Министерство образования и науки Российской Федерации Федеральное агентство по образованию

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет по лабораторной работе №2 дисциплины

«Вычислительная математика»

Выполнил студент группы ИВТ-22 /Крючков И. С/ Проверил /Исупов К. С./

Киров 2021

# Задание:

Вариант 10

1. Решить систему линейных уравнений 4-го порядка методом Гаусса с точностью е=0,001.

Уравнения системы:

2,20\*x1-3,17\*x2+1,24\*x3-0,87\*x4=0,46

1,50\*x1+2,11\*x2-0,45\*x3+1,44\*x4=1,50

0,86\*x1-1,44\*x2+0,62\*x3+0,28\*x4=-0,12

0,48\*x1+1,25\*x2-0,63\*x3-0,97\*x4=0,35

2. Решить систему линейных уравнений 4-го порядка с точностью е=0,0001:

- методом простой итерации.

Уравнения системы:

x1=0,22\*x2-0,11\*x3+0,31\*x4+2,70

x2=0,38\*x1-0,12\*x3+0,22\*x4-1,50

x3=0,11\*x1+0,23\*x2-0,51\*x4+1,20

x4=0,17\*x1-0,21\*x2+0,31\*x3-0,17

3. Решить систему линейных уравнений 3-го порядка методом обратной матрицы с точностью е=0,001.

Уравнения системы:

2\*x1+x2+x3=5

x1+3\*x2+2\*x3=8

4\*x1+x2+2\*x3=9

4. Решить систему нелинейных уравнений 2-го порядка методом Ньютона с точностью е=0,001.

Уравнения системы:

x+exp(0,1-y)+2,1=0

y+cos(x)=0

5. Проверить результаты с помощью системы Mathcad.

**Теоретические сведения:**

**Метод Гаусса**

Состоит из двух этапов:

1) Прямой ход;

2) Обратный ход.

На этапе прямого хода систему уравнений записывают в виде матрицы коэффициентов перед неизвестными, которую затем с помощью элементарных преобразований приводят в треугольному (ступенчатому) виду (либо устанавливают, что система несовместна). А именно, среди элементов первого столбца матрицы выбирают ненулевой, перемещают его на крайнее верхнее положение перестановкой строк и вычитают получившуюся после перестановки первую строку из остальных строк, помножив её на величину, равную отношению первого элемента каждой из этих строк к первому элементу первой строки, обнуляя тем самым столбец под ним. После того, как указанные преобразования были совершены, первую строку и первый столбец мысленно вычёркивают и продолжают пока не останется матрица нулевого размера. Если на какой-то из итераций среди элементов первого столбца не нашёлся ненулевой, то переходят к следующему столбцу и проделывают аналогичную операцию. На этапе обратного хода осуществляется выражение всех получившихся базисных переменных через небазисные и построение фундаментальной системы решений, либо, если все переменные являются базисными, то выражение в численном виде единственного решения системы линейных уравнений. Эта процедура начинается с последнего уравнения, из которого выражают соответствующую базисную переменную (а она там всего одна) и подставляют в предыдущие уравнения, и так далее, поднимаясь по «ступенькам» наверх.

**Метод простых итераций**

Для того, чтобы построить итеративную процедуру метода Якоби, необходимо провести предварительное преобразование системы уравнений 𝐴𝑥 = 𝑏 к итерационному виду 𝑥 = 𝐵𝑥 + 𝑔. Оно может быть осуществлено по одному из следующих правил:

где в принятых обозначениях D означает матрицу, у которой на главной диагонали стоят соответствующие элементы матрицы A, а все остальные нули; тогда как матрицы U и L содержат верхнюю и нижнюю треугольные части A, на главной диагонали которых нули, E—единичная матрица.

Формула для расчёта приближения выглядит следующим образом:

Условия остановки процесса итерации:

**Метод обратной матрицы**

Суть метода состоит в умножении левой и правой части матричной записи уравнения на матрицу, обратную матрице коэффициентов перед неизвестными (основной матрице). При этом в левой части получается вектор, содержащий неизвестные, а в правой части значения этого вектора.

Само собой разумеется, что на этот метод накладывается ограничение |A| ≠ 0, в противном случае найти обратную матрицу будет невозможно. Обратная матрица получается, как транспонированная матрица алгебраических дополнений основной матрицы, умноженная на 1/|A|

**Метод Ньютона**

Пусть задана система двух нелинейных уравнений

и известно приближенное решение (x0, y0).

Точное решение можно выразить следующим образом:

где ∆x и ∆y — невязки приближенного и точного решений.

Введем обозначения:

Получим систему линейных уравнений относительно невязок:

Решение этой линейной системы позволяет найти невязки и вычислить приближенные значения корней, которые принимаются за новое начальное приближение.

Условия окончания итераций:

**Практическая часть**

**Метод Гаусса**

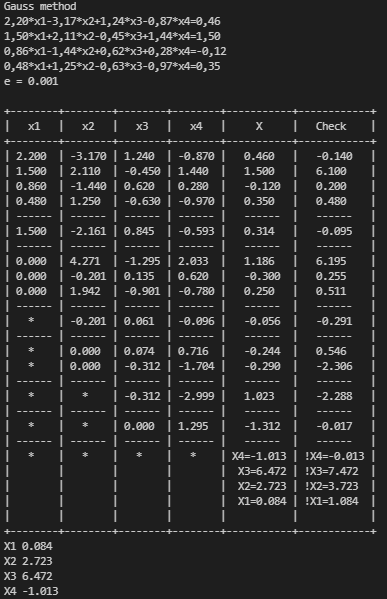


Рис. 1 – Экранная форма метода Гаусса

Проверка в системе MathCad

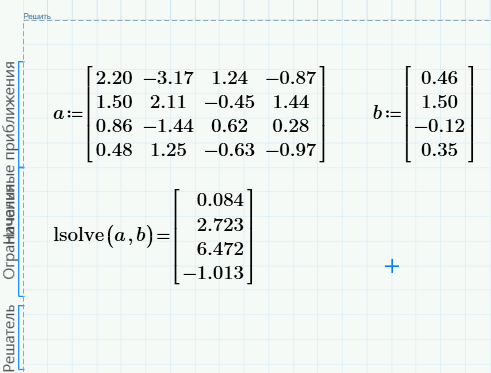


Рис. 2 – Проверка метода Гаусса в системе MathCad

**Метод простых итераций**

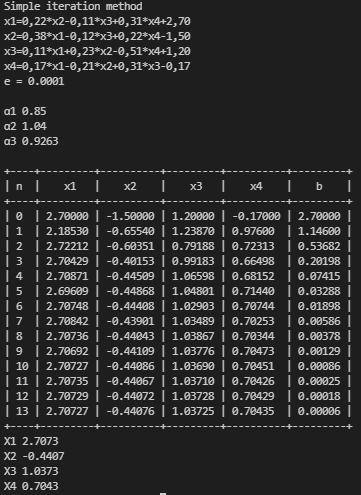


Рис. 3 – Экранная форма метода простых итераций

Проверка в системе MathCad

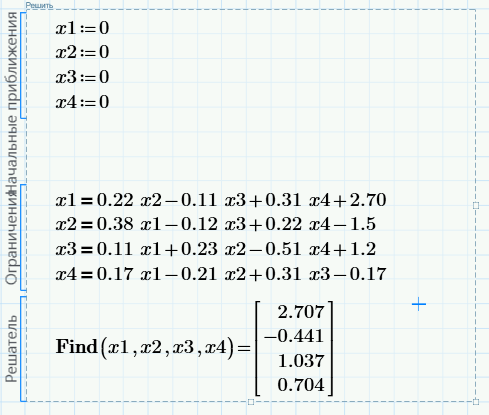


Рис. 4 – Проверка метода простых итераций в системе MathCad

**Метод обратной матрицы**

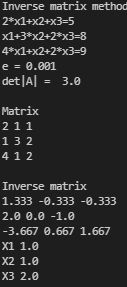


Рис. 5 – Экранная форма метода обратной матрицы

Проверка в системе MathCad

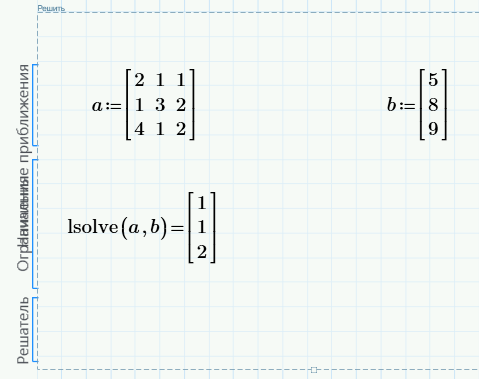


Рис. 6 – Проверка метода обратной матрицы в системе MathCad

**Метод Ньютона**

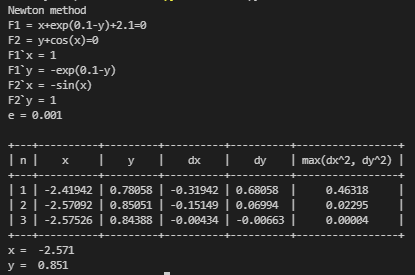


Рис. 7 – Экранная форма метода обратной Ньютона

Проверка метода Ньютона в системе MathCad

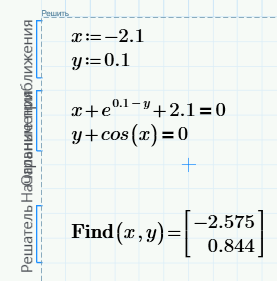


Рис. 8 – Проверка метода Ньютона в системе MathCad

**Листинг кода:**

def gauss(a, b, eps):

n = len(b)

for i in range(n): # по столбцам

# поиск опорного элемента

z = i

for h in range(z+1, n):

if(abs(a[z][i]) < abs(a[h][i]) or a[z][i] == 0):

a[z], a[h] = a[h], a[z]

b[z], b[h] = b[h], b[z]

# прямой ход

for j in range(i+1, n): # по строкам c i+1

m = -a[j][i]/a[i][i]

for k in range(i, n): # по столбцам с i

a[j][k] += a[i][k]\*m

b[j] += b[i]\*m

# обратный ход

x = [0] \* n

for i in range(n-1, -1, -1):

r = 0

for j in range(i+1, n):

r += a[i][j] \* x[j]

x[i] = (b[i] - r)/a[i][i]

return [round(i, int(-log10(eps))) for i in x]

def simple\_iterations(a, b, eps):

x0 = b[:]

x1 = b[:]

n = len(b)

tablex = PrettyTable()

tablex.field\_names = ["n", "x1", "x2", "x3", "x4", 'b']

tablex.float\_format = '.5'

tablex.add\_row([0, x1[0], x1[1], x1[2], x1[3], abs(max(x1, key=abs))])

k = 0

while True:

row = [0]\*6

k += 1

row[0] = k

for i in range(n):

x1[i] = b[i]

for j in range(n):

x1[i] += x0[j] \* a[i][j]

row[i+1] = x1[i]

atmp = [0]\*n

for i in range(n):

atmp[i] = x1[i] - x0[i]

row[-1] = abs(max(atmp, key=abs))

tablex.add\_row(row)

if abs(max(atmp, key=abs)) <= eps:

break

else:

x0 = x1[:]

print(tablex)

return [round(x, int(-log10(eps))) for x in x0]

def reverse\_matrix(a,b):

a0 = LA.inv(a)

ar = np.array(a0)

br = np.array(b)

return ar.dot(br)

def newton\_method(funcs, s, eps):

f1, f2, f1x, f1y, f2x, f2y = funcs

x0, y0 = s

tablex = PrettyTable()

tablex.field\_names = ["n", "x", "y", "dx", "dy", 'max(dx^2, dy^2)']

tablex.float\_format = '.5'

k = 0

while True:

k += 1

row = [0]\*6

row[0] = k

a = [

[f1x(x0, y0), f1y(x0, y0)],

[f2x(x0, y0), f2y(x0, y0)]

]

b = [-f1(x0, y0), -f2(x0, y0)]

# определитель a

d = (a[0][0] \* a[1][1]) - (a[0][1] \* a[1][0])

dx = (b[0] \* a[1][1] - b[1] \* a[0][1]) / d

dy = (b[1] \* a[0][0] - b[0] \* a[1][0]) / d

row[1] = x0+dx

row[2] = y0+dy

row[3] = dx

row[4] = dy

row[5] = max(dx\*\*2, dy\*\*2)

tablex.add\_row(row)

if max(dx\*\*2, dy\*\*2) <= eps:

break

else:

x0 += dx

y0 += dy

print(tablex)

return round(x0, int(-log10(eps))), round(y0, int(-log10(eps)))

**Вывод:** в ходе выполнения лабораторной работы был закреплен лекционный материал по теме «Численные методы решения систем уравнений». Закреплены на практике численные методы решение нелинейных уравнений: метод Гаусса, метод простых итерация, метод обратной матрицы и метод Ньютона, а так же были изучены особенности их применения. Для всех приведённых в задании методов была написана программа, реализующая их.