Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота № 2

з дисципліни «Спеціальні розділи математики-2.  
Чисельні методи»

на тему

«**Розв’язання систем лінійних алгебраїчних рівнянь (СЛАР) прямими методами. Звичайний метод Гауса та метод квадратних коренів**»

Виконав:

студент гр. ІС-34

Колосов Ігор

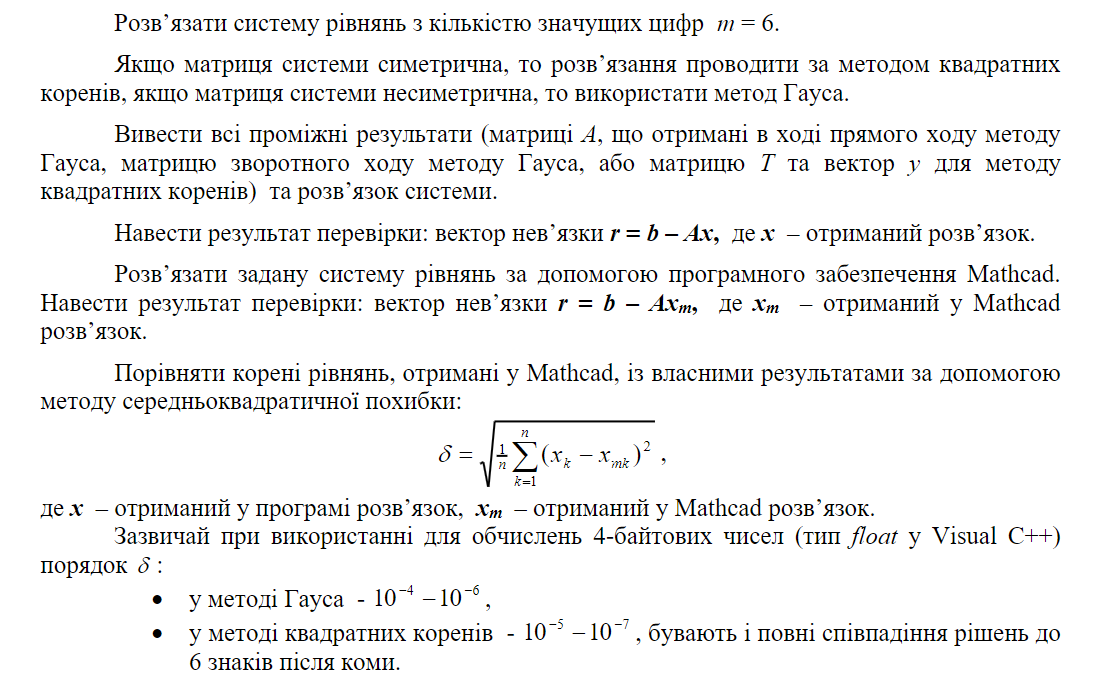
Викладач:

доц. Рибачук Л.В.

Київ – 2024

**Зміст**

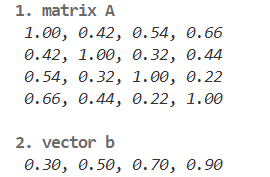
**1. Постановка задачі**

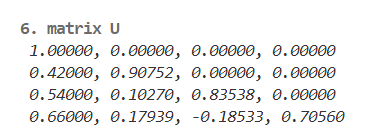


**2. Вихідна система рівнянь**

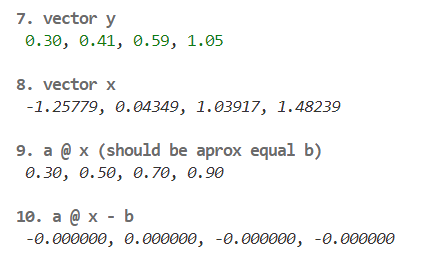


**3. Проміжні результати**

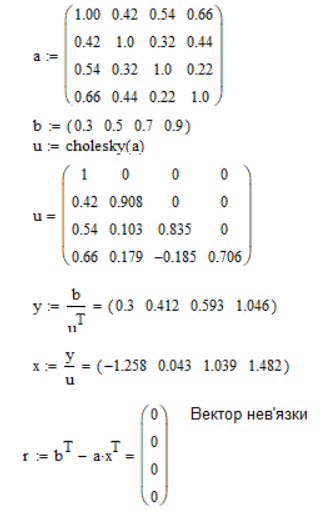
****

****

**4. Вектор нев’язки**

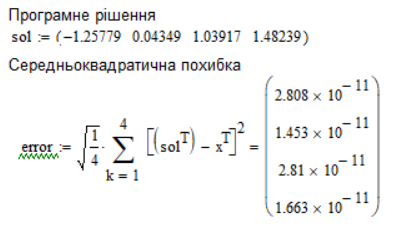
****

**5. Розв’язок задачі у Mathcad**

****

**6. Порівняння розв’язку у Mathcad та власного**

****

****

**7. Лістинг програми**

| import numpy as np  from lib\_print import ANSI, printer, highlight, pidx  counter = pidx()  vector\_b = np.array([0.3, 0.5, 0.7, 0.9]).T  matrix\_a = np.array([  [1.00, 0.42, 0.54, 0.66],  [0.42, 1.00, 0.32, 0.44],  [0.54, 0.32, 1.00, 0.22],  [0.66, 0.44, 0.22, 1.00]  ])  a = matrix\_a.copy()  b = vector\_b.copy()  n = a.shape[0]  u = np.zeros\_like(a)  print(printer(  a, default\_style=ANSI.Styles.ITALIC,  text=f"{ANSI.Styles.BOLD}{counter.str()}matrix A{ANSI.Styles.RESET}"), "\n")  print(printer(  b.reshape(-1, 1), default\_style=ANSI.Styles.ITALIC,  text=f"{ANSI.Styles.BOLD}{counter.str()}vector b{ANSI.Styles.RESET}"), "\n")  h\_ij = highlight([], ANSI.FG.GREEN, 0, "case i == j")  h\_else = highlight([], ANSI.FG.BLUE, 0, "case else")  h\_ki = highlight([], ANSI.FG.YELLOW, 0, "u[k, i] at else case")  h\_kj = highlight([], ANSI.FG.CYAN , 0, "u[k, j] at else case")  for i in range(n):  for j in range(i, n):  if i == j:  sum = np.sum([u[k, i]\*\*2 for k in range(i)])  u[i, i] = np.sqrt(a[i, i] - sum)  h\_ij.indicies.append((i, i))  else:  sum = np.sum([u[k, i] \* u[k, j] for k in range(i)])  u[i, j] = (a[i, j] - sum) / u[i, i] # чому тут верхня трикутна матриця  for k in range(i):  h\_kj.indicies.append((k, j))  h\_ki.indicies.append((k, i))  h\_else.indicies.append((i, j))  print(printer(a,  f"{ANSI.Styles.BOLD}{counter.str()}matrix A\n -> (values taken from A at decomposition step) {ANSI.Styles.RESET}", [h\_ij, h\_else],  print\_description=True, default\_style=ANSI.FG.BRIGHT\_BLACK), "\n")  print(printer(  u, formatting="5.5f", higlights=[h\_ki], default\_style=ANSI.FG.BRIGHT\_BLACK, print\_description=True,  text=f"{ANSI.Styles.BOLD}{counter.str()}matrix U (during decomposition){ANSI.Styles.RESET}"), "\n")  print(printer(  u, formatting="5.5f", higlights=[h\_kj], default\_style=ANSI.FG.BRIGHT\_BLACK, print\_description=True,  text=f"{ANSI.Styles.BOLD}{counter.str()}matrix U (during decomposition){ANSI.Styles.RESET}"), "\n")  print(printer(  u, formatting="5.5f", default\_style=ANSI.Styles.ITALIC,  text=f"{ANSI.Styles.BOLD}{counter.str()}matrix U{ANSI.Styles.RESET}"), "\n")  # T'y = b  y = np.zeros\_like(b)  for i in range(n):  sum = np.sum([u[k, i] \* y[k] for k in range(i)])  y[i] = (b[i] - sum) / u[i, i]  print(printer(  y.reshape(-1, 1), default\_style=ANSI.FG.GREEN,  text=f"{ANSI.Styles.BOLD}{counter.str()}vector y{ANSI.Styles.RESET}"), "\n")  # Tx = y  x = np.zeros\_like(b)  for i in range(n-1, -1, -1):  sum = np.sum([u[i, k] \* x[k] for k in range(i+1, n)])  x[i] = (y[i] - sum) / u[i, i]  print(printer(  x.reshape(-1, 1), default\_style=ANSI.Styles.ITALIC, formatting="5.5f",  text=f"{ANSI.Styles.BOLD}{counter.str()}vector x{ANSI.Styles.RESET}"), "\n")  print(printer(  (a @ x).reshape(-1, 1), default\_style=ANSI.Styles.ITALIC,  text=f"{ANSI.Styles.BOLD}{counter.str()}a @ x (should be aprox equal b) {ANSI.Styles.RESET}"), "\n")  print(printer(  (a @ x - b).reshape(-1, 1), default\_style=ANSI.Styles.ITALIC, formatting=".6f",  text=f"{ANSI.Styles.BOLD}{counter.str()}a @ x - b {ANSI.Styles.RESET}"), "\n")  print(printer(  a - u.T @ u, default\_style=ANSI.Styles.ITALIC, formatting=".6f",  text=f"{ANSI.Styles.BOLD}{counter.str()}a - u.T @ u {ANSI.Styles.RESET}"), "\n") |
| --- |

**8. Лістинг програми; Додаток: libprint.py**

| import numpy as np  import dataclasses  from typing import List, Tuple  @dataclasses.dataclass  class highlight:  indicies:List[Tuple[int, int]]  style:str  priority: int # 0 heighest  description:str  def printer(  array:np.ndarray,  text:str = "",  higlights: List[highlight] = [],  formatting:str = "0.2f",  separator:str = ", ",  pre\_row\_str:str = " ",  print\_description:bool = False,  print\_text: bool = True,  default\_style:str = "",  reset\_style:str = "\033[0m",  ):  higlights = sorted(higlights, key=lambda h: h.priority, reverse=True)  if not higlights:  higlights = [highlight([], ANSI.Styles.RESET, 0, "")]  output = [["" for \_ in range(array.shape[0])] for \_ in range(array.shape[1])]  descriptions = ""  for n, hlght in enumerate(higlights):  for j, row in enumerate(array):  for i, val in enumerate(row):  if (i, j) in hlght.indicies:  output[i][j] = f"{f'{hlght.style}{val:{formatting}}'}{reset\_style}"  elif output[i][j] == "":  output[i][j] = f"{f'{default\_style}{val:{formatting}}{reset\_style}'}"  descriptions += f" \* {n: 2d}. {hlght.style}{hlght.description}{reset\_style}\n"  mat\_str = pre\_row\_str + f"\n{pre\_row\_str}".join(separator.join(map(str, row)) for row in output)    out = ""  if print\_text:  out += text + ANSI.Styles.RESET + "\n"  if print\_description:  out += descriptions + ""  out += mat\_str    return out  class pidx:  def \_\_init\_\_(self):  self.counter = 0    def get(self):  self.counter += 1  return self.counter  def str(self):  return f"{self.get(): 2d}. "  class ANSI:  class Styles:  RESET = "\033[0m"  BOLD = "\033[1m"  DIM = "\033[2m"  ITALIC = "\033[3m"  UNDERLINE = "\033[4m"  BLINK = "\033[5m"  REVERSE = "\033[7m"  HIDDEN = "\033[8m"  STRIKETHROUGH = "\033[9m"  class FG:  BLACK = "\033[30m"  RED = "\033[31m"  GREEN = "\033[32m"  YELLOW = "\033[33m"  BLUE = "\033[34m"  MAGENTA = "\033[35m"  CYAN = "\033[36m"  WHITE = "\033[37m"  # Bright colors  BRIGHT\_BLACK = "\033[90m"  BRIGHT\_RED = "\0ANSI.Sty33[91m"  BRIGHT\_GREEN = "\033[92m"  BRIGHT\_YELLOW = "\033[93m"  BRIGHT\_BLUE = "\033[94m"  BRIGHT\_MAGENTA = "\033[95m"  BRIGHT\_CYAN = "\033[96m"  BRIGHT\_WHITE = "\033[97m"  class BG:  BLACK = "\033[40m"  RED = "\033[41m"  GREEN = "\033[42m"  YELLOW = "\033[43m"  BLUE = "\033[44m"  MAGENTA = "\033[45m"  CYAN = "\033[46m"  WHITE = "\033[47m"  # Bright colors  BRIGHT\_BLACK = "\033[100m"  BRIGHT\_RED = "\033[101m"  BRIGHT\_GREEN = "\033[102m"  BRIGHT\_YELLOW = "\033[103m"  BRIGHT\_BLUE = "\033[104m"  BRIGHT\_MAGENTA = "\033[105m"  BRIGHT\_CYAN = "\033[106m"  BRIGHT\_WHITE = "\033[107m"  if \_\_name\_\_ == "\_\_main\_\_":  mat = np.random.random((5, 5))  h1 = highlight([(0, 0), (4, 4)], ANSI.Styles.BOLD + ANSI.BG.GREEN, 0, "aboba1")  h2 = highlight([(4, 2), (2, 1)], ANSI.Styles.ITALIC + ANSI.BG.RED, 0, "aboba2")  mat\_str = printer(mat, "cool text phronebius id k ksk kssk sk", [h1, h2])  print(mat\_str) |
| --- |