Маматкасымова А.Т., Сатыбаев А.Дж.

РАЗРАБОТКА ПРОГРАММЫ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ЛИНЕЙНОЙ АЛГЕБРЫ

A.T. Mamatkasymova, A.Dzh. Satybaev

DEVELOPMENT PROGRAM FOR THE SOLUTION LINEAR ALGEBRA

УДК: 372.681.5

В данной работе разработаны алгоритмы и программы на языке C++ для решения основных задач линейной алгебры. Для наглядности полученных результатов разработана демонстрационно-тестирующая программа. Результаты могут быть использованы на практике для решения систем линейных уравнений и других задач линейной алгебры.

In this paper, the algorithms and programs in C++ language for solving the basic problems of linear algebra. For clarity, the results developed demonstration and test program. The residts can be used in practice for solving systems of linear equations and other linear algebra problems.

Постановка задачи. При решении задачи линейной алгебры методом Крамера приходиться вычислять определитель матрицы det (A) размером пхп:

$$A = \begin{Vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{Vmatrix} . \qquad 1)$$

$$A' = \begin{Vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a'_{22} & \dots & a'_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a'_{nn} \end{Vmatrix} ,$$

при этом det(A) = det(A′).

Формула для пересчета элементов матрицы имеет вид:

$$a'_{jk} = a_{jk} \left(1 - \frac{a_{ik}}{a_{ii}}\right) , \qquad (2)$$

где i - номер столбца, в котором элементы, лежащие ниже главной диагонали, превращаются в нули;
j - номер элемента в обрабатываемом столбце (т.е. номер строки);
k - номер элемента в текущей строке.

Алгоритм приведения матрицы к треугольному виду включает в себя 3 цикла:
- внешний цикл, i = 1 .. n-1 ; - средний цикл, j = i+1 .. n ; - внутренний цикл, k = i+1 .. n

Теперь искомый определитель вычисляется как произведение диагональных элементов:

$$\det(A) = \prod_{i=1}^{n} a_{ii} .$$

Описанный выше алгоритм дает результат не всегда. Если при выполнении i-того шага внешнего цикла диагональный элемент a_{ii} оказывается равным нулю, а среди элементов i-того столбца с номерами от i+1 до n есть хотя бы один не нулевой, алгоритм завершается безрезультатно (из-за невозможности вычислений по формуле (2). Для того, чтобы это не происходило, используется прием «выбор главного элемента».

При выполнении очередного шага цикла по i предварительно выполняются следующие операции:

1) находится максимальный по модулю элемент среди элементов i-того столбца от a_{ii} до a_{ni} ;
2) если найденный элемент a_{li} равен нулю, процесс вычисления завершается с выдачей результата det(A) = 0;
3) если l≠i , тогда строки исходной матрицы с номерами i,l поменять местами.

После завершения преобразования матрицы, определитель вычисляется по формуле:

$$\det(A) = (-1)^p \prod_{i=1}^{n} a_{ll} ,$$

где p – число выполненных операций перемены строк местами.

Обратной к матрице A называется матрица A^{-1}, обладающая свойством:

$$A \cdot A^{-1} = A^{-1} \cdot A = I ,$$

где I – единичная диагональная матрица. Опишем один из универсальных и эффективных методов расчета обратной матрицы (метод Жордана-Гаусса описан как «метод исключений»).

Пусть имеем матрицу A вида (1) и пусть B – единичная диагональная матрица такого же размера. Создадим рабочую матрицу R размером N×2N просто присоединив матрицу B справа к матрице A :

$$R = \left\| \begin{matrix} a_{11} & a_{12} & \dots & a_{1n} & 1 & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & a_{2n} & 0 & 1 & \dots & 0 \\ \dots & & & & & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} & 0 & 0 & \dots & 1 \end{matrix} \right\| .$$

Левую часть матрицы R будем называть подматрицей A, правую – подматрицей B. Весь процесс преобразования матрицы R разобьем на 3 этапа.

1 этап. Выполним преобразования строк матрицы так, чтобы все элементы, лежащие ниже диагональных элементов подматрицы A, обратились в нули. При этом может использоваться выбор главного элемента.

2 этап. Выполним преобразования так, чтобы все элементы, лежащие выше диагональных элементов подматрицы A, обратились в нули. Преобразования надо выполнять в обратном порядке: со столбца номер n и до столбца номер 2.

3 этап. Каждую строку расширенной матрицы R с номером i делим на диагональный элемент a_{ii} .

После завершения процедуры подматрица A превращается в единичную диагональную матрицу, а подматрица B будет равна искомой обратной матрице A^{-1}. Алгоритм имеет порядок $O(n^3)$.

Методы решения систем линейных уравнений. Задача поиска решений системы линейных уравнений имеет не только самостоятельное значение, но часто является составной частью алгоритма решения многих нелинейных задач. Основными методами решения задачи являются СЛАУ:

- метод Гаусса; - метод обращения матрицы; - итерационные методы.

Метод Гаусса. Пусть имеем систему линейных уравнений:

$$\begin{matrix} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{matrix}$$

Простой метод Гаусса состоит в следующем.

Составим расширенную матрицу, приписав к матрице коэффициентов СЛАУ дополнительный столбец – правые части уравнения:

$$\left\| \begin{matrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \dots & & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{matrix} \right\| .$$

Выполним над строками расширенной матрицы преобразования, аналогичные выше:

$$a'_{jk} = a_{jk}\,(1 - \frac{a_{ik}}{a_{ii}}) , \quad b'_j = b_j\,(1 - \frac{b_i}{a_{ii}}) ,$$

и приведем ее к треугольному виду:

$$\left\| \begin{matrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ 0 & a_{22} & \dots & a_{2n} & b_2 \\ \dots & & & & \\ 0 & 0 & \dots & a_{nn} & b_n \end{matrix} \right\| .$$

Теперь можно вычислить искомые величины x_i , начиная с последнего, т.е. вначале находится x_n , затем x_{n-1}, x_{n-2}, … , x_1. Формула для вычислений имеет вид:

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{k=i+1}^{n} a_{ik} x_k \right) .$$

Для расширения возможностей и повышения устойчивости приведенного алгоритма используется выбор главного элемента. Порядок метода Гаусса равен $O(n^3)$.

Метод обращения матрицы. Представим систему линейных уравнений в форме:

$$A \cdot \vec{X} = \vec{B} .$$

Умножим последнее равенство слева на A^{-1}:

$$A^{-1} \cdot A \cdot \vec{X} = A^{-1} \cdot \vec{B} .$$

Учитывая, что $A^{-1} \cdot A = I$, формально получаем искомое решение:

$$\vec{X} = A^{-1} \cdot \vec{B}$$

Таким образом, решение системы выполняется в два этапа:

1) вычисление обратной матрицы;
2) умножение обратной матрицы на вектор правых частей СЛАУ.

Несмотря на то, что метод обращения матрицы имеет такой же порядок, как и метод Гаусса - $O(n^3)$, по объему вычислений он проигрывает ему в несколько раз. Однако, если СЛУ необходимо решать многократно и при этом изменяется лишь вектор правых частей, метод обращения матрицы становится все же выгодным.

Описание класса Matrix для решения задач линейной алгебры. Класс имеет приватные и общедоступные члены-данные и члены-функции (методы). Для хранения компонентов матрицы используется одномерный динамический массив элементов типа параметра шаблона. Для создания объекта предусмотрены три конструктора: конструктор по умолчанию, конструктор с параметрами, конструктор копирования: и деструктор. Для выполнения множества матричных операций созданы перегруженные операции: присваивания (=), сложения (+), вычитания (-), умножения (*) и т.п. На базе операторов ввода/ вывода С++ разработаны функции ввода матриц из потока и вывода их в поток, предусматривающие в случае файлового ввода/вывода и текстовую форму хранения, и бинарную.

Доступ к членам-данным класса - числу строк и столбцов матрицы осуществляется с помощью методов size_row () и size col (). Для доступа к элементам матрицы создан перегруженный оператор вызова функции operator () (dim x, dim x), где dim - переопределенный тип unsigned char. Вызов функции используется как оператор индексирования, принимающий два аргумента. Аналогично создан оператор вызова функции с одним аргументом operator() (dim x). Для данного класса - это очень важные перегруженные операторы, т.к. они используются во всех функциях и операторах, в которых происходит обращение к элементам матрицы.

Описание функций, конструкторов и деструкторов класса:

1. *Конструктор по умолчанию Matrix ():*

Конструктор по умолчанию создает матрицу нулевого размера. В дальнейшем размер этой матрицы можно изменить с помощью функции newsize(i, j).

2. *Конструктор с параметрами Matrix (dim, dim-1):*

Это обычный конструктор с параметрами, который принимает в качестве параметров размеры матрицы и создает одномерный динамический массив размером m*n, где m - число строк, an- число столбцов матрицы. С целью возможности использовать его для создания векторов, второй параметр конструктора задан как умалчиваемый со зйачением 1. Для первоначальной «инициализации» элементов матрицы нулями используется функция setmem ().

3. *Конструктор копирования Matrix (const Matrix &):*

Конструктор принимает в качестве параметра ссылку на объект класса (на существующую матрицу), определяет ее размер, выделяет для нее память и копирует в эту память содержимое матрицы, принимаемой по ссылке. Таким образом, создается точная копия матрицы с текущими значениями ее элементов.

4. *Деструктор ~Matrix ():*

Деструктор высвобождает память, выделенную конструкторами для элементов матрицы.

5. *Функция операции присваивания "=" Matrix& operator = (Matrix&):*

Данная функция сравнивает адрес передаваемого по ссылке объекта с адресом собственного класса, чтобы не предпринялась попытка присвоит объект самому себе. После этого создается новый массив с числом элементов, равным числу элементов массива принимаемого по ссылке, и в этот массив заносится содержимое принимаемого массива. Возвращается разыменованный указатель this (return *this;).

6. *Функции операций суммирования, вычитания, умножения матриц и умножения матрицы на число:*

Эти функции реализованы как дружественные функции и алгоритмы этих функций аналогичны по своему составу. Общий вид прототипа этих функций: friend Matrix operator @(const Matrix&, const Matrix&). Применение дружественных функций в данном случае целесообразно для того, чтобы иметь возможность передавать в оператор функцию объекты в любой последовательности. В этих операторах-функциях вначале создается временный объект типа матрица (с помощью конструктора копирования), в который копируется первая матрица и который при выходе из функции является возвращаемым объектом. Затем эта матрица суммируется (вычитается, умножается) с матрицей, стоящей после знака оператора по соответствующим правилам матричных операций. Для доступа к элементам матрицы и индексирования используются перегруженные операторы вызова функции operator () (dim x, dim x) и operator () (dim x).

7. Функция - оператор Matrix operator^A (int):

Этот оператор-функция реализован как член класса и предназначен для возведения матрицы в степень. В случае, когда значение входного параметра равно минус единице осуществляется вызов функции вычисления обратной матрицы методом преобразований Гаусса, для чего разработана отдельная функция Matrix & Gauss(dim, dim). Таким образом, использование этого оператора позволяет решать систему линейных алгебраических уравнений в представлении X - (A^A-1)*B, где X и B -вектора неизвестных и правых частей соответственно.

8. Функция - оператор Matrix operator! ():

Оператор для определения транспонированной матрицы.

9. Функция - оператор friend VARTYPE operator % (const Matrix&, const Matrix&):

Функция вычисления скалярного произведения векторов. В ней в начале проверяется, являются ли передаваемые объекты векторами, а затем вычисляется скалярное произведение.

10. Функции-члены VARTYPE determ () и VARTYPE vmodule ():

Первая функция вычисляет определитель собственного объекта (матрицы). При этом используются функция Matrix & Gauss(dim, dim). Функция VARTYPE vmodule() вычисляет длину (модуль) вектора

11. Функция операции вывода friend ostream& operator « (ostream&, Matrix&):

Данная функция не может быть членом класса, поэтому чтобы иметь доступ к приватным элементам класса, она объявлена "дружественной" функцией. Она имеет два параметра: ссылку на поток, который находится слева от знака операции «, и ссылку на объект, который находится слева от знака операции, данные этого объекта и будут выводиться. Затем следует форматированный вывод в поток всех элементов массива и возвращается поток.

Если требуется вывести данные в файл, нужно открыть его присоединением к потоку ofstream.

12. Функция операции ввода friend istream& operator » (istream&, Matrix&):

Так же как и предыдущая, данная функция не может быть членом класса, а поэтому для доступа к приватным элементам класса объявлена "дружественной" функцией класса. Она так же имеет два параметра: ссылку на поток, который находится слева от знака », и ссылку объект, который находится слева от знака операции, в него и будут вводиться данные из потока. Затем следует ввод данных из потока в элементы массива и возвращается поток. Для ввода данных из файла, нужно открыть его присоединением к потоку ifstream.

13.Функции-члены dim write (ofstream&) и dim read (ifstream&):

Функции предназначены для вывода в файл и ввода из файла матриц в из двоичном представлении. Для этого необходимо передать в них соответствующую ссылку на открытый файл.

Нами составлена алгоритм программа результаты тестирования класса Matrix. Сложение матриц A и B: на языке C++ , а также тестирующая программа и получены

A:			B:		C=A+B:		
1.1 2.2	3.3	2	7 2		3.1	9.2	5.3
2.4 1.1	4.4	4	8 1		6.4	9.1	5.4
1.3 2.1	4.1	6	4 1		7.3	6.1	5.1

Вычитание матриц A и B:

A:			B:		C=A-B:		
1.1 2.2	3.3	2	7 2		-0.9	-4.8	1.3
2.4 1.1	4.4	4	8 1		-1.6	-6.9	3.4
1.3 2.1	4.1	6	4 1		-4.7	-1.9	3.1

Сложение матриц A и B:

A:			B:		C=A*B:		
1.1 2.2	3.3	2	7 2		30.8	38.5	7.7
2.4 1.1	4.4	4	8 1		35.6	43.2	10.3
1.3 2.1	4.1	6	4 1		35.6	42.3	8.8

Сложение векторов Вычитание векторов Умножение матрицы на вектор

V: H:	X=V+H	X=V-H:	A:	C=A*V:
2.1 * 1.1	3.2	1	1.1 2.2 3.3	14.212
3.31 2.1	5.41	1.21	2.4 1.1 4.4	14.841
1.4 3.1	4.5	-1.7	1.3 2.1 4.1	15.421

```
Запись матрицы в файл  Считывание матрицы из файла   Вычисление обратной матрицы
      D:                      E:                          A:
      1 2 3                   1 2 3                       1.1 2.2 3.3
      2 5 6                   2 5 6                       2.4 1.1 4.4
      7 3 9                   7 3 9                       1.3 2.1 4.1
      C=A^-1:                    Решение алгебраического уравнения   A^-1:
2.009346 0.88785 -2.570093          2.009346 0.88785 -2.570093
1.750212 -0.093458 -1.308411        1.750212 -0.093458 -1.308411
-1.53356 -0.233645 1.728972         -1.53356 -0.233645 1.728972
V:      X:
2.1     3.56028   Определение детерминанта матрицы determinant of A = -2.354
3.31    1.534325  Определение длины (модуля) вектора modul of V = 4.162463
1.4     -1.57328  Вычисление скалярного произведения векторов scalar product V*U= 16.
```

Вывод. В результате данной работы были разработаны два класса функций для решения простейших задач линейной алгебры. Число этих функций сравнительно невелико, однако можно легко добавить в эти классы более сложные функции, построенные на базе уже имеющихся. Классы позволяют работать с матрицами и векторами, элементы которых могут быть любого типа, однако на практике чаще всего используется целый тип и тип чисел с плавающей запятой.

Классы написаны на языке C++, однако могут быть легко переписаны на любом из современных языков программирования, так как приведены довольно простые алгоритмы всех компонентных функций. Были максимально предусмотрены все возможные ошибки, которые могут возникнуть при использовании функций данных классов. Особое внимание уделялось разумному выделению памяти под объекты во время выполнения программы, поэтому все функции были тщательно отлажены.

Классы Matrix и Vector могут быть эффекивно применены на практике в задачах, требующих операций с матрицами и векторами, а также связанных с решением систем линейных алгебраических уравнений.

Литература:

1. Бахвалов Н.С. Численные методы. Т. 1.2. М.: Наука, 1993 г.
2. Бахвалов Н.С. Основы вычислительной математики. М.:МГУ, 1970 г.
3. Боглаев Ю.П. Вычислительная математика и программирование. М..ВШ, 1990 г.
4. Воеводин В.В. Вычислительные методы линейной алгебры. М.: Наука, 1997 г.
5. Годунов С.К. Решение систем линейных уравнений. М.: Нука, 1980 г.
6. Мак Кракен Д., Дорн У. Численные методы и программирование на Фортране. М.:ИЛ, 1977
7. Самарский А.А. и др. Численные методы. М.: Нука, 1986 г.
8. Дискретная математика, конспект лекций. В. Г. Засовенко. Запорожье, 1998 г.
9. Начальный курс С и С++. Б. И. Березин. Москва: "ДИАЛОГ-МИФИ", 1999 г.
10. Язык программирования С++. Б. Страуструп. Киев:"ДиаСофт", 1993 г.

Рецензент: д.пед.н., профессор Калдыбаев С.К.