Concept of the course “Numerical methods in object methodology”

Aleksandr P. Polishchuk[1], *Sergei A.* Semerikov[1]

[1]*Kryvyi Rih State Pedagogical University, 54 Gagarin Ave., Kryvyi Rih, 50086, Ukraine*

Abstract. The tasks for which computers were created – routine calculations of an industrial, scientific and military nature – required the creation of a whole class of new methods focused not on manual but on machine calculations. The first programming languages did not have convenient means for reflecting such objects often used in computational mathematics as matrices, vectors, polynomials, etc. Further development of programming languages followed the path of embedding mathematical objects into languages as data types, which led to their complication. So, for example, an attempt to make a universal language Ada, in which there are even such data types as dictionaries and queues, led to the fact that the number of keywords in it exceeded 350, making it almost unusable for learning and use. The compromise solution between these two extremes can be the following: let the programmer himself create the data types that he needs in his professional work. Programming languages that implement this approach are called object-oriented. This, on the one hand, makes it possible to make the language quite easy by reducing the number of keywords, and on the other, expandable, adapting to specific tasks by introducing keywords for creating and using new data types.

Keywords: C++, numerical methods, mathematical classes

Задачи, ради которых и были созданы компьютеры – рутинные расчёты производственного, научного и военного характера, – потребовали создания целого класса новых методов, ориентированных не на ручные, а на машинные вычисления. Первые языки программирования не обладали удобными средствами для отражения таких часто используемых в вычислительной математике объектов, как матрицы, вектора, полиномы и т.д. Дальнейшее развитие языков программирования шло по пути встраивания математических объектов в языки как типов данных, что вело к их усложнению. Так, например, попытка сделать универсальный язык Ада, в котором есть даже такие типы данных, как словари и очереди, привела к тому, что количество ключевых слов в нём превысило 350, сделав его практически непригодным для изучения и использования.

Компромиссным решением между этими двумя крайностями может быть следующее: пусть программист сам создаёт типы данных, которые ему необходимы в его профессиональной деятельности. Языки программирования, в которых реализован такой подход, называют объектно-ориентированными. Это, с одной стороны, позволяет сделать язык достаточно лёгким путём уменьшения количества ключевых слов, а

KMITO 1999: Conference on Computer Simulation and Information Technology in Education, April 19–21, 1999, Kryvyi Rih, Ukraine

apol@cabletv.dp.ua (A. P. Polishchuk); cc@kpi.dp.ua (S. A. Semerikov)

https://kdpu.edu.ua/semerikov (S. A. Semerikov)

ACNS Conference Series: Social Sciences and Humanities

с другой – расширяемым, приспосабливаемым к конкретным задачам введением ключевых слов для создания и использования новых типов данных.

Наиболее удобным инструментом для создания классов математических объектов является объектно-ориентированное программирование и его поддержка в языке C++. Этот язык дает возможность варьировать методы создания математических объектов путем определения в классе необходимого количества конструкторов, осуществить переопределение стандартных операций для вновь созданных классов, использовать мощный механизм одиночного и множественного наследования свойств базовых классов в производных классах, создавать параметризованные классы и функции с подстановкой типов параметров в процессе конструирования соответствующих объектов. Последовательное наращивание иерархии математических типов на базе уже созданных позволяет существенно снизить трудоемкость программирования за счет исключения повторяющихся последовательностей действий и избежать внесения в программы новых ошибок.

Одной из самых важных учебно-методических линий в курсе алгебры средней школы является линия уравнений и неравенств, а также их систем. В высшей школе эта линия продолжается на более высоком уровне в курсе линейной алгебры и векторных пространств, что позволило нам выделить следующие математические объекты: вектора, матрицы и многочлены. Наш выбор обуславливался ещё тем, что все они тесно связаны друг с другом. Так, например, матрицу можно рассматривать как упорядоченный кортеж арифметических векторов, а многочлен можно задать вектором его коэффициентов.

Для каждого из этих типов в процессе написания работы понадобилось определить множество процедур, в большинстве своём – бинарных алгебраических операций над элементами соответствующих множеств. Результатом работы явилось создание библиотеки классов параметризованных векторных, матричных и полиномиальных объектов, расширяющих возможности языка C++ по работе с такими объектами. Рассмотрим детальнее механизм реализации этой библиотеки.

Базовым в нашей иерархии классов является класс для работы с арифметическими векторами. В нашей интерпретации вектор – это упорядоченный кортеж некоторых объектов (или, иными словами, массив определённой длины). Тип чисел, составляющих компоненты вектора, не является строго фиксированным – этот тип можно использовать как параметр конструктора векторного объекта, то есть вы можете иметь целочисленные, вещественные, комплексные и т.д. вектора различных длин. Основные операции, определённые для векторов – это сложение, вычитание, отрицание, скалярное умножение, умножение вектора на скаляр, сравнение векторов, вывод вектора в поток и ввод его из потока, нахождение модуля вектора и нормирование его по модулю, а также ряд операций сокращённого сложения, вычитания и т.д. Разумеется, большинство этих операций определены только для векторов совпадающих размерностей.

Для реализации арифметических операций был использован механизм перегрузки операций, благодаря которому, например, при сложении двух векторов в программе достаточно поставить знак “+” между объектами векторного класса точно так же, как это делается при сложении стандартных целых и вещественных чисел. Такая запись является более естественной, чем вызов функции Add, хотя, по сути, ничем от неё не отличается. Это свойство является особенностью именно C++, в отличие от других

объектно-ориентированных языков.

Следующим классом, разработанным нами, был класс, предназначенный для хранения и использования, пожалуй, самых важных алгебраических объектов – матриц. Структура матричного класса подобна структуре векторного, но более содержательная. Это обусловлено множеством операций, которые необходимо было запрограммировать для эффективной работы с новым типом данных – "матрица". Как и векторный класс, матричный богат конструкторами, которые позволяют создавать матрицы, инициализируя их данными из памяти, из файла, из другой матрицы и так далее. Кроме того, допустимо создание «пустой», т.е. нулевой матрицы заданного размера.

В качестве внутреннего представления матрицы был выбран упорядоченный кортеж векторов – объектов векторного класса, рассмотренного нами ранее, что позволяет использовать для матричных операций перегруженные операции класса "вектор". Например, для сложения двух матриц совпадающих размерностей нам достаточно сложить вектора, их составляющие, что позволяет вместо двух циклов сложения элементов матриц использовать один цикл сложения векторов, составляющих строки матрицы.

Для матриц совпадающей размерности определены операции сложения, вычитания, а также их сокращённые аналоги; умножение матрицы на скаляр, транспонирование, сравнение матриц, вывод в поток и ввод из потока. Для отдельных типов матриц определена операция умножения.

Практический интерес при работе с матрицами представляет решение систем линейных алгебраических уравнений, задаваемых соответственно матрицами коэффициентов при неизвестных и вектор-столбцом свободных членов. В связи с этим были реализованы следующие методы решения СЛАУ: метод Гаусса с выбором главного элемента, метод ортогонализации векторов матрицы, метод обратной матрицы, метод Крамера.

Вычисление обратной матрицы и детерминанта можно производить либо аналитически, либо численно. Нами были реализованы оба этих подхода. Так, Вы можете воспользоваться рекурсивной функцией вычисления детерминанта разложением его по какой-либо строке и функцией обращения матрицы с использованием алгебраических дополнений. При этом с ростом порядка матрицы количество операций сложения и умножения возрастает настолько, что эти методы стают малоэффективными. Более эффективным является вычисление детерминанта и обратной матрицы косвенно, путём решения системы уравнений.

Перегрузка перечисленных операций даёт нам возможность записи операций над матрицами в близкой к алгебраической форме. К примеру, решение системы уравнений может быть записано как решение матричного уравнения $\mathbf{AX} = \mathbf{B}$, $\mathbf{X} = \mathbf{A^{-1}} * \mathbf{B}$, где операция возведения в степень "-1" не что иное, как перегруженная функция обращения матрицы, а "$*$" – операция умножения матриц.

Наличие, наряду с умножением и обращением, операции транспонирования, позволяет нам одной строчкой программы записать решение задачи МНК – метода наименьших квадратов. Пусть $\mathbf{X}$ и $\mathbf{Y}$ – соответственно матрицы независимых и зависимых переменных, $\mathbf{A}$ – неизвестный вектор оценки МНК. Тогда $\mathbf{A} = X$ транспонированное, умноженное на $\mathbf{X}$ (всё в минус первой степени), умноженное

на произведение транспонированной матрицы независимых переменных на вектор-столбец зависимых переменных:

*matrix a=(&(x*x))*(x*y);*

Класс для работы с многочленами от одной переменной базируется на векторе – действительно, операции над многочленами сводятся к действиям над коэффициентами при соответствующих степенях неизвестной. Это даёт возможность использовать арифметический вектор для представления многочлена. Как и в предыдущих классах, полиномиальные объекты имеют набор методов для конструирования, сложения, вычитания, умножения полинома на полином и полинома на скаляр, сравнения, деления с остатком и т.п.

Для полиномиальных объектов мы можем найти производную любого порядка и неопределённый интеграл любой кратности в аналитической форме; результатом будет полином, коэффициенты которого получаются по соответствующим правилам. Кроме того, в любой точке мы можем найти функциональное значение полинома.

В задачах прикладной математики полиномиальные объекты, как правило, используются при решении алгебраических уравнений, что побудило нас к реализации ряда методов решения уравнений разного порядка. Согласно основной теореме алгебры, полином n-ной степени имеет ровно n корней. Это позволяет считать, что решением полиномиального уравнения в комплексной области является комплексный вектор решений с размерностью, равной степени многочлена, компонентами которого являются его корни.

Для многочленов с действительными коэффициентами нами были рассмотрены метод Кардано-Тартальи для решения уравнений 3-ей степени и базирующийся на нём метод Феррари решения уравнений 4-ой степени. Более универсальными методами для комплексных многочленов являются методы решения квадратных уравнений и метод Ньютона для поиска корней многочлена любой степени. В последнем методе мы, находя очередной корень, отделяем его, понижая степень многочлена по схеме Горнера, ищем корень многочлена меньшей степени и т.д., до отделения всех корней.

Алгоритмы и программы решения проблемы собственных значений для несимметричных комплексных матриц, реализованные в традиционной методологии, всегда отличались громоздкостью. Для её решения используются все разработанные нами классы – вектора, матрицы и полиномы, существенно сокращая объём программы и повышая её наглядность.

В стройном здании математики более сложные математические объекты строятся из более простых. Так, комплексные числа представляются в виде пары вещественных координат вектора по вещественной и мнимой осям комплексной плоскости. Многомерный числовой вектор может быть представлен совокупностью его вещественных или комплексных координат по осям многомерной координатной системы (или как частный случай матрицы – одностолбцовой или однострочной), матрица может быть представлена как вектор векторов, полином известного порядка может быть задан как вектор его коэффициентов.

Для каждого класса математических объектов определен допустимый набор математических операций и способы их реализации; например, операции умножения

определены для вещественных чисел, векторов и матриц, но имеют, естественно, различный смысл и алгоритмы реализации. Операция определения нулей специфична для полиномов, транспонирования и вычисления собственных значений и собственных векторов – для матриц, определения модуля – для векторов.

Представляется целесообразным построить курс вычислительных методов по принципу определения иерархии математических классов, объекты которых конструировались бы затем в программе путем объявления, то есть синтаксически так же, как и стандартные для используемого языка типы (целые, вещественные и пр.) с определением внутри класса всех необходимых для их использования в вычислениях операций. При этом под термином “операция” можно понимать как общепринятые для простых типов операции, например, арифметические, так и любые, базирующиеся на данном математическом классе вычисления, – например, решение системы линейных алгебраических уравнений или вычисление коэффициентов регрессии для заданной матрицы или вычисление корней полинома наряду с операциями полиномиальной арифметики – сложения, умножения, деления полиномов.

Нам неизвестны пособия с систематическим изложением методов вычислений на базе объектно-ориентированного подхода в программной реализации на языке C++. Уже сейчас во многих средних и высших учебных заведениях осуществляется преподавание языка C и C++ и предлагаемая работа может, по нашему мнению, оказать положительное влияние на эффективность учебного процесса в области вычислительной математики.

Представляемое учебное пособие [1] рассчитано на сравнительно небольшой двухсеместровый курс численных методов (2 часа в неделю лекций + 2 часа лабораторных работ в компьютерном классе), который читается студентам специальности “Математика и информатика” педагогических институтов. Изложение курса предполагает владение основами объектно-ориентированного программирования на языке C++.

В соответствии с уже изложенной концепцией объектно-ориентированной программной реализации многие методы вычислений инкапсулируются в классы математических объектов, с которыми они работают; например, метод наименьших квадратов и метод решения систем линейных алгебраических уравнений будут размещены в классе матриц, а полиномиальная арифметика и методы вычисления полиномиальных нулей – в классе полиномов.

Глава 2 посвящена рассмотрению специальных математических типов (и операций над ними) и определению соответствующих им классов в терминах языка C++. Вначале в качестве иллюстрации рассматривается необходимый при изучении последующего материала (например, методов вычисления корней полиномов при наличии среди них комплексных) предположительно знакомый слушателю и реализованный в библиотеке C++ класс комплексных чисел; его программная реализация взята прямо из среды разработки Borland C++ и по возможности откомментирована – этот материал служит своеобразным образцом в реализации других рассмотренных в этой главе математических классов – векторов, полиномов, матриц. Изучение матричного класса сопровождается изложением методов решения основных задач линейной алгебры – систем линейных уравнений, вычисления собственных значений и векторов матриц.

Глава 3 содержит изложение методов полиномиальной и экспоненциальной аппроксимации функций и их программную реализацию, также инкапсулированную в

виде функций-членов специального класса.

Глава 4 является естественным прикладным продолжением предыдущей и содержит методы численного интегрирования и дифференцирования функций с использованием рассмотренных методов приближения функций.

Глава 5 посвящена методам решения обыкновенных дифференциальных уравнений; при этом рассматриваются не только численные методы, а иллюстрируется применение приближенных численных методов при программной реализации аналитических решений. Например, при решении дифференциальных уравнений методами операционного исчисления возникает задача вычисления корней характеристических уравнений, которая может быть решена численно.

Глава 6 содержит введение в поисковые методы определения экстремумов функций при отсутствии и наличии шумов в определении значения функции и методы программирования соответствующих задач.

Апробация курса “Численные методы в объектной методологии” в Криворожском педуниверситете в течение последних лет свидетельствует о повышении качества усвоения учебного материала за счёт переключения внимания обучаемого с деталей программной реализации на сам метод благодаря приближению программной записи алгоритма к естественной математической и использованию таких типов данных, как векторы, матрицы и полиномы.

Наличие готовой библиотеки математических объектов существенно ускоряет процесс программной реализации метода, сокращая не только время, но и объём программы, делая её более “прозрачной” за счёт повышения уровня абстракции до операций над новыми типами данных (это проявляется, например, в использовании для умножения матриц вместо трёх вложенных циклов знака умножения). Параметризация программ позволяет порождать из шаблонов типов специализированные реализации, делая, к примеру, из параметризованной матрицы действительную, комплексную либо функциональную простой подстановкой типа (double, complex, function) в угловые скобки поле имени параметризованного объекта.

Применение этих типов позволило расширить традиционный курс численных методов разделами, обычно вызывающими трудности в программной реализации в процедурной идеологии (символическое исчисление, линейное и динамическое программирование), по-новому взглянуть на традиционные методы и расширить область их применения.

References

[1] Polishchuk, A.P. and Semerikov, S.A., 1999. *Metody vychislenii v klassakh iazyka C++ [Numerical methods in C++].* Krivoi Rog: Izdatelskii otdel KGPI. Available from: http://elibrary.kdpu.edu.ua/handle/0564/755.