Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

**Решение уравнений с одной переменной**

Лабораторная работа №1 по дисциплине

«Вычислительная математика»

Выполнил:

Студент гр. 434-1

Ю.А. Богомолов

Проверил:

Доцент каф. АСУ

Кандидат технических наук

В.В. Романенко

Оглавление

[1 Введение 3](#_Toc466902024)

[2 Основная часть 4](#_Toc466902025)

[2.1 Методы решения 4](#_Toc466902026)

[2.2 Интервальные методы 5](#_Toc466902027)

[2.3 Итерационные методы 7](#_Toc466902028)

[2.4 Главная функция программы 9](#_Toc466902029)

[2.5 Входные и выходные данные 12](#_Toc466902030)

[3 Заключение 15](#_Toc466902031)

# 1 Введение

В ходе данной лабораторной работы необходимо реализовать ряд методов решения уравнений:

, (1.1)

где – скалярный аргумент функции .

При этом предполагается, что отделение корней уже произведено, т.е. на отрезке находится только одно решение уравнения, т.е. один нуль функции . В этом случае выполняется условие

. (1.2)

Решение должно быть найдено с абсолютной погрешностью по аргументу и/или абсолютной погрешностью по значению функции , т.е.

и/или , (1.3)

где – точное решение уравнения, а – приближённое.

# 2 Основная часть

## 2.1 Методы решения

Имеется несколько методов решения поставленной задачи: метод дихотомии, метод хорд, метод Ньютона, комбинированный метод, метод итераций и метод золотого сечения. Метод итераций и метод Ньютона являются итеративными методами решения, методы дихотомии, хорд, золотого сечения – интервальными. Комбинированный же метод является смесью интервального и итеративного метода. В связи с некоторыми условиями его и метод хорд будем считать итеративными методами.

Заметим, что можно сгруппировать данные методы по типу, и воспользуемся этим. Вспомним, что выбранный язык программирования (C++) является объектно-ориентированным, и попытаемся сделать структуры по типам методов, а потом унаследуем от этих структур структуры самих методов.

## 2.2 Интервальные методы

Общее для интервальных методов – то, что они на каждом шаге делят выбранный интервал на некоторые меньшие интервалы. Метод дихотомии делит интервал на две части, а метод золотого сечения – на три. Поэтому объявим следующую структуру:

*Таблица 2.1 – Листинг 2.1*

|  |
| --- |
| Листинг 2.1 – Родительский класс для интервальных методов |
| struct IntervalByIntervalMethod {  virtual pair<double, double>  operator () (double, double, const char\*) const = 0;  }; |

Объявляем эту структуру без полей для данных, только с методом, т.к. будем передавать её как функтор из главной функции программы в функцию для решения задачи выбранным методом.

*Таблица 2.2 – Листинг 2.2*

|  |
| --- |
| Листинг 2.2 – Классы для каждого из интервальных методов |
| struct IntervalByDichotomyMethod : public IntervalByIntervalMethod {  pair<double, double> operator () (double a, double b, const char\*) const {  double tmp = (a + b) / 2.0;  return make\_pair(tmp, tmp);  }  };  struct IntervalByGoldenSectionMethod : public IntervalByIntervalMethod {  pair<double, double> operator () (double a, double b, const char\*) const {  double d = a + (b - a) / GOLDEN\_SECTION;  double c = a + (b - a) / (GOLDEN\_SECTION \* GOLDEN\_SECTION);  return make\_pair(d, c);  }  }; |

Возвращаемое значение у методов структур – пара точек, которые поделят интервал на несколько частей. Так как метод дихотомии делит интервал только в одной точке, в нём возвращаем две одинаковые точки.

*Таблица 2.3 – Листинг 2.3*

|  |
| --- |
| Листинг 2.3 – Функция для решения задачи интервальным методом |
| tuple<double, double, double> findRootByIntervalMethod(  const char\* expr,  double a,  double b,  double epsx,  double epsy,  const IntervalByIntervalMethod& method  ) {  while (fabs(b - a) / 2 >= epsx || fabs(EvalPolStr(expr, (b + a) / 2.0)) >= epsy)  {  if (b < a)  swap(a, b);  double c, d;  tie(d, c) = method(a, b, expr);  double ev\_a, ev\_d;  ev\_a = EvalPolStr(expr, a);  ev\_d = EvalPolStr(expr, d);  if (ev\_a \* ev\_d <= 0.0)  b = d;  else  a = c;  }  double res = round((b + a) / (2.0 \* epsx)) \* epsx;  return make\_tuple(  res,  fabs(b - a) / 2,  fabs(EvalPolStr(expr, res)));  } |

Эта функция решает задачу выбранным методом, экземпляр структуры которого передаётся в качестве аргумента.

## 2.3 Итерационные методы

Теперь необходимо проделать всё то же самое и для итерационных методов:

*Таблица 2.4 – Листинг 2.4*

|  |
| --- |
| Листинг 2.4 – Решение задачи итерационными методами |
| struct NextStepByIterationsMethod {  // Начальное приближение  virtual double operator () (double a, double b, const char\* expr) const = 0;  // Следующее приближение (со сдвигом границ для  // комбинированного метода)  virtual tuple<double, double, double>  operator () (double a, double b, double x, const char\* expr) const = 0;  };  struct NextStepByChordsMethod : public NextStepByIterationsMethod {  double operator () (double a, double b, const char\*) const {  return (a + b) / 2.0;  }  tuple<double, double, double>  operator () (double a, double b, double, const char\* expr) const  {  double fa = EvalPolStr(expr, a);  double fb = EvalPolStr(expr, b);  double tmp = a - (b - a) \* fa / (fb - fa);  return EvalPolStr(expr, a) \* EvalPolStr(expr, tmp) <= 0.0 ?  make\_tuple(a, tmp, tmp) :  make\_tuple(tmp, b, tmp);  }  };  struct NextStepByNewtonMethod : public NextStepByIterationsMethod {  double operator () (double a, double b, const char\* expr) const {  return EvalPolStr(expr, a, 0) \* EvalPolStr(expr, a, 2) > 0 ? a : b;  }  tuple<double, double, double>  operator () (double a, double b, double x, const char\* expr) const  {  return make\_tuple(a, b,  x - EvalPolStr(expr, x, 0) / EvalPolStr(expr, x, 1));  }  };  struct NextStepByMethodOfIterations : public NextStepByIterationsMethod {  double operator () (double a, double b, const char\*) const {  return (a + b) / 2.0;  }  tuple<double, double, double>  operator () (double a, double b, double x, const char\* expr) const  {  double rx = EvalPolStr(expr, x, 0);  return make\_tuple(a, b, x - rx / max\_der(a, b, expr));  }  };  struct NextStepByCombinedMethod : public NextStepByIterationsMethod {  double operator () (double a, double b, const char\*) const {  return (a + b) / 2.0;  }  tuple<double, double, double>  operator () (double a, double b, double, const char\* expr) const {  static NextStepByNewtonMethod mNewton;  static NextStepByChordsMethod mChords;  double ax = get<2>(mNewton(0, 0, a, expr));  if (isnan(ax))  ax = a;  double bx = get<2>(mNewton(0, 0, b, expr));  if (isnan(bx))  bx = b;  double c, d, unused;  tie(c, d, unused) = mChords(a, b, 0, expr);  if (fabs(c - a) < 1e-10)  c = d;  if (isnan(c))  c = (ax + bx) / 2.0;  return EvalPolStr(expr, a, 0) \* EvalPolStr(expr, a, 2) > 0 ?  make\_tuple(ax, c, (ax + c) / 2.0) :  make\_tuple(c, bx, (c + bx) / 2.0);  }  };  tuple<double, double, double> findRootByIterationsMethod(  const char\* expr,  double a,  double b,  double epsx,  double epsy,  const NextStepByIterationsMethod& method  ) {  double curx = method(a, b, expr), prevx;  do {  prevx = curx;  tie(a, b, curx) = method(a, b, curx, expr);  } while (fabs(curx - prevx) > epsx || fabs(EvalPolStr(expr, curx)) > epsy);  return make\_tuple(  curx,  fabs(curx - prevx),  fabs(EvalPolStr(expr, curx)));  } |

У структур интервальных методов реализуется два метода – поиск начального приближения и поиск следующего приближения. В комбинированном методе и методе хорд при этом сдвигаются границы, так как они являются интервальными методами.

## 2.4 Главная функция программы

Осталось только написать функцию , которая будет считывать входные данные, решать задачу выбранным методом и выводить результат.

*Таблица 2.5 – Листинг 2.5*

|  |
| --- |
| Листинг 2.5 – Главная функция программы |
| int main() {  freopen\_in("input.txt");  freopen\_out("output.txt");  ios\_base::sync\_with\_stdio(false);  cin.tie(NULL);  size\_t testNumber = 0;  int methodType;  while (cin >> methodType >> ws) {  cout << ++testNumber << ". ";  Error = 0;  string expr;  getline(cin, expr);  char\* polStatement = CreatePolStr(expr.c\_str(), 0);  double a, b, epsx, epsy;  cin >> a >> b >> epsx >> epsy;  double ach\_x, ach\_y;  if (Error) {  cout << "Error: " << Error << '\n';  continue;  }  double root = 0.0;  switch (methodType) {  case 1:  // Метод дихотомии  cout << "Dychotomy method:\n";  tie(root, ach\_x, ach\_y) =  findRootByIntervalMethod(polStatement, a, b, epsx,  epsy, IntervalByDichotomyMethod());  break;  case 2:  // Метод хорд  cout << "Chords method:\n";  tie(root, ach\_x, ach\_y) =  findRootByIterationsMethod(polStatement, a, b, epsx,  epsy, NextStepByChordsMethod());  break;  case 3:  // Метод Ньютона  cout << "Newton method:\n";  tie(root, ach\_x, ach\_y) =  findRootByIterationsMethod(polStatement, a, b, epsx,  epsy, NextStepByNewtonMethod());  break;  case 4:  // Комбинированный метод  cout << "Combined method:\n";  tie(root, ach\_x, ach\_y) =  findRootByIterationsMethod(polStatement, a, b, epsx,  epsy, NextStepByCombinedMethod());  break;  case 5:  // Метод итераций  cout << "Iterations method:\n";  tie(root, ach\_x, ach\_y) =  findRootByIterationsMethod(polStatement, a, b, epsx,  epsy, NextStepByMethodOfIterations());  break;  case 6:  // Метод золотого сечения  cout << "Golden section method:\n";  tie(root, ach\_x, ach\_y) =  findRootByIntervalMethod(polStatement, a, b, epsx,  epsy, IntervalByGoldenSectionMethod());  break;  }  cout << "Expression: " << expr << '\n';  if (Error)  cout << "Error: " << Error << '\n';  else {  cout << "x\* = " << defaultfloat << round\_by(root, epsx) << '\n';  double fx = EvalPolStr(polStatement, root, 0);  cout << "f(x\*) = " << defaultfloat << round\_by(fx, epsy) << '\n';  cout << "eps\* by x = " << scientific << ach\_x << '\n';  cout << "eps\* by f(x) = " << scientific << ach\_y << '\n';  }  cout << '\n';  delete[] polStatement;  }  close\_files;  return 0;  } |

## 2.5 Входные и выходные данные

*Таблица 2.6 – Листинг 2.6*

|  |  |
| --- | --- |
| Листинг 2.6 – Входные и выходные данные | |
| input.txt | output.txt |
| 1  sin(x)  3 4  0.001  0.001 | 1. Dychotomy method:  Expression: sin(x)  x\* = 3.142  f(x\*) = -0  eps\* by x = 9.765625e-04  eps\* by f(x) = 4.073464e-04 |
| 1  x - 17.345  0 100  0.00001  0.001 | 2. Dychotomy method:  Expression: x - 17.345  x\* = 17.345  f(x\*) = 0  eps\* by x = 5.960464e-06  eps\* by f(x) = 1.000000e-05 |
| 2  x - 17.345  0 100  0.00001  0.001 | 3. Chords method:  Expression: x - 17.345  x\* = 17.345  f(x\*) = 0  eps\* by x = 0.000000e+00  eps\* by f(x) = 0.000000e+00 |
| 3  x - 17.345  0 100  0.00001  0.001 | 4. Newton method:  Expression: x - 17.345  x\* = 17.345  f(x\*) = 0  eps\* by x = 0.000000e+00  eps\* by f(x) = 0.000000e+00 |
| 4  x - 17.345  0 100  0.00001  0.001 | 5. Combined method:  Expression: x - 17.345  x\* = 17.345  f(x\*) = 0  eps\* by x = 0.000000e+00  eps\* by f(x) = 0.000000e+00 |
| 5  x - 17.345  0 100  0.00001  0.001 | 6. Iterations method:  Expression: x - 17.345  x\* = 17.345  f(x\*) = 0  eps\* by x = 0.000000e+00  eps\* by f(x) = 0.000000e+00 |
| 6  x - 17.345  0 100  0.00001  0.001 | 7. Golden section method:  Expression: x - 17.345  x\* = 17.345  f(x\*) = 0  eps\* by x = 6.344215e-06  eps\* by f(x) = 3.552714e-15 |
| 1  x ^ 2 - 17  0 10  0.00001  0.001 | 8. Dychotomy method:  Expression: x ^ 2 - 17  x\* = 4.12311  f(x\*) = 0  eps\* by x = 9.536743e-06  eps\* by f(x) = 3.607210e-05 |
| 2  x ^ 2 - 17  0 10  0.00001  0.001 | 9. Chords method:  Expression: x ^ 2 - 17  x\* = 4.1231  f(x\*) = -0  eps\* by x = 9.350669e-06  eps\* by f(x) = 5.495296e-05 |
| 3  x ^ 2 - 17  0 10  0.00001  0.001 | 10. Newton method:  Expression: x ^ 2 - 17  x\* = 4.12311  f(x\*) = 0  eps\* by x = 6.664036e-06  eps\* by f(x) = 4.440537e-11 |
| 4  x ^ 2 - 17  0 10  0.00001  0.001 | 11. Combined method:  Expression: x ^ 2 - 17  x\* = 4.12311  f(x\*) = -0  eps\* by x = 4.675325e-06  eps\* by f(x) = 3.115730e-11 |
| 5  x ^ 2 - 17  0 10  0.00001  0.001 | 12. Iterations method:  Expression: x ^ 2 - 17  x\* = 4.12312  f(x\*) = 0  eps\* by x = 7.339839e-06  eps\* by f(x) = 8.627071e-05 |
| 6  x ^ 2 - 17  0 10  0.00001  0.001 | 13. Golden section method:  Expression: x ^ 2 - 17  x\* = 4.1231  f(x\*) = -0  eps\* by x = 7.035842e-06  eps\* by f(x) = 4.639000e-05 |
| 1  x ^ 3 + x ^ 2 - 256  0 10  0.00001  0.001 | 14. Dychotomy method:  Expression: x ^ 3 + x ^ 2 - 256  x\* = 6.03316  f(x\*) = 0  eps\* by x = 9.536743e-06  eps\* by f(x) = 1.285887e-04 |
| 2  x ^ 3 + x ^ 2 - 256  0 10  0.00001  0.001 | 15. Chords method:  Expression: x ^ 3 + x ^ 2 - 256  x\* = 6.03316  f(x\*) = -0  eps\* by x = 4.694097e-06  eps\* by f(x) = 4.295120e-04 |
| 3  x ^ 3 + x ^ 2 - 256  0 10  0.00001  0.001 | 16. Newton method:  Expression: x ^ 3 + x ^ 2 - 256  x\* = 6.03316  f(x\*) = 0  eps\* by x = 7.883359e-06  eps\* by f(x) = 1.187004e-09 |
| 4  x ^ 3 + x ^ 2 - 256  0 10  0.00001  0.001 | 17. Combined method:  Expression: x ^ 3 + x ^ 2 - 256  x\* = 6.03316  f(x\*) = 0  eps\* by x = 1.951062e-11  eps\* by f(x) = 0.000000e+00 |
| 5  x ^ 3 + x ^ 2 - 256  0 10  0.00001  0.001 | 18. Iterations method:  Expression: x ^ 3 + x ^ 2 - 256  x\* = 6.03315  f(x\*) = -0.001  eps\* by x = 3.461598e-06  eps\* by f(x) = 6.879474e-04 |
| 6  x ^ 3 + x ^ 2 - 256  0 10  0.00001  0.001 | 19. Golden section method:  Expression: x ^ 3 + x ^ 2 - 256  x\* = 6.03315  f(x\*) = -0.001  eps\* by x = 7.035842e-06  eps\* by f(x) = 1.084043e-03 |

# 3 Заключение

В ходе выполнения данной лабораторной работы я научился реализовывать методы для решения уравнений вида

, (3.1)

а также попрактиковался в обнаружении закономерностей в схожих методах и написании функторов.