УДК 519.61

Объектная реализация методов вычислительной алгебры

Л.В. Маркова, Е.А. Корчевская, А.Н. Красоткина
Учреждение образования «Витебский государственный университет имени П.М. Машерова»

В статье рассматривается интеграционный путь формирования профессиональных компетенций будущих специалистов в области IT-технологий. Предлагается новый подход к построению учебного материала дисциплины «Вычислительные методы алгебры», обеспечивающий реализацию принципа единства теоретической полноты и практической направленности заданий посредством применения изученных в курсе «Программирование» методов объектно-ориентированной технологии программирования. В рекомендуемом подходе все вычислительно-конструктивные понятия линейной алгебры рассматриваются в качестве объектов и представляются единой классификационной иерархией. Наряду с четырьмя основными классами, объединяющими функциональные группы общих и элементарных матриц, иерархия включает классы самих методов вычислительной алгебры. Такой подход дает возможность студентам изучать базовые вычислительные алгоритмы и приобретать навыки программирования на основе современной технологии.

Ключевые слова: *компетенции, профессиональные навыки, алгоритмы, методы, вычислительная алгебра, программирование.*

Object implementation of Computer Algebra methods

L.V. Markova, E.A. Korchevskaya, A.N. Krasotkina
Educational establishment «Vitebsk State University named after P.M. Masherov»

The article scrutinizes the integration path of formation of professional competence of future specialists in the field of information technologies. It offers a new approach to the compilation of the academic material of the course of Computer Methods of Algebra, which provides for the implementation of the principle of the unity of theoretical comprehensiveness and practical focus in tasks through application of object-oriented programming techniques studied within the course of Programming. Within the proposed approach all computational design concepts of Linear Algebra are treated as objects and laid out in single classification hierarchy. Together with the four main classes that combine the basic functional groups of the general and elementary matrices, the hierarchy includes classes of the Computer Algebra methods. This approach allows students to learn basic computer algorithms and acquire programming skills on the basis of modern technology.

Key words: *competence, job skills, algorithms, methods, Computer Algebra, programming.*

Сегодня, в XXI веке, который называют веком технологий, ни одна конкурентоспособная сфера жизни человека не может обходиться без высоких технологий. Это особо касается сферы образования. Современные технологии в образовании рассматриваются как средство, с помощью которого может быть реализована новая образовательная парадигма и повышается качество образования. Качество образования – это соответствие образования интересам человека, общества и государства, поэтому в сегодняшних условиях основной тенденцией образования является его профессионализация. Нынешние образовательные технологии опираются на принципы научности, креативности, вариативности, интегрированности, системности и практической направленности. Построение учебного процесса на основе использования современных научных достижений в области изучаемого предмета через совместную творческую деятельность преподавателя и студента на занятии при многообразии вариантов решения поставленной проблемы позволяет достичь основного результата – связать теорию с формированием практических умений и навыков [1].

Цель исследования – на основе применения принципа органического единства теоретической полноты и законченности предлагаемых заданий и их прикладной, практической направленности анализ возможности приобретения и совершенствования профессиональных компетенций студентами, обучающимися по специальности 1-31 03 03 Прикладная математика.

Материал и методы. Объектом исследования является процесс формирования профессиональных компетенций у студентов, обучающихся по специальности «Прикладная математика». Основные методы – системный подход,

деятельностная парадигма образования, анализ научной, педагогической и нормативной литературы. Исследования проводились на базе кафедры прикладной математики и механики ВГУ имени П.М. Машерова. Материалами послужили труды теоретиков и практиков по данной проблеме, а также многолетний опыт работы авторов. На первичном этапе был осуществлен анализ содержания курсов «Программирование» и «Методы вычислительной алгебры», которые читаются студентам специальности «Прикладная математика» в 1–4 семестрах.

Результаты и их обсуждение. В соответствии с программой дисциплины «Вычислительные методы алгебры» для студентов, обучающихся по специальности 1-31 03 03 Прикладная математика в 4 семестре, предусмотрен лабораторный практикум, цель которого состоит в том, чтобы:

- ознакомить студентов с основными численными методами решения задач линейной алгебры и изучить эти методы;
- получить практический опыт использования подобных алгоритмов для решения задач вычислительной математики;
- усовершенствовать практические навыки применения современных технологий программирования.

Программировать лабораторные задания предлагается на основе объектно-ориентированной технологии (ООП) [2]. В таком подходе все вычислительно-конструктивные понятия линейной алгебры рассматриваются в качестве объектов и представляются единой классификационной иерархией.

В рамках предмета «Вычислительные методы алгебры» иерархия матричных классов выглядит следующим образом:

- AbstractMatrix
  - Vector
  - DiagonalMatrix
    - ThreeDiagonalMatrix
    - EMatrix
  - SquareMatrix
    - FrobeniusMatrix
    - JacobiMatrix
  - AugmentMatrix
    - SwapMatrix

Вершиной матричной иерархии является абстрактный класс «AbstractMatrix», обобщающий свойства всех матриц. Далее иерархия продолжается тремя основными классами: «Vector» (Вектор), «SquareMatrix» (Квадратная матрица) и «AugmentMatrix» (Расширенная матрица). Эти классы объединяют основные функциональные группы элементарных матриц.

Механизм наследования классов позволяет строить иерархии, в которых производные классы получают элементы родительских или базовых классов и могут дополнять их или изменять их свойства.

Класс «Vector» является родителем трех основных классов: «DiagonalMatrix» (Диагональная матрица), «ThreeDiagonalMatrix» (Трехдиагональная матрица), «EMatrix» (Единичная матрица).

Класс «DiagonalMatrix» определяет группу диагональных матриц, содержащих ненулевые элементы только на главной диагонали. На основе класса диагональных матриц строится класс «ThreeDiagonalMatrix» – трехдиагональных матриц.

Класс «EMatrix» представляет традиционный математический объект – единичную матрицу. Данный тип матрицы достаточно часто используется в вычислительной математике для представления более сложных матричных типов.

Класс «SquareMatrix» реализует семейство элементарных квадратных матриц. В рамках этого класса выделяют две группы – класс «FrobeniusMatrix» (Матрица Фробениуса) и класс «JacobiMatrix» (Матрица Якоби), которые необходимы для решения задач на собственные значения.

Класс «AugmentMatrix» представляет собой группу расширенных матриц, в которой отдельно рассматривается класс матриц перестановок «SwapMatrix». Этот класс определяет элементарную перестановку пары столбцов или пары строк. Данный матричный класс соответствует преобразованиям переупорядочения столбцов и строк в основной матрице, которые обычно применяются в методах выбора главного элемента.

Создание конкретного матричного класса в рамках ООП подхода сводится к реализации набора операций с элементарными матрицами, который определяется родительским классом «AbstractMatrix». В данном классе описывается общая структура, которая потом будет конкретизирована и дополнена в производных классах. По мере продвижения вниз по иерархии классы приобретают все больше определенных черт. Таким образом, организованное множественное наследование позволяет одному классу обладать свойствами двух и более родительских классов.

Для основных матричных объектов виртуально определяются необходимые операции – получение и установка значения элемента, подсчет количества строк и столбцов матрицы, сложение, вычитание, умножение матриц. Опе-

рации матричного умножения определяются в двух вариантах, которые соответствуют левостороннему и правостороннему умножению.

Исходя из того, что класс «AbstractMatrix» является вершиной иерархии, то в данном классе следует описать две виртуальные функции getElement и setElement, необходимые для получения и установки значения соответственно. Для иллюстрации используем язык программирования C++.

```
class AbstractMatrix{
   public:
       // количество строк матрицы
       int size;
       // получение значения элемента
        virtual double getElement(int i, int j) = 0;
       // установка значения элемента
        virtual void setElement(int i, int j, double
element)=0;
 };
```

Класс «Vector» наследуется от абстрактного класса «AbstractMatrix» и является его потомком, поэтому в данном классе переопределяются виртуальные методы абстрактного класса и добавляются свои методы для работы с векторами.

```
class Vector : public AbstractMatrix{
       // элементы вектора
        double *elements;
   public:
      // размер вектора
        int size;
      /*конструктор*/
        Vector();
        Vector(int size);
        Vector(int size, double *elements);
       . . . . . .
       // получение значения элемента вектора
        double getElement(int i, int j);
       // установка значения элемента вектора
        void setElement(int i, int j, double element);
       // получение количества строк
        int getRowCount();
            . . . . . . };
```

Операция умножения двух квадратных матриц реализуется в классе «SquareMatrix» и может выглядеть следующим образом (язык программирования C++):

```
SquareMatrix SquareMatrix :: operator *(SquareMatrix B){
         int n = this->size;
         int m = B.getColCount();
        // объявление результирующей матрицы
         double **C = new double*[n];
         for(int i=0; i < n; i++){
             C[i] = new double[m];
             }
        // вычисление произведения матриц
         for(int i = 0; i < n; i++){
             for(int j = 0; j < m; j++){
                 for(int k = 0; k < n; k++){
                     C[i][j] += this->elements[i][k] *
B.elements[k][j];
                         }
                 }
             }
        // создание матрицы
         return SquareMatrix(n, C);
      };
```

Для эффективного применения ООП необходимо также сами методы линейной алгебры рассматривать как реализацию объектов соответствующих классов в их целостной объектной классификации [3]. При этом ООП классификация должна следовать классификации задач линейной алгебры, поскольку результатом решения различных постановок являются объекты разных типов, что должно отражаться в спецификациях алгоритмических классов. Кроме того, в классификации желательно отразить деление алгоритмов на прямые и итерационные, поскольку принципы их организации существенно отличаются, а это неизбежно приводит к различиям и в их программной реализации.

Ввиду сделанных замечаний объектную классификацию алгоритмов линейной алгебры представим условной иерархией следующего вида:

- VMA
  - FactorizationAlgorithms
    - LU_decomposition
    - LDU_decomposition
    - STS_decomposition
    - SDS_decomposition
  - Eigenvalues
  - SLAU
    - DirectMethods
      - DirectMethodsFactorization
        - gaussMethod
        - squareMethod
      - DirectMethodsNF
        - gaussJordanMethod
        - gaussChooseElement
        - sweepMethod
    - IterationMethods
      - simpleIterationMethod
      - zeidelMethod
      - methodSkorSpusk

Подобный подход обеспечивает существенную программную общность, поскольку реализация методов решения новых классов линейных задач, отличающихся от имеющихся типами матричных объектов, сводится к созданию производных классов в рамках единой матричной иерархии. При этом основная часть вычислительных методов непосредственно реализуется в общих матричных классах и автоматически наследуется всеми производными классами.

В данной схеме вершиной иерархии является базовый класс «VMA». Далее все алгоритмы линейной алгебры «VMA» классифицируются на алгоритмы факторизации «FactorizationAlgorithms», основные алгоритмы решения систем линейных алгебраических уравнений «SLAU» и класс «Eigenvalues», который предназначен для реализации численных методов нахождения собственных значений и собственных векторов матриц.

В классе «SLAU» описаны методы для решения систем линейных алгебраических уравнений. Данные методы подразделяются на прямые и итерационные. В связи с этим выделены классы для реализации прямых методов решения СЛАУ «DirectMethods» и итерационных – «IterationMethods». Класс «DirectMethods» разбивается на два производных класса: прямые методы, использующие факторизацию, им соответствует класс «DirectMethodsFactorization» и прямые методы, не использующие факторизацию, – «DirectMethodsNF».

Класс «FactorizationAlgorithms» содержит методы факторизации, среди которых выделены четыре метода: LU, LDU, STS и SDS-разложение.

LU-разложение – это представление квадратной матрицы A в виде произведения нижней треугольной матрицы L на верхнюю треугольную матрицу U.

LDU-разложение – это представление квадратной матрицы A в виде произведения LDU, где L – нижняя треугольная матрица с единичной диагональю, D – диагональная матрица, а U – верхняя треугольная матрица с единичной диагональю.

STS-разложение – это представление симметричной положительно определенной матрицы A в виде произведения матриц S^T и S, где S – верхняя треугольная матрица, S^T – транспонированная к ней матрица (нижняя треугольная).

SDS-разложение – это представление симметричной матрицы A в виде произведения матриц S^T, D и S, где S – верхняя треугольная матрица, S^T – транспонированная к ней матрица (нижняя треугольная), D – диагональная матрица с элементами, равными +1 или –1.

На языке программирования C++ реализация алгоритма STS-разложения для произвольной симметричной матрицы имеет вид:

```
void FactorizationAlgorithms :: STS_decomposition
(SquareMatrix A, SquareMatrix S) {
      for (int i = 0; i < A. getRowCount(); i++) {
         for (int j = 0; j < i; j++) {
      double sum = 0;
         for (int k = 0; k < j; k++) {
      sum += S.getElement(k, i) * S.getElement(k, j);
                              }
      S.setElement(j, i,(A.getElement(i, j)-
sum)/S.getElement(j, j));
                              }
      double temp = A.getElement(i, i);
      for (int k = 0; k < i; k++) {
      temp -= S.getElement(k, i) * S.getElement(k, i);
                              }
       S.setElement(i, i, sqrt(temp)); } };
```

При таком подходе студенты имеют возможность изучать основные вычислительные алгоритмы и приобретать навыки программирования на основе современной технологии. Причем существенно упрощается программная реализация изучаемого метода, позволяющая доминировать его теоретической алгоритмической и практической составляющим. В качестве подтверждения вышесказанного приведем пример реализации метода квадратного корня на языке программирования C++.

```
void DirectMethodsFactorization :: squareMe-
thod(SquareMatrix A, Vector f){
   //Создание матриц S, вектора x, вектора y
   int n = A.getRowCount();
   SquareMatrix S = SquareMatrix(n);
   Vector x = Vector(n);
   Vector y = Vector(n);
   /* STS – разложение. В результате факторизации
   имеем матрицу S – верхнюю треугольную матрицу */
   FactorizationAlgorithms FA;
   FA. STS_decomposition (A, S);
   /* Решение системы STy = f. Для решения систе-
   мы необходимо получить транспонированную
   матрицу ST – нижнюю треугольную матрицу, для
   этого необходимо вызвать соответствующий ме-
   тод, реализованный в классе «SquareMatrix»*/
   /* Решение системы Sx = y. Матрица S – верхняя
   треугольная матрица, следовательно, искомый
   вектор решения находим обратным ходом метода
```

```
    Гаусса.*/
    for(int i = n-1; i >= 0; i--){
    double sum = 0;
       for(int j = n-1; j > i; j--){
      sum += x.getElement(j) * S.getElement(i, j);
                }
    x.setElement(i, (y.getElement(i) -
sum)/S.getElement(i, i));
                }
// вывод вектора решений };
```

Заключение. Таким образом, для формирования профессиональных компетенций будущих программистов необходимо при построении учебного процесса усиливать роль принципа интеграции на основе современных методических приемов и тенденций в области IT-технологий.

ЛИТЕРАТУРА

1. Маркова, Л.В. Формирование профессиональных компетенций у студентов специальности «Прикладная математика» / Л.В. Маркова, Н.Д. Адаменко, О.Г. Казанцева, Е.А. Корчевская // Весн. Віцебск. дзярж. ун-та. – 2012. – № 1(67). – С. 116–121.
2. Маркова, Л.В. Обучение вычислительной математике. Современные аспекты / Л.В. Маркова, Е.А. Корчевская // Инновационные технологии обучения физико-математическим дисциплинам: материалы междунар. науч.-практ. интернет-конф., посвященной 60-летию доктора физико-математических наук, профессора Н.Т. Воробьева, Витебск, 21–22 июня 2011 г. – Витебск, 2011. – С. 128–129.
3. Семенов, В.А. Объектно-ориентированный подход к программированию прямых методов линейной алгебры / В.А. Семенов, О.А. Тарлапан // Вопросы кибернетики. Приложения системного программирования / под ред. В.П. Иванникова. – М.: Науч. совет по комплексной проблеме «Кибернетика» РАН, 1996. – Вып. 2. – С. 147–170.

Поступила в редакцию 01.03.2013. Принята в печать 24.04.2013.
Адрес для корреспонденции: e-mail: kat955@mail.ru – Маркова Л.В.