**ИНСТИТУТ ТРАНСПОРТА И СВЯЗИ**



ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК И ТЕЛЕКОММУНИКАЦИЙ

КАФЕДРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОМПЬЮТЕРНЫХ СИСТЕМ

**Лабораторная работа №1**

по дисциплине

„Численные методы”

Тема: «Методы решения системы линейных уравнений.

Число обусловленности матрицы.»

Выполнил: Денис Белов, Андрей Савкин, Евгений Хрущ

Рига.

2020.

1. Формулировка задания

В данной работе необходимо было реализовать метод исключения Гаусса с ведущим элементом и индивидуальный метод: метод Гаусса-Зейделя. В зависимости является ли метод итерационным или прямым, получить результат приведённых примеров.

Так же реализовать экспериментальный расчёт числа обусловленности матрицы и получить результаты двух систем.

1. Метод исключения Гаусса с ведущим элементом

## ****gauss\_elimination.py****

|  |  |
| --- | --- |
|  | def gauss\_elimination(matrix: list, vars: list, print\_only\_results: bool = False, matrix\_name: str = "") -> list: |
|  | """ |
|  | returns list of solution variables X |
|  | """ |
|  |  |
|  | matrix\_len = len(matrix) |
|  | matrix\_copy = copy.deepcopy(matrix) |
|  | print('\n-------------------------------------Gauss Elimination - ' + matrix\_name) |
|  | if not print\_only\_results: |
|  | print('Length of the matrix (n):', matrix\_len) |
|  |  |
|  | print('Matrix before pivotisation (initial):\n') |
|  | for (var, row) in zip(vars, matrix\_copy): |
|  | print(var, \*["%0.4f" % elem for elem in row], sep='\t') |
|  |  |
|  | # Find the pivot element - we need to put it as the first row in the matrix |
|  | for i in range(matrix\_len): |
|  | for k in range(i + 1, matrix\_len): |
|  | if abs(matrix\_copy[i][i]) < abs(matrix\_copy[k][i]): |
|  | for j in range(0, matrix\_len + 1): |
|  | # Swapping elements |
|  | matrix\_copy[i][j], matrix\_copy[k][j] = matrix\_copy[k][j], matrix\_copy[i][j] |
|  |  |
|  | if not print\_only\_results: |
|  | print('\nMatrix after pivotisation:\n') |
|  | for (var, row) in zip(vars, matrix\_copy): |
|  | print(var, \*["%0.4f" % elem for elem in row], sep='\t') |
|  |  |
|  | # Main Gauss Elimination loop |
|  | # Forward elimination -- Straight step (Nulling the bottom-left corner) |
|  | for i in range(matrix\_len - 1): |
|  | for k in range(i + 1, matrix\_len): |
|  | coefficient = matrix\_copy[k][i] / matrix\_copy[i][i] # Coefficient |
|  |  |
|  | # Make the elements below the pivot elements equal to zero |
|  | # or eliminate the variables |
|  | for j in range(0, matrix\_len + 1): |
|  | matrix\_copy[k][j] -= coefficient \* matrix\_copy[i][j] |
|  |  |
|  | if not print\_only\_results: |
|  | print('\nMatrix after gauss elimination:\n') |
|  | for (var, row) in zip(vars, matrix\_copy): |
|  | print(var, \*["%0.4f" % elem for elem in row], sep='\t ') |
|  |  |
|  | # List of variables values (x, y, z, ...) |
|  | # Initializing all variables values with zero |
|  | vars\_values = [0] \* matrix\_len |
|  |  |
|  | # Back substitution -- Reversed step (Nulling upper-right corner) |
|  | for i in range(matrix\_len - 1, -1, -1): |
|  | # Make the variable to be calculated equal to the rhs of the |
|  | # last equation |
|  | vars\_values[i] = matrix\_copy[i][matrix\_len] |
|  |  |
|  | for j in range(i + 1, matrix\_len): |
|  | # Subtracting all the lhs values except the coefficient |
|  | # of the variable whose value is being calculated |
|  | if j != i: |
|  | vars\_values[i] -= matrix\_copy[i][j] \* vars\_values[j] |
|  |  |
|  | # Finally, divide the rhs by the coefficient of the variable |
|  | # to be calculated |
|  | vars\_values[i] /= matrix\_copy[i][i] |
|  |  |
|  | print('\nSolution:') |
|  | for (var, val) in zip(vars, vars\_values): |
|  | print(var, '= %0.4f' %(val)) |
|  |  |
|  | print('-------------------------------------\n') |
|  |  |
|  | return vars\_values |

1. Индивидуальный метод: метод Гаусса-Зейделя

|  |
| --- |
|  |
| def gauss\_seidel(equations: list, vars: list, e: float, print\_only\_results: bool = False, matrix\_name: str = "", matrix: list = None) -> list: |
|  | """ |
|  | equations -- list of lambda equations with any count of arguments. |
|  | Example: [lambda x, y: x + y, lambda x, y: x - y] |
|  |  |
|  | vars -- list of symbols to specify the names of the equations variables. |
|  | Example: ['x', 'y', 'z'] |
|  |  |
|  | e -- tolerable error |
|  | Example: 0.001 |
|  |  |
|  | matrix -- the whole matrix (AB) |
|  |  |
|  | Note: you can also pass the matrix instead of equations |
|  |  |
|  | returns list of solution variables X |
|  | """ |
|  |  |
|  | # Reading tolerable error (required accuracy) |
|  | #e = float(input('Enter tolerable error: ')) |
|  |  |
|  | # Implementation of Gauss Seidel Iteration |
|  | print('\n-------------------------------------Gauss Seidel - ' + matrix\_name) |
|  | print('Using equations list...') if matrix is None else print ('Using matrix...') |
|  | print('With accuracy of ' + str(e) + '\n') |
|  | if not print\_only\_results: |
|  | print('Iter', \*vars, sep = "\t") |
|  |  |
|  | condition = True |
|  |  |
|  | # Initializing all variables values with zero if using equations |
|  | # or set each X as B if using matrix, as the first approximation |
|  | vars\_values = [0] \* len(equations) if matrix is None else [row[-1] for row in matrix] |
|  |  |
|  | if not print\_only\_results: |
|  | print(0, \*["%0.4f" % elem for elem in vars\_values], sep="\t") |
|  |  |
|  | iteration = 1 |
|  |  |
|  | while condition: |
|  | e\_list = [] |
|  |  |
|  | # Calculating all variables |
|  | if matrix is None: |
|  | iterate\_equations(equations, e\_list, vars\_values) |
|  | else: |
|  | iterate\_matrix(matrix, e\_list, vars\_values) |
|  |  |
|  | if not print\_only\_results: |
|  | print(iteration, \*["%0.4f" % elem for elem in vars\_values], sep="\t") |
|  | iteration += 1 |
|  |  |
|  | # Checking if all current errors are greater than required error e |
|  | condition = check\_error\_rate(e\_list, e) |
|  |  |
|  | print('\nSolution:') |
|  | for (var, val) in zip(vars, vars\_values): |
|  | print(var,'= %0.3f' %(val)) |
|  |  |
|  | print('-------------------------------------\n') |
|  |  |
|  | return vars\_values |
|  |  |
|  | def iterate\_equations(equations, e\_list, vars\_values): |
|  | # Calculating all variables |
|  | for i, eq in enumerate(equations): |
|  | # Calculating the i-th lambda of equations list |
|  | new\_value = eq(\*vars\_values) |
|  |  |
|  | # Adding i-th error to the e\_list |
|  | e\_list.append(abs(vars\_values[i] - new\_value)) |
|  |  |
|  | # Set current i-th vars\_values variable to it's newly calculated new\_value |
|  | vars\_values[i] = new\_value |
|  |  |
|  | def iterate\_matrix(matrix, e\_list, vars\_values): |
|  | # Calculating all variables |
|  | for i, row in enumerate(matrix): |
|  | # Calculating the i-th X variable |
|  | # Initializing current X with B coefficient value |
|  | new\_value = row[-1] |
|  | for j in range(0, len(row) - 1): |
|  | if i != j: |
|  | # Subtracting the non-diagonal values multiplied by calculated previously values |
|  | new\_value -= (row[j] \* vars\_values[j]) |
|  |  |
|  | # Adding the value on a diagonal of the matrix |
|  | new\_value \*= 1/row[i] |
|  |  |
|  | # Adding i-th error to the e\_list |
|  | e\_list.append(abs(vars\_values[i] - new\_value)) |
|  |  |
|  | # Set current i-th vars\_values variable to it's newly calculated new\_value |
|  | vars\_values[i] = new\_value |
|  |  |
|  | def check\_error\_rate(e\_list: list, e): |
|  | return all([current\_e > e for current\_e in e\_list]) |
|  |  |

1. Экспериментальное определение числа обусловленности матрицы

|  |
| --- |
| def solve\_with\_gauss\_elimination(matrix: list, vars: list, print\_only\_results: bool = False, matrix\_name: str = ""): |
|  | """ Solves the specified matrix using the Gauss Elimination |
|  | """ |
|  | vectorX = ge.gauss\_elimination(matrix, vars, print\_only\_results, matrix\_name) |
|  | summary\_data = summary(matrix, vars, vectorX, print\_only\_results, matrix\_name) |
|  | summary\_gauss\_elimination(summary\_data) |
|  |  |
|  | def solve\_with\_gauss\_seidel(matrix: list, equations: list, vars: list, e: float, print\_only\_results: bool = False, |
|  | matrix\_name: str = "", use\_matrix: bool = False): |
|  | """ Solve the specified equations using the Gauss Seidel |
|  | """ |
|  | vectorX = gs.gauss\_seidel(equations, vars, e, print\_only\_results, matrix\_name, matrix if use\_matrix else None) |
|  | summary\_data = summary(matrix, vars, vectorX, print\_only\_results, matrix\_name) |
|  | summary\_data.add\_equations(equations) |
|  | summary\_data.add\_errors\_vec(e) |
|  | summary\_data.set\_use\_matrix(use\_matrix) |
|  | summary\_gauss\_seidel(summary\_data) |
|  |  |
|  | class SummaryData: |
|  | def \_\_init\_\_(self, matrix: list, adjusted\_matrix: list, vars: list, X: list, AX: list, |
|  | B: list, condA: float, deltaB: list, print\_only\_results: bool = False, matrix\_name: str = ""): |
|  | self.matrix = matrix |
|  | self.adjusted\_matrix = adjusted\_matrix |
|  | self.vars = vars |
|  | self.X = X |
|  | self.AX = AX |
|  | self.B = B |
|  | self.condA = condA |
|  | self.deltaB = deltaB |
|  | self.print\_only\_results = print\_only\_results |
|  | self.matrix\_name = matrix\_name |
|  |  |
|  | def add\_equations(self, equations: list): |
|  | self.equations = equations |
|  |  |
|  | def add\_errors\_vec(self, e: list): |
|  | self.e = e |
|  |  |
|  | def set\_use\_matrix(self, use\_matrix: bool): |
|  | self.use\_matrix = use\_matrix |
|  |  |
|  | def summary(matrix: list, vars: list, vectorX: list, print\_only\_results: bool = False, matrix\_name: str = "") -> SummaryData: |
|  | """ Summarizes the data available after solving the matrix |
|  | """ |
|  | if not print\_only\_results: |
|  | show\_A\_B(matrix) |
|  | AX = get\_matrixAX(matrix, vectorX) |
|  | B = get\_vectorB\_unpacked(matrix) |
|  | if not print\_only\_results: |
|  | show\_AX\_B(matrix, vectorX) |
|  | condA = get\_matrix\_cond(get\_matrixA(matrix)) |
|  | print("\nCondition number Cond(A) of this matrix is =", condA) |
|  | deltaB = get\_deltaB(matrix) |
|  | print("\nDeltaB vector:", deltaB) |
|  | adjusted\_matrix = append\_vectorB(matrix, array(B) + array(deltaB)) |
|  | if not print\_only\_results: |
|  | full\_print(adjusted\_matrix, vars, 'Modified (adjusted) matrix') |
|  |  |
|  | return SummaryData(matrix, adjusted\_matrix, vars, vectorX, AX, B, condA, deltaB, print\_only\_results, matrix\_name) |
|  |  |
|  | def summary\_gauss\_elimination(data: SummaryData): |
|  | deltaX = ge.gauss\_elimination(data.adjusted\_matrix, data.vars, data.print\_only\_results, data.matrix\_name + " with B + deltaB") |
|  | if not data.print\_only\_results: |
|  | print("DeltaX vector = ", deltaX) |
|  | check\_condition\_number(data.matrix, data.X, deltaX) |
|  |  |
|  | def summary\_gauss\_seidel(data: SummaryData): |
|  | deltaX = gs.gauss\_seidel(data.equations, data.vars, data.e, data.print\_only\_results, |
|  | data.matrix\_name + " with B + deltaB", data.adjusted\_matrix if data.use\_matrix else None) |
|  | if not data.print\_only\_results: |
|  | print("DeltaX vector = ", deltaX) |
|  | check\_condition\_number(data.matrix, data.X, deltaX) |

1. Вывод

Сравнивая метод Гаусса с ведущим элементом и метод Гаусса-Зейделя, можно заметить, что … для вычисления системы 2 из задания.

Так же если рассматривать расчёт числа обусловленности матрицы, то величина погрешности у систем 3 и 5 из задания будет….