Название статьи

Автор: Имя Фамилия

Дата: 1 декабря 2024 г.

Содержание

Введение

В данной работе рассматривается метод Ричардсона - итерационный процесс решения систем линейных алгебраических уравнений (СЛАУ) вида $Ax = b$, где A - квадратная матрица, x - искомый вектор, а b - вектор правой части. Также представлена программная реализация этого процесса с тестированием на предложенных данных и анализ полученных результатов.

К целям данной работы относится не только теоретическое ознакомление с методом Ричардсона, но и программная релизация данного метода.

1 Постановка задачи

- Изучить теорию по методу Ричадсона.
- Реализовать метод программно на ЯП Python.
- Применить программу для тестовых данных и проанализировать точность.

1.1 Необходимые условия

1. **Симметричность матрицы**. Хотя метод Ричардсона может работать для несимметричных матриц, для теоретической гарантии сходимости проще рассматривать случай, когда A — симметричная матрица.
2. **Положительная определённость**. Это гарантирует, что уравнение $Ax = b$ имеет единственное решение, а спектр собственных значений A будет положительным ($\forall i : \lambda_i > 0$).

1.2 Тестовые данные

$$A_0 = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2.5 & 1 \\ 1 & 1 & 3 \end{pmatrix};$$

$$A_1 = \begin{pmatrix} -0.168700 & 0.353699 & 0.008540 & 0.733624 \\ 0.353699 & 0.056519 & -0.723182 & -0.076440 \\ 0.008540 & -0.723182 & 0.015938 & 0.342333 \\ 0.733624 & -0.076440 & 0.342333 & -0.045744 \end{pmatrix};$$

$$A_2 = \begin{pmatrix} 1.00 & 0.42 & 0.54 & 0.66 \\ 0.42 & 1.00 & 0.32 & 0.44 \\ 0.54 & 0.32 & 1.00 & 0.22 \\ 0.66 & 0.44 & 0.22 & 1.00 \end{pmatrix};$$

$$A_3 = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$

В качестве вектора правой части будем использовать вектор, состоящий из единиц; длина которого равно количеству строк/столбцов рассматриваемой матрицы.

2 Метод Ричардсона

Метод Ричардсона[1] — это метод, используемый для повышения точности численных вычислений, особенно при оценке пределов или приближённых решений. Метод часто применяется в задачах, связанных с решением дифференциальных уравнений, вычислением интегралов, интерполяцией или решением систем линейных уравнений.

Итерационная формула метода:

$$\frac{x^{(k+1)} - x^{(k)}}{\tau_{k+1}} + Ax^{(k)} = b.$$

Отсюда получается следующий итерационный процесс:

$$x^{(k+1)} = \left(b - Ax^{(k)}\right) \tau_{k+1} + x^{(k)}.$$

Чебышовский набор итерационных парметров:

$$\tau_k = \frac{\tau_0}{1 + \rho_0 v_k} \text{ - параметр релаксации на следующей итерации,}$$

$$\tau_0 = \frac{2}{\lambda_{\min}(A) + \lambda_{\max}(A)} \text{ - оптимальный базовый параметр релаксации,}$$

$$\rho_0 = \frac{1 - \eta}{1 + \eta},$$

$$\eta = \frac{\lambda_{\min}(A)}{\lambda_{\max}(A)},$$

$$v_k = \cos\left(\frac{(2k-1)\pi}{2n}\right) \text{ - представление многочлена Чебышева.}$$

3 Программаная реализация

```
import numpy as np
import math
from calculus.third_course.support.Rotation import RotationWithBarriers
from calculus.third_course.support.checker import check_answer


class Richardson:
 def __init__(self, A: np.array, b: np.array, x: np.array, p: int = 4):
  self._tol = math.pow(10, - (p-1))
  self._eta = None
  self._tau_0 = None
  self._rho_0 = None
  self._A = A
  self._b = b
  self._lambda_min = None
  self._lambda_max = None
  self._n = 8
  self._x = x
  self._p = p

 def _compute_SZ(self):
  rotator = RotationWithBarriers(self._A, self._p)
  sz = rotator.compute()
  self._lambda_max = sz[-1]
  self._lambda_min = sz[0]

 def _compute_tau_0(self):
  self._tau_0 = 2 / (self._lambda_min + self._lambda_max)

 def _compute_eta(self):
  self._eta = self._lambda_min / self._lambda_max

 def _compute_rho_0(self):
  self._rho_0 = (1 - self._eta) / (1 + self._eta)

 def _v_k(self, k):
  return np.cos((2 * k - 1) * np.pi / (2 * self._n))
```

```
 def _t_k(self, k):
  return self._tau_0 / (1 + self._rho_0 * self._v_k(k))

 def _compute_x(self, x: np.array):
  x_k = x

  for k in range(1, self._n + 1):
   x_k = (self._b - self._A @ x_k) * self._t_k(k) + x_k

 return x_k

 def compute(self):
  self._compute_SZ()
  self._compute_eta()
  self._compute_rho_0()
  self._compute_tau_0()
  x = self._x
  iterations = 0

  while np.linalg.norm(self._A @ x - self._b) > self._tol:
   x = self._compute_x(x)
   iterations += 1

  return x, iterations * self._n

def main():
 A = np.array([[2, 1],
    [1, 2]])
 b = np.array([4,5])
 x = np.zeros_like(b)
 richardson = Richardson(A, b, x)
 x, iterations = richardson.compute()

 print(f"Iteration proccess complete in {iterations} iterations)
 check_answer(A, b, x)
```

4 Анализ результатов решения

Рассмотрим результаты для матрицы A_0:

$$\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2.5 & 1 \\ 1 & 1 & 3 \end{pmatrix} x = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

При заданой точности $tol = 10^{-4}$ и $n = 8$, был получен результат за 16 итерации:

$$x = \begin{pmatrix} 1.18509336 \\ 1.7598606 \\ 4.55504604 \end{pmatrix}$$

Невязка результата:

$$||Ax - b||_2 = 1.04e - 07$$

Результат удовлетворяет заданной точности.

Рассмотрим результаты для матрицы A_1:

$$\begin{pmatrix} -0.168700 & 0.353699 & 0.008540 & 0.733624 \\ 0.353699 & 0.056519 & -0.723182 & -0.076440 \\ 0.008540 & -0.723182 & 0.015938 & 0.342333 \\ 0.733624 & -0.076440 & 0.342333 & -0.045744 \end{pmatrix} x = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

Для данной матрицы метод Ричардсона расходится, т.к. матрица A_1 обладает следующими собственными числами:

$$\lambda(A) = \{-0.94356786, -0.74403641, 0.68784308, 0.85777418\}.$$

По условиям, накладываемым на матрицу A, метод не гарантирует сходимость из-за наличия отрицательных собственных значений.

Рассмотрим результаты для матрицы A_2:

$$\begin{pmatrix} 1.00 & 0.42 & 0.54 & 0.66 \\ 0.42 & 1.00 & 0.32 & 0.44 \\ 0.54 & 0.32 & 1.00 & 0.22 \\ 0.66 & 0.44 & 0.22 & 1.00 \end{pmatrix} x = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

При заданой точности $tol = 10^{-4}$ и $n = 8$, был получен результат за 24 итерации:

$$x = \begin{pmatrix} 0.24226071 \\ 0.6382838 \\ 0.79670669 \\ 2.3227488 \end{pmatrix}$$

Невязка результата:

$$||Ax - b||_2 = 1.66e - 06$$

Результат удовлетворяет заданной точности.

Рассмотрим результаты для матрицы A_3:

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} x = \begin{pmatrix} 4 \\ 5 \end{pmatrix}.$$

При заданой точности $tol = 10^{-4}$ и $n = 8$, был получен результат за 16 итерации:

$$x = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Невязка результата:

$$||Ax - b||_2 = 1.81e - 08$$

Результат удовлетворяет заданной точности.

5 Заключение

В результате лабораторной работы был успешно реализован метод Ричардсона. Проведено тестирование программы на разных видах матриц, в том числе неположительно определнных. Полученные результаты показывают эффективность метода для решения СЛАУ.

Список литературы

[1] Абакумов М.В.б Соснин Н.В. *Лекции по численным методам.* МАКС Пресс, Москва, 2022.