Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

Институт прикладной математики и механики

Курсовой проект по дисциплине

«Численные методы»

Выполнил студент

гр.3630102/90003 А.К. Белоногов

Преподаватель С.Б. Добрецова

Санкт-Петербург

2021

**Оглавление**

**задание 3**

**алгоритмы решения 3**

**условия применимости и проверка 4**

**тестовый пример 4**

**контрольные тесты 6**

**Модульная структура программы 6**

**Численный анализ 7**

**Вывод 8**

# **Лабораторная работа 1. Численные методы решения алгебраических и трансцендентных уравнений**

## **Задание:**

Найти решение функций с помощью метода половинного деления и метода простых итераций.

Функции:

1. x^4-18\*x-10 = 0
2. 5^x-6\*x=7

## **Задача:**

## 1)Исследовать применимость метода и его свойства.

## 2)Решить алгебраическое и трансцендентное уравнения, используя данные методы, сравнить их эффективность, подтвердив численным экспериментом достоверность решения.

## 3)Проверить решение, используя средствами пакета MATLAB.

## 

## **Метод половинного деления**

Условия применимости:

Если функция f(x) непрерывна на отрезке [a,b] и на его концах принимает разные знаки, то метод половинного деления сходиться к точному решению уравнения.

Алгоритм метода:

Допустим, что единственный корень уравнения лежит на отрезке [a,b].

a0 = a

b0 = b

1. Расчет середины отрезка [аk,bk].

ck-1 = (ak-1 + bk-1)/2

При k = 1,2…

1. k-итерация (k=1,2,3…). Сужение отрезка [аk-1,bk-1].

Если f(аk-1)\*f(c k-1)<0, то аk = а k-1, bk **=** c k-1

Если f(а k-1)\*f(c k-1)>0, то аk = c k-1, bk **=** b k-1

1. Расчет середины отрезка [аk,bk].

ck = (ak + bk)/2

1. Проверка условия итерационного процесса. Если условие выполнено, то сk – корень уравнения:

| сk - сk-1 | < ℇ

Иначе перейти к 2 пункту.

**Метод простых итераций**

**Условие применимости:**

Если функция f(x) непрерывна на отрезке [a,b], Достаточное условие сходимости таково:

**Алгоритм метода:**

1. Задаем начальное приближение отрезка, на нулевой итерации и находим константу:

x0 = а

1. Проверка условия сходимости |φ`(х)|<1
2. Выражаем х

x1 = φ(x0) – приближение корня на первой итерации

1. Проверка условия остановки итерационного процесса, если условие

выполнено, то x1 – корень уравнения:

| x1 - x0 | < ℇ

Иначе перейти к 5 пункту.

1. k-итерация (k=2,3,4…) аналогична 3 пункту. Раcсчет приближения корня k-ой итерации.

xk = φ(xk-1)

1. Проверка условия остановки итерационного процесса, если условие выполнено, то xk – корень уравнения:

| xk - xk-1 | < ℇ

Иначе перейти к 4 пункту.

Анализ задачи

Теорема и верхней границе:

Для функции – полинома существует теорема о верхней границе корней, по которой можем посчитать максимальную/минимальную границы положительных/отрицательных корней.

Найдем верхнюю границу положительного x\* < 1 + (a/a0)^(1/m)

где 𝑎 – максимальный модуль из отрицательных коэффициентов при переменных уравнения.

𝑎0 > 0 - коэффициент при переменной в старшей степени (если он меньше нуля, то умножим всё уравнение на -1).

m – номер первого отрицательного коэффициента (из 𝑎0 , 𝑎1 ,…).

Для нашего алгебраического уравнения:

𝑎 = 18

𝑎0 = 1

𝑚 = 1

Следовательно x\* < 1 + (18/1)^(1/1) = 19

**Т****естовый пример к методам**

1. Решим с помощью метода половинного деления уравнение

x^2 + x -6= 0

Корнями этого уравнения являются значения -3 и 2. Найдем второй корень с помощью метода половинного деления на промежутке [0,3].

Проверим лежит ли в этом промежутке наш корень, т.е. функция принимает значение разных знаков на концах промежутков.

Формула для проверки условия: f(а)\*f(b) < 0

a0=0

b0=3

f(0) = 0 – 6 = = -6 < 0

f(3) = 9 + 3 – 6 = 6 > 0

Функция принимает значение разных знаков на концах промежутков. Следовательно, мы можем продолжить решение.

x0 = (a0 + b0)/2 = (0+3)/2 = 1.5

Получилось два промежутка. Из них надо выбрать тот на котором функция принимает значение разных знаков.

F(1.5) = 2.25 +1.5 – 6 = -2.25 < 0

Следовательно, промежуток [1.5, 3] нам подходит т.к f(1.5)\*f(3) < 0.

a1=1.5

b1=3

x1 = (a1 + b1)/2 = (1.5+3)/2 = 2.25

Получилось два промежутка. Из них надо выбрать тот на котором функция принимает значение разных знаков.

f(2.25) = 5.0625 + 2.25 – 6 = 1.3125 > 0

Следовательно, промежуток [1.5, 2.25] нам подходит т.к f(1.5)\*f(2.25) < 0.

a2=1.5

b2=2.25

x2 = (a2 + b2)/2 = (1.5+2.25)/2 = 1.875

Получилось два промежутка. Из них надо выбрать тот на котором функция принимает значение разных знаков.

F(1.875) = 3.5156625 +1.875 – 6 = -0.609375 < 0

Следовательно, промежуток [1.875, 2.25] нам подходит т.к f(1.875)\*f(2.25) < 0.

a3=1.875

b3=2.25

x3 = (a3 + b3)/2 = (1.875+2.25)/2 = 2.0625

По итогу мы все ближе приближаемся к корню x=2.

1. Решим это же уравнение модифицированным методом Простых итераций .

Так же найдем второй корень (x=-3) с помощью метода простых итераций .

x^2+x-6=0 => x=(6-x)/x

x0 = -4

Проверим, что первая производная по модулю была меньше 1:

|φ’(x0)| = -6/x0^2 = 0.375 < 1

Следовательно метод простых итераций сходится к точному решению.

Проводим расчёты, для получения корня:

x1 = φ(x0)  = (6 - ( - 4))/(- 4) = -2.5

Продолжаем алгоритм:

x2 = φ(x1) = (6 - ( - 2.5))/(- 2.5) = -3.4

По итогу мы все ближе приближаемся к корню x=-3.

**Контрольные тесты**

Метод половинного деления:

1. y(x) = x^4-18x-10; x∈[-9, 1]; ℇ = 0.1; x =-0.565214
2. y(x) = x^4-18x-10; x∈[-9, 1]; ℇ = 0.01; x =-0.553751
3. y(x) = x^4-18x-10; x∈[-9, 1]; ℇ = 0.001; x =-0.550886
4. y(x) = x^4-18x-10; x∈[-9, 1]; ℇ = 0.00001; x = --0.550886

Метод простых итераций

1. y(x) = x^4-18x-10; x0=-1; ℇ = 0.1; x = -0.552083
2. y(x) = x^4-18x-10; x0=-1; ℇ = 0.01; x = -0.550394
3. y(x) = x^4-18x-10; x0=-1 ; ℇ = 0.001; x = -0.550457
4. y(x) = x^4-18x-10; x0=-1; ℇ = 0.00001; x =-0.550455

Видно что при уменьшении eps возрастает точность вычислений и оба метода приходят к одному решению

**Модульная структура программы**

/\*

# Структура для хранения отрезков [a,b]

@ Первое поле - левая граница

@ Второе поле - правая граница

\*/

typedef struct Line\_Segment

/\*

# Структура для хранения результата и кол-ва итераций

@ Первое поле - результат

@ Второе поле - кол-во итераций

\*/

typedef struct Result\_And\_iter

/\*

# Функция, которая делает подстановку точки в уравнение полинома

@ точка на отрезке

@ Возвращает: значение полинома при подстановке точки

\*/

double f1(double x)

/\*

# Функция, которая делает подстановку точки в трансцендентное уравнения

@ точка на отрезке

@ Возвращает: значение трансцендентного уравнения при подстановке точки

\*/

double f2(double x)

/\*

# Функция, которая делает подстановку точки в производную полинома

@ точка на отрезке

@ Возвращает: значение производной полинома при подстановке точки

\*/

double df1(double x)

/\*

# Функция, которая делает подстановку точки в производную трансцендентного уравнения

@ точка на отрезке

@ Возвращает: значение производной трансцендентного уравнения при подстановке точки

\*/

double df2(double x)

/\*

# Функция, которая находит корень на отрезке методом половинного деления у функции

@ отрезок [l,r]

@ функция f

@ точность eps

@ Возвращает: точку пересечения графика с осью ординат и кол-во итераций

\* Внутри используются функция: Equation\_Pol

\*/

,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

double bin\_poisk(double l, double r, double f(double), double eps)

/\*

# Функция, которая находит корень на отрезке методом простых итераций

@ начальное приближение

@ функция f

@ точность eps

@ Возвращает: точку пересечения графика с осью ординат и кол-во итераций

\* Внутри используются функция: Equation\_Trans

\*/

double fixed\_point(double x0, double fixed\_f(double), double eps)

/\* Главная функция, в которой указаны промежутки, в которых лежит корень

\* трансцедентного уравнения и полинома

\* Внутри используются все функции

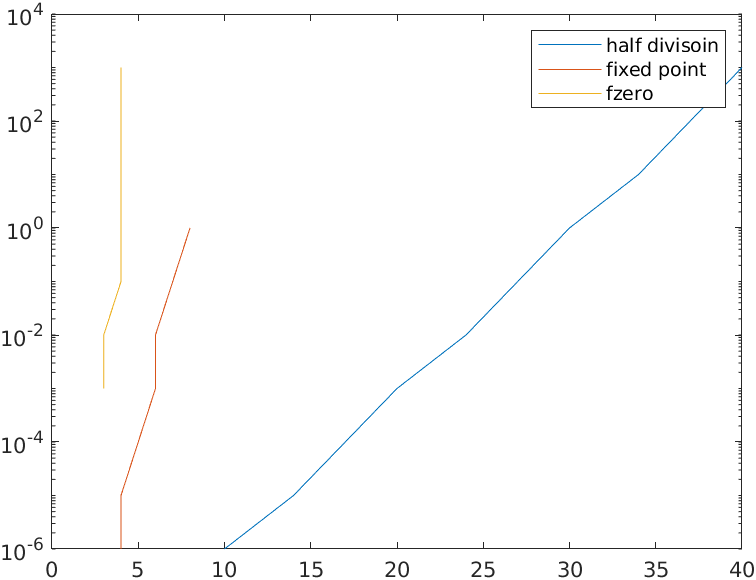
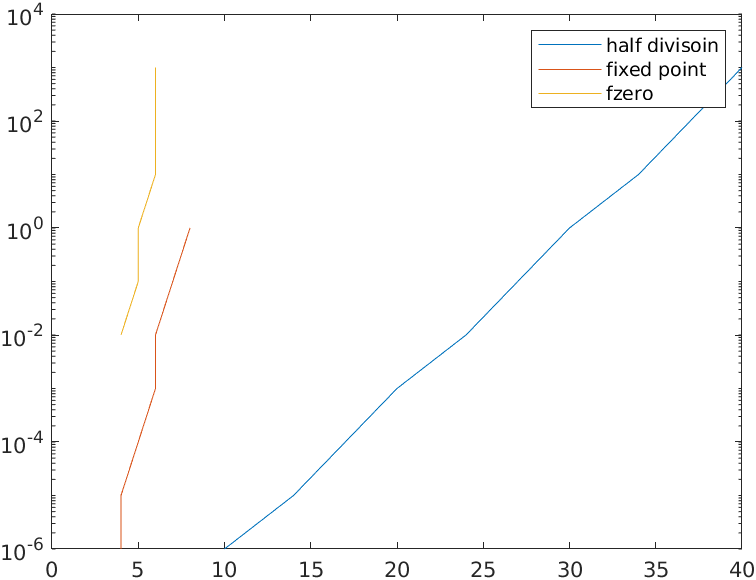
\*/

int main()

**Численный анализ методов**

Рассмотрим зависимости количества итераций от точности приближения.

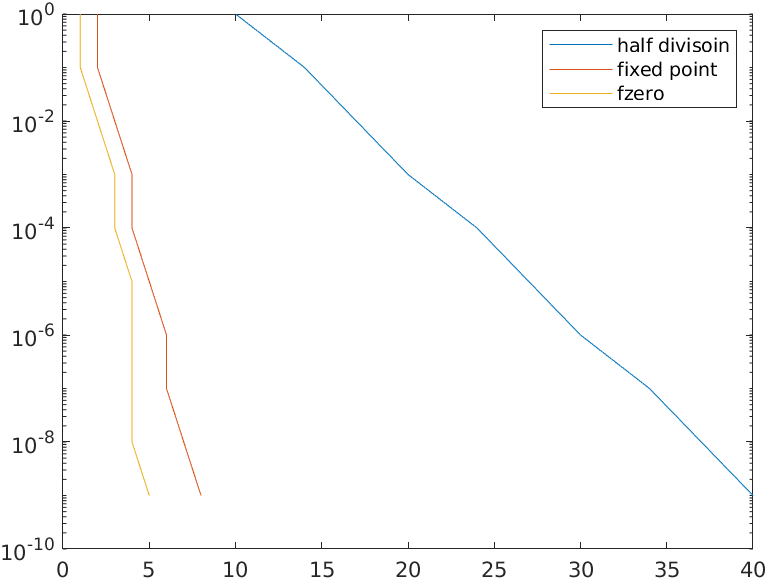
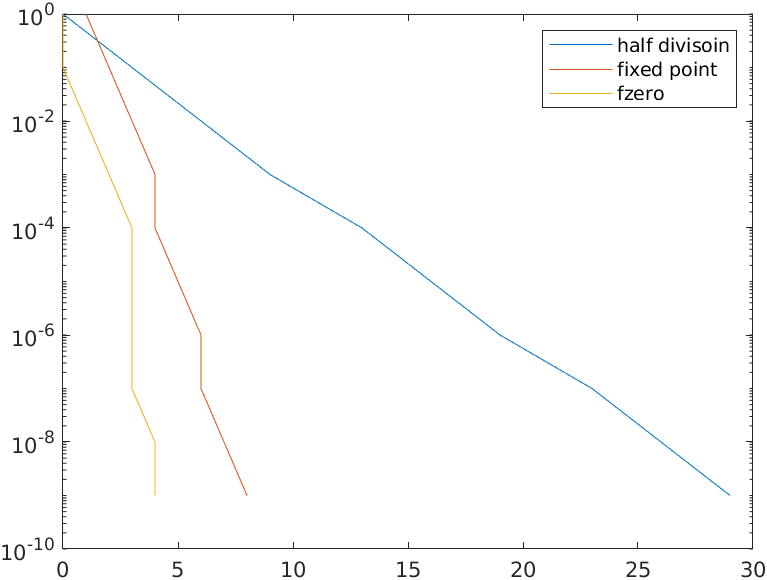
Первый график для x^4-18x-10 , второй 5^x-6x-7.



На графиках мы наблюдаем, что при работе с функцией, метод простых итераций выполняет меньшее количество итераций, чем метод половинного деления , и следовательно работает быстрее. Также метод простых итераций работает медленнее, чем функция fzero.

Рассмотрим зависимости количества итераций от точности решения(eps).

Первый график для x^4-18x-10 , второй 5^x-6x-7.



Зависимость итераций от точности

На графиках мы наблюдаем, туже закономерность» fzero всех быстрее при любых условиях затем идёт метод простых итераций далее метод ньютона

**Вывод**

Можно сделать вывод о том, что при работе с функцией нужно применять fzero. Так как происходят меньше итераций, а следовательно тратится меньше времени на нахождение корня. Однако fzero и метод простых итераций имеет свои ограничения, что не позволяет однозначно сказать об их превосходстве над методом половинного деления.