МІНІСТЕРСВО ОСВІТИ І НАУКИ УКРАІНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

ЛАБОРАТОРНА РОБОТА №4

з навчальної дисципліни

«*Чисельні методи*»

на тему:

*«*Чисельне інтегрування та пошук власних значень*»*

*варіант 8*

Виконав:

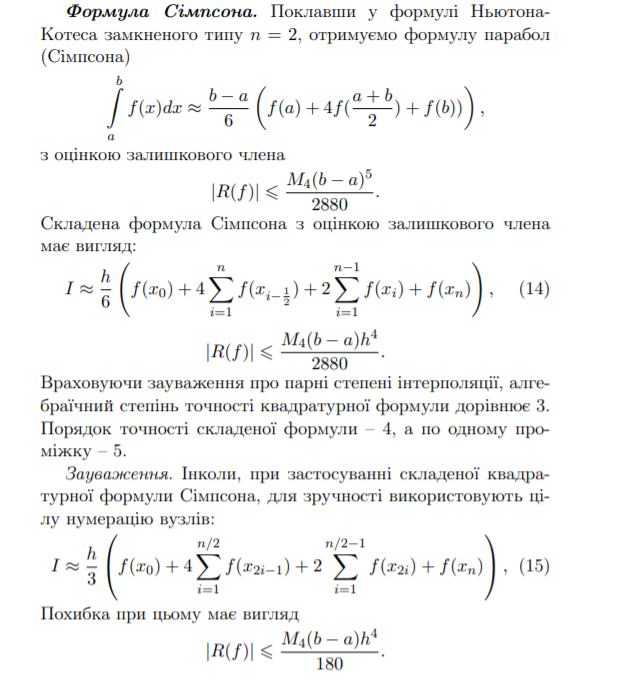
студент ІІ курсу, групи К-27

спеціальності «Комп’ютерні науки. Інформатика»

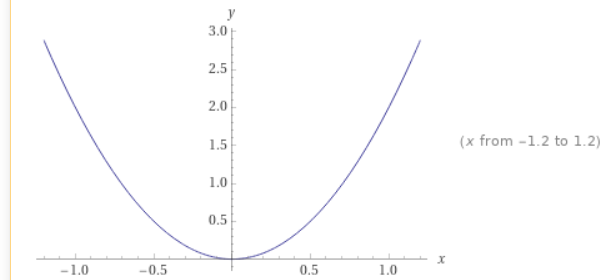
*Некряч Владислав Вадимович*

*Київ, 2021 рік*

1. **Метод Сімпсона**

****

**Постанова задачі**



Так як функція не має 4 похідної, то аналітично ми не зможемо вирахувати довжину кроку. Щоб вийти з цієї ситуації, чисельно вирахуємо довжину кроку з допомогою правила Рунге. Отримали довжину кроку h = 0.5 (див. мал. нижче).  
  
**Код:**

export const f = (x: number) => { return 2 \* x \* x; }

const eps = Math.pow(10, -3)

export const run = (a: number, b: number) => {

  let nByTwo = 2

  let prevCalc = calculateForN(nByTwo, 3, 5)

  let currCalc = calculateForN(nByTwo \*= 2, 3, 5)

  // Runge's rule

  while ((1/15) \* Math.abs(currCalc - prevCalc) > eps) {

    prevCalc = currCalc

    currCalc = calculateForN(nByTwo \*= 2, 3, 5)

  }

  return currCalc

}

const calculateForN = (nByTwo: number, a: number, b: number) => {

  let result = 0

  const h = (b - a) / nByTwo;

  const vertices = []

  let currentStep = a

  while (currentStep <= b) {

    vertices.push(currentStep)

    currentStep += h

  }

  result = f(vertices[0]) + f(vertices[vertices.length - 1])

  for (let i = 1; i < vertices.length - 1; i++) {

    if (i % 2 === 0)

    {

      result += f(vertices[i]) \* 2

    }

    else {

      result += f(vertices[i]) \* 4

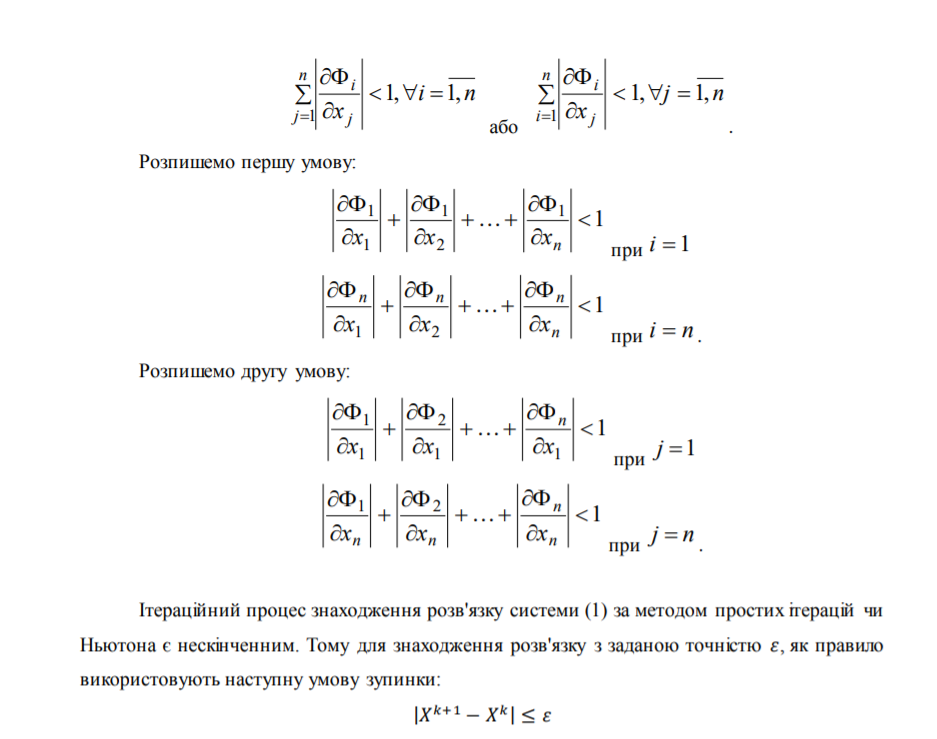
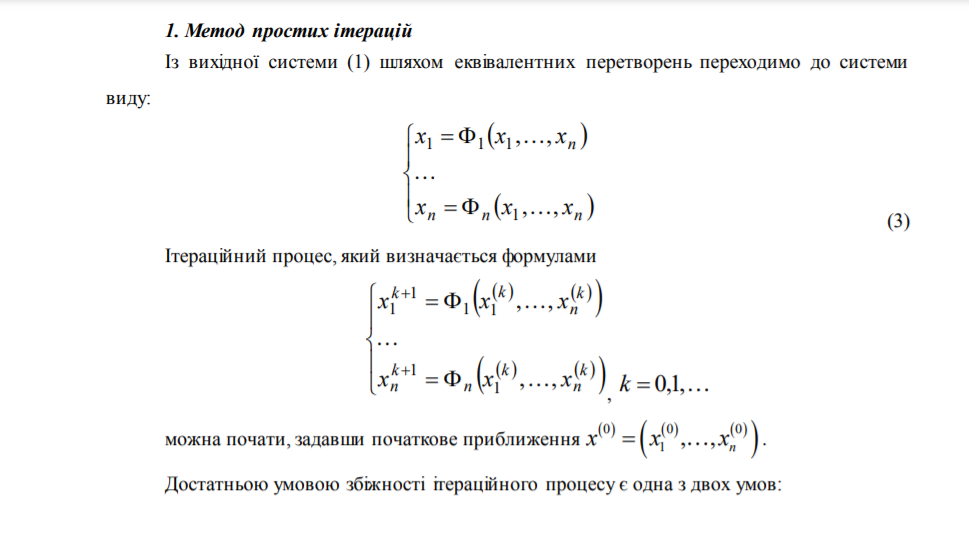
    }

  }

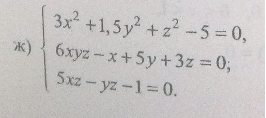
  console.log('step length', h)

  return result \* h / 3

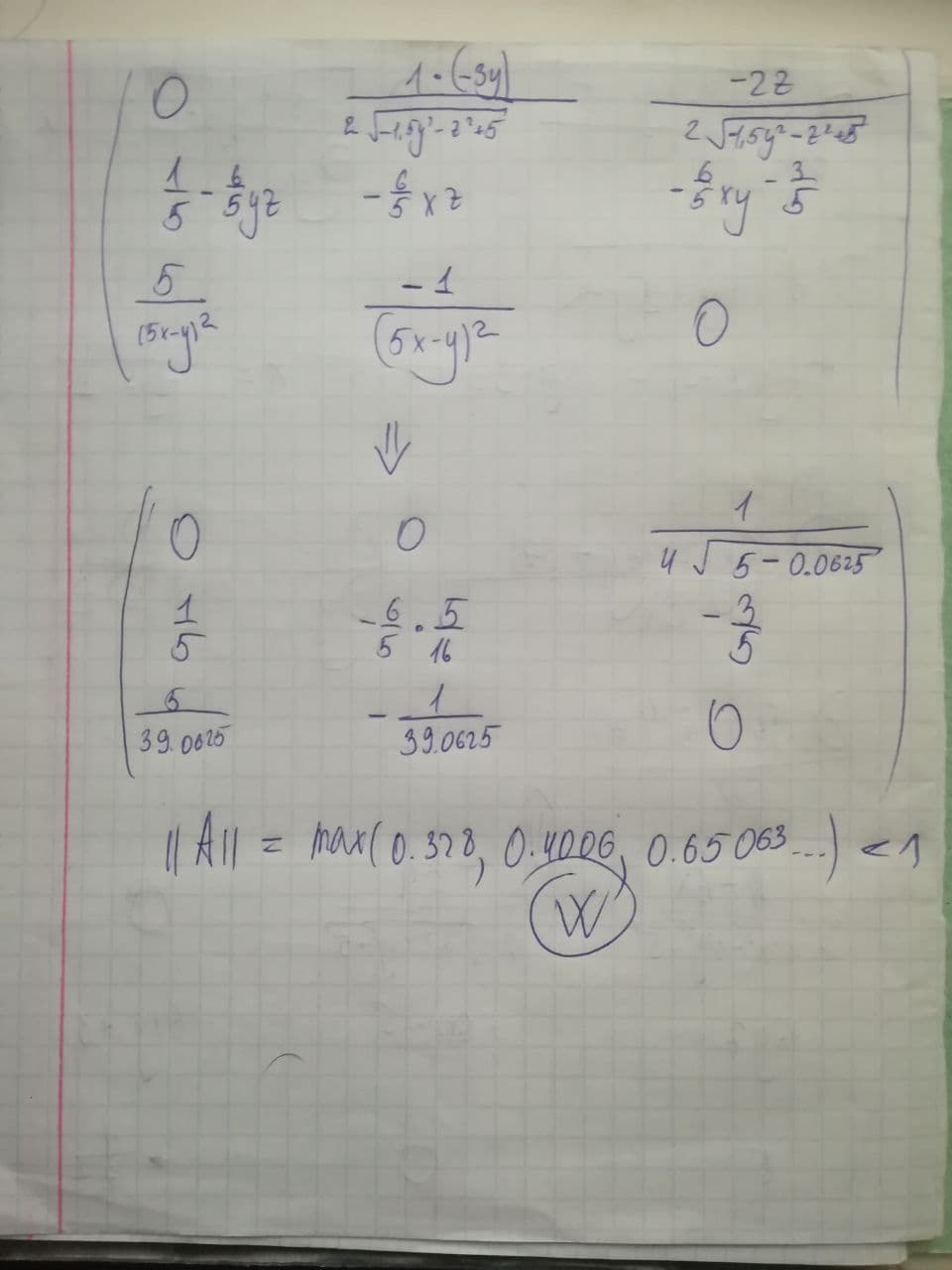
}

**2 Метод простої ітерації**

**Постанова задачі**

****

**Розрахунки**

Матриця Якобі + перевірка на норму  


Код:

const eps = Math.pow(10, -3)

export const solver = () => {

  let prevX = 0

  let prevY = 0

  let prevZ = 0

  let curX = 1.25

  let curY = 0

  let curZ = 0.25

  do {

    prevX = curX

    prevY = curY

    prevZ = curZ

    curX = Math.sqrt((-(1.5 \* prevY \* prevY) - (prevZ \* prevZ) + 5) / 3)

    curY = (-6 / 5 \* prevX \* prevY \* prevZ) + (1 / 5 \* prevX) - (3 / 5 \* prevZ)

    curZ = 1 / (5 \* prevX - prevY)

  } while (!compare([prevX, curX], [prevY, curY], [prevZ, curZ]))

  return [curX, curY, curZ]

}

const compare = (x: [number, number], y: [number, number], z: [number, number]) => {

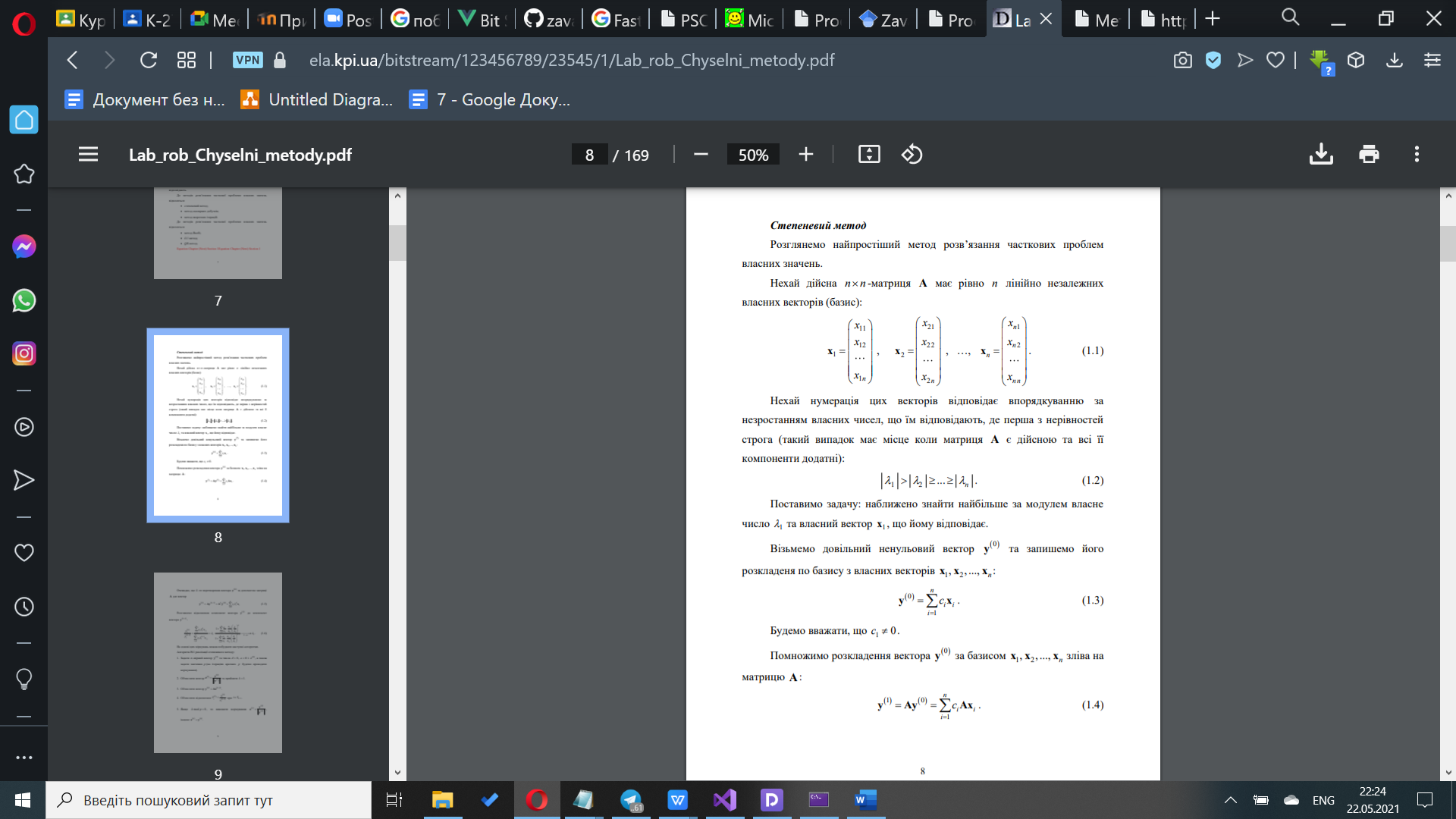
  if (Math.abs(x[0] - x[1]) < eps && Math.abs(y[0] - y[1]) < eps && Math.abs(z[0] - z[1]) < eps)

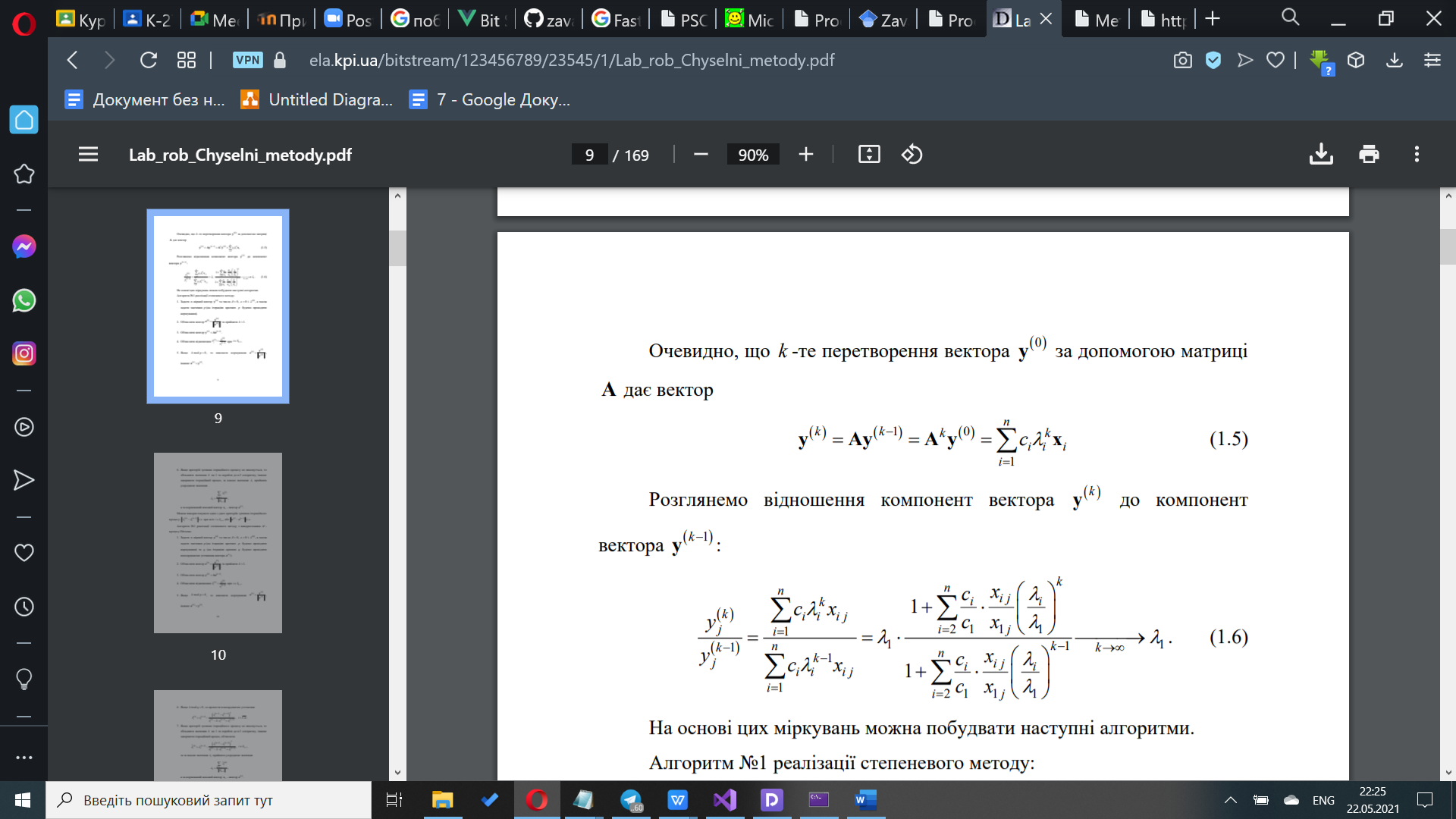
    return true

  return false

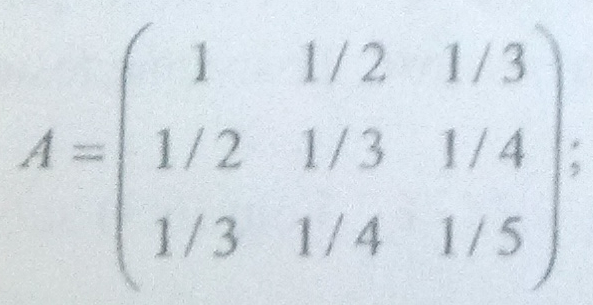
}

**3 Степеневий метод**





**Постановка задачі  
Знайти мін. власне число:**



**Результат**

  
  
  
Код:

type Array3 = [number, number, number] | []

type Array3x3 = [Array3, Array3, Array3]

const eps = Math.pow(10, -3)

export const solver = () => {

  let A: Array3x3 = [

    [1, 1/2, 1/3],

    [1/2, 1/3, 1/4],

    [1/3, 1/4, 1/5]

  ]

  let maxA = matrixNorm(A)

  console.log('\_\_\_\_\_\_\_\_\_\_\_\_-')

  let B: Array3x3 = [[], [], []]

  for (let i = 0; i < 3; i++) {

    for (let j = 0; j < 3; j++) {

      if (i == j)

      {

        B[i][j] = maxA - A[i][j]

      }

      else {

        B[i][j] = (-1) \* A[i][j]

      }

    }

  }

  let maxB = maxEigen(B)

  console.log("\_\_\_\_\_\_\_\_\_\_\_\_\_-")

  return maxA - maxB

}

const matrixNorm = (A: Array3x3) => {

  let max = -1

  for (let i = 0; i < 3; i++) {

    let cur = 0

    for (let j = 0; j < 3; j++) {

      cur += Math.abs(A[i][j])

    }

    if (max < cur) max = cur

  }

  return max

}

const maxEigen = (A: Array3x3) => {

  let currX: Array3 = [-1, -1, 0]

  let prevX: Array3 = [0, 0, 0]

  let currMu = 0

  let prevMu = 0

  do {

    let norm = vectorNorm(currX)

    for (let i = 0; i < 3; i++) {

      prevX[i] = currX[i] / norm

    }

    currX = multiplyMatrixByVector(A, prevX)

    prevMu = currMu

    currMu = dotProduct(prevX, currX)

  } while (Math.abs(currMu - prevMu) > eps)

  return currMu

}

const dotProduct = (x: Array3, y: Array3) => {

  let res = 0

  for (let i = 0; i < 3; i++) {

    res += x[i] \* y[i]

  }

  return res

}

const vectorNorm = (x: Array3) => {

  let res = 0

  for (let i = 0; i < 3; i++) {

    res += x[i] \* x[i]

  }

  return Math.sqrt(res)

}

const multiplyMatrixByVector = (matrix: Array3x3, vector: Array3) => {

  const newVector: Array3 = new Array(3).fill(0) as Array3

  for (let i = 0; i < 3; i++) {

    for (let j = 0; j < 3; j++) {

      newVector[i] += matrix[i][j] \* vector[j]

    }

  }

  return newVector

}