Министерство образования Республики Беларусь
Учреждение образования «Витебский государственный университет имени П.М. Машерова»

Л.В. Маркова, Е.А. Корчевская, А.Н. Красоткина

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ АЛГЕБРЫ. ПРАКТИКУМ

Пособие

Рекомендовано учебно-методическим объединением по естественнонаучному образованию в качестве пособия для студентов учреждений высшего образования, обучающихся по специальности 1-31 03 03 «Прикладная математика (по направлениям)»

Витебск
ВГУ имени П.М. Машерова
2013

УДК 519.61(076.5)
ББК 22.193я73-5
М26

Печатается по решению научно-методического совета учреждения образования «Витебский государственный университет имени П.М. Машерова». Протокол № 9 от 20.06.2013 г.

Авторы: доценты кафедры прикладной математики и механики ВГУ имени П.М. Машерова, кандидаты физико-математических наук **Л.В. Маркова, Е.А. Корчевская;** преподаватель кафедры прикладной математики и механики ВГУ имени П.М. Машерова **А.Н. Красоткина**

Р е ц е н з е н т ы :
заведующий кафедрой вычислительной математики Белорусского государственного университета, кандидат физико-математических наук *П.А. Мандрик;* научный сотрудник государственного научного учреждения «Институт математики НАН Беларуси», кандидат физико-математических наук *М.А. Курдина*

Маркова, Л.В.

М26 Вычислительные методы алгебры. Практикум : пособие / Л.В. Маркова, Е.А. Корчевская, А.Н. Красоткина. – Витебск : ВГУ имени П.М. Машерова, 2013. – 148 с.
ISBN 978-985-517-401-2.

Пособие представляет собой руководство к выполнению лабораторно-практических работ по дисциплине «Вычислительные методы алгебры», содержит краткие теоретические сведения, все необходимые соотношения и формулы, методические указания, примеры, а также варианты заданий для выполнения лабораторной работы. Предложен новый подход к построению практической части учебного материала дисциплины «Вычислительные методы алгебры» на основе современной технологии объектно-ориентированного программирования. Такой подход способствует формированию у студентов профессиональных компетенций. Настоящее пособие составлено в соответствии с программой дисциплины «Вычислительные методы алгебры» для специальности 1-31 03 03 «Прикладная математика», но может быть использовано для подготовки студентов других специальностей, имеющих в своих учебных планах вычислительную математику.

УДК 519.61(076.5)
ББК 22.193я73-5

ISBN 978-985-517-401-2

С О Д Е Р Ж А Н И Е

П Р Е Д И С Л О В И Е

Настоящее пособие составлено в соответствии с программой дисциплины «Вычислительные методы алгебры» для студентов специальности 1-31 03 03 «Прикладная математика», но может быть использовано для подготовки студентов других специальностей, имеющих в своих учебных планах вычислительную математику.

Использование предлагаемого практикума, по мнению авторов, позволит достичь следующих целей:

1) ознакомить читателя с основными численными методами решения задач линейной алгебры и изучить их;

2) получить практический опыт использования этих алгоритмов для решения задач вычислительной математики;

3) совершенствовать практические навыки технологии объектно-ориентированного программирования на основе применения ее к задачам и методам линейной алгебры.

Пособие представляет собой руководство к выполнению лабораторно-практических работ по дисциплине «Вычислительные методы алгебры» и состоит из четырех глав:

- Элементы теории погрешностей.
- Объектно-ориентированный подход к программированию методов линейной алгебры.
- Решение систем линейных алгебраических уравнений.
- Вычисление собственных значений и собственных векторов матриц.

Каждая глава содержит краткие теоретические сведения, все необходимые соотношения и формулы, методические указания и примеры, а также задания для выполнения лабораторных работ в соответствии с учебной программой курса «Вычислительные методы алгебры».

Программировать лабораторные задания предлагается на основе технологии объектно-ориентированного программирования (ООП). В таком подходе все вычислительно-конструктивные понятия линейной алгебры рассматриваются в качестве объектов и представляются единой классификационной иерархией. Наряду с широкими классами общих, специальных и элементарных матриц, иерархия включает классы самих задач и методов линейной алгебры. Данный 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 частных же классах переопределяется лишь часть методов, реализация которых возможна или целесообразна с учетом специфических математических, вычислительных и программных особенностей вводимых матричных типов.

Отправной точкой для применения ООП к программированию методов линейной алгебры и разработки унифицированного подхода к их реализации послужило то обстоятельство, что прямые методы базируются теоретически на тех или иных элементарных матричных преобразованиях, которые приводят задачу к эквивалентной, но более простой форме, допускающей ее непосредственное решение. Причем набор типов преобразований, необходимый для реализации большинства прямых методов, оказывается относительно небольшим.

При такой постановке занятий студенты имеют возможность изучить основные вычислительные алгоритмы и приобрести навыки практической реализации численного решения модельных и прикладных задач на основе принципов современного программирования.

Авторы выражают благодарность рецензентам: заведующему кафедрой вычислительной математики Белорусского государственного университета, кандидату физико-математических наук П.А. Мандрику и научному сотруднику государственного научного учреждения «Институт математики НАН Беларуси», кандидату физико-математических наук М.А. Курдиной за ценные советы и замечания.

Глава 1
ЭЛЕМЕНТЫ ТЕОРИИ ПОГРЕШНОСТЕЙ

П 1.1 Источники погрешностей

Анализ ошибок (или, как говорят чаще, погрешностей) является неотъемлемой частью процесса решения прикладной задачи. Часть этих погрешностей связана с вычислениями, которые в наше время производятся на ЭВМ. С увеличением скорости производства вычислений и с вовлечением в счетный процесс чисел с большим количеством значащих цифр, как это делается в ЭВМ, потребность в оценке фактической точности результата лишь возрастает. При этом следует правильно рассматривать сам термин «ошибка», который в данном случае выражает объективно неизбежную погрешность, сопровождающую процесс решения задачи, начиная с измерения исходных значений. Задача анализа ошибок сводится, по существу, к отысканию их надежных границ и соблюдению условий, обеспечивающих их минимальное распространение.

Возникновение, накопление и распространение ошибок проходят через все стадии решения прикладной задачи, начиная с получения значений исходных данных.

Погрешность решения задачи обуславливается следующими причинами:

1) математическое описание задачи является неточным, т.к. при описании модели всегда присутствуют определенные ограничения;

2) неточность задания исходных данных модели. Исходные данные чаще всего известны приближенно, что связано с неточностью измерения числовых параметров различными приборами;

3) метод, применяемый для решения задачи, как правило, является приближенным. Получение точного решения исходной математической задачи требует неограниченного или неприемлемо большого числа арифметических операций, поэтому вместо точного решения задачи приходится прибегать к приближенному;

4) действия над приближенными числами, а также обязательное округление, связанное с двоичным представлением чисел и конечностью разрядной сетки ЭВМ, вносят дополнительную погрешность.

Погрешности, соответствующие этим причинам, называют:

1) погрешностью математической модели;

2) погрешностью исходных данных (неустранимая погрешность);

3) погрешностью метода;

4) вычислительной погрешностью.

П 1.2 Вычисление абсолютной и относительной погрешностей

Пусть X – точное значение некоторой величины, а x – наилучшее из известных приближений. В этом случае ошибка (или погрешность) приближения x определяется разностью $X - x$. Обычно знак этой ошибки не имеет решающего значения, поэтому рассматривают абсолютную величину ошибки:

$$e_x = |X - x|. \qquad (1)$$

Величина e_x, называемая абсолютной погрешностью приближенного значения x, в большинстве случаев остается неизвестной, так как для ее вычисления нужно знать точное значение X. На практике обычно удается установить верхнюю границу абсолютной погрешности, т.е. такое число Δx, для которого справедливо неравенство

$$\Delta x \geq |X - x|. \qquad (2)$$

Число Δx называют предельной абсолютной погрешностью (или границей абсолютной погрешности) приближения x.

Таким образом, предельная абсолютная погрешность приближенного числа x – это всякое число Δx, не меньшее абсолютной погрешности e_x этого числа.

По абсолютной погрешности нельзя в полной мере судить о точности измерений или вычислений. Качество приближения измеряется с помощью относительной погрешности, которая определяется как отношение ошибки e_x к модулю значения X (когда оно неизвестно, то к модулю приближения x).

Предельной относительной погрешностью (или границей относительной погрешности) δx приближенного числа называется отношение предельной абсолютной погрешности к абсолютному значению приближения x:

$$\delta x = \frac{\Delta x}{|x|}. \qquad (3)$$

Формула (3) позволяет при необходимости выражать абсолютную погрешность через относительную:

$$\Delta x = |x| \delta x. \qquad (4)$$

Относительную погрешность выражают обычно в процентах, и тогда она умножается на 100.

Неравенство (2) позволяет установить приближения к точному значению *X* по недостатку и избытку [4]:

$$x - \Delta x < X < x + \Delta x, \qquad (5)$$

которые могут рассматриваться как одна из возможных пар значений соответственно нижней границы (НГ) и верхней границы (ВГ) приближения *x:*

$$НГ_x = x - \Delta x; \quad ВГ_x = x + \Delta x. \qquad (6)$$

Пример 1. Найти абсолютную и относительную погрешности числа $\pi = 3{,}14159265358...$, заданного тремя цифрами после запятой, т.е. $x = 3{,}141$.

Тогда абсолютную погрешность рассчитаем по формуле (1) $e_x = |\pi - x| = 0{,}0005926...$.

Предельной абсолютной погрешностью будет верхняя граница абсолютной погрешности (2), т.е. $\Delta x = 0{,}0006$.

Вычислим предельную относительную погрешность по формуле (3) $\delta x = \dfrac{0{,}0006}{3{,}141} \cdot 100\% \approx 0{,}02\% \cdot$

Тогда можем записать $\pi = 3{,}141 \pm 0{,}0006$; $\pi = 3{,}141(1 \pm 0{,}02\%)$.

В дальнейшем рассматриваются только предельные погрешности, поэтому слово «предельная» будет опускаться.

П 1.3 Округление чисел

Значащими цифрами числа называют все цифры в его записи, начиная с первой ненулевой слева.

Пример 1. У чисел а = 0,0<u>3045</u>, а = 0,0<u>3045000</u> значащими цифрами являются подчеркнутые цифры. Число значащих цифр в первом случае равно 4, во втором – 7.

Цифра числа называется верной (в широком смысле), если абсолютная погрешность этого числа не превосходит единицы разряда, в котором стоит эта цифра.

Цифра числа называется верной (в строгом смысле), если абсолютная погрешность этого числа не превосходит половины единицы разряда, в котором стоит эта цифра.

Излишне сохраненные цифры, помимо верных, называются сомнительными.

Вычислить приближенное число с точностью $\varepsilon = 10^{-n}$ означает, что необходимо сохранить верной значащую цифру, стоящую в n-м разряде после запятой.

В приближённых вычислениях часто приходится округлять как точные, так и приближённые числа. Округлением числа называют замену его близким по величине, но с меньшим количеством значащих цифр. Округление числа производят путем отбрасывания одной или нескольких последних цифр в десятичном представлении числа.

При округлении соблюдают следующие правила.

- Если первая из отброшенных цифр больше 5, то к последней оставшейся цифре прибавляется единица.
- Если первая из отбрасываемых цифр меньше 5, то оставшиеся десятичные знаки сохраняются без изменения.
- Если первая из отброшенных цифр равна 5, а среди остальных отброшенных цифр есть ненулевые, то к последней оставшейся цифре прибавляется единица.
- Если первая из отброшенных цифр равна 5 и остальные отброшенные цифры нулевые, то последняя оставшаяся цифра не изменяется, если она четная, и увеличивается на единицу, если она нечетная.

Во многих практических задачах пользуются упрощёнными правилами округления, согласно которым цифра, если за ней стоят цифры 0, 1, 2, 3, 4, при округлении не изменяется и увеличивается на 1 в противоположном случае.

Также различают округление к большему (с избытком) и к меньшему (с недостатком).

Погрешности округления в ЭВМ числа x, обусловленные конечностью разрядной сетки, могут быть вычислены по формуле:

$$\delta(x) = \frac{1}{\alpha_m}\left(\frac{1}{s}\right)^{n-1},$$

где α_m – первая значащая (отличная от нуля) цифра; s – основание системы счисления; n – разрядность компьютера.

Пример 2. Округлив число 0,1544 до трех значащих цифр, определить абсолютную и относительную погрешности полученного приближенного числа.

Пусть X = 0,1544, тогда x = 0,154 – исходное число, округленное до трех значащих цифр. Рассчитаем абсолютную погрешность: $e_x = |X - x| = 0,0004$.

Вычислим относительную погрешность:

$$\delta x = \frac{0,0004}{0,1544} \cdot 100\% \approx 0,26\% \cdot$$

Тогда можем записать

$$X = 0,154 \pm 0,0004;\ X = 0,154\ 1 \pm 0,26\% \ .$$

Пример 3. Определить количество верных цифр в числе $a = 2,91385$, если известна его абсолютная погрешность $\Delta a = 0,0097$.

Используем определение верной цифры:

$\Delta a \le 1$	2	верно
$\Delta a \le 0,1$	9	верно
$\Delta a \le 0,01$	1	верно
$\Delta a \le 0,001$	3	неверно

В итоге получили, что в данном числе в широком смысле верны 3 цифры.

Пример 4. Определить количество верных цифр в числе $a = 18,572$, если известна его относительная погрешность $\delta a = 0,4\%$.

Вычислим абсолютную погрешность по формуле (1.4):

$$\Delta a = 18,572 * 0,4 / 100 = 0,074288.$$

Используем определение верной в строгом смысле цифры:

$\Delta a \le 10/2$	1	верно
$\Delta a \le 1/2$	8	верно
$\Delta a \le 0,1/2$	5	неверно

Верными в строгом смысле являются только цифры 1, 8.

П 1.4 Вычисление погрешностей арифметических операций

Рассмотрим правило вычисления погрешностей арифметических операций и функций по погрешности аргументов (без учета ошибок округления). При вычислении абсолютных погрешностей обычно используются формулы дифференцирования, в которых дифференциалы независимых переменных заменяются абсолютными погрешностями: $dx_i = e_x^i$.

Пусть $X_1>0$, $X_2>0$ – точные значения величин, и заданы предельные абсолютные погрешности Δx_1 и Δx_2, т.е. $X_1 = x_1 \pm \Delta x_1$, $X_2 = x_2 \pm \Delta x_2$. Необходимо найти погрешность суммы $X = X_1 + X_2$.

Так как дифференциал $dX = dX_1 + dX_2$, то $e_x = e_x^1 + e_x^2 \le \Delta x_1 + \Delta x_2$. Отсюда следует, что $\Delta x = \Delta x_1 + \Delta x_2$.

Теперь предположим, что $X_1 > X_2$, и найдем погрешность разности $X = X_1 - X_2$. Тогда $dX = dX_1 - dX_2$ и $e_x = e_x^1 - e_x^2 \le \Delta x_1 + \Delta x_2$. Поэтому $\Delta x = \Delta x_1 + \Delta x_2$.

Абсолютная погрешность суммы и разности двух приближенных чисел равна сумме абсолютных погрешностей слагаемых.

Это правило справедливо для произвольного числа слагаемых. Так если $x_1, x_2, \ldots, x_n$ имеют одну и ту же погрешность Δx, то погрешность суммы этих слагаемых будет равна $n\Delta x$. Но реально погрешности могут иметь разные знаки и поэтому взаимно компенсировать друг друга. По правилу Чеботарева при $n > 10$ погрешность суммы можно принять равной $\sqrt{3n}\Delta x$ [20].

Для относительной погрешности суммы и разности двух чисел $X = X_1 \pm X_2$ получаем

$$\delta x = \frac{\Delta x}{x} = \frac{\Delta x_1 \pm \Delta x_2}{x_1 \pm x_2} = \frac{\Delta x_1 \cdot x_1}{x_1 \pm x_2 \;\; x_1} + \frac{\Delta x_2 \cdot x_2}{x_1 \pm x_2 \;\; x_2} = \frac{x_1}{x_1 \pm x_2}\delta x_1 + \frac{x_2}{x_1 \pm x_2}\delta x_2.$$

Для произвольного числа слагаемых

$$\delta x = \frac{x_1}{x}\delta x_1 + \frac{x_2}{x}\delta x_2 + \ldots + \frac{x_n}{x}\delta x_n,$$

где $x = x_1 + x_2 + \ldots + x_n,\ x_i > 0,\ i = 1, 2, \ldots, n$.

Пусть $\max_i \delta x_i = M,\ \min_i \delta x_i = m.$

Тогда $\delta x \le \frac{x_1 \cdot M + \ldots + x_n \cdot M}{x} = M,\ \delta x \ge \frac{x_1 \cdot m + \ldots + x_n \cdot m}{x} = m.$

Поэтому $m \le \delta x \le M$.

Аналогично при выполнении умножения и деления получаем погрешности (при тех же предположениях).

Найдем погрешность произведения $X = X_1 \cdot X_2$:

$$\Delta x = e_x^1 \cdot x_2 + e_x^2 \cdot x_1 \le \Delta x_1 \cdot x_2 + \Delta x_2 \cdot x_1,$$

$$\delta x = \frac{\Delta x}{x_1 \cdot x_2} = \delta x_1 + \delta x_2.$$

Найдем погрешность частного $X = X_1 / X_2$:

$$\Delta x = \frac{e_x^1 \cdot x_2 - e_x^2 \cdot x_1}{x_2^{\,2}} \le \frac{x_1 \cdot \Delta x_2 + x_2 \cdot \Delta x_1}{x_2^{\,2}},$$

$$\delta x = \delta x_1 + \delta x_2.$$

Следовательно, относительная погрешность произведения двух чисел равна сумме относительных погрешностей его сомножителей.

Аналогичное правило выполняется и для частного от деления двух чисел.

Также получаются формулы для погрешностей арифметических действий при вычислении функций многих переменных. Так, для $z = f(x_1, x_2, ..., x_n)$ имеем следующие формулы для погрешностей [20]:

$$\Delta z = \left|\frac{\partial f}{\partial x_1}\right| \Delta x_1 + \left|\frac{\partial f}{\partial x_2}\right| \Delta x_2 + ... + \left|\frac{\partial f}{\partial x_n}\right| \Delta x_n,$$

$$\delta z = \left|\frac{x_1}{f} \cdot \frac{\partial f}{\partial x_1}\right| \delta x_1 + \left|\frac{x_2}{f} \frac{\partial f}{\partial x_2}\right| \delta x_2 + ... + \left|\frac{x_n}{f} \frac{\partial f}{\partial x_n}\right| \delta x_n =$$

$$= \left|x_1 \cdot \frac{\partial \ln f}{\partial x_1}\right| \delta x_1 + \left|x_2 \frac{\partial \ln f}{\partial x_2}\right| \delta x_2 + ... + \left|x_n \frac{\partial \ln f}{\partial x_n}\right| \delta x_n.$$

П 1.5 Оценка погрешности по способу границ

Способ границ применяется для оценки погрешности результата расчета по формуле, содержащей приближенные величины.

Пусть a – приближенное исходное данное, так что заданы его границы: $НГ(a) \le a \le ВГ(a)$. Нужно найти результат y, зависящий от a и поэтому также приближенный: $y = f\ a$.

Найдем границы y, т.е. два таких числа $НГ\ y$ и $ВГ\ y$, что $НГ(y) \le y \le ВГ(y)$. Если результат y увеличится с ростом a (например, площадь квадрата в зависимости от длины стороны), то, по определению границ, $ВГ(y) \ge f(a)$ для любого $a \in [НГ(a), ВГ(a)]$ и можно принять $ВГ\ y = f\ ВГ\ a$. Аналогично $НГ\ y = f\ НГ\ a$. Если y убывает с ростом a (например, давление воздуха с высотой), то $ВГ\ y = f\ НГ\ a$ и $НГ\ y = f\ ВГ\ a$.

Для нахождения границ результата нужны два расчета по одному и тому же алгоритму с исходными данными $НГ\ a$ и $ВГ\ a$. Границы результата округляют так: $НГ$ – с недостатком, $ВГ$ – с избытком. При этом в их записи сохранить все цифры до первой слева, отличие в которой $НГ\ y$ и $ВГ\ y$ уже существенно.

Тогда значение y можем найти следующим образом:

$$y = \frac{НГ\ y\ + ВГ(y)}{2}.$$

А границы погрешности оцениваются:

$$\Delta y \le \frac{ВГ\ y\ - НГ(y)}{2}.$$

Случай функции нескольких переменных.

Пусть a, b – исходные данные, известные приближенно:

$$НГ\ a\ \le a \le ВГ(a),\ НГ\ b\ \le b \le ВГ(b).$$

Если результат $y = f($ [illegible], b монотонно зависит от своих аргументов, то крайние его значения достигаются при некоторых комбинациях граничных значений исходных данных.

В общем случае следует провести 2 * 2 = 4 вычислений и выбрать из них пару: наибольший ($ВГ(y)$) $ВГ\ y$ и наименьший (НГ(y)) результаты. В случае трех исходных данных рассматривается 2^3 = 8 расчетов и т.д. задача упрощается, если из ее постановки ясен характер зависимости: рост или убывание хотя бы по одному аргументу.

Пример 1. Произвести расчет по заданной формуле для приведенных исходных данных. Рассчитать границы, погрешность и значение результата.

$$G = \frac{4 \cdot \pi^2 \cdot [illegible]_1^2 - a_2^2}{T_1^2 \cdot a_1 - T_2^2 \cdot a_2}.$$

$$a_1 = 1.22 \pm 0.002,$$

$$a_2 = 2.33 \pm 0.002,$$

$$T_1 = 4.61 \pm 0.01,$$

$$T_2 = 6.72 \pm 0.02.$$

Проведем некоторые предварительные расчеты по длинной формуле. Выясним, как зависит функция G от a_1, a_2, T_1, T_2.

Зафиксируем a_1, a_2, T_1 и вычислим

G(1.22, 2.33, 4.61, 6.74) = 1.946536,

G(1.22, 2.33, 4.61, 6.70) = 1.977531.

Видно, что с возрастанием T_2 убывает функция G.

Зафиксируем a_1, a_2, T_2 и вычислим

G(1.22, 2.33, 4.62, 6.72) = 1.964724,

G(1.22, 2.33, 4.60, 6.72) = 1.959158.

Видно, что G возрастает с ростом T_1.

Зафиксируем a_1, T_1, T_2 и вычислим
G(1.22, 2.332, 4.61, 6.72) = 1.964339,
G(1.22, 2.328, 4.61, 6.72) = 1.959528.
Видно, что G возрастает с ростом a_2.
Зафиксируем a_2, T_1, T_2 и вычислим
G(1.222, 2.33, 4.61, 6.72) = 1.960553,
G(1.218, 2.33, 4.61, 6.72) = 1.963309.
Видно, что с возрастанием a_1 убывает функция G.
Найдем границы G:
ВГ(G) = G(*НГ*(a_1), *ВГ*(a_2), *ВГ*(T_1), *НГ*(T_2)) = 1.984156,
НГ(G) = G(*ВГ*(a_1), *НГ*(a_2), *НГ*(T_1), *ВГ*(T_2)) = 1.940027.
Видно, что различие в третьей цифре уже существенно.
Округляем *НГ* – с недостатком, *ВГ* – с избытком.
ВГ(G) = 1.99,
НГ(G) = 1.94.

Тогда $G = \frac{НГ\ G\ + ВГ(G)}{2} = 1.965,\ \Delta y \le \frac{ВГ\ G\ - НГ(G)}{2} = 0.025.$

Лабораторная работа № 1

Цель: изучить правила нахождения погрешностей и область неопределенности результата.

Задание

1. Округляя следующие числа до трех значащих цифр, определить абсолютную и относительную погрешности полученных приближенных чисел:

Варианты заданий

1	2,1514	**6**	-392,85	**11**	46,453
2	0,16152	**7**	0,1545	**12**	0,088748
3	0,01204	**8**	0,003922	**13**	0,34484
4	1,225	**9**	625,55	**14**	0,096835
5	-0,0015281	**10**	94,525	**15**	0,037862

2. Определить количество верных цифр в числе, если известна его a) абсолютная, b) относительная погрешности:

Варианты заданий

1	a) x=0.3941, Δ_x=0.25*10^{-2} b) a=1.8921, δ_a=0.1*10^{-2}	**2**	a) x=0.1132, Δ_x=0.1*10^{-3} b) a=0.2218, δ_a=0.2*10^{-1}
3	a) x=38.2543, Δ_x=0.27*10^{-2} b) a=22.351, δ_a=0.1	**4**	a) x=293.481, Δ_x=0.1 b) a=0.02425, δ_a=0.5*10^{-2}
5	a) x=2.325, Δ_x=0.1*10^{-1} b) a=0.000135, δ_a=0.15	**6**	a) x=14.00231, Δ_x=0.1*10^{-3} b) a=9.3598, δ_a=0.1%
7	a) x=0.0842, Δ_x=0.15*10^{-2} b) a=0.11452, δ_a=10%	**8**	a) x=0.00381, Δ_x=0.1*10^{-4} b) a=48361, δ_a=1%

9 a) x=-32.285, $\Delta_x=0.2*10^{-2}$
b) a=592.8, $\delta_a=2\%$

10 a) x=-0.2113, $\Delta_x=0.5*10^{-2}$
b) a=14.9360, $\delta_a=1\%$

11 a) x=8.3445, $\Delta_x=0.2*10^{-2}$
b) a=46.453, $\delta_a=0.15\%$

12 a) x=0.38725, $\Delta_x=4.2*10^{-4}$
b) a=5.8425, $\delta_a=0.2\%$

13 a) x=-6.4257, $\Delta_x=2.4*10^{-3}$
b) a=17.2834, $\delta_a=0.3\%$

14 a) x=22.553, $\Delta_x=0.1*10^{-2}$
b) a=2.8546, $\delta_a=0.3\%$

15 a) x=0.75244, $\Delta_x=0.1*10^{-4}$
b) a=-97041.6, $\delta_a=5\%$

3. Произвести расчет по заданной формуле для приведенных исходных данных. Рассчитать границы, погрешность и значение результата.

Варианты заданий

1. $$V=\sqrt{\frac{e}{p}\times\frac{1-\eta}{(+\eta)\cdot(-2\cdot\eta)}},$$
где $e=(2.32\pm0.02)\times10^{-6}$, $\eta=0.33\pm0.01$, $p=(3.26\pm0.01)\times10^{3}$.

2. $$Z=x\cdot tg\alpha-\frac{g\cdot x^2}{2\cdot v^2\cdot\cos^2\alpha},$$
где $\alpha=\frac{\pi}{4}$, $g=9.81\pm0.01$, $x=12.23\pm0.01$, $v=15.0$.

3. $$p(x)=\frac{1}{\sqrt{2\cdot\pi\cdot\sigma}}^{\frac{-(x-a)^2}{2\cdot\sigma^2}},$$
где $\sigma=(3.32\pm0.01)\times10^{-1}$, $x=2.23\pm0.01$, $a=2.32\pm0.01$.

4. $$S=\frac{v}{\alpha}\times\left[(-\alpha\cdot t)\cdot\ln(-\alpha\cdot t)+\alpha\cdot t\right],$$
где $t=4.20\pm0.02$, $v=300\pm1$, $\alpha=(3.22\pm0.03)\times10^{-2}$.

5. $$G=\frac{4\cdot\pi^2\cdot\left[a_1^2-a_2^2\right]}{T_1^2\cdot a_1-T_2^2\cdot a_2},$$
где $a_1=1.22\pm0.002$, $a_2=2.33\pm0.002$, $T_1=4.61\pm0.01$, $T_2=6.72\pm0.02$.

6. $$C=B\times\frac{\cos E-e}{1-e\cdot\cos E},$$
где $B=3.22\pm0.22$, $E=45.0\pm0.5$, $e=0.33\pm0.01$,

7. $$a=\frac{a_0}{\sqrt{\left[1-\omega^2/\omega_0^2\right]^2+4\cdot n^2\cdot\omega_0^2/\omega^2}},$$
где $a_0=12.34\pm0.02$, $\omega=5000\pm5$, $\omega_0=4000\pm5$, $n=4$.

8. $$f_k = \frac{\sin\frac{(2\cdot k+1)\cdot\pi\cdot x}{2\cdot L}\times\cos\frac{(2\cdot k+1)\cdot\pi\cdot t}{2\cdot L}}{(2\cdot k+1)^2},$$

где $k=2, x=0.22\pm0.01,\ t=0.87\pm0.02, L=6.33\pm0.02$.

9. $$V=\sqrt{\frac{g\cdot\lambda}{2\cdot\pi}+\frac{2\cdot\pi\cdot\alpha}{\lambda\cdot p}},$$

где $g=9.81\pm0.01,\ \lambda=2.32\pm0.02,\ \alpha=7,\ p=7.82\pm0.01$.

10. $$f=\arccos\frac{a_1\cdot b_1+a_2\cdot b_2}{\sqrt{a_1^2+a_2^2}\times\sqrt{b_1^2+b_2^2}},$$

где $a_1=1.20\pm0.02,\ a_2=5.6,\ b_1=3.42\pm0.01,\ b_2=7.82\pm0.02$.

11. $$T=\frac{2\cdot\pi}{\sqrt{\frac{1}{L\cdot C}-\left[\frac{R}{2\cdot L}\right]^2}},$$

где $L=93.3\pm0.01,\ C=0.50\pm0.02,\ R=\ 2.33\pm0.02\ \times10^{-2}$.

12. $$I=I_0\times\frac{\sin^2\ \pi\cdot\alpha/\lambda\times\sin\beta}{\pi\cdot\alpha/\lambda\times\sin\beta\ ^2},$$

где $I_0=2.3,\ \beta=0.302\pm0.032,\ \alpha=0.45\pm0.001,\ \lambda=11.5\pm0.2$.

13. $$\omega=\frac{Q\cdot j^2-R}{\sqrt{j^2+Q^2\cdot\ j^2-1}},$$

где $Q=1.23\pm0.01,\ j=4.56\pm0.02,\ R=-\ 31.1\pm0.1\ \times10^{-1}$.

14. $$E=-\frac{2\cdot Q\cdot R}{\left[v^2+h^2\right]^{3/2}},$$

где $Q=12.2\pm0.1,\ R=5.7\pm0.5,\ h=6.23\pm0.02,\ v=\ 1.33\pm0.01\ \times10^3$.

15. $$y=\frac{e_0}{R}\times\left[1-e^{\frac{-R}{L}\cdot t}\right],$$

где $e_0=2.34\pm0.01,\ R=5.60\pm0.02,\ L=12,\ t=2.12\pm0.02$.

Глава 2
ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ПОДХОД К ПРОГРАММИРОВАНИЮ МЕТОДОВ ЛИНЕЙНОЙ АЛГЕБРЫ

Прямые методы линейной алгебры – это методы решения систем линейных алгебраических уравнений (СЛАУ) и методы решения проблемы собственных значений. Известно, что прямые методы теоретически базируются на тех или иных элементарных матричных преобразованиях, которые, в конечном счете, приводят задачу к эквивалентной, но более простой форме, допускающей ее непосредственное решение [21]. Набор таких преобразований сравнительно невелик. Исходя из технологии объектно-ориентированного программирования (ООП) эти преобразования могут быть представлены матричными операциями с объектами особого рода – абстрактными матрицами [21]. Важной методологической составляющей реализации технологии ООП в курсе изучения вычислительных методов алгебры является определение необходимого набора таких матриц и представление их классами единой матричной иерархии, наряду с основными функциональными матрицами. Далее все прямые методы линейной алгебры рассматриваются как композиции матричных преобразований. При этом их программная реализация основывается на организации обобщенного суперкласса вычислительных методов, применимого ко всем конкретным типам задач линейной алгебры.

В рамках предмета «Вычислительные методы алгебры» иерархия матричных классов может выглядеть следующим образом.

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

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.

Класс «Vector» представлен тремя основными классами: «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. В группе матриц «EMatrix» 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.

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

Класс «AugmentMatrix» представляет собой группу расширенных матриц.

Создание конкретного матричного класса в рамках ООП подхода сводится к реализации набора операций с элементарными матрицами [1; 21]. Этот набор определяется родительским классом «AbstractMatrix». Для основных матричных объектов виртуально определяются необходимые операции – получения и установки значения элемента, подсчета количества строк и столбцов матрицы, сложение, вычитание, умножение матриц. Операции матричного умножения определяются в двух вариантах, которые соответствуют левостороннему и правостороннему умножению.

Значительная часть операций непосредственно реализуется в верхних матричных классах. Тем не менее, переопределение их в конкретных классах с учетом частных математических свойств, особенностей конкретных структур данных позволяет при необходимости добиться наибольшей эффективности программного кода.

Представленная выше классификация элементарных матриц определяет базовый набор матричных преобразований, участвующих в

методах линейной алгебры. Описанная классификация не является минимальной. Для более экономичной работы с памятью и эффективной организации вычислений целесообразно определять некоторые самостоятельные классы, соответствующие целым композициям тех или иных элементарных преобразований. Использование классов такого рода позволяет избежать затрат на избыточное хранение эквивалентных данных и оптимизировать сами процедуры матричных преобразований.

Другая возможность экономии памяти при проведении и хранении необходимых матричных преобразований состоит в размещении их непосредственно на месте основной матрицы. Техника in-place размещения применяется при процедурной реализации матричных факторизаций. При объектной реализации следует предусмотреть специальные классы элементарных матриц в рамках единой иерархии, которые через ссылку на основную матрицу определяют методы доступа к ее сегментам, хранящим элементы преобразования [15; 21].

Для эффективного применения ООП необходимо также сами методы линейной алгебры рассматривать как реализацию объектов соответствующих классов в их целостной объектной классификации [21]. При этом классификация объектно-ориентированного подхода должна следовать классификации задач линейной алгебры, т.к. результатом решения различных постановок являются объекты разных типов, и это 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 в их программной реализации.

В силу сделанных замечаний объектная классификация алгоритмов линейной алгебры может быть представлена схемой, в которой все алгоритмы линейной алгебры объединены в базовый класс «VMA». В рамках этого класса «VMA» происходит классификация на алгоритмы факторизации «FactorizationAlgorithms», основные алгоритмы решения систем линейных алгебраических уравнений «SLAU» и алгоритмы проблем собственных значений «Eigenvalues».

Объектная классификация методов факторизации представлена наиболее важными группами методов LU, LDU и QR-разложения произвольной матрицы, а также S^TS и S^TDS-разложений для симметричных матриц.

Принципы построения всех алгоритмических классов очень близки. В группах алгоритмов «SLAU» и «Eigenvalues» выделены классы «DirectMethods», «DirectMethodsE», «IterationMethods», «IterationMethodsE» для реализации прямых и итерационных алгоритмов

соответственно. При организации этих классов в схеме предусматривается выделение основных изучаемых методов.

Описанная условная иерархия выглядит следующим образом.

- VMA
  - FactorizationAlgorithms
    - LU_decomposition
    - LDU_decomposition
    - STS_decomposition
    - SDS_decomposition
    - QR_decomposition
  - Eigenvalues
    - DirectMethodsE
      - danilevskyMethod
    - IterationMethodsE
      - PartialProblem
        - iterationMethod
      - CompleteProblem
        - jacobiMethod
        - qrMethod
  - SLAU
    - DirectMethods
      - DirectMethodsFactorization
        - gaussMethod
        - gaussChooseElement
        - gaussJordanMethod
        - squareMethod
      - DirectMethodsNF
        - sweepMethod
    - IterationMethods
      - simpleIterationMethod
      - zeidelMethod
      - methodSkorSpusk

В группе «SLAU» прямые алгоритмы решения систем линейных алгебраических уравнений «DirectMethods» в свою очередь подразделяются на алгоритмы непосредственного решения систем линейных алгебраических уравнений «DirectMethodsFactorization» (Прямые методы, использующие факторизацию) и «DirectMethodsNF» (Прямые методы, не использующие факторизацию).

Класс методов «DirectMethodsFactorization» предусматривает реализацию «gaussMethod» метода Гаусса, метода Гаусса с выбором главного элемента «gaussChooseElement», метода Жордана-Гаусса «gaussJordanMethod» и метода квадратного корня «squareMethod».

Класс методов «DirectMethodsNF» содержит метод прогонки «sweepMethod».

Класс итерационных методов решения алгебраических уравнений «IterationMethods» предусматривает реализацию метода простой итерации «simpleIterationMethod», метода Зейделя «zeidelMethod» и метода скорейшего спуска «methodSkorSpusk».

В классе «DirectMethodsE» («Прямые методы нахождения собственных значений») реализуется метод Данилевского «danilevskyMethod» для решения полной проблемы нахождения собственных значений и собственных векторов матрицы.

От класса «Итерационные методы нахождения собственных значений» («IterationMethodsE») наследуется класс «CompleteProblem» (Полная проблема нахождения собственных значений). В данном классе реализуются «qrMethod» (QR-алгоритм) и «jacobiMethod» (метод Якоби) для нахождения собственных значений произвольной квадратной матрицы.

Класс «PartialProblem» (Частичная проблема нахождения собственных значений) наследуется от класса «IterationMethodsE» (Итерационные методы нахождения собственных значений). В данном классе реализован «iterationMethod» итерационный метод нахождения наибольшего по модулю собственного значения и соответствующего собственного вектора.

Разработанный ООП-подход к программированию вычислительных методов алгебры представлен в данном пособии в виде заданий к 16 лабораторным работам.

П 2.1 Создание матричной иерархии классов

Иерархия классов для работы с матрицами представлена на схеме:

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

Рис. 1. **Объектная классификация абстрактных матриц.**

Вершиной матричной иерархии является абстрактный класс «AbstractMatrix». Абстрактные классы предназначены для представления общих понятий, которые предполагается конкретизировать в производных классах. В данном классе описывается общая структура класса без поведения, которая потом будет конкретизирована и дополнена в производных. Класс «AbstractMatrix» воплощает наиболее общие черты разрабатываемого семейства классов.

Потомками абстрактного класса являются классы: «Vector» (Вектор), «SquareMatrix» (Квадратная матрица) и «AugmentMatrix» (Расширенная матрица). 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 двух основных классов: «DiagonalMatrix» (Диагональная 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, содержащих ненулевые элементы на трех диагоналях: главной, первой сверху и первой снизу.

Класс «EMatrix» содержит методы для работы с единичной матрицей. На основе класса «EMatrix» определяется класс «SwapMatrix» – класс матриц перестановок. Матрица перестановок образуется из единичной матрицы изменением порядка расположения строк. Данный матричный класс соответствует преобразованиям переупорядочения столбцов и строк в основной матрице, которые обычно применяются в методах выбора главного элемента.

Класс «SquareMatrix» предназначен для реализации семейства элементарных квадратных матриц. Потомками этого класса являются – класс «FrobeniusMatrix» (Матрица Фробениуса), класс «JacobiMatrix» (Матрица Якоби), которые необходимы для решения задач на собственные значения.

Класс «AugmentMatrix» представляет собой группу расширенных матриц.

Классы, находящиеся ближе к началу иерархии, объединяют в себе наиболее общие черты для всех нижележащих классов. По мере продвижения вниз по иерархии классы приобретают все больше конкретных черт. Таким образом, организованное множественное наследование позволяет одному классу обладать свойствами двух и более родительских классов [1].

Пример 1. Реализовать следующую иерархию классов на языке программирования C++.

- AbstractMatrix
  - Vector
    - DiagonalMatrix

Исходя из того, что класс «AbstractMatrix» является вершиной иерархии, то в данном классе следует описать следующие виртуальные функции: получение значения элемента (getElement), установка значения элемента (setElement), получение количества строк (getRowCount) и получение количества столбцов (getColCount). Виртуальная функция – это функция, объявленная ключевым словом virtual в базовом классе и переопределенная в одном или нескольких производных классах. Виртуальные функции являются особыми функциями. Во время исполнения программы компилятор при вызове объекта производного класса с помощью указателя или ссылки на него определяет, какую функцию вызвать, основываясь на типе объекта. Для разных объектов будут вызываться разные версии одной и той же виртуальной функции [1].

Введем в классе «AbstractMatrix» целочисленное поле size, обозначающее количество строк матрицы. Данная размерность является общей для предложенной иерархии. Получаем реализацию класса «AbstractMatrix».

```
class AbstractMatrix{
      protected:
       int size;  /*Количество строк матрицы*/
      public:
       /* Получение значения элемента */
       virtual double getElement(int i, int j) = 0;
       /* Установка значения элемента*/
       virtual void setElement(int i, int j, double element)=0;
       /* Получение количества столбцов */
       virtual   int getColCount() = 0;
       /* Получение количества строк */
       virtual   int getRowCount() = 0;
};
```

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

```
class Vector : public AbstractMatrix{
      double *elements;    /* Элементы вектора*/
      public:
```

```
    /*Конструкторы*/
    /*Конструктор по умолчанию*/
     Vector();
    /* Конструкторы с параметрами*/
     Vector(int size);
     Vector(int size, double *elements);
     . . . . . .
    /* Получение значения элемента вектора*/
     double getElement(int i);
    /* Установка значения элемента вектора*/
     void setElement(int i, double element);
     /* Получение количества строк */
     int getRowCount();
    . . . . . .
};
```

Класс «DiagonalMatrix» является потомком класса «Vector». В данном классе переопределяются виртуальные методы абстрактного класса и добавляются свои методы для работы с диагональными матрицами. Не стоит забывать, что конструкторы не наследуются, поэтому производный класс должен иметь собственные конструкторы.

Реализация класса «DiagonalMatrix» выглядит следующим образом:

```
class DiagonalMatrix : public Vector{
    public:
    /* Вектор элементов на диагонали*/
     Vector elements;
    /*Конструкторы*/
     DiagonalMatrix();
     DiagonalMatrix(int size);
     DiagonalMatrix(int size, double *elements);
     . . . . . .
    /* Получение значения элемента  */
     double getElement(int i, int j);
    /* Установка значения элемента */
     void setElement(int i, int j, double element);
    /* Получение количества столбцов */
     int getColCount();
    /* Получение количества строк    */
     int getRowCount();
     . . . . . .
};
```

Пример 2. Реализовать в классе «Vector» конструкторы и следующие методы класса: получение значения элемента (getElement), установка значения элемента (setElement), получение количества строк (getRowCount) и получение количества столбцов (getColCount).

Внутри методов-членов класса ссылаться на объект класса с помощью неявно передаваемого указателя this.

Реализация на языке программирования C++ может иметь следующий вид:

```
/*Параметризованные конструкторы*/
Vector :: Vector(int size){
/* Внутри методов-членов класса идентификаторы относятся к тому объекту, для которого функции были вызваны. На этот объект внутри функций-членов класса можно ссылаться с помощью неявно передаваемого указателя this.
*/
        this->size = size;
        this->elements = new double[this->size];
}

Vector :: Vector(int size, double *elements){
       /*Создание матрицы*/
        this->size = size;
        this->elements = new double[this->size];
       /*Заполнение матрицы*/
        for (int i=0; i < this->size; i++){
          this->elements[i] = elements[i];
       }
}

/* Получение значения элемента  */
double Vector :: getElement(int i){
        return this->elements[i];
}

/* Установка значения элемента */
void Vector :: setElement(int i, double element){
        this->elements[i] = element;
}

/* Получение количества строк  */
int Vector :: getRowCount(){
        return 1;
}

/* Получение количества столбцов */
int Vector :: getColCount(){
        return this->size;
}
```

Пример 3. Реализовать метод для сложения двух квадратных матриц двумя способами: с помощью функции и перегруженного оператора.

Реализация метода на языке программирования C++ может иметь следующий вид:

```
/* Функция*/
SquareMatrix SquareMatrix :: addition(SquareMatrix A, SquareMatrix B){
  int n = this->size;
  /*Объявление результирующей матрицы*/
      double **C = new double*[n];
      for (int i=0; i < n; i++){
        C[i] = new double[n]; }
 /* Вычисление суммы матриц*/
      for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
         C[i][j] = A.elements[i][j] + B.elements[i][j];
                  } }
         /* Создание матрицы */
        return SquareMatrix(n, C);
       }

/* Перегруженный оператор сложения */
SquareMatrix SquareMatrix :: operator +(SquareMatrix B){
   int n = this->size;
  /* Объявление результирующей матрицы*/
   double **C = new double*[n];
   for (int i=0; i < n; i++){
           C[i] = new double[n];
     }
  /* Вычисление суммы матриц*/
   for (int i = 0; i < n; i++){
     for (int j = 0; j < n; j++){
    C[i][j]=this->elements[i][j] + B.elements[i][j];
           }
     }
  /* Создание матрицы */
     return SquareMatrix(n, C);
   }
```

Лабораторная работа № 2

Цель: изучить иерархию классов. Реализовать иерархию классов для работы с матрицами.

Задание

Реализуйте иерархию матричных классов, приведенную на рис. 1.

Реализуйте конструкторы, функции, методы, необходимые для работы с матрицами.

Для класса «Vector» реализуйте методы сложения, вычитания векторов, скалярного произведения, умножения вектора на число, нахождения длины вектора.

Для класса «SquareMatrix» реализуйте методы: сложение матриц, вычитание матриц, перемножение двух квадратных матриц, перемножение матрицы на вектор, умножение матрицы на число, транспонирования матрицы. Для реализации данных методов можно использовать перегруженные операторы. Реализовать методы для нахождения первой и второй нормы матрицы.

В классе «AugmentMatrix» предусмотрите методы формирования расширенной матрицы: из квадратной матрицы и вектора, из двух квадратных матриц. Реализуйте метод умножения расширенной матрицы на квадратную матрицу.

Ввод матриц организовать двумя способами: с клавиатуры и из файла. Реализовать метод для файлового ввода симметричной матрицы, а именно только ее верхней треугольной части.

Предусмотреть два режима вывода матриц: на экран и в файл.

Продумать и реализовать методы работы с матрицами классов «DiagonalMatrix», «ThreeDiagonalMatrix» и «EMatrix». С целью экономии памяти данные классы хранят только значения ненулевых коэффициентов.

Доступ к элементам осуществить при помощи «геттеров» и «сеттеров».

П 2.2 Создание иерархии классов вычислительных методов алгебры

Рассмотрим иерархию классов вычислительных методов алгебры следующего вида:

- VMA
  - FactorizationAlgorithms
    - LU_decomposition
    - LDU_decomposition
    - STS_decomposition
    - SDS_decomposition
    - QR_decomposition
  - Eigenvalues
    - DirectMethodsE
      - danilevskyMethod
    - IterationMethodsE
      - PartialProblem
        - iterationMethod
      - CompleteProblem
        - jacobiMethod
        - qrMethod

- SLAU
  - DirectMethods
    - DirectMethodsFactorization
      - gaussMethod
      - gaussChooseElement
      - gaussJordanMethod
      - squareMethod
    - DirectMethodsNF
      - sweepMethod
  - IterationMethods
    - simpleIterationMethod
    - zeidelMethod
    - methodSkorSpusk

Рис. 2. **Объектная классификация методов линейной алгебры.**

Вершиной иерархии является базовый класс «VMA».

Класс «FactorizationAlgorithms» содержит методы факторизации.

Класс «Eigenvalues» предназначен для реализации численных методов нахождения собственных значений и собственных векторов матриц. Методы этого класса подразделяются на прямые – «DirectMethodsE» и итерационные – «IterationMethodsE». Класс «IterationMethodsE» разбивается на два производных класса: итерационные методы, применяемые к решению частичной проблемы нахождения собственных значений, им соответствует класс «PartialProblem» и итерационные методы, применяемые к решению полной проблемы нахождения собственных значений – «CompleteProblem».

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».

Среди алгоритмов факторизации выделены пять методов: LU, LDU, S^TS, S^TDS и QR-разложение.

LU-разложение – это представление квадратной матрицы A в виде произведения нижней треугольной матрицы L на верхнюю треугольную матрицу U, т.е. $A=LU$. Данное разложение возможно в случае, когда все угловые миноры матрицы A отличны от нуля, и является единственным, если заранее оговорены элементы главной диагонали треугольных матриц [19].

Пусть L – нижняя треугольная матрица с единичной диагональю, а U – верхняя треугольная матрица с ненулевыми диагональными элементами. Значения элементов матрицы L и U находятся по рекуррентным соотношениям:

$$l_{ii} = 1,$$

$$l_{ik} = \frac{a_{ik} - \sum_{j=1}^{k-1} l_{ij} u_{jk}}{u_{kk}} \quad при \quad k < i, \qquad (1)$$

$$u_{ik} = a_{ik} - \sum_{j=1}^{i-1} l_{ij} u_{jk} \quad ïðè \quad i \le k, \quad i,k = 1,\ 2,\ ....,\ n.$$

LU-разложение, полученное по формулам (1), применяется для построения LDU-разложения.

Пример 1. Для заданной матрицы $A = \begin{pmatrix} 2 & 1 & 4 \\ 3 & 2 & 1 \\ 1 & 3 & 3 \end{pmatrix}$ получить LU-разложение.

Решение:

Воспользуемся рекуррентными формулами (1)

$$u_{11} = a_{11} = 2; \quad u_{12} = a_{12} = 1; \quad u_{13} = a_{13} = 4;$$

При к=1:

$$l_{21} = \frac{a_{21}}{u_{11}} = \frac{3}{2}; \quad l_{31} = \frac{a_{31}}{u_{11}} = \frac{1}{2};$$

$$u_{22} = a_{22} - l_{21} u_{12} = 2 - \frac{3}{2} \cdot 1 = 0{,}5;$$

При к=2:

$$l_{32} = \frac{a_{32} - l_{31} \cdot u_{12}}{u_{22}} = \frac{3 - 1 \cdot 0{,}5}{0.5} = 5;$$

$$u_{23} = a_{23} - l_{21} \cdot u_{13} = 1 - \frac{3}{2} \cdot 4 = -5;$$

При к=3: $u_{33} = a_{33} - l_{31} \cdot u_{13} - l_{32} \cdot u_{23} = 3 - \frac{1}{2} \cdot 4 - 5 \cdot (-5) = 26.$

В результате получены две треугольные матрицы:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{3}{2} & 1 & 0 \\ \frac{1}{2} & 5 & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} 2 & 1 & 4 \\ 0 & \frac{1}{2} & -5 \\ 0 & 0 & 26 \end{pmatrix}.$$

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.

$$L=\begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix}, U=\begin{pmatrix} 1 & \frac{u_{12}}{u_{11}} & \cdots & \frac{u_{1n}}{u_{11}} \\ 0 & 1 & \cdots & \frac{u_{2n}}{u_{22}} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}, D=\begin{pmatrix} u_{11} & 0 & \cdots & 0 \\ 0 & u_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix}.$$

Элементы l_{ij}, u_{ij}, i, $j=1, 2, ..., n$ рассчитываются по формулам (1).

Пример 2. Дана матрица $A=\begin{pmatrix} 2 & 1 & 4 \\ 3 & 2 & 1 \\ 1 & 3 & 3 \end{pmatrix}$. Получить ее LDU-разложение.

Решение:

В предыдущем примере получены матрицы L и U вида:

$$L=\begin{pmatrix} 1 & 0 & 0 \\ \frac{3}{2} & 1 & 0 \\ \frac{1}{2} & 5 & 1 \end{pmatrix}, \quad U=\begin{pmatrix} 2 & 1 & 4 \\ 0 & \frac{1}{2} & -5 \\ 0 & 0 & 26 \end{pmatrix}.$$

Матрица L не изменяется. Матрицу D выделим из матрицы U. Получаем решение:

$$L=\begin{pmatrix} 1 & 0 & 0 \\ \frac{3}{2} & 1 & 0 \\ \frac{1}{2} & 5 & 1 \end{pmatrix}, \quad D=\begin{pmatrix} 2 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 26 \end{pmatrix}, \quad U=\begin{pmatrix} 1 & \frac{1}{2} & 2 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{pmatrix}.$$

S^TDS-разложение – это представление симметрической 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.

$$S=\begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ 0 & s_{22} & \cdots & s_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & s_{nn} \end{pmatrix}, \quad D=\begin{pmatrix} d_{11} & 0 & \cdots & 0 \\ 0 & d_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & d_{nn} \end{pmatrix}.$$

Значения коэффициентов данных матриц находятся по формулам [19]

$$\begin{aligned} & s_{11}=\sqrt{|a_{11}|}, \quad s_{1j}=\frac{a_{1j}}{d_{11}s_{11}}, \quad j=2,\ 3,\ \ldots,\ n, \\ & s_{jj}=\sqrt{\left|a_{jj}-\sum_{k=1}^{i-1}d_{kk}|s_{kj}|^2\right|}, \ \ s_{ij}=0, \quad i>j, \\ & s_{ij}=\frac{a_{ij}-\sum_{k=1}^{i-1}s_{ki}s_{kj}d_{kk}}{s_{ii}d_{ii}}, \quad i<j, \\ & d_{11}=sign a_{11}, \\ & d_{ii}=sign\left(a_{ii}-\sum_{k=1}^{i-1}|s_{ki}|^2 d_{kk}\right), \ \ i>1. \end{aligned} \qquad (2)$$

S^TS-разложение – это представление симметрической 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).

Для положительной определенности матрицы достаточно выполнения требования положительности диагональных элементов и их диагонального преобладания, т.е. $a_{ii}>0$, $a_{ii}>\sum\limits_{\substack{j=1 \\ j\neq i}}^{n}|a_{ij}|, \quad \forall i=1,\ 2,\ ...,\ n$

$$S=\begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ 0 & s_{22} & \cdots & s_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & s_{nn} \end{pmatrix}, \quad S^T=\begin{pmatrix} s_{11} & 0 & \cdots & 0 \\ s_{21} & s_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ s_{n1} & s_{n2} & \cdots & s_{nn} \end{pmatrix}.$$

Значения коэффициентов матрицы S находятся по формулам [19].

$$s_{11} = \sqrt{a_{11}}, \quad s_{1j} = \frac{a_{1j}}{s_{11}}, \quad j = 2,\ 3,\ \ldots,\ n,$$

$$s_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} s_{ki}^2}, \quad i = 2,\ 3,\ \ldots,\ n, \tag{3}$$

$$s_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} s_{ki} s_{kj}}{s_{ii}}, \quad i < j, \quad i = 2,\ 3,\ \ldots,\ j-1, \quad j = 2,\ \ldots,\ n.$$

Пример 3. Дана матрица $A = \begin{pmatrix} 2.65 & -0.52 & 1.02 \\ -0.52 & 2.53 & 0.46 \\ 1.02 & 0.46 & 2.16 \end{pmatrix}$. Получить ее S^TS-разложение.

Решение:

Матрица A является симметрической и положительно определенной. Воспользуемся формулами (2), получаем:

$$s_{11} = \sqrt{a_{11}} = \sqrt{2.65} = 1.628,$$

$$s_{12} = \frac{a_{12}}{s_{11}} = \frac{-0.52}{1.628} = -0.319,$$

$$s_{13} = \frac{a_{12}}{s_{11}} = \frac{1.02}{1.628} = 0.627,$$

$$s_{22} = \sqrt{a_{22} - s_{12}^2} = \sqrt{2.53 - (-0.319)^2} = 1.558,$$

$$s_{23} = \frac{a_{23} - (s_{12} \cdot s_{13})}{s_{22}} = \frac{0.46 - (-0.319) \cdot 0.627}{1.558} = 0.424,$$

$$s_{33} = \sqrt{a_{33} - (s_{13}^2 + s_{23}^2)} = \sqrt{2.16 - (0.627^2 + 0.424^2)} = 1.26.$$

Таким образом, получили матрицу $S = \begin{pmatrix} 1.628 & -0.319 & 0.627 \\ 0 & 1.558 & 0.424 \\ 0 & 0 & 1.26 \end{pmatrix}$.

QR-разложение – это представление квадратной матрицы A в виде произведения ортогональной матрицы Q на верхнюю треугольную матрицу R.

QR-разложение может быть получено различными методами. Рассмотрим построение QR-разложения с помощью преобразования Хаусхолдера, которое позволяет обратить в нуль группу поддиагональных элементов столбца матрицы.

Преобразование Хаусхолдера осуществляется с использованием матрицы Хаусхолдера (матрица отражения), имеющей следующий вид [20]:

$$H = E - 2ww^T, \tag{4}$$

где w – произвольный ненулевой вектор-столбец, E – единичная матрица.

Любая матрица такого вида является симметрической и ортогональной.

Рассмотрим подробнее реализацию данного преобразования. Положим $A_0 = A$ и построим преобразование Хаусхолдера H_1, переводящее матрицу A_0 в матрицу A_1 с нулевыми элементами первого столбца под главной диагональю

$$A_0 = \begin{pmatrix} a_{11}^0 & a_{12}^0 & \dots & a_{1n}^0 \\ a_{21}^0 & a_{22}^0 & \dots & a_{2n}^0 \\ \dots & \dots & \dots & \dots \\ a_{n1}^0 & a_{n2}^0 & \dots & a_{nn}^0 \end{pmatrix} \xrightarrow{H_1} A_1 = \begin{pmatrix} a_{11}^1 & a_{12}^1 & \dots & a_{1n}^1 \\ 0 & a_{22}^1 & \dots & a_{2n}^1 \\ \dots & \dots & \dots & \dots \\ 0 & a_{n2}^1 & \dots & a_{nn}^1 \end{pmatrix}.$$

Для этого построим вектор [20]:

$$w_1^T = \mu_1 \left(a_{11}^0 - s_1, a_{21}^0, \dots, a_{n1}^0 \right), \qquad (5)$$

где $s_1 = \left(\sum_{j=2}^{n} \left(a_{j2}^0 \right)^2 \right)^{1/2}$, $\mu_1 = \left(2 s_2 \left(s_2 - a_{22}^0 \right) \right)^{-1/2}$.

Матрица Хаусхолдера строится согласно формуле (4)

$$H_1 = E - 2 w_1 w_1^T .$$

Тогда легко проверить, что

$$A_1 = H_1 A_0 = \left(\mathrm{E} - 2 w_1 w_1^T \right) A_0 = \begin{pmatrix} a_{11}^1 & a_{12}^1 & \dots & a_{1n}^1 \\ 0 & a_{22}^1 & \dots & a_{2n}^1 \\ \dots & \dots & \dots & \dots \\ 0 & a_{n2}^1 & \dots & a_{nn}^1 \end{pmatrix}. \qquad (6)$$

Для того чтобы обнулить поддиагональные элементы второго столбца матрицы A_1, выберем вектор w_2 так, что

$$w_2^T = \mu_2 \left(0, a_{22}^1 - s_2, a_{32}^1, \dots, a_{n2}^1 \right),$$

где $s_2 = \left(\sum_{j=2}^{n} \left(a_{j2}^1 \right)^2 \right)^{1/2}$, $\mu_2 = \left(2 s_2 \left(s_2 - a_{22}^1 \right) \right)^{-1/2}$.

Тогда лежащие ниже главной диагонали элементы двух первых столбцов матрицы $A_2 = H_2 A_1 = H_2 H_1 A_0 = \left(E - 2w_2 w_2^T\right)\left(E - 2w_1 w_1^T\right) A_0$ обратятся в нуль.

Продолжая этот процесс с помощью векторов w_i, имеющих нули в первых *i-1* позициях, получаем:

$$H_{n-1}...H_2 H_1 A_0 = \left(E - 2w_{n-1} w_{n-1}^T\right) ... \left(E - 2w_2 w_2^T\right)\left(E - 2w_1 w_1^T\right) A_0 = R, \quad (7)$$

где R – верхняя треугольная матрица.

Положим

$$Q = H_1 H_2 ... H_{n-1} = \left(E - 2w_1 w_1^T\right)\left(E - 2w_2 w_2^T\right) ... \left(E - 2w_{n-1} w_{n-1}^T\right). \quad (8)$$

Тогда можно записать, что $Q^T A_0 = R$.

Так как каждая матрица $H_i = \left(E - 2w_i w_i^T\right)$ ортогональна, то и их произведение Q также будет ортогональной матрицей. Следовательно, $Q^{-1} = Q^T$.

Тогда имеет место соотношение $A = Q\,R$, которое и представляет собой QR-разложение матрицы *A*.

Пример 4. Дана матрица $A = \begin{pmatrix} 2 & 1 & 4 \\ 5 & 2 & 7 \\ 1 & 3 & 3 \end{pmatrix}$. Получить ее QR-разложение.

Решение:

Выберем вектор w_1 согласно формуле (5):

$$w_1 = \begin{pmatrix} \mu_1 \cdot a_{11} - s_1 \\ \mu_1 \cdot a_{21} \\ \mu_1 \cdot a_{31} \end{pmatrix},$$

где $\mu_1 = \left[2 \cdot s_1 \cdot (s_1 - a_{11})\right]^{-1/2}$, $s_1 = \left[\sum_{j=1}^{3} a_{j1}^{\,2}\right]^{1/2}$.

Тогда $w_1 = \begin{pmatrix} -0.563 \\ 0.81 \\ 0.162 \end{pmatrix}$.

Построим матрицу A_1 согласно формуле (6):

$$A_1 = H_1 A = \left(E - 2 \cdot w_1 \cdot w_1^T\right) A,$$

$$A_1 = \begin{pmatrix} 5.477 & 2.739 & 8.398 \\ 0 & -0.5 & 0.675 \\ 0 & 2.5 & 1.735 \end{pmatrix}.$$

Выберем вектор w_2 по формуле (5):

$$w_2 = \begin{pmatrix} \mu_2 \cdot 0 \\ \mu_2 \cdot \left(a_{22}^1 - s_2 \right) \\ \mu 2 \cdot a_{32}^1 \end{pmatrix},$$

где $\mu_2 = \left[2 \cdot s_2 \cdot \left(s_2 - a_{22}^1 \right) \right]^{-1/2}$, $s_2 = \left[\sum_{j=2}^{3} \left(a_{j2}^1 \right)^2 \right]^{1/2}$.

Тогда $w2 = \begin{pmatrix} 0 \\ -0{,}773 \\ 0.634 \end{pmatrix}$.

Построим матрицу A_2 по формуле (6):

$$A_2 = H_2 A = \left(E - 2 \cdot w_2 \cdot w_2^T \right) A_1,$$

$$A_2 = \begin{pmatrix} 5.477 & 2.739 & 8.398 \\ 0 & 2.55 & 1.569 \\ 0 & 0 & 1.003 \end{pmatrix}.$$

Построим матрицу Q согласно формуле (8):

$$Q = H_1 H_2 = \left(E - 2 \cdot w_1 \cdot w_1^T \right) \left(E - 2 \cdot w_2 \cdot w_2^T \right),$$

$$Q = \begin{pmatrix} 0.365 & 0 & 0.931 \\ 0.913 & -0.196 & -0.358 \\ 0.183 & 0.981 & -0.072 \end{pmatrix}.$$

Убедимся, что матрица Q – ортогональна.

$$Q^{-1} = \begin{pmatrix} 0.365 & 0.913 & 0.943 \\ 0 & -0.196 & 0.981 \\ 0.931 & -0.358 & -0.072 \end{pmatrix} = Q^T.$$

Построим верхнюю треугольную матрицу R по формуле (7):

$$R = H_2 H_1 A = \left(E - 2 \cdot w_2 \cdot w_2^T \right) \left(E - 2 \cdot w_1 \cdot w_1^T \right) A,$$

$$R = \begin{pmatrix} 5.477 & 2.739 & 8.398 \\ 0 & 2.55 & 1.569 \\ 0 & 0 & 1.003 \end{pmatrix}.$$

Получаем решение:

$$Q = \begin{pmatrix} 0.365 & 0 & 0.931 \\ 0.913 & -0.196 & -0.358 \\ 0.183 & 0.981 & -0.072 \end{pmatrix}, \quad R = \begin{pmatrix} 5.477 & 2.739 & 8.398 \\ 0 & 2.55 & 1.569 \\ 0 & 0 & 1.003 \end{pmatrix}.$$

Пример 5. Реализовать алгоритм S^TS-разложения для произвольной симметрической матрицы.

При описании алгоритма используются объекты класса «SquareMatrix» и методы данного класса: getRowCount (количество строк матрицы), getElement (получение значения элемента) и setElement (установка значения). Реализация примера на языке программирования C++ может иметь следующий 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));
                                                          }
                              }
```

Пример 6. Реализовать следующую иерархию классов.

- VMA
- FactorizationAlgorithms
  - LU_decomposition
  - LDU_decomposition
  - STS_decomposition
  - SDS_decomposition
  - QR_decomposition
- Eigenvalues
  - DirectMethodsE

- IterationMethodsE
- SLAU
  - DirectMethods
  - IterationMethods

Реализуем пример на языке программирования C++

```
 /* Базовый класс «ВМА» */
class VMA{
      public:
           VMA(){};
};

/*Класс «Алгоритмы факторизации»  */
class FactorizationAlgorithms : public VMA{
      public:
           FactorizationAlgorithms(){};
   /* LU разложение   */
void LU_decomposition(SquareMatrix A, SquareMatrix LU);
   /* LDU разложение*/
void LDU_decomposition (SquareMatrix A, SquareMatrix L, DiagonalMatrix D, Squa-
reMatrix U);
   /* STS разложение*/
void STS_decomposition (SquareMatrix A, SquareMatrix S);
   /* SDS разложение */
void SDS_decomposition (SquareMatrix A, SquareMatrix S, SquareMatrix D);
   /* QR разложение */
void QR_decomposition (SquareMatrix A, SquareMatrix Q, SquareMatrix R);
};

/* Класс «Собственные значения» */
class Eigenvalues : public VMA{
      public:
        Eigenvalues (){};
};

/* Класс «Прямые методы нахождения собственных значений»*/
class DirectMethodsE : public Eigenvalues {
      public:
           DirectMethodsE(){};
};

/* Класс «Итерационные методы нахождения собственных значений»*/
class IterationMethodsE : public Eigenvalues {
      public:
           IterationMethodsE (){};
};

/*   Класс «Системы линейных алгебраических уравнений» */
class SLAU : public VMA{
      public:
```

```
        SLAU(){};
};

/*   Класс  «Прямые методы решения СЛАУ»    */
class DirectMethods : public SLAU{
    public:
        DirectMethods(){};
};

/* Класс «Итерационные методы решения СЛАУ»  */
class IterationMethods : public SLAU{
    public:
        IterationMethods(){};};
```

Лабораторная работа № 3

Цель: изучить иерархию классов, представленную на рис. 2. Реализовать иерархию классов вычислительных методов алгебры. Изучить методы факторизации. Реализовать методы для LU-, LDU-, S^TDS - , S^TS и QR-разложения.

Задание

Разработайте базовый класс «VMA» с производными от него классами – «Алгоритмы факторизации» («FactorizationAlgorithms»), «СЛАУ» («SLAU»), «Собственные значения» («Eigenvalues»).

От класса «Eigenvalues» наследуйте 2 класса – «Прямые методы нахождения собственных значений» («DirectMethodsE») и «Итерационные методы нахождения собственных значений» («IterationMethodsE»).

От класса «СЛАУ» наследуйте 2 класса – «Прямые методы решения СЛАУ» («DirectMethods») и «Итерационные методы решения СЛАУ» («IterationMethods»).

В классе «FactorizationAlgorithms» реализуйте методы для LU-, LDU-, S^TS-, S^TDS и QR-разложения (LU_decomposition, LDU_decomposition, STS_decomposition, SDS_decomposition, QR_decomposition).

Для работы с матрицами используйте объекты матричного класса «SquareMatrix» и методы, реализованные в данном классе. Доступ к элементам осуществляется с помощью методов getElement и setElement.

Глава 3
РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

В главе 3 рассматриваются методы решения систем линейных алгебраических уравнений. К решению таких систем приводят многие прикладные задачи. Наряду с проблемой решения неоднородной системы линейных алгебраических уравнений в данной главе будет изучена проблема обращения матриц, а также задача вычисления определителя матрицы.

Линейная неоднородная задача для систем линейных алгебраических уравнений (СЛАУ) записывается в виде [19; 8]

$$AX = f \quad \text{или} \quad \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \dots \\ f_n \end{pmatrix}, \qquad (1)$$

где $A = a_{ij} \in R^{n \times n}$ – действительная матрица размера ($n*n$), i, j – переменные, соответствующие номерам строк и столбцов (целые числа); $f = (f_1, \dots, f_n)^T \in R^n$ – вектор-столбец, $X = (x_1, \dots, x_n)^T \in R^n$ – вектор-столбец неизвестных, R^n – n-мерное евклидово пространство, верхний индекс «Т» обозначает операцию транспонирования.

Требуется найти решение $X^* = (x_1^*, \dots, x_n^*)^T \in R^n$ системы (1), подстановка которого в (1) приводит к верному равенству $AX^* = f$.

Из линейной алгебры известно, что решение задачи (1) существует и единственно, если детерминант матрицы A отличен от нуля, т.е. $\det A = |A| \neq 0$ (A – невырожденная матрица, называемая также неособенной). Если определитель матрицы системы равен нулю, то система уравнений либо не имеет решения, либо имеет их бесчисленное множество.

Характер задачи и точность получаемого решения в большой степени зависят от ее обусловленности, являющейся важнейшим математическим понятием, влияющим на выбор метода ее решения.

Обусловленность задачи определяется числом обусловленности $\nu(A) = \|A\| \cdot \|A^{-1}\|$, где $\|A\|$ – норма матрицы A, а $\|A^{-1}\|$ – норма обратной матрицы. Чем больше это число, тем хуже обусловленность системы.

Число обусловленности характеризует степень зависимости относительной погрешности решения СЛАУ от погрешности исходных данных. Пусть ΔA, Δf, ΔX – погрешности коэффициентов матрицы, правой части и вектора решений соответственно, тогда справедливы следующие неравенства [19]:

$$\frac{\|\Delta X\|}{\|X\|} \le \nu(A)\frac{\|\Delta f\|}{\|f\|}, \quad \frac{\|\Delta X\|}{\|X\|} \le \nu(A)\frac{\|\Delta A\|}{\|A+\Delta A\|}.$$

Таким образом, чем больше число обусловленности, тем сильнее влияние погрешности исходных данных на конечный результат. Матрица с большим числом обусловленности $\nu(A) >> 1$ называется плохо обусловленной, а системы уравнений с такой матрицей называются плохо обусловленными системами.

При решении СЛАУ на компьютере из-за погрешностей округлений не всегда удается получить точное равенство определителя нулю, т.е. можем получить $\det A \approx 0$. В этом случае малые погрешности вычислений могут привести к существенным погрешностям в решении, как в случае плохо обусловленной системы.

Поясним связь величины определителя матрицы системы с понятием обусловленности на примере двумерной задачи [8].

Имеем систему из двух линейных уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = f_1, \\ a_{21}x_1 + a_{22}x_2 = f_2. \end{cases}$$

Точным решением этой задачи является вектор $X^* = (x_1^*, x_2^*)^T$, компоненты которого определяются координатами точки пересечения двух прямых, соответствующих уравнениям (см. рис. 1).

$a_{11}x_1 + a_{12}x_2 = f_1,$
$a_{21}x_1 + a_{22}x_2 = f_2.$

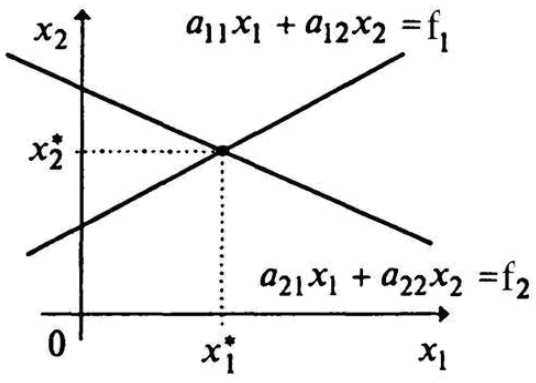


Рис. 1. **Точное решение.**

Если коэффициенты системы или вектор $f = (f_1, f_2)^T$ заданы с некоторыми погрешностями, то можем получить три системы линейных уравнений с различным характером обусловленности.

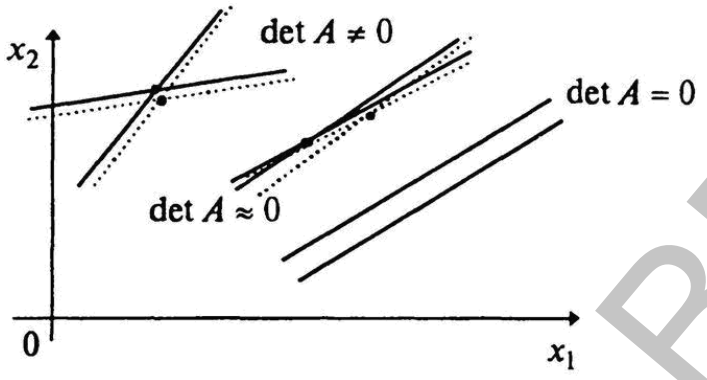


Рис. 2. **Характер обусловленности систем.**

Рис. 2 является иллюстрацией возможных вариантов. Если *det A* существенно отличен от нуля, то точка пересечения пунктирных прямых, смещенных относительно сплошных прямых из-за погрешностей задания *A* и *f*, сдвигается несильно. Это свидетельствует о хорошей обусловленности системы.

Если определитель матрицы близок к нулю ($\det A \approx 0$), то небольшие погрешности исходных данных могут привести к большим погрешностям в решении. На рис. 2 прямые системы уравнений близки к параллельным, координаты точки пересечения этих прямых весьма чувствительны к изменению коэффициентов системы.

При $\det A = 0$ прямые параллельны или совпадают, тогда решение задачи не существует или оно не единственно.

Условие $\det A \approx 0$ является необходимым для плохой обусловленности системы линейных уравнений, но не достаточным. Например, система уравнений n-го порядка с диагональной матрицей с элементами $a_{ii} = 0.1$ не является плохо обусловленной, хотя ее определитель мал $\det A = 10^{-n}$.

Методы решения систем линейных алгебраических уравнений можно разделить на две большие группы: прямые и итерационные.

Под прямыми (точными) понимаются такие методы, которые позволяют в предположении отсутствия округлений получить точные значения неизвестных как результат выполнения конечного числа арифметических операций. Результат таких методов может быть выражен аналитической формулой.

К прямым методам относят методы исключения неизвестных (метод Гаусса и его модификации), метод квадратного корня для решения СЛАУ.

В данной группе методов рассмотрим отдельно методы решения систем на основе факторизации, в основу которых положена идея разложения искомой матрицы в виде произведения других матриц специального вида, или матричных разложений (факторизаций). Как правило, после факторизации матрицы, задача решения системы линейных алгебраических уравнений существенно упрощается.

При большом числе уравнений прямые методы решения СЛАУ (за исключением метода прогонки) становятся труднореализуемыми, прежде всего из-за сложности хранения и обработки матриц большой размерности.

Методы, в которых при вычислении последующего приближения решения используются предыдущие, уже известные приближенные решения, называются итерационными. Для начала вычислений итерационные методы требуют задания одного или нескольких начальных приближений, которые задаются произвольно. Окончание итераций определяется либо заданием максимального числа итераций, либо условием $\Delta^{(k+1)} = \max\left|x_i^{(k+1)} - x_i^{(k)}\right| \le \varepsilon \quad i = 0, 1, \ldots$. Здесь $x_i^{(k)}$ – i-тая компонента вектора приближенного решения, k – номер итерации, $\varepsilon > 0$ – заданная точность для поиска решения.

Условия и скорость сходимости каждого итерационного метода существенно зависят от свойств матрицы системы.

К итерационным методам решения СЛАУ относятся метод простых итераций, метод Зейделя и др.

П 3.1 Метод Гаусса решения систем линейных алгебраических уравнений

Рассмотрим метод Гаусса. Имеем систему линейных алгебраических уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = f_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = f_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = f_n. \end{cases} \qquad (1)$$

или в матричном виде $AX = f,$ (2)

где A – вещественная квадратная матрица порядка n, f – заданный и X – искомый векторы. Будем предполагать, что определитель матрицы отличен от нуля.

Метод Гаусса решения системы (1) реализуется в два этапа: прямой и обратный ход. Первый этап состоит в последовательном ис-

ключении неизвестных x_1, x_2, ..., x_n из системы (1). Предположим, что $a_{11} \neq 0$. Разделим первое уравнение на a_{11}, получаем:

$$x_1 + c_{12}x_2 + \ldots + c_{1n}x_n = g_1, \tag{3}$$

где $c_{1j} = \dfrac{a_{1j}}{a_{11}}, \quad j = 2, \ldots, n, \quad g_1 = \dfrac{f_1}{a_{11}}.$

Рассмотрим теперь оставшиеся уравнения системы (1):

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = f_i, \quad i = 2, 3, \ldots, n. \tag{4}$$

Умножим (3) на a_{i1} и вычтем полученное уравнение из *i*-ого уравнения системы (4). В результате получим следующую систему уравнений:

$$\begin{cases} x_1 + c_{12}x_2 + \ldots + c_{1n}x_n = g_1, \\ \quad a_{22}^{(1)}x_2 + \ldots + a_{2n}^{(1)}x_n = f_2^{(1)}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \quad a_{n1}^{(1)}x_1 + a_{n2}^{(1)}x_2 + \ldots + a_{nn}^{(1)}x_n = f_n^{(1)}, \end{cases} \tag{5}$$

где $a_{ij}^{(1)} = a_{ij} - c_{1j}a_{i1}, \quad f_i^{(1)} = f_i - g_1 a_{i1}, \quad i, j = 2, 3, \ldots, n.$ (6)

Тем самым осуществили первый шаг метода Гаусса и получили систему (5), в которой матрица системы имеет вид:

$$\begin{bmatrix} 1 & \times & \ldots & \times \\ 0 & \times & \ldots & \times \\ \ldots & \ldots & \ldots & \ldots \\ 0 & \times & \ldots & \times \end{bmatrix},$$

т.е. неизвестное x_1 содержится только в первом уравнении.

Если $a_{22}^{(1)} \neq 0$, то из системы (5) можно исключить неизвестное x_2 и получить систему, эквивалентную (1) с матрицей следующей структуры:

$$\begin{bmatrix} 1 & \times & \times. & \ldots\times \\ 0 & 1 & \times & \ldots\times \\ \ldots & \ldots & \ldots & \ldots\times \\ 0 & 0 & \times & \ldots\times \end{bmatrix}.$$

При этом первое уравнение полученной системы остается без изменения.

Исключая аналогичным образом неизвестные $x_3,\ x_4,\ ...,\ x_n$, придем окончательно к системе уравнений вида

$$\begin{cases} x_1 + c_{12}x_2 + \ldots + c_{1n}x_n = g_1, \\ \qquad x_2 + c_{23}x_3 + \ldots + c_{2n}x_n = g_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \qquad\qquad\qquad x_n = g_n. \end{cases} \tag{7}$$

Матрица системы (7) будет иметь вид верхней треугольной матрицы с единичными элементами на главной диагонали

$$\begin{bmatrix} 1 & \times & \times & \times & \times \\ 0 & 1 & \times & \times & \times \\ \ldots & \ldots & \ldots & 1 & \times \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Получение системы (7) составляет прямой ход метода Гаусса, в котором коэффициенты уравнений преобразуются по следующему правилу [18]:

$$\begin{gathered} a_{kj}^{(0)} = a_{kj},\ \ k,\ \ j = 1,\ 2,\ \ldots,\ n, \\ c_{kj} = \frac{a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad j = k+1,\ k+2,\ \ldots,\ n, \quad k = 1,\ 2,\ \ldots,\ n-1, \\ a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{ik}^{(k-1)} c_{kj}, \quad i,\ \ j = k+1,\ k+2,\ \ldots,\ n, \quad k = 1,\ 2,\ \ldots,\ n-1. \end{gathered} \tag{8}$$

Вычисление правых частей системы (7) осуществляется по формулам

$$\begin{gathered} f_k^{(0)} = f_k, \\ g_k = \frac{f_k^{(k-1)}}{a_{kk}^{(k-1)}}, \quad k = 1,\ 2,\ \ldots,\ n, \\ f_i^{(k)} = f_i^{(k-1)} - a_{ik}^{(k-1)} g_k, \quad i = k+1,\ k+2,\ \ldots,\ n, \quad k = 1,\ 2,\ \ldots,\ n-1. \end{gathered} \tag{9}$$

Второй этап, т.е. обратный ход, заключается в нахождении неизвестных $x_1,\ x_2,\ ...,\ x_n$ из системы (7). Поскольку матрица системы имеет треугольный вид, можно последовательно, начиная с x_n, найти все неизвестные. Общие формулы обратного хода имеют вид [18]

$$x_i = g_i - \sum_{j=i+1}^{n} \left(\alpha_{ij} x_j \right), \quad i = n-1, \ldots, 1, \quad x_n = g_n . \qquad (10)$$

Пример 1. Решить систему уравнений методом Гаусса.

$$\begin{cases} 3x_1 - x_2 = 5, \\ -2x_1 + x_2 + x_3 = 0, \\ 2x_1 - x_2 + 4x_3 = 15. \end{cases}$$

Решение:

Прямой ход.

Разделим первое уравнение на 3, получим систему:

$$\begin{cases} x_1 - \frac{1}{3} x_2 = \frac{5}{3}, \\ -2x_1 + x_2 + x_3 = 0, \\ 2x_1 - x_2 + 4x_3 = 15. \end{cases}$$

Умножим первое уравнение на 2 и сложим со вторым уравнением системы, получим:

$$\begin{cases} x_1 - \frac{1}{3} x_2 = \frac{5}{3}, \\ \frac{1}{3} x_2 + x_3 = \frac{10}{3}, \\ 2x_1 - x_2 + 4x_3 = 15. \end{cases}$$

Умножим первое уравнение на –2 и сложим с третьим уравнением системы, получим:

$$\begin{cases} x_1 - \frac{1}{3} x_2 = \frac{5}{3}, \\ \frac{1}{3} x_2 + x_3 = \frac{10}{3}, \\ \frac{1}{3} x_2 + 4x_3 = \frac{35}{3}. \end{cases}$$

Разделим второе уравнение на 1/3, получим:

$$\begin{cases} x_1 - \frac{1}{3} x_2 = \frac{5}{3}, \\ x_2 + 3x_3 = 10, \\ \frac{1}{3} x_2 + 4x_3 = \frac{35}{3}. \end{cases}$$

Умножим второе уравнение на 1/3 и сложим с третьим уравнением системы, получим:

$$\begin{cases} x_1 - \frac{1}{3}x_2 = \frac{5}{3}, \\ x_2 + 3x_3 = 10, \\ 5x_3 = 15. \end{cases}$$

Разделим третье уравнение на 5, получим:

$$\begin{cases} x_1 - \frac{1}{3}x_2 = \frac{5}{3}, \\ x_2 + 3x_3 = 10, \\ x_3 = 3. \end{cases}$$

Обратный ход.

Из третьего уравнения находим $x_3 = 3$.

Из второго уравнения находим $x_2 = 10 - 3 \cdot 3 = 1$.

Из первого уравнения получаем $x_1 = \frac{5}{3} + \frac{1}{3} = 2$.

Установим связь метода Гаусса с LU-разложением. Прямой ход метода Гаусса преобразует исходную систему уравнений (1) в эквивалентную систему

$$UX = g, \tag{11}$$

где U – верхняя треугольная матрица с единицами на главной диагонали.

Проанализировав соотношения (9), можно записать:

$$f = Lg, \tag{12}$$

где L – нижняя треугольная матрица с ненулевыми элементами $a_{11}^{(0)} = a_{11}, a_{kk}^{(k-1)}$ $k = 1, 2, ..., n$ – на главной диагонали. Выразим из последнего уравнения вектор g, получим:

$$g = L^{-1} f\ . \tag{13}$$

Подставляя (13) в (11), получаем

$$UX = L^{-1} f$$

или

$$LUX = f. \tag{14}$$

Сопоставляя (14) и уравнение (2), приходим к выводу, что в результате применения метода Гаусса получено разложение исходной матрицы A в произведение $A = LU$, где L – нижняя треугольная

матрица с ненулевыми элементами на главной диагонали, U – верхняя треугольная матрица с единичной главной диагональю [19].

Значения элементов матрицы L и U находятся по рекуррентным соотношениям [19]

$$u_{ii} = 1,$$

$$l_{ik} = a_{ik} - \sum_{j=1}^{k-1} l_{ij} u_{jk} \quad при \quad k \le i,$$

$$u_{ik} = \frac{a_{ik} - \sum_{j=1}^{i-1} l_{ij} u_{jk}}{l_{ii}} \quad ïðè \quad i < k, \quad i,k = 1, 2, ..., n. \qquad (15)$$

Таким образом, метод Гаусса можно трактовать следующим образом. Сначала производится разложение матрицы A в произведение двух треугольных матриц L и U, т.е. $A = LU$, а затем последовательно решаются две системы уравнений [19]:

$$Lg = f,$$

$$UX = g.$$

Решая первую систему (прямой ход), находим вектор g. Одновременно происходит разложение $A = LU$.

В результате решения второй системы (обратный ход) находим решение задачи – вектор X.

Пример 2. Решить систему методом Гаусса, используя LU-разложение:

$$\begin{cases} 3x_1 - x_2 = 5, \\ -2x_1 + x_2 + x_3 = 0, \\ 2x_1 - x_2 + 4x_3 = 15. \end{cases}$$

Решение:

Выполним операцию факторизации. Представим матрицу системы $A = \begin{pmatrix} 3 & -1 & 0 \\ -2 & 1 & 1 \\ 2 & -1 & 4 \end{pmatrix}$ в виде $A = LU$

$$l_{11} = a_{11} = 3; \quad l_{21} = a_{21} = -2; \quad l_{31} = a_{31} = 2;$$

При k=1:

$$u_{12} = \frac{a_{12}}{l_{11}} = -\frac{1}{3}; \quad u_{13} = \frac{a_{13}}{l_{11}} = 0;$$

$$l_{22} = a_{22} - l_{21}u_{12} = 1 - 2\frac{1}{3} = \frac{1}{3};$$

При k =2:

$$l_{32} = a_{32} - l_{31} \cdot u_{12} = -1 + 2\frac{1}{3} = -\frac{1}{3};$$

$$u_{23} = \frac{1}{l_{22}}\ a_{23} - l_{21} \cdot u_{13}\ = \frac{1}{1/3}\ 1 - \ -2\ \cdot 0\ = 3;$$

При k =3: $l_{33} = a_{33} - l_{31} \cdot u_{13} - l_{32} \cdot u_{23} = 4 - 2 \cdot 0 - 3\frac{-1}{3} = 5.$

В результате получены две треугольные матрицы:

$$L = \begin{pmatrix} 3 & 0 & 0 \\ -2 & 1/3 & 0 \\ 2 & -1/3 & 5 \end{pmatrix}, \quad U = \begin{pmatrix} 1 & -1/3 & 0 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}.$$

Решим систему линейных уравнений $Lg = f$. Найдем вектор g:

$$\underbrace{\begin{pmatrix} 3 & 0 & 0 \\ -2 & 1/3 & 0 \\ 2 & -1/3 & 5 \end{pmatrix}}_{L} \cdot \underbrace{\begin{pmatrix} g_1 \\ g_2 \\ g_3 \end{pmatrix}}_{g} = \underbrace{\begin{pmatrix} 5 \\ 0 \\ 15 \end{pmatrix}}_{f} \Rightarrow \begin{cases} 3g_1 = 5, \\ -2g_1 + g_2/3 = 0, \\ 2g_1 - g_2/3 + 5g_3 = 0. \end{cases} \Rightarrow \begin{cases} g_1 = 5/3, \\ g_2 = 10, \\ g_3 = 3. \end{cases}$$

Решим систему линейных уравнений $UX = g$. Найдем искомый вектор X:

$$\underbrace{\begin{pmatrix} 1 & -1/3 & 0 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}}_{U} \cdot \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}}_{X} = \underbrace{\begin{pmatrix} 5/3 \\ 10 \\ 3 \end{pmatrix}}_{g} \Rightarrow \begin{cases} x_1 - x_2/3 = 5/3, \\ x_2 + 3x_3 = 10, \\ x_3 = 3. \end{cases} \Rightarrow \begin{cases} x_1 = 2, \\ x_2 = 1, \\ x_3 = 3. \end{cases}$$

Пример 3. Реализовать алгоритм LU-разложения для произвольной матрицы.

При описании алгоритма используются объекты класса «SquareMatrix» и методы данного класса: getRowCount (количество строк матрицы), getElement (получение значения элемента) и setElement (установка значения).

Реализация на языке программирования C++ может иметь следующий вид:

```
void FactorizationAlgorithms :: LU_decomposition (SquareMatrix A, SquareMatrix
LU){
    int n = A. getRowCount ();
```

```
/*Нахождение первого столбца матрицы L и первой строки матрицы U*/
      for (int i = 0; i < n; i++){
        LU.setElement(i, 0, A.getElement(i, 0));
        if ((i+1)<n)
 LU.setElement(0,i+1,A.getElement(0,i+1)/LU.getElement(0, 0));
}
       /*Вычисление  по  формулам (15)*/
      for (int i = 1; i < n; i++){
       for (int j = 1; j < n; j++){
        if (i >= j) /*Нижняя треугольная матрица*/
          {
      double sum = 0;
      for (int k = 0; k < j; k++)
      sum += LU.getElement(i, k)*LU.getElement(k, j);
      LU.setElement(i, j, A.getElement(i, j) - sum);
          }
       else /*Верхняя треугольная матрица*/
         {
       sum = 0;
       for (int k = 0; k < i; k++)
     sum += LU.getElement(i, k)*LU.getElement(k, j);
     LU.setElement(i,j,(A.getElement(i,j)-sum)/LU.getElement(i,i));
                  }
            }
       }
```

В результате факторизации имеем квадратную матрицу, в которой на главной диагонали и ниже расположены элементы матрицы *L*, выше главной диагонали расположены элементы матрицы *U*. Использование памяти при такой структуре полученной матрицы является оптимальным.

Пример 4. Сформировать на языке программирования C++ алгоритм метода Гаусса на основе факторизации.

При описании алгоритма используются объекты и методы классов «SquareMatrix», «Vector» и «FactorizationAlgorithms».

```
void DirectMethodsFactorization :: gaussMethod(SquareMatrix A, Vector f){

     /*Создание матрицы LU, векторов g и x */
      int n = A.getColCount();
      SquareMatrix LU = SquareMatrix(n);
      Vector x = Vector(n);
      Vector g = Vector(n);
     /*   LU – факторизация.  */
      FactorizationAlgorithms FA;
      FA.LU_decomposition(A, LU);
     /* Решение системы Lg = f (прямой ход метода Гаусса), где L – нижняя
треугольная матрица с ненулевыми элементами на главной диагонали  */
```

```
.....
 /* Решение системы  Ux = g (обратный ход метода Гаусса), где U – верхняя
треугольная матрица с единичной главной диагональю */
.....
 /*  Вывод вектора - решения  */
  }
```

Лабораторная работа № 4

Цель: изучить метод Гаусса для решения систем линейных алгебраических уравнений. Применить LU-разложение матриц для решения линейных систем методом Гаусса.

Задание

1. Дополните класс «Алгоритмы факторизации» («FactorizationAlgorithms») методом LU-разложения, используя для построения формулы (15).

2. Разработайте класс «Алгоритмы решения СЛАУ на основе факторизации» («DirectMethodsFactorization»), который наследуется от класса «Прямые методы решения СЛАУ» («DirectMethods»). Реализуйте в данном классе метод Гаусса на основе LU-разложения по формулам (15).

3. Решите систему линейных алгебраических уравнений методом Гаусса ($\varepsilon = 0.001$) в соответствии с вариантом.

4. Решите ту же задачу, используя пакет для математических вычислений.

5. Сравните результат выполнения п. 3 с решением, полученным в п. 4.

Варианты заданий

№ 1

$$\begin{cases} 4.4x_1 - 2.5x_2 + 19.2x_3 - 10.8x_4 = 4.3 \\ 5.5x_1 - 9.3x_2 - 14.2x_3 + 13.2x_4 = 6.8 \\ 7.1x_1 - 11.5x_2 + 5.3x_3 - 6.7x_4 = -1.8 \\ 14.2x_1 + 23.4x_2 - 8.8x_3 + 5.3x_4 = 7.2 \end{cases}$$

№ 2

$$\begin{cases} 8.2x_1 - 3.2x_2 + 14.2x_3 + 14.8x_4 = -8.4 \\ 5.6x_1 - 12x_2 + 15x_3 - 6.4x_4 = 4.5 \\ 5.7x_1 + 3.6x_2 - 12.4x_3 - 2.3x_4 = 3.3 \\ 6.8x_1 + 13.2x_2 - 6.3x_3 - 8.7x_4 = 14.3 \end{cases}$$

№ 3

$$\begin{cases} 5.7x_1 - 7.8x_2 - 5.6x_3 - 8.3x_4 = 2.4 \\ 6.6x_1 + 13.1x_2 - 6.3x_3 + 4.3x_4 = -5.5 \\ 14.7x_1 - 2.8x_2 + 5.6x_3 - 12.1x_4 = 8.6 \\ 18.5x_1 + 12.7x_2 - 23.7x_3 + 5.7x_4 = 14.7 \end{cases}$$

№ 4

$$\begin{cases} 3.8x_1 + 14.2x_2 + 6.3x_3 - 15.5x_4 = 2.8 \\ 8.3x_1 - 6.6x_2 + 5.8x_3 + 12.2x_4 = -4.7 \\ 6.4x_1 - 8.5x_2 - 4.3x_3 + 8.8x_4 = 7.7 \\ 17.1x_1 - 8.3x_2 + 14.4x_3 - 7.2x_4 = 13.5 \end{cases}$$

№ 5

$$\begin{cases} 15.7x_1 + 6.6x_2 - 5.7x_3 + 11.5x_4 = -2.4 \\ 8.8x_1 - 6.7x_2 + 5.5x_3 - 4.5x_4 = 5.6 \\ 6.3x_1 - 5.7x_2 - 23.4x_3 + 6.6x_4 = 7.7 \\ 14.3x_1 + 8.7x_2 - 15.7x_3 - 5.8x_4 = 23.4 \end{cases}$$

№ 6

$$\begin{cases} 4.3x_1 - 12.1x_2 + 23.2x_3 - 14.1x_4 = 15.5 \\ 2.4x_1 - 4.4x_2 + 3.5x_3 + 5.5x_4 = 2.5 \\ 5.4x_1 + 8.3x_2 - 7.4x_3 - 12.7x_4 = 8.6 \\ 6.3x_1 - 7.6x_2 + 1.34x_3 + 3.7x_4 = 12.1 \end{cases}$$

№ 7

$$\begin{cases} 14.4x_1 - 5.3x_2 + 14.3x_3 - 12.7x_4 = -14.4 \\ 23.4x_1 - 14.2x_2 - 5.4x_3 + 2.1x_4 = 6.6 \\ 6.3x_1 - 13.2x_2 - 6.5x_3 + 14.3x_4 = 9.4 \\ 5.6x_1 + 8.8x_2 - 6.7x_3 - 23.8x_4 = 7.3 \end{cases}$$

№ 8

$$\begin{cases} 1.7x_1 + 10x_2 - 1.3x_3 + 2.1x_4 = 3.1 \\ 3.1x_1 + 1.7x_2 - 2.1x_3 + 5.4x_4 = 2.1 \\ 3.3x_1 - 7.7x_2 + 4.4x_3 - 5.1x_4 = 1.9 \\ 10x_1 - 20.1x_2 + 20.4x_3 + 1.7x_4 = 1.8 \end{cases}$$

№ 9

$$\begin{cases} 1.7x_1 - 1.8x_2 + 1.9x_3 - 57.4x_4 = 10 \\ 1.1x_1 - 4.3x_2 + 1.5x_3 - 1.7x_4 = 19 \\ 1.2x_1 + 1.4x_2 + 1.6x_3 + 1.8x_4 = 20 \\ 7.1x_1 - 1.3x_2 - 4.1x_3 + 5.2x_4 = 10 \end{cases}$$

№ 10

$$\begin{cases} 6.1x_1 + 6.2x_2 - 6.3x_3 + 6.4x_4 = 6.5 \\ 1.1x_1 - 1.5x_2 + 2.2x_3 - 3.8x_4 = 4.2 \\ 5.1x_1 - 5.0x_2 + 4.9x_3 - 4.8x_4 = 4.7 \\ 1.8x_1 + 1.9x_2 + 2.0x_3 - 2.1x_4 = 2.2 \end{cases}$$

№ 11

$$\begin{cases} 2.2x_1 - 3.1x_2 + 4.2x_3 - 5.1x_4 = 6.01 \\ 1.3x_1 + 2.2x_2 - 1.4x_3 + 1.5x_4 = 10 \\ 6.2x_1 - 7.4x_2 + 8.5x_3 - 9.6x_4 = 1.1 \\ 1.2x_1 + 1.3x_2 + 1.4x_3 + 4.5x_4 = 1.6 \end{cases}$$

№ 12

$$\begin{cases} 35.8x_1 + 2.1x_2 - 34.5x_3 - 11.8x_4 = 0.5 \\ 27.1x_1 - 7.5x_2 + 11.7x_3 - 23.5x_4 = 12.8 \\ 11.7x_1 + 1.8x_2 - 6.5x_3 + 7.1x_4 = 1.7 \\ 6.3x_1 + 10x_2 + 7.1x_3 + 3.4x_4 = 20.8 \end{cases}$$

№ 13

$$\begin{cases} 35.1x_1 + 1.7x_2 + 37.5x_3 - 2.8x_4 = 7.5 \\ 45.2x_1 + 21.1x_2 - 1.1x_3 - 1.2x_4 = 11.1 \\ -21.1x_1 + 31.7x_2 + 1.2x_3 - 1.5x_4 = 2.1 \\ 31.7x_1 + 18.1x_2 - 31.7x_3 + 2.2x_4 = 0.5 \end{cases}$$

№ 14

$$\begin{cases} 1.1x_1 + 11.2x_2 + 11.1x_3 - 13.1x_4 = 1.3 \\ -3.3x_1 + 1.1x_2 + 30.1x_3 - 20.1x_4 = 1.1 \\ 7.5x_1 + 1.3x_2 + 1.1x_3 + 10x_4 = 20 \\ 1.7x_1 + 7.5x_2 - 1.8x_3 + 2.1x_4 = 1.1 \end{cases}$$

№ 15

$$\begin{cases} 7.5x_1 + 1.8x_2 - 2.1x_3 - 7.7x_4 = 1.1 \\ -10x_1 + 1.3x_2 - 20x_3 - 1.4x_4 = 1.5 \\ 2.8x_1 - 1.7x_2 + 3.9x_3 + 4.8x_4 = 1.2 \\ 10x_1 + 31.4x_2 - 2.1x_3 - 10x_4 = -1.1 \end{cases}$$

№ 16

$$\begin{cases} 30.1x_1 - 1.4x_2 + 10x_3 - 1.5x_4 = 10 \\ -17.5x_1 + 11.1x_2 + 1.3x_3 - 7.5x_4 = 1.3 \\ 1.7x_1 - 21.1x_2 + 7.1x_3 - 17.1x_4 = 10 \\ 2.1x_1 + 2.1x_2 + 3.5x_3 + 3.3x_4 = 1.7 \end{cases}$$

П 3.2 Метод Гаусса с выбором главного элемента для решения систем линейных алгебраических уравнений

Имеем систему линейных алгебраических уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = f_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = f_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = f_n. \end{cases} \quad (1)$$

или в матричном виде $AX = f,$ (2)

где A – вещественная квадратная матрица порядка n, f – заданный и X – искомый векторы. Будем предполагать, что определитель матрицы отличен от нуля.

В методе Гаусса возможность проведения процесса исключения гарантируется условием неравенства нулю главных миноров матрицы A

$$a_{11} \neq 0, \ \det\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \neq 0, \ \ldots, \ \det A \neq 0.$$

Однако при вычислениях заранее неизвестно, все ли главные миноры матрицы A отличны от нуля. При этом может оказаться, что система (1) имеет единственное решение, несмотря на то, что какой-либо из главных миноров матрицы A равен нулю. Кроме того, фиксация ведущего элемента в случае его относительной малости может привести в процессе вычислений к сильному накоплению погрешностей. Избежать таких ситуаций позволяет метод Гаусса с выбором главного элемента.

Основная идея метода Гаусса с выбором главного элемента состоит в том, чтобы на очередном шаге исключать не следующее по номеру неизвестное, а то неизвестное, коэффициент при котором является наибольшим по модулю. Таким образом, в качестве ведущего элемента здесь выбирается главный, т.е. наибольший по модулю элемент.

На практике обычно используются следующие варианты метода Гаусса с выбором главного элемента [19]:

1) метод Гаусса с выбором главного элемента по строке. Ведущий элемент на k-ом шаге исключения выбирается как максимальный по модулю среди элементов k-ой строки. Это равносильно перенумерации переменных на каждом этапе исключения;

2) метод Гаусса с выбором главного элемента по столбцу. Ведущий элемент на k-ом шаге исключения выбирается как главный элемент k-ого столбца. Такой вариант метода Гаусса предусматривает перенумерацию уравнений на каждом этапе исключения;

3) метод Гаусса с выбором главного элемента по всей матрице. Ведущий элемент на *k*-ом шаге исключения выбирается как максимальный по модулю среди всех элементов неприведенной части матрицы, т.е. главный элемент по матрице. Такой вариант предусматривает на каждом этапе исключения соответствующую перенумерацию переменных и перестановку уравнений.

Пример 1. Решить систему уравнений методом Гаусса с выбором главного элемента по столбцу.

$$\begin{cases} -3x_1 + 2.099x_2 + 6x_3 = 3.901, \\ 10x_1 - 7x_2 = 7, \\ 5x_1 - x_2 + 5x_3 = 6. \end{cases}$$

Решение:

Прямой ход.

Максимальным по модулю среди элементов первого столбца является элемент второй строки $a_{21} = 10$. Переставим 1-е и 2-е уравнения, переместив, таким образом, выбранный элемент на место ведущего.

$$\begin{cases} 10x_1 - 7x_2 = 7, \\ -3x_1 + 2.099x_2 + 6x_3 = 3.901, \\ 5x_1 - x_2 + 5x_3 = 6. \end{cases}$$

Проводим первый шаг исключения, как в методе Гаусса. Имеем

$$\begin{cases} x_1 - 0.7x_2 = 0.7, \\ -0.001x_2 + 6x_3 = 6.001, \\ 2.5x_2 + 5x_3 = 2.5. \end{cases}$$

Максимальным по модулю среди элементов второго столбца является элемент третьей строки $a_{32} = 2.5$. Переставим 2-е и 3-е уравнения. Получим

$$\begin{cases} x_1 - 0.7x_2 = 0.7, \\ 2.5x_2 + 5x_3 = 2.5, \\ -0.001x_2 + 6x_3 = 6.001. \end{cases}$$

Проводим второй шаг исключения. Имеем

$$\begin{cases} x_1 - 0.7x_2 = 0.7, \\ x_2 + 2x_3 = 1, \\ 6.002x_3 = 6.002. \end{cases}$$

Разделим третье уравнение на 6.002, получим

$$\begin{cases} x_1 - 0.7x_2 = 0.7, \\ x_2 + 2x_3 = 1, \\ x_3 = 1. \end{cases}$$

В результате применения обратного хода, имеем $x_1 = 0$, $x_2 = -1$, $x_3 = 1$.

Рассмотрим метод Гаусса с выбором главного элемента на основе факторизации.

Матрицей перестановок P называется квадратная матрица, у которой в каждой строке и в каждом столбце только один элемент отличен от нуля и равен единице.

Элементарной матрицей перестановок P_{kl} называется матрица, полученная из единичной матрицы перестановкой k-й и l-й строк.

Например, элементарными матрицами перестановок третьего порядка являются матрицы

$$P_{12} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad P_{13} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad P_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Отметим следующие свойства элементарных матриц перестановок, вытекающие непосредственно из их определения [19]:

- Произведение двух (а следовательно, и любого числа) элементарных матриц перестановок является матрицей перестановок (не обязательно элементарной).
- Для любой квадратной матрицы A матрица $P_{kl}A$ отличается от A перестановкой k-й и l-й строк.
- Для любой квадратной матрицы A матрица AP_{kl} отличается от A перестановкой k-го и l-го столбцов.

Поясним применение элементарных матриц перестановок для описания метода Гаусса с выбором главного элемента по столбцу. Рассмотрим следующий пример системы третьего порядка:

$$\begin{cases} x_1 + x_2 + x_3 = f_1, \\ 2x_1 + \quad x_3 = f_2, \\ \quad 5x_2 + 3x_3 = f_3. \end{cases} \tag{3}$$

Матрица системы имеет вид

$$A=\begin{pmatrix}1 & 1 & 1\\ 2 & 0 & 1\\ 0 & 5 & 3\end{pmatrix}. \qquad (4)$$

Максимальный элемент первого столбца матрицы A находится во второй строке. Поэтому в системе (3) необходимо поменять местами первую и вторую строки и перейти к эквивалентной системе

$$\begin{cases}2x_1 \quad + x_3 = f_2,\\ x_1 + x_2 + x_3 = f_1,\\ \quad 5x_2 + 3x_3 = f_3.\end{cases} \qquad (5)$$

Систему (5) можно записать в виде

$$P_{12}AX = P_{12}f\ , \qquad (6)$$

т.е. система (6) получается из системы (3) путем умножения на матрицу перестановок $P_{12}=\begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}$.

Далее, к системе (5) надо применить первый шаг обычного метода исключения Гаусса, для того чтобы привести матрицу системы к виду $\begin{pmatrix}1 & \times & \times\\ 0 & \times & \times\\ 0 & \times & \times\end{pmatrix}$.

Этот шаг эквивалентен умножению матрицы системы (6) слева на элементарную нижнюю треугольную матрицу $L_1=\begin{pmatrix}\frac{1}{a_{11}^1} & 0 & 0\\ \frac{-a_{21}^1}{a_{11}^1} & 1 & 0\\ \frac{-a_{31}^1}{a_{11}^1} & 0 & 1\end{pmatrix}$,

т.е. в нашем случае $L_1=\begin{pmatrix}\frac{1}{2} & 0 & 0\\ \frac{-1}{2} & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$.

В результате от (6) перейдем к системе

$$L_1 P_{12} AX = L_1 P_{12} f \,. \qquad (7)$$

Имеем:

$$\begin{cases} x_1 \quad + \dfrac{1}{2} x_3 = \dfrac{f_2}{2}, \\ x_2 + \dfrac{1}{2} x_3 = f_1 - \dfrac{f_2}{2}, \\ 5x_2 + 3x_3 = f_3. \end{cases} \qquad (8)$$

Из последних двух уравнений системы (8) необходимо исключить неизвестное x_2. Максимальным элементом второго столбца системы (8) является элемент третьей строки. Следовательно, в системе (8) необходимо поменять местами вторую и третью строки и тем самым перейти к эквивалентной системе (9)

$$\begin{cases} x_1 \quad + \dfrac{1}{2} x_3 = \dfrac{f_2}{2}, \\ 5x_2 + 3x_3 = f_3, \\ x_2 + \dfrac{1}{2} x_3 = f_1 - \dfrac{f_2}{2}. \end{cases} \qquad (9)$$

которую можно записать в матричном виде как

$$P_{23} L_1 P_{12} AX = P_{23} L_1 P_{12} f \,. \qquad (10)$$

Таким образом, система (10) получена применением к системе (7) элементарной матрицы перестановок $P_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$.

Далее, к системе (10) необходимо применить второй шаг исключения обычного метода Гаусса, для того чтобы привести матрицу системы к виду $\begin{pmatrix} 1 & \times & \times \\ 0 & 1 & \times \\ 0 & 0 & \times \end{pmatrix}$.

Этот шаг эквивалентен умножению матрицы системы (10) слева на элементарную нижнюю треугольную матрицу $L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \dfrac{1}{a_{22}^{(1)}} & 0 \\ 0 & \dfrac{-a_{32}^{(1)}}{a_{22}^{(1)}} & 1 \end{pmatrix}$,

т.е. для данного примера $L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{5} & 0 \\ 0 & \frac{-1}{5} & 1 \end{pmatrix}$.

В результате получим систему

$$L_2 P_{23} L_1 P_{12} AX = L_2 P_{23} L_1 P_{12} f \qquad (11)$$

или

$$\begin{cases} x_1 + \frac{1}{2} x_3 = \frac{f_2}{2}, \\ x_2 + \frac{3}{5} x_3 = \frac{f_3}{5}, \\ -\frac{1}{10} x_3 = f_1 - \frac{f_2}{2} - \frac{f_3}{5}. \end{cases} \qquad (12)$$

Заключительный шаг прямого хода метода Гаусса состоит в замене последнего уравнения системы (12) уравнением

$$x_3 = -10\left(f_1 - \frac{f_2}{2} - \frac{f_3}{5} \right).$$

Что эквивалентно умножению (11) на матрицу $L_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{a_{33}} \end{pmatrix}$,

т.е в нашем случае $L_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -10 \end{pmatrix}$.

Таким образом, для рассмотренного примера процесс исключения Гаусса с выбором главного элемента по столбцу записывается в виде

$$L_3 L_2 P_{23} L_1 P_{12} AX = L_3 L_2 P_{23} L_1 P_{12} f \,. \qquad (13)$$

В результате получим систему

$$\begin{cases} x_1 + \frac{1}{2} x_3 = \frac{f_2}{2}, \\ x_2 + \frac{3}{5} x_3 = \frac{f_3}{5}, \\ x_3 = -10\left(f_1 - \frac{f_2}{2} - \frac{f_3}{5} \right). \end{cases} \qquad (14)$$

По построению матрица

$$U = L_3 L_2 P_{23} L_1 P_{12} A \qquad (15)$$

является верхней треугольной матрицей с единичной главной диагональю.

Для нахождения неизвестных к системе (14) применяется обратный ход обычного метода Гаусса.

Отличие метода Гаусса с выбором главного элемента по столбцу (строке, всей матрице) от обычного метода Гаусса состоит в том, что в качестве сомножителей в (15) наряду с элементарными треугольными матрицами L_k могут присутствовать элементарные матрицы перестановок P_{kl}.

Метод Гаусса с выбором главного элемента по столбцу эквивалентен обычному методу Гаусса, примененному к системе, полученной из исходной системы перестановкой уравнений

$$PAX = Pf. \qquad (16)$$

Справедливо разложение [19]

$$PA = LU, \qquad (17)$$

где L – нижняя треугольная матрица с отличными от нуля диагональными элементами и U – верхняя треугольная матрица с единичной главной диагональю.

Следует подчеркнуть, что в методе Гаусса с выбором главного элемента по столбцу (строке, всей матрице) матрица P не задается заранее, а строится в процессе исключения.

Пример 2. Описать алгоритм метода Гаусса с выбором главного элемента по столбцу на основе факторизации.

При описании алгоритма использовать объекты и методы классов «SquareMatrix», «Vector», «AugmentMatrix» и «SwapMatrix».

При реализации метода эффективнее использовать расширенную матрицу системы и проводить необходимые преобразования над ней. Приведем возможное описание класса «AugmentMatrix» (Расширенная матрица) на языке программирования C++.

```
/*   Класс «AugmentMatrix»*/
class AugmentMatrix : public AbstractMatrix{
      /* Массив элементов матрицы*/
        double **elements;
      /* Количество столбцов в матрице. Количество строк матрицы задавали ранее
в родительском классе «AbstractMatrix» */
        int ColCount;
      public:
```

```
    /* Конструктор класса – расширенная матрица, состоящая из квадратной матрицы и вектора */
      AugmentMatrix(SquareMatrix A, Vector f);
     /* Конструктор класса – расширенная матрица, состоящая из  двух квадратных матриц */
      AugmentMatrix(SquareMatrix A, SquareMatrix B);
    /* Получение элемента по индексу*/
     double getElement(int i, int j);
    /* Установка значения элемента   */
     void setElement(int i, int j, double element);
    /* Получение количества строк в матрице  */
     int getColCount();
    /* Получение количества столбцов в матрице  */
     int getRowCount();
    /*Умножение квадратной матрицы и расширенной*/
AugmentMatrix operator *(SquareMatrix A, AugmentMatrix B);
   /*Умножение матрицы перестановок и расширенной  */
AugmentMatrix operator *(SwapMatrix A, AugmentMatrix B);
};
```

В методе Гаусса с выбором главного элемента будем применять расширенную матрицу, состоящую из квадратной матрицы и вектора. Приведем пример данного конструктора класса «AugmentMatrix» на языке программирования C++.

```
/* Конструктор – расширенная матрица, состоящая из квадратной матрицы и вектора*/
AugmentMatrix :: AugmentMatrix(SquareMatrix A, Vector f){
    /* Задание размеров матрицы (количество строк и столбцов матрицы)*/
     this->ColCount = A.getColCount() + 1;
     this->size = A.getRowCount();
    /* Создание матрицы*/
     this->elements = new double*[this->size];
     for (int i=0; i < this->size; i++){
         this->elements[i]=new double[this->ColCount];
    }
    /* Заполнение матрицы*/
     for (int i=0; i < this->size; i++){
        for (int j=0; j < this->ColCount; j++)
          if (j < this->ColCount - 1){
           this->elements[i][j] = A.getElement(i, j);
          } else{
            this->elements[i][j] = f.getElement(i);
                }
         }
    }
```

Основная операция рассматриваемого метода – это перемножение матриц. Реализация данной операции на языке C++ имеет вид

```
/*Умножение квадратной и расширенной матрицы*/
AugmentMatrix operator *(SquareMatrix A, AugmentMatrix B) {
    AugmentMatrix C(B.getRowCount(), B.getColCount());
    for (int i = 0; i < A.getRowCount(); i++) {
     for (int j = 0; j < B.getColCount(); j++) {
       C.elements[i][j]=0;
      for (int t = 0 ; t < B.getRowCount(); t++)
  C.elements[i][j] += A.elements[i][t] * B.elements[t][j];
                              }
                            }
                            return C;}
```

Для перестановки строк матрицы используется класс «SwapMatrix» (Матрица перестановок). Матрица перестановок получается из единичной матрицы изменением порядка расположения строк. Следовательно, для экономии памяти необходимо хранить только позицию единиц в каждой строке. Опишем возможную структуру данного класса. Реализация на языке программирования C++ имеет вид

```
/*Матрица перестановок*/
class SwapMatrix: public EMatrix {
      /* Используется для хранения позиции единицы в каждой   строке*/
       int *Indexes;
      public:
      /*Конструктор*/
      SwapMatrix (int size);
      /* Перестановка строк*/
      void swapRows(int i1, int i2);
      /* Перестановка столбцов*/
      void swapCols(int j1, int j2);
      /* Получение количества строк в матрице   */
       int getColCount();
      /* Получение количества столбцов в матрице  */
       int getRowCount();
      /* Установка значения элемента */
     void setElement(int i, int j, double element);
      /* Получение элемента по индексу */
     double getElement(int i, int j);
};
/* Конструктор*/
SwapMatrix:: SwapMatrix(int size):EMatrix(size) {
/* Indexes[i] описывает положение единицы в строке  i*/
     this->Indexes = new int[size];
     for (int i = 0; i < size; ++i)
     this->Indexes[i] = i;

/* Для единичной матрицы, номер строки совпадает с положением единицы в ней,
то есть
1 0 0                    Indexes[0] = 0;
```

```
0 1 0                    Indexes[1] = 1;
0 0 1                    Indexes[2] = 2;
```
для перестановочной матрицы, индексы будут уже совпадать не везде, например если поменять строки 2 и 3
```
1 0 0                    Indexes[0] = 0;
0 0 1                    Indexes[1] = 2;
0 1 0                    Indexes[2] = 1;
```
Таким методом не нужно хранить всю матрицу в памяти.
```
*/
}

/* Перестановка строк*/
void SwapMatrix ::swapRows(int i1, int i2) {
    int k = this->Indexes[i1];
    this->Indexes[i1] = Indexes[i2];
    this->Indexes[i2] = k;
    return;
}
```

Все необходимые классы и методы реализованы, опишем алгоритм метода Гаусса с выбором главного элемента по столбцу на языке С++.

```
void DirectMethodsNF::gaussChooseElement(SquareMatrix A, Vector f){
    /* Создание расширенной матрицы системы                    */
     AugmentMatrix Au = AugmentMatrix(A, f);
     /*цикл*/
 for (int i = 0; i < Au.getRowCount()-1; ++i) {

     /*Поиск значения и индекса максимального элемента в столбце. Данный метод реализован в классе «SquareMatrix».
Value – значение максимального элемента
Index – индекс максимального элемента*/

    /*Создание матрицы перестановок*/
     SwapMatrix P(Au.getRowCount());
    /*Перестановка строк*/
     P.swapRows(i,Index);

    /*Умножение матрицы перестановок и расширенной матрицы. Данный метод описан в классе «AugmentMatrix»*/
     Au = P*Au;

    /* Создание квадратной матрицы L и инициализация ее единичной. Метод инициализации квадратной матрицы единичной описан в классе «SquareMatrix»*/
    /*Формирование матрицы L*/
    L.setElement(i, i, 1/Au.getElement(i,i));
    for (int j = i+1; j < Au.getRowCount(); ++j) {
      if (Au.getElement(i,i) != 0)
```

```
L.setElement(j,i,-(Au.getElement(j,i)/Au.getElement(i,i)));
          else
            L.setElement(j, i, 0);
               }
       /*Умножение квадратной и расширенной матрицы. Данный метод описан в
классе «AugmentMatrix»*/
            Au = L*Au;
                          }

      /*Создание вектора решений*/
       Vector X(Au.getRowCount());
      /* Вычисление вектора решений обратным ходом метода Гаусса.*/

}
```

Лабораторная работа № 5

Цель: изучить метод Гаусса с выбором главного элемента для решения систем линейных алгебраических уравнений.

Задание

1. Реализуйте в классе «DirectMethodsFactorization» метод Гаусса с выбором главного элемента («gaussChooseElement»). Для реализации метода используйте объекты и методы матричных классов «SquareMatrix», «Vector», «AugmentMatrix» и «SwapMatrix». Продумайте и реализуйте методы работы с матрицей класса «SwapMatrix». Возможная структура класса «SwapMatrix» приведена в примере 2. Для эффективной реализации метода все необходимые матричные операции производите над объектом класса «AugmentMatrix», описанным в примере 2.

2. Решите систему линейных алгебраических уравнений методом Гаусса с выбором главного элемента ($\varepsilon = 0.001$) в соответствии с вариантом.

3. Решите ту же задачу, используя пакет для математических вычислений.

4. Сравните результат выполнения п. 2 с решением, полученным в п. 3

Варианты заданий

№ 1

$$\begin{cases} 0.34x_1 + 0.71x_2 + 0.63x_3 = 2.08 \\ 0.71x_1 - 0.65x_2 - 0.18x_3 = 0.17 \\ 1.17x_1 - 2.35x_2 + 0.75x_3 = 1.28 \end{cases}$$

№ 2

$$\begin{cases} 3.75x_1 - 0.28x_2 + 0.17x_3 = 0.75 \\ 2.11x_1 - 0.11x_2 - 0.12x_3 = 1.11 \\ 0.22x_1 - 3.17x_2 + 1.81x_3 = 0.05 \end{cases}$$

№ 3

$$\begin{cases} 0.21x_1 - 0.18x_2 + 0.75x_3 = 0.11 \\ 0.13x_1 + 0.75x_2 - 0.11x_3 = 2.00 \\ 3.01x_1 - 0.33x_2 + 0.11x_3 = 0.13 \end{cases}$$

№ 4

$$\begin{cases} 0.13x_1 - 0.14x_2 - 2.00x_3 = 0.15 \\ 0.75x_1 + 0.18x_2 - 0.77x_3 = 0.11 \\ 0.28x_1 - 0.17x_2 + 0.39x_3 = 0.12 \end{cases}$$

№ 5

$$\begin{cases} 3.01x_1 - 0.14x_2 - 0.15x_3 = 1.00 \\ 1.11x_1 + 0.13x_2 - 0.75x_3 = 0.13 \\ 0.17x_1 - 2.11x_2 + 0.71x_3 = 0.17 \end{cases}$$

№ 6

$$\begin{cases} 0.92x_1 - 0.83x_2 + 0.62x_3 = 2.15 \\ 0.24x_1 - 0.54x_2 + 0.43x_3 = 0.62 \\ 0.73x_1 - 0.81x_2 - 0.67x_3 = 0.88 \end{cases}$$

№ 7

$$\begin{cases} 1.24x_1 - 0.87x_2 - 3.17x_3 = 0.46 \\ 2.11x_1 - 0.45x_2 + 1.44x_3 = 1.50 \\ 0.48x_1 + 1.25x_2 - 0.63x_3 = 0.35 \end{cases}$$

№ 8

$$\begin{cases} 0.64x_1 - 0.83x_2 + 4.2x_3 = 2.23 \\ 0.58x_1 - 0.83x_2 + 1.43x_3 = 1.71 \\ 0.86x_1 + 0.77x_2 + 0.88x_3 = -0.54 \end{cases}$$

№ 9

$$\begin{cases} 0.32x_1 - 0.42x_2 + 0.85x_3 = 1.32 \\ 0.63x_1 - 1.43x_2 - 0.58x_3 = -0.44 \\ 0.84x_1 - 2.23x_2 - 0.52x_3 = 0.64 \end{cases}$$

№ 10

$$\begin{cases} 0.73x_1 + 1.24x_2 - 0.38x_3 = 0.58 \\ 1.25x_1 + 0.66x_2 - 0.78x_3 = 0.66 \\ 0.75x_1 + 1.22x_2 - 0.83x_3 = 0.92 \end{cases}$$

№ 11

$$\begin{cases} 0.62x_1 - 0.44x_2 - 0.86x_3 = 0.68 \\ 0.83x_1 + 0.42x_2 - 0.56x_3 = 1.24 \\ 0.58x_1 - 0.37x_2 - 0.62x_3 = 0.87 \end{cases}$$

№ 12

$$\begin{cases} 1.26x_1 - 2.34x_2 + 1.17x_3 = 3.14 \\ 0.75x_1 + 1.24x_2 - 0.48x_3 = -1.17 \\ 3.44x_1 - 1.85x_2 + 1.16x_3 = 1.83 \end{cases}$$

№ 13

$$\begin{cases} 0.46x_1 + 1.72x_2 + 2.53x_3 = 2.44 \\ 1.53x_1 - 2.32x_2 - 1.83x_3 = 2.83 \\ 0.75x_1 + 0.86x_2 + 3.72x_3 = 1.06 \end{cases}$$

№ 14

$$\begin{cases} 2.47x_1 + 0.65x_2 - 1.88x_3 = 1.24 \\ 1.34x_1 + 1.17x_2 + 2.54x_3 = 2.35 \\ 0.86x_1 - 1.73x_2 - 1.08x_3 = 3.15 \end{cases}$$

№ 15

$$\begin{cases} 4.24x_1 + 2.73x_2 - 1.55x_3 = 1.87 \\ 2.34x_1 + 1.27x_2 + 3.15x_3 = 2.16 \\ 3.05x_1 - 1.05x_2 - 0.63x_3 = -1.25 \end{cases}$$

№ 16

$$\begin{cases} 0.43x_1 + 1.24x_2 - 0.58x_3 = 2.71 \\ 0.74x_1 + 0.83x_2 + 1.17x_3 = 1.26 \\ 1.43x_1 - 1.58x_2 + 0.83x_3 = 1.03 \end{cases}$$

П 3.3 Решение системы линейных алгебраических уравнений методом Жордана-Гаусса

Рассмотрим метод Жордана-Гаусса.

Имеем систему линейных алгебраических уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = f_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = f_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = f_n. \end{cases} \quad (1)$$

или в матричном виде $AX = f$. (2)

Суть метода Жордана-Гаусса состоит в том, чтобы привести матрицу A к единичному виду, тогда вектор решения будет совпадать со столбцом свободных членов. Алгоритмически метод Жордана-Гаусса объединяет прямой и обратный ход метода Гаусса [3].

Первый шаг метода Жордана-Гаусса аналогичен первому шагу метода Гаусса, т.е. первое уравнение необходимо разделить на a_{11} – коэффициент при неизвестном x_1 и исключить это неизвестное – x_1 из остальных уравнений системы. На втором шаге необходимо исключить неизвестное x_2 из всех уравнений системы (в том числе из первого) кроме второго. На третьем шаге оставляем неизвестное x_3 только в третьем уравнении и т.д. Окончательно получаем систему вида

$$\begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ \ldots \\ g_n \end{bmatrix}.$$

Столбец свободных членов в последней расширенной матрице и есть решение системы (1).

Пример 1. Решить с помощью метода Жордана-Гаусса систему уравнений:

$$\begin{cases} 5x_1 + 4x_2 - 0.5x_3 = 1.5, \\ 4x_1 - 6.8x_2 + 0.4x_3 = -3, \\ x_1 + 2x_2 + 0.5x_3 = 1.5. \end{cases}$$

Решение:

Разделим первое уравнение на 3, получим следующую систему:

$$\begin{cases} x_1 + \frac{4}{5}x_2 - 0.1x_3 = 0.3, \\ 4x_1 - 6.8x_2 + 0.4x_3 = -3, \\ x_1 + 2x_2 + 0.5x_3 = 1.5. \end{cases}$$

Сложим первое уравнение со вторым и третьим, умножив соответственно на –4 и –1, получим систему:

$$\begin{cases} x_1 + \frac{4}{5}x_2 - 0.1x_3 = 0.3, \\ -10x_2 + 0.8x_3 = -4.2, \\ \frac{6}{5}x_2 + 0.6x_3 = 1.2. \end{cases}$$

Разделим второе уравнение на –10, получим следующую систему:

$$\begin{cases} x_1 + \frac{4}{5}x_2 - 0.1x_3 = 0.3, \\ x_2 - 0.08x_3 = 0.42, \\ \frac{6}{5}x_2 + 0.6x_3 = 1.2. \end{cases}$$

Сложим второе уравнение с первым и третьим, умножив соответственно на –4/5 и –6/5, получим:

$$\begin{cases} x_1 - 0.036x_3 = -0.036, \\ x_2 - 0.08x_3 = 0.42, \\ 0.696x_3 = 0.696. \end{cases}$$

Разделим третье уравнение на 0.696, получим следующую систему:

$$\begin{cases} x_1 - 0.036x_3 = -0.036, \\ x_2 - 0.08x_3 = 0.42, \\ x_3 = 1. \end{cases}$$

Сложим третье уравнение с первым и вторым, умножив его на 0.036 и 0.08, получим:

$$\begin{cases} x_1 = 0, \\ x_2 = 0.5, \\ x_3 = 1. \end{cases}$$

Рассмотрим метод Жордана-Гаусса на основе факторизации.

На первом шаге исключения матрица системы (2) приводится к виду $\begin{pmatrix} 1 & \times & \ldots & \times \\ 0 & \times & \ldots & \times \\ \ldots & \ldots & \ldots & \ldots \\ 0 & \times & \ldots & \times \end{pmatrix}$.

Этот шаг эквивалентен умножению матрицы системы (2) слева на элементарную нижнюю треугольную матрицу $L_1 = \begin{pmatrix} \frac{1}{a_{11}} & 0 & \ldots & 0 \\ \frac{-a_{21}}{a_{11}} & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ \frac{-a_{n1}}{a_{11}} & 0 & \ldots & 1 \end{pmatrix}$.

В результате имеем систему $L_1 AX = L_1 f$. (3)

На втором шаге исключения матрица системы (3) приводится к виду $\begin{pmatrix} 1 & 0 & \ldots & \times \\ 0 & 1 & \ldots & \times \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & \times \end{pmatrix}$.

Для этого необходимо умножить матрицу системы (3) слева на элементарную нижнюю треугольную матрицу $L_2 = \begin{pmatrix} 1 & \frac{-a_{12}^{(1)}}{a_{22}^{(1)}} & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & \frac{-a_{n2}^{(1)}}{a_{22}^{(1)}} & \ldots & 1 \end{pmatrix}$.

В результате получим систему $L_2 L_1 AX = L_2 L_1 f$. (4)

Продолжая этот процесс, в итоге приходим к системе

$$L_n L_{n-1} ... L_2 L_1 AX = L_n L_{n-1} ... L_2 L_1 f \ , \qquad (5)$$

где на κ-ом шаге исключения элементарная нижняя треугольная матрица L_k имеет вид $L_k = \begin{pmatrix} 1 & 0 & \frac{-a_{1k}^{(-1)}}{a_{kk}^{(-1)}} & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & \frac{1}{a_{kk}^{(-1)}} & 0 & 0 \\ \dots & \dots & \dots & 1 & \dots \\ 0 & 0 & \frac{-a_{nk}^{(-1)}}{a_{kk}^{(-1)}} & \dots & 1 \end{pmatrix}$.

Исходя из того, что $L_n L_{n-1} ... L_2 L_1 A = E$, перепишем систему (5) в виде

$$EX = L_n L_{n-1} ... L_2 L_1 f \, . \qquad (6)$$

Таким образом, правая часть системы (6) представляет собой искомый вектор решения.

Пример 2. Решить с помощью метода Жордана-Гаусса на основе факторизации систему уравнений:

$$\begin{cases} 5x_1 + 4x_2 - 0.5x_3 = 1.5, \\ 4x_1 - 6.8x_2 + 0.4x_3 = -3, \\ x_1 + 2x_2 + 0.5x_3 = 1.5. \end{cases}$$

Решение:

Перепишем систему в матричном виде

$$\begin{pmatrix} 5 & 4 & -0.5 \\ 4 & -6.8 & 0.4 \\ 1 & 2 & 0.5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1.5 \\ -3 \\ 1.5 \end{pmatrix}.$$

Применим первый шаг исключения, для этого умножим матрицу системы слева на матрицу $L_1 = \begin{pmatrix} \frac{1}{5} & 0 & 0 \\ \frac{-4}{5} & 1 & 0 \\ \frac{-1}{5} & 0 & 1 \end{pmatrix}$.

Получим систему $L_1 AX = L_1 f$ или в развернутом виде

$$\begin{pmatrix} 1 & 0.8 & -0.1 \\ 0 & -10 & 0.8 \\ 0 & 1.2 & 0.6 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0.3 \\ -4.2 \\ 1.2 \end{pmatrix}.$$

Далее применим второй шаг исключения, для этого матрицу системы, полученной на предыдущем шаге, умножим слева на матрицу $L_2 = \begin{pmatrix} 1 & \frac{0.8}{10} & 0 \\ 0 & \frac{-1}{10} & 0 \\ 0 & \frac{1.2}{10} & 1 \end{pmatrix}$.

Имеем систему $L_2 L_1 AX = L_2 L_1 f$ или

$$\begin{pmatrix} 1 & 0 & -0.036 \\ 0 & 1 & -0.08 \\ 0 & 0 & 0.696 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -0.036 \\ 0.42 \\ 0.696 \end{pmatrix}.$$

На заключительном этапе имеем систему $L_3 L_2 L_1 AX = L_3 L_2 L_1 f$, где матрица $L_3 = \begin{pmatrix} 1 & 0 & \frac{-0.036}{-0.696} \\ 0 & 1 & \frac{-0.08}{-0.696} \\ 0 & 0 & \frac{1}{0.696} \end{pmatrix}$.

В результате получим систему $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0.5 \\ 1 \end{pmatrix}$.

Отсюда следует, что искомый вектор найден $X = \begin{bmatrix} 0 \\ 0.5 \\ 1 \end{bmatrix}$.

Лабораторная работа № 6

Цель: изучить метод Жордана-Гаусса для решения системы линейных алгебраических уравнений.

Задание

1. Реализуйте в классе «DirectMethodsFactorization» метод Жордана-Гаусса («gaussJordanMethod»). Для реализации метода используйте объекты и методы класса «SquareMatrix», «Vector» и

«AugmentMatrix». Доступ к элементам осуществляйте с помощью методов getElement и setElement, реализованных в данных классах. Для эффективной реализации метода все необходимые матричные операции производите над объектом класса «AugmentMatrix».

2. Решите систему линейных алгебраических уравнений методом Жордана-Гаусса ($\varepsilon = 0.001$) в соответствии с вариантом.

3. Решите ту же задачу, используя пакет для математических вычислений.

4. Сравните результат выполнения п. 2 с решением, полученным в п. 3.

Варианты заданий

№ 1

$$\begin{cases} 0.34x_1 + 0.71x_2 + 0.63x_3 = 2.08 \\ 0.71x_1 - 0.65x_2 - 0.18x_3 = 0.17 \\ 1.17x_1 - 2.35x_2 + 0.75x_3 = 1.28 \end{cases}$$

№ 2

$$\begin{cases} 3.75x_1 - 0.28x_2 + 0.17x_3 = 0.75 \\ 2.11x_1 - 0.11x_2 - 0.12x_3 = 1.11 \\ 0.22x_1 - 3.17x_2 + 1.81x_3 = 0.05 \end{cases}$$

№ 3

$$\begin{cases} 0.21x_1 - 0.18x_2 + 0.75x_3 = 0.11 \\ 0.13x_1 + 0.75x_2 - 0.11x_3 = 2.00 \\ 3.01x_1 - 0.33x_2 + 0.11x_3 = 0.13 \end{cases}$$

№ 4

$$\begin{cases} 0.13x_1 - 0.14x_2 - 2.00x_3 = 0.15 \\ 0.75x_1 + 0.18x_2 - 0.77x_3 = 0.11 \\ 0.28x_1 - 0.17x_2 + 0.39x_3 = 0.12 \end{cases}$$

№ 5

$$\begin{cases} 3.01x_1 - 0.14x_2 - 0.15x_3 = 1.00 \\ 1.11x_1 + 0.13x_2 - 0.75x_3 = 0.13 \\ 0.17x_1 - 2.11x_2 + 0.71x_3 = 0.17 \end{cases}$$

№ 6

$$\begin{cases} 0.92x_1 - 0.83x_2 + 0.62x_3 = 2.15 \\ 0.24x_1 - 0.54x_2 + 0.43x_3 = 0.62 \\ 0.73x_1 - 0.81x_2 - 0.67x_3 = 0.88 \end{cases}$$

№ 7

$$\begin{cases} 1.24x_1 - 0.87x_2 - 3.17x_3 = 0.46 \\ 2.11x_1 - 0.45x_2 + 1.44x_3 = 1.50 \\ 0.48x_1 + 1.25x_2 - 0.63x_3 = 0.35 \end{cases}$$

№ 8

$$\begin{cases} 0.64x_1 - 0.83x_2 + 4.2x_3 = 2.23 \\ 0.58x_1 - 0.83x_2 + 1.43x_3 = 1.71 \\ 0.86x_1 + 0.77x_2 + 0.88x_3 = -0.54 \end{cases}$$

№ 9

$$\begin{cases} 0.32x_1 - 0.42x_2 + 0.85x_3 = 1.32 \\ 0.63x_1 - 1.43x_2 - 0.58x_3 = -0.44 \\ 0.84x_1 - 2.23x_2 - 0.52x_3 = 0.64 \end{cases}$$

№ 10

$$\begin{cases} 0.73x_1 + 1.24x_2 - 0.38x_3 = 0.58 \\ 1.25x_1 + 0.66x_2 - 0.78x_3 = 0.66 \\ 0.75x_1 + 1.22x_2 - 0.83x_3 = 0.92 \end{cases}$$

№ 11

$$\begin{cases} 0.62x_1 - 0.44x_2 - 0.86x_3 = 0.68 \\ 0.83x_1 + 0.42x_2 - 0.56x_3 = 1.24 \\ 0.58x_1 - 0.37x_2 - 0.62x_3 = 0.87 \end{cases}$$

№ 12

$$\begin{cases} 1.26x_1 - 2.34x_2 + 1.17x_3 = 3.14 \\ 0.75x_1 + 1.24x_2 - 0.48x_3 = -1.17 \\ 3.44x_1 - 1.85x_2 + 1.16x_3 = 1.83 \end{cases}$$

№ 13

$$\begin{cases} 0.46x_1 + 1.72x_2 + 2.53x_3 = 2.44 \\ 1.53x_1 - 2.32x_2 - 1.83x_3 = 2.83 \\ 0.75x_1 + 0.86x_2 + 3.72x_3 = 1.06 \end{cases}$$

№ 14

$$\begin{cases} 2.47x_1 + 0.65x_2 - 1.88x_3 = 1.24 \\ 1.34x_1 + 1.17x_2 + 2.54x_3 = 2.35 \\ 0.86x_1 - 1.73x_2 - 1.08x_3 = 3.15 \end{cases}$$

№ 15

$$\begin{cases} 4.24x_1 + 2.73x_2 - 1.55x_3 = 1.87 \\ 2.34x_1 + 1.27x_2 + 3.15x_3 = 2.16 \\ 3.05x_1 - 1.05x_2 - 0.63x_3 = -1.25 \end{cases}$$

№ 16

$$\begin{cases} 0.43x_1 + 1.24x_2 - 0.58x_3 = 2.71 \\ 0.74x_1 + 0.83x_2 + 1.17x_3 = 1.26 \\ 1.43x_1 - 1.58x_2 + 0.83x_3 = 1.03 \end{cases}$$

П 3.4 Метод квадратного корня для решения систем линейных алгебраических уравнений

Рассмотрим алгоритм метода квадратного корня.

Метод предназначен для решения СЛАУ с симметрическими матрицами.

Пусть надо решить систему

$$AX = f, \qquad (1)$$

где матрица A – вещественная, симметрическая; X – искомый вектор решения; f – вектор правой части системы.

Матрицу A можно представить в виде $A = S^T D S$, где S – верхняя треугольная матрица, S^T – транспонированная к ней матрица (нижняя треугольная), D – диагональная матрица с элементами, равными +1 или –1.

Значения коэффициентов матрицы S и D находятся по формулам (2), описанным в п 2.2.

Исходную систему (1) заменяем двумя эквивалентными ей системами с треугольными матрицами: $S^T D Y = f$, $SX = Y$.

Решая их, получаем [19; 8]:

$$y_1 = \frac{f_1}{s_{11} d_{11}}, \quad y_k = \frac{f_k - \sum_{s=1}^{k-1} s_{sk} y_s d_{ss}}{s_{kk} d_{kk}}, \; k = 2, \ldots, n, \qquad (2)$$

$$x_n = \frac{y_n}{s_{nn}}, \quad x_k = \frac{y_k - \sum_{s=k+1}^{n} s_{ks} x_s}{s_{kk}}, \; k = n-1, \; n-2, \ldots, 1.$$

Если матрица A является симметричной и положительно определенной, то ее можно представить в виде произведения $A = S^T S$, S –

верхняя треугольная матрица, S^T – транспонированная к ней матрица (нижняя треугольная).

Значения коэффициентов матрицы *S* находятся по формулам (3), описанным в п 2.2.

В этом случае решение системы линейных алгебраических уравнений (1) сводится к последовательному решению двух систем с треугольными матрицами [19; 8]:

$$S^T Y = f, \tag{3}$$

$$SX = Y. \tag{4}$$

Т.к. матрица S^T системы (3) является нижней треугольной, то можно сразу выписать ее решение:

$$y_1 = \frac{f_1}{s_{11}} \tag{5}$$

$$y_i = \frac{\left(f_i - \sum_{j=1}^{i-1} s_{ji} y_j \right)}{s_{ii}} \quad i = 2, 3, \ldots, n. \tag{6}$$

Определив таким образом вектор *Y*, можем найти из системы (4) искомое решение. Это решение находим обратным ходом метода Гаусса, т.к. матрица *S* – верхняя треугольная. Имеем:

$$x_n = \frac{y_n}{s_{nn}}, \tag{7}$$

$$x_i = \frac{\left(y_i - \sum_{\mathrm{j=i+1}}^{\mathrm{n}} s_{\mathrm{ij}} x_{\mathrm{j}} \right)}{s_{\mathrm{ii}}} \quad i = n\text{-}1, n\text{-}2, \ldots, 1. \tag{8}$$

Пример 1. Решить систему линейных уравнений методом квадратного корня.

$$\begin{cases} 2.65x_1 - 0.52x_2 + 1.02x_3 = 1.25, \\ -0.52x_1 + 2.53x_2 + 0.46x_3 = 0.95, \\ 1.02x_1 + 0.46x_2 + 2.16x_3 = 1.33. \end{cases}$$

Решение:

Матрица системы является симметрической и положительно определенной. Вычислим коэффициенты матрицы *S*:

$$s_{11} = \sqrt{a_{11}} = \sqrt{2.65} = 1.628;$$

$$s_{12} = \frac{a_{12}}{s_{11}} = \frac{-0.52}{1.628} = -0.319;$$

$$s_{13} = \frac{a_{12}}{s_{11}} = \frac{1.02}{1.628} = 0.627;$$

$$s_{22} = \sqrt{a_{22} - s_{12}^2} = \sqrt{2.53 - (-0.319)^2} = 1.558;$$

$$s_{23} = \frac{a_{23} - (s_{12} \cdot s_{13})}{s_{22}} = \frac{0.46 - (-0.319) \cdot 0.627}{1.558} = 0.424;$$

$$s_{33} = \sqrt{a_{33} - (s_{13}^2 + s_{23}^2)} = \sqrt{2.16 - (0.627^2 + 0.424^2)} = 1.26.$$

Получим матрицу $S = \begin{pmatrix} 1.628 & -0.319 & 0.627 \\ 0 & 1.558 & 0.424 \\ 0 & 0 & 1.26 \end{pmatrix}$.

Вычислим вектор *Y*:

$$y_1 = \frac{f_1}{s_{11}} = \frac{1.25}{1.628} = 0.768;$$

$$y_2 = \frac{f_2 - s_{12} \cdot y_1}{s_{22}} = \frac{0.95 - (-0.319) \cdot 0.768}{1.558} = 0.767;$$

$$y_3 = \frac{f_3 - (s_{13} \cdot y_1 + s_{23} \cdot y_2)}{s_{33}} = \frac{1.33 - (0.627 \cdot 0.768 + 0.424 \cdot 0.767)}{1.26} = 0.415.$$

Находим искомый вектор решения *X*:

$$x_3 = \frac{y_3}{s_{33}} = \frac{0.415}{1.26} = 0.329;$$

$$x_2 = \frac{y_2 - s_{23} \cdot x_3}{s_{22}} = \frac{0.767 - 0.424 \cdot 0.329}{1.558} = 0.403;$$

$$x_1 = \frac{y_1 - (s_{12} \cdot x_2 + s_{13} \cdot x_3)}{s_{11}} = \frac{0.768 - ((-0.319) \cdot 0.403 + 0.627 \cdot 0.329)}{1.628} = 0.424;$$

$$X = \begin{bmatrix} 0.424 \\ 0.403 \\ 0.329 \end{bmatrix}.$$

Пример 2. Для системы уравнений с симметрической матрицей описать алгоритм метода квадратного корня, используя S^TS-разложение.

При описании алгоритма используются объекты и методы классов «SquareMatrix», «Vector» и «FactorizationAlgorithms».

Реализация примера на языке программирования C++ может иметь следующий вид:

```
void DirectMethodsFactorization ::squareMethod(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);

        /* S^TS – разложение. В результате факторизации имеем матрицу S – верхнюю треугольную матрицу */
         FactorizationAlgorithms FA;
         FA. STS_decomposition (A, S);

        /* Решение системы S^TY = f. */
        /* Решение системы SX = Y. Матрица S – верхняя треугольная матрица, следовательно искомый вектор решения находим обратным ходом метода Гаусса.*/
X.setElement(n-1,(Y.getElement(n-1)/S.getElement(n-1,n-1)));
        for (int i = n-2; 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
}
```

Лабораторная работа № 7

Цель: изучить метод квадратного корня для решения систем линейных алгебраических уравнений.

Задание

1. Реализуйте в классе «DirectMethodsFactorization» метод квадратного корня («squareMethod»), используя методы для S^TDS- и S^TS-разложения («SDS_decomposition», «STS_decomposition»), описанные в классе «Алгоритмы факторизации» («FactorizationAlgorithms»).

2. Решите систему линейных алгебраических уравнений методом квадратного корня ($\varepsilon = 0.001$) в соответствии с вариантом.

3. Решите ту же задачу, используя пакет для математических вычислений.

4. Сравните результат выполнения п. 2 с решением, полученным в п. 3.

Варианты заданий

№ 1

$$\begin{cases} 3.14x_1 - 0.12x_2 + 1.17x_3 = 1.27 \\ -0.12x_1 + 2.32x_2 - 1.45x_3 = 2.13 \\ 1.17x_1 - 1.45x_2 + 5.18x_3 = 3.14 \end{cases}$$

№ 2

$$\begin{cases} 2.45x_1 + 1.75x_2 - 0.24x_3 = 1.23 \\ 1.75x_1 - 3.16x_2 + 0.18x_3 = 3.43 \\ -0.24x_1 + 0.18x_2 - 1.85x_3 = -0.16 \end{cases}$$

№ 3

$$\begin{cases} 2.65x_1 - 1.27x_2 + 0.18x_3 = 2.25 \\ -1.27x_1 + 2.73x_2 - 0.46x_3 = 0.93 \\ 0.18x_1 - 0.46x_2 + 2.16x_3 = 1.33 \end{cases}$$

№ 4

$$\begin{cases} 3.23x_1 + 1.62x_2 + 0.65x_3 = 1.28 \\ 1.62x_1 - 2.33x_2 - 0.43x_3 = 0.87 \\ 0.65x_1 - 0.43x_2 + 2.16x_3 = -2.87 \end{cases}$$

№ 5

$$\begin{cases} 2.93x_1 + 1.42x_2 - 1.55x_3 = 2.48 \\ 1.42x_1 - 2.87x_2 + 0.36x_3 = -0.75 \\ -1.55x_1 + 0.36x_2 - 2.44x_3 = 1.83 \end{cases}$$

№ 6

$$\begin{cases} 3.42x_1 - 1.15x_2 + 1.07x_3 = 2.48 \\ -1.15x_1 + 3.76x_2 - 1.18x_3 = 1.15 \\ 1.07x_1 - 1.18x_2 + 2.23x_3 = 0.88 \end{cases}$$

№ 7

$$\begin{cases} 2.23x_1 - 0.71x_2 + 0.63x_3 = 1.28 \\ -0.71x_1 + 5.45x_2 - 1.34x_3 = 0.64 \\ 0.63x_1 - 1.34x_2 + 2.77x_3 = -0.87 \end{cases}$$

№ 8

$$\begin{cases} 2.63x_1 + 1.27x_2 - 0.84x_3 = 1.51 \\ 1.27x_1 + 3.65x_2 + 1.27x_3 = -0.63 \\ -0.84x_1 + 1.27x_2 - 2.21x_3 = 2.15 \end{cases}$$

№ 9

$$\begin{cases} 3.78x_1 + 1.08x_2 - 1.35x_3 = 0.35 \\ 1.08x_1 - 2.28x_2 + 0.37x_3 = 1.27 \\ -1.35x_1 + 0.37x_2 + 2.86x_3 = 0.47 \end{cases}$$

№ 10

$$\begin{cases} 4.83x_1 + 2.18x_2 - 1.73x_3 = 0.28 \\ 2.18x_1 - 4.41x_2 + 1.03x_3 = -1.18 \\ -1.73x_1 + 1.03x_2 + 2.27x_3 = 0.72 \end{cases}$$

№ 11

$$\begin{cases} 2.74x_1 - 1.18x_2 + 0.23x_3 = 0.16 \\ -1.18x_1 + 2.71x_2 - 0.52x_3 = 1.81 \\ 0.23x_1 - 0.52x_2 + 1.62x_3 = -1.25 \end{cases}$$

№ 12

$$\begin{cases} 3.35x_1 - 0.72x_2 + 1.38x_3 = 0.88 \\ -0.72x_1 + 3.45x_2 - 1.18x_3 = 1.72 \\ 1.38x_1 - 1.18x_2 + 2.93x_3 = -0.72 \end{cases}$$

№ 13

$$\begin{cases} 5.48x_1 + 0.75x_2 - 1.23x_3 = 0.83 \\ 0.75x_1 - 2.96x_2 + 1.64x_3 = -1.12 \\ -1.23x_1 + 1.64x_2 - 3.55x_3 = 0.47 \end{cases}$$

№ 14

$$\begin{cases} 2.16x_1 - 0.18x_2 + 1.26x_3 = 1.83 \\ -0.18x_1 + 4.63x_2 - 2.73x_3 = 0.54 \\ 1.26x_1 - 2.73x_2 + 5.15x_3 = 1.72 \end{cases}$$

№ 15

$$\begin{cases} 5.63x_1 - 1.72x_2 + 0.37x_3 = -0.75 \\ -1.72x_1 - 3.27x_2 + 0.62x_3 = 1.27 \\ 0.37x_1 + 0.62x_2 - 4.43x_3 = 2.74 \end{cases}$$

№ 16

$$\begin{cases} 3.36x_1 + 0.92x_2 - 1.87x_3 = 2.15 \\ 0.92x_1 - 2.24x_2 + 0.77x_3 = -2.06 \\ -1.87x_1 + 0.77x_2 - 3.16x_3 = 0.97 \end{cases}$$

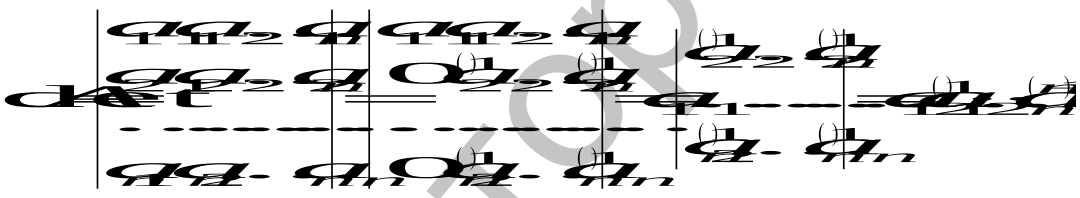
П 3.5 Вычисления определителя и нахождения обратной матрицы

Рассмотрим способы вычисления определителя.

Пусть дана матрица $A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$. Требуется вычислить ее определитель.

Для вычисления определителей матриц можно применять алгоритмы прямых методов решения систем линейных алгебраических уравнений $AX = f$ [19; 20].

Преобразования прямого хода в методе Гаусса, приводящие матрицу A системы к треугольному виду таковы, что они не изменяют определителя матрицы A. Учитывая, что определитель треугольной матрицы равен произведению диагональных элементов, имеем.

 (1)

Таким образом, определитель матрицы равен произведению всех ведущих элементов при ее преобразовании методом Гаусса [18].

Если использовать метод Гаусса с выбором главного элемента, то необходимо учесть, что при перестановке столбцов или строк знак определителя меняется на противоположный. Следовательно, значение определителя после приведения матрицы к треугольному виду будет вычисляться по формуле:

$$\det A = -1^{\ell} a_{11} a_{22}^{(1)} a_{33}^{(2)} \dots a_{nn}^{(n-1)},$$

где ℓ – сумма перестановок строк и столбцов, осуществляемых в процессе исключения.

Для нахождения определителя симметрических положительно определенных матриц применим метод квадратного корня [18]. Определитель вычисляется следующим образом:

$$\det A = s_{11}^2 \cdot s_{22}^2 \cdot \ldots \cdot s_{nn}^2 . \qquad (2)$$

Для вычисления определителя матрицы можно использовать ее LDU-разложение [6].

Представим матрицу A в виде $A = LDU$,

где $L = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix}$, $U = \begin{pmatrix} 1 & u_{12} & \cdots & u_{1n} \\ 0 & 1 & \cdots & u_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$, $D = \begin{pmatrix} d_{11} & 0 & \cdots & 0 \\ 0 & d_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & d_{nn} \end{pmatrix}$.

Определитель матрицы A вычисляется по формуле:

$$\det A = \det L \cdot \det D \cdot \det U = d_{11} \cdot d_{22} \cdot \ldots \cdot d_{nn}. \qquad (3)$$

Пример 1. Вычислить определитель матрицы $A = \begin{pmatrix} 2 & 1 & 4 \\ 3 & 2 & 1 \\ 1 & 3 & 3 \end{pmatrix}$, используя ее LDU-разложение.

Решение:

Проведем факторизацию, получаем

$$L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{3}{2} & 1 & 0 \\ \frac{1}{2} & 5 & 1 \end{pmatrix}, \; U = \begin{pmatrix} 1 & \frac{1}{2} & 2 \\ 0 & 1 & -10 \\ 0 & 0 & 1 \end{pmatrix}, \; D = \begin{pmatrix} 2 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 26 \end{pmatrix}.$$

По формуле (3) вычисляем определитель

$$\det A = \det L \cdot \det D \cdot \det U = d_{11} \cdot d_{22} \cdot \ldots \cdot d_{nn} = 2 \cdot \frac{1}{2} \cdot 26 = 26.$$

Пример 2. Вычислить определитель матрицы A по схеме метода квадратного корня

$$A = \begin{pmatrix} 1.42 & 0.54 & 0.47 & -0.22 \\ 0.54 & 1.88 & -0.35 & 0.31 \\ 0.47 & -0.35 & 1.48 & 0.57 \\ -0.22 & 0.31 & 0.57 & 1.53 \end{pmatrix}.$$

Решение:

Матрица A является симметричной положительно определенной. Используя формулы метода квадратного корня, находим матрицу S:

$$s_{11} = \sqrt{a_{11}},$$

$$s_{1j} = \frac{a_{1j}}{s_{11}}, \quad j = \overline{2,n},$$

$$s_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} s_{ki} \cdot s_{kj}}{s_{ii}}, \quad i < j, \quad i = \overline{2, j-1}, \quad j = \overline{2,n},$$

$$s_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} s_{ki}^2}, \quad i = \overline{2,n},$$

$$s_{11} = \sqrt{a_{11}} = \sqrt{1.42} = 1.1916,$$

$$s_{12} = \frac{a_{12}}{s_{11}} = \frac{0.54}{0.1916} = 0.4532,$$

$$s_{13} = \frac{0.3944}{0.1916} = 0.3944,$$

$$s_{22} = \sqrt{a_{22} - s_{12}^2} = \sqrt{1.88 - (0.4532)^2} = 1.2941,$$

$$s_{23} = \frac{a_{23} - s_{12} \cdot s_{13}}{s_{22}} = \frac{-0.35 - 0.4532 \cdot 0.3944}{1.2941} = -0.4086,$$

$$s_{24} = \frac{a_{24} - s_{12} \cdot s_{14}}{s_{22}} = \frac{0.31 - 0.4532 \cdot (-0.1844)}{1.2941}$$

и так далее.

Получаем матрицу

$$S = \begin{pmatrix} 1.1961 & 0.4532 & 0.3944 & -0.1846 \\ 0 & 1.2941 & -0.4086 & 0.3042 \\ 0 & 0 & 1.0759 & 0.713 \\ 0 & 0 & 0 & 0.946 \end{pmatrix}.$$

По формуле (2) вычисляем определитель:

$$\Delta = (1.1916)^2 \cdot (1.2941)^2 \cdot (1.0759)^2 \cdot (1.946)^2 = 2.4635.$$

Рассмотрим методы для нахождения обратной матрицы [19].

Пусть A – невырожденная матрица n-го порядка. Нахождение матрицы, обратной данной матрице A, эквивалентно решению матричного уравнения:

$$AX = E, \tag{4}$$

где $X = A^{-1}$ – искомая матрица, E – единичная матрица n-го порядка.

Уравнение (1) можно записать в виде системы n^2 уравнений:

$$\sum_{k=1}^{n} a_{ik} x_{kj} = \delta_{ij}, \quad i, j = 1, 2, \ldots, n, \tag{5}$$

где $\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$ – символ Кронекера. В развернутом виде (5) выглядит следующим образом:

$$\begin{bmatrix} a_{11}\ a_{12} \ldots a_{1n} \\ a_{21}\ a_{22} \ldots a_{2n} \\ \ldots\ldots\ldots\ldots \\ a_{n1}\ a_{n2} \ldots a_{nn} \end{bmatrix} \begin{bmatrix} x_{11}\ x_{12} \ldots x_{1n} \\ x_{21}\ x_{22} \ldots x_{2n} \\ \ldots\ldots\ldots\ldots \\ x_{n1}\ x_{n2} \ldots x_{nn} \end{bmatrix} = \begin{bmatrix} 1\ 0\ \ldots\ 0 \\ 0\ 1\ \ldots\ 0 \\ \ldots\ldots\ldots \\ 0\ 0\ \ldots\ 1 \end{bmatrix}.$$

Система (5) распадается на n независимых систем линейных алгебраических уравнений с одной и той же матрицей A, но с различными правыми частями

$$AX^{(j)} = \delta^{(j)}; \ j = 1, 2, \ldots, n, \tag{6}$$

где $X^{(j)} = \begin{bmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{nj} \end{bmatrix}$; $\delta^{(j)} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$ j-ая компонента.

Полученные системы (6) можно решать одновременно методом Гаусса. При этом, т.к. все системы имеют одну и ту же матрицу A, достаточно один раз совершить прямой ход. Но для каждой системы (6) делается обратный ход.

Для обращения матрицы весьма эффективным является метод Жордана-Гаусса [19].

Пример 3. Найти обратную матрицу, используя метод Жордана-Гаусса.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 2 \\ 3 & 2 & 1 \end{pmatrix}.$$

Решение:

Составим объединенную таблицу:

A			e_1	e_2	e_3
1	2	3	1	0	0
2	1	2	0	1	0
3	2	1	0	0	1

Прибавим первую строку ко второй строке, умножив на –2 и к третьей строке, умножив на –3:

A			e_1	e_2	e_3
1	2	3	1	0	0
0	–3	–4	–2	1	0
0	–4	–8	–3	0	1

Делим коэффициенты во второй строке на –3, чтобы значение коэффициента a_{22} было равно 1:

A			e_1	e_2	e_3
1	2	3	1	0	0
0	1	1.333	0.667	–0.333	0
0	–4	–8	–3	0	1

Прибавим вторую строку таблицы к первой и третьей, умножив на –2 и 4 соответственно:

A			e_1	e_2	e_3
1	0	0.333	–0.333	0.667	0
0	1	1.333	0.667	–0.333	0
0	0	1	0.125	0.5	–0.375

Прибавим третью строку таблицы к первой и ко второй, умножив на –0.333 и –1.333 соответственно:

A			e_1	e_2	e_3
1	0	0	–0.375	0.5	0.125
0	1	0	0.5	–1	0.5
0	0	1	0.125	0.5	–0.375

В полученной таблице столбцы e_1, e_2 и e_3 составляют искомую обратную матрицу

$$A^{-1} = \begin{pmatrix} -0.375 & 0.5 & 0.125 \\ 0.5 & -1 & 0.5 \\ 0.125 & 0.5 & -0.375 \end{pmatrix}.$$

В результате применения метода Жордана-Гаусса на основе факторизации (см. п 3.3) получим матрицу $E = L_n L_{n-1} .. L_2 L_1 A$, откуда следует, что $A^{-1} = L_n L_{n-1} .. L_2 L_1$. Таким образом, достигается разложение обратной матрицы на элементарные сомножители.

Находить обратную матрицу можно, исходя из ее LU-разложения [19].

Имеем невырожденную квадратную матрицу A. Представим ее в виде $A = LU$, где L – матрица вида $L = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ l_{n1} & l_{n2} & \cdots & 1 \end{pmatrix}$,

U – матрица вида $U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix}$.

Для нахождения обратной матрицы необходимо решить систему (7)

$$LUX = E, \qquad (7)$$

где E – единичная матрица, $X = A^{-1}$ – искомая матрица.

Представим единичную матрицу как совокупность векторстолбцов:

$$e_1 = \begin{pmatrix} 1 \\ 0 \\ \ldots \\ 0 \end{pmatrix}, \; e_2 = \begin{pmatrix} 0 \\ 1 \\ \ldots \\ 0 \end{pmatrix}, \; \ldots, \; e_n = \begin{pmatrix} 0 \\ 0 \\ \ldots \\ 1 \end{pmatrix}.$$

Решение системы (7) состоит из двух этапов.

На первом этапе производится решение систем следующего вида: $LY_1 = e_1, \; LY_2 = e_2, \; ..., \; LY_n = e_n$. Решением являются вектора $Y_i, \; i = 1, \; ..., \; n$

На втором этапе необходимо решить системы вида: $UX_1 = Y_1, \; UX_2 = Y_2, \; ..., \; UX_n = Y_n$.

Искомая обратная матрица состоит из векторов-решений $X_1, \; X_2, \; ..., \; X_n$.

Пример 4. Найти обратную матрицу для матрицы $A = \begin{pmatrix} 3 & 1 & 2 \\ 7 & 1 & 5 \\ 2 & 4 & 3 \end{pmatrix}$, используя LU-разложение.

Решение:

Проведем факторизацию

$$u_{11} = a_{11} = 3; \quad u_{12} = a_{12} = 1; \quad u_{13} = a_{13} = 2;$$

При к=1:

$$l_{21} = \frac{a_{21}}{u_{11}} = \frac{7}{3}; \quad l_{31} = \frac{a_{31}}{u_{11}} = \frac{2}{3};$$

$$u_{22} = a_{22} - l_{21}u_{12} = 1 - \frac{7}{3} \cdot 1 = -\frac{4}{3};$$

При к=2:

$$l_{32} = \frac{a_{32} - l_{31} \cdot u_{12}}{u_{22}} = \frac{4 - \frac{2}{3} \cdot 1}{-\frac{4}{3}} = \frac{-5}{2};$$

$$u_{23} = a_{23} - l_{21} \cdot u_{13} = 5 - \frac{7}{3} \cdot 2 = \frac{1}{3};$$

При к=3: $u_{33} = a_{33} - l_{31} \cdot u_{13} - l_{32} \cdot u_{23} = 3 - \frac{2}{3} \cdot 2 + \frac{5}{2} \cdot \frac{1}{3} = \frac{5}{2}.$

В результате получены две треугольные матрицы:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{7}{3} & 1 & 0 \\ \frac{2}{3} & \frac{-5}{2} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 3 & 1 & 2 \\ 0 & -\frac{4}{3} & \frac{1}{3} \\ 0 & 0 & \frac{5}{2} \end{pmatrix}.$$

Решим системы $LY_1 = e_1$, $LY_2 = e_2$, $LY_3 = e_3$, где $e_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$,

$e_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$, $e_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$.

Составим объединенную таблицу:

L			e_1	e_2	e_3
1	0	0	1	0	0
7/3	1	0	0	1	0
2/3	–5/2	1	0	0	1

Прибавим первую строку ко второй строке, умножив на –7/3 и к третьей строке, умножив на –2/3:

L			e_1	e_2	e_3
1	0	0	1	0	0
0	1	0	–7/3	1	0
0	–5/2	1	–2/3	0	1

Прибавим вторую строку к третьей строке, умножив на 5/2:

L			e_1	e_2	e_3
1	0	0	1	0	0
0	1	0	–7/3	1	0
0	0	1	–13/2	5/2	1

Следовательно $Y_1 = \begin{pmatrix} 1 \\ -7/3 \\ -13/2 \end{pmatrix}$, $Y_2 = \begin{pmatrix} 0 \\ 1 \\ 5/2 \end{pmatrix}$, $Y_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$.

Теперь решаем системы $UX_1 = Y_1$, $UX_2 = Y_2$, $UX_3 = Y_3$.

Объединенная таблица имеет вид:

U			Y_1	Y_2	Y_3
3	1	2	1	0	0
0	–4/3	1/3	–7/3	1	0
0	0	5/2	–13/2	5/2	1

Третью строку делим на 5/2:

U			Y_1	Y_2	Y_3
3	1	2	1	0	0
0	–4/3	1/3	–7/3	1	0
0	0	1	–13/5	1	2/5

Прибавим третью строку ко второй, умножив на –1/3, и к первой, умножив на –2:

U			Y_1	Y_2	Y_3
3	1	0	31/5	–2	–4/5
0	–4/3	0	–22/15	2/3	–2/15
0	0	1	–13/5	1	2/5

Вторую строку делим на –4/3:

U			Y_1	Y_2	Y_3
3	1	0	31/5	–2	–4/5
0	1	0	11/10	–1/2	1/10
0	0	1	–13/5	1	2/5

Прибавим вторую строку к первой, умножив на –1:

U			Y_1	Y_2	Y_3
3	0	0	51/10	–3/2	–9/10
0	1	0	11/10	–1/2	1/10
0	0	1	–13/5	1	2/5

Разделим первую строку на 3:

U			Y_1	Y_2	Y_3
1	0	0	17/10	–1/2	–3/10
0	1	0	11/10	–1/2	1/10
0	0	1	–13/5	1	2/5

Следовательно, столбцы $X_1 = \begin{pmatrix} 17/10 \\ 11/10 \\ -13/5 \end{pmatrix}$ $X_2 = \begin{pmatrix} -1/2 \\ -1/2 \\ 1 \end{pmatrix}$ $X_3 = \begin{pmatrix} -3/10 \\ 1/10 \\ 2/5 \end{pmatrix}$

и составляют обратную матрицу $A^{-1} = \begin{pmatrix} \frac{17}{10} & \frac{-1}{2} & \frac{-3}{10} \\ \frac{11}{10} & \frac{-1}{2} & \frac{1}{10} \\ \frac{-13}{5} & 1 & \frac{2}{5} \end{pmatrix}$.

Лабораторная работа № 8

Цель: изучить методы для вычисления определителя и нахождения обратной матрицы.

Задание

1. Дополните класс «Квадратная матрица» («SquareMatrix») методами «determinant» (определитель матрицы) и «inverseMatrix» (обратная матрица). Для нахождения определителя и обратной матрицы используйте LDU и LU-разложение соответственно.

2. Найдите определитель матрицы ($\varepsilon = 0.001$) в соответствии с вариантом.

3. Обратите матрицу ($\varepsilon = 0.001$) в соответствии с вариантом.

4. Решите те же задачи, используя пакет для математических вычислений.

5. Сравните результат выполнения п. 2 и п. 3 с решением, полученным в п. 4.

Варианты заданий

№ 1

$$A = \begin{pmatrix} 1 & 4 & 1 & 3 \\ 0 & -1 & 3 & -1 \\ 3 & 1 & 0 & 2 \\ 1 & -2 & 5 & 1 \end{pmatrix}$$

№ 2

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 4 & 2 & 3 \\ 1 & 10 & 3 & 6 \\ 6 & 10 & 1 & 4 \end{pmatrix}$$

№ 3

$$A = \begin{pmatrix} 1 & 2 & 3 & -2 \\ 2 & -1 & -2 & -3 \\ 3 & 2 & -1 & 2 \\ 2 & -3 & 2 & 1 \end{pmatrix}$$

№ 4

$$A = \begin{pmatrix} -2 & 2 & 1 & 0 \\ 1 & -3 & 3 & 7 \\ 2 & -1 & 2 & -3 \\ -5 & 4 & -1 & 2 \end{pmatrix}$$

№ 5

$$A = \begin{pmatrix} 1 & -1 & -1 & 1 \\ -1 & 2 & 2 & 0 \\ 0 & -1 & 1 & 4 \\ 1 & 1 & -1 & -1.5 \end{pmatrix}$$

№ 6

$$A = \begin{pmatrix} 3 & -2 & 2 & 0 \\ 2 & 1 & 1 & -2 \\ 3 & -1 & 2 & 1 \\ 1 & 2 & -1 & -1 \end{pmatrix}$$

№ 7

$$A = \begin{pmatrix} 5 & -4 & 0 & 2 \\ -1 & 1 & 1 & -1 \\ 2 & 3 & 1 & -6 \\ 1 & 0 & 2 & -1 \end{pmatrix}$$

№ 8

$$A = \begin{pmatrix} 4 & -1 & 0 & 1 \\ 3 & 2 & -1 & 2 \\ 0 & 2 & 2 & 1 \\ -1 & 1 & -3 & -1 \end{pmatrix}$$

№ 9

$$A = \begin{pmatrix} 1 & 2 & 0 & 1 \\ -1 & -3 & 3 & -1 \\ 0 & 4 & -10 & 2 \\ 1 & -1 & 2 & -1 \end{pmatrix}$$

№ 10

$$A = \begin{pmatrix} 1 & 4 & -3 & 0 \\ 0 & 4 & 1 & 2 \\ -1 & 2 & 4 & 1 \\ 1 & 0 & -1 & 5 \end{pmatrix}$$

№ 11

$$A=\begin{pmatrix} 2 & -1 & 1 & 2 \\ 1 & 2 & -1 & 1 \\ 3 & 0 & -1 & -3 \\ 1 & -1 & 1 & 3 \end{pmatrix}$$

№ 12

$$A=\begin{pmatrix} 2 & 1 & 2 & 0 \\ -1 & -3 & 3 & -1 \\ 1 & 3 & -8 & 1 \\ 1 & -1 & 2 & -1 \end{pmatrix}$$

№ 13

$$A=\begin{pmatrix} 2 & 3 & 0 & 1 \\ -1 & 1 & 3 & 0 \\ 0 & 2 & -1 & 1 \\ 3 & -1 & 1 & -2 \end{pmatrix}$$

№ 14

$$A=\begin{pmatrix} -1 & 3 & 3 & 2 \\ -2 & 2 & 2 & 1 \\ 0 & 1 & 2 & 0 \\ -1 & 3 & 3 & 3 \end{pmatrix}$$

№ 15

$$A=\begin{pmatrix} 1 & 0 & 3 & 4 \\ 0 & 1 & 5 & 1 \\ -3 & 4 & 10 & 1 \\ 0 & -6 & 0 & -1 \end{pmatrix}$$

№ 16

$$A=\begin{pmatrix} 3 & 1 & 3 & 3 \\ 2 & 2 & 1 & 3 \\ 1 & 0 & 2 & 0 \\ 1 & 1 & 1 & 3 \end{pmatrix}$$

П 3.6 Решение системы линейных алгебраических уравнений методом прогонки

Рассмотрим метод прогонки. Этот метод применим в случае, когда матрица системы является трехдиагональной.

Имеем систему линейных алгебраических уравнений с трехдиагональной матрицей:

$$\begin{cases} b_1x_1 + c_1x_2 = f_1, \\ a_2x_1 + b_2x_2 + c_2x_3 = f_2, \\ a_3x_2 + b_3x_3 + c_3x_4 = f_3, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n-1}x_{n-2} + b_{n-1}x_{n-1} + c_{n-1}x_n = f_{n-1}, \\ a_n\ x_{n-1}\ +\ b_nx_n\ = f_n. \end{cases} \qquad (1)$$

Достаточным условием устойчивости метода прогонки является условие преобладания диагональных элементов в матрице A, в которой $a_i \neq 0,\ c_i \neq 0,\ i = 2,\ 3,\ \ldots,\ n-1$:

$$|b_i| \geq |a_i| + |c_i|,\ i = 1,\ 2,\ \ldots,\ n,$$

причем строгое неравенство имеет место хотя бы при одном i [7; 8].

Решение системы будем искать в виде [4; 18]

$$x_i = A_i x_{i+1} + B_i, \quad i = 1,\ 2,\ ...,\ n-1, \tag{2}$$

где A_i, B_i, $i = 1,\ 2,\ ...,\ n$ – прогоночные коэффициенты.

Для их определения выразим из первого уравнения системы (1) x_1 через x_2, получим:

$$x_1 = \frac{-c_1}{b_1} x_2 + \frac{f_1}{b_1} = A_1 x_2 + B_1, \tag{3}$$

откуда $A_1 = \frac{-c_1}{b_1}, \quad B_1 = \frac{f_1}{b_1}.$ (4)

Из второго уравнения системы (1) с помощью (3) выразим x_2 через x_3, получим:

$$x_2 = \frac{-c_2}{b_2 + a_2 A_1} x_3 + \frac{f_2 - a_2 B_1}{b_2 + a_2 A_1} = A_2 x_2 + B_2,$$

откуда $A_2 = \frac{-c_2}{b_2 + a_2 A_1}, \quad B_2 = \frac{f_2 - a_2 B_1}{b_2 + a_2 A_1}.$ (5)

Продолжая этот процесс, получим из *i*-го уравнения системы (1)

$$x_i = \frac{-c_i}{b_i + a_i A_{i-1}} x_{i+1} + \frac{f_i - a_i B_{i-1}}{b_i + a_i A_{i-1}}, \quad i = 1, 2, \ ..., n-1. \tag{6}$$

следовательно $A_i = \frac{-c_i}{b_i + a_i A_{i-1}}, \quad B_i = \frac{f_i - a_i B_{i-1}}{b_i + a_i A_{i-1}}, \quad i = 2, ..., n$ (7)

Таким образом, прямой ход метода прогонки по определению прогоночных коэффициентов завершен. Коэффициенты A_i, B_i, $i = 1,\ 2,\ ...,\ n$ находятся по формулам (4), (7).

Обратный ход прогонки состоит в нахождении неизвестных x_n, x_{n-1}, ..., x_1

$$x_n = \frac{f_n - a_n B_{n-1}}{b_n + a_n A_{n-1}} \tag{8}$$

и далее, используя формулу (2) и значения прогоночных коэффициентов (4) , (7), последовательно вычисляем все неизвестные x_{n-1}, ..., x_1.

Рассмотренный метод (4), (7), (8) называется правой прогонкой. В этом случае определение неизвестных происходит в направлении убывания индексов.

Аналогично, начиная с последнего уравнения СЛАУ (1), можно вывести формулы левой прогонки (9) – (12). В этом алгоритме значение неизвестных находятся в направлении возрастания индексов [4; 19].

$$A_n^* = -\frac{a_n}{b_n},\ B_n^* = \frac{f_n}{b_n} \tag{9}$$

$$A_{i-1}^* = \frac{-a_i}{c_i A_i^* + b_i},\ B_{i-1}^* = \frac{f_i - c_i B_i^*}{c_i A_i^* + b_i},\quad i = n-1,\ n-2,\ ....,\ 1 \tag{10}$$

$$x_1 = \frac{f_1 - b_1 B_1^*}{c_1 A_1^* + b_1} \tag{11}$$

$$x_{i+1} = A_i^* x_i + B_i^*,\qquad i = 1,\ 2,\ ...,\ n-1 \tag{12}$$

Пример 1. Решить систему уравнений методом прогонки:

$$\begin{cases} 2x_1 + x_2 = 4, \\ 2x_1 + 3x_2 - x_3 = 9, \\ x_2 - x_3 + 3x_4 = 12, \\ x_3 - x_4 = -4. \end{cases}$$

Решение:

Представим систему в матричном виде

$$\begin{pmatrix} 2 & 1 & 0 & 0 \\ 2 & 3 & -1 & 0 \\ 0 & 1 & -1 & 3 \\ 0 & 0 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 9 \\ 12 \\ -4 \end{pmatrix}.$$

Прямой ход. Вычисляем прогоночные коэффициенты:

$$A_1 = -\frac{c_1}{b_1} = -\frac{1}{2},$$

$$B_1 = \frac{f_1}{b_1} = \frac{4}{2} = 2,$$

$$A_2 = -\frac{c_2}{A_1 a_2 + b_2} = -\frac{-1}{-\frac{1}{2}\cdot 2 + 3} = \frac{1}{2},$$

$$B_2 = \frac{f_2 - a_2 B_1}{A_1 a_2 + b_2} = \frac{9 - 2\cdot 2}{-\frac{1}{2}\cdot 2 + 3} = \frac{5}{2},$$

$$A_3 = -\frac{c_3}{A_2 a_3 + b_3} = -\frac{3}{\frac{1}{2}\cdot 1 - 1} = 6,$$

$$B_3 = \frac{f_3 - a_3 B_2}{A_2 a_3 + b_3} = \frac{12 - 1\cdot\frac{5}{2}}{\frac{1}{2}\cdot 1 - 1} = -19.$$

Обратный ход. Вычисляем неизвестные x_i:

$$x_4 = \frac{f_4 - a_4 B_3}{b_4 + a_4 A_3} = \frac{-4 - 1\cdot(-19)}{-1 + 1\cdot 6} = 3,$$

$$x_3 = A_3 x_4 + B_3 = 6\cdot 3 - 19 = -1,$$

$$x_2 = A_2 x_3 + B_2 = \frac{1}{2}\cdot(-1) + \frac{5}{2} = 2,$$

$$x_1 = A_1 x_2 + B_1 = -\frac{1}{2}\cdot 2 + 2 = 1.$$

Лабораторная работа № 9

Цель: изучить метод прогонки для решения системы линейных алгебраических уравнений.

Задание

1. В классе «Алгоритмы решения СЛАУ, не использующие факторизацию» («DirectMethodsNF») реализуйте метод прогонки («sweepMethod»). В методе прогонки матрица системы представляет собой объект класса «ThreeDiagonalMatrix», а прогоночные коэффициенты – объекты класса «Vector». Для работы с матричными объектами используйте методы, реализованные в данных классах.

2. Решите систему линейных алгебраических уравнений методом прогонки ($\varepsilon = 0.001$) в соответствии с вариантом.

3. Решите ту же задачу, используя пакет для математических вычислений.

4. Сравните результат выполнения п. 2 с решением, полученным в п. 3.

Варианты заданий

№ 1

$$\begin{cases} 3.14x_1 - 0.12x_2 = 1.27 \\ -0.12x_1 + 2.32x_2 - 1.45x_3 = 2.13 \\ -1.45x_2 + 5.18x_3 = 3.14 \end{cases}$$

№ 2

$$\begin{cases} 2.45x_1 + 1.75x_2 = 1.23 \\ 1.75x_1 - 3.16x_2 + 0.18x_3 = 3.43 \\ 0.18x_2 - 1.85x_3 = -0.16 \end{cases}$$

№ 3

$$\begin{cases} 2.65x_1 - 1.27x_2 = 2.25 \\ -1.27x_1 + 2.73x_2 - 0.46x_3 = 0.93 \\ -0.46x_2 + 2.16x_3 = 1.33 \end{cases}$$

№ 4

$$\begin{cases} 3.23x_1 + 1.62x_2 = 1.28 \\ 1.62x_1 - 2.33x_2 - 0.43x_3 = 0.87 \\ -0.43x_2 + 2.16x_3 = -2.87 \end{cases}$$

№ 5

$$\begin{cases} 2.93x_1 + 1.42x_2 = 2.48 \\ 1.42x_1 - 2.87x_2 + 0.36x_3 = -0.75 \\ 0.36x_2 - 2.44x_3 = 1.83 \end{cases}$$

№ 6

$$\begin{cases} 3.42x_1 - 1.15x_2 = 2.48 \\ -1.15x_1 + 3.76x_2 - 1.18x_3 = 1.15 \\ -1.18x_2 + 2.23x_3 = 0.88 \end{cases}$$

№ 7

$$\begin{cases} 2.23x_1 - 0.71x_2 = 1.28 \\ -0.71x_1 + 5.45x_2 - 1.34x_3 = 0.64 \\ -1.34x_2 + 2.77x_3 = -0.87 \end{cases}$$

№ 8

$$\begin{cases} 2.63x_1 + 1.27x_2 = 1.51 \\ 1.27x_1 + 3.65x_2 + 1.27x_3 = -0.63 \\ 1.27x_2 - 2.21x_3 = 2.15 \end{cases}$$

№ 9

$$\begin{cases} 3.78x_1 + 1.08x_2 = 0.35 \\ 1.08x_1 - 2.28x_2 + 0.37x_3 = 1.27 \\ 0.37x_2 + 2.86x_3 = 0.47 \end{cases}$$

№ 10

$$\begin{cases} 4.83x_1 + 2.18x_2 = 0.28 \\ 2.18x_1 - 4.41x_2 + 1.03x_3 = -1.18 \\ 1.03x_2 + 2.27x_3 = 0.72 \end{cases}$$

№ 11

$$\begin{cases} 2.74x_1 - 1.18x_2 = 0.16 \\ -1.18x_1 + 2.71x_2 - 0.52x_3 = 1.81 \\ -0.52x_2 + 1.62x_3 = -1.25 \end{cases}$$

№ 12

$$\begin{cases} 3.35x_1 - 0.72x_2 = 0.88 \\ -0.72x_1 + 3.45x_2 - 1.18x_3 = 1.72 \\ -1.18x_2 + 2.93x_3 = -0.72 \end{cases}$$

№ 13

$$\begin{cases} 5.48x_1 + 0.75x_2 = 0.83 \\ 0.75x_1 - 2.96x_2 + 1.64x_3 = -1.12 \\ 1.64x_2 - 3.55x_3 = 0.47 \end{cases}$$

№ 14

$$\begin{cases} 2.16x_1 - 0.18x_2 = 1.83 \\ -0.18x_1 + 4.63x_2 - 2.73x_3 = 0.54 \\ -2.73x_2 + 5.15x_3 = 1.72 \end{cases}$$

№ 15

$$\begin{cases} 5.63x_1 - 1.72x_2 = -0.75 \\ -1.72x_1 - 3.27x_2 + 0.62x_3 = 1.27 \\ 0.62x_2 - 4.43x_3 = 2.74 \end{cases}$$

№ 16

$$\begin{cases} 3.36x_1 + 0.92x_2 = 2.15 \\ 0.92x_1 - 2.24x_2 + 0.77x_3 = -2.06 \\ 0.77x_2 - 3.16x_3 = 0.97 \end{cases}$$

П 3.7 Метод простых итераций решения систем линейных алгебраических уравнений

Рассмотрим метод простых итераций [19]. Имеем систему линейных алгебраических уравнений

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = f_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = f_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = f_n \end{cases} \quad (1)$$

или в матричном виде $AX = f$. (2)

Приведем систему (1) к эквивалентному виду

$$\begin{cases} x_1 = g_1 + b_{11}x_1 + b_{12}x_2 + \ldots + b_{1n}x_n, \\ x_2 = g_2 + b_{21}x_1 + b_{22}x_2 + \ldots + b_{2n}x_n, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n = g_n + b_{n1}x_1 + b_{n2}x_2 + \ldots + b_{nn}x_n \end{cases} \quad (3)$$

или в матричной форме

$$X = g + BX, \quad (4)$$

где $X = \begin{pmatrix} x_1 \\ \ldots \\ x_n \end{pmatrix}$, $g = \begin{pmatrix} g_1 \\ \ldots \\ g_n \end{pmatrix}$, $B = \begin{pmatrix} b_{11} & \ldots & b_{1n} \\ \ldots & \ldots & \ldots \\ b_{n1} & \ldots & b_{nn} \end{pmatrix}$.

Такое приведение может быть выполнено различными способами.

Итерационную последовательность векторов

$$X^{(k+1)} = g + BX^{(k)}, \quad k = 0,\ 1,\ 2,\ \ldots . \quad (5)$$

называют методом простых итераций. Вектор начального приближения $X^{(0)}$ выбирается произвольно. В качестве начального приближения вектора неизвестных $X^{(0)}$ можно также принимать вектор правых частей: $X^{(0)} = g$, т.е. $(x_1^{(0)},\ x_2^{(0)},\ \ldots,\ x_n^{(0)})^T = (g_1,\ g_2,\ \ldots,\ g_n)^T$.

В качестве критерия сходимости метода простых итераций имеет место следующее утверждение [20]:

Для того чтобы метод простой итерации сходился при любом начальном приближении $X^{(0)}$, необходимо и достаточно, чтобы $|\lambda_i\ B| < 1$, где $\lambda_i\ B$ – все собственные значения матрицы B. Метод будет сходиться также в случае

$$\|B\| < 1, \tag{6}$$

так как $|\lambda_i \ B| < \forall \|B\|$, где $\|B\|$ – первая или вторая норма матрицы.

Таким образом выбор матрицы B для системы (5) метода простой итерации должен подчиняться требованиям сходимости и не может быть совсем произвольным.

Если матрица A имеет диагональное преобладание, т.е. ее элементы удовлетворяют неравенствам

$$|a_{ii}| > \sum_{j=1}^{n} |a_{ij}| \quad j \neq i, \ i = 1, \ 2, \ ..., \ n, \tag{7}$$

то в качестве матрицы B можно задать матрицу с элементами

$$b_{ij} = \begin{cases} -\dfrac{a_{ij}}{a_{ii}}, & i, \ j = 1, \ 2, \ ..., \ n \quad i \neq j, \\ 0, & i = j, \ i = 1, \ 2, \ ..., \ n \end{cases} \tag{8}$$

В этом случае систему (1) в эквивалентном виде (3) можно записать следующим образом:

$$x_i = -\sum_{j}^{n} \left(a_{ij} \frac{x_j}{a_{ii}} \right) + \frac{f_i}{a_{ii}} \quad j \neq i, \ i = 1, \ 2, \ ..., \ n, \tag{9}$$

т.е.

$$b_{ii} = 0, \ b_{ij} = -\frac{a_{ij}}{a_{ii}} \quad \text{ïðè} \ \ j \neq i, \quad g_i = \frac{f_i}{a_{ii}}, \ i = 1, \ 2, \ ..., \ n.$$

Описанная модификация метода простой итерации, связанная с делением уравнений на диагональные элементы матрицы системы, называется методом Якоби [10].

Если $\|B\| < 1$, то для метода простых итераций известна оценка погрешности [18]:

$$\left\| X^* - X^{k+1} \right\| \leq \|B\|^{k+1} \left\| X^0 \right\| + \frac{\|B\|^{k+1}}{1 - \|B\|} \|g\|, \tag{10}$$

где X^* – точное решение.

Выход из итерационного процесса осуществляется по результатам выполнения неравенства

$$\frac{\|B\|}{1 - \|B\|} \left\| X^{k+1} - X^k \right\| \leq \varepsilon, \tag{11}$$

где $\varepsilon > 0$ – заданная точность, которую необходимо достигнуть при решении исходной задачи и $\|B\| < 1$. Заменив (11) более простым условием $\left\| X^{k+1} - X^{k} \right\| \le \varepsilon$ и используя понятие первой нормы вектора, получим условие прерывания итерационного процесса в виде

$$\max_{1 \le i \le n} \left| X_i^{k+1} - X_i^{k} \right| < \varepsilon . \qquad (12)$$

Пример 1. Методом простых итераций решить систему линейных уравнений с точностью $\varepsilon = 0.01$, приведя ее к виду, удобному для итераций.

$$\begin{cases} 2x_1 + 2x_2 + 10x_3 = 14, \\ 10x_1 + x_2 + x_3 = 12, \\ 2x_1 + 10x_2 + x_3 = 13. \end{cases}$$

Решение:

Очевидно, что матрица этой системы не удовлетворяет условиям диагонального преобладания. Переставим уравнения местами так, чтобы выполнялось условие преобладания диагональных элементов.

$$\begin{cases} 10x_1 + x_2 + x_3 = 12, \\ 2x_1 + 10x_2 + x_3 = 13, \\ 2x_1 + 2x_2 + 10x_3 = 14. \end{cases}$$

Преобразуем эту систему к эквивалентному виду (5). Для этого выразим из первого уравнения x_1, из второго x_2, из третьего x_3, получаем

$$\begin{cases} x_1 = -0.1x_2 - 0.1x_3 + 1.2, \\ x_2 = -0.2x_1 - 0.1x_3 + 1.3, \\ x_3 = -0.2x_1 - 0.2x_2 + 1.4. \end{cases}$$

Имеем

$$B = \begin{pmatrix} 0 & -0.1 & -0.1 \\ -0.2 & 0 & -0.1 \\ -0.2 & -0.2 & 0 \end{pmatrix}, \quad g = \begin{pmatrix} 1.2 \\ 1.3 \\ 1.4 \end{pmatrix}.$$

Заметим, что $\|B\|_1 = \max\{0.2;\ 0.3;\ 0.4\} = 0.4 < 1$, следовательно, условие сходимости выполнено.

Зададим вектор начального приближения $X^{0} = g = \begin{pmatrix} 1.2 \\ 1.3 \\ 1.4 \end{pmatrix}$.

Выполним расчеты по формуле (6)

$$\begin{cases} x_1^{(k+1)} = -0.1x_2^{(k)} - 0.1x_3^{(k)} + 1.2, \\ x_2^{(k+1)} = -0.2x_1^{(k)} - 0.1x_3^{(k)} + 1.3 \quad k = 0,\ 1,\ ..., \\ x_3^{(k+1)} = -0.2x_1^{(k)} - 0.2x_2^{(k)} + 1.4. \end{cases}$$

На первой итерации имеем систему

$$\begin{cases} x_1^{(1)} = -0.1x_2^{(0)} - 0.1x_3^{(0)} + 1.2 = 0.93, \\ x_2^{(1)} = -0.2x_1^{(0)} - 0.1x_3^{(0)} + 1.3 = 0.92, \\ x_3^{(1)} = -0.2x_1^{(0)} - 0.2x_2^{(0)} + 1.4 = 0.90. \end{cases}$$

$$X^{(1)} = \begin{bmatrix} 0.93 \\ 0.92 \\ 0.90 \end{bmatrix}.$$

Условие окончания итерационного процесса $\max\limits_{1 \le i \le 3} \left| x_i^{1} - x_i^{0} \right| < \varepsilon$ не выполняется, значит продолжаем процесс. На второй итерации получаем систему

$$\begin{cases} x_1^{(2)} = -0.1x_2^{(1)} - 0.1x_3^{(1)} + 1.2 = 1.018, \\ x_2^{(2)} = -0.2x_1^{(1)} - 0.1x_3^{(1)} + 1.3 = 1.024, \\ x_3^{(2)} = -0.2x_1^{(1)} - 0.2x_2^{(1)} + 1.4 = 1.03. \end{cases}$$

$$X^{(2)} = \begin{bmatrix} 1.018 \\ 1.024 \\ 1.03 \end{bmatrix}.$$

Условие окончания итерационного процесса $\max\limits_{1 \le i \le 3} \left| x_i^{2} - x_i^{1} \right| < \varepsilon$ не выполняется, продолжаем итерировать до требуемой точности.

На пятой итерации требуемая точность достигнута

$$X = \begin{bmatrix} 0.9996 \\ 0.9995 \\ 0.9993 \end{bmatrix}.$$

Пример 2. Описать алгоритм метода простой итерации на языке программирования C++.

При описании алгоритма используются объекты и методы классов «SquareMatrix», «Vector».

```
void IterationMethods :: simpleIterationMethod(SquareMatrix A, Vector f, double
tochnost){

    /* Создание вспомогательной матрицы B, векторов g, x1 и вектора решений x
*/

      SquareMatrix B = SquareMatrix(A.getRowCount());
      Vector g = Vector(A.getColCount());
      Vector x = Vector(A.getColCount());
      Vector x1 = Vector(A.getColCount());

      /* Получение канонического вида системы: X = BX + g, т.е. матрицы B и g
*/
      for (int i = 0; i < A.getRowCount(); i++){
        for (int j = 0; j < A.getColCount(); j++){
          if ( i == j){
            B.setElement(i, j, 0);
                }
          if (i != j){
  B.setElement(i,j,-A.getElement(i,j)/A.getElement(i, i));
                }
           }
     g.setElement(i, f.getElement(i)/A.getElement(i, i));
       }

      /* Построение итерационной последовательности и вычисление значения
вектора на текущей итерации по формуле X(k+1) = BX(k) + g    */
 .....
      /*Вывод вектора решений*/
 }
```

Данный алгоритм основан на матричном представлении метода простой итерации. Однако можно обойтись без построения матрицы *B* и вектора *g*, воспользовавшись непосредственно формулами (10) для описания итерационного процесса. Это позволит сэкономить вычислительные ресурсы.

Лабораторная работа № 10

Цель: изучить метод простых итераций решения систем линейных алгебраических уравнений.

Задание

1. В классе «Итерационные методы решения СЛАУ» («IterationMethods») реализуйте метод простых итераций («simpleIterationMethod»). Для реализации методов используйте объекты и методы матричных классов «SquareMatrix», «Vector».

2. Решите систему линейных алгебраических уравнений методом простых итераций ($\varepsilon = 0.001$) в соответствии с вариантом.

3. Решите те же задачи, используя пакет для математических вычислений.

4. Сравните результат выполнения п. 2 с решением, полученным в п. 3.

Варианты заданий

№ 1

$$\begin{cases} 2.7x_1 + 0.3x_2 + 1.3x_3 = 2.1 \\ 0.5x_1 - 3.7x_2 + 2.8x_3 = 1.7 \\ 1.1x_1 + 2.8x_2 - 5.8x_3 = 0.8 \end{cases}$$

№ 2

$$\begin{cases} 5.7x_1 + 2.8x_2 + 1.9x_3 = 0.7 \\ 0.1x_1 + 3.4x_2 + 1.8x_3 = 1.1 \\ 0.2x_1 - 1.7x_2 + 1.3x_3 = 2.8 \end{cases}$$

№ 3

$$\begin{cases} 3.1x_1 + 0.8x_2 + 1.9x_3 = 0.2 \\ 1.9x_1 + 3.1x_2 + 1.1x_3 = 2.1 \\ 0.5x_1 + 3.8x_2 + 4.8x_3 = 5.6 \end{cases}$$

№ 4

$$\begin{cases} 9.1x_1 + 3.6x_2 + 4.8x_3 = 9.8 \\ 2.8x_1 + 6.1x_2 + 2.8x_3 = 6.7 \\ 0.9x_1 + 1.2x_2 + 5.7x_3 = 5.8 \end{cases}$$

№ 5

$$\begin{cases} 5.3x_1 + 2.1x_2 + 2.8x_3 = 0.8 \\ 2.1x_1 + 4.7x_2 + 1.8x_3 = 5.7 \\ 2.7x_1 + 1.8x_2 + 6.1x_3 = 3.2 \end{cases}$$

№ 6

$$\begin{cases} 7.6x_1 + 2.8x_2 + 3.7x_3 = 10.1 \\ 1.8x_1 + 4.1x_2 + 1.7x_3 = 9.7 \\ 2.9x_1 + 2.1x_2 + 5.8x_3 = 7.8 \end{cases}$$

№ 7

$$\begin{cases} 3.7x_1 - 2.5x_2 + 0.7x_3 = 6.5 \\ 0.5x_1 + 3.3x_2 + 1.7x_3 = -0.24 \\ 1.6x_1 + 2.3x_2 - 7.5x_3 = 4.3 \end{cases}$$

№ 8

$$\begin{cases} 5.4x_1 - 2.3x_2 + 1.4x_3 = -3.5 \\ 1.2x_1 + 4.7x_2 - 2.3x_3 = 2.7 \\ 3.4x_1 + 2.4x_2 + 7.4x_3 = 1.9 \end{cases}$$

№ 9

$$\begin{cases} 4.6x_1 + 1.8x_2 - 1.7x_3 = 3.8 \\ 2.7x_1 - 5.6x_2 + 1.9x_3 = 0.4 \\ 1.5x_1 + 0.5x_2 + 3.3x_3 = -1.6 \end{cases}$$

№ 10

$$\begin{cases} 5.6x_1 + 2.7x_2 - 1.7x_3 = 1.9 \\ 3.4x_1 - 6.6x_2 - 0.7x_3 = -2.4 \\ 0.8x_1 + 1.3x_2 + 3.7x_3 = 1.2 \end{cases}$$

№ 11

$$\begin{cases} 2.7x_1 + 0.9x_2 - 1.5x_3 = 3.5 \\ 2.5x_1 - 4.8x_2 + 0.7x_3 = 2.6 \\ 1.1x_1 + 2.7x_2 - 5.1x_3 = -0.14 \end{cases}$$

№ 12

$$\begin{cases} 4.5x_1 - 3.5x_2 + 0.4x_3 = 2.5 \\ 3.1x_1 - 8.6x_2 - 2.3x_3 = -1.5 \\ 0.8x_1 + 2.4x_2 - 7.5x_3 = 6.4 \end{cases}$$

№ 13

$$\begin{cases} 6.8x_1 + 3.7x_2 - 1.2x_3 = 5.2 \\ 2.4x_1 + 7.3x_2 - 2.7x_3 = 3.8 \\ 2.4x_1 - 0.5x_2 + 4.5x_3 = -0.6 \end{cases}$$

№ 14

$$\begin{cases} 5.4x_1 - 1.2x_2 - 0.5x_3 = 0.52 \\ 1.4x_1 + 3.3x_2 + 0.8x_3 = -0.8 \\ 2.4x_1 - 1.1x_2 + 5.8x_3 = 1.8 \end{cases}$$

№ 15

$$\begin{cases} 7.8x_1 + 2.3x_2 + 2.8x_3 = 1.8 \\ 1.3x_1 + 4.1x_2 + 1.8x_3 = 2.3 \\ 1.5x_1 + 3.3x_2 + 5.8x_3 = 3.4 \end{cases}$$

№ 16

$$\begin{cases} 3.8x_1 + 1.1x_2 - 2.3x_3 = 4.8 \\ -2.1x_1 + 8.9x_2 - 2.8x_3 = 3.3 \\ 1.8x_1 + 1.1x_2 - 4.1x_3 = 5.8 \end{cases}$$

П 3.8 Метод Зейделя решения систем линейных алгебраических уравнений

Рассмотрим метод Зейделя [20]. Этот метод является модификацией метода простых итераций и приводит к более быстрой сходимости, т.е. для получения решения с заданной точностью требуется выполнить меньшее количество итераций, а следовательно потребуется меньше затрат машинного времени.

Имеем систему линейных алгебраических уравнений

$$AX = f\ . \tag{1}$$

Приведем систему (1) к эквивалентному виду

$$X = g + BX\ . \tag{2}$$

В методе Зейделя последовательность итерационных приближений строится по правилу

$$x_i^{(k+1)} = \sum_{j=1}^{i-1} b_{ij} x_j^{(k+1)} + \sum_{j=i}^{n} b_{ij} x_j^{(k)} + g_i, \quad i = 1,\ 2,\ ...,\ n \tag{3}$$

или в развернутом виде

$$\begin{cases} x_1^{(k+1)} = b_{11}x_1^{(k)} + b_{12}x_2^{(k)} + \ldots + b_{1n-1}x_{n-1}^{(k)} + b_{1n}x_n^{(k)} + g_1, \\ x_2^{(k+1)} = b_{21}x_1^{(k+1)} + b_{22}x_2^{(k)} + \ldots + b_{2n-1}x_{n-1}^{(k)} + b_{2n}x_n^{(k)} + g_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n^{(k+1)} = b_{n1}x_1^{(k+1)} + b_{n2}x_2^{(k+1)} + \ldots + b_{nn-1}x_{n-1}^{(k+1)} + b_{nn}x_n^{(k)} + g_n. \end{cases} \tag{4}$$

Суть метода состоит в следующем: для вычисления первой компоненты x_1^{k+1} вектора X^{k+1} необходимо знать компоненты $x_1^{(k)}$, $x_2^{(k)}$, ..., $x_n^{(k)}$ вектора X^k. При нахождении второй компоненты x_2^{k+1} вектора X^{k+1} используются только что найденное значение x_1^{k+1} и известные значения компонент $x_2^{(k)}$, ..., $x_n^{(k)}$ вектора X^k и т.д. Таким образом, при вычислении компоненты x_i^{k+1} вектора неизвестных на (*k+1*)-й итерации используются $x_1^{(k+1)}$, $x_2^{(k+1)}$, ..., $x_{i-1}^{(k+1)}$, уже вычисленные на (*k+1*)-й итерации. Значения остальных компонент $x_i^{(k)}$, $x_{i+1}^{(k)}$, ..., $x_n^{(k)}$ берутся из предыдущей итерации.

Вектор начального приближения X^0 можно выбирать произвольно. Возьмем в качестве начального приближения вектора неизвестных $X^{(0)}$ вектор правых частей, т.е. $X^0 = g$, тогда метод Зейделя можно записать следующим образом

$$X^{k+1} = PX^{k+1} + QX^k + g, \qquad (5)$$

где *P=G-D*, *Q=B-P*.

$$G = \begin{bmatrix} b_{11} & 0 & \dots & 0 \\ b_{21} & b_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{bmatrix}, \quad D = \begin{bmatrix} b_{11} & 0 & \dots & 0 \\ 0 & b_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & b_{nn} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{bmatrix}.$$

Метод Зейделя (5) можно трактовать как разновидность общего итерационного процесса:

$$X^{(k+1)} = FX^{(k)} + f, \qquad (6)$$

где $F = E - P^{-1}Q$, $f = E - P^{-1}g$.

Критерием сходимости метода Зейделя служит следующее утверждение [20]:

Для того чтобы метод Зейделя сходился при любом X^0, необходимо и достаточно, чтобы $|\lambda_i(F)| < 1$, где $\lambda_i(F)$ – все собственные значения матрицы *F*.

Применять данный критерий на практике неудобно, поэтому используют достаточные признаки сходимости:

1. Метод Зейделя сходится, если выполняется неравенство $\|B\| < 1$, где $\|B\|$ – первая или вторая норма матрицы.

2. Для сходимости метода Зейделя достаточно чтобы $\|B\|=1$, но хотя бы при одном *i* выполнялось условие $\sum_{j=1}^{n}|b_{ij}|<1$.

Условие прерывания итерационного процесса имеет вид

$$\max_{1\le i\le n}\left|X_i^{k+1}-X_i^{k}\right|<\varepsilon, \qquad (7)$$

где ε – заданная точность.

Если для одной и той же системы метод простой итерации и метод Зейделя сходятся, то последний предпочтительнее.

Области сходимости этих двух методов различны, т.е. существуют системы, для которых метод простой итерации сходится, а метод Зейделя – нет, и наоборот.

Пример 1. Методом Зейделя решить систему линейных уравнений с точностью $\varepsilon=0.01$.

$$\begin{cases}2x_1+2x_2+10x_3=14,\\ 10x_1+x_2+x_3=12,\\ 2x_1+10x_2+x_3=13.\end{cases}$$

Решение:

Приведение СЛАУ к эквивалентному виду аналогично приведению в методе простых итераций

$$\begin{cases}x_1=-0.1x_2-0.1x_3+1.2,\\ x_2=-0.2x_1-0.1x_3+1.3,\\ x_3=-0.2x_1-0.2x_2+1.4.\end{cases}$$

Заметим, что $\|B\|_1=\max\{0.2;\ 0.3;\ 0.4\}=0.4<1$, следовательно, условие сходимости метода (2) выполнено.

Зададим вектор начального приближения $x^0=g=\begin{pmatrix}1.2\\1.3\\1.4\end{pmatrix}$.

Выполним расчеты по формуле (4)

$$\begin{cases}x_1^{(k+1)}=-0.1x_2^{(k)}-0.1x_3^{(k)}+1.2,\\ x_2^{(k+1)}=-0.2x_1^{(k+1)}-0.1x_3^{(k)}+1.3 \qquad k=0,\ 1,\ ...,\\ x_3^{(k+1)}=-0.2x_1^{(k+1)}-0.2x_2^{(k+1)}+1.4.\end{cases}$$

На первой итерации имеем систему

$$\begin{cases} x_1^{(1)} = -0.1x_2^{(0)} - 0.1x_3^{(0)} + 1.2 = 0.93, \\ x_2^{(1)} = -0.2x_1^{(0)} - 0.1x_3^{(0)} + 1.3 = 0.974, \\ x_3^{(1)} = -0.2x_1^{(0)} - 0.2x_2^{(0)} + 1.4 = 1.0192. \end{cases}$$

$$X^{(1)} = \begin{bmatrix} 0.93 \\ 0.974 \\ 1.0192 \end{bmatrix}.$$

Условие окончания итерационного процесса $\max\limits_{1 \le i \le 3} \left| x_i^{\,1} - x_i^{\,0} \right| < \varepsilon$ не выполняется, значит продолжаем процесс. На второй итерации получаем

$$\begin{cases} x_1^{(2)} = -0.1x_2^{(1)} - 0.1x_3^{(1)} + 1.2 = 1.0007, \\ x_2^{(2)} = -0.2x_1^{(1)} - 0.1x_3^{(1)} + 1.3 = 0.998, \\ x_3^{(2)} = -0.2x_1^{(1)} - 0.2x_2^{(1)} + 1.4 = 1.0003. \end{cases} \quad X^{(2)} = \begin{bmatrix} 1.0007 \\ 0.998 \\ 1.0003 \end{bmatrix}.$$

Условие окончания итерационного процесса $\max\limits_{1 \le i \le 3} \left| x_i^{\,2} - x_i^{\,1} \right| < \varepsilon$ не выполняется. Продолжаем итерировать.

На третьей итерации требуемая точность достигнута

$$X = \begin{bmatrix} 1.00017 \\ 0.9999 \\ 0.9999 \end{bmatrix}.$$

Лабораторная работа № 11

Цель: изучить метод Зейделя решения систем линейных алгебраических уравнений.

Задание

1. В классе «Итерационные методы решения СЛАУ» («IterationMethods») реализуйте метод Зейделя («zeidelMethod»). Для реализации методов используйте объекты и методы матричных классов «SquareMatrix», «Vector».

2. Решите систему линейных алгебраических уравнений методом Зейделя ($\varepsilon = 0.001$) в соответствии с вариантом.

3. Решите те же задачи, используя пакет для математических вычислений.

4. Сравните результат выполнения п. 2 с решением, полученным в п. 3.

Варианты заданий

№ 1

$$\begin{cases} 2.7x_1 + 0.3x_2 + 1.3x_3 = 2.1 \\ 0.5x_1 - 3.7x_2 + 2.8x_3 = 1.7 \\ 1.1x_1 + 2.8x_2 - 5.8x_3 = 0.8 \end{cases}$$

№ 2

$$\begin{cases} 5.7x_1 + 2.8x_2 + 1.9x_3 = 0.7 \\ 0.1x_1 + 3.4x_2 + 1.8x_3 = 1.1 \\ 0.2x_1 - 1.7x_2 + 1.3x_3 = 2.8 \end{cases}$$

№ 3

$$\begin{cases} 3.1x_1 + 0.8x_2 + 1.9x_3 = 0.2 \\ 1.9x_1 + 3.1x_2 + 1.1x_3 = 2.1 \\ 0.5x_1 + 3.8x_2 + 4.8x_3 = 5.6 \end{cases}$$

№ 4

$$\begin{cases} 9.1x_1 + 3.6x_2 + 4.8x_3 = 9.8 \\ 2.8x_1 + 6.1x_2 + 2.8x_3 = 6.7 \\ 0.9x_1 + 1.2x_2 + 5.7x_3 = 5.8 \end{cases}$$

№ 5

$$\begin{cases} 5.3x_1 + 2.1x_2 + 2.8x_3 = 0.8 \\ 2.1x_1 + 4.7x_2 + 1.8x_3 = 5.7 \\ 2.7x_1 + 1.8x_2 + 6.1x_3 = 3.2 \end{cases}$$

№ 6

$$\begin{cases} 7.6x_1 + 2.8x_2 + 3.7x_3 = 10.1 \\ 1.8x_1 + 4.1x_2 + 1.7x_3 = 9.7 \\ 2.9x_1 + 2.1x_2 + 5.8x_3 = 7.8 \end{cases}$$

№ 7

$$\begin{cases} 3.7x_1 - 2.5x_2 + 0.7x_3 = 6.5 \\ 0.5x_1 + 3.3x_2 + 1.7x_3 = -0.24 \\ 1.6x_1 + 2.3x_2 - 7.5x_3 = 4.3 \end{cases}$$

№ 8

$$\begin{cases} 5.4x_1 - 2.3x_2 + 1.4x_3 = -3.5 \\ 1.2x_1 + 4.7x_2 - 2.3x_3 = 2.7 \\ 3.4x_1 + 2.4x_2 + 7.4x_3 = 1.9 \end{cases}$$

№ 9

$$\begin{cases} 4.6x_1 + 1.8x_2 - 1.7x_3 = 3.8 \\ 2.7x_1 - 5.6x_2 + 1.9x_3 = 0.4 \\ 1.5x_1 + 0.5x_2 + 3.3x_3 = -1.6 \end{cases}$$

№ 10

$$\begin{cases} 5.6x_1 + 2.7x_2 - 1.7x_3 = 1.9 \\ 3.4x_1 - 6.6x_2 - 0.7x_3 = -2.4 \\ 0.8x_1 + 1.3x_2 + 3.7x_3 = 1.2 \end{cases}$$

№ 11

$$\begin{cases} 2.7x_1 + 0.9x_2 - 1.5x_3 = 3.5 \\ 2.5x_1 - 4.8x_2 + 0.7x_3 = 2.6 \\ 1.1x_1 + 2.7x_2 - 5.1x_3 = -0.14 \end{cases}$$

№ 12

$$\begin{cases} 4.5x_1 - 3.5x_2 + 0.4x_3 = 2.5 \\ 3.1x_1 - 8.6x_2 - 2.3x_3 = -1.5 \\ 0.8x_1 + 2.4x_2 - 7.5x_3 = 6.4 \end{cases}$$

№ 13

$$\begin{cases} 6.8x_1 + 3.7x_2 - 1.2x_3 = 5.2 \\ 2.4x_1 + 7.3x_2 - 2.7x_3 = 3.8 \\ 2.4x_1 - 0.5x_2 + 4.5x_3 = -0.6 \end{cases}$$

№ 14

$$\begin{cases} 5.4x_1 - 1.2x_2 - 0.5x_3 = 0.52 \\ 1.4x_1 + 3.3x_2 + 0.8x_3 = -0.8 \\ 2.4x_1 - 1.1x_2 + 5.8x_3 = 1.8 \end{cases}$$

№ 15

$$\begin{cases} 7.8x_1 + 2.3x_2 + 2.8x_3 = 1.8 \\ 1.3x_1 + 4.1x_2 + 1.8x_3 = 2.3 \\ 1.5x_1 + 3.3x_2 + 5.8x_3 = 3.4 \end{cases}$$

№ 16

$$\begin{cases} 3.8x_1 + 1.1x_2 - 2.3x_3 = 4.8 \\ -2.1x_1 + 8.9x_2 - 2.8x_3 = 3.3 \\ 1.8x_1 + 1.1x_2 - 4.1x_3 = 5.8 \end{cases}$$

П 3.9 Итерационные методы вариационного типа решения систем линейных алгебраических уравнений

Имеем систему линейных алгебраических уравнений

$$AX = f. \tag{1}$$

Канонической формой итерационного метода решения системы (1) называется его запись в виде [19]

$$B_{k+1}\frac{X^{(k+1)} - X^{(k)}}{\tau_{k+1}} + AX^{(k)} = f, \quad k = 0,\ 1,\ 2,\ \ldots, \tag{2}$$

где $A_k = Q_k R_k -$ вещественная невырожденная матрица порядка $A_{k+1} = R_k Q_k$, задающая тот или иной итерационный метод, Q_k – итерационный параметр.

Если $B_{к+1}=E$, где E – единичная матрица, то итерационный метод (2) называется явным, в противном случае неявным.

Если $\tau_{k+1} = \tau$ и $B_{k+1} = B$ не зависят от номера итераций, то итерационный метод (2) называется стационарным, и нестационарным – в противном случае.

Рассмотрим метод скорейшего спуска.

Пусть имеем систему (1) с симметричной положительно определенной матрицей A.

Обозначим через

$$r^{\,k} = AX^{\,k} - f \tag{3}$$

невязку, которая получается при подстановке приближенного значения $X^{(k)}$, полученного на k-й итерации, в уравнение (1).

Рассмотрим явный нестационарный итерационный метод [20]

$$\frac{X^{(k+1)} - X^{(k)}}{\tau_{k+1}} + AX^{(k)} = f. \tag{4}$$

Перепишем метод (4) с учетом равенства (3), имеем

$$X^{\,k+1} = X^{\,k} - \tau_{k+1} r^{\,k}. \tag{5}$$

Выберем итерационный параметр τ_{k+1} из условия минимума $\left\| z^{\,k+1} \right\|$ при заданном векторе $z^{\,k}$, где $z^{\,k+1} = X^{\,k+1} - X^*$, X^* – точное решение. Поскольку погрешность $z^{\,k}$ удовлетворяет уравнению $z^{\,k+1} = z^{\,k} - \tau_{k+1} A z^{\,k}$, получим [19]

$$\left\| z^{k+1} \right\|^2 = \left\| z^{k} \right\|^2 - 2\tau_{k+1} \left(Az^{k}, Az^{k} \right) + \tau_{k+1} \left(A^2 z^{k}, Az^{k} \right).$$

Величина $\left\| z^{k+1} \right\|^2$ будет минимальной, если положить

$$\tau_{k+1} = \frac{\left(Az^{k}, Az^{k} \right)}{\left(A^2 z^{k}, Az^{k} \right)}.$$

Величина z^{k} неизвестна, так как неизвестно точное решение X^*. Однако надо учесть, что погрешность $z^{k} = X^{k} - X^*$ и невязка r^{k} связаны равенством $Az^{k} = r^{k}$, следовательно, вычисление τ_{k+1} можно проводить по формуле

$$\tau_{k+1} = \frac{\left(r^{k}, r^{k} \right)}{\left(Ar^{k}, r^{k} \right)}. \qquad (6)$$

Таким образом, в методе скорейшего спуска переход от *k*-ой итерации к *(k+1)*-ой осуществляется следующим образом: по найденному значению X^{k} вычисляется вектор невязки (3) и по формуле (6) находится параметр τ_{k+1}, затем по формуле (5) вычисляется вектор приближенного решения X^{k+1}. Вектор начального приближения X^{0} выбирается произвольно.

Для погрешности метода скорейшего спуска справедлива оценка [19].

$$\left\| X^{(n)} - X^* \right\| \le \rho_0^n \left\| X^{(0)} - X^* \right\| \quad n = 0, 1, \ldots,$$

где $\rho_0 = \dfrac{1-\xi}{1+\xi}, \xi = \dfrac{\lambda_{\min}(A)}{\lambda_{\max}(A)}$.

Пример 1. Методом скорейшего спуска решить систему линейных алгебраических уравнений $AX = f$, где

$$A = \begin{pmatrix} 2{,}5 & -0{,}9 & 0{,}2 \\ -0{,}9 & 3{,}8 & -0{,}1 \\ 0{,}2 & -0{,}1 & 0{,}9 \end{pmatrix}, \quad f = \begin{pmatrix} -0{,}7 \\ 2{,}5 \\ 0{,}1 \end{pmatrix}.$$

Решение:

Выберем начальное приближение $X^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

Рассчитаем вектор невязки по формуле $r^{(0)} = AX^{(0)} - f$.

Получим $r^{(0)} = \begin{pmatrix} 0,7 \\ -2,5 \\ -0,1 \end{pmatrix}$.

Вычислим $\tau_1 = \dfrac{r^0, r^0}{Ar^0, r^0}$, $\tau_1 = 0,2406.$

Приближение X^1 вычислим по формуле: $X^{(1)} = X^{(0)} - \tau_1 r^{(0)}$, имеем $X^{(1)} = \begin{pmatrix} -0,168 \\ 0,601 \\ 0,024 \end{pmatrix}$.

Вычислим вектор невязки по формуле $r^{(1)} = AX^{(1)} - f$, получим $r^{(1)} = \begin{pmatrix} -0,258 \\ -0,065 \\ -0,172 \end{pmatrix}$.

Продолжаем итерации. Имеем $\tau_2 = \dfrac{r^1, r^1}{Ar^1, r^1}$, $\tau_2 = 0,5168.$

Приближение X^2 находим по формуле : $X^{(2)} = X^{(1)} - \tau_2 r^{(1)}$, получим $X^{(2)} = \begin{pmatrix} -0,035 \\ 0,635 \\ 0,113 \end{pmatrix}$.

Найдем вектор невязки по формуле $r^{(2)} = AX^{(2)} - f$, имеем $r^{(2)} = \begin{pmatrix} 0,063 \\ -0,066 \\ -0,069 \end{pmatrix}$.

Вычислим $\tau_3 = \dfrac{r^2, r^2}{Ar^2, r^2}$, $\tau_3 = 0,3678.$

Приближение X^3 вычислим по формуле: $X^{(3)} = X^{(2)} - \tau_3 r^{(2)}$, получим $X^{(3)} = \begin{pmatrix} -0,058 \\ 0,659 \\ 0,138 \end{pmatrix}$.

Данный итерационный процесс можно продолжать до получения решения с требуемой точностью.

Пример 2. В рамках метода скорейшего спуска реализовать нахождение скалярного произведения векторов.

Данный метод описан в классе «Vector». Реализация на языке программирования C++ может иметь следующий вид:

```
double Vector :: mul(Vector r) {
      double result = 0;
      for (int i=0; i< this.getColCount ; i++)
            {
       result += this.elements[i] * r.elements[i];
                }
        return result;
       }
```

Лабораторная работа № 12

Цель: изучить метод скорейшего спуска решения систем линейных алгебраических уравнений.

Задание

1. В классе «Итерационные методы решения СЛАУ» («IterationMethods») реализуйте метод скорейшего спуска («methodSkorSpusk»). Для реализации метода используйте объекты класса «SquareMatrix» и «Vector». Для перемножения матрицы и вектора, для нахождения скалярного произведения векторов используйте методы, реализованные в данных классах. Доступ к элементам осуществляйте с помощью методов getElement и setElement.

2. Решите систему линейных алгебраических уравнений методом скорейшего спуска ($\varepsilon = 0.001$) в соответствии с вариантом.

3. Решите эту же задачу, используя пакет для математических вычислений.

4. Сравните результат выполнения п. 2 с решением, полученным в п. 3.

Варианты заданий

№ 1

$$\begin{cases} 2.7x_1 + 0.3x_2 + 1.3x_3 = 2.1 \\ 0.5x_1 - 3.7x_2 + 2.8x_3 = 1.7 \\ 1.1x_1 + 2.8x_2 - 5.8x_3 = 0.8 \end{cases}$$

№ 2

$$\begin{cases} 5.7x_1 + 2.8x_2 + 1.9x_3 = 0.7 \\ 0.1x_1 + 3.4x_2 + 1.8x_3 = 1.1 \\ 0.2x_1 - 1.7x_2 + 1.3x_3 = 2.8 \end{cases}$$

№ 3

$$\begin{cases} 3.1x_1 + 0.8x_2 + 1.9x_3 = 0.2 \\ 1.9x_1 + 3.1x_2 + 1.1x_3 = 2.1 \\ 0.5x_1 + 3.8x_2 + 4.8x_3 = 5.6 \end{cases}$$

№ 4

$$\begin{cases} 9.1x_1 + 3.6x_2 + 4.8x_3 = 9.8 \\ 2.8x_1 + 6.1x_2 + 2.8x_3 = 6.7 \\ 0.9x_1 + 1.2x_2 + 5.7x_3 = 5.8 \end{cases}$$

№ 5

$$\begin{cases} 5.3x_1 + 2.1x_2 + 2.8x_3 = 0.8 \\ 2.1x_1 + 4.7x_2 + 1.8x_3 = 5.7 \\ 2.7x_1 + 1.8x_2 + 6.1x_3 = 3.2 \end{cases}$$

№ 6

$$\begin{cases} 7.6x_1 + 2.8x_2 + 3.7x_3 = 10.1 \\ 1.8x_1 + 4.1x_2 + 1.7x_3 = 9.7 \\ 2.9x_1 + 2.1x_2 + 5.8x_3 = 7.8 \end{cases}$$

№ 7

$$\begin{cases} 3.7x_1 - 2.5x_2 + 0.7x_3 = 6.5 \\ 0.5x_1 + 3.3x_2 + 1.7x_3 = -0.24 \\ 1.6x_1 + 2.3x_2 - 7.5x_3 = 4.3 \end{cases}$$

№ 8

$$\begin{cases} 5.4x_1 - 2.3x_2 + 1.4x_3 = -3.5 \\ 1.2x_1 + 4.7x_2 - 2.3x_3 = 2.7 \\ 3.4x_1 + 2.4x_2 + 7.4x_3 = 1.9 \end{cases}$$

№ 9

$$\begin{cases} 4.6x_1 + 1.8x_2 - 1.7x_3 = 3.8 \\ 2.7x_1 - 5.6x_2 + 1.9x_3 = 0.4 \\ 1.5x_1 + 0.5x_2 + 3.3x_3 = -1.6 \end{cases}$$

№ 10

$$\begin{cases} 5.6x_1 + 2.7x_2 - 1.7x_3 = 1.9 \\ 3.4x_1 - 6.6x_2 - 0.7x_3 = -2.4 \\ 0.8x_1 + 1.3x_2 + 3.7x_3 = 1.2 \end{cases}$$

№ 11

$$\begin{cases} 2.7x_1 + 0.9x_2 - 1.5x_3 = 3.5 \\ 2.5x_1 - 4.8x_2 + 0.7x_3 = 2.6 \\ 1.1x_1 + 2.7x_2 - 5.1x_3 = -0.14 \end{cases}$$

№ 12

$$\begin{cases} 4.5x_1 - 3.5x_2 + 0.4x_3 = 2.5 \\ 3.1x_1 - 8.6x_2 - 2.3x_3 = -1.5 \\ 0.8x_1 + 2.4x_2 - 7.5x_3 = 6.4 \end{cases}$$

№ 13

$$\begin{cases} 6.8x_1 + 3.7x_2 - 1.2x_3 = 5.2 \\ 2.4x_1 + 7.3x_2 - 2.7x_3 = 3.8 \\ 2.4x_1 - 0.5x_2 + 4.5x_3 = -0.6 \end{cases}$$

№ 14

$$\begin{cases} 5.4x_1 - 1.2x_2 - 0.5x_3 = 0.52 \\ 1.4x_1 + 3.3x_2 + 0.8x_3 = -0.8 \\ 2.4x_1 - 1.1x_2 + 5.8x_3 = 1.8 \end{cases}$$

№ 15

$$\begin{cases} 7.8x_1 + 2.3x_2 + 2.8x_3 = 1.8 \\ 1.3x_1 + 4.1x_2 + 1.8x_3 = 2.3 \\ 1.5x_1 + 3.3x_2 + 5.8x_3 = 3.4 \end{cases}$$

№ 16

$$\begin{cases} 3.8x_1 + 1.1x_2 - 2.3x_3 = 4.8 \\ -2.1x_1 + 8.9x_2 - 2.8x_3 = 3.3 \\ 1.8x_1 + 1.1x_2 - 4.1x_3 = 5.8 \end{cases}$$

Глава 4
ВЫЧИСЛЕНИЕ СОБСТВЕННЫХ ЗНАЧЕНИЙ И СОБСТВЕННЫХ ВЕКТОРОВ МАТРИЦ

Пусть A – действительная числовая квадратная матрица размера (n*n).

Ненулевой вектор $X = (x_1, \ldots, x_n)^T$, удовлетворяющий условию

$$AX = \lambda X , \qquad (1)$$

называется собственным вектором матрицы A.

Число λ в равенстве (1) называется собственным значением. Говорят, что собственный вектор X соответствует (принадлежит) собственному значению λ.

Равенство (1) равносильно однородной относительно X системе:

$$A - \lambda E \; X = 0, \qquad X \neq 0 \; . \qquad (2)$$

Система (2) имеет ненулевое решение для вектора X (при известном λ) при условии $|A - \lambda E| = 0$. Это равенство есть характеристическое уравнение:

$$|A - \lambda E| = P_n \; \lambda \; = 0 , \qquad (3)$$

где $P_n \; \lambda$ – характеристический многочлен n-й степени.

Корни $\lambda_1, \; \lambda_2, \ldots, \lambda_n$ характеристического уравнения (3) являются собственными (характеристическими) значениями матрицы A, а соответствующие каждому собственному значению λ_i, $i = 1, \ldots, n$, ненулевые векторы X^i, удовлетворяющие системе

$$AX^i = \lambda_i X^i \;\; \text{или} \;\; (A - \lambda_i E) X^i = 0, \quad i = 1, 2, \ldots, n, \qquad (4)$$

являются собственными векторами.

Требуется найти собственные значения и собственные векторы заданной матрицы. Поставленная задача часто именуется второй задачей линейной алгебры [7].

Проблема собственных значений (частот) возникает при анализе поведения мостов, зданий, летательных аппаратов и других конструкций, характеризующихся малыми смещениями от положения равновесия, а также при анализе устойчивости численных схем. Характеристическое уравнение вместе с его собственными значениями и собственными векторами является основным в теории механических или

электрических колебаний на макроскопическом или микроскопическом уровнях.

Множество всех собственных значений матрицы A называется спектром матрицы A. Различают полную и частичную проблему собственных значений, когда необходимо найти весь спектр и собственные векторы либо часть спектра, например: $\rho\ A = \max_i \left|\lambda_i\ A\right|$ и $\min_i \left|\lambda_i\ A\right|$. Величина $\rho\ A$ называется спектральным радиусом.

Если для собственного значения λ_i, $i = 1, \ldots, n$ найден собственный вектор X^i, то вектор μX^i, где μ – произвольное число, также является собственным вектором, соответствующим этому же собственному значению λ_i, т.е. все собственные векторы матрицы определяются с точностью до числового множителя.

Попарно различным собственным значениям соответствуют линейно независимые собственные векторы; k-кратному корню характеристического уравнения соответствует не более k линейно независимых собственных векторов.

Симметрическая матрица имеет полный спектр действительных собственных значений λ_i, $i = 1, \ldots, n$; k-кратному корню характеристического уравнения симметрической матрицы соответствует ровно k линейно независимых собственных векторов.

К настоящему времени создано немало специальных вычислительных приемов, упрощающих численное нахождение собственных значений и собственных векторов матрицы. Все эти методы, как и в случае проблемы численного решения системы линейных алгебраических уравнений, можно разделить на точные и итерационные методы [8].

К первой группе относятся методы, по которым сначала строят собственный многочлен матрицы, затем, находя его корни, получают собственные значения матрицы и уже по ним находят соответствующие собственные векторы. Методы этой группы получили название точных методов. Точные методы позволяют решать полную проблему собственных значений, т.е. дают возможность находить все собственные значения матрицы и все принадлежащие им собственные векторы. Полная проблема собственных значений в некоторых случаях может быть решена также и специальными итерационными методами. Эти методы, конечно, более трудоемки, чем точные методы.

В методах второй группы собственные значения матрицы определяются непосредственно, без обращения к собственному многочлену, при этом одновременно вычисляются и соответствующие собственные векторы. Вычислительные схемы таких методов носят итерационный характер. В них используется многократное умножение матрицы на вектор. Схемы этого типа обычно приводят к последователь-

ности векторов, имеющей своим пределом собственный вектор, и к числовой последовательности, предел которой является соответствующим собственным значением. Как правило, итерационные методы позволяют с достаточной точностью определить лишь первые (наибольшие по модулю) собственные значения и соответствующие им собственные векторы. Поэтому методы этой группы чаще всего применяются к решению частичной проблемы собственных значений, т.е. их чаще используют лишь для отыскания одного или нескольких собственных значений матрицы и соответствующих собственных векторов. Большим достоинством итерационных методов перед точными является простота и единообразие производимых действий, что особенно ценно при использовании быстродействующих вычислительных машин.

Полная и частичная проблемы собственных значений сильно различаются как по методам их решения, так и по области приложений. Для решения полной проблемы собственных значений используют метод Данилевского, QR-алгоритм, итерационный метод вращения (метод Якоби). Степенной метод нахождения наибольшего по модулю собственного значения и соответствующего собственного вектора – метод решения частичной проблемы собственных значений.

П 4.1 Метод Данилевского для нахождения собственных значений и собственных векторов

Рассмотрим метод Данилевского.

Этот метод основан на том, что преобразование подобия $S^{-1}AS$ не изменяет характеристического многочлена матрицы A [10]. Матрицы, связанные преобразованием подобия, имеют одинаковые спектры. Поэтому, удачно подобрав преобразование подобия, можно получить матрицу, собственный многочлен, который записывается непосредственно по ее виду. М.А. Данилевский предложил исходную матрицу A приводить преобразованием подобия $S^{-1}AS$ к так называемой канонической форме Фробениуса.

$$\hat{O} = \begin{bmatrix} p_1 & p_2 & p_3 & \cdots & p_{n-1} & p_n \\ 1 & 0 & 0 & \ldots & 0 & 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots & & & & & \\ 0 & 0 & 0 & \ldots & 1 & 0 \end{bmatrix}. \qquad (1)$$

Характеристический полином матрицы Φ можно записать в виде

$$\left|\Phi\text{-}\lambda\mathrm{E}\right| = \begin{vmatrix} p_1-\lambda & p_2 & \ldots & p_n \\ 1 & -\lambda & \ldots & 0 \\ 0 & 1 & -\lambda & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & -\lambda \end{vmatrix} = -1^{\,n}\ \lambda^n - p_1\lambda^{n-1} - \ldots p_n\ = -1^{\,n} P\ \lambda\ . \quad (2)$$

Таким образом, элементы $p_1, p_2, \ldots, p_n$ первой строки матрицы Фробениуса являются соответствующими коэффициентами ее собственного многочлена, а значит, и собственного многочлена исходной матрицы *A*, связанной с матрицей *Ф* преобразованием подобия:

$$\hat{O} = S^{-1}AS\,. \quad (3)$$

Решая уравнение

$$P\ \lambda\ = 0\,, \quad (4)$$

найдем собственные значения матриц Фробениуса и *A*.

Следовательно, основная задача сводится к отысканию матрицы *S*, которая обеспечивает преобразование подобия от матрицы *A* к матрице *Ф*.

Рассмотрим эти преобразования и построим матрицу Фробениуса.

Возьмем единичную матрицу, размерность которой соответствует размерности матрицы *A*.

В единичной матрице *n–1* строка заменяется строкой, сформированной из элементов *n*-ой строки матрицы *A*, взятых с противоположным знаком, после их деления на элемент $a_{n,n-1} \neq 0$. Только элемент *n–1* столбца формируется по-иному. В эту позицию выставляется элемент, обратный элементу $a_{n,n-1}$.

Получим матрицу

$$M_{n-1} = \begin{bmatrix} 1 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 & 0 \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ \dfrac{-a_{n,1}}{a_{n,n-1}} & \dfrac{-a_{n,2}}{a_{n,n-1}} & \ldots & \dfrac{-a_{n,n-2}}{a_{n,n-1}} & \dfrac{1}{a_{n,n-1}} & \dfrac{-a_{n,n}}{a_{n,n-1}} \\ 0 & 0 & \ldots & 0 & 0 & 1 \end{bmatrix}.$$

Умножение матрицы *A* справа на матрицу M_{n-1} дает матрицу AM_{n-1}, в которой последняя строка принимает нужный вид, т.е. совпадает с последней строкой матрицы Фробениуса. Затем полученную

матрицу умножим слева на матрицу M_{n-1}^{-1} (обратную), которая существует, так как $|M_{n-1}| = \frac{1}{a_{n,n-1}} \neq 0$, т.е в единичной матрице *(n–1)*-строка заменяется *n*-строкой исходной матрицы *A*.

$$M_{n-1}^{-1} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n,1} & a_{n2} & \dots & a_{n,n-1} & a_{n,n} \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}.$$

Очевидно, что преобразование $M_{n-1}^{-1} A M_{n-1}$ не изменяет последнюю строку матрицы AM_{n-1}. Таким образом, после выполнения первого шага метода Данилевского, получаем матрицу $A^1 = M_{n-1}^{-1} A M_{n-1}$:

$$A^1 = \begin{bmatrix} a_{11}^1 & a_{12}^1 & \dots & a_{1,n-1}^1 & a_{1,n}^1 \\ a_{21}^1 & a_{22}^1 & \dots & a_{2,n-1}^1 & a_{2,n}^1 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n-1,1}^1 & a_{n-1,2}^1 & \dots & a_{n-1,n-1}^1 & a_{n-1,n}^1 \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}.$$

При этом матрицы M_{n-1}^{-1} и M_{n-1} записываются непосредственно по виду матрицы *A*.

Второй шаг метода Данилевского аналогичен первому. Приведем вторую снизу строку матрицы A^1 к форме Фробениуса при сохранении неизменной первой снизу строки. Для этого проводим преобразование: $A^2 = M_{n-2}^{-1} A^1 M_{n-2} \Leftrightarrow M_{n-2}^{-1} M_{n-1}^{-1} A M_{n-1} M_{n-2}$, где

$$M_{n-2} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{-a_{n-1,1}^1}{a_{n-1,n-2}^1} & \frac{-a_{n-1,2}^1}{a_{n-1,n-2}^1} & \dots & \frac{1}{a_{n-1,n-2}^1} & \frac{-a_{n-1,n-1}^1}{a_{n-1,n-2}^1} & \frac{-a_{n-1,n}^1}{a_{n-1,n-2}^1} \\ 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix}.$$

$$M_{n-2}^{-1}=\begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n-1,1}^{1} & a_{n-1,2}^{1} & \dots & a_{n-1,n-1}^{1} & a_{n-1,n}^{1} \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}.$$

Итак, если $a_{n,n-1}\neq 0,\quad a_{n-1,n-2}^{1}\neq 0,\dots,a_{21}^{n-2}\neq 0$, то после *n–1* шагов будем иметь:

$$M_1^{-1}M_2^{-1}\dots M_{n-1}^{-1}AM_{n-1}M_{n-2}\dots M_1=A^{n-1}, \qquad (5)$$

где матрицы M_{n-i} формируются из единичной матрицы заменой элементов *n–i* строки на элементы, полученные делением элементов *n–i+1* строки матрицы *A*, взятых с противоположным знаком, на *n–i* элемент столбца из этой строки.

Преобразование (5) можно записать следующим образом:

$$A^{(n-1)}=\begin{bmatrix} a_{11}^{(n-1)} & a_{12}^{(n-1)} & \dots & a_{1,n-1}^{(n-1)} & a_{1n}^{(n-1)} \\ 1 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}=\begin{bmatrix} p_1 & p_2 & \dots & p_{n-1} & p_n \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}=$$

$$=\hat{O}=S^{-1}AS.$$

Тем самым исходная матрица *A* посредством преобразования подобия с матрицей $S=M_{n-1}M_{n-2}...M_1$ будет приведена к канонической форме Фробениуса, по виду первой строки которой записывается собственный многочлен: $\mathrm{P}\ \lambda=\lambda^{\mathrm{n}}-\mathrm{p}_1\lambda^{\mathrm{n-1}}-\dots-\mathrm{p}_{\mathrm{n}}$.

Решая уравнения (4), найдем собственные значения матрицы *Ф*, а следовательно и матрицы *A*.

Пример 1. С помощью преобразования подобия привести матрицу *A* к канонической форме Фробениуса. Найти собственные значения матрицы *A*.

$$A=\begin{pmatrix} 2.2 & 1 & 0.5 & 2 \\ 1 & 1.3 & 2 & 1 \\ 0.5 & 2 & 0.5 & 1.6 \\ 2 & 1 & 1.6 & 2 \end{pmatrix}.$$

Решение:

Первый этап

$$M_{n-1}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{-2}{1.6} & \frac{-1}{1.6} & \frac{1}{1.6} & \frac{-2}{1.6} \\ 0 & 0 & 0 & 1 \end{pmatrix}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1.25 & -0.625 & 0.625 & -1.25 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$M_{n-1}^{-1}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 1 & 1.6 & 2 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$A^{1}=M_{n-1}^{-1}AM_{n-1}=\begin{pmatrix} 1.575 & 0.6875 & 0.3125 & 1.375 \\ -1.5 & 0.05 & 1.25 & -1.5 \\ 1.45 & 4.125 & 4.375 & 2.81 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Второй этап

$$M_{n-2}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{-1.45}{4.125} & \frac{1}{4.125} & \frac{-4.375}{4.125} & \frac{-2.81}{4.125} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}=$$

$$=\begin{pmatrix} 1 & 0 & 0 & 0 \\ -0.3515 & 0.2424 & -1.0606 & -0.6812 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$M_{n-2}^{-1}=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1.45 & 4.125 & 4.375 & 2.81 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$A^{2}=M_{n-2}^{-1}A^{1}M_{n-2}=\begin{pmatrix} 1.3333 & 0.1667 & -0.4167 & 0.9067 \\ -4.3267 & 4.6667 & 7.1433 & -5.0133 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Третий этап

$$M_{n-3}=\begin{pmatrix} \frac{1}{-4.3267} & \frac{-4.6667}{-4.3267} & \frac{-7.1433}{-4.3267} & -\frac{-5.0133}{-4.3267} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}=$$

$$=\begin{pmatrix} -0.2311 & 1.0786 & 1.651 & -1.1587 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$M_{n-3}^{-1}=\begin{pmatrix} -4.3267 & 4.6667 & 7.1433 & -5.0133 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$A^3=M_{n-3}^{-1}A^2M_{n-3}=\begin{pmatrix} 6 & 0.2 & -12.735 & 2.7616 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},$$

$$\Phi=\begin{pmatrix} 6 & 0.2 & -12.735 & 2.7616 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Первая строка матрицы Φ определяет коэффициенты характеристического уравнения матрицы A, которое имеет вид $\lambda^4-6\lambda^3-0.2\lambda^2+12.735\lambda-2.7616=0$.

Корни этого уравнения являются собственными значениями матрицы Φ, а следовательно матрицы A. Решаем уравнение, получаем

$$\lambda_1=-1.4201,\quad \lambda_2=0.2226,\quad \lambda_3=1.5454,\quad \lambda_4=5.652.$$

Найдем собственные векторы матрицы A.

Собственные векторы матриц Φ и A, принадлежащие одним и тем же собственным значениям, будут различны, но между ними существует связь [10].

Если X – собственный вектор матрицы A, принадлежащий собственному значению λ, а вектор Y – собственный вектор матрицы

Фробениуса $\Phi = \mathrm{S}^{-1} AS$, принадлежащий тому же собственному значению λ, то вектор SY также будет собственным вектором матрицы A, соответствующим собственному значению λ, т.е. $X = SY$.

Действительно, так как $\Phi Y = \lambda Y$ и $\hat{O} = S^{-1} AS$, то следовательно $S^{-1} ASY = \lambda Y$.

Умножая это равенство слева на матрицу S, получаем $ASY = \lambda SY$, учитывая, что $AX = \lambda X$, имеем

$$X = SY. \tag{6}$$

Итак, собственные векторы матрицы A легко определить по соответствующим собственным векторам матрицы Φ.

Найдем собственные векторы матрицы Φ. Имеем $\Phi Y = \lambda Y \Leftrightarrow$

$$\begin{bmatrix} p_1 & p_2 & \cdots & p_{n-1} & p_n \\ 1 & 0 & \ldots & 0 & 0 \\ \ldots\ldots\ldots\ldots\ldots & & & & \\ 0 & 0 & \ldots & 1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_n \end{bmatrix} = \lambda \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_n \end{bmatrix}.$$

Отсюда получаем систему

$$\left.\begin{aligned} p_1 y_1 + p_2 y_2 + \ldots + p_n y_n &= \lambda y_1 \\ y_1 &= \lambda y_2 \\ \ldots\ldots\ldots & \\ y_{n-1} &= \lambda y_n \end{aligned}\right\}. \tag{7}$$

Так как собственный вектор матрицы определен с точностью до постоянного множителя, то полагаем $y_n = 1$.

Тогда из предыдущих равенств системы можно последовательно найти остальные координаты вектора Y

$$Y = \begin{bmatrix} \lambda^{n-1} \\ \lambda^{n-2} \\ \vdots \\ \lambda \\ 1 \end{bmatrix}. \tag{8}$$

Равенство $p_1y_1 + ... + p_ny_n = \lambda y_1$ можно использовать для контроля вычислений $X = SY = M_{n-1}M_{n-2}...M_1Y$.

Собственный вектор $X^{(i)}$, соответствующий числу λ_i, определяется равенством:

$$X^{(i)} = M_{n-1}M_{n-2}...M_1Y^{(i)}, \qquad (9)$$

где $Y^{(i)} = (\lambda_i^{n-1}, \lambda_i^{n-2}, \ldots, \lambda_i, 1)^T$ – собственный вектор матрицы Фробениуса, соответствующий собственному значению λ_i.

Отдельные координаты вектора $X^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)})^T$ находим из равенства (9), в предположении, что $x_n^{(i)} = 1$.

Пример 2. Найти собственные векторы матрицы из предыдущего примера.

Решение:

Матрица Фробениуса построена

$$\Phi = \begin{pmatrix} 6 & 0.2 & -12.735 & 2.7616 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Собственные значения найдены

$$\lambda_1 = -1.4201, \quad \lambda_2 = 0.2226, \quad \lambda_3 = 1.5454, \quad \lambda_4 = 5.652.$$

Найдем собственные вектора матрицы A.

Матрица подобия имеет вид

$$S = M_{n-1}M_{n-2}M_{n-3} = \begin{pmatrix} -0.2311 & 1.0786 & 1.651 & -1.1587 \\ 0.0812 & -0.1367 & -1.641 & -0.2739 \\ 0.2381 & -1.2628 & -0.4132 & 0.3696 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Собственные векторы матрицы Фробениуса:

$$\vec{y}_1 = \begin{pmatrix} -1.4201^{\,3} \\ -1.4201^{\,2} \\ -1.4201 \\ 1 \end{pmatrix} = \begin{pmatrix} -2.8638 \\ 2.0166 \\ -1.4201 \\ 1 \end{pmatrix}, \qquad \vec{y}_2 = \begin{pmatrix} 0.2226^{\,3} \\ 0.2226^{\,2} \\ 0.2226 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.011 \\ 0.0496 \\ 0.2226 \\ 1 \end{pmatrix},$$

$$\vec{y}_3 = \begin{pmatrix} 1.5454^{\,3} \\ 1.5454^{\,2} \\ 1.5454 \\ 1 \end{pmatrix} = \begin{pmatrix} 3.691 \\ 2.3883 \\ 1.5454 \\ 1 \end{pmatrix}, \qquad \vec{y}_4 = \begin{pmatrix} 5.652^{\,3} \\ 5.652^{\,2} \\ 5.652 \\ 1 \end{pmatrix} = \begin{pmatrix} 180.5568 \\ 31.9455 \\ 5.652 \\ 1 \end{pmatrix}.$$

Собственные векторы матрицы *A*

$$\vec{x}_1 = S \cdot \vec{y}_1 = \begin{pmatrix} -0.6663 \\ 1.548 \\ -2.2723 \\ 1 \end{pmatrix}, \qquad \vec{x}_2 = S \cdot \vec{y}_2 = \begin{pmatrix} -0.7402 \\ -0.6451 \\ 0.2176 \\ 1 \end{pmatrix},$$

$$\vec{x}_3 = S \cdot \vec{y}_3 = \begin{pmatrix} 3.1157 \\ -2.8365 \\ -2.4059 \\ 1 \end{pmatrix}, \qquad \vec{x}_4 = S \cdot \vec{y}_4 = \begin{pmatrix} 0.8975 \\ 0.7531 \\ 0.69 \\ 1 \end{pmatrix}.$$

Пример 3. В рамках метода Данилевского реализовать операцию перемножения матриц.

Данный метод описан в матричном классе «SquareMatrix».

Реализация на языке программирования C++ может иметь следующий вид:

```
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);
            };
```

Лабораторная работа № 13

Цель: изучить метод Данилевского для нахождения собственных значений и собственных векторов.

Задание

1. В классе «FrobeniusMatrix» («Матрица Фробениуса»), который наследуется от класса «SquareMatrix» («Квадратная матрица»), сформируйте матрицу Фробениуса.

2. В классе «DirectMethodsE» («Прямые методы нахождения собственных значений») реализуйте метод Данилевского («danilevskyMethod») для нахождения собственных значений и собственных векторов матрицы.

Для реализации метода используйте объекты матричных классов «FrobeniusMatrix», «SquareMatrix» и «Vector». Для выполнения основных матричных операций (перемножение матриц, умножение матрицы на вектор) используйте методы, реализованные в матричном классе «SquareMatrix».

3. Методом Данилевского найдите собственные значения и собственные векторы матрицы в соответствии с вариантом.

4. Решите ту же задачу, используя пакет для математических вычислений.

5. Сравните результат выполнения п. 3 с решением, полученным в п. 4.

Варианты заданий

№ 1

$$A=\begin{pmatrix} 1 & 1.5 & 2.5 & 3.5 \\ 1.5 & 1 & 2 & 1.6 \\ 2.5 & 2 & 1 & 1.7 \\ 3.5 & 1.6 & 1.7 & 1 \end{pmatrix}$$

№ 2

$$A=\begin{pmatrix} 1 & 1.2 & 2 & 0.5 \\ 1.2 & 1 & 0.4 & 1.2 \\ 2 & 0.4 & 2 & 1.5 \\ 0.5 & 1.2 & 1.5 & 1 \end{pmatrix}$$

№ 3

$$A=\begin{pmatrix} 1 & 1.2 & 2 & 0.5 \\ 1.2 & 1 & 0.5 & 1 \\ 2 & 0.5 & 2 & 1.5 \\ 0.5 & 1 & 1.5 & 0.5 \end{pmatrix}$$

№ 4

$$A=\begin{pmatrix} 2.5 & 1 & -0.5 & 2 \\ 1 & 2 & 1.2 & 0.4 \\ -0.5 & 1.2 & -1 & 1.5 \\ 2 & 0.4 & 1.5 & 1 \end{pmatrix}$$

№ 5

$$A=\begin{pmatrix} 2 & 1 & 1.4 & 0.5 \\ 1 & 1 & 0.5 & 1 \\ 1.4 & 0.5 & 2 & 1.2 \\ 0.5 & 1 & 1.2 & 0.5 \end{pmatrix}$$

№ 6

$$A=\begin{pmatrix} 2 & 1.2 & -1 & 1 \\ 1.2 & 0.5 & 2 & -1 \\ -1 & 2 & -1.5 & 0.2 \\ 1 & -1 & 0.2 & 1.5 \end{pmatrix}$$

№ 7

$$A=\begin{pmatrix} 2 & 1.5 & 3.5 & 4.5 \\ 1.5 & 2 & 2 & 1.6 \\ 3.5 & 2 & 2 & 1.7 \\ 4.5 & 1.6 & 1.7 & 2 \end{pmatrix}$$

№ 8

$$A=\begin{pmatrix} 1 & 0.5 & 1.2 & -1 \\ 0.5 & 2 & -0.5 & 0 \\ 1.2 & -0.5 & -1 & 1.4 \\ -1 & 0 & 1.4 & 1 \end{pmatrix}$$

№ 9

$$A=\begin{pmatrix} 1.2 & 0.5 & 2 & 1 \\ 0.5 & 1 & 0.8 & 2 \\ 2 & 0.8 & 1 & 1 \\ 1 & 2 & 1 & 2 \end{pmatrix}$$

№ 10

$$A=\begin{pmatrix} 0.5 & 1.2 & 1 & 0.9 \\ 1.2 & 2 & 0.5 & 1.2 \\ 1 & 0.5 & 1 & 1 \\ 0.9 & 1.2 & 1 & 2.2 \end{pmatrix}$$

№ 11

$$A=\begin{pmatrix} 1.6 & 0.4 & 1 & 2 \\ 0.4 & 1 & 0.5 & 1 \\ 1 & 0.5 & 0 & 0.2 \\ 2 & 1 & 0.2 & 0.5 \end{pmatrix}$$

№ 12

$$A=\begin{pmatrix} 2 & 1.5 & 4.5 & 5.5 \\ 1.5 & 3 & 2 & 1.6 \\ 4.5 & 2 & 3 & 1.7 \\ 5.5 & 1.6 & 1.7 & 3 \end{pmatrix}$$

№ 13

$$A=\begin{pmatrix} 1.6 & 1 & 1.4 & 1 \\ 1 & 1 & 0.5 & 2 \\ 1.4 & 0.5 & 2 & 1.2 \\ 1 & 2 & 1.2 & 0.5 \end{pmatrix}$$

№ 14

$$A=\begin{pmatrix} 2.4 & 0.5 & 2 & 1 \\ 0.5 & 1 & 0.8 & 2 \\ 2 & 0.8 & 1 & 0.5 \\ 1 & 2 & 0.5 & 1.2 \end{pmatrix}$$

№ 15

$$A=\begin{pmatrix} 0.5 & 1.2 & 2 & 1 \\ 1.2 & 2 & 0.5 & 1.2 \\ 2 & 0.5 & 1 & 0.5 \\ 1 & 1.2 & 0.5 & 1.6 \end{pmatrix}$$

№ 16

$$A=\begin{pmatrix} 1.8 & 1.6 & 1.7 & 1.8 \\ 1.6 & 2.8 & 1.5 & 1.3 \\ 1.7 & 1.5 & 3.8 & 1.4 \\ 1.8 & 1.3 & 1.4 & 4.8 \end{pmatrix}$$

П 4.2 Итерационный степенной метод нахождения наибольшего по модулю собственного значения и соответствующего собственного вектора

Рассмотрим степенной метод нахождения наибольшего по модулю собственного значения и соответствующего собственного вектора [10].

Рассмотрим вещественную квадратную матрицу

$$A=\begin{bmatrix} a_{11} & a_{12} & ... & a_{1,n-1} & a_{1n} \\ a_{21} & a_{22} & ... & a_{2,n-1} & a_{2n} \\ ... & ... & .... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{n,n-1} & a_{nn} \end{bmatrix}.$$

Пусть $\lambda_1,\ \lambda_2,\ ...,\ \lambda_n$ – ее собственные значения, а $X^{(1)},\ X^{(2)},\ ...,\ X^{(n)}$ – собственные вектора, соответствующие этим собственным значениям, т.е.

$$AX^{(i)}=\lambda_i X^{(i)},\quad i=1,\ 2,\ ...,\ n.$$

Будем считать, что матрица A обладает полной системой линейно-независимых собственных векторов, т.е. все $X^{(1)},\ X^{(2)},\ ...,\ X^{(n)}$ предполагаем линейно-независимыми.

Это будет иметь место, например, если матрица A симметрична ($A=A^T$) или если все ее собственные значения $\lambda_1,\ \lambda_2,\ ...,\ \lambda_n$ различны.

Тогда систему собственных векторов $X^{(1)},\ X^{(2)},\ ...,\ X^{(n)}$ можно считать базисом n-мерного векторного пространства.

Выберем некоторый вектор $Y^{(0)}=(y_1^{(0)},\ y_2^{(0)},\ \ldots,\ y_n^{(0)})^T$ в качестве начального итерационного приближения и рассмотрим следующую последовательность векторов:

$$Y^{(0)},\ Y^{(1)}=AY^{(0)},\ Y^{(2)}=AY^{(1)},\ ...,\ Y^{(k)}=AY^{(k-1)}. \qquad (1)$$

Выразим вектор $Y^{(k)}=(y_1^{(k)},\ y_2^{(k)},\ \ldots,\ y_n^{(k)})^T$ на k-ой итерации через вектор начального приближения.

$$Y^{(2)}=AY^{(1)}=A(AY^{(0)})=A^2Y^{(0)},\ ...,\ Y^{(k)}=A^kY^{(0)}.$$

Разложим вектор начального приближения $Y^{(0)}=(y_1^{(0)},\ y_2^{(0)},\ \ldots,\ y_n^{(0)})^T$ по базису $X^{(1)},\ X^{(2)},\ ...,\ X^{(n)}$

$$Y^{(0)}=a_1X^{(1)}+a_2X^{(2)}+...+a_nX^{(n)}, \qquad (2)$$

где $a_1,\ a_2,\ \ldots,\ a_n$ – некоторые вещественные числа.

Тогда для вектора Y^{k} получаем

$$Y^{k} = A^{k}Y^{0} = a_1A^{k}X^{1} + a_2A^{k}X^{2} + \ldots + a_nA^{k}X^{n}.$$

Из определения любого собственного вектора X_i можем записать

$$AX^{i} = \lambda_i X^{i} \Rightarrow A^{k}X^{i} = A^{k-1}AX^{i} = A^{k-1}\lambda_i X^{i} \Rightarrow A^{k}X^{i} = \lambda_i^{k}X^{i}.$$

Следовательно,

$$Y^{k} = A^{k}Y^{0} = a_1\lambda_1^{k}X^{1} + a_2\lambda_2^{k}X^{2} + \ldots + a_n\lambda_n^{k}X^{n} \qquad (3)$$

или по компонентам

$$y_i^{(k)} = a_1\lambda_1^{k}x_{1i} + a_2\lambda_2^{k}x_{2i} + \ldots + a_n\lambda_n^{k}x_{ni}, \quad i = 1, 2, \ldots, n, \qquad (4)$$

где y_i^{k} – i-ая компонента вектора Y^{k}; $x_{1i} \neq 0$, $x_{2i} \neq 0$, …, $x_{ni} \neq 0$ – i-ые компоненты собственных векторов $X^{(1)}$, $X^{(2)}$, ..., $X^{(n)}$ соответственно.

Рассмотрим случай, когда матрица A имеет единственное максимальное по модулю вещественное собственное значение.

Обозначим его λ_1 и пронумеруем остальные собственные значения в порядке убывания их модулей, так что будем иметь

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots \geq |\lambda_n|. \qquad (5)$$

Будем считать, что $a_1 \neq 0$. Если это не так, то выбираем другой вектор начального приближения Y^{0}.

В силу (4), (5) имеем для i-ой компоненты вектора Y^{k}:

$$y_i^{(k)} = a_1\lambda_1^{k}x_{1i}\left[1 + \frac{a_2x_{2i}}{a_1x_{1i}}\left(\frac{\lambda_2}{\lambda_1}\right)^{k} + \frac{a_3x_{3i}}{a_1x_{1i}}\left(\frac{\lambda_3}{\lambda_1}\right)^{k} + \ldots + \frac{a_nx_{ni}}{a_1x_{1i}}\left(\frac{\lambda_n}{\lambda_1}\right)^{k}\right] = a_1\lambda_1^{k}x_{1i}\left[1 + O\left(\frac{\lambda_2}{\lambda_1}\right)^{k}\right]. \qquad (6)$$

Так как в силу соотношения (5) все слагаемые вида $\left(\frac{\lambda_n}{\lambda_1}\right)^{k} \to 0$ при больших значениях k.

Аналогично для i-ой компоненты вектора Y^{k+1} можем записать:

$$y_i^{(k+1)} = a_1\lambda_1^{k+1}x_{1i}\left[1 + O\left(\frac{\lambda_2}{\lambda_1}\right)^{k+1}\right]. \qquad (7)$$

Рассмотрим отношение соответствующих компонент вектора Y на двух соседних итерациях:

$$\frac{y_i^{(k+1)}}{y_i^{(k)}} = \lambda_1 \frac{1 + O\left(\frac{\lambda_2}{\lambda_1}\right)^{k+1}}{1 + O\left(\frac{\lambda_2}{\lambda_1}\right)^{k}} \xrightarrow[k\to\infty]{} \lambda_1, \quad i = 1,\ 2,\ \ldots,\ n. \qquad (8)$$

Построим итерационную последовательность

$$\lambda_1^{\,0} = \frac{y_i^{\,1}}{y_i^{\,0}}, \quad \lambda_1^{\,1} = \frac{y_i^{\,2}}{y_i^{\,1}}, \quad \ldots \quad , \lambda_1^{\,k} = \frac{y_i^{\,k+1}}{y_i^{\,k}}, \quad \ldots \qquad (9)$$

Эта последовательность сходится к исходному собственному значению λ_1 при $k \to \infty$ и любом $i=1, 2, \ldots, n$.

Таким образом, для нахождения λ_1 с точностью ε нужно выбрать любой номер i и построить итерационную последовательность (9), заканчивая итерации по условию

$$\left| \lambda_1^{\,k+1} - \lambda_1^{\,k} \right| < \varepsilon .$$

Аналогичным образом можно найти собственный вектор X_1, соответствующий собственному значению λ_1:

$$Y^{(k)} = a_1\lambda_1^k X^{(1)} + a_2\lambda_2^k X^{(2)} + \ldots + a_n\lambda_n^k X^{(n)} =$$

$$= a_1\lambda_1^k \left[X^{(1)} + \frac{a_2}{a_1}\left(\frac{\lambda_2}{\lambda_1}\right)^k X^{(2)} + \ldots + \frac{a_n}{a_1}\left(\frac{\lambda_n}{\lambda_1}\right)^k X^{(n)} \right] = a_1\lambda_1^k X^{(1)} \quad \text{ïðè} \quad k \to \infty.$$

Так как собственный вектор определяется с точностью до постоянного множителя, отсюда находим

$$X^{(1)} = \frac{1}{a_1\lambda_1^k} Y_1^{\,k} \quad \text{или} \quad X^{(1)} \approx Y^{\,k} \quad \text{при} \quad k \to \infty,$$

где $Y^{\,k}$ строится по правилу (1).

Итерационным методом может быть найдено любое последующее собственное значение матрицы. Запишем формулу для нахождения собственного значения λ_2 и собственного вектора $X^{\,2}$ [20]:

$$\lambda_2 \approx \frac{y_i^{\,k+1} - \lambda_1 y_i^{\,k}}{y_i^{\,k} - \lambda_1 y_i^{\,k-1}}, \quad X^{\,2} \approx Y^{\,k+1} - \lambda_1 Y^{\,k} .$$

Пример 1. Методом итераций определить наибольшее по модулю собственное значение и соответствующий ему собственный вектор с точностью $\varepsilon = 10^{-3}$

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2.5 & 1 \\ 1 & 1 & 3 \end{pmatrix}.$$

Решение:

Строим последовательность векторов по правилу:

$$Y^{(0)},\ Y^{(1)} = AY^{(0)},\ Y^{(2)} = AY^{(1)},\ \ldots,\ Y^{(k)} = AY^{(k-1)},$$

где Y^0 – произвольный вектор.

Тогда имеем

$$\lambda_1 = \frac{y_i^{k+1}}{y_i^k},$$

где y_i^{k+1} и y_i^k – одноименные координаты двух последовательных векторов.

В качестве начального приближения возьмем $Y^0 = 1,1,1^T$, найдем $Y^1 = AY^0$,

$$Y^1 = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2.5 & 1 \\ 1 & 1 & 3 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 4.5 \\ 5 \end{pmatrix},$$

$$\lambda_1^1 = \frac{y_1^1}{y_1^0} = 4.$$

Найдем $Y^2 = AY^1$

$$Y^2 = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2.5 & 1 \\ 1 & 1 & 3 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 4.5 \\ 5 \end{pmatrix} = \begin{pmatrix} 17.5 \\ 20.25 \\ 23.5 \end{pmatrix},$$

$$\lambda_1^2 = \frac{y_1^2}{y_1^1} = 4.375.$$

Проверяем условие $\left|\lambda_1^2 - \lambda_1^1\right| < \varepsilon$. Не выполняется, следовательно продолжаем итерационный процесс.

Найдем $Y^3 = AY^2$

$$Y^{3} = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2.5 & 1 \\ 1 & 1 & 3 \end{pmatrix} \cdot \begin{pmatrix} 17.5 \\ 20.25 \\ 23.5 \end{pmatrix} = \begin{pmatrix} 78.75 \\ 91.625 \\ 108.25 \end{pmatrix},$$

$$\lambda_1^{\,3} = \frac{y_1^{\,3}}{y_1^{\,2}} = 4.5.$$

Проверяем условие $\left|\lambda_1^{\,3} - \lambda_1^{\,2}\right| < \varepsilon$. Условие не выполняется, продолжаем итерации.

Точность достигнута на 8 итерации. Получаем

$$Y^{8} = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2.5 & 1 \\ 1 & 1 & 3 \end{pmatrix} \cdot \begin{pmatrix} 3.372 \cdot 10^4 \\ 3.924 \cdot 10^4 \\ 4.69 \cdot 10^4 \end{pmatrix} = \begin{pmatrix} 1.536 \cdot 10^5 \\ 1.787 \cdot 10^5 \\ 2.137 \cdot 10^5 \end{pmatrix},$$

$\lambda_1^{\,8} = \frac{y_1^{\,8}}{y_1^{\,7}} = 4.5552$ – собственное значение.

Собственный вектор

$X^{1} \approx Y^{k} \Rightarrow X^{1} \approx Y^{8} = 1.536 \cdot 10^5; 1.787 \cdot 10^5; 2.137 \cdot 10^5\ ^{T}$

или $X^{1} = 0.719; 0.836; 1\ ^{T}$.

Пример 2. В рамках итерационного метода нахождения наибольшего по модулю собственного значения и соответствующего собственного вектора реализовать метод умножения матрицы на вектор.

Данный метод описан в классе «SquareMatrix».

Реализация на языке программирования C++ может иметь следующий вид:

```
Vector SquareMatrix :: multiply_vec (SquareMatrix A, Vector f){
  /*Результирующий вектор*/
  double *res = new double[A.getRowCount()];
  for (int i = 0; i < A.getRowCount(); i++){
   for (int j = 0; j < A.getColCount(); j++){
     res[i] += A.elements[i][j] * f.getElement(j);
                        }
                    }
              }
   /*  Создание вектора  */
  return Vector(A.getRowCount(), res);
}
```

Лабораторная работа № 14

Цель: изучить итерационный степенной метод нахождения наибольшего по модулю собственного значения и соответствующего собственного вектора.

Задание

1. Разработайте класс «Частичная проблема нахождения собственных значений» («PartialProblem»), который наследуется от класса «Итерационные методы нахождения собственных значений» («IterationMethodsE»). В данном классе реализуйте итерационный метод («iterationMethod») нахождения наибольшего по модулю собственного значения и соответствующего собственного вектора.

Для реализации метода используйте объекты матричных классов «SquareMatrix» и «Vector». Доступ к элементам осуществляйте с помощью методов getElement и setElement, реализованных в данных классах. Для операции умножения матрицы на вектор используйте метод, реализованный в классе «SquareMatrix».

2. Методом итераций определите наибольшее по модулю собственное значение и соответствующий ему собственный вектор матрицы ($\varepsilon = 0.001$) в соответствии с вариантом.

3. Решите ту же задачу, используя пакет для математических вычислений.

4. Сравните результат выполнения п. 2 с решением, полученным в п. 3.

Варианты заданий

№ 1

$$A = \begin{pmatrix} 2.1 & 1 & 1.1 \\ 1 & 2.6 & 1.1 \\ 1.1 & 1.1 & 3.1 \end{pmatrix}$$

№ 2

$$A = \begin{pmatrix} 2.4 & 1 & 1.4 \\ 1 & 2.9 & 1.4 \\ 1.4 & 1.4 & 3.4 \end{pmatrix}$$

№ 3

$$A = \begin{pmatrix} 1.3 & 0.4 & 0.5 \\ 0.4 & 1.3 & 0.3 \\ 0.5 & 0.3 & 1.3 \end{pmatrix}$$

№ 4

$$A = \begin{pmatrix} 1.6 & 0.7 & 0.8 \\ 0.7 & 1.6 & 0.3 \\ 0.8 & 0.3 & 1.6 \end{pmatrix}$$

№ 5

$$A = \begin{pmatrix} 2.2 & 1 & 1.2 \\ 1 & 2.7 & 1.2 \\ 1.2 & 1.2 & 3.2 \end{pmatrix}$$

№ 6

$$A = \begin{pmatrix} 2.5 & 1 & 1.5 \\ 1 & 3 & 1.5 \\ 1.5 & 1.5 & 3.5 \end{pmatrix}$$

№ 7

$$A = \begin{pmatrix} 1.4 & 0.5 & 0.6 \\ 0.5 & 1.4 & 0.3 \\ 0.6 & 0.3 & 1.4 \end{pmatrix}$$

№ 8

$$A = \begin{pmatrix} 1.7 & 0.8 & 0.9 \\ 0.8 & 0.7 & 0.3 \\ 0.9 & 0.3 & 1.7 \end{pmatrix}$$

№ 9

$$A=\begin{pmatrix} 2.3 & 1 & 1.3 \\ 1 & 2.8 & 1.3 \\ 1.3 & 1.3 & 3.3 \end{pmatrix}$$

№ 10

$$A=\begin{pmatrix} 2.6 & 1 & 1.6 \\ 1 & 3.1 & 1.6 \\ 1.6 & 1.6 & 3.6 \end{pmatrix}$$

№ 11

$$A=\begin{pmatrix} 3.5 & 1 & 2.5 \\ 1 & 4 & 2.5 \\ 2.5 & 2.5 & 4.5 \end{pmatrix}$$

№ 12

$$A=\begin{pmatrix} 1.8 & 0.9 & 1 \\ 0.9 & 1.8 & 0.3 \\ 1 & 0.3 & 1.8 \end{pmatrix}$$

№ 13

$$A=\begin{pmatrix} 1.5 & 0.6 & 0.7 \\ 0.6 & 1.5 & 0.3 \\ 0.7 & 0.3 & 1.5 \end{pmatrix}$$

№ 14

$$A=\begin{pmatrix} 2.7 & 1 & 1.7 \\ 1 & 3.2 & 1.7 \\ 1.7 & 1.7 & 3.7 \end{pmatrix}$$

№ 15

$$A=\begin{pmatrix} 1.4 & 1.2 & -1.3 \\ 1.2 & 0.9 & 0.4 \\ -1.3 & 0.4 & 0.8 \end{pmatrix}$$

№ 16

$$A=\begin{pmatrix} 3.2 & 1 & 2.2 \\ 1 & 3.7 & 2.2 \\ 2.2 & 2.2 & 4.2 \end{pmatrix}$$

П 4.3 QR-алгоритм для нахождения собственных значений матрицы

При решении полной проблемы собственных значений для несимметричных матриц эффективным является подход, основанный на приведении матриц к подобным, имеющим треугольный или квазитреугольный вид. Одним из наиболее распространенных методов этого класса является QR-алгоритм, позволяющий находить как вещественные, так и комплексные собственные значения.

Основная идея QR-алгоритма состоит в разложении матрицы A в произведение ортогональной и верхней треугольной матрицы. Предполагаем, что все собственные значения λ_1, λ_2, ..., λ_n матрицы A различны по абсолютной величине и $\lambda_1 > \lambda_2 > ... > \lambda_n$.

Процедура QR-разложения, рассмотренная в П 2.2, многократно используется в R-алгоритме вычисления собственных значений.

Осуществляя поочередно факторизацию и перестановку сомножителей, получаем последовательность матриц [20]

$$A_k = Q_k R_k, \quad A_{k+1} = R_k Q_k, \quad k = 0, 1, \ldots, \qquad (1)$$

где Q_k − ортогональная матрица, R_k − верхняя треугольная матрица, $A_0 = A$.

Таким образом, каждая итерация реализуется в два этапа. На первом этапе осуществляется разложение матрицы A_k в произведение ортогональной Q_k и верхней треугольной R_k матриц, а на втором – полученные матрицы перемножаются в обратном порядке.

Из равенства (1) получаем $R_k = Q_k^{-1} A_k$, отсюда $A_{k+1} = Q_k^{-1} R_k Q_k$. Получаем, что все матрицы A_k подобны между собой и подобны исходной матрице *A*.

При отсутствии у матрицы кратных собственных значений последовательность A_k сходится к верхней треугольной матрице (в случае, когда все собственные значения вещественны) или к верхней квазитреугольной матрице (если имеются комплексно-сопряженные пары собственных значений).

Таким образом, каждому вещественному собственному значению будет соответствовать столбец со стремящимися к нулю поддиагональными элементами и в качестве критерия сходимости итерационного процесса для таких собственных значений можно использовать следующее равенство:

$$\left(\sum_{i=m+1}^{n} \left(a_{im}^{(k)} \right)^2 \right)^{1/2} \le \varepsilon, \quad m = 1,\ 2,\ \ldots,\ n-1.$$

При этом соответствующее собственное значение принимается равным диагональному элементу данного столбца.

Каждой комплексно-сопряженной паре соответствует диагональный блок размерностью 2x2. Принципиально то, что элементы этих блоков изменяются от итерации к итерации без видимой закономерности, в то время как комплексно-сопряженные собственные значения, определяемые каждым блоком, имеют тенденцию к сходимости. Это обстоятельство необходимо учитывать при формировании критерия выхода из итерационного процесса. Если в ходе итераций прослеживается комплексно-сопряженная пара собственных значений, соответствующая блоку, образуемому элементами *j*-го и *j+1*-го столбцов, то, несмотря на значительное изменение в ходе итераций самих этих элементов, собственные значения, соответствующие данному блоку и определяемые из решения квадратного уравнения

$$a_{jj}^{k} - \lambda^{k} \quad a_{j+1\,j+1}^{k} - \lambda^{k} = a_{jj+1}^{k} a_{j+1\,j}^{k},$$

начиная с некоторого *k* отличаются незначительно.

Для практического QR-алгоритма нет завершенной теории сходимости. На практике алгоритм работает всегда или почти всегда [5].

Пример 1. Используя QR-алгоритм, найти собственные значения матрицы $A = \begin{pmatrix} 1 & 3 & 1 \\ 1 & 1 & 4 \\ 4 & 3 & 1 \end{pmatrix}$ с точностью $\varepsilon = 0.01$.

Решение:

Выберем вектор w_1 согласно формуле (5), рассмотренной в П 2.2:

$$w_1 = \begin{pmatrix} \mu_1 \cdot a_{11} - s_1 \\ \mu_1 \cdot a_{21} \\ \mu_1 \cdot a_{31} \end{pmatrix},$$

где $\mu_1 = \left[2 \cdot s_1 \cdot (s_1 - a_{11}) \right]^{-1/2}$, $s_1 = \left[\sum_{j=1}^{3} a_{j1}^{\ 2} \right]^{1/2}$.

Тогда $w_1 = \begin{pmatrix} -0.618 \\ 0.191 \\ 0.763 \end{pmatrix}$.

Построим матрицу A_1 согласно формуле (6), рассмотренной в П 2.2:

$$A_1 = H_1 A = \left(E - 2 \cdot w_1 \cdot w_1^T \right) A,$$

$$A_1 = \begin{pmatrix} 4.243 & 3.771 & 2.121 \\ 0 & 0.762 & 3.654 \\ 0 & 2.049 & -0.383 \end{pmatrix}.$$

Выберем вектор w_2:

$$w_2 = \begin{pmatrix} \mu_2 \cdot 0 \\ \mu_2 \cdot (a_{22}^1 - s_2) \\ \mu 2 \cdot a_{32}^1 \end{pmatrix},$$

где $\mu_2 = [2 \cdot s_2 \cdot (s_2 - a_{22}^1)]^{-1/2}$, $s_2 = \left[\sum_{j=2}^{3} a_{j2}^{1\ 2} \right]^{1/2}$.

Тогда $w_2 = \begin{pmatrix} 0 \\ -0,571 \\ 0.821 \end{pmatrix}$.

Построим матрицу A_2 по формуле (6) из П 2.2:

$$A_2 = H_2 A = \left(E - 2 \cdot w_2 \cdot w_2^T\right) A_1,$$

$$A_2 = \begin{pmatrix} 4.243 & 3.771 & 2.121 \\ 0 & 2.186 & 0.915 \\ 0 & 0 & 3.558 \end{pmatrix}.$$

Построим матрицу Q согласно формуле (8), описанной в П 2.2:

$$\mathrm{Q} = H_1 H_2 = \left(E - 2 \cdot w_1 \cdot w_1^T\right)\left(E - 2 \cdot w_2 \cdot w_2^T\right),$$

$$\mathrm{Q} = \begin{pmatrix} 0.236 & 0.966 & -0.108 \\ 0.236 & 0.051 & 0.97 \\ 0.943 & -0.254 & -0.216 \end{pmatrix}.$$

Убедимся, что матрица Q – ортогональна.

$$\mathrm{Q}^{-1} = \begin{pmatrix} 0.236 & 0.236 & 0.943 \\ 0.966 & 0.051 & -0.254 \\ -0.108 & 0.97 & -0.216 \end{pmatrix} = \mathrm{Q}^{\mathrm{T}}.$$

Построим верхнюю треугольную матрицу R по формуле (7) из П 2.2:

$$\mathrm{R} = H_2 H_1 A = \left(E - 2 \cdot w_2 \cdot w_2^T\right)\left(E - 2 \cdot w_1 \cdot w_1^T\right) A,$$

$$\mathrm{R} = \begin{pmatrix} 4.243 & 3.771 & 2.121 \\ 0 & 2.186 & 0.915 \\ 0 & 0 & 3.558 \end{pmatrix}.$$

После первой итерации матрица A вычисляется по формуле $A^1 = R \cdot Q$ и имеет вид:

$$A^1 = \begin{pmatrix} 3.889 & 3.75 & 2.745 \\ 1.378 & -0.121 & 1.924 \\ 3.355 & -0.904 & -0.767 \end{pmatrix}.$$

Первая итерация завершена. Поддиагональные элементы матрицы достаточно велики, поэтому итерационный процесс необходимо продолжать.

Продолжая итерационный процесс, получим соответственно на 6-й и 7-й итерациях следующие матрицы:

$$A^{6} = \begin{pmatrix} 6.34 & 0.945 & -0.732 \\ 0.035 & -2.529 & 1.69 \\ 0.023 & -1.86 & -0.811 \end{pmatrix},$$

$$A^{7} = \begin{pmatrix} 6.34 & 0.27 & 1.13 \\ -0.001 & -2.01 & -2.58 \\ 0.001 & 0.98 & -1.33 \end{pmatrix}.$$

Видно, что поддиагональные элементы первого столбца становятся достаточно малыми, и, следовательно, диагональный элемент $a_{11}^{(}$ может быть принят в качестве собственного значения. В то же время отчетливо прослеживается комплексно-сопряженная пара собственных значений, соответствующая блоку, образуемому элементами второго и третьего столбцов. Собственные значения, соответствующие данному блоку, определяются из решения квадратного уравнения

$$a_{22}^{7} - \lambda^{7} \quad a_{33}^{7} - \lambda^{7} \quad = a_{23}^{7} a_{32}^{7} .$$

Таким образом, окончательное решение задачи можно записать в виде: $\lambda_1 = 6.34$, $\lambda_2 = -1.67 + 1.55i$, $\lambda_3 = -1.67 - 1.55i$.

Лабораторная работа № 15

Цель: изучить QR-алгоритм для нахождения собственных значений матрицы.

Задание

1. Разработайте класс «Полная проблема нахождения собственных значений» («CompleteProblem»), который наследуется от класса «Итерационные методы нахождения собственных значений» («IterationMethodsE»). В данном классе реализуйте QR-алгоритм («qrMethod») для нахождения собственных значений матрицы, используя метод для QR-разложения (QR_decomposition), описанный в классе «Алгоритмы факторизации» («FactorizationAlgorithms»).

Для реализации метода используйте объекты и методы матричных классов «SquareMatrix», «EMatrix», «Vector». Для выполнения основных матричных операций (перемножение матриц, вычитание матриц) ис-

пользуйте методы, реализованные в данных матричных классах.

2. Используя QR-алгоритм, найдите собственные значения матрицы в соответствии с вариантом.

3. Решите ту же задачу, используя пакет для математических вычислений.

4. Сравните результат выполнения п. 2 с решением, полученным в п. 3.

Варианты заданий

№ 1

$$A=\begin{pmatrix}1 & 4 & 1 & 3\\ 0 & -1 & 3 & -1\\ 3 & 1 & 0 & 2\\ 1 & -2 & 5 & 1\end{pmatrix}$$

№ 2

$$A=\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & 4 & 2 & 3\\ 1 & 10 & 3 & 6\\ 6 & 10 & 1 & 4\end{pmatrix}$$

№ 3

$$A=\begin{pmatrix}1 & 2 & 3 & -2\\ 2 & -1 & -2 & -3\\ 3 & 2 & -1 & 2\\ 2 & -3 & 2 & 1\end{pmatrix}$$

№ 4

$$A=\begin{pmatrix}-2 & 2 & 1 & 0\\ 1 & -3 & 3 & 7\\ 2 & -1 & 2 & -3\\ -5 & 4 & -1 & 2\end{pmatrix}$$

№ 5

$$A=\begin{pmatrix}1 & -1 & -1 & 1\\ -1 & 2 & 2 & 0\\ 0 & -1 & 1 & 4\\ 1 & 1 & -1 & -1.5\end{pmatrix}$$

№ 6

$$A=\begin{pmatrix}3 & -2 & 2 & 0\\ 2 & 1 & 1 & -2\\ 3 & -1 & 2 & 1\\ 1 & 2 & -1 & -1\end{pmatrix}$$

№ 7

$$A=\begin{pmatrix}5 & -4 & 0 & 2\\ -1 & 1 & 1 & -1\\ 2 & 3 & 1 & -6\\ 1 & 0 & 2 & -1\end{pmatrix}$$

№ 8

$$A=\begin{pmatrix}4 & -1 & 0 & 1\\ 3 & 2 & -1 & 2\\ 0 & 2 & 2 & 1\\ -1 & 1 & -3 & -1\end{pmatrix}$$

№ 9

$$A=\begin{pmatrix}1 & 2 & 0 & 1\\ -1 & -3 & 3 & -1\\ 0 & 4 & -10 & 2\\ 1 & -1 & 2 & -1\end{pmatrix}$$

№ 10

$$A=\begin{pmatrix}1 & 4 & -3 & 0\\ 0 & 4 & 1 & 2\\ -1 & 2 & 4 & 1\\ 1 & 0 & -1 & 5\end{pmatrix}$$

№ 11

$$A=\begin{pmatrix}2 & -1 & 1 & 2\\ 1 & 2 & -1 & 1\\ 3 & 0 & -1 & -3\\ 1 & -1 & 1 & 3\end{pmatrix}$$

№ 12

$$A=\begin{pmatrix}2 & 1 & 2 & 0\\ -1 & -3 & 3 & -1\\ 1 & 3 & -8 & 1\\ 1 & -1 & 2 & -1\end{pmatrix}$$

№ 13

$$A=\begin{pmatrix} 2 & 3 & 0 & 1 \\ -1 & 1 & 3 & 0 \\ 0 & 2 & -1 & 1 \\ 3 & -1 & 1 & -2 \end{pmatrix}$$

№ 14

$$A=\begin{pmatrix} -1 & 3 & 3 & 2 \\ -2 & 2 & 2 & 1 \\ 0 & 1 & 2 & 0 \\ -1 & 3 & 3 & 3 \end{pmatrix}$$

№ 15

$$A=\begin{pmatrix} 1 & 0 & 3 & 4 \\ 0 & 1 & 5 & 1 \\ -3 & 4 & 10 & 1 \\ 0 & -6 & 0 & -1 \end{pmatrix}$$

№ 16

$$A=\begin{pmatrix} 3 & 1 & 3 & 3 \\ 2 & 2 & 1 & 3 \\ 1 & 0 & 2 & 0 \\ 1 & 1 & 1 & 3 \end{pmatrix}$$

П 4.4 Метод Якоби для нахождения собственных значений и собственных векторов

Итерационный метод вращения (метод Якоби) решает полную проблему нахождения собственных значений и собственных векторов вещественной симметрической матрицы без использования характеристического уравнения.

Вычисление всех собственных значений и собственных векторов можно свести к отысканию такой ортогональной матрицы T, для которой произведение $B=T^TAT$ представляет диагональную матрицу, причем столбцы матрицы T будут являться соответствующими собственными векторами матрицы A. Матрица T находится как предел бесконечного произведения элементарных матриц вращений, каждая из которых имеет вид [20]:

$$T_m(\varphi)=\begin{bmatrix} 1 & & & & & & & \\ & \ddots & & & & & \mathbf{0} & \\ & & 1 & & & & & \\ & & & \cos\varphi & \dots & -\sin\varphi & & \\ & & & \vdots & \ddots & \vdots & & \\ & & & \sin\varphi & \dots & +\cos\varphi & & \\ & & & & & & 1 & \\ & \mathbf{0} & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ i\text{-я строка} \\ \\ k\text{-я строка} \\ \\ \\ \end{matrix}$$

i-й столбец k-й столбец

Если необходимо обратить в нуль элемент a_{ik} матрицы A, то $cos\varphi$ и $sin\varphi$ нужно выбрать по формулам

$$c = \cos\phi = \sqrt{\frac{1+z}{2}}, \quad s = \sin\phi = \varepsilon\sqrt{\frac{1-z}{2}}, \tag{1}$$

где $z = \sqrt{1 - \frac{2a_{ik}^{\ 2}}{a_{ii} - a_{kk}^{\ 2} + 2a_{ik}^{\ 2}}}$, $\varepsilon = \begin{cases} sign a_{ik}, a_{ii} = a_{kk}, \\ sign \frac{a_{ii} - a_{kk}}{a_{ik}}, a_{ii} \neq a_{kk}. \end{cases}$

Тогда получим матрицу $B = T_m^T A T_m$ с измененными i-м и k-м столбцами и строками:

$$\begin{aligned} b_{ii} &= \cos^2\varphi a_{ii} + \sin^2\varphi a_{kk} + 2\cos\varphi\sin\varphi a_{ik}, \\ b_{kk} &= \sin^2\varphi a_{ii} + \cos^2\varphi a_{kk} - 2\cos\varphi\sin\varphi a_{ik}, \\ b_{ik} &= b_{ki} = 0, \\ b_{ij} &= b_{ji} = \cos\varphi a_{ji} + \sin\varphi a_{jk,} \\ b_{kj} &= b_{jk} = -\sin\varphi a_{ji} + \cos\varphi a_{jk,} \end{aligned} \tag{2}$$

$j = 1, \ldots, n, \quad j \neq i, \quad j \neq k,$

$b_{jl} = a_{jl}$ *â îñòàëüíûõ ñëó÷àÿõ*

Отметим, что выполняется соотношение $b_{ii}^2 + b_{kk}^2 = a_{ii}^2 + a_{kk}^2 + 2a_{ik}^2$, т.е. сумма квадратов диагональных элементов увеличивается. Элементы, которые однажды обратились в нуль, при последующих шагах снова могут стать ненулевыми. Если на каждом шаге данного преобразования подобия брать максимальный по модулю наддиагональный элемент преобразуемой матрицы, то в пределе получится диагональная матрица. Исследование сходимости является сложным вопросом и в данном пункте не рассматривается.

Заметим, что по мере того, как A_m при $m \to \infty$ превращается в диагональную матрицу, на диагонали которой стоят собственные значения в некоторой последовательности, зависящей от выбранных вначале пар i, k, в столбцах матрицы $T_1 \ldots T_m$ появляются стоящие в соответствующей последовательности нормированные собственные векторы.

В качестве критерия окончания итерационного процесса используется условие малости суммы квадратов внедиагональных элементов [10].

Пример 1. Используя метод Якоби, найти собственные значения и векторы матрицы с точностью $\varepsilon = 0.001$

$$A = \begin{pmatrix} 2.2 & 1 & 0.5 & 2 \\ 1 & 1.3 & 2 & 1 \\ 0.5 & 2 & 0.5 & 1.6 \\ 2 & 1 & 1.6 & 2 \end{pmatrix}.$$

Решение:

Выберем максимальный по модулю наддиагональный элемент матрицы A.

Пусть $(i, k) = (1, 4)$, $a_{14} = 2$. По формулам (1) вычислим c и s: $c = 0{,}7245$; $s = 0{,}6892$. Тогда матрица вращения T_1 будет иметь вид:

$$T_1 = \begin{pmatrix} 0.7245 & 0 & 0 & -0.6892 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.6892 & 0 & 0 & 0.7245 \end{pmatrix}.$$

Матрица B_1, вычисленная, согласно $B_1 = T_1^T A T_1$, будет иметь вид:

$$B_1 = \begin{pmatrix} 4.1025 & 1.4138 & 1.465 & 0 \\ 1.4138 & 1.3 & 2 & 0.0353 \\ 1.465 & 2 & 0.5 & 0.8147 \\ 0 & 0.0353 & 0.8147 & 0.0975 \end{pmatrix}.$$

Первая итерация завершена.

Выберем максимальный по модулю наддиагональный элемент матрицы B_1. Пусть $(i, k) = (2, 3)$, $b_{23}^1 = 2$. Проверим условие окончания итерационного процесса

$$1.4138^{\,2} + 1.465^{\,2} + 2^2 + 0.0353^{\,2} + 0.0975^{\,2}\,{}^{1/2} > \varepsilon,$$

следовательно, процесс необходимо продолжить. По формулам (1) вычислим c и s: $c = 0{,}7733$; $s = 0{,}643$. Тогда матрица вращения T_2 будет иметь вид:

$$T_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.7733 & -0.643 & 0 \\ 0 & 0.643 & 0.7733 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Матрица B_2, вычисленная, согласно $B_2 = T_2^T B_1 T_2$, будет иметь вид:

$$B_2 = \begin{pmatrix} 4.1025 & 2.0221 & 0.2367 & 0 \\ 2.0221 & 2.9396 & 0 & 0.5438 \\ 0.2367 & 0 & -1.1396 & 0.6067 \\ 0 & 0.5438 & 0.6067 & 0.0975 \end{pmatrix}.$$

Вторая итерация завершена.

Выберем максимальный по модулю наддиагональный элемент матрицы B_2. Пусть $(i, k) = (1, 2)$, $b_{23}^2 = 2.0221$. Условие окончания итерационного процесса $2.0221^{\,2} + 0.2367^{\,2} + 0.5438^{\,2} + 0.6067^{\,2})^{1/2} > \varepsilon$ не выполняется, переходим к следующей итерации. По формулам (1) вычислим c и s: $c = 0{,}7989$; $s = 0{,}6015$. Тогда матрица вращения T_3 будет иметь вид:

$$T_3 = \begin{pmatrix} 0.7989 & -0.6015 & 0 & 0 \\ 0.6015 & 0.7989 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Матрица B_3, вычисленная, согласно $B_3 = T_3^T B_2 T_3$, будет иметь вид:

$$B_3 = \begin{pmatrix} 5.6251 & 0 & 0.1891 & 0.3271 \\ 0 & 1.417 & -0.1424 & 0.4344 \\ 0.1891 & -0.1424 & -1.1396 & 0.6067 \\ 0.3271 & 0.4344 & 0.6067 & 0.0975 \end{pmatrix}.$$

Продолжаем итерационный процесс до тех пор, пока не будет достигнута заданная точность наддиагональных элементов.

На двенадцатой итерации имеем:

$$B_{12} = \begin{pmatrix} 5.652 & 0 & 0 & 0 \\ 0 & 1.5454 & 0.0002 & 0 \\ 0 & 0.0002 & -1.4201 & 0 \\ 0 & 0 & 0 & 0.2226 \end{pmatrix}.$$

Так как сумма квадратов наддиагональных элементов меньше $\varepsilon = 0.001$, то процесс завершается.

На главной диагонали матрицы B_{12} находятся собственные значения матрицы A: $\lambda_1 = 5.652$, $\lambda_2 = 1.5454$, $\lambda_3 = -1.4201$, $\lambda_4 = 0.2226$.

Столбцы матрицы $T_1 \cdot T_2 \cdot T_3 \cdot ... \cdot T_{12}$ являются собственными векторами матрицы A

$$T_1 \cdot T_2 \cdot ... \cdot T_{12} = \begin{pmatrix} 0.5317 & -0.6289 & 0.222 & -0.5219 \\ 0.4462 & 0.5726 & -0.5159 & -0.4549 \\ 0.4088 & 0.4856 & 0.7573 & 0.1534 \\ 0.5925 & -0.2018 & -0.3333 & 0.7051 \end{pmatrix}.$$

$$\vec{x}_1 = \begin{pmatrix} 0.5317 \\ 0.4462 \\ 0.4088 \\ 0.5925 \end{pmatrix}, \quad \vec{x}_2 = \begin{pmatrix} -0.6289 \\ 0.5726 \\ 0.4856 \\ -0.2018 \end{pmatrix}, \quad \vec{x}_3 = \begin{pmatrix} 0.222 \\ -0.5159 \\ 0.7573 \\ -0.3333 \end{pmatrix}, \quad \vec{x}_4 = \begin{pmatrix} -0.5219 \\ -0.4549 \\ 0.1534 \\ 0.7051 \end{pmatrix}.$$

Лабораторная работа № 16

Цель: изучить метод Якоби для нахождения собственных значений и собственных векторов.

Задание

1. В классе «JacobiMatrix» («Матрица Якоби»), который наследуется от класса «SquareMatrix» («Квадратная матрица»), сформируйте матрицу вращения.

2. В классе «Полная проблема нахождения собственных значений» («CompleteProblem») реализуйте метод Якоби («jacobiMethod») для нахождения собственных значений и собственных векторов матрицы.

Для реализации метода используйте объекты матричных классов «SquareMatrix» и «JacobiMatrix». Для выполнения основных матричных операций (перемножение матриц, транспонирование) используйте методы, реализованные в классе «SquareMatrix».

3. Используя метод Якоби, найдите собственные значения и собственные векторы матрицы в соответствии с вариантом.

4. Решите ту же задачу, используя пакет для математических вычислений.

5. Сравните результат выполнения п. 3 с решением, полученным в п. 4.

Варианты заданий

№ 1

$$A=\begin{pmatrix} 1 & 1.5 & 2.5 & 3.5 \\ 1.5 & 1 & 2 & 1.6 \\ 2.5 & 2 & 1 & 1.7 \\ 3.5 & 1.6 & 1.7 & 1 \end{pmatrix}$$

№ 2

$$A=\begin{pmatrix} 1 & 1.2 & 2 & 0.5 \\ 1.2 & 1 & 0.4 & 1.2 \\ 2 & 0.4 & 2 & 1.5 \\ 0.5 & 1.2 & 1.5 & 1 \end{pmatrix}$$

№ 3

$$A=\begin{pmatrix} 1 & 1.2 & 2 & 0.5 \\ 1.2 & 1 & 0.5 & 1 \\ 2 & 0.5 & 2 & 1.5 \\ 0.5 & 1 & 1.5 & 0.5 \end{pmatrix}$$

№ 4

$$A=\begin{pmatrix} 2.5 & 1 & -0.5 & 2 \\ 1 & 2 & 1.2 & 0.4 \\ -0.5 & 1.2 & -1 & 1.5 \\ 2 & 0.4 & 1.5 & 1 \end{pmatrix}$$

№ 5

$$A=\begin{pmatrix} 2 & 1 & 1.4 & 0.5 \\ 1 & 1 & 0.5 & 1 \\ 1.4 & 0.5 & 2 & 1.2 \\ 0.5 & 1 & 1.2 & 0.5 \end{pmatrix}$$

№ 6

$$A=\begin{pmatrix} 2 & 1.2 & -1 & 1 \\ 1.2 & 0.5 & 2 & -1 \\ -1 & 2 & -1.5 & 0.2 \\ 1 & -1 & 0.2 & 1.5 \end{pmatrix}$$

№ 7

$$A=\begin{pmatrix} 2 & 1.5 & 3.5 & 4.5 \\ 1.5 & 2 & 2 & 1.6 \\ 3.5 & 2 & 2 & 1.7 \\ 4.5 & 1.6 & 1.7 & 2 \end{pmatrix}$$

№ 8

$$A=\begin{pmatrix} 1 & 0.5 & 1.2 & -1 \\ 0.5 & 2 & -0.5 & 0 \\ 1.2 & -0.5 & -1 & 1.4 \\ -1 & 0 & 1.4 & 1 \end{pmatrix}$$

№ 9

$$A=\begin{pmatrix} 1.2 & 0.5 & 2 & 1 \\ 0.5 & 1 & 0.8 & 2 \\ 2 & 0.8 & 1 & 1 \\ 1 & 2 & 1 & 2 \end{pmatrix}$$

№ 10

$$A=\begin{pmatrix} 0.5 & 1.2 & 1 & 0.9 \\ 1.2 & 2 & 0.5 & 1.2 \\ 1 & 0.5 & 1 & 1 \\ 0.9 & 1.2 & 1 & 2.2 \end{pmatrix}$$

№ 11

$$A=\begin{pmatrix} 1.6 & 0.4 & 1 & 2 \\ 0.4 & 1 & 0.5 & 1 \\ 1 & 0.5 & 0 & 0.2 \\ 2 & 1 & 0.2 & 0.5 \end{pmatrix}$$

№ 12

$$A=\begin{pmatrix} 2 & 1.5 & 4.5 & 5.5 \\ 1.5 & 3 & 2 & 1.6 \\ 4.5 & 2 & 3 & 1.7 \\ 5.5 & 1.6 & 1.7 & 3 \end{pmatrix}$$

№ 13

$$A=\begin{pmatrix} 1.6 & 1 & 1.4 & 1 \\ 1 & 1 & 0.5 & 2 \\ 1.4 & 0.5 & 2 & 1.2 \\ 1 & 2 & 1.2 & 0.5 \end{pmatrix}$$

№ 14

$$A=\begin{pmatrix} 2.4 & 0.5 & 2 & 1 \\ 0.5 & 1 & 0.8 & 2 \\ 2 & 0.8 & 1 & 0.5 \\ 1 & 2 & 0.5 & 1.2 \end{pmatrix}$$

№ 15

$$A=\begin{pmatrix} 0.5 & 1.2 & 2 & 1 \\ 1.2 & 2 & 0.5 & 1.2 \\ 2 & 0.5 & 1 & 0.5 \\ 1 & 1.2 & 0.5 & 1.6 \end{pmatrix}$$

№ 16

$$A=\begin{pmatrix} 1.8 & 1.6 & 1.7 & 1.8 \\ 1.6 & 2.8 & 1.5 & 1.3 \\ 1.7 & 1.5 & 3.8 & 1.4 \\ 1.8 & 1.3 & 1.4 & 4.8 \end{pmatrix}$$

П Р И Л О Ж Е Н И Я

Приложение 1

Основные сведения о матрицах

- Квадратной матрицей размера $n \times n$ называется совокупность $n \cdot n$ чисел, расположенных в виде квадратной таблицы, содержащей n строк и n столбцов.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}.$$

- Главная диагональ – это часть матрицы, состоящая из элементов $a_{11},\ a_{22},\ ...,\ a_{nn}$.

- Побочная диагональ – это часть матрицы, состоящая из элементов $a_{1,n},\ a_{2,n-1},\ ...,\ a_{n,1}$.

- Квадратная матрица называется диагональной, если $a_{ij} = 0$ при $i \neq j$.

$$A = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix}.$$

- Верхняя треугольная матрица – все элементы, расположенные ниже главной диагонали, равны нулю.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix}.$$

- Нижняя треугольная матрица – все элементы, расположенные выше главной диагонали, равны нулю.

$$A=\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}.$$

- Трехдиагональная матрица – матрица, у которой все ненулевые элементы располагаются на трех диагоналях: главной, первой сверху и первой снизу.

$$A=\begin{pmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{pmatrix}.$$

- Ленточная – квадратная матрица, все ненулевые элементы которой примыкают к главной диагонали.

$$A=\begin{pmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & a_{42} & a_{43} & a_{44} \end{pmatrix}.$$

- Единичная матрица – эта матрица, на главной диагонали которой стоят единицы, а остальные элементы равны нулю.

$$E=\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}.$$

- Расширенная матрица – это матрица, полученная «склеиванием» двух матриц, как правило, для осуществления одних и тех же элементарных преобразований со строками сразу в двух матрицах. Например, для матриц $A=\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$, $B=\begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix}$ расширенная матрица будет выглядеть следующим образом:

$$(A\,|\,B)=\left(\begin{array}{ccc|ccc} a_{11} & a_{12} & a_{13} & b_{11} & b_{12} & b_{13} \\ a_{21} & a_{22} & a_{23} & b_{21} & b_{22} & b_{23} \\ a_{31} & a_{32} & a_{33} & b_{31} & b_{32} & b_{33} \end{array}\right).$$

- Суммой матриц A и B называется матрица C такая, что элемент $c_{ij}=a_{ij}+b_{ij}$.

- Матрица $C=A\cdot B$ называется произведением матриц A и B, если элемент

$$c_{ij}=\sum_{k=1}^{n}a_{ik}\cdot b_{kj}=a_{i1}\cdot b_{1j}+a_{i2}\cdot b_{2j}+...+a_{in}\cdot b_{nj}\quad \forall\, i,\ j=1,\ ...,\ n. \qquad (1)$$

  Причем умножение матриц не является коммутативным, т.е. $A\cdot B\neq B\cdot A$ (за исключением умножения на единичную матрицу: $A\cdot E=E\cdot A$). Поэтому для $C=B\cdot A$ будет справедлива формула:

$$c_{ij}=\sum_{k=1}^{n}b_{ik}\cdot a_{kj}=b_{i1}\cdot a_{1j}+b_{i2}\cdot a_{2j}+...+b_{in}\cdot a_{nj}\quad \forall\, i,\ j=1,\ ...,\ n. \qquad (2)$$

  Данные формулы (1) и (2) будут называться соответственно левостороннее и правостороннее произведение матриц.

- Вектор – это матрица, состоящая из одного столбца (вектор- столбец), т.е. $j=1$. $X=\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$.

- При умножении матрицы A на вектор X мы получим некоторый вектор $f=\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$, элементы которого вычисляются по формуле:

$$f_i=\sum_{j=1}^{n}a_{ij}\cdot x_j\quad \forall i,\ j=1,\ ...,\ n.$$

- Квадратные матрицы A и B одинакового порядка называются подобными, если существует невырожденная матрица P такого же порядка, такая что $B=P^{-1}\cdot A\cdot P$.

- Матрицей, обратной данной матрице A, называется матрица A^{-1} такая, что произведение A на A^{-1} равняется единичной матрице $A \cdot A^{-1} = A^{-1} \cdot A = E$.

- Ортогональной называют такую квадратную матрицу A, для которой выполняется равенство $A^{-1} = A^T$.

- Невырожденной называют квадратную матрицу A, определитель которой не равен 0.

- Квадратная матрица называется симметрической, если $a_{ij} = a_{ji}$ для любых i, j, т.е. ее элементы расположены симметрично относительно главной диагонали.

- Транспонированная матрица – матрица A^T, полученная из исходной матрицы A заменой строк на столбцы:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \quad A^T = \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{n1} \\ a_{12} & a_{22} & \cdots & a_{n2} \\ \cdots & \cdots & \cdots & \cdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{pmatrix}.$$

- Признаки положительной определенности матрицы

1. Критерий Сильвестра. Чтобы матрица A была положительно определенной, необходимо и достаточно, чтобы все ее главные миноры были положительны.

2. Достаточное условие. Диагональное преобладание, т.е. свойство $a_{ii} > \sum_{j=1, j\neq i}^{n} a_{ij}$ влечет положительную определенность матрицы.

- Нормой вектора X называется поставленное в соответствие этому вектору неотрицательное число $\|X\|$, удовлетворяющее аксиомам:

1. Положительная определенность, т.е. для любого ненулевого вектора его норма больше нуля и равна нулю только для ноль вектора $\|X\| > 0 \ \forall 0 \neq X \in R^n, \|0\| = 0$

2. Однородность $\|\alpha X\| = |\alpha| \|X\|, \ \forall \alpha = const, \ \forall X \in R^n$

3. $\|Y + X\| \leq \|X\| + \|Y\|, \ \forall Y, X \in R^n$

Существует несколько способов введения нормы вектора. Наиболее употребительными являются следующие:

1) первая (кубическая) $\|X\|_I = \max\limits_{1 \leq i \leq n} |x_i|$;

2) вторая (октаэдрическая) $\|X\|_{II} = \sum_{i=1}^{n} |x_i|$;

3) третья (сферическая) $\|X\|_{III} = \sqrt{X, X} = \sqrt{x_1^2 + ... + x_n^2} = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2}$.

- Нормой матрицы A называется поставленное этой матрице в соответствие неотрицательное число $\|A\|$ такое, что

1) $\|A\| > 0 \ \forall 0 \neq A \in H, \ \|0\| = 0$;

2) $\|\alpha A\| = |\alpha| \|A\| \ \forall \alpha = const, \ \forall A \in H$;

3) $\|A + B\| \leq \|A\| + \|B\|, \ \forall A, B \in H$;

4) $\|A \cdot B\| \leq \|A\| \cdot \|B\|, \ \forall A, B \in H$.

Здесь H – линейное пространство квадратных матриц n-го порядка.

Норма матрицы, как и норма вектора, может быть определена по-разному.

1) первая $\|A\|_I = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}|$;

2) вторая $\|A\|_{II} = \max_{1 \leq j \leq n} \sum_{i=1}^{n} |a_{ij}|$;

3) $\|A\|_{III} = \sqrt{\Lambda}$, где Λ – наибольшее собственное значение матрицы $A^T A$.

- Если для любой матрицы A и любого вектора X выполняется неравенство $\|A \cdot B\| \leq \|A\| \cdot \|B\|$, то говорят, что норма матрицы согласована с данной нормой вектора.

- Нормой матрицы A, подчиненной данной норме вектора, называется число $\|A\| = \sup \frac{\|AX\|}{\|X\|}$, $0 \neq X \in H$ – верхняя грань (т.е. максимальное число) множество норм такого вида.

- Собственным значением квадратной матрицы A n-го порядка

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

называется такое число λ, при котором для некоторого ненулевого вектора $X = x_1, x_2, \ldots, x_n$ имеет место равенство

$AX = \lambda X$.

Любой ненулевой вектор X, удовлетворяющий этому равенству, называется собственным вектором матрицы A, соответствующим собственному значению λ.

- Матрица $C = A - \lambda E = \begin{bmatrix} a_{11}-\lambda & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22}-\lambda & \ldots & a_{2n} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn}-\lambda \end{bmatrix}$ называется характеристической матрицей данной матрицы A.
- Уравнение $|A - \lambda E| = 0$ называется характеристическим уравнением матрицы A, а полином $|A - \lambda E|$ – характеристическим полиномом.
- Совокупность всех собственных значений $\lambda_1, \lambda_2, \ldots, \lambda_n$ матрицы A называется спектром этой матрицы.
- Спектральным радиусом $\rho(A)$ матрицы A называется максимум из модулей собственных значений этой матрицы.
- Собственные значения треугольной матрицы равны ее диагональным элементам.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix} \quad \lambda_1 = a_{11}; \quad \lambda_2 = a_{22}; \quad \ldots \quad \lambda_n = a_{nn}.$$

Приложение 2

Функции MathCad

Для контроля вычислений можно использовать стандартные функции пакета MathCad:

eigenvals (A) – возвращает вектор из собственных значений матрицы **A**.

eigenvec (A, λ) – нормированный собственный вектор матрицы *A*, соответсвующий ее собственному значению λ.

eigenvecs (A) – возвращает матрицу, чьими столбцами являются собственные векторы матрицы *A*. Порядок расположения собственных векторов соответсвует порядку собственных значений, возвращаемых функцией eigenvals.

max(A), min(A) – вычисление максимального и минимального элементов матрицы или вектора.

norm1(A) – возвращает первую норму матрицы или вектора.

norm2(A) – возвращает вторую норму матрицы или вектора.

norme(A) – возвращает третью норму матрицы или вектора.

Для решения системы уравнений необходимо выполнить следующее:

• Задать начальное приближение для всех неизвестных, входящих в систему уравнений, так как Mathcad решает систему с помощью итерационных методов.

• Оформить блок решения системы с помощью ключевого слова **Given**. Оно указывает Mathcad, что далее следует система уравнений.

• Записать систему уравнений. Для печати символа = необходимо использовать комбинацию клавиш **[Ctrl]**+[=] .

• Для решения системы вызвать функцию **Find** или **Minerr**.

Find(x1,x2,…,xn), Minerr(x1,x2,…,xn) – возвращает решение системы уравнений в виде вектора. В качестве аргументов указываются имена искомых неизвестных. Число аргументов должно быть равно числу неизвестных.

Функция **Minerr** использует тот же самый алгоритм, что и функция **Find**. Различие состоит в следующем: если в результате поиска решения не может быть получено дальнейшее уточнение текущего приближения к решению, **Minerr** возвращает это приближение, а функция **Find** возвращает в этом случае сообщение об ошибке "*решение не найдено*".

Л И Т Е Р А Т У Р А

1. Архангельский А.Я. C++ Builder 6: справоч. пособие. – М.: Бином-Пресс, 2002. − Кн. 1. Язык C++. – 564 с.
2. Архангельский А.Я. C++ Builder 6: справоч. пособие. – М.: Бином-Пресс, 2002. – Кн. 2. Классы и компоненты. – 528 с.
3. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. М.: Бином, 2004. – 636 с.
4. Бахвалов Н.С. Численные методы в задачах и упражнениях: учеб. пособие. − М.: Высшая школа (Высшая математика), 2000. – 190 с.
5. Икрамов Х.Д. Нессиметричная проблема собственных значений. Численные методы. − М.: Наука, 1991. – 240 с.
6. Исаков В.Н. Элементы численных методов. – М.: Изд. Центр «Академия», 2003.
7. Калиткин Н.Н. Численные методы: учеб. пособие. – М.: Наука, 1978. – 512 с.
8. Киреев В.И., Пантелеев А.В. Численные методы в примерах и задачах: учеб. пособие. – М.: «Высшая школа», 2006. – 480 с.
9. Костомаров Д.П., Фаворский А.П. Программирование и численные методы. – М.: МГУ, 2004.
10. Крылов В.И., Бобков В.В., Монастырный П.И. Вычислительные методы высшей математики: учеб. пособие. – Минск: Вышэйшая школа, 1972. – 585 с.
11. Маркова Л.В., Адаменко Н.Д., Казанцева О.Г., Корчевская Е.А. Формирование профессиональных компетенций у студентов специальности «Прикладная математика» // Вестн. Витебск. гос. ун-та. – 2012. – № 1(67). – С. 116–121.
12. Маркова Л.В., Корчевская Е.А. Обучение вычислительной математике. Современные аспекты // 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.
13. Маркова Л.В., Корчевская Е.А., Красоткина А.Н. Объектная реализация методов вычислительной алгебры // Вестн. Витебск. гос. ун-та. – 2013. – № 2(74). – С. 18–22.
14. Марчук П.И. Методы вычислительной математики. − М.: Наука, 1989. – 608 с.

15. Павловская Т.А. C/C++. Программирование на языке высокого уровня: учебник для вузов. − СПб.: Питер, 2003. – 461 с.
16. Рыжиков Ю.И. Вычислительные методы. – СПб.: БХВ-Петербург, 2007.
17. Ращиков В.И., Рошаль А.С. Численные методы решения физических задач. – СПб.: «Лань», 2005.
18. Самарский А.А. Введение в численные методы. – М.: Наука, 1983. – 272 с.
19. Самарский А.А., Гулин А.В. Численные методы. – М.: Наука, 1989. – 432 с.
20. Сборник задач по методам вычислений: учеб. пособие для студ. учреждений, обеспечивающих получение высш. образования по физико-математическим спец. / под ред. П.И. Монастырного. − Минск: Изд. центр БГУ, 2007. – 376 с.
21. Семенов В.А., Тарлапан О.А. Объектно-ориентированный подход к программированию прямых методов линейной алгебры // Вопросы кибернетики. Приложения системного программирования / под ред. В.П. Иванникова. – М.: Науч. совет по комплексной проблеме «Кибернетика» РАН, 1996. − Вып. 2. – С. 147−170.
22. Турчак Л.И. Основы численных методов: учеб. пособие для студ. вузов. − 2-е изд., перераб. и доп. – М.: Физматлит, 2002. – 300 с.
23. Устинов С.М., Зимницкий В.А. Вычислительная математика. – СПб.: БХВ-Петербург, 2009.
24. Формалев В.Ф., Ревезников Д.Л. Численные методы. – М.: ФИЗМАТЛИТ, 2004. – 400 с.

Учебное издание

МАРКОВА Людмила Васильевна, **КОРЧЕВСКАЯ** Елена Алексеевна
КРАСОТКИНА

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ АЛГЕБРЫ. ПРАКТИКУМ

Пособие

Технический редактор *Г.В. Разбоева*
Корректор *Л.В. Моложавая*
Компьютерный дизайн *И.В. Волкова*

Подписано в печать .2013. Формат 60x84 $^{1}/_{16}$. Бумага офсетная.
Усл. печ. л. 8,60. Уч.-изд. л. 3,60. Тираж экз. Заказ

Издатель и полиграфическое исполнение – учреждение образования
«Витебский государственный университет имени П.М. Машерова».
ЛИ № 02330/110 от 30.01.2013.

Отпечатано на ризографе учреждения образования
«Витебский государственный университет имени П.М. Машерова».
210038, г. Витебск, Московский проспект, 33.