Лабораторная работа №1

Тужилкина Н.Г., БПМ-18-2

# Решение уравнения с одним неизвестным

Пусть дано нелинейное уравнение

— функция, определенная и непрерывная на некотором промежутке.

Требуется найти корень уравнения (1), т.е. число , которое путем его подстановки в (1) превращает уравнение в верное числовое равенство.

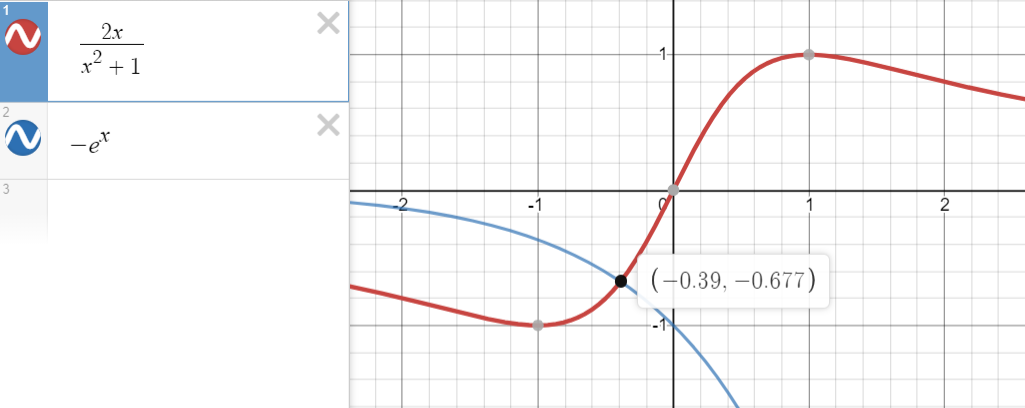


Рисунок . Абсцисса пересечения графиков функций (красный) и (синий) – -0.39

## Цель работы:

1. Изучить методы решения уравнения с одним неизвестным:
   1. Бисекция
   2. Метод секущих
   3. Метод ложной позиции
   4. Метод Ньютона
2. Написать и отладить программу расчёта корней уравнения (1) с точностью , , .
3. Представить результаты в наглядной форме.
4. Сделать выводы о скорости сходимости методов.

### Формальное задание исходных данных

Отрезок, на котором будет производиться поиск корня, остается одинаковым для всех методов.

При положительна. Примем .

При отрицательна. Примем .

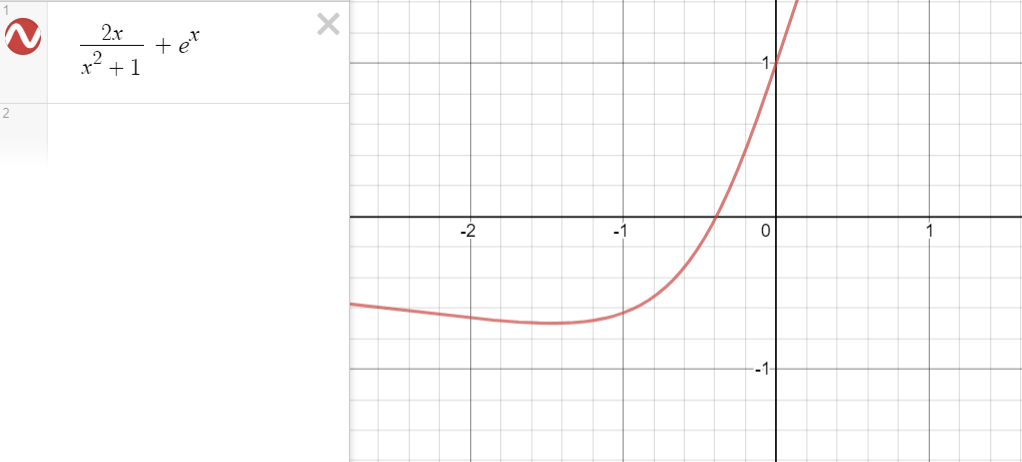


Рисунок . Согласно графику функции , она действительно принимает отрицательное значение в точке -2 и положительное в точке 0

## Бисекция

### Изложение метода

Дано уравнение

Пусть на отрезке функция имеет значения, противоположные по знаку. По следствию из теоремы Больцано–Коши

корень принадлежит отрезку .

Для уточнения положения корня строится система вложенных отрезков, каждый из которых содержит корень уравнения. Для этого находится середина текущего отрезка, и в качестве следующего отрезка из двух возможных выбирается тот, на концах которого функция имеет разные знаки.

Итерации продолжаются до тех пор, пока длина текущего отрезка не станет меньше заданной точности. В качестве приближенного значения корня берется середина последнего отрезка.

### Описание алгоритма

1. Найти начальный отрезок .
2. Найти середину текущего отрезка, :
3. Если , то .

Если , то .

В результате рассматривается новый отрезок .

1. Если – требуемая точность, то процесс завершить. Найти приближенное значение корня по формуле:

Если , положить и перейти к п.2.

### Текст программы

#include <iostream>

#include <math.h>

int sign(const double& x) {

if (x < 0) return -1;

if (x == 0) return 0;

if (x > 0) return 1;

}

double func(const double& x) {

return 2 \* x / (x \* x + 1) + exp(x);

}

double bisect(double a, double b, const double& eps) {

double x(0);

while (b - a > eps) {

x = (a + b) / 2;

(sign(func(a)) != sign(func(x))) ? b = x : a = x;

}

x = (a + b) / 2;

return x;

}

int main() {

bisect(-2, 0, 0.01);

bisect(-2, 0, 0.001);

bisect(-2, 0, 0.0001);

}

### Представление результатов

С точностью :

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг |  |  |  |
| 0 | -2 | 0 | -1 |
| 1 | -1 | 0 | -0.5 |
| 2 | -0.5 | 0 | -0.25 |
| 3 | -0.5 | -0.25 | -0.375 |
| 4 | -0.5 | -0.375 | -0.4375 |
| 5 | -0.4375 | -0.375 | -0.40625 |
| 6 | -0.40625 | -0.375 | -0.390625 |
| 7 | -0.390625 | -0.375 | -0.382813 |
|  | | | |

С точностью :

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг |  |  |  |
| 0 | -2 | 0 | -1 |
| 1 | -1 | 0 | -0.5 |
| 2 | -0.5 | 0 | -0.25 |
| 3 | -0.5 | -0.25 | -0.375 |
| 4 | -0.5 | -0.375 | -0.4375 |
| 5 | -0.4375 | -0.375 | -0.40625 |
| 6 | -0.40625 | -0.375 | -0.390625 |
| 7 | -0.390625 | -0.375 | -0.382813 |
| 8 | -0.390625 | -0.382813 | -0.386719 |
| 9 | -0.390625 | -0.386719 | -0.388672 |
| 10 | -0.390625 | -0.388672 | -0.389648 |
|  | | | |

С точностью :

|  |  |  |  |
| --- | --- | --- | --- |
| *Шаг* |  |  |  |
| 0 | -2 | 0 | -1 |
| 1 | -1 | 0 | -0.5 |
| 2 | -0.5 | 0 | -0.25 |
| 3 | -0.5 | -0.25 | -0.375 |
| 4 | -0.5 | -0.375 | -0.4375 |
| 5 | -0.4375 | -0.375 | -0.40625 |
| 6 | -0.40625 | -0.375 | -0.390625 |
| 7 | -0.390625 | -0.375 | -0.382813 |
| 8 | -0.390625 | -0.382813 | -0.386719 |
| 9 | -0.390625 | -0.386719 | -0.388672 |
| 10 | -0.390625 | -0.388672 | -0.389648 |
| 11 | -0.390625 | -0.389648 | -0.390137 |
| 12 | -0.390137 | -0.389648 | -0.389893 |
| 13 | -0.390137 | -0.389893 | -0.390015 |
| 14 | -0.390137 | -0.390015 | -0.390076 |
|  | | | |

### Оценка точности результатов

Точность 0.01 достигается после 8-ми итераций, точность 0.001 — после 11-ти итераций. Точность 0.0001 достигнута после 15-ти итераций.

### Скорость сходимости

Середина -го отрезка – точка – дает приближение к корню , имеющее оценку погрешности

Следовательно, метод бисекции сходится со скоростью геометрической прогрессии, знаменатель которой – . По сравнению с другими методами метод бисекции сходится довольно медленно.

## Метод секущих

### Изложение метода

Секущая ­— прямая, проходящая через две точки на графике функции. Положим, что у нас есть две точки, , в которых значения функции равны соответственно . Тогда уравнение прямой, проходящей через эти точки, будет

Для точки пересечения с осью абсцисс получим уравнение

В качестве приближений к корню уравнения принимаются последовательные значения точек пересечения секущей с осью абсцисс.

Метод работает и в случае, если начальные точки выбраны по одну и ту же сторону от корня (то есть, корня нет на отрезке между начальными приближениями), но при этом возможны случаи, когда метод не сходится.

Метод секущих является двухшаговым, то есть новое приближение определяется двумя предыдущими итерациями.

### Описание алгоритма

1. Задать два начальных приближения – .
2. Приближения к корню находятся по итерационной формуле:
3. Продолжать вычисления, пока не станет меньше или равно заданному значению погрешности.

### Текст программы

#include <iostream>

#include <math.h>

#define min(a,b) ((a) <= (b) ? (a) : (b))

#define max(a,b) ((a) > (b) ? (a) : (b))

double func(const double& x) {

return 2 \* x / (x \* x + 1) + exp(x);

}

double secant(double x1, double x2, const double& eps) {

double x3(0);

while (abs(x2 - x1) > eps) {

x3 = x2 - func(x2) \* (x2 - x1) / (func(x2) - func(x1));

x1 = x2;

x2 = x3;

}

return x3;

}

int main() {

secant(-2, 0, 0.01);

secant(-2, 0, 0.001);

secant(-2, 0, 0.0001);

}

### Представление результатов

С точностью :

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг |  |  |  |
| 0 | -2 | 0 | -1.20144 |
| 1 | -1.20144 | 0 | -0.714026 |
| 2 | -1.20144 | -0.714026 | 0.267761 |
| 3 | -0.714026 | 0.267761 | -0.516113 |
| 4 | -0.516113 | 0.267761 | -0.431621 |
| 5 | -0.516113 | -0.431621 | -0.384428 |
| 6 | -0.431621 | -0.384428 | -0.390237 |
|  | | | |

С точностью :

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг |  |  |  |
| 0 | -2 | 0 | -1.20144 |
| 1 | -1.20144 | 0 | -0.714026 |
| 2 | -1.20144 | -0.714026 | 0.267761 |
| 3 | -0.714026 | 0.267761 | -0.516113 |
| 4 | -0.516113 | 0.267761 | -0.431621 |
| 5 | -0.516113 | -0.431621 | -0.384428 |
| 6 | -0.431621 | -0.384428 | -0.390237 |
| 7 | -0.390237 | -0.384428 | -0.390018 |
|  | | | |

С точностью :

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг |  |  |  |
| 0 | -2 | 0 | -1.20144 |
| 1 | -1.20144 | 0 | -0.714026 |
| 2 | -1.20144 | -0.714026 | 0.267761 |
| 3 | -0.714026 | 0.267761 | -0.516113 |
| 4 | -0.516113 | 0.267761 | -0.431621 |
| 5 | -0.516113 | -0.431621 | -0.384428 |
| 6 | -0.431621 | -0.384428 | -0.390237 |
| 7 | -0.390237 | -0.384428 | -0.390018 |
| 8 | -0.390237 | -0.390018 | -0.390017 |
|  | | | |

### Оценка точности результатов

Точность 0.01 достигается после 7-ти итераций, точность 0.001 — после 8-ми итераций. Точность 0.0001 достигнута после 9-ой итераций.

### Скорость сходимости

Показано, что метод секущих сходится к простому корню сверхлинейно со скоростью

Эта скорость достаточно высока для того, чтобы обеспечить методу широкую область применения.

## Метод ложной позиции

### Изложение метода

Нелинейная функция заменяется линейной функцией на интервале , и корень линейного уравнения берется в качестве следующего приближения к корню нелинейного уравнения . Корень , – не корень . В итоге получаются два интервала, и . Следующее приближение находится по формуле:

Метод ложной позиции можно рассматривать как комбинацию метода секущих и метода дихотомии — в методе секущих в качестве точек следующей итерации выбираются последние рассчитанные точки, в данном методе выбираются те точки, в которых функция имеет разный знак.

### Описание алгоритма

1. Определить отрезок , такой, что .
2. Найти приближение по формуле
3. В качестве следующего рассматриваемого отрезка выбрать отрезок, или , на котором функция меняет знак.
4. При заданной точности вычисления нужно вести до тех пор, пока длина очередного рассматриваемого отрезка не станет меньше или равна .

### Текст программы

#include <iostream>

#include <math.h>

#include <iomanip>

double func(const double& x) {

return 2 \* x / (x \* x + 1) + exp(x);

}

double false\_pos(double x0, double x1, const double& eps) {

double delx(x1 - x0), x(0);

double f0(func(x0)), f1(func(x1));

while (delx > eps) {

x = x1 \* f0 / (f0 - f1) + x0 \* f1 / (f1 - f0);

double f(func(x));

if (f0 \* f > 0) {

delx = x - x0;

x0 = x;

f0 = f;

}

else {

delx = x1 - x;

x1 = x;

f1 = f;

}

}

return x;

}

int main() {

false\_pos(-2, 0, 0.01);

false\_pos(-2, 0, 0.001);

false\_pos(-2, 0, 0.0001);

}

### Представление результатов

С точностью :

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг |  |  |  |
| 0 | -2 | 0 | -1.20144 |
| 1 | -1.20144 | 0 | -0.714026 |
| 2 | -0.714026 | 0 | -0.490347 |
| 3 | -0.490347 | 0 | -0.416187 |
| 4 | -0.416187 | 0 | -0.396396 |
| 5 | -0.396396 | 0 | -0.391542 |
|  | | | |

С точностью :

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг |  |  |  |
| 0 | -2 | 0 | -1.20144 |
| 1 | -1.20144 | 0 | -0.714026 |
| 2 | -0.714026 | 0 | -0.490347 |
| 3 | -0.490347 | 0 | -0.416187 |
| 4 | -0.416187 | 0 | -0.396396 |
| 5 | -0.396396 | 0 | -0.391542 |
| 6 | -0.391542 | 0 | -0.39038 |
| 7 | -0.39038 | 0 | -0.390103 |
|  | | | |

С точностью :

|  |  |  |  |
| --- | --- | --- | --- |
| Шаг |  |  |  |
| 0 | -2 | 0 | -1.20144 |
| 1 | -1.20144 | 0 | -0.714026 |
| 2 | -0.714026 | 0 | -0.490347 |
| 3 | -0.490347 | 0 | -0.416187 |
| 4 | -0.416187 | 0 | -0.396396 |
| 5 | -0.396396 | 0 | -0.391542 |
| 6 | -0.391542 | 0 | -0.39038 |
| 7 | -0.39038 | 0 | -0.390103 |
| 8 | -0.390103 | 0 | -0.390037 |
|  | | | |

### Оценка точности результатов

Точность 0.01 достигается после 5-ти итераций, точность 0.001 — после 7-ми итераций. Точность 0.0001 достигнута после 8-ми итераций.

### Скорость сходимости

Метод ложного положения обладает только линейной сходимостью. Сходимость тем выше, чем меньше отрезок .

## Метод Ньютона

### Изложение метода

Задается начальное приближение и проводится касательная к кривой в точке . В качестве следующего приближения выбирается точка пересечения этой касательной с осью абсцисс. Процесс построения касательных и нахождения точек пересечения с осью абсцисс повторяется до тех пор, пока приращение не станет меньше заданной величины .

### Описание алгоритма

1. Задать начальное приближение так, чтобы выполнялось неравенство , положить .
2. Вычислить по формуле
3. Если , процесс завершить и положить

Если , положить и перейти к п.2.

### Текст программы

#include <iostream>

#include <math.h>

#include <iomanip>

double func(const double& x) {

return 2 \* x / (x \* x + 1) + exp(x);

}

double deriv(const double& x) {

return 2 / (x \* x + 1) - 4 \* x \* x / pow(x \* x + 1, 2) + exp(x);

}

double newton(double x0, const double& eps) {

double delx(eps + 1), x(x0);

while (delx > eps) {

x = x0 - func(x0) / deriv(x0);

delx = abs(x - x0);

x0 = x;

}

return x;

}

int main() {

newton(0, 0.01);

newton(0, 0.001);

newton(0, 0.0001);

}

### Представление результатов

С точностью :

|  |  |  |
| --- | --- | --- |
| Шаг |  |  |
| 0 | 0 | -0.333333 |
| 1 | -0.333333 | -0.38737 |
| 2 | -0.38737 | -0.39001 |
|  | | |

С точностью :

|  |  |  |
| --- | --- | --- |
| Шаг |  |  |
| 0 | 0 | -0.333333 |
| 1 | -0.333333 | -0.38737 |
| 2 | -0.38737 | -0.39001 |
| 3 | -0.39001 | -0.390017 |
|  | | |

С точностью :

|  |  |  |
| --- | --- | --- |
| Шаг |  |  |
| 0 | 0 | -0.333333 |
| 1 | -0.333333 | -0.38737 |
| 2 | -0.38737 | -0.39001 |
| 3 | -0.39001 | -0.390017 |
|  | | |

### Оценка точности результатов

При начальном приближении точность 0.01 достигается после 3-х итераций, точность 0.001 и 0.0001 — после 2-х итераций.

Начальное приближение не удовлетворяет условию , однако всё же можно достичь точности 0.01 после 257 итераций и 0.001, 0.0001 после 258.

### Скорость сходимости

Данный метод быстро сходится (имеет квадратичную сходимость) и допускает различные модификации. Однако этот метод эффективен при весьма жестких ограничениях на характер функции :

1. существование второй производной на множестве ;
2. удовлетворение первой производной условию ;
3. знакопостоянство .

Поэтому его желательно использовать совместно с другими методами, чтобы достигнуть диапазона, где указанные условия начинают выполняться.

## Вывод

Метод бисекции применим для любой непрерывной функции. Достоинства данного метода — высокая надежность и простота. Однако он неприменим для корней четной кратности и также не может быть обобщен на случай комплексных корней и на системы уравнений.

Метод Ньютона является, возможно, самым популярным методом решения нелинейных уравнений. В данном методе ошибка убывает быстрее, чем в остальных методах, поскольку у него скорость сходимости квадратичная. Недостатки метода Ньютона — локальность и ограничения на характер функции, а также необходимость вычисления производных на каждом шаге. Если функция сложна, то аналитическое дифференцирование сопряжено со значительными трудностями.

В таком случае пользуются методом секущих. Поскольку знание производной не требуется, то при том же объёме вычислений в методе секущих (несмотря на меньший порядок сходимости) можно добиться большей точности, чем в методе Ньютона.

Метод ложного положения отличается от метода секущих только тем, что всякий раз берутся не последние 2 точки, а те точки, которые находятся вокруг корня.

Для решения уравнений часто применяются комбинированные алгоритмы, сочетающие достоинства различных методов.