Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана» (МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа №1 по предмету «Численные методы линейной алгебры»:

«Алгоритм «разделяй и властвуй» для трехдиагональной симметрической матрицы»

Студент ИУ9-72(Б) _____ М.М. Масягин

(Подпись, дата)

Преподаватель _____ А.Ю. Голубков

(Подпись, дата)

Москва, 2019 г.

Оглавление

Введение

Численные методы линейной алгебры (ЧМЛА) — это раздел современной науки, находящийся на пересечении вычислительной математики и линейной алгебры. Целью данной дисциплины является разработка и анализ алгоритмов для численного решения матричных задач.

ЧМЛА имеют большое прикладное значение. Например, на них основываются вся современная компьютерная графика и науки о машинном обучении и анализе данных.

Одной из наиболее важных задач ЧМЛА является задача нахождения собственных значений и собственных векторов матриц, как в общем виде, так и для специальных матриц.

В данной лабораторной работе рассматривается алгоритм поиска собственных векторов и собственных значений для трехдиагональной симметрической матрицы методом «разделяй и властвуй».

1 Постановка задачи

Пусть дана симметрическая трехдиагональная матрица:

$$T = \begin{pmatrix} a_{11} & b_{12} & & \\ a_{21} & a_{22} & \ddots & \\ & \ddots & \ddots & b_{n-1n} \\ & & b_{nn-1} & a_{nn} \end{pmatrix}$$

Необходимо представить ее в виде произведения трех матриц: $Q \cdot \Lambda \cdot Q^T$ (1), где Q — матрица, столбцы которой являются собственными векторами матрицы T, Λ - диагональная матрица с соответствующими собственными значениями на главной диагонали, Q^T - транспонированная матрица Q.

Данная задача может быть решена классическим QR-алгоритмом, главными преимуществами которого являются простота реализации и устойчивость (алгоритм считается вычислительно устойчивым, т.к. производится ортогональными преобразованиями подобия). Однако QR-алгоритм не подходит для обработки очень больших матриц в силу своей вычислительной сложности, поэтому в 1981 Дж. Куппен разработал новый алгоритм для решения данной задачи, основывающийся на методе «разделяй и властвуй». О нем и пойдет речь далее.

2 Описание алгоритма

Основная идея алгоритма

Основной идеей метода «разделяй и властвуй» является разбиение исходной задачи на более мелкие, простые подзадачи, которые в свою очередь также рекурсивно разбиваются на еще более простые подзадачи. Не является исключением и данный алгоритм. Его действие состоит из трех основных шагов:

1. Разбиваем исходную симметрическую трехдиагональную матрицу на две матрицы (также симметрические и трехдиагональные) меньшего размера;
2. Находим разложения (1) маленьких матриц;
3. На их основе строим разложение (1) исходной матрицы;

Разбиение трехдиагональной матрицы

Запишем исходную матрицу T в следующем виде:

$$T=\begin{pmatrix} a_{11} & b_{12} & & & & & & & \\ a_{21} & a_{22} & \ddots & & & & & & \\ & \ddots & \ddots & b_{m-1m} & & & & & \\ & & b_{mm-1} & a_{mm} & b_{mm+1} & & & & \\ & & & b_{m+1m} & a_{m+1m+1} & b_{m+1m+2} & & & \\ & & & & b_{m+2m+1} & a_{m+2m+2} & \ddots & & \\ & & & & & \ddots & \ddots & b_{n-1n} & \\ & & & & & & b_{nn-1} & a_{nn} \end{pmatrix}=$$

$$=\begin{pmatrix} a_{11} & b_{12} & & & & & & & \\ a_{21} & a_{22} & \ddots & & & & & & \\ & \ddots & \ddots & b_{m-1m} & & & & & \\ & & b_{mm-1} & a_{mm}\pm b_{mm+1} & b_{mm+1} & & & & \\ & & & b_{m+1m} & a_{m+1m+1}\pm b_{mm+1} & b_{m+1m+2} & & & \\ & & & & b_{m+2m+1} & a_{m+2m+2} & \ddots & & \\ & & & & & \ddots & \ddots & b_{n-1n} & \\ & & & & & & b_{nn-1} & a_{nn} \end{pmatrix}+\begin{pmatrix} \ddots & \ddots & & & \\ \ddots & \pm b_{mm+1} & b_{mm+1} & & \\ & b_{mm+1} & \pm b_{mm+1} & \ddots & \\ & & \ddots & \ddots & \end{pmatrix}=$$

$$= \begin{pmatrix} T_1 & \\ & T_2 \end{pmatrix} + \rho \cdot u \cdot u^T \quad \text{, где} \quad \rho = \pm b_m \quad \text{и} \quad u = \begin{pmatrix} \pm e_m \\ e_1 \end{pmatrix} \quad (2).$$

Разбиение можно выбрать по любому m, однако наиболее логично использовать $m = \frac{n}{2}$. В таком случае e_m - вектор размера $m = \frac{n}{2}$, а e_1 - это вектор размера $n - m$.

Решение для малых матриц

Пусть у нас имеются решения задачи $Q_i \cdot \Lambda_i \cdot Q_i^T$ (3) для двух малых матриц $i = 1,2$. Вычислим на их основе разложение для исходной матрицы Q .

Подставим (3) в (2) и получим:

$$\begin{pmatrix} Q_1^T & \\ & Q_2^T \end{pmatrix} \cdot \left(\begin{pmatrix} T_1 & \\ & T_2 \end{pmatrix} + \rho \cdot u \cdot u^T \right) \cdot \begin{pmatrix} Q_1 & \\ & Q_2 \end{pmatrix} = \begin{pmatrix} \Lambda_1 & \\ & \Lambda_2 \end{pmatrix} + \rho \cdot v \cdot v^T \quad \text{, где} \quad v = \begin{pmatrix} Q_1^T & \\ & Q_2^T \end{pmatrix} \cdot u = \begin{pmatrix} \pm lr(Q_1) \\ \pm fr(Q_2) \end{pmatrix}$$

$lr(Q_1)$ - последняя строка Q_1 , $fr(Q_2)$ - первая строка Q_2 .

Наконец, мы подошли к задаче поиска собственных значений:

$(D + \rho \cdot v \cdot v^T) \cdot X = \Lambda \cdot X$, где $D = \Lambda_1 \otimes \Lambda_2 = diag(\lambda_1, \ldots, \lambda_n)$.

Пусть $D + \rho \cdot v \cdot v^T = Q \cdot \Lambda \cdot Q^T$ - спектральное разложение, тогда:

$$T = \begin{pmatrix} Q_1 & \\ & Q_2 \end{pmatrix} \cdot Q \cdot \Lambda \cdot Q^T \cdot \begin{pmatrix} Q_1^T & \\ & Q_2^T \end{pmatrix} .$$

Дефляция

Некоторые собственные значения можно вычислить путем «дефляции».

Так, если в векторе v имеются нулевые элементы (элементы, которые можно считать нулями с некоторым приближением), то для них мы имеем:

$$(v_i = 0 \rightarrow v^T \cdot e_i = 0) \rightarrow (D + \rho \cdot v \cdot v^T) \cdot e_i = d_i \cdot e_i .$$

Таким образом, мы можем легко отыскать собственные вектора и собственные значения для всех нулей.

Если на диагонали матрицы D встречаются одинаковые элементы (элементы, которые можно считать одинаковыми с некоторым приближением), например, $d_i = d_j$, $i < j$, то мы можем найти вращение $G(i, j, \varphi)$, переводящее 0 в j-ую позицию v .

$$G^T \mathbf{v} = G(i,j,\varphi)^T \mathbf{v} = \begin{bmatrix} \times \\ \vdots \\ \sqrt{{v_i}^2 + {v_j}^2} \\ \vdots \\ 0 \\ \vdots \\ \times \end{bmatrix} \begin{matrix} \\ \\ \leftarrow\ i \\ \\ \leftarrow\ j \\ \\ \\ \end{matrix}$$

Заметим, что для $\forall \varphi$ истинно: $G(i, j, \varphi)^T \cdot D \cdot G(i, j, \varphi) = D$, $d_i = d_j$.

Таким образом, если в матрице D есть n одинаковых значений (или значений, которых мы можем считать одинаковыми), мы можем избавиться от $n-1$ значения.

В результате использования дефляции мы сильно упрощаем исходную задачу (дефляция «срабатывает» достаточно часто) поиска собственных значений матрицы $(D + \rho \cdot v \cdot v^T)$. Она сводится к задаче поиска собственных значений матрицы:

$\begin{pmatrix} D_1 + \rho \cdot v_1 \cdot v_1^T & 0 \\ 0 & D_2 \end{pmatrix} = G^T \cdot (D + \rho \cdot v \cdot v \cdot T) \cdot G + E$, где у D_1 нет совпадающих значений на главной диагонали, а v_1 не содержит нулей. Матрица G - это вращений.

Поиск собственных значений

Очевидно, что в $(D+\rho\cdot v\cdot v^T)$ $\rho\neq 0$. Также после дефляции мы знаем, что вектор v не содержит нулей, а все элементы D различны.

Мы можем переупорядочить диагональные элементы матрицы $(D+\rho\cdot v\cdot v^T)$ так, чтобы выполнялось неравенство: $d_1<d_2<...<d_{n-1}<d_n$.

Заметим, что перестановки элементов D также переставляют и элементы вектора v .

Пусть (λ,x) - собственное значение и собственный вектор матрицы $(D+\rho\cdot v\cdot v^T)$. Тогда $(D-I\cdot\lambda)\cdot x=-\rho\cdot v\cdot v^T\cdot x$, $x=\rho\cdot(\lambda I-D)^{-1}\cdot v\cdot(v^T\cdot x)$ (4).

Данное равенство показывает, что x пропорционален $(\lambda\cdot I-D)^{-1}$. Мы можем потребовать, чтобы все x были нормированы, то есть $\|x\|=1$. В таком случае $x=\frac{(\lambda\cdot I-D)^{-1}}{\left\|(\lambda\cdot I-D)^{-1}\right\|}$ (5). Домножив (4) слева на v^T получим:

$$v^T\cdot x=v^T\cdot\rho\cdot(\lambda I-D)^{-1}\cdot v\cdot(v^T\cdot x) \quad .$$

С учетом того, что $v^T\cdot x\neq 0$ λ является собственным значением тогда и только тогда, когда:

$$f(\lambda)=1-\rho\cdot v^T\cdot(\lambda\cdot I-D)^{-1}\cdot v=1-\rho\cdot\sum_{k=1}^{n}\frac{v_k^2}{\lambda-d_k} \quad .$$

Данное равенство называется «вековым уравнением».

Его корни лежат между диагональными элементами матрицы D , причем если $\rho>0$, то $d_1<\lambda_1<d_2<\lambda_2<...<d_n<\lambda_n$, а если $\rho<0$, то $\lambda_1<d_1<\lambda_2<d_2<...<\lambda_n<d_n$.

Таким образом, необходимо вычислить $n-1$ значение λ_i из интервалов (d_i,d_{i+1}) и одно значение из интервала $(d_n,+\infty)$ или $(-\infty,d_1)$.

Соответствующие им собственные вектора могут быть вычислены по формуле (5).

v_k Вычисляются из соотношения:

$$v_k = \sqrt{\frac{\prod_{j=1}^{k-1}(d_k - \lambda_j)\prod_{j=k}^{n}(\lambda_j - d_k)}{\rho\prod_{j=1}^{k-1}(d_k - d_j)\prod_{j=k+1}^{n}(d_j - d_k)}}.$$

Решение векового уравнения

Само вековое уравнение решается посредством итерационного процесса.

Перепишем вековое уравнение в виде:

$$f(\lambda) = 1 + \underbrace{\sum_{k=1}^{i}\frac{v_k^2}{d_k - \lambda}}_{\psi_1(\lambda)} + \underbrace{\sum_{k=i+1}^{n}\frac{v_k^2}{d_k - \lambda}}_{\psi_2(\lambda)} = 1 + \psi_1(\lambda) + \psi_2(\lambda).$$

где $\psi_1(\lambda)$ - сумма положительных слагаемых, а $\psi_2(\lambda)$ - отрицательных.

$$h_1(\lambda) := \hat{c}_1 + \frac{c_1}{d_i - \lambda} \qquad h_1(\lambda_j) = \psi_1(\lambda_j) \qquad h_1'(\lambda_j) = \psi_1'(\lambda_j).$$

$$c_1 = \psi_1'(\lambda_j)(d_i - \lambda_j)^2 > 0,$$

$$\hat{c}_1 = \psi_1(\lambda_j) - \psi_1'(\lambda_j)(d_i - \lambda_j) = \sum_{k=1}^{i} v_k^2 \frac{d_k - d_i}{(d_k - \lambda_j)^2} \le 0.$$

$$h_2(\lambda) := \hat{c}_2 + \frac{c_2}{d_{i+1} - \lambda} \qquad h_2(\lambda_j) = \psi_2(\lambda_j) \quad h_2'(\lambda_j) = \psi_2'(\lambda_j)$$

$$c_2 = \psi_2'(\lambda_j)(d_{i+1} - \lambda_j)^2 > 0,$$

$$\hat{c}_2 = \psi_2(\lambda_j) - \psi_2'(\lambda_j)(d_{i+1} - \lambda_j) = \sum_{k=i+1}^{n} v_k^2 \frac{d_k - d_{i+1}}{(d_k - \lambda)^2} \ge 0.$$

$$h(\lambda) = 1 + h_1(\lambda) + h_2(\lambda) = \underbrace{(1 + \hat{c}_1 + \hat{c}_2)}_{c_3} + \frac{c_1}{d_i - \lambda} + \frac{c_2}{d_{i+1} - \lambda}.$$

Возьмем произвольное начальное λ_1 из нужного интервала и подставим его в данные формулы — выразим λ_2 и т.д. Обычно достаточно 2-3 шагов, чтобы алгоритм сошелся.

Итоговый алгоритм

1. На вход подается трехдиагональная симметрическая матрица T с вещественными элементами;
2. Если матрица T имеет вид (a), то вернем: $(\Lambda = T, Q = 1)$;
3. Иначе разобьем матрицу T на две подматрицы: $T1$ и $T2$.
    1. Запустим алгоритм со входной матрицей T_1 и выходными матрицами Q_1, Λ_1;
    2. Запустим алгоритм со входной матрицей T_2 и выходными матрицами Q_2, Λ_2;
4. На основе матриц $Q_1, \Lambda_1, Q_2, \Lambda_2$ получим матрицы Q, Λ для матрицы T. Вернем их. Работа алгоритма завершена.

3 Реализация

Алгоритм написан в двух вариантах:

1. В однопоточном виде на чистом C99 без каких-либо зависимостей, то есть все «классы» матриц, векторов и т.д. реализованы собственноручно. В программе используются 16-байтные числа с плавающей точкой («long long double»), при этом если перевести программу на 8-байтные числа с плавающей точкой («double»), то ее можно скомпилировать и на C89-совместимом компиляторе. Таким образом обеспечивается легкая переносимость алгоритма практически на все существующие платформы;
2. В однопоточном виде на GNU Octave (данный вариант взят из дополнений к лекциям P. Arbenz).

4 Тестирование

Алгоритм тестировался на множестве матриц размера от 4x4 до 10x10 с фиксированным параметром точности: 0.0000001 и максимальным разрешенным числом итераций: 10.

Собственноручно написанный алгоритм показал невысокую производительность в связи с неэффективным методом вычисления обратной матрицы. Однако алгоритм на GNU Octave обсчитывает матрицу 10x10 менее чем за 0.5 с, что является удовлетворительным результатом.

Пример матрицы, используемой при тестировани:

$$\begin{pmatrix} 1488 & 322 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 322 & 228 & 48 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 48 & 282 & 30 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & -1001 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 1.25 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 7 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 & 5 & 22 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 22 & 11 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 1 & 55 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 55 & 5 \end{pmatrix}$$

5 Возможные оптимизации

Реализованную программу можно оптимизировать как в алгоритмическом, так и в техническом плане. Кратко опишем возможные улучшения.

Как показывает практика, алгоритм Куппена имеет смысл использовать лишь для матриц достаточно большого порядка (25+). Для матриц меньшего порядка более эффективным является классическое QR-разложение. Таким образом, вызов основной рекурсивной функции алгоритма можно ограничить порядком матрицы не 1, а 25, для которого в свою очередь вызывать подпрограмму QR-разложения. Подобная оптимизация используется в библиотеке вычислительных алгоритмов LAPACK.

Серьезным преимуществом всех задач, решаемых методом «разделяй и властвуй», являтеся простота и очевидность их переноса на параллельные/распределенные платформы. Исключением не является и алгоритм Куппена. Для его распараллеливания достаточно инициализировать пул потоков и передавать каждому потоку пула свою подматрицу. Это может дать практически линейное увеличение производительности с увеличением числа потоков в пуле.

Также для повышения точности алгоритма можно воспользоваться его модификаций Гу-Эйзенштадта, незначительно увеличивающей вычислительную сложность.

Заключение

Таким образом, в ходе лабораторной работы был изучен способ вычисления собственных векторов и собственных значений симметрических трехдиагональных матриц методом «разделяй и властвуй», приведена его однопоточная реализация на языке Си, предложены различные варианты оптимизации алгоритма и написанной программы.

Список литературы

1. Arbenz P. Lecture Notes on Solving Large Scale Eigenvalue Problems // 2018
2. Arbenz P. Numerical Methods for Solving Large Scale Eigenvalue Problems // URL: http://people.inf.ethz.ch/arbenz/ewp/index.html (дата обращения: 24.12.2019)
3. Demmel J. Applied Numerical Linear Algebra // 1996
4. Rutter J. Serial Implementation of Cuppen's Divide and Conquer Algorithm for the Symmetric Eigenvalue Problem // 1994
5. Голубков. А. Ю. Лекции по вычислительным методам линейной алгебры // 2013 — 2018;
6. AlgoWiki: Метод «разделяй и властвуй» вычисления собственных значений и векторов симметричной трехдиагональной матрицы // URL: https://algowiki-project.org (дата обращения: 24.12.2019)