МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ
имени М.В.Ломоносова
Факультет вычислительной математики и кибернетики

Практическое задание по курсу лекций «Численные методы линейной алгебры»
Задание №1
Отчет
о выполненном задании

студента 303 учебной группы факультета ВМК МГУ
Курбацкого Вячеслава Константиновича

Москва
2024

Содержание

1 QR разложение невырожденной матрицы

1.1 Постановка задачи

Требуется реализовать 2 метода получения QR разложения невырожденной матрицы A. Оба метода определены вариантом задания и были разобраны на лекциях. Также нужно сравнить точность и время работы каждого метода и решить СЛАУ с наилучшим из них, посчитав норму невязки решения и правой части.

1.2 Описание методов

Определение. *QR разложением* матрицы $A \in R^{n \times n}$ называется представление матрицы в виде:

$$A = QR$$

Где $Q \in R^{n \times n}$ - ортогональная матрица, а $R \in R^{n \times n}$ - верхняя треугольная матрица. Наложение дополнительного условия неотрицательности диагональных элементов матрицы R гарантирует единственность такого разложения. Так же на лекциях было доказано, что такое разложение существует для любой невырожденной матрицы A (В самом деле оно существует и для вырожденных матриц, но в данном курсе такой случай не рассматривается). Невырожденность матрицы A гарантирует невырожденность матрицы R, т.к. $detA = detQdetR \Rightarrow detR = detA/detQ$. Это означает, что на диагонали треугольной матрицы R нет нулевых элементов, т.к. $detR = r_{11}r_{22} \dots r_{nn} \neq 0$.
В данной работе рассмотрены 2 метода получения QR разложения: метод отражений (Хаусхолдера) и процесс ортогонализации Грамма-Шмидта.

1.2.1 Процесс ортогонализации Грамма-Шмидта.

Пусть имеется матрица $A \in R^{n \times n}$, $detA \neq 0$. При условии невырожденности матрицы, её столбцы будут образовывать базис пространства R^n (будут являться линейно независимой системой векторов), поэтому к столбцам можно применить процесс ортогонализации Грамма-Шмидта. Обозначим через A^j - j-столбец матрицы A, через Q^j - соответственно j-столбец матрицы Q. Тогда алгоритм будет выглядеть следующим образом:

$$\begin{cases} B^1 := A^1, \quad Q^1 = \frac{B^1}{\|B^1\|_2} \\ \\ B^k := A^k - \sum_{s=1}^{k-1}(A^k, Q^s)Q^s, \quad Q^k = \frac{B^k}{\|B^k\|_2}, \quad k = \overline{2, n} \end{cases}$$

Нетрудно показать, что полученная система векторов $\{Q^j\}_{j=1}^n$ будет ортонормированной, т.е. вектора удовлетворяют соотношению:

$$(Q^i, Q^j) = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

Таким образов, собрав полученные вектора в столбцы матрицы Q получится матрица: $QQ^T = Q^TQ = I$ - ортогональная матрица. Так же легко заметить, что Q^k выражается через первые k столбцов матрицы A, поэтому матрица коэффициентов $R = (r_{ij})$, где $r_{ij} = (Q^i, A^j)$ при $i \neq j$ и $r_{ii} = \|B^i\|_2 > 0$ будет верхней треугольной и удовлетворять соотношению $A = QR$ - искомое QR разложение.

На лекции было показано, что вычислительная сложность процесса ортогонализации составляет $n^3 + O(n^2)$. Данный алгоритм прост в реализации, но требует относительно много вычислений, по сравнению, например, с методом отражений.

1.2.2 Метод отражений (Хаусхолдера)

Определение. Матрицей отражений (матрицей Хаусхолдера) называется матрица вида

$$Z = I - 2ww^T,$$

где I - единичная матрица и $\|w\|_2 = 1$. Данная матрица обладает рядом полезных на практике свойств:

- $Z = Z^* = Z^{-1}$ (то есть матрица ортогональна)
- $detZ = -1$
- $\lambda_1 = 1$, $\lambda_i = -1$ при $i = \overline{2, n}$ - собственные значения матрицы.

Линейный оператор, матрица которого в единичном базисе $e_1, e_2 \dots e_n$ является матрицей отражений, называется оператором отражений. В силу инвариантности относительно подобия, выше указанные свойства сохраняются в любом базисе, а следовательно являются свойствами самого оператора отражений.

На лекции был доказан следующий факт:

Утверждение. Пусть x и y - ненулевые векторы, причем $\|x\|_2 = \|y\|_2$. Тогда если

$$w = \frac{x - y}{\|x - y\|_2} \text{ или } w = -\frac{x - y}{\|x - y\|_2},$$

то заданный этим вектором оператор отражения будет таким, что $y = Zx$.
На основе этого утверждения строится ещё один метод нахождения QR разложения матрицы - метод отражений (Хаусхолдера):

Пусть

$$A^{(0)} = \begin{bmatrix} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} & \dots & a_{1n}^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & a_{23}^{(0)} & \dots & a_{2n}^{(0)} \\ a_{31}^{(0)} & a_{32}^{(0)} & a_{33}^{(0)} & \dots & a_{3n}^{(0)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(0)} & a_{n2}^{(0)} & a_{n3}^{(0)} & \dots & a_{nn}^{(0)} \end{bmatrix}$$

и пусть $x^{(1)} = \begin{pmatrix} a_{11}^{(0)} & a_{21}^{(0)} & \dots & a_{n1}^{(0)} \end{pmatrix}^T$ - первый столбец матрицы $A^{(0)}$ и $y^{(1)} = \begin{pmatrix} \|x^{(1)}\|_2 & 0 & \dots & 0 \end{pmatrix}^T$. Эти векторы имеют одинаковую длину, поэтому согласно утверждению можно построить матрицу $Z^{(1)}$ такую, что: $Z^{(1)}x^{(1)} = y^{(1)}$. Тогда:

$$A^{(1)} = Z^{(1)}A^{(0)} = \begin{bmatrix} \|x^{(1)}\|_2 & a_{12}^{(1)} & a_{13}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \dots & a_{2n}^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & \dots & a_{3n}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(1)} & a_{n3}^{(1)} & \dots & a_{nn}^{(1)} \end{bmatrix}$$

В общем случае после исключения x_{k-1} из уравнений k, $k+1$, $\dots n$ имеем матрицу:

$$A^{(k-1)} = \begin{bmatrix} R^{(k-1)} & B \\ \Theta & \tilde{A}^{(k-1)} \end{bmatrix}$$

где $R^{(k-1)}$ - верхняя треугольная размера $(k-1) \times (k-1)$, Θ - нулевая и

$$\tilde{A}^{(k-1)} = \begin{bmatrix} a_{kk}^{(k-1)} & a_{kk+1}^{(k-1)} & \dots & a_{kn}^{(k-1)} \\ a_{k+1k}^{(k-1)} & a_{k+1k+1}^{(k-1)} & \dots & a_{k+1n}^{(k-1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{nk}^{(k-1)} & a_{nk+1}^{(k-1)} & \dots & a_{nn}^{(k-1)} \end{bmatrix}$$

Тогда возьмем $x^{(k)} = \begin{pmatrix} a_{kk}^{(k-1)} & a_{k+1k}^{(k-1)} & \dots & a_{nk}^{(k-1)} \end{pmatrix}^T \in R^{n-k+1}$ и $y^{(k)} = \begin{pmatrix} \|x^{(k)}\|_2 & 0 & \dots & 0 \end{pmatrix}^T \in R^{n-k+1}$. Возьмём $w^{(k)}$ как в утверждении и построим матрицу:

$$Z^{(k)} = \begin{bmatrix} I_{k-1} & \Theta \\ \Theta & I_{n-k+1} - 2w^{(k)}w^{(k)^T} \end{bmatrix}$$

В данных обозначениях матрица I_j - единичная матрица размера $j \times j$. Θ - нулевые блоки нужных размеров. В таком случае:

$$A^{(k)} = Z^{(k)}A^{(k-1)} = \begin{bmatrix} R^{(k-1)} & B \\ \Theta & \tilde{A}^{(k)} \end{bmatrix}$$

где

$$\tilde{A}^{(k)} = \begin{bmatrix} \|x^{(k)}\|_2 & a_{kk+1}^{(k)} & \dots & a_{kn}^{(k)} \\ 0 & a_{k+1k+1}^{(k)} & \dots & a_{k+1n}^{(k)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{nk+1}^{(k)} & \dots & a_{nn}^{(k)} \end{bmatrix}$$

Применяя эти рассуждения при всех $k = \overline{1, n-1}$ получаем:

$$R = Z^{(n-1)}Z^{(n-2)} \dots Z^{(1)}A = Q^T A \Leftrightarrow A = QR$$

Замечание. Если вектора x и y близки друг к другу, то есть величина $\|x - y\|_2 << 1$, то при вычислении вектора w происходит деление на очень маленькое число с плавающей точкой, что может приводить к большим ошибкам округления. В таком случае рекомендуется отражать $x \to -y$ с добавлением коэффициента -1 к матрице Z. Тогда $w = \frac{x+y}{\|x+y\|_2}$ и $Z = -(I - 2ww^T)$. Такое преобразование приведет к такому же результату, но к тому же помогает избегать деления на очень маленькое число с плавающей точкой. На практике удобно определять вид матрицы по знаку скалярного произведения: $(x, y) < 0 \Rightarrow$ угол между векторами тупой, поэтому допустимо брать $x - y$. В случае острого угла следует брать $x + y$. Именно такой подход и реализован в написанной программе.

На лекции было показано, что сложность вычислений данного метода составляет $\frac{2}{3}n^3 + O(n^2)$. Данный метод требует меньше вычислений, чем процесс ортогонализации Грамма-Шмидта, но он более труден в реализации, посколько необходимо реализовать уменьшение размерности векторов с каждым шагом (эта трудность относится именно к программной реализации, а не аналитическому нахождению разложения). Кроме того, нужно следить за делением на близкие к нулю числа с плавающей точкой - тоже дополнительное усложнее и уменьшение надежности.

1.3 Программная реализация

Заданием было предложено реализовать упомянутые методы на языке программирования С (С++). Для реализации были использованы контейнеры $std :: vector$ и написанные на их основе классы для матриц. Для уменьшения влияния чисто программных недочётов были использованы флаги компиляции $-O3 - march = native$ для максимальной оптимизации.

1.3.1 Время, потраченное на построение и погрешность вычеслений

Для усреднения полученных значений программа была запущена 1000 раз (каждый раз вектор x генерировался случайно, $x_k \in [-1, 1]$ - согласно условию задания). В качестве матричной нормы используется норма, подчиненная максимум-норме: $\|x\| = \max\limits_{i=1,2...n} |x_i|$. В следующей таблице представлены полученные значения:

Метод	Среднее время (мс)	$\|A - QR\|$
Хаусхолдер	0.681552	9.05193e-13
Грамм-Шмидт	0.872347	1.30826e-13

Хоть по теоретическим оценкам можно было ожидать, что метод Хаусхолдера в среднем будет в 1.5 раза быстрее, но программная реализация накладывает свои особенности, такие как работа с памятью и оптимизация вычислений, поэтому полученные на практике значения могут немного расходиться с теоретическими. Тем не менее даже на достаточно небольшой матрице размера 100 (матрица соответствует варианту задания) можно заметить, что метод Хаусхолдера работает быстрее процесса ортогонализации, но в то же время даёт точность почти на порядок хуже. Это можно объяснить как и особенностями конкретной реализации, так и тем, что метод Хаусхолдера, как известно, не самый надежный в плане вычислений.

2 Решение СЛАУ

В задании предложено решить СЛАУ с наиболее точным методом получения QR разложения. Но для начала стоит напомнить, как решать системы линейных алгебраических уравнений $Ax = f$ при помощи QR разложения матрицы A.

2.1 Обратный ход метода Гаусса

$$Ax = f \Leftrightarrow QRx = f \Leftrightarrow Rx = Q^T f = \tilde{f}$$

Таким образом, система сводится к системе с верхней треугольной матрицей R. Решение такой системы получается через обратный ход метода Гаусса:

$$\begin{cases} x_n = \tilde{f}_n / r_{nn} \\ \\ x_m = [\tilde{f}_m - \sum_{j=m+1}^{n} r_{mj} x_j] / r_{mm}. \quad m = \overline{n-1, 1} \end{cases}$$

Как было упомянуто в начале, в силу невырожденности A, все $r_{ii} \neq 0$ при $i = \overline{1, n}$, поэтому все арифметические операции корректны. Пользуясь этими соотношениями, решим систему $Ax = f$ пользуясь процессом ортогонализации Грамма-Шмидта.

2.2 Результаты вычислений. Невязка.

Как и в случае вычисления $\|A - QR\|$ здесь используется максимум норма.

Метод	$\|x - \tilde{x}\|$	$\|f - A\tilde{x}\|$	Время (мс)
Грамм-Шмидт	1.66533e-15	1.13687e-13	0.005775

Точность нахождения решения довольно неплохая. Однако при увеличении размера матрицы точность вычислений будет падать на порядки и станет хуже, чем у метода Хаусхолдера (эти данные не представлены в отчёте, т.к. это не просится в задании, при желании можно запустить программу на тестах с матрицами большей размерности).

3 Выводы.

Были реализованы и сравнены 2 метода построения QR разложения невырожденной матрицы. Для каждого решения были оценены нормы разности $A - QR$ и для наиболее точного метода была решена система $Ax = f$ и оценена норма невязок $x - \tilde{x}$ и $f - A\tilde{x}$, где $\tilde{x}$ - численное решение системы, полученное реализацией обратного хода метода Гаусса, а x - вектор, такой что $f = Ax$. В следующем разделе приведены инструкции для компиляции и запуска программы.

4 Инструкции по запуску программы.

- Запустить скрипт run.sh командой

  ```
  $ ./run.sh
  ```

  Данный скрипт создаст директорию build и внутри неё скомпилирует проект с помощью cmake и Makefile.

- находясь в директории build команда

  ```
  $ ./main 100 < ../tests/SLAU_var_6.txt
  ```

  запустит программу на матрице размера 100×100, заданной вариантом. Кроме того, предусмотрена возможность тестирования на матрицах размера 1024, 2048 и 4096:

  ```
  $ ./main 1024 < ../tests/in1024.txt
  $ ./main 2048 < ../tests/in2048.txt
  $ ./main 4096 < ../tests/in4096.txt
  ```

  Во всех случаях вывод будет примерно такой:

  ```
  Time (Householder): 0.654763ms
  ||A - QR|| = 8.95672e-13

  Time: (Gramm-Shmidt): 0.860597ms
  ||A - QR|| = 1.30826e-13

  Time: (Householder system): 0.021012ms
  ||x - x_f|| = 3.44169e-15
  ||f - Ax_f|| = 2.20268e-13

  Time: (Gramm-Shmidt system): 0.026521ms
  ||x - x_f|| = 1.88738e-15
  ||f - Ax_f|| = 1.35003e-13
  ```