифмическими). Также должны убедиться, что обратное преобразование и (η) не сложное.