Санкт-Петербургский Национальный Исследовательский
Университет Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий

Лабораторная работа № 4

Выполнили:
Азатжонова М. А.
Маркозубова А. К.

Санкт-Петербург,
2024

СОДЕРЖАНИЕ

Стр.

Задание

Исходные данные в задании: матрица $A = A_0 + m \cdot E$, (E - единичная матрица), A_0 - симметричная матрица со случайными элементами от -1.0 до 1.0, вектор b - случайный вектор (нужно воспользоваться генератором)

1. Для матрицы A решите частичную проблему собственных значений при помощи степенного метода, критерий остановки итерационного процесса можно принять самостоятельно (например, рассмотрев невязку, либо расстояние от приближения до приближения – на выбор)

2. Воспользуйтесь результатами пункта 1 и решите систему линейных алгебраических уравнений $Ax = b$ итерационным путем (параметр τ теперь легко подобрать, поскольку известно максимальное собственное число матрицы A). Критерий остановки итерационного процесса: верхняя оценка ошибки приближенного решения по теореме об апостериорной оценке ошибки, $\epsilon \leqslant 0,001$

3. Решите ту же систему линейных алгебраических уравнений при помощи метода Якоби либо при помощи метода Гаусса Зейделя.

Ход решения

Генерация исходных данных

Для выполнения задания необходимо задать следующие данные:

- Симметричную случайную матрицу A_0, элементы которой находятся в диапазоне $[-1.0, 1.0]$.
- Матрицу A, которая определяется как:

$$A = A_0 + m \cdot E,$$

  где E — единичная матрица, а m — масштабный множитель. Добавление $m \cdot E$ гарантирует, что матрица A положительно определённая, а следовательно, все её собственные значения положительны.
- Случайный вектор b с элементами из диапазона $[-1.0, 1.0]$.

Симметрия матрицы A_0 достигается усреднением:

$$A_0 = \frac{A_0 + A_0^T}{2}.$$

Частичная проблема собственных значений (степенной метод)

Первый этап работы был посвящен нахождению максимального собственного числа матрицы с использованием степенного метода. Это итерационный метод, позволяющий эффективно находить собственное значение матрицы, которое является её наибольшим по модулю собственным числом. Задача решалась следующим образом.

На начальном этапе задавалось начальное приближение x, которое представляло собой нормированный вектор, все элементы которого равны 1: $x = \frac{Vector(n,1)}{VectorNorm(Vector(m,1),2)}$.

На каждом шаге алгоритма вычислялся новый вектор путем умножения матрицы на текущий вектор. Затем полученный вектор нормировался, что-

бы предотвратить его чрезмерный рост, и результат сохранялся как новое приближение: $x_{k+1} = \frac{A \cdot x_k}{||A \cdot x_k||}$

Для оценки текущего максимального собственного числа использовалась формула: $\lambda = (Ax_k, x_k)$.

Критерием завершения итерационного процесса служила норма невязки: процесс завершался, когда $||Ax - \lambda x||$ становилось меньше заданного порога.

Итоговое значение максимального собственного числа оказалось равным 6.458958762.

Используя обратную матрицу, мы вычислили минимальное собственное число. Для этого аналогичным образом было вычислено максимальное собственное число для обратной матрицы, найденное значение - обратно для искомого минимального собственного числа: $\mu = 2.876059799$.

Метод простых итераций

На втором этапе решалась система методом простых итераций. Вектор был сгенерирован случайным образом с элементами в диапазоне от -10 до 10. Зная максимальное и минимальное собственные числа матрицы M и μ, мы определили параметр τ: $\tau = \frac{2}{M+\mu}$.

Этот параметр использовался для формирования матрицы S и вектора c: $S = E - \tau \cdot A$, $c = \tau \cdot b$, которые необходимы для итерационного процесса $x_{k+1} = Sx_k + c$.

Сходимость контролировалась через верхнюю оценку ошибки: $err = \frac{q}{1-q}||x_{k+1}|x_k|$, где q - норма матрицы S . Процесс завершился, когда ошибка достигла значения менее 0.001.

Итоговое решение СЛАУ выглядело следующим образом:

$$\begin{bmatrix} -1.83798020719786858 \\ 0.427001709497668692 \\ -0.0727610256719473104 \\ 2.08906088488598884 \\ -2.22269133567310817 \end{bmatrix}$$

Рисунок 1 — Решение СЛАУ методом простых итераций

Метод Якоби

На последнем этапе мы решили ту же систему методом Якоби. Матрица была разложена на сумму диагональной части D, нижней треугольной U и верхней треугольной L.

В методе Якоби новое приближение вычисляется на основе предыдущего приближения и разложения :

$$x_{k+1} = D^{-1} \cdot (b - (L + U) \cdot x_k).$$

Для удобства расчетов выражение переписывается в виде:

$S = -D^{-1} \cdot (L + U), c = D^{-1} \cdot b$. Тогда $x_{k+1} = Sx_k + c$.

Сходимость контролировалась через верхнюю оценку ошибки, аналогично предыдущему методу. Решение, полученное методом Якоби, совпало с результатами, найденными ранее:

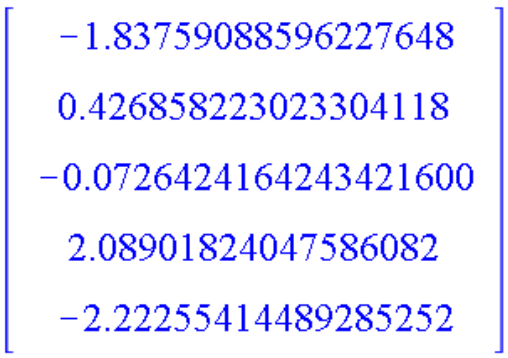

Рисунок 2 — Решение СЛАУ методом Якоби

Выводы

В ходе лабораторной работы мы исследовали степенной метод для нахождения максимального собственного числа, а также методы простых итераций и Якоби для решения систем линейных уравнений. Знание спектральных характеристик матрицы позволило оптимально выбрать параметр , что ускорило сходимость итерационного процесса.

Методы простых итераций и Якоби продемонстрировали схожую точность, успешно решив систему с заданной точностью . Эти методы применимы для численного решения систем линейных уравнений, особенно в случаях, когда матрица обладает хорошими спектральными свойствами.

Приложение

Ниже представлен полный код программы, написанный в программе Maple 13.0, который был написан для выполнения данной лабораторной работы.

```
> with(LinearAlgebra):
> with(plots):
> with(ArrayTools):
> m := 5:
> A0 := RandomMatrix(m, m, generator = -1.0 .. 1.0):
> A0 := (A0 + Transpose(A0))/2:
> A := A0 + m*IdentityMatrix(m)
```

$$A := \begin{bmatrix} 5.63460644130686639 & 0.217588734486948710 & -0.842987756896709572 & 0.710049673678763149 & -0.727698500319634678 \\ 0.217588734486948710 & 4.15635105750636758 & 0.909472078128555016 & 0.227556938400064856 & 0.354350112995760135 \\ -0.842987756896709572 & 0.909472078128555016 & 4.45795393743363810 & -0.262099523526285183 & -0.238243359452697368 \\ 0.710049673678763149 & 0.227556938400064856 & -0.262099523526285183 & 4.52594256908028836 & 0.396266482085543492 \\ -0.727698500319634678 & 0.354350112995760135 & -0.238243359452697368 & 0.396266482085543492 & 4.32436461638648595 \end{bmatrix} \quad (1)$$

```
>
>
> x := Vector(n, 1)/VectorNorm(Vector(m, 1), 2):
>
> k := 10^-6:
>
> r := 10^6:
> while r > k do
    y := A • x:
    x := y / VectorNorm(y, 2):
    lambda := A • x • x:
    r := VectorNorm(A • x - lambda • x, 2):
  end do:
>
> M := lambda
```

$$M := 6.458958762 \quad (2)$$

```
> Ainv := MatrixInverse(A):
> xinv := Vector(n, 1)/VectorNorm(Vector(m, 1), 2):
> rinv := 10^6:
> while rinv > k do
    y := Ainv • xinv:
    xinv := y / VectorNorm(y, 2):
    linv := Ainv • xinv • xinv:
    rinv := VectorNorm(Ainv • xinv - linv • xinv, 2):
  end do:
```

```
> μ := 1/linv
```

$$\mu := 2.876059799 \tag{3}$$

```
> b := RandomVector(5, generator = -10.0 .. 10.0) :
```

```
> τ := 2/(μ + M)
```

$$\tau := 0.2142470298 \tag{4}$$

```
> S := IdentityMatrix(m) - τ·A :
> q := MatrixNorm(S, 2) :
> c := τ • b
```

$$c := \begin{bmatrix} -1.52134759850616419 \\ 0.213648002973252621 \\ 0.341529422090160129 \\ 1.58239517230348036 \\ -1.55942460695003748 \end{bmatrix} \tag{5}$$

```
> x := Vector(m, 1) :
>
> q := MatrixNorm(S, 2) :
>
> err := 100 :
> while err > 0.001 do
    x_prev := x :
    x := S • x + c :
    err := q/(1 - q)·VectorNorm(x - x_prev, 2) :
    end do:
>
>
>
>
> x
```

$$\begin{bmatrix} -1.83798020719786858 \\ 0.427001709497668692 \\ -0.0727610256719473104 \\ 2.08906088488598884 \\ -2.22269133567310817 \end{bmatrix} \tag{6}$$

```
>
>
```

```
>
>
>
> L := LowerTriangle(A, -1)
```

$$L := \begin{bmatrix} 0., & 0., & 0., & 0., & 0. \\ 0.217588734486948710, & 0., & 0., & 0., & 0. \\ -0.842987756896709572, & 0.909472078128555016, & 0., & 0., & 0. \\ 0.710049673678763149, & 0.227556938400064856, & -0.262099523526285183, & 0., & 0. \\ -0.727698500319634678, & 0.354350112995760135, & -0.238243359452697368, & 0.396266482085543492, & 0. \end{bmatrix} \quad (7)$$

```
> d := DiagonalMatrix(Diagonal(A))
```

$$d := \begin{bmatrix} 5.63460644130686639, & 0, & 0, & 0, & 0 \\ 0, & 4.15635105750636758, & 0, & 0, & 0 \\ 0, & 0, & 4.45795393743363810, & 0, & 0 \\ 0, & 0, & 0, & 4.52594256908028836, & 0 \\ 0, & 0, & 0, & 0, & 4.32436461638648595 \end{bmatrix} \quad (8)$$

```
> U := UpperTriangle(A, 1)
```

$$U := \begin{bmatrix} 0., & 0.217588734486948710, & -0.842987756896709572, & 0.710049673678763149, & -0.727698500319634678 \\ 0., & 0., & 0.909472078128555016, & 0.227556938400064856, & 0.354350112995760135 \\ 0., & 0., & 0., & -0.262099523526285183, & -0.238243359452697368 \\ 0., & 0., & 0., & 0., & 0.396266482085543492 \\ 0., & 0., & 0., & 0., & 0. \end{bmatrix} \quad (9)$$

```
> S := -d^(-1).(L + U)
```

$$S := \begin{bmatrix} -0., & -0.038616491986347513 4, & 0.149608986124892624, & -0.126015841758431168, & 0.129148061696897380 \\ -0.052350903828006 4385, & -0., & -0.218815029227631992, & -0.054749210365447 1512, & -0.085255097101531 8960 \\ 0.189097457875933572, & -0.204011098116487322, & -0., & 0.058793681407388 2140, & 0.053442310709439 4237 \\ -0.156884375539712490, & -0.050278353056147 2514, & 0.057910483733678 0684, & -0., & -0.087554465404113 2270 \\ 0.168278710255406778, & -0.081942700125934 6312, & 0.055093263539783 9432, & -0.091635770162385 3050, & -0. \end{bmatrix} \quad (10)$$

```
> c := d^(-1).b
```

$$c := \begin{bmatrix} -1.26023070279927296 \\ 0.239922957163517164 \\ 0.357583718827990392 \\ 1.63189082496525284 \\ -1.68316723299546234 \end{bmatrix} \quad (11)$$

```
> x := Vector(m, 0) :
> err := VectorNorm(A • x − b, infinity)
```

$$err := 7.38584415280178774 \tag{12}$$

```
> q := MatrixNorm(S, infinity) :
>
> while err > 0.001 do
  x_prev := x :
  x := S • x + c :
  err := q/(1 − q) · VectorNorm(x − x_prev, infinity) :
  end do:
>
>
>
>
>
> x
```

$$\begin{bmatrix} -1.83759088596227648 \\ 0.426858223023304118 \\ -0.0726424164243421600 \\ 2.08901824047586082 \\ -2.22255414489285252 \end{bmatrix} \tag{13}$$

```
>
>
>
>
>
```