Выполнил:  
Игнатьев Даниил  
Студент группы 3630102/80004

Отчет

первая лабораторная работа по численным методам

Даниил Игнатьев

2019

Оглавление

[1. Формулировка задания: 2](#_Toc22899551)

[1.1. Алгоритмы: 2](#_Toc22899552)

[1.2.Анализ задач 3](#_Toc22899553)

[2. Тестовый пример 4](#_Toc22899554)

[2.1. МПД 4](#_Toc22899555)

[2.2. Метод Ньютона 4](#_Toc22899556)

[3. Модульная структура программы: 5](#_Toc22899557)

[4. Контрольные тесты 6](#_Toc22899558)

[5. Численный анализ решения: 7](#_Toc22899559)

[6. Вывод: 8](#_Toc22899560)

# Формулировка задания:

Решить уравнения - трансцендентное и полиномиальное - методами половинного деления и простых итераций, и исследовать зависимость числа итераций от точности и начального приближения

## 1.1. Алгоритмы:

Метод половинного деления

Метод половинного деления трудно использовать в функции, в которой корни расположены плотно друг к другу. В том случае если корней на входном отрезке больше одного и они в разных половинах отрезка, то данный метод найдет только один корень. В противном случае, может не найти вовсе.

Требуемые условия: Наличие единственного корня на входном отрезке.

Шаг 1: проверить выполнения требуемых условий, если не выполняются перейти к шагу 6

Шаг 2: вычислить длину отрезка, если она удовлетворяет необходимой точности - перейти к шагу 5

Шаг 3: вычислить середину отрезка, проверить значение функции в середине отрезка и заменить один из концов отрезка серединой, тот который имеет знак значения функции такой же, как и у значения в середине. Если значение функции в середине отрезка равно 0 перейти к шагу 5

Шаг 4: перейти к шагу 2

Шаг 5: ответ - середина отрезка

Шаг 6: конец

Метод Ньютона

Метод Ньютона (метод касательных) применяется в том случае, если уравнение *f(x) = 0* имеет корень http://statistica.ru/upload/medialibrary/chisl-methods-resh-ur/11.png, и выполняются условия:

1) функция *y= f(x)* непрерывно дифференцируема на интервале {\displaystyle (a,\,b)} http://statistica.ru/upload/medialibrary/chisl-methods-resh-ur/11.png

2) *f(a)·f(b) < 0* (функция принимает значения разных знаков на концах отрезка [*a;b*]);

3) производные *f'(x)* и *f''(x)* сохраняют знак на отрезке [*a;b*] (т.е. функция *f(x)* либо возрастает, либо убывает на отрезке [*a;b*], сохраняя при этом направление выпуклости);

4) 𝑓(𝑥0)∗𝑓′′(𝑥0)>0

Шаг 1: Задается начальное приближение {\displaystyle x\_{0}}x­0

Шаг 2: вычисляют новое приближение: {\displaystyle x\_{n+1}=x\_{n}-{\frac {f(x\_{n})}{f'(x\_{n})}}}

(Выполняется с условием того, что )

Шаг 3: вычислить длину отрезка, если она удовлетворяет необходимой точности - перейти к шагу 4, иначе – к шагу 2

Где

Шаг 4: ответ - xn+1

## Анализ задач

Полином

Рассмотрим полином x^4 - x^3 - 2x^2 + 3x – 3 = 0.   
Верхняя граница  
Делим полином на старший коэффициент, после среди коэффициентов ищем максимальный коэффициент по абсолютному значению и индекс первого отрицательного коэффициента. После вычисляем по формуле . Далее аналогично вычисляем нижнюю границу для –х.   
После для заменяем x на и аналогично вычисляем для и   
Получаем что, его корни лежат на отрезке [-2.73, -0.59] и [0.5, 4].   
Решив уравнение вручную, получаем, что оно имеет корни +-.

Трансцендентное ур-ие

Уравнение 3 + 5x + 2 = 0 имеет один корень -0.69.  
Следовательно, подходит отрезок [-2; 4].  
(на этом промежутке функция имеет непрерывные первую у вторую производные и первая производная сохраняет знак).

# Тестовый пример

## 2.1. МПД

Возьмем простое уравнение y = x+2. Его корень = -2.  
Протестируем на нем метод половинного деления на отрезке [-4,4] :

X1 = -3;

X2 = 0;

1-ая итерация: x1 = -3; x2 = -1.5;

2-ая итерация: x1 = -2.25; x2 = -1.5;

3-ая итерация: x1 = -2.25; x2 = -1.875;

С каждой итерацией границы приближаются к корню. Значит, алгоритм работает.

## 2.2. Метод Ньютона

Возьмем простое уравнение y = x^2. Его корень = 0.  
Протестируем на нем метод половинного деления на отрезке [-3,1] .

Производная = 2х.  
X0 = -3;  
  
1-ая итерация: х0 = -3 - 9/(-6) ; x0 = -1.5;

2-ая итерация: x0 = -1.5 – 2.25/(-3); x0 = -0.75

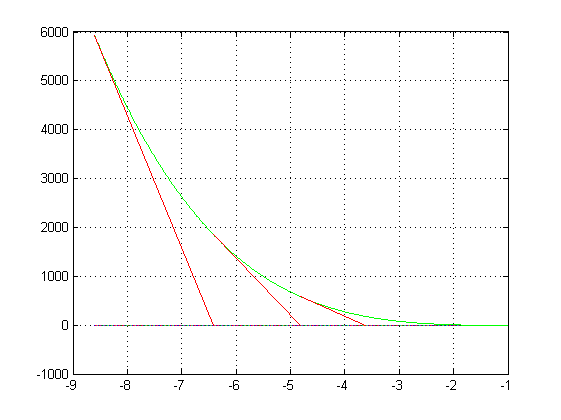
3-ая итерация: x0 = -0.75 – 0.5625/(-1.5); x0 = -0.375;

С каждой итерацией границы приближаются к корню. Значит, алгоритм работает.  
Пример работы метода Ньютона:

1-ая итерация

3-я итерация

2-ая итерация



# Модульная структура программы:

Программа состоит из вызовов четырех функций:

Входные данные:

Double x1,x2 – границы отрезка, на котором находится корень.  
double E – требуемая точность.

Int \*poly – массив, содержащий целочисленные коэффициенты полинома.

function method\_half\_partition(double x1, double x2, double E, int\* poly)  
Ищет корень методом половинного деления с начальными границами х1 и х2 и с точностью Е.  
В отличии от метода Ньютона (в основном, из-за производных), мпд легко пишется для любой функции. Возвращает корень.  
Функция способна считать как полиномиальное ур-ние, так и трансцендентное.  
Во втором случае вместо указателя на массив коэффициентов передается нулевой указатель.Записывает в файл кол-во итераций.

function method\_Newton(double x1, double x2, double E, int\* poly)

Ищет корень полинома методом Ньютона с начальными границами х1 и х2 и с точностью Е.  
Возвращает значение Х.  
Записывает в файл кол-во итераций.

function method\_Newton\_trans(double x0, double x1, double E)

Ищет корень трансцендентного уравнения методом Ньютона с начальными границами х0 и х1 и с точностью Е.  
Возвращает значение Х.  
Записывает в файл кол-во итераций.

function Counting\_poly(double x, int\* poly)  
Функция, считающая значение любого полинома при заданном Х.  
Возвращает результат ф-ии.  
  
Также используются макроподстановки для подсчета значений трансцендентных ф-ий и их производных:  
#define Counting\_transcendent(x) ( 3\*pow(e,x) + 5\*x + 2)   
#define derivate\_transcendent(x) (3\*pow(e,x) + 5)

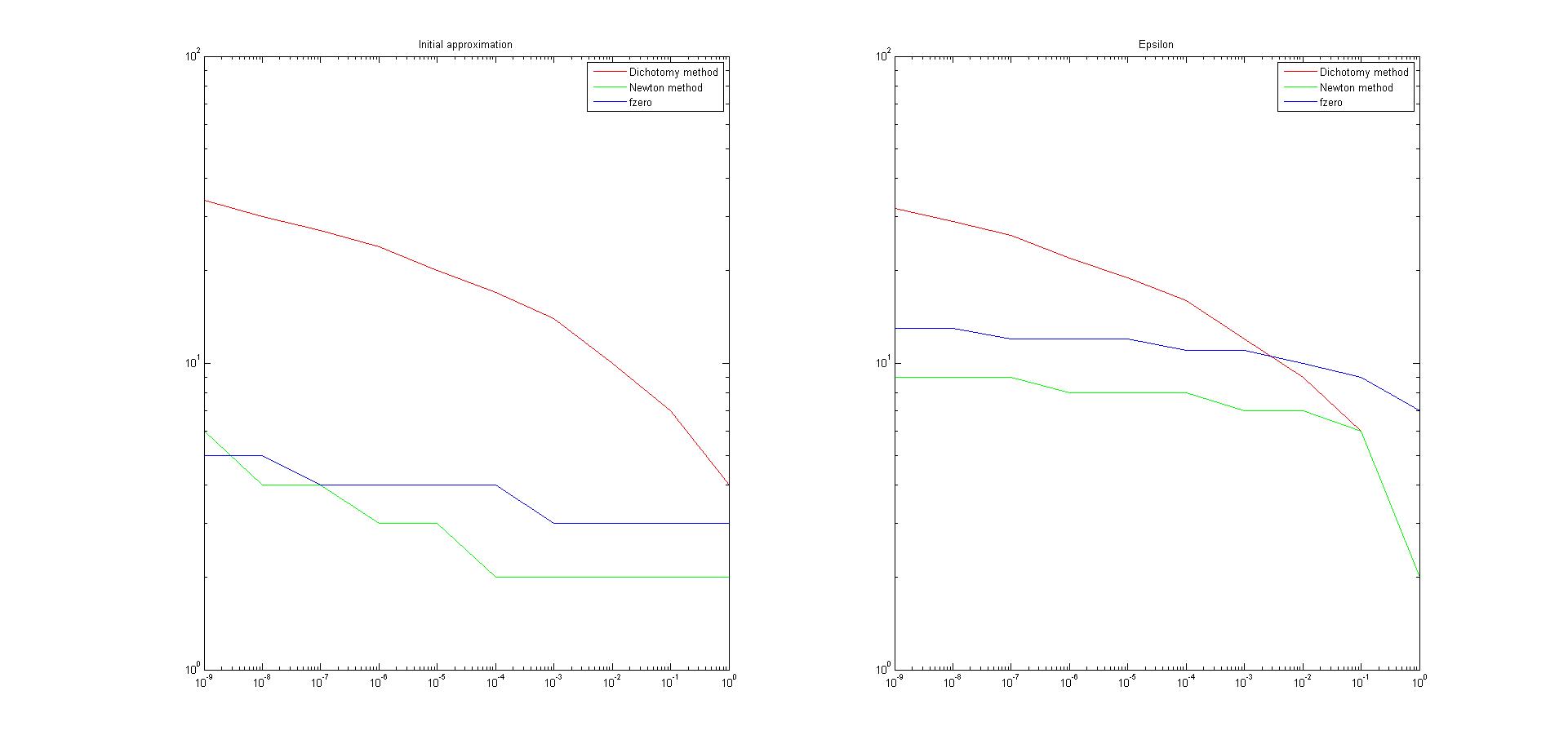
# 4. Контрольные тесты

Полиномиальное уравнение .  
Для поиска зависимости кол-ва итераций от необходимой точности будем рассматривать исходную функцию с начальным приближением x = -5 на промежутке [-5, -1] (на этом промежутке производная функции сохраняет положительный знак).

Трансцендентное уравнение .   
Для поиска зависимости кол-ва итераций от необходимой точности будем рассматривать исходную функцию с начальным приближением x (0) = -2 на промежутке [-2, 4] (на этом промежутке производная функции сохраняет положительный знак).

Для поиска зависимости кол-ва итераций от начального приближения будем рассматривать функцию на том же промежутке. Построим последовательность приближений от действительного корня, вычисленного предварительно с помощью функции fzero() и приблизительно равного x\* ≈ -0.69842.

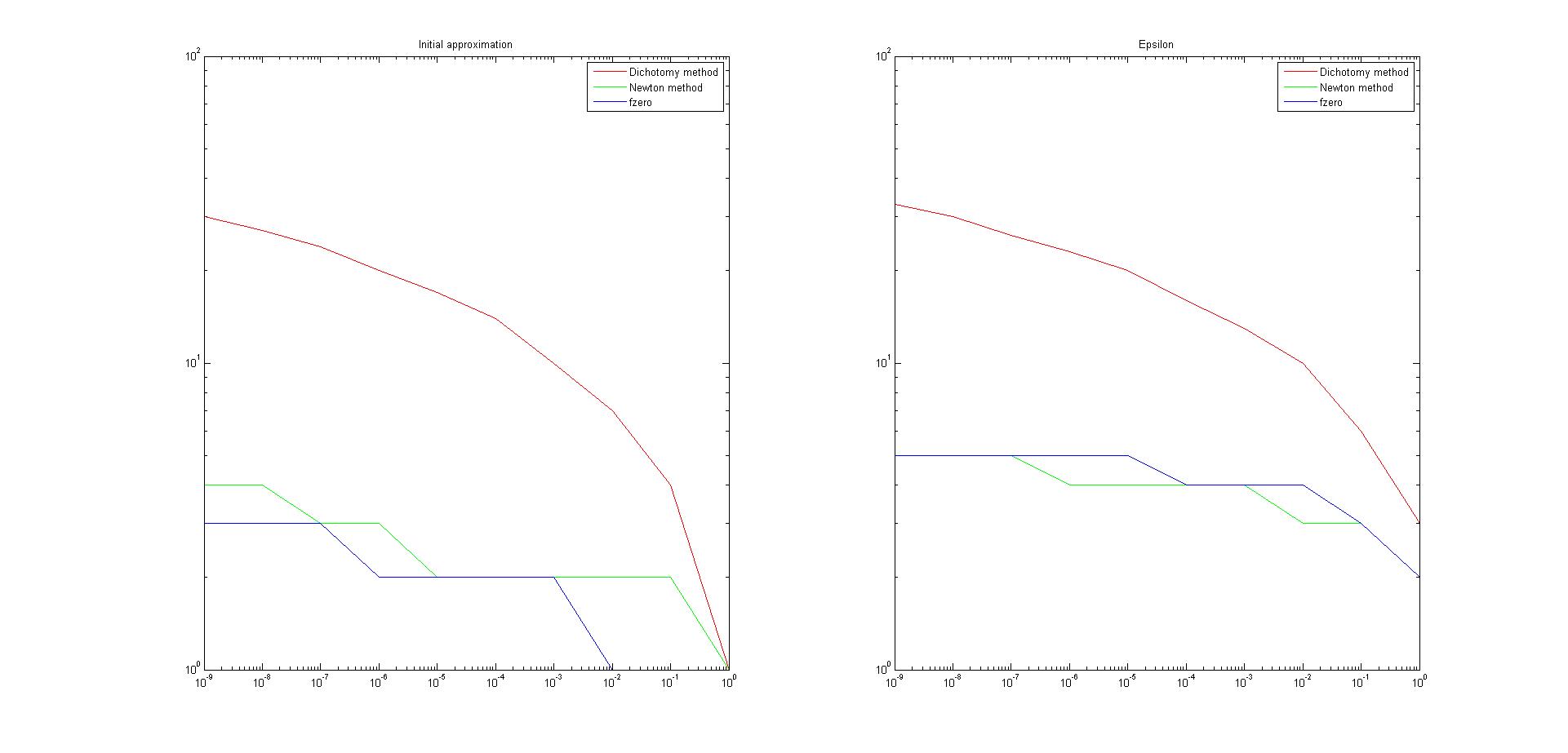
Для обоих уравнений   
Приближение к корню вычисляется по формуле: x\* 0.5\*  
Требуемая точность вычисляется по формуле:   
Где i изменятся от 0 до 9 с шагом 1 в зависимости от номера теста.

5. Численный анализ решения:

Здесь представлены графики исследования полиномиальной ф-ии.

Анализируя зависимость от начального приближения, видим, что, как и характерно для МПД, он показывает логарифмическую зависимость (в логарифмических координатах).

Из исследований видно, что, начиная с расстояния до корня, большего 10^-9 метод Ньютона работает быстрее fzero.  
  
Однако для точности, стремящейся к 1, метод Ньютона и МПД сходятся в одной точке и превосходят по скорости fzero.



Здесь представлены графики исследования трансцендентной ф-ии.  
  
Аналогично с предыдущим примером, МПД показывает логарифмическую зависимость (в логарифмических координатах).

При изменяющемся расстоянии до корня fzero совпадает либо превосходит метод Ньютона, однако для изменяющейся точности ситуация обратная. Возможно, что начиная с точности 10^-7 и меньше, зависимости Метода Ньютона и fzero эквивалентны.  
  
  
Дополнение: пример вызова ф-ии fzero для подсчета кол-ва итераций в зависимости от расстояния до корня и последующее сохранение значения.  
 «options = optimset('TolX', 10^-9);

[h, f, exitflag, output] = fzero('x^4 - x^3 -2\*x^2 +3\*x - 3', (-1.73205080757 - 0.5\*10^-i ) );  
 y(i) = output.iterations;»  
(Для зависимости итераций от приближения к корню)

«options = optimset('TolX', 10^-i);

[h, f, exitflag,output] = fzero('3\*exp(x) + 5\*x + 2',-2, options);

y(i) = output.iterations;»

(Для зависимости итераций от точности)

# 6. Вывод:

МПД - прост и интуитивно понятен. Этот метод обеспечивает гарантированную сходимость метода независимо от сложности функции (при выполненных необходимых условиях) - и это весьма важное свойство.   
Недостатком метода является то же самое - метод никогда не сойдется быстрее, т.е. сходимость метода всегда равна сходимости в наихудшем случае.

Метод Ньютона – немного сложнее реализуем для сложных функций.  
На высоких порядках точности работает быстрее МПД.   
Недостатки:   
1) если начальное приближение недостаточно близко к решению, то метод может не сойтись.  
2) Если не существует вторая [производная](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8) в точке корня, то [скорость сходимости](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D0%BE%D1%80%D0%BE%D1%81%D1%82%D1%8C_%D1%81%D1%85%D0%BE%D0%B4%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8) метода может быть заметно снижена.  
  
Следовательно, для общего случая лучше использовать метод Ньютона, а в случае отсутствия строгих требований к точности (хватит нескольких знаков после запятой) рациональным становится и использование МПД. Но из всех трех самым приоритетным по количеству итераций является fzero, так как эта функция с большой точностью оценивает ситуацию и подбирает верный метод нахождения корня.