Московский авиационный институт
(Национальный исследовательский университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Лабораторная работа № 2

по курсу «Численные методы»
Тема: численные методы решения СЛАУ.

Студент: Сыроежкин К.Г.
Группа: 80-304б
Преподаватель: Гидаспов В.Ю.
Оценка:

Москва, 2021

Задание 1

1) Постановка задачи:

Реализовать методы простой итерации и Ньютона решения нелинейных уравнений в виде программ, задавая в качестве входных данных точность вычислений. С использованием разработанного программного обеспечения найти положительный корень нелинейного уравнения (начальное приближение определить графически). Проанализировать зависимость погрешности вычислений от количества итераций.

Вариант 16:

$$xe^x + x^2 - 1 = 0.$$

2) Теория:

Метод простой итерации

1) Исходное уравнение $f(x) = 0$ заменяется эквивалентным уравнением с выделенным линейным членом $x = \varphi(x)$

2) Выбирается начальное приближение $x^{(0)}$ и начиная с него строится последовательность $x^{(k+1)} = \varphi(x^k)$. Если $\varphi(x)$ – непрерывная ф-ция, а x^k – сходится, то решением уравнения будет $x^{(k)}$

Условия сходимости метода и оценка его погрешности определяются теоремой [2]:

Теорема 2.3. Пусть функция $\varphi(x)$ определена и дифференцируема на отрезке [a,b]. Тогда если выполняются условия:

1) $\varphi(x) \in [a,b] \quad \forall x \in [a,b]$,

2) $\exists q : |\varphi'(x)| \le q < 1 \quad \forall x \in (a,b)$,

то уравнение (2.5) имеет и притом единственный на [a,b] корень $x^{(*)}$;

3) Условие окончания итерационного процесса можно записать так:

$$\frac{q}{1-q}\left|x^{(k+1)} - x^{(k)}\right| \le \varepsilon$$

Метод Ньютона

1) Выбирается отрезок [a,b] такой, что $f(a)f(b) < 0$

2) Выбирается начальное приближение $x^{(0)}$ на [a,b] так, чтобы $f(x^{(0)})\,f''(x^{(0)}) > 0$

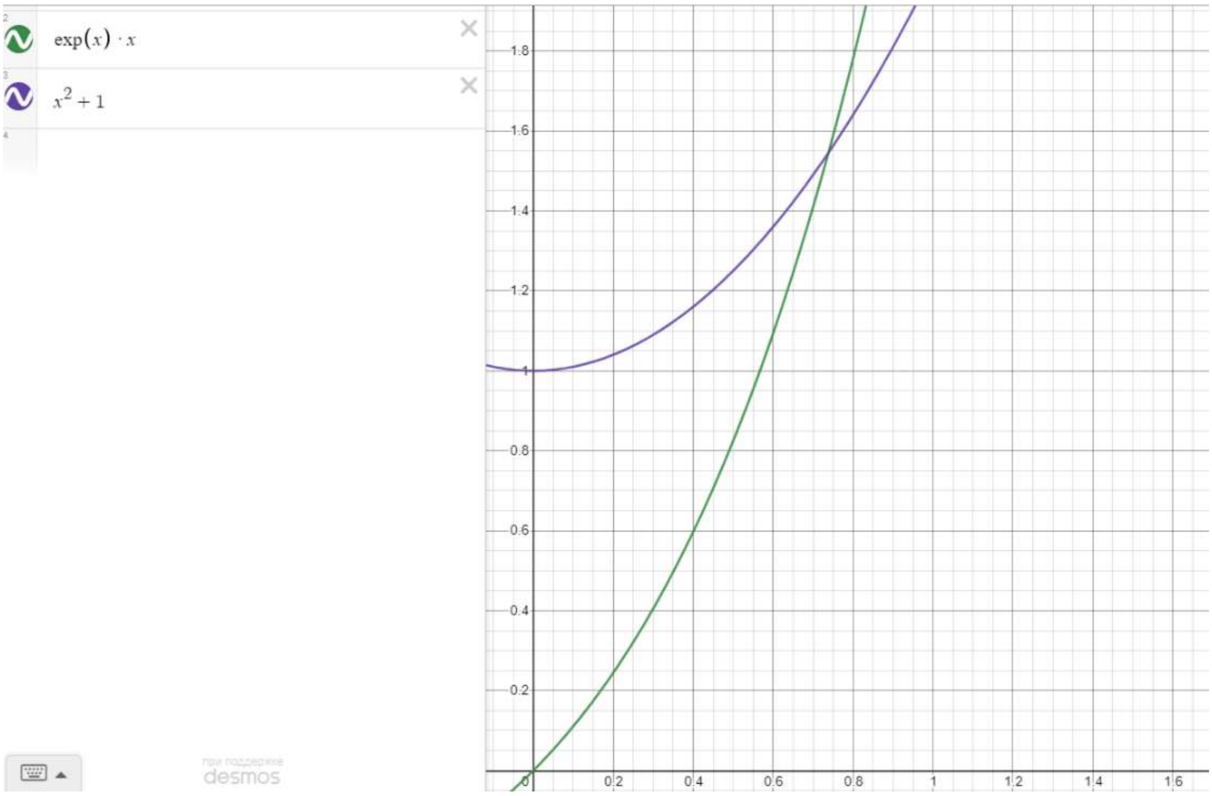
3) Строится последовательность $x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$, которая, согласно теореме, будет сходится корню.

4) Условие окончания итерационного процесса можно записать так:

$$\left|x^{(k+1)} - x^{(k)}\right| < \varepsilon$$

3) Полученный ответ:

Приближение:




```
Метод простых итераций:
[[ 0.          0.7         0.75545106]
 [ 1.          0.75545106  0.73196567]
 [ 2.          0.73196567  0.74105599]
 [ 3.          0.74105599  0.73739452]
 [ 4.          0.73739452  0.73884719]
 [ 5.          0.73884719  0.7382673 ]
 [ 6.          0.7382673   0.73849822]
 [ 7.          0.73849822  0.73840618]
 [ 8.          0.73840618  0.73844285]
 [ 9.          0.73844285  0.73842824]
 [10.          0.73842824  0.73843406]]
Метод Ньютона:
[[0.00000000e+00 8.00000000e-01 5.83683622e-02]
 [1.00000000e+00 7.41631638e-01 3.19041210e-03]
 [2.00000000e+00 7.38441226e-01 8.82488493e-06]]
```

4) Код программы:

```
import numpy as np

def func(x):
    return np.exp(x)*x-x*x-1
```

```
def dfunc(x):
    return np.exp(x)*(x+1)-2*x

def express_func(x):
    return np.log((1+x*x)/x)

def simple_iter(func,a,b,eps,flag):
    q = 0.99
    x = (a+b)/2
    count = 0
    ans = []
    while(True):
        ans.append([count, x, func(x)])
        x0 = x
        x = func(x)
        count+=1
        if (q/(1-q)*np.abs(x-x0)<eps) or (count>100):
            break
    if flag == True:
        return np.array(ans)
    else:
        return np.array(ans[-1])


def newton(func,dfunc,x0,eps,flag):
    x = x0
    count = 0
    ans = []
    while True:
        ans.append([count, x, func(x)/dfunc(x)])
        x0 = x
        x -= func(x)/dfunc(x)
        count+=1
        if (np.abs(x-x0)<eps) or (count>100):
            break
    if flag == True:
        return np.array(ans)
    else:
        return np.array(ans[-1])

print("Метод простых итераций:")
print(simple_iter(express_func,0.6,0.8,0.001, True))
print("Метод Ньютона:")
print(newton(func,dfunc,0.8,0.001,True))
```

5) Выводы:

Метод Ньютона работает быстрее, чем метод итераций, но требует серьезного начального предположения.

Задание 2

1) Постановка задачи:

Вариант 16:

$$\begin{cases} ax_1 - \cos x_2 = 0, \\ ax_2 - e^{x_1} = 0. \end{cases}$$

a = 2

2) Теория:

Метод простой итерации

1) Исходная система заменяется на эквивалентную систему вида

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, ..., x_n) \\ x_2 = \varphi_2(x_1, x_2, ..., x_n) \\ \dots\dots\dots\dots\dots\dots\dots \\ x_n = \varphi_n(x_1, x_2, ..., x_n) \end{cases}$$

2) Выбираем начальное приближение $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)})^{\mathrm{T}}$
и строим последующие приближения по формулам:

$$\begin{cases} x_1^{(k+1)} = \varphi_1(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}) \\ x_2^{(k+1)} = \varphi_2(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}) \\ \dots\dots\dots\dots\dots\dots\dots \\ x_n^{(k+1)} = \varphi_n(x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)}) \end{cases}$$

3) Условие сходимости итерационного процесса: $\max\limits_{\mathbf{x}\in G} \|\boldsymbol{\varphi}'(\mathbf{x})\| \le q < 1,$

4) Условие окончания итерационного процесса:

$$\frac{q}{1-q}\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| \le \varepsilon,$$

$$\left\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right\| = \max_i \left|x_i^{(k+1)} - x_i^{(k)}\right|.$$

Метод Ньютона

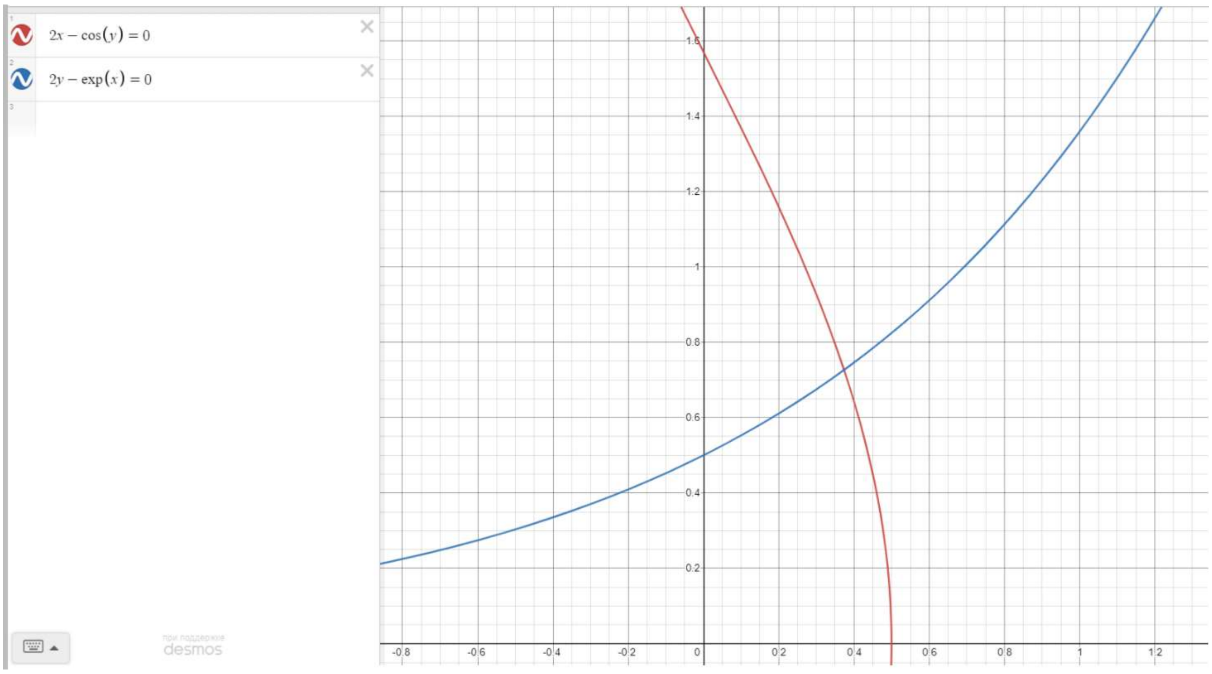
Итерационная формула метода: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{J}^{-1}(\mathbf{x}^{(k)})\mathbf{f}(\mathbf{x}^{(k)})$

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ & \cdots\cdots\cdots\cdots\cdots\cdots & \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

Условие окончания итерационного процесса можно записать так:

$$\left| x^{(k+1)} - x^{(k)} \right| < \varepsilon$$

3) Полученный ответ:




```
Метод итераций:
[[0 array([0.3, 0.8])]
 [1 array([0.34835335, 0.6749294 ])]
 [2 array([0.39037553, 0.70836639])]
 [3 array([0.37971285, 0.73876778])]
 [4 array([0.36964944, 0.73093238])]
 [5 array([0.37227615, 0.72361359])]
 [6 array([0.37470904, 0.72551682])]
 [7 array([0.3740783 , 0.72728407])]
```

```
 [8 array([0.37349141, 0.72682548])]
 [9 array([0.37364382, 0.72639904])]
 [10 array([0.37378547, 0.72650976])]
 [11 array([0.3737487 , 0.72661268])]
 [12 array([0.37371451, 0.72658596])]
 [13 array([0.37372339, 0.72656112])]]
Метод Ньютона:
[[0 array([0.3, 0.8])]
 [1 array([0.3750461 , 0.72558022])]
 [2 array([0.37372862, 0.72657074])]]
```

4) Код программы:

```
import numpy as np

def system(x):
    return [2*x[0]-np.cos(x[1]), 2*x[1]-np.exp(x[0])]

def express_system(x):
    return [np.cos(x[1])/2, np.exp(x[0])/2]

def dsys_dx1(x):
    return [2,-np.exp(x[0])]

def dsys_dx2(x):
    return [np.sin(x[1]), 2]

def simple_iter(system,x0,eps, flag):
    q=0.99
    count = 0
    x = x0.copy()
    ans = []
    while True:
        ans.append([count, np.array(x)])
        x0 = x.copy()
        x = system(x0)
        count+=1
        if q/(1-q)*max(abs(x0[0]-x[0]),(abs(x0[1]-x[1])))<eps or count>100:
            break
    if flag:
        return np.array(ans)
    else:
        return np.array(ans[-1])

def newton(sys, dsysdx1,dsysdx2,x0,eps, flag):
    ans = []
    count = 0
    x = x0.copy()
```

```
    while True:
        x0 = x.copy()
        ans.append([count,np.array(x)])
        f = sys(x0)
        fdx1 = dsysdx1(x0)
        fdx2 = dsysdx2(x0)
        x[0] -= (f[0]*fdx2[1]-f[1]*fdx2[0])/(fdx1[0]*fdx2[1]-fdx1[1]*fdx2[0])
        x[1] -= (f[1]*fdx1[0]-fdx1[1]*f[0])/(fdx1[0]*fdx2[1]-fdx1[1]*fdx2[0])
        count+=1
        if max(abs(x0[0]-x[0]),(abs(x0[1]-x[1])))<eps or count > 100:
            break
    if flag:
        return np.array(ans)
    else:
        return np.array(ans[-1])

print("Метод итераций:")
print(simple_iter(express_system, [0.3, 0.8],0.001, True))
print("Метод Ньютона:")
print(newton(system,dsys_dx1,dsys_dx2,[0.3, 0.8],0.001, True))
```

5) Выводы:

Метод Ньютона работает быстрее, чем метод итераций, но требует серьезного начального предположения.