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

Лабораторная работа «БДЗ-1»

Работу выполнил
Учащийся группы ПИН-33
Карпеченков Михаил Владимирович
Под руководством
Ярошевича Владимира Александровича

Москва 2023

1. (а) Представить слагаемые и результат в виде *нормализованного* числа с плавающей точкой двойной точности: $(-1)^s \cdot 2^{e-1023} \cdot 1.f$, где $1.f$ записано в двоичном виде. (б) Если результат неточный (не умещается целиком в мантиссе), то указать относительную погрешность ошибки. Исходные данные в десятичной системе счисления.

$$-1593,5859375 \cdot 2^{128} + 1619,09765625 \cdot 2^{141}$$

№1 М. Карпенко
ПИН-3

Даны два числа:

$-1593,5859375 \cdot 2^{128} + 1619,09765625 \cdot 2^{141}$

Алгоритм:

1) Представим модуль числа в двоичный вид

2) Определим характеристику, которая рассчитывается по формуле

$\log_2(|x|)+1$, где x - исходное число

Если число отрицательное, то необходимо отнять единицу

3) Нормализовать мантиссы чисел, добавив в начало мантиссы единицу, если она не стоит на первом месте

4) Выровнять порядки для сложения, сдвигая мантиссу числа с меньшей характеристикой.

5) Сложить мантиссы двух чисел и нормализовать результат

6) Вычислить относительную погрешность ошибки, если это необходимо. Делается это по следующей формуле:

$\delta = \left|\frac{\Delta x}{x}\right|$

Решение

1) для числа $-1593,5859375 \cdot 2^{128}$

характеристика: $128+1023=1151$

степень двойки: $2^{1151-1023} = 2^{128}$

мантисса: 1,0011010000100111001011 1

в нормал. форме: $-1.01110101100010 01001 \cdot 2^{128}$

$\boxed{1}$

для числа $1619.09765625 \cdot 2^{141} =$

$1,1001010110001001 \cdot 2^{141}$

2) Произведем сложение

$-10111010110001001001 \times 2^{128} + 1,1001010110001001 \cdot 2^{141}$

$= 0,001111 \cdot 2^{141}$

В данном случае результат уменьшается в мантиссе, поэтому относительная погрешность ошибки равна нулю

2. Написать последовательность инструкций Matlab, формирующих указанную матрицу. Около каждой инструкции указать промежуточный результат в виде матрицы. Разрешается использовать матричные функции (`eye`, `repmat`, `flipud` и др.). Использовать циклы нельзя.

Входные данные:	*Нужно получить:*
Целое $n \geqslant 20$	$\left.\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & \cdots & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & \cdots & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & \cdots & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & \cdots & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & \cdots & 0 \end{pmatrix}\right\\}$ $2n$ строк

```
function A = alternating_matrix(n)
    % создание матрицы из единиц и нулей
    B = [1 0; 0 1];
    C = [0 1; 1 0];
    % повторение матрицы B и C и объединение их в одну матрицу D
    D = repmat({B}, n, n);
    D(1:2:end, 1:2:end) = {C};
    % конкатенация матриц в одну большую матрицу A
    A = cell2mat(D);
end
```

Файл Task2.m

```
matrix=alternating_matrix(20);
```

```
disp(matrix);

%Для проверки
[m, n] = size(matrix); % определение размерности матрицы
disp(m); % вывод количества строк матрицы
disp(n); % вывод количества столбцов матрицы
```

```
>> Task2
  Columns 1 through 20

     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0
     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1
     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0
     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1
     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0
     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1
     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0
     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1
     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0
     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1
     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0
     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1
     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0
     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1
     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0
     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1
     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0
     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1
     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0
     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1
     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0
     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1     1     0     0     1
     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0
     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1
```

3. (а) Локализовать корни уравнения (для каждого корня z_i указать отрезок $[a_i, b_i]$, содержащий только один этот корень z_i). Для *каждого* корня (б) построить итерационный процесс $x_{n+1} = \varphi(x_n)$, сходящийся к корню и (в) указать начальное значение x_0. Указание: локализацию проводить перебором интервалов $[a_i, b_i]$ или средствами математического анализа.

$$-x^3 - x^2 + 10x - 1 = 0$$

Перебором получил следующие интервалы:

$$x_1 \in [-5; -3]$$

$$x_2 \in [0; 1]$$

$$x_3 \in [2; 3]$$

Было предпринято большое количество попыток наайти корни методом простых итераций, но они провалились, поэтому использовал для вычислений метод Ньютона:

Файл Task3.m

```
clear; clc;
syms x;
format long
f=matlabFunction(-x^3 - x^2 + 10*x - 1);
```

```
p=[-1,-1,10,-1];
Roots=sort(roots(p))
disp('Вычисленя при помощи метода Ньютона')
x0=0.1
NewtonsMethod(f,x0,1,10^-9)
x0=-3.8
NewtonsMethod(f,x0,1,10^-9)
x0=2.5
NewtonsMethod(f,2.5,1,10^-9)
```

Файл NewtonsMethod.m

Реализация метода Ньютона (такой же файл есть и в ЛР):

```
function[Xnext] = NewtonsMethod(f,x0,p,e)
    format long;
    syms x;
    X=x0;
    n=0;
    Xnext=X+2*e;
    while(abs(Xnext-X)>=e)
        X=Xnext;
        der=matlabFunction(diff(f,x,1));
        Xnext=X-p*f(X)/der(X);
        n=n+1;
    end
    fprintf("Количество итераций: %d\n", n);
end
```

Вывод:

```
Roots =

  -3.743017747195621
   0.101126064468313
   2.641891682727315

Вычисленя при помощи метода Ньютона

x0 =

   0.100000000000000

Количество итераций: 3

ans =

   0.101126064468313


x0 =

  -3.800000000000000

Количество итераций: 4
```

```
ans =

  -3.743017747195625


x0 =

   2.500000000000000

Количество итераций: 5

ans =

   2.641891682727311
```

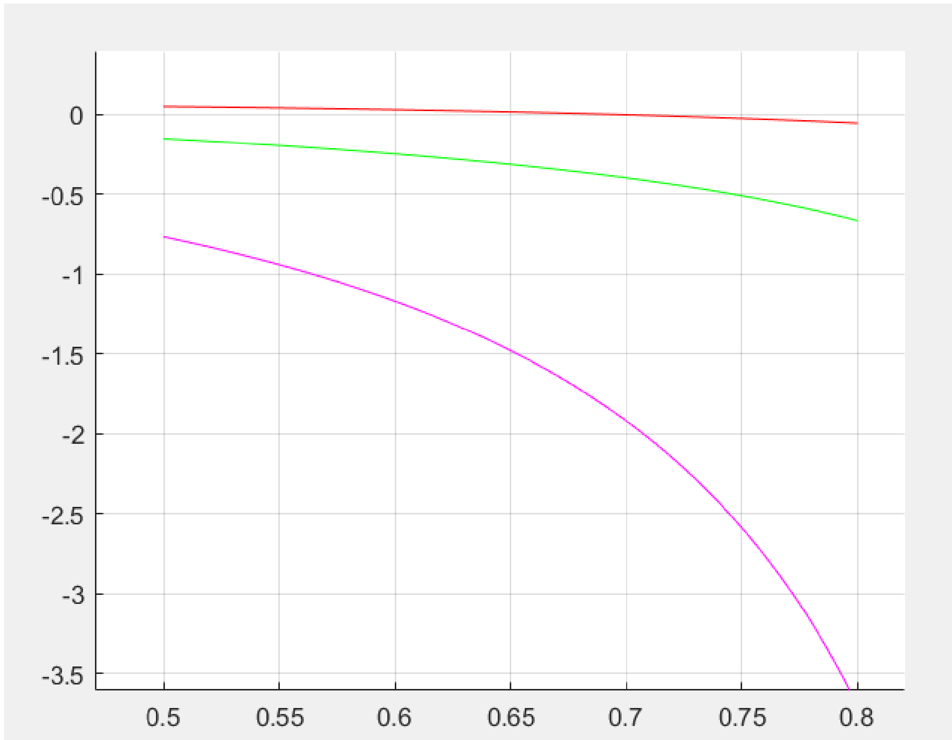
4. Известно, что интервалу $[a, b]$ принадлежит *только* корень x_* уравнения (другие корни интервалу не принадлежат). (а) Построить итерационный процесс Ньютона $x_{n+1} = x_n - f(x_n)/f'(x_n)$ и (б) обосновать какую из границ интервала $[a, b]$ можно принять за x_0. Указание: в пункте (б) выяснить знаки производных $f'(x)$ и $f''(x)$ и использовать соответствующую теорему.

$$\arccos x + x - 3/2 = 0, \quad x_* \in [0,5; 0,8]$$

Файл Task4.m

```
clear; clc;
syms x;
f=matlabFunction(acos(x) + x - 3/2)
hold on; grid on;
fplot(f,[0.5 0.8], 'Color','r')
fplot(diff(f,x,1),[0.5 0.8],'Color','g')
fplot(diff(f,x,2),[0.5 0.8],'Color','m')
x0=0.8;
NewtonsMethod(f,x0,1,10^-9)
```

Красная кривая – график самой функции; Зеленая кривая – 1-ая производная; Фиолетовая — 2ая



Благодаря графику мы определили знаки производных и функций на отрезке.

Проверил выполнение теоремы 2.3 (достаточное условие сходимости последовательности к корню) (получил, что и 1-ая и 2-ая производная меньше нуля на рассматриваемом отрезке => беру за x0 правый конец отрезка)

Вывод:

```
f =

  function_handle with value:

    @(x)x+acos(x)-3.0./2.0

Количество итераций: 5

ans =

   0.688176681004484
```

5. (а) Построить интерполяционный многочлен Лагранжа для функции $f(x)$ по узлам x_i. (б) Оценить сверху погрешность $|R_n(x)|$ приближения функции многочленом.

$$\cos\sqrt{x}+1 \qquad x_0=1,\ x_1=3,\ x_2=5$$

Файл Task5.m

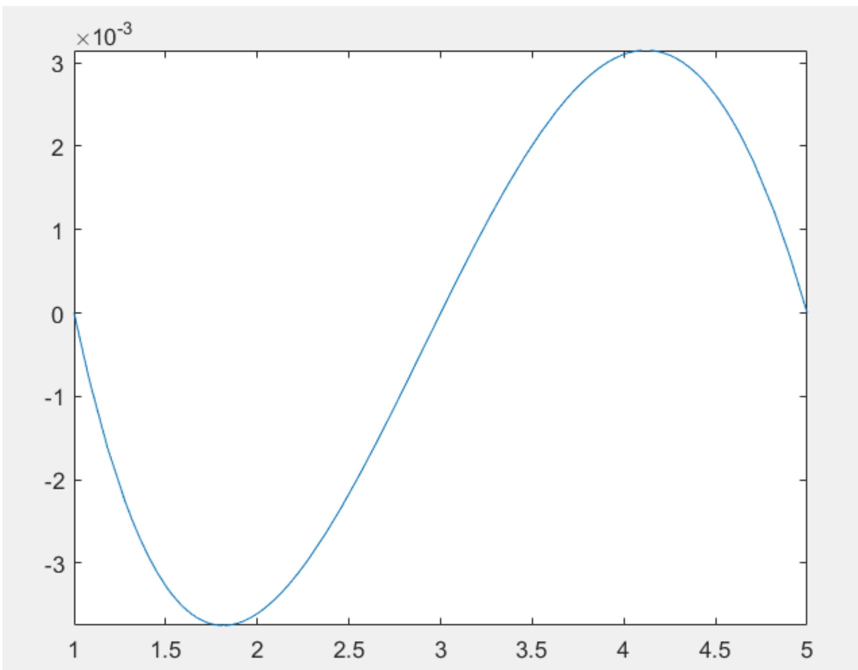
```
clear; clc; clf;
syms x
X=[1, 3, 5]
y=matlabFunction(1+cos(sqrt(x)));
P=LagPoly(X,y(X));
P(x)
Error(X,y,P);
```

Файл LagPoly.m

```
function[P] = LagPoly(t,F)
syms x;
temp = repmat (t',1,length(t));
power = repmat (0:(length(t)-1),length(t),1);
A = temp.^power;
B=F';
X=inv(A)*B;
P=@(x)(sum(vpa(X'.*x.^power(1,:),10)));
end
```

Файл Error.m

```
function[res] = Error(X,y,P)
    syms x;
    dfminus=matlabFunction(-diff(y,x,length(X)));
    df=matlabFunction(diff(y,x,length(X)));
    w=matlabFunction(prod(x-X));
    maxwX=fminbnd(matlabFunction((-1)*prod(x-X)),X(1)-1,X(length(X))+1);
    maxw=abs(w(maxwX));
    maxdf=abs(df(fminbnd(dfminus,X(1),X(length(X)))));
    MaxTheory=maxdf/factorial(length(X))*maxw
    Prminus=matlabFunction(-abs(P(x)-y(x)));
    Pr=matlabFunction(abs(P(x)-y(x)));
    MaxPractice=Pr(fminbnd(Prminus,X(1),X(length(X))))
    R=feval(df,x)/factorial(length(X))*w(x);
    fplot(R,[1 5]);
    res=[MaxTheory, MaxPractice];
end
```



Вывод:

```
X =

     1     3     5
ans =

0.030517813320044273606157503309078*x^2 - 0.47250067550159235985063332918799*x +
1.9822851680496877957438073281082

MaxTheory =
   0.002946114245952

MaxPractice =
   0.003511942051904
```