

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ЗАКЛАД
«ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА»

Навчально-науковий інститут математики та інформаційних технологій

Кафедра математики та інформатики

Кольвенко Катерина Сергіївна

**РОЗРОБКА ДОДАТКА РОЗВ'ЯЗУВАННЯ СИСТЕМ ЛІНІЙНИХ
АЛГЕБРАЇЧНИХ РІВНЯНЬ**

**кваліфікаційна робота
здобувача вищої освіти другого (магістерського) рівня
освітньої програми «Інформатика»
за спеціальністю 014.09 Середня освіта (Інформатика)**

Особистий підпис – _____

Науковий керівник – _____ проф., д.т.н. Юрій КОЗУБ

В.о. зав. кафедри – _____ проф., д.т.н. Юрій КОЗУБ

Полтава – 2024

АНОТАЦІЯ

Кваліфікаційна робота: 52 с., 3 рис., 4 табл., 28 джерел, 5 додатків.

Мета роботи – розробити алгоритми реалізації методів Гауса та Зейделя для розв’язання СЛАР у вигляді програмного додатку.

Об’єкт проектування – системи лінійних алгебраїчних рівнянь.

Методи дослідження – методи Гауса та Зейделя для розв’язання систем лінійних алгебраїчних рівнянь.

Результати роботи. В ході виконання дипломного проекту було створено додаток операційної системи Windows з графічним інтерфейсом користувача для вирішення систем лінійних алгебраїчних рівнянь прямими та ітераційними методами. Було реалізовано два досить простих методи: це метод Гауса і метод Зейделя, та їхні більш ефективні варіанти – LU-розклад і метод релаксації. Перевага цих методів в тому, що вони дозволяють без додаткових обчислень швидко знаходити розв’язання системи з різними значеннями вільних членів.

Ці методи було реалізовано у вигляді модуля Delphi, що може бути використаний підчас розробки інших додатків.

Висновок. У результаті розробки отримано програмне забезпечення, що застосовується для розв’язання систем лінійних алгебраїчних рівнянь.

Ключові слова. АЛГОРИТМ, СИСТЕМА ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ, МЕТОД ГАУСА, МЕТОД ЗЕЙДЕЛЯ, LU-РОЗКЛАД, МЕТОД ВЕРХНЬОЇ РЕЛАКСАЦІЇ, АЛГОРИТМ, ФОРМА.

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ

СЛАР – система лінійних алгебраїчних рівнянь

WYSIWYG – What You See Is What You Get – «що бачиш, те й отримуєш»

ЕОМ – електронно-обчислювальна машина

VCL – Visual Component Library

ЗМІСТ

Вступ.....	5
РОЗДІЛ 1. Існуючі програмні продукти для розв’язання СЛАР	7
РОЗДІЛ 2. Системи лінійних алгебраїчних рівнянь	11
2.1. Визначення системи лінійних алгебраїчних рівнянь	11
2.2. Методи розв’язання СЛАР	12
2.2.1. Прямі та ітераційні методи	12
2.2.2. Розв’язання систем алгебраїчних лінійних рівнянь методом Гауса.....	13
2.2.3. Розв’язання систем алгебраїчних лінійних рівнянь методом Зейделя.....	19
2.3. Сучасні методи розв’язання СЛАР	20
2.3.1. Метод релаксації.....	20
2.3.2. Метод LU-розкладу	22
<i>Висновки до розділу 2</i>	<i>28</i>
РОЗДІЛ 3. Розробка додатку для вирішення СЛАР.....	29
3.1. Опис середовища розробки Delphi	29
3.2. Опис модулів та класів додатку.....	32
3.3. Інтерфейс головного вікна	36
<i>Висновки до розділу 3</i>	<i>37</i>
Висновки.....	38
Література.....	39
РОЗДІЛ 4. Додаток А. Лістинг модуля Matrix.pas	42
Додаток Б. Лістинг модуля Vector.pas	44
Додаток В. Лістинг модуля slesolvers.pas.....	45
Додаток Г. Лістинг модуля unitmain.pas.....	47
Додаток Д. Опис використаних компонентів	51

ВСТУП

Актуальність обраної теми полягає в тому, що у більшості розрахункових математичних задач є потреба в розв'язанні систем лінійних алгебраїчних рівнянь (СЛАР) [1]. Це не дивно, оскільки математичні моделі тих або інших процесів або відразу будуються як лінійні, або зводяться до таких за допомогою дискретизації або лінеаризації. Необхідність розв'язання СЛАР виникає при розв'язанні багатовимірних анізотропних крайових задач, в задачах обчислювальної гідродинаміки, в теорії електричних ланцюгів, в задачах управління і контролю, при розв'язанні рівнянь балансів і збереження в механіці, гідравліці, в завданнях оцінки і пророцтва критичних ситуацій та ін.

Тому важко переоцінити роль, яку грає вибір ефективного методу розв'язання СЛАР. Сучасна обчислювальна математика має в розпорядженні великий арсенал методів, а математичне забезпечення ЕОМ – багато пакетів прикладних програм, що дозволяють вирішувати різні лінійні системи. Щоб орієнтуватися серед методів і програм та в потрібний момент зробити оптимальний вибір, треба розбиратися в основах побудов методів і алгоритмів розв'язання СЛАР.

Практичною цінністю даної роботи є те, що розроблені модулі для розв'язання систем лінійних алгебраїчних рівнянь можна в подальшому використовувати при розробці програм в середовищі Delphi.

Мета роботи – розробити алгоритми реалізації розв'язання СЛАР методами Гауса та Зейделя у вигляді програмного додатку.

Об'єкт дослідження – системи лінійних алгебраїчних рівнянь.

Предмет дослідження – методи Гауса та Зейделя та їх складові методи релаксації і LU-розкладу для розв'язання систем лінійних алгебраїчних рівнянь.

Задачі:

- Розглянути підходи до побудови методів розв'язання СЛАР.
- Розробити алгоритми обраних методів.

- Реалізувати ці алгоритми у вигляді додатка.
- Проаналізувати результати роботи алгоритмів.

Структура роботи складається з чотирьох розділів. В першому розділі розглянуто програми для розв'язання систем лінійних рівнянь. Другий розділ присвячено методам розв'язання СЛАР, розглянуто алгоритми, які реалізовано в програмі. В третьому – описано процес створення додатку для розв'язання СЛАР. Четвертий розділ містить питання з охорони праці.

РОЗДІЛ 1. ІСНУЮЧІ ПРОГРАМНІ ПРОДУКТИ ДЛЯ РОЗВ'ЯЗАННЯ СЛАР

MATLAB — продукт для числового аналізу з вбудованою мовою програмування. Створена компанією The MathWorks, це досить простий засіб для роботи з математичними матрицями, малювання функцій, роботи з алгоритмами, створення робочих оболонок (user interfaces) з програмами в інших мовах програмування. Хоча цей продукт спеціалізується на чисельному обчисленні, спеціальні інструментальні засоби працюють з програмним забезпеченням Maple, що робить його повноцінною системою для роботи з алгеброю.

MATLAB має більше ніж мільйон користувачів на виробництвах і науковців. Ціна базової комерційної версії без інструментів близько 2000 дол. США.

MATLAB надає користувачеві велику кількість функцій для аналізу даних, які покривають майже всі області математики, зокрема:

- Матриці та лінійна алгебра — алгебра матриць, лінійні рівняння, власні значення і вектори, сингулярності, факторизація матриць та інше.
- Многочлени та інтерполяція — корені многочленів, операції над многочленами та їх диференціювання, інтерполяція та екстраполяція кривих...
- Математична статистика та аналіз даних — статистичні функції, статистична регресія, цифрова фільтрація, швидке перетворення Фур'є та інші.
- Обробка даних — набір спеціальних функцій, включаючи побудову графіків, оптимізацію, пошук нулів, чисельне інтегрування та інше.
- Диференційні рівняння — вирішення диференційних і диференційно-алгебраїчних рівнянь, диференційних рівнянь із

запізнюванням, рівнянь з обмеженнями, рівнянь в часткових похідних та інше.

- Розріджені матриці — спеціальний клас даних пакету MATLAB, що використовується у спеціалізованих додатках.
- Цілочисельна арифметика — виконання операцій цілочисельної арифметики в середовищі MATLAB.

Mathcad — система комп'ютерної алгебри з класу систем автоматизованого проектування, орієнтована на підготовку інтерактивних документів з обчисленнями і візуальним супроводженням, відрізняється легкістю використання і застосування для колективної роботи.

Mathcad має простий і інтуїтивний для використання інтерфейс користувача. Для введення формул і даних можна використовувати як клавіатуру, так і спеціальні панелі інструментів.

Деякі з математичних можливостей Mathcad (версії до 13.1 включно) засновані на підмножині системи комп'ютерної алгебри Maple (МКМ, Maple Kernel Mathsoft). Версії 14 та 15 використовують символічне ядро MuPAD.

Робота здійснюється в межах робочого аркуша, на якому рівняння і вирази відображаються графічно, на противагу текстовому запису в мовах програмування. При створенні документів-програм використовується принцип WYSIWYG (What You See Is What You Get – «що бачиш, те й отримуєш»).

Незважаючи на те, що ця програма здебільшого орієнтована на користувачів-непрограмістів, Mathcad також використовується в складніших проектах, щоб візуалізувати результати математичного моделювання, шляхом використання найбільш поширених обчислень і традиційних мов програмування.

Mathcad містить сотні операторів і вбудованих функцій для вирішення різних технічних завдань. Програма дозволяє виконувати чисельні і символічні обчислення, проводити операції з скалярними величинами, векторами і матрицями, автоматично переводити одні одиниці вимірювання в інші.

Серед можливостей Mathcad є:

- Розв'язання диференціальних рівнянь, в тому числі і чисельними методами.
- Побудова двовимірних і тривимірних графіків (в різних системах координат, контурні, векторні тощо).
- Використання грецького алфавіту (верхній і нижній регістр) як в тексті, так і у рівняннях.
- Символьні обчислення.
- Операції з векторами і матрицями.
- Символьне розв'язання систем рівнянь.
- Згладжування кривих.
- Виконання підпрограм.
- Знаходження коренів функцій і поліномів.
- Статистичні функції і розподіли ймовірностей.
- Пошук власних значень і власних векторів.
- Обчислення з розмірностями.

Maple – комерційна система комп'ютерної алгебри від компанії Waterloo Maple Inc. Першу версію було розроблено та оприлюднено в 1980-му році групою Symbolic Computation Group з університету Ватерлоо, місто Ватерлоо, Онтаріо, Канада. Остання версія містить понад 5000 функцій для більшості розділів сучасної математики, моделювання та інтерактивної візуалізації, підтримує мову програмування Maple, і дозволяє комбінувати алгоритми, результати обчислення, математичні формули, текст, графіку, діаграми та анімацію зі звуком в електронному документі.

Можливості Maple:

- символічні обчислення і чисельні методи
- математичні функції та методи
- розв'язування рівнянь
- диференціальні рівняння
- лінійна алгебра

- оптимізація
- програмування
- операції з розмірностями та одиницями вимірювання величин
- редактор математичних формул
- візуалізація, графіки, інтерактивні меню та асистенти
- шаблони-приклади для стандартних проблем
- елементи для розробки графічних інтерфейсів
- доступ до MapleCloud-сховища для обміну документами між користувачами та колегами
- понад 30 палітр відсортованих для створення та редагування математичних виразів
- розпізнавання рукописних формул
- інструментарій для фінансового моделювання
- статистичне моделювання
- фізичні моделі.

Висновки до розділу 1

Більшість програм комп'ютерної алгебри наділені можливостями розв'язання систем лінійних рівнянь, але вони є громіздкими та в більшості своїй платними. Тому використання цих програм лише для вирішення вузького класу задач лінійної алгебри є не раціональним.

СЛАР називають однорідною, якщо $b_1 = b_2 = \dots = b_m = 0$. Інакше її називають неоднорідною.

Рішенням СЛАР, та і взагалі всякої системи рівнянь, називають такий набір невідомих x_1^0, \dots, x_n^0 , при підстановці яких кожне рівняння системи перетворюється на тотожність. Будь-яке конкретне рішення СЛАР також називають її особистим рішенням.

СЛАР називають сумісною, якщо вона має які або рішення. Інакше її називають не сумісною. Однорідна СЛАР завжди сумісна, оскільки нульовий набір значень її невідомих завжди є рішенням. Для неоднорідних СЛАР можливі різні випадки.

Якщо СЛАР має рішення, і притому єдине, то її називають визначеною, а якщо рішення неєдине - те невизначеною. При $m=n$, тобто коли в системі 1 кількість рівнянь співпадає з кількістю невідомих, СЛАР називають квадратною.

Ми зупинимося на рішенні тільки таких СЛАР, у яких матриця є квадратною і невиродженою. В цьому випадку система має рішення і притому єдине. Для його знаходження використовують різні методи.

2.2.Методи розв'язання СЛАР

2.2.1. Прямі та ітераційні методи

Будь-який чисельний метод лінійної алгебри можна розглядати як деяку послідовність виконання арифметичних операцій над елементами вхідних даних.

Вживані на практиці чисельні методи рішення СЛАР діляться на дві групи - прямі і ітераційні.

У *прямих* (чи *точних*) методах рішення системи отримують за кінцеве число арифметичних дій. До них відносяться відоме правило Крамера знаходження рішення за допомогою визначників, метод послідовного виключення невідомих (метод Гауса) і його модифікації, метод прогону і інші.

матрицю коефіцієнтів системи (2), через

$$\mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix} - \text{стовпець її вільних членів, і через}$$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} - \text{стовпець з невідомих (вільний вектор)}$$

тоді система (2) може бути записана у вигляді матричного рівняння $A\mathbf{x}=\mathbf{b}$.

При рішенні СЛАР методом Гауса всілякі перетворення проводять не над рівняннями (2), а над так званою розширеною матрицею системи, яка виходить шляхом додавання до основної матриці A стовпця вільних членів b .

Перший етап рішення системи рівнянь, званий прямим ходом методу Гауса, полягає в приведенні розширеної матриці (3) до трикутного вигляду. Це означає, що усі елементи матриці (2) нижче за головну діагональ мають дорівнювати нулю.

$$A' = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{pmatrix} \quad (3)$$

Для формування першого стовпця матриці (4) необхідно з кожного рядка (починаючи з другого) відняти перший, помножений на деяке число M .

$$A' = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & b_1 \\ 0 & a_{22} & a_{23} & \dots & a_{2n} & b_2 \\ 0 & 0 & a_{33} & \dots & a_{3n} & b_3 \\ 0 & 0 & 0 & \dots & a_{4n} & b_4 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_{nn} & b_n \end{pmatrix} \quad (4)$$

Оскільки метою цих перетворень являється обнулення першого елементу рядка, то M вибирається з умови:

$$a_{11} = a_{21} - Ma_{11}$$

$$M = \frac{a_{21}}{a_{11}}$$

Таким чином перетворення i -го рядка відбуватиметься таким чином:

$$a_{i1} = a_{i1} - Ma_{11}$$

$$a_{i2} = a_{i2} - Ma_{12} \dots a_{ii} = a_{ii} - Ma_{1i} \dots a_{in} = a_{in} - Ma_{1n}$$

$$b_i = b_i - Mb_1$$

Коефіцієнт M для i -го рядка вибирається з умови:

$$a_{i1} - Ma_{11} = 0$$

та дорівнює

$$M = \frac{a_{i1}}{a_{11}}$$

Після проведення подібних перетворень для усіх рядків матриця (3) прийме вид:

$$A' = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ 0 & a_{22} & \dots & a_{2n} & b_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & a_{n2} & \dots & a_{nn} & b_n \end{pmatrix}$$

Очевидно, якщо повторити описаний вище алгоритм для наступних стовпців матриці (3), причому починати перетворювати другий стовпець з третього елементу, третій стовпець – з четвертого і так далі, то в результаті буде отримана матриця (4).

Зазначимо, якщо в матриці (3) на головній діагоналі зустрівся елемент a_{kk} , рівний нулю, то розрахунок коефіцієнта

$$M = \frac{a_{ik}}{a_{kk}}$$

для k -го рядка буде неможливий. Уникнути ділення на нуль можна, позбавившись від нульових елементів на головній діагоналі. Для цього перед обнуленням елементів в k -му стовпці необхідно знайти в ньому максимальний по модулю елемент, запам'ятати номер рядка, в якому він знаходиться, і поміняти її місцями з k -м.

В результаті виконання прямого методу Гауса матриця (3) перетвориться в матрицю (4), а система рівнянь (2) матиме наступний вигляд:

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n = b_1 \\ \quad a_{22}x_2 + a_{23}x_3 + \dots + a_{2n}x_n = b_2 \\ \quad \quad a_{33}x_3 + \dots + a_{3n}x_n = b_3 \\ \quad \quad \quad \dots \quad \quad \dots \\ \quad \quad \quad \quad a_{nn}x_n = b_n \end{array} \right. \quad (5)$$

Рішення системи (5) називають зворотним ходом методу Гауса.

Останнє рівняння системи (5) має вигляд:

$$a_{nn}x_{nn} = b_n.$$

Тоді, якщо $a_{nn} \neq 0$, то

$$x_n = \frac{b_n}{a_{nn}}.$$

У випадку, якщо $a_{nn} = 0$ і $b_n = 0$,

то система (5), а отже і система (2) має нескінченну безліч рішень.

При $a_{nn} = 0$ і $b_n \neq 0$

система (5), а отже і система (2) рішення не має.

Передостаннє $(n - 1)$ -е рівняння системи (5) має вигляд:

$$a_{n-1n-1}x_{n-1} + a_{n-1n}x_n = b_{n-1}.$$

Тобто

$$x_{n-1} = \frac{b_{n-1} - a_{n-1n}x_n}{a_{n-1n-1}}$$

Таким чином, формула для обчислення i -го значення x матиме вигляд:

$$x_i = \frac{b_i - \sum_{j=i+1}^n a_{ij}x_j}{a_{ii}}$$

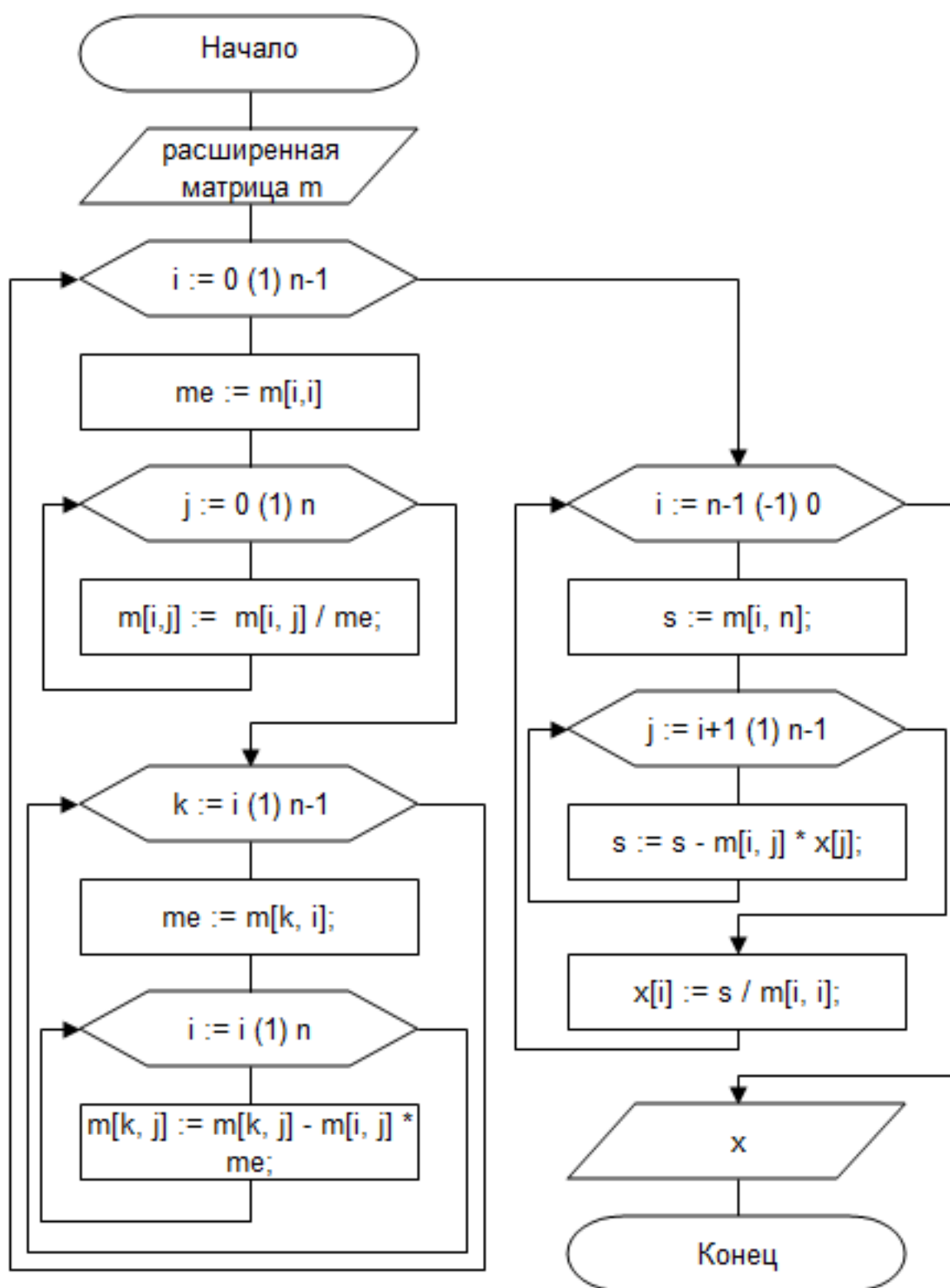


Рисунок 2.1. Блок-схема алгоритму метода Гауса

2.2.3. Розв'язання систем алгебраїчних лінійних рівнянь методом Зейделя

Для вирішення **методом Зейделя** система лінійних рівнянь алгебри $Ax = b$ має бути приведена до виду $x = Gx + f$, де G - деяка матриця, f - перетворений вектор вільних членів. Потім вибирається початкове наближення - довільний вектор $x^{(0)}$ - і будується рекурентна послідовність векторів $x^{(1)}$, $x^{(2)}$, ..., $x^{(k)}$, ... по формулі

$$x^{(k)} = Gx^{(k-1)} + f.$$

Для збіжності цієї послідовності при будь-якому початковому наближенні *необхідно і достатньо*, щоб усі власні значення матриці G були за абсолютною величиною менше одиниці. На практиці це важко перевірити, і зазвичай користуються *достатніми умовами збіжності* - ітерації сходяться, якщо яка-небудь норма матриці менше одиниці, тобто

$$\|G\|_1 = \max_{1 \leq i \leq n} \sum_{j=1}^n |g_{ij}| = \alpha < 1 \text{ или } \|G\|_2 = \max_{1 \leq j \leq n} \sum_{i=1}^n |g_{ij}| = \beta < 1.$$

Чим менше норма матриці G , тим швидше сходиться ітераційний процес.

Перетворення системи можна здійснити, просто вирішуючи кожне i -е рівняння відносно x_i :

$$x_i = -\frac{1}{a_{ii}} \left[\sum_{j=1, j \neq i}^n a_{ij} x_j - b_i \right].$$

Метод Зейделя використовує наступний алгоритм побудови наближень:

$$x_i^{(k)} = -\frac{1}{a_{ii}} \left[\sum_{j=1}^{i-1} a_{ij} x_j^{(k)} + \sum_{j=i+1}^n a_{ij} x_j^{(k-1)} - b_i \right]$$

Якщо A - матриця з домінуючою діагоналлю, т.е. $|a_{ii}| > \sum_{j \neq i}^n |a_{ij}|$,

то метод Зейделя сходиться при будь-якому початковому наближенні $x^{(0)}$.

$$x_i^k = \omega x_i^k + (1 - \omega)x_i^{k-1}$$

Метод Зейделя сходиться приблизно так само, як геометрична прогресія зі знаменником $\|G\|$. Якщо норма матриці G близька до 1, то швидкість збіжності дуже повільна.

2.3. Сучасні методи розв'язання СЛАР

2.3.1. Метод релаксації

Для прискорення збіжності метода Зейделя використовується метод *релаксації*. Суть його в тому, що отримане по методу Зейделя чергове значення перераховується по формулі :

$$x_i^{n+1} + \omega \sum_{j < i} \frac{a_{ij}}{a_{ii}} x_j^{n+1} = (1 - \omega)x_i^n - \omega \sum_{j > i} \frac{a_{ij}}{a_{ii}} x_j^n + \omega \frac{d_i}{a_{ii}}, \quad i = 1, \dots, m$$

Тут ω - параметр релаксації. Якщо $\omega < 1$ - *нижня релаксація*, якщо $\omega > 1$ - *верхня релаксація*. Параметр (підбирають так, щоб збіжність методу досягалася за мінімальне число ітерацій).

Якщо матриця - симетрична позитивно певна матриця, тоді метод верхньої релаксації сходиться при $0 < \omega < 2$.

Припинення обчислень зумовлюється наступним:

$$\max_{1 \leq i \leq m} |x_i^{n+1} - x_i^n| < \varepsilon,$$

де $\varepsilon > 0$ – наперед задане число.

Цей метод є *однокроковим ітераційним методом*, коли для знаходження $x^{(k+1)}$ вимагається пам'ятати тільки одну попередню ітерацію $x^{(k)}$.

Погрішність ітерації обчислюється за формулою:

$$\delta = \max_{1 \leq i \leq n} |x_i^k - x_i^{k-1}|$$

де n - порядок матриці A .

Якщо δ менше заданої точності ε , то ітераційний процес припиняють.

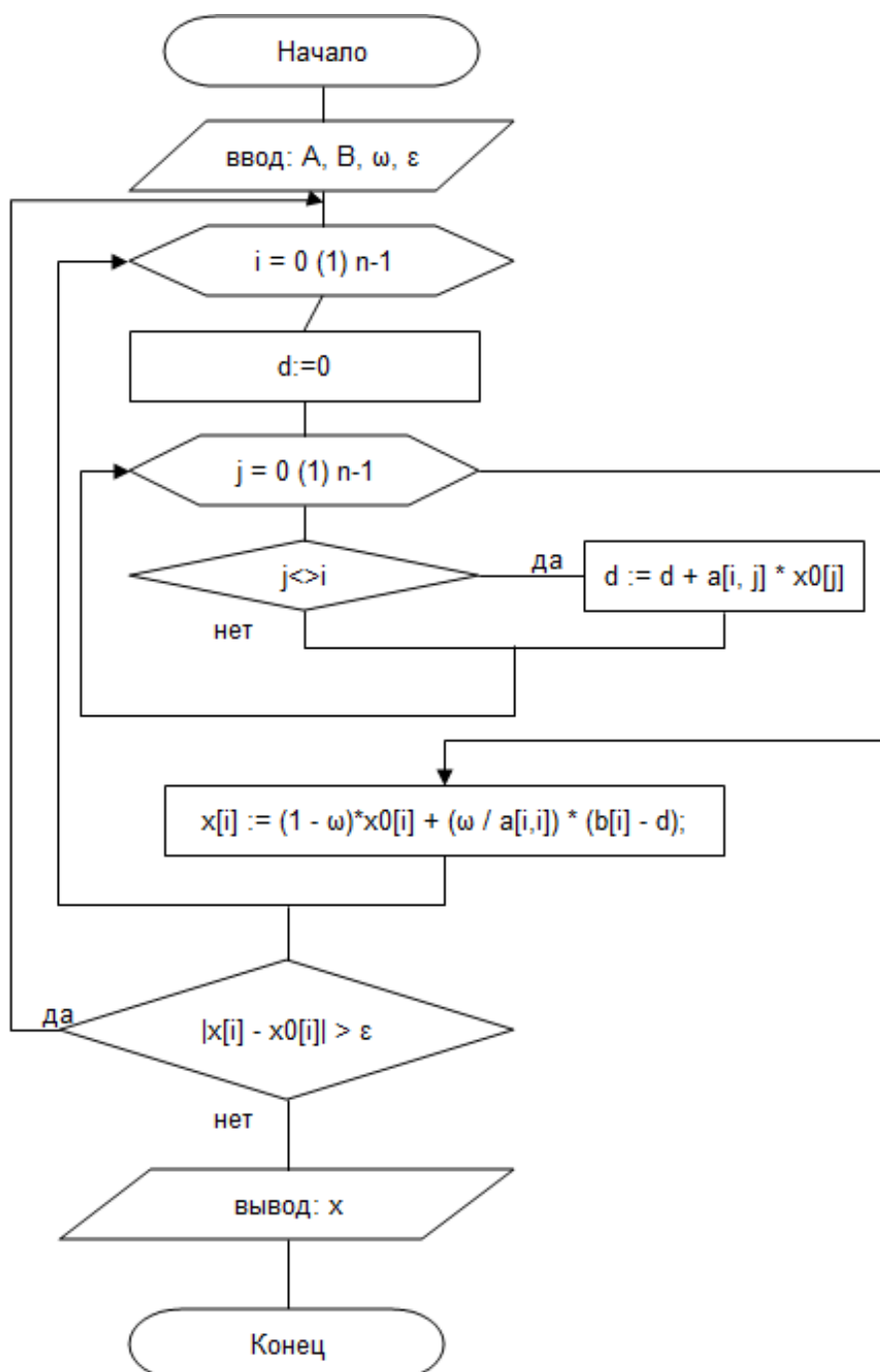


Рисунок 2.2. Блок-схема алгоритму верхньої релаксації

Елементи головної діагоналі називаються головними. Слід зазначити, що якщо в ході розрахунків по цьому алгоритму на головній діагоналі виявиться

нульовий елемент, то станеться збій програми. Для того, щоб уникнути цього, слідує перестановку рядків так, щоб на головній діагоналі знаходилися максимальні елементи рядків. Т. е., якщо в k -у рядку максимальним є i -й елемент, необхідно поміняти місцями k -ий та i -ий рядки, і поміняти місцями відповідні елементи вектору b . Такий вибір головного елемента потрібний для збіжності ітераційного процесу.

Перевагою ітераційного методу верхніх релаксацій є те, що при його реалізації програмним шляхом алгоритм обчислень має простий вигляд і дозволяє використовувати всього один масив для невідомого вектору.

2.3.2. Метод LU-розкладу

Алгоритми цього методу досить близькі до методу Гауса. Основна перевага методу LU – факторизації в порівнянні з методом Гауса є можливість більш простого одержання розв’язку для різних векторів b системи лінійних алгебраїчних рівнянь

$$A \cdot x = b \quad (6)$$

A – матриця розміру $n \times n$ із сталими коефіцієнтами;

b – n -мірний вектор вільних членів;

x – n -мірний вектор невідомих.

Матриця системи A представляється у вигляді добутку двох трикутних матриць L та U

$A = L \cdot U$, де

$$L = \begin{bmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & \dots & 0 \\ l_{31} & l_{32} & l_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{bmatrix}; \quad U = \begin{bmatrix} 1 & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & 1 & u_{23} & \dots & u_{2n} \\ 0 & 0 & 1 & \dots & u_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix},$$

L – нижня трикутна матриця, U – верхня трикутна матриця.

Зауважимо, що на головній діагоналі матриці U стоять одиниці. Це в свою чергу означає, що визначник матриці A дорівнює добутку діагональних елементів l_{ii} матриці L . ($\det U = 1$; $\det L = l_{11}l_{22}\dots l_{nn}$; $\det A = \det L$).

Отже, систему (6) можна записати у вигляді

$$L \cdot U \cdot x = b. \quad (7)$$

Введемо допоміжний вектор y як

$$U \cdot x = y \quad (8)$$

Тоді можна записати, що

$$L \cdot y = b. \quad (9)$$

Розв'язування системи (6) або (7) відбувається в два етапи: спочатку розв'язується система $L \cdot y = b$, а потім $U \cdot x = y$.

Така послідовність розв'язку дає перевагу методу (в порівнянні з методом Гауса) – якщо потрібно розв'язати декілька систем з однією і тією ж матрицею A , задача суттєво спрощується, оскільки зберігаються матриці L та U .

Розв'язок системи $L \cdot y = b$ називається прямим ходом, а системи $U \cdot x = y$ – оберненим ходом.

Розглянемо прямий хід. Завдяки спеціальній формі матриці L вектор y можна легко визначити. Для цього (9) перепишемо у вигляді системи рівнянь

$$\begin{cases} l_{11}y_1 & = b_1 \\ l_{21}y_1 + l_{22}y_2 & = b_2 \\ l_{31}y_1 + l_{32}y_2 + l_{33}y_3 & = b_3 \\ \dots & \dots \\ l_{n1}y_1 + l_{n2}y_2 + l_{n3}y_3 + \dots + l_{nn}y_n & = b_n \end{cases}$$

Звідси можна одержати, що в загальному вигляді

$$y_1 = b_1 / l_{11} \quad (10)$$

$$y_i = (b_i - \sum_{m=1}^{i-1} l_{im}y_m) / l_{ii} \quad \text{для } i = \overline{2, n}.$$

При оберненому ході вектор x визначається з системи рівнянь (8)

$$x_1 + u_{12}x_2 + u_{13}x_3 + \dots + u_{1n}x_n = y_1$$

$$x_2 + u_{23}x_3 + \dots + u_{2n}x_n = y_2$$

$$x_3 + \dots + u_{3n}x_n = y_3$$

.....

$$x_n = y_n$$

Починаючи з останнього рівняння, можна послідовно знайти компоненти вектора x . В загальному вигляді вони визначаються за формулами

$$x_n = y_n$$

$$x_i = y_i - \sum_{m=i+1}^n u_{im} x_m \quad \text{для } i = \overline{n-1, 1}. \quad (11)$$

Тепер розглянемо LU – розклад матриці A .

Елементи матриці L та U визначаються за наступними формулами:

$$l_{i1} = a_{i1}; \quad \text{де } (i = \overline{1, n}) \quad l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}, \quad \text{де } (j = \overline{2, i})$$

$$u_{ii} = 1; \quad u_{1j} = \frac{a_{1j}}{a_{11}}, \quad \text{де } (j = \overline{2, n})$$

$$u_{ij} = \frac{1}{l_{ii}} (a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}), \quad \text{де } (i = \overline{2, j}).$$

Ці ж самі формули можна записати у більш зручному вигляді (метод Краута).

У варіанті методу, що називається методом Краута, використовується наступна послідовність знаходження елементів матриць L та U ($k = \overline{1, n}$) – де k – крок.

Для всіх $k = \overline{1, n}$

$$l_{ik} = a_{ik} - \sum_{m=1}^{k-1} l_{im} u_{mk}, \quad \text{де } i = \overline{k, n} \quad (12)$$

$$u_{kj} = (a_{kj} - \sum_{m=1}^{k-1} l_{km} u_{mj}) / l_{kk}, \quad \text{де } j = \overline{k+1, n} \quad (13)$$

Домовимось, як звичайно, вважати значення суми рівним нулю, якщо верхня границя (межа) сумування менша від нижньої.

Крім того, $u_{kk} = 1$.

Тобто, при $k=1$

$$l_{i1} = a_{i1}, \quad \text{для всіх } i = \overline{1, n}$$

$$u_{1j} = \frac{a_{1j}}{a_{11}}, \quad \text{для всіх } j = \overline{2, n}.$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

На місце матриці A записується матриця такого вигляду:

$$\begin{bmatrix} l_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ l_{21} & l_{22} & u_{23} & \dots & u_{2n} \\ l_{31} & l_{32} & l_{33} & \dots & u_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{bmatrix}$$

Алгоритм

1) $k=1$

Обчислюємо

$$l_{i1} = a_{i1}, \quad \text{для всіх } i = \overline{1, n}$$

$$u_{1j} = \frac{a_{1j}}{a_{11}}, \quad \text{для всіх } j = \overline{2, n}.$$

2) $k = k + 1$

$$l_{ik} = a_{ik} - \sum_{m=1}^{k-1} l_{im} u_{mk}, \quad \text{де } i = \overline{k, n}$$

$$u_{kj} = (a_{kj} - \sum_{m=1}^{k-1} l_{km} u_{mj}) / l_{kk}, \quad \text{де } j = \overline{k+1, n}.$$

Продовжуємо до тих пір, доки $k = n$.

Необхідні в цих співвідношеннях значення елементів матриць L та U обчислюються на попередніх етапах процесу.

Кожний елемент a_{ij} матриці A потрібен тільки для обчислення відповідних елементів матриці L та U (тобто, в подальшому процесі a_{ij} не потрібні). Оскільки нульові елементи матриць L та U , а також одиничну діагональ матриці U запам'ятовувати не потрібно, тобто в процесі обчислення матриці L та U можуть бути записані на місце матриці A . Причому матриця L розташована в нижньому трикутнику ($i \geq j$), U – відповідно в верхньому трикутнику ($i < j$) матриці A .

Розклад матриці A на матриці L та U , як правило, об'єднують з прямим ходом. Може бути використана наступна послідовність обчислень: спочатку обчислюється перший стовпець матриці L , потім перший рядок матриці U та перший елемент вектора u ; далі – другий стовпець матриці L , другий рядок матриці U та другий елемент вектора u і т.д.

Матрицю A можна розкласти на дві трикутні матриці L та U при умові, що головні діагональні мінори матриці A відмінні від нуля. Якщо ця умова не виконується, наприклад, для k -го головного діагонального мінора, то при обчисленні елемента l_{kk} матриці L за формулою (7) він стане рівним нулю. Це в свою чергу призведе до неможливості обчислення елементів u_{kj} матриці U

(ділення на нуль). Усунути таку ситуацію можна було б шляхом перевірки умов нерівності нуля головних діагональних мінорів до розкладу $A = L U$. Однак така перевірка пов'язана із значними затратами машинного часу, що перевищує затрати часу пов'язані із розкладом $A = L U$ (крім того перевірка – це обчислення тих самих визначників).

Тому доцільніше проводити перевірку умови $l_{kk} = 0$ безпосередньо в процесі обчислення елементів l_{ij} та u_{ij} . Тоді при виконанні цієї умови потрібно переставляти відповідний k -й рядок матриці A з наступними $k + s$ рядками ($s = \overline{1, n - k}$), до тих пір, доки не виконається умова $l_{k+s, k+s} \neq 0$.

Висновки до розділу 2

У розділі розглянуто прямі та ітераційні методи розв'язання СЛАР такі як метод Гаусса, метод Зейделя, метод релаксації та метод LU-розкладу.

У прямих (чи точних) методах рішення системи отримують за кінцеве число арифметичних дій, кількість яких залежить від кількості рівнянь та невідомих у системі. На точність цих методів впливає полки округлення ЕОМ під час виконання арифметичних операцій.

Ітераційні методи є нескінченними і знаходять рішення системи як послідовність наближень $x(k)$, де k - номер ітерації, поки не буде досягнута задана точність.

РОЗДІЛ 3. РОЗРОБКА ДОДАТКУ ДЛЯ ВИРІШЕННЯ

СЛАР

3.1. Опис середовища розробки Delphi

Серед великої різноманітності продуктів для розробки застосувань Delphi займає одно з провідних місць. Delphi віддають перевагу розробники з різним стажем, звичками, професійними інтересами. За допомогою Delphi написана колосальна кількість застосувань, десятки фірм і тисячі програмістів-одиначок розробляють для Delphi додаткові компоненти.[27].

У основі такої загальноновизнаної популярності лежить той факт, що Delphi, як ніяка інша система програмування, задовольняє викладеним вище вимогам. Дійсно, застосування за допомогою Delphi розробляються швидко, причому взаємодія розробника з інтерактивним середовищем Delphi не викликає внутрішнього відторгнення, а навпаки, залишає відчуття комфорту. Delphi - додатки ефективні, якщо розробник дотримує певні правила (та досить часто - навіть якщо не дотримує). Ці застосування надійні і при експлуатації мають передбачувану поведінку.[27].

Пакет Delphi - продовження лінії компіляторів мови Pascal корпорації Borland. Pascal як мова дуже проста, а строгий контроль типів даних сприяє ранньому виявленню помилок і дозволяє швидко створювати надійні і ефективні програми. Корпорація Borland постійно збагачувала мову. Колись у версію 4.0 були включені засоби роздільної трансляції, пізніше, починаючи з версії 5.5, з'явилися об'єкти, а до складу шостої версії пакету увійшла повноцінна бібліотека класів Turbo Vision, що реалізовує віконну систему в текстовому режимі роботи відеоадаптера. Це був один з перших продуктів, що містили інтегроване середовище розробки програм.

У класі інструментальних засобів для початкуючих програмістів продуктам компанії Borland довелося конкурувати з середовищем Visual Basic

корпорації Microsoft, де питання інтеграції і зручності роботи були вирішені краще. Коли на початку 70-х років Н. Вірт опублікував повідомлення про Pascal, це був компактний, з невеликою кількістю основних понять і зарезервованих слів мова програмування, націлена на навчання студентів. Мова, на якій належить працювати користувачеві Delphi, відрізняється від початкового не лише наявністю безлічі нових понять і конструкцій, але і ідейно: в ній замість мінімізації числа понять і використання найпростіших конструкцій (що, безумовно, добре для навчання, але не завжди виправдано в практичній роботі), перевага віддається зручності роботи професійного користувача. Як мову Turbo Pascal природно порівнювати з його найближчими конкурентами - численними варіаціями на тему мови Basic (в першу чергу з Visual Basic корпорації Microsoft) і з C++.[27, 28]. Я вважаю, що Turbo Pascal істотно перевершує Basic за рахунок повноцінного об'єктного підходу, що включає розвинені механізми *інкапсуляції, спадкоємства і поліморфізму*. Остання версія мови, вживана в Delphi, по своїх можливостях наближається до C++. З основних механізмів, властивих C++, відсутнє тільки множинне спадкоємство. (Втім, цим красивим і потужним механізмом породження нових класів користується лише невелика частина програмістів, що пишуть на C++.) Плюси застосування мови Pascal очевидні: з одного боку, на відміну від Visual Basic, заснованого на інтерпретації проміжного коду, для нього є компілятор, що генерує машинний код, що дозволяє отримувати значно швидші програми. З іншої - на відміну від C++ синтаксис мови Pascal сприяє побудові дуже швидких компіляторів. [28].

Середовище програмування нагадує пакет Visual Basic. У вашому розпорядженні декілька окремих вікон: меню і інструментальні панелі, Object Inspector (у якому можна бачити властивості об'єкту і пов'язані з ним події), вікна візуального будівника інтерфейсів (Visual User Interface Builder), Object Browser (що дозволяє вивчати ієрархію класів і переглядати списки їх полів, методів і властивостей), вікна управління проектом (Project Manager) і редактора.

Delphi містить повноцінний текстовий редактор типу Brief, призначення клавіш в якому відповідають прийнятим в Windows стандартам, а глибина ієрархії операцій Undo неограниченна. Як це стало вже обов'язковим, реалізовано колірне виділення різних лексичних елементів програми. Процес побудови застосування досить простий. Треба вибрати форму (у поняття форми входять звичайні, діалогові, батьківські і дочірні вікна MDI), задати її властивості і включити в неї необхідні компоненти (видимі і, якщо знадобиться, що не відображаються) : меню, інструментальні панелі, рядок стану і т. п., задати їх властивості і далі написати (за допомогою редактора початкового коду) обробники подій. Вікна типу Object Browser стали невід'ємною частиною систем програмування на об'єктно-орієнтованих мовах. Робота з ними стає можливою відразу після того, як ви скомпілювали застосування.

Project Manager - це окреме вікно, де перераховуються модулі і форми, що становлять проект. При кожному модулі вказується маршрут до каталогу, в якому знаходиться початковий текст. Жирним шрифтом виділяються змінені, але ще не збережені частини проекту. У верхній частині вікна є набір кнопок : додати, видалити, показати початковий текст, показати форму, задати опції і синхронізувати вміст вікна з текстом файлу проекту, т. е. з головною програмою на мові Pascal.

Visual Component Library (VCL) багатство палітри об'єктів для побудови призначеного для користувача інтерфейсу - один з ключових чинників при виборі інструменту візуального програмування. При цьому для користувача має значення як число елементів, включених безпосередньо в середу, так і доступність елементів відповідного формату на ринку. [27].

3.2. Опис модулів та класів додатку

Модуль **Matrix.pas** містить опис та реалізацію класу **TMatrix**. Цей клас реалізує структуру даних для збереження матриці в пам'яті та методи маніпуляції з ними.

Для реалізації матриці використано динамічний масив, які, на відміну від статичних, не мають ніякої передрозподіленої пам'яті. Розміри таких масивів встановлюються безпосередньо перед тим, як вони використовуватимуться. Наприклад:

```
SetLength(dynArray, 5);
```

встановлює розмір одновимірному масиву `dynArray` в 5 елементів. При цьому буде розподілена пам'ять.

Усі динамічні масиви починаються з індексу = 0.

Для збереження елементів матриці використовується приватне поле `FElements`, яке визначається наступним чином:

```
FElements: array of array of Double;
```

Розмір цього масиву ініціюється за допомогою функції `SetLength` в конструкторі, яких є два: для створення квадратної матриці, та з різними розмірностями.

```
constructor TMatrix.Create(N: Word);
```

```
var
```

```
    I: Integer;
```

```
begin
```

```
    self.n := N;
```

```
    self.m := N;
```

```
    SetLength(FElements, self.n);
```

```
    for I := 0 to self.n - 1 do
```

```
        SetLength(FElements[i], self.m);
```

```
end;
```

Для доступу до елементів використовується публічна властивість:


```
property Elements[m, n: Word]: Double read
GetElement write SetElement; default;
```

Функції `GetElement` та `SetElement` повертають та встановлюють елемент з індексом m, n масиву `FElement`.

Модуль **Vector.pas** містить реалізацію класу **TVector**. За структурою та функціональністю подібний до описаного вище класу `TMatrix`, за винятком того, що слугує для зберігання одномірного вектору.

Модуль **SLESolvers.pas** містить функції, що реалізують відповідні методи розв'язання СЛАР. В інтерфейсній частині модуля оголошені прототипи двох функцій:

```
function Gauss(const a: TMatrix; const b: TVector): TVector;
function Seidel(const a: TMatrix; const b: TVector; const epsilon: double;
const maxiter: integer; var niter: integer): TVector;
```

Функція **Gauss** приймає на вхід матрицю коефіцієнтів a та вільних членів b . Результатом є вектор `TVector` із значеннями невідомих.

Для реалізації методу Гауса спочатку будується розширена матриця m

```
m:= TMatrix.Create(b.Capacity, b.Capacity + 1);
for i := 0 to b.Capacity - 1 do
begin
  for j := 0 to b.Capacity - 1 do
    m[i, j] := a[i, j];
  m[i, b.Capacity] := b[i];
end;
```

шляхом додавання до метриці коефіцієнтів стовпця вільних членів. Потім для кожного рядка виконується наступна послідовність дій: поточний i -й рядок ділимо на i -й елемент

```
me := m[i, i];
for j := 0 to b.Capacity do
  m[i, j] := m[i, j] / me;
```

Далі з наступних рядків віднімається поточний, помножений на перший елемент рядка, що оброблюється:

```
for k := i+1 to b.Capacity - 1 do
  begin
    me := m[k, i];
    for j := i to b.Capacity do
      m[k, j] := m[k, j] - m[i, j] * me;
    end;
```

На цьому прямий хід закінчується матриця *m* повинна прийняти трикутну форму.

Далі виконується зворотній хід для обчислення значень невідомих. Зберігатися вони будуть у векторі *res*.

```
res:= TVector.Create(b.Capacity);
```

Починаючи з останнього рядка обчислюються значення невідомих.

```
for i := b.Capacity - 1 downto 0 do
  begin
    s := m[i, b.Capacity];
    for j := i + 1 to b.Capacity - 1 do
      s := s - m[i, j] * res[j];
    res[i] := s / m[i, i];
  end;
```

Функція **Seidel** має визначення:

```
function Seidel(const a: TMatrix; const b: TVector;
const epsilon: double; const maxiter: integer; var niter:
integer): TVector;
```

Окрім матриці коефіцієнтів *a* та вектору вільних членів до функції передаються також необхідна точність обчислення *epsilon* та кількість повторень, після якої необхідно припинити виконання функції *maxiter*. Після

завершення виконання функції в змінній *niter* буде значення кількості ітерацій, за яку було знайдено рішення.

Для методу Зейделя необов'язково будувати розширену матрицю. Тому функція відразу розпочинається з циклу від 1 до *maxiter*.

```
for k := 1 to maxiter do
```

В якому по-перше зберігається номер поточної ітерації

```
niter := k;
```

Після обчислення нових значень коренів системи, якщо вони не задовольняють необхідній точності, це значення встановлюється у 0.

```
if abs(s) > epsilon then
```

```
    niter := 0;
```

Інакше значення залишається і використовуються для переривання основного циклу, якщо рішення було знайдено раніше, ніж за *maxiter* повторень.

```
if niter <> 0 then
```

```
    break;
```

На кожній ітерації обчислюється нові наближення коренів. Додатковий масив для збереження попередніх значень не потрібен, тому що для обчислення коренів використовуються щойно отримані значення невідомого.

```
s := b[i];
```

```
for j := 0 to b.Capacity - 1 do
```

```
    s := s - a[i, j] * res[j];
```

```
s := s / a[i, i];
```

```
res[i] := res[i] + s;
```

3.3. Інтерфейс головного вікна

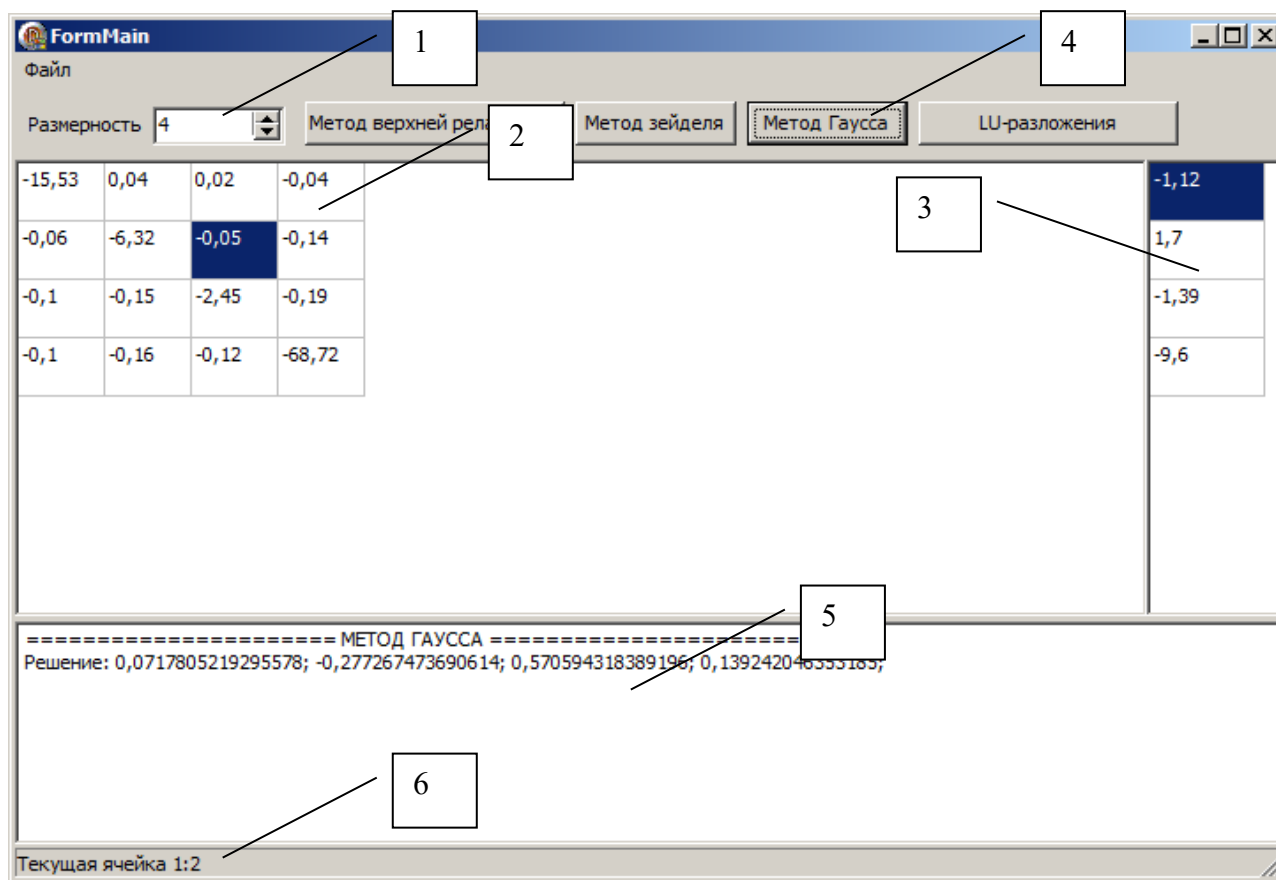


Рисунок 3.1. Головне вікно програми

Головне вікно програми містить наступні елементи:

1. Лічильник для встановлення розмірності системи рівнянь.
2. Таблиця для вводу коефіцієнтів при невідомих.
3. Таблиця для вводу вільних членів.
4. Кнопки для виконання необхідного методу.
5. Поле для виводу результатів.
6. Статус, що вказує, в якій комірці системи зараз знаходиться курсор.

Для реалізації графічного інтерфейсу користувача було використано бібліотеки VCL та її різноманітні компоненти (додаток Д).

Висновки до розділу 3

Мова програмування Object Pascal, що використовується в середовищі Delphi, має необхідні структури даних та мовні конструкції для ефективної реалізації розрахункових задач. Наявність строгої типізації дозволяє уникнути труднознаходжуваних помилок.

Бібліотека візуальних компонент VCL дозволяє швидко створити графічний інтерфейс користувача, що робить використання додатку зручним, а відображення результатів наочним.

Цими засобами Delphi було реалізовано алгоритми для наступних методів: Гауса та LU-розкладу (точні), Зейделя та методу релаксації (ітераційні). Створений додаток має графічний інтерфейс користувача, дозволяє завантажувати систему рівнянь з файлу.

ВИСНОВКИ

Серед найбільш важливих завдань лінійної алгебри має рішення системи лінійних алгебраїчних рівнянь. Чисельні методи розв'язання СЛАР бувають точними або ітераційними. Методи першої групи дозволяють отримати рішення за скінчену кількість кроків, в залежності від кількості рівнянь в системі. Ітераційні методи дозволяють отримати приблизне рішення із заданою точністю, але вони мають простішу реалізацію на ЕОМ.

В ході виконання дипломного проекту було створено додаток операційної системи Windows з графічним інтерфейсом користувача для вирішення систем лінійних алгебраїчних рівнянь прямими та ітераційними методами. Було реалізовано два досить простих методи: це метод Гауса і метод Зейделя, та їхні більш ефективні варіанти – LU-розклад і метод релаксації. Перевага цих методів в тому, що вони дозволяють без додаткових обчислень швидко знаходити розв'язання системи з різними значеннями вільних членів.

Ці методи було реалізовано у вигляді модуля Delphi, що може бути використаний підчас розробки інших додатків.

Для досягнення мети роботи виконано наступні завдання:

- Розглянуто дві групи чисельних методів для розв'язання СЛАР: прямі та ітераційні.
- Розроблено алгоритми для наступних методів: Гауса та LU-розкладу (точні), Зейделя та методу релаксації (ітераційні).
- Створено додаток, що реалізує розроблені алгоритми мовою Pascal в середовищі Delphi.

В дипломній роботі було розглянуто загальні положення з охорони праці для працівників, що використовують ЕОМ у своїй професійній діяльності та методи розрахунку зниження шуму, що відповідають санітарним нормам України. При дослідженні доведено, що наслідком шкідливої дії виробничого шуму можуть бути професійні захворювання, підвищення ступеня ризику травм та нещасних випадків, зниження продуктивності праці.

ЛІТЕРАТУРА

1. Валях Е. Последовательно-параллельные вычисления. / Е. Валях. – М.: Мир, 1985. – 456 с.
2. Калиткин Н.Н. Чистенные методы. / Н.Н. Калиткин, – М : Главная редакция физико-математической литературы изд-ва «Наука», 1978. –512 с.
3. Иванов В.В. Методы вычислений на ЭВМ: Справочное пособие / В.В. Иванов– Киев:Наук.думка, 1986. – 584 с.
4. Бахвалов Н. С. Численные методы : [учеб. пособие] / Н. С. Бахвалов, Н. П. Жидков, Г. М. Кобельков. - Изд. 3-е, доп. и перераб. - М. : БИНОМ. Лаб. знаний, 2003. - 632 с.
5. Волков Е.А. Численные методы / Е.А. Волков. – М.: Наука, – 1987, – 248 с.
6. Мудров А.Е. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль / А.Е. Мудров.– Томск: МП "РАСКО", 1991. - 272 с.
7. Самарский А.А. Численные методы / Самарский А.А., Гулин А.В. – М.: Наука, – 1989, – 432 с.
8. Буслов В.А. Численные методы II. Решение уравнений : [Курс лекций] / В.А. Буслов, С.Л. Яковлев. – Санкт-Петербург, 2001. – 45 с.
9. Ващенко Г.В. Вычислительная математика. Основы конечных методов решения систем линейных алгебраических уравнений / Г.В. Ващенко. – Красноярск: СибГТУ, 2005. – 75 с
10. Вержбицкий В.М. Численные методы. Линейная алгебра и нелинейные уравнения / В.М. Вержбицкий. – М: Высшая школа, 2000. – 274 с.
11. Воеводин В.В. Матрицы и вычисления / Воеводин В.В., Кузнецов Ю.А. – М: Наука, 1984. – 320 с.

12. Лоусон Ч. Численное решение задач метода наименьших квадратов / Ч. Лоусон . – М.: Наука, 1986. – 232 с
13. А.М. Островский А.М. Решение уравнений и систем уравнений / А.М. Островский. – М: Издательство иностранной литературы, 1963. – 220 с
14. Потабенко Н.А. Численные методы. Решение задач линейной алгебры и уравнений в частных производных / Потабенко Н.А. – М:Издательство МАИ, 1997. – 88 с.
15. Ракитин В.И. Практическое руководство по методам вычислений с приложением программ для персональных компьютеров / В.И.Ракитин, В.Е.Первушин .– М:Высшая школа, 1998. – 383 с.
- 16.Фаддеев Д.К. Вычислительные методы линейной алгебры / Д.К. Фаддеев, В.Н. Фаддеева. – М.: «Наука», 1963.- 656 с.
17. Кац И. С. Применение метода верхней релаксации к решению трехмерных задач для уравнения второго порядка / И. С. Кац // Вычислительная математика и математическая физика. – 1968. – №8:1
18. Разработка приложений в Delphi. Стандартные визуальные компоненты [Электронный ресурс]. – Режим доступа: <http://www.program.lukiseller.ru/startdelphi.html>
19. Сухарев М.В. Основы Delphi / Сухарев М.В. Профессиональный поход: Наука и техника, 2004.-420 с.
20. Фленов М.В. Программирование в Delphi глазами хакера / Фленов М.В. – BHV-Санкт-Петербург, 2004.-330 с.
21. Понамарев В. Самоучитель Delphi 7 Studio / Вячеслав Понамарев — Санкт-Петербург: БХВ-Петербург, 2003 г.- 504 с.
- 22.Осипов Д. Delphi. Профессиональное программирование / Дмитрий Осипов — Москва: Символ-Плюс, 2006 г.- 1056 с.

23. Климова Л. М. Delphi 7. Основы программирования. Решение типовых задач. Самоучитель / Л. М. Климова — Москва: КУДИЦ-Образ, 2006 г.- 480 с.
24. Культин Н. Б. Delphi в задачах и примерах (+ CD-ROM) / Н. Б. Культин — Москва: БХВ-Петербург, 2008 г.- 288 с.
25. Архангельский А. Я. Приемы программирования в Delphi на основе VCL / А. Я. Архангельский — Москва: Бином-Пресс, 2009 г.- 944 с.
26. Чеснокова О. В. Delphi 2007. Алгоритмы и программы / О. В. Чеснокова — Москва: НТ Пресс, 2008 г.- 368 с.
27. Шумаков П. В. Delphi 3.0 и создание баз данных / П. В. Шумаков. — Москва, 1997. — 538 с.
28. Епанешников В.А. Программирование в среде Delphi 2.0 / А.М.Епанешников, В.А.Епанешников. — М.: Диалог-Мифи, 1997г.- 235с.

РОЗДІЛ 4. ДОДАТОК А. ЛІСТИНГ МОДУЛЯ MATRIX.PAS

```

unit Matrix;

interface

type
  TMatrix = class
  private
    FElements: array of array of Double;
    m: Word; //кол-во строк
    n: Word; //кол-во столбцов

    procedure SetElement(m, n: Word; Value: Double);
    function GetElement(m, n: Word): Double;
  public
    property Elements[m, n: Word]: Double read GetElement write SetElement; default;
    constructor Create(N: Word); overload;
    constructor Create(M, N: Word); overload;

  end;

implementation

{ TMatrix }

constructor TMatrix.Create(N: Word);
var
  I: Integer;
begin
  self.n := N;
  self.m := N;

  SetLength(FElements, self.n);
  for I := 0 to self.n - 1 do
    SetLength(FElements[I], self.m);
  end;

constructor TMatrix.Create(M, N: Word);
var
  i: Integer;
begin
  Self.m := M;
  Self.n := N;

  SetLength(FElements, self.m);
  for I := 0 to self.m - 1 do
    SetLength(FElements[I], self.n);
  end;

function TMatrix.GetElement(m, n: Word): Double;
begin

```

```
    Result := FElements[m, n];  
end;
```

```
procedure TMatrix.SetElement(m, n: Word; Value: Double);  
begin  
    FElements[m, n] := Value;  
end;  
  
end.
```

ДОДАТОК Б. ЛІСТИНГ МОДУЛЯ VECTOR.PAS

```

unit Vector;

interface
type
  TVector = class
  private
    FElements: array of Double;
    n: Word; //КОЛ-ВО СТОЛБЦОВ

    procedure SetElement(n: Word; Value: Double);
    function GetElement(n: Word): Double;
  public
    property Elements[n: Word]: Double read GetElement write SetElement; default;
    property Capacity: Word read n;
    constructor Create(N: Word);

  end;

implementation

constructor TVector.Create(N: Word);
var
  I: Integer;
begin
  self.n := N;

  SetLength(FElements, self.n);
end;

function TVector.GetElement(n: Word): Double;
begin
  Result := FElements[n];
end;

procedure TVector.SetElement(n: Word; Value: Double);
begin
  FElements[n] := Value;
end;

end.

```

ДОДАТОК В. ЛІСТИНГ МОДУЛЯ SLESOLVERS.PAS

```

unit SLESolvers;

interface
uses Matrix, Vector;

function Gauss(const a: TMatrix; const b: TVector): TVector;
function Seidel(const a: TMatrix; const b: TVector; const epsilon: double; const maxiter: integer;
var niter: integer): TVector;

implementation

function Gauss(const a: TMatrix; const b: TVector): TVector;
var
  res: TVector;
  m: TMatrix;
  i: Integer;
  j: Integer;
  me: Double; //Ведущий элемент
  s: Double;
  k: Integer;
begin
  m:= TMatrix.Create(b.Capacity, b.Capacity + 1);
  for i := 0 to b.Capacity - 1 do
  begin
    for j := 0 to b.Capacity - 1 do
      m[i, j] := a[i, j];
    m[i, b.Capacity] := b[i];
  end;

  for i := 0 to b.Capacity - 1 do
  begin
    me := m[i,i];
    for j := 0 to b.Capacity do
      m[i,j] := m[i, j] / me;

    for k := i+1 to b.Capacity - 1 do
    begin
      me := m[k, i];
      for j := i to b.Capacity do
        m[k, j] := m[k, j] - m[i, j] * me;
      end;
    end;

    res := TVector.Create(b.Capacity);

    for i := b.Capacity - 1 downto 0 do
    begin
      s := m[i, b.Capacity];
      for j := i + 1 to b.Capacity - 1 do
        s := s - m[i, j] * res[j];

```

```

    res[i] := s / m[i, i];
end;

```

```

    Result:= res;
end;

```

```

function Seidel(const a: TMatrix; const b: TVector; const epsilon: double; const maxiter: integer;
var niter: integer): TVector;

```

```

var

```

```

    res: TVector;
    m: TMatrix;
    i, j, k: integer;
    s: Double;

```

```

begin

```

```

    //m := TMatrix.Create(b.Capacity, b.Capacity + 1);

```

```

    res := TVector.Create(b.Capacity);

```

```

    // for i := 0 to b.Capacity - 1 do

```

```

    // begin

```

```

    //   for j := 0 to b.Capacity - 1 do

```

```

    //     m[i, j] := a[i, j];

```

```

    //   m[i, b.Capacity] := b[i];

```

```

    // end;

```

```

    for k := 1 to maxiter do

```

```

    begin

```

```

        niter := k;

```

```

        for i := 0 to b.Capacity - 1 do

```

```

        begin

```

```

            s := b[i];

```

```

            for j := 0 to b.Capacity - 1 do

```

```

                s := s - a[i, j] * res[j];

```

```

            s := s / a[i, i];

```

```

            res[i] := res[i] + s;

```

```

            if abs(s) > epsilon then

```

```

                niter := 0;

```

```

            end;

```

```

            if niter <> 0 then

```

```

                break;

```

```

        end;

```

```

    Result := res;

```

```

end;

```

```

end.

```

ДОДАТОК Г. ЛІСТИНГ МОДУЛЯ UNITMAIN.PAS

```

unit UnitMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Spin, Grids;

type
  TFormMain = class(TForm)
    GridMatrix: TStringGrid;
    SpinEditRank: TSpinEdit;
    Label1: TLabel;
    GridB: TStringGrid;
    MemoOut: TMemo;
    ButtonSolve: TButton;
    Button1: TButton;
    procedure SpinEditRankChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ButtonSolveClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
    procedure SetGridSize(rank: Word);
  public
    { Public declarations }
  end;

var
  FormMain: TFormMain;

implementation

{$R *.dfm}
uses StrUtils, Matrix, Vector, SLESolvers;

procedure TFormMain.Button1Click(Sender: TObject);
var
  m: TMatrix;
  v: TVector;
  rank: Word;
  i: Integer;
  j: Integer;
  s: string;
  x: TVector;
  niter: Integer;

begin
  rank := SpinEditRank.Value;

```

```

m := TMatrix.Create(rank);
v := TVector.Create(rank);

for i := 0 to rank-1 do
  for j := 0 to rank - 1 do
    m[i,j] := StrToFloat(GridMatrix.Cells[j,i]);

for i := 0 to rank - 1 do
  v[i] := StrToFloat(GridB.Cells[0, i]);

x := Seidel(m, v, 0.0001, 1000, niter);
MemoOut.Lines.Add(IntToStr(niter));
if niter > 0 then
begin
  s := "";
  for i := 0 to x.Capacity - 1 do
    s := s + FloatToStr(x[i]) + ' ';
  MemoOut.Lines.Add(s);
end;

end;

procedure TFormMain.ButtonSolveClick(Sender: TObject);
var
  m: TMatrix;
  v: TVector;
  rank: Word;
  i: Integer;
  j: Integer;
  s: string;
  x: TVector;

begin
  rank := SpinEditRank.Value;

  m := TMatrix.Create(rank);
  v := TVector.Create(rank);

  for i := 0 to rank-1 do
    for j := 0 to rank - 1 do
      m[i,j] := StrToFloat(GridMatrix.Cells[j,i]);

  for i := 0 to rank - 1 do
    v[i] := StrToFloat(GridB.Cells[0, i]);

  for i := 0 to rank-1 do
  begin
    s := "";
    for j := 0 to rank - 1 do
      s:= s+ FloatToStr(m[i,j]) + ' * X' + IntToStr(i+1) + IntToStr(j+1) + ' + ';
    s := LeftStr(s,Length(s)-2);

```



```

    s := s + ' ' + FloatToStr(v[i]);
    MemoOut.Lines.Add(s);
end;

x := Gauss(m, v);

s := "";
for i := 0 to x.Capacity - 1 do
    s := s + FloatToStr(x[i]) + ' ';
    MemoOut.Lines.Add(s);
end;

procedure TFormMain.FormCreate(Sender: TObject);
begin
    GridMatrix.Cells[0,0] := '0';
    GridB.Cells[0,0] := '0';
    SetGridSize(3);

    SpinEditRank.Value := 4;
    // SetGridSize(4);

    GridMatrix.Cells[0, 0] := '0,78';
    GridMatrix.Cells[1, 0] := '-0,02';
    GridMatrix.Cells[2, 0] := '-0,12';
    GridMatrix.Cells[3, 0] := '-0,14';
    gridB.Cells[0,0] := '0,76';

    GridMatrix.Cells[0, 1] := '-0,02';
    GridMatrix.Cells[1, 1] := '0,86';
    GridMatrix.Cells[2, 1] := '-0,04';
    GridMatrix.Cells[3, 1] := '0,06';
    gridB.Cells[0,1] := '0,08';

    GridMatrix.Cells[0, 2] := '-0,12';
    GridMatrix.Cells[1, 2] := '-0,04';
    GridMatrix.Cells[2, 2] := '0,72';
    GridMatrix.Cells[3, 2] := '-0,08';
    gridB.Cells[0,2] := '1,12';

    GridMatrix.Cells[0, 3] := '-0,14';
    GridMatrix.Cells[1, 3] := '0,06';
    GridMatrix.Cells[2, 3] := '-0,08';
    GridMatrix.Cells[3, 3] := '0,74';
    gridB.Cells[0,3] := '0,68';

end;

procedure TFormMain.SetGridSize(rank: Word);
var
    oldrank: Word;
    i: Word;
    j: Word;

```

```
begin
  oldrank := GridMatrix.ColCount;

  GridMatrix.ColCount := rank;
  GridMatrix.RowCount := rank;
  GridB.RowCount := rank;

  if oldrank < rank then
    begin
      for i := oldrank to rank - 1 do
        for j := 0 to rank - 1 do
          GridMatrix.Cells[j, i] := '0';
        for i := 0 to oldrank - 1 do
          for j := oldrank to rank - 1 do
            GridMatrix.Cells[j, i] := '0';

          for i := oldrank to rank - 1 do
            GridB.Cells[0, i] := '0';
          end;
        end;

      procedure TFormMain.SpinEditRankChange(Sender: TObject);
      begin
        SetGridSize((Sender as TSpinEdit).Value);
      end;

    end.
```

ДОДАТОК Д. ОПИС ВИКОРИСТАНИХ КОМПОНЕНТІВ

Назва компоненти	Функціональне призначення	Опис
TSpinEdit	Завдання розмірності системи	Компонент класу TSpinEdit (вкладка Samples палітри інструментів) призначений виключно для введення цілих чисел. Дві кнопки, що знаходяться в правій частині компонента, надають можливість покрокової зміни значення чисел.
TStringGrid	Завдання коефіцієнтів та вільних членів	Дозволяє відображати текстові дані в осередках, розташованих в рядках і стовпцях. У таблиці рядків знаходиться багато властивостей і методів для контролю і відображення даних, так само як і для використання переваг їх табличного розташування.
TButton	Запуск виконання обчислень	Зазвичай за допомогою компонента TButton (Кнопка) користувач ініціює виконання якого-небудь фрагмента коду або цілої програми.
TStatusbar	Вивід додаткової інформації	Є стрічкою стану в стилі Windows. Компонент TStatusBar (Стрічка стану) зручно використовувати для відображення різної інформації, корисної при виконанні програми. Стрічка стану може мати одну або декілька панелей для виведення текстової інформації.
TSplitter	Зміна розмірів елементів на формі	Компонент TSplitter призначений для ручного (за допомогою миші) управління розмірами контейнерів TPanel, TGroupBox або подібних до

		них під час виконання програми. Візуально він є невеликою вертикальною або горизонтальною смугою, розташованою між двома сусідніми контейнерами
ТМемо	Вивід результатів обчислень	Компонент ТМемо (Область перегляду) призначений для виводу на екран декількох рядків тексту. Текст зберігається у властивості Lines класу TStrings і, таким чином, є пронумерованим набором рядків (нумерація починається з нуля).