Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Московский институт электронной техники»

**Лабораторная работа**

**«Методы дихотомии, Ньютона, простых итераций»**

Работу выполнил

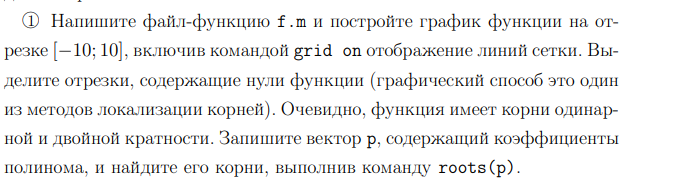
Учащийся группы ПИН-33

Карпеченков Михаил Владимирович

Под руководством

Васекина Бориса Васильевича

**Москва 2023**



F=@(x)x^3-3\*x^2-9\*x-5

hold on; grid on; xlabel('x'); ylabel('y');

fplot(F, [-10 10]);

format long

p=[1 -3 -9 -5];

roots(p)

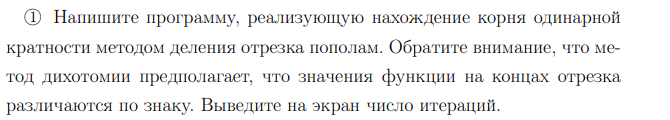
ans =

4.999999999999993 + 0.000000000000000i

-0.999999999999999 + 0.000000022357613i

-0.999999999999999 - 0.000000022357613i





function[x] = dihotomiya(f,a,b,e)

x=0;

n=floor(log2(abs((b-a))/e))+1;

c=0;

if(f(a)\*f(b)>0)

fprintf("В данном отрезке нет корня!")

else

if(f(a)\*f(b)==0)

if(f(a)==0)

x=a;

end

if(f(b)==0)

x=b;

end

end

if(f(a)\*f(b)<0)

for i=1:1:n

c=(b+a)/2;

if(f(a)\*f(c)<0)

b=c;

end

if(f(a)\*f(c)>0)

a=c;

end

if(f(a)\*f(c)==0)

x=c;

break;

end

if(abs((b-a))<=e)

x=a;

break;

end

end

fprintf("Количество итераций: %d\n", n);

end

end

end

x=dihotomiya(F,2.254789632457,6.000656584848,10^(-14))

F =

function\_handle with value:

@(x)x^3-3\*x^2-9\*x-5

Количество итераций: 49

x =

4.999999999999998





Метод Ньютона:

a1=3.5451515481644815554;

b1=8.51154896511841565;

x0=b1; mul=1; e=10^(-6)

x=NewtonForTasks(F,x0,a1,b1,mul,e)

a1=-1.1823219223882;

b1=-0.923280154896511841565;

x0=a1; mul=1; e=10^(-6)

x=NewtonForTasks(F,x0,a1,b1,mul,e)

Модифицированный метод Ньютона (для корня кратности 2)

a1=-1.1823219223882;

b1=-0.923280154896511841565;

x0=a1; mul=2; e=10^(-6)

x=NewtonForTasks(F,x0,a1,b1,mul,e)

Количество итераций: 6

x =

5.000000000000015

Количество итераций: 18

x =

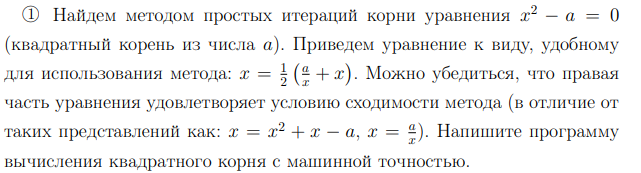
-1.000000716474364

Количество итераций: 3

x =

-0.999999999995509

Действительно, можно заметить, что модифицированный метод Ньютона дает верный результат в разы быстрее, чем обычный метод Ньютона.



Правая часть действительно удовлетворяет условию сходимости метода (достаточно взять 1-ую производную от правой части уравнения, чтобы понять, что модуль этой производной не превосходит 1)

function[Xnext] = SIM(fi,x0,e)

X=x0;

Xnext=X+2\*e;

n=0;

while(abs(Xnext-X)>=e)

X=Xnext;

Xnext=fi(Xnext)

n=n+1;

end

fprintf("Количество итераций: %d\n", n);

end

a=3;

fi=@(x)1/2\*(a/x+x)

x0=1;

e=10^(-10);

solution=SIM(fi,x0,e)

fi =

function\_handle with value:

@(x)1/2\*(a/x+x)

Количество итераций: 6

solution =

1.732050807568877



Из условия сходимости метода простых итераций (МПИ) находим область сходимости:

A a, в свою очередь, находится из того, что корень уравнения x^2=a (которое мы и решаем) должен находится в области сходимости (ввиду условия, которое я упоминал выше)

Это можно проверить в матлабе:

function[Xnext] = SIM(fi,x0,e)

X=x0;

Xnext=X+2\*e;

n=0;

while(abs(Xnext-X)>=e)

X=Xnext;

Xnext=fi(Xnext);

n=n+1;

end

fprintf("Количество итераций: %d\n", n);

end

Возьмем a чуть больше, чем 1:

fi=@(x)x^2+x-1.0001;

x1=solve(diff(fi,x)==-1)

x2=solve(diff(fi,x)==1)

x0=-0.5;

e=10^(-3);

s=SIM(fi,x0,e)

Алгоритм зацикливается (даже на плохой точности (10^-3)), вот последние значения X:

Xnext =

-1.010000000000671

Xnext =

-0.989999999999315

Xnext =

-1.010000000000671

Xnext =

-0.989999999999315

Xnext =

-1.010000000000671

Xnext =

-0.989999999999315

Xnext =

-1.010000000000671

Xnext =

-0.989999999999315

Xnext =

-1.010000000000671

Xnext =

-0.989999999999315

Xnext =

-1.010000000000671

Xnext =

-0.989999999999315

Xnext =

-1.010000000000671

Xnext =

-0.989999999999315

Xnext =

-1.010000000000671

Xnext =

-0.989999999999315

Xnext =

-1.010000000000671

Xnext =

-0.989999999999315

Xnext =

-1.010000000000671

Xnext =

-0.989999999999315

Xnext =

-1.010000000000671

Xnext =

-0.989999999999315

Теперь a=0.9999:

x1 =

-1

x2 =

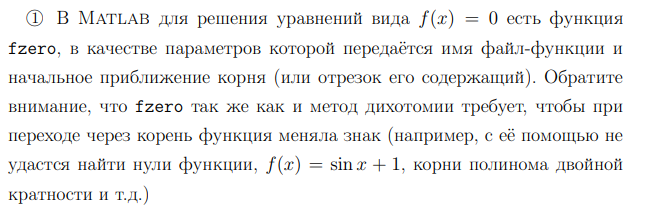
0

Количество итераций: 29954

s =

-0.999449900190642

В первом случае мы вышли за область сходимости, поэтому алгоритм зациклился, а во втором случае получили верный ответ.



Попробуем выполнить следующий код:

fi = @(x) sin(x) + 1;

x0 = fzero(fi,[-pi pi])

Error using fzero (line 290)

The function values at the interval endpoints must differ in sign.

Error in f (line 43)

x0 = fzero(fi,[-pi pi])

Получили вполне ожидаемую ошибку (необходима смена знака функции при переходе через корень)



График функции – синусоида (обычная), которая колеблется около прямой

Будет 3 корня (один около нуля, второй около -2 и третий симметрично второму – около 2)



Метод дихотомии:

fi=@(x)sin(x)-x/2;

hold on; grid on; xlabel('x'); ylabel('y');

fplot(fi, [-3 3]);

e=10^(-6);

x=dihotomiya(fi,-3,-1,e)

x=dihotomiya(fi,-1,1,e)

x=dihotomiya(fi,1,3,e)

Количество итераций: 21

x =

-1.895494461059570

Количество итераций: 21

x =

0

Количество итераций: 21

x =

1.895493507385254

Похоже на правду.

Метод Ньютона (модифицированный применять нет смысла, все корни кратности 1):

Количество итераций: 4

x =

-1.895494267033981

e =

1.000000000000000e-06

Количество итераций: 3

x =

9.529120656610879e-22

e =

1.000000000000000e-06

Количество итераций: 4

x =

1.895494267033981

Ответы сошлись с предыдущими ответами.

МПИ (метод простых итераций):

fi=@(x)sin(x)+x/2;

e=10^(-6);

x0=cos(pi)

s=SIM(fi,x0,e)

x0=0.001;

s=SIM(fi,x0,e)

x0=cos(pi/3)

s=SIM(fi,x0,e)

Количество итераций: 11

s =

-1.895494156499375

Количество итераций: 28

s =

1.895494087325744

x0 =

0.500000000000000

Количество итераций: 13

s =

1.895494172982301

Можно заметить, что мы вместо корня x=0 получили корень x=1.895494172982301

Это произошло, потому что 0 не находится в области сходимости функции (её область сходимости: cos(x)<0.5). Поэтому, чтобы получить 0 нужно подобрать какую-нибудь другую функцию для этого метода.

**Вывод:** В этой лабораторной работе я познакомился с методами решения с заданной точностью нелинейных уравнений и применил их на практике.