Название статьи

Автор: Имя Фамилия

Дата: 19 декабря 2024 г.

Содержание

1 Введение **3**

2 Основная часть **3**

2.1 Программная реализация 5

3 Заключение **7**

4 Приложение **7**

5 Решения теоретических задач **7**

5.1 Неравенство 1 7

5.2 Неравенство 2 7

Список литературы **8**

*

1 Введение

Решение систем линейных алгебраических уравнений (далее СЛАУ) - является одной из фундаментальных задач вычислительной математики, которая широко применяется в различных облостях, начиная с инженерии и заканчивая экономикой и построенимем моделей в биологии. Для решениях СЛАУ были созданы различные методы, каждый из которых имеет свои плюсы и минусы, которые важны для конкретных задач.

В рамках данной курсовой работы исследуются метод Гаусса с выбором ведущего элемента по всей матрицы.

Цель работы: исследовать метод Гаусса с выбором ведущего элемента по всей матрицы, сравнить с обычным методом гаусса.

Задачи:

1. Изучить теоретические основы метода Гаусса с выбором главного элемента по всей матрицы
2. Программно реализовать алгоритм метода
3. Провести вычислительные эксперименты с различными типами матриц
4. Сравнить результаты с обычным методом Гаусса
5. Сформулировать выводы

Работа направлена на развитие навыков программирования численных методов и углубленное понимание их свойсств.

2 Основная часть

Метод Гаусса с выбором ведущего элемента по всей матрице

Рассмотрим систему линейных уравнений:

$$A\mathbf{x} = \mathbf{b},$$

где A — квадратная матрица размера $n \times n$, $\mathbf{x}$ — вектор неизвестных, а $\mathbf{b}$ — вектор правых частей.

Метод Гаусса с выбором ведущего элемента по всей матрице выполняется в три этапа:

1. Прямой ход

1. На k-м шаге ($k = 1, 2, \ldots, n-1$) среди оставшихся элементов матрицы A (начиная с k-й строки и k-го столбца) выбирается элемент с максимальным абсолютным значением:
$$|a_{p,q}| = \max_{i \geq k, j \geq k} |a_{i,j}|,$$
где p и q — индексы строки и столбца, соответствующих максимальному элементу.

2. Выполняется перестановка строк k-й и p-й, а также столбцов k-го и q-го. Эти перестановки должны быть учтены в решении.

3. С использованием выбранного ведущего элемента $a_{k,k}$ производится исключение переменных: строки матрицы и элементы правой части преобразуются по следующим формулам:

$$a_{i,j} := a_{i,j} - \frac{a_{i,k} \cdot a_{k,j}}{a_{k,k}}, \quad b_i := b_i - \frac{a_{i,k} \cdot b_k}{a_{k,k}}, \quad \forall i > k, \forall j \geq k+1.$$

2. Обратный ход

После завершения прямого хода матрица A превращается в верхнюю треугольную матрицу. Решение системы находится методом подстановки, начиная с последнего уравнения:
$$x_i = \frac{b_i - \sum_{j=i+1}^{n} a_{i,j} x_j}{a_{i,i}}, \quad i = n, n-1, \ldots, 1.$$

3. Учет перестановок

Если в процессе решения выполнялись перестановки столбцов, то решение $\mathbf{x}$ необходимо переставить обратно в соответствии с первоначальным порядком столбцов.

2.1 Программная реализация

Для реализации этого алгоритма был выбран ЯП Python из-за простоты реализации подобных методов

```python
class Solver:
 def __init__(self, A: np.array, b: np.array):
  self._A = A.astype(float)
  self._n = A.shape[0]
  self._A_resolve = None
  self._b = b.astype(float)
  self._perm = list(range(self._n))
 def _find_max_elem(self, k: int):
  max_value = np.max(np.abs(self._A_resolve[k:, k:]))
  max_index = np.where(abs(self._A_resolve[k:, k:]) == max_value)
  return max_index[0][0] + k, max_index[1][0] + k
 def solveGaussWithAllMax(self):
  self._A_resolve = self._A.copy()
  for k in range(self._n - 1):
   max_row, max_col = divmod(np.abs(self._A_resolve[k:, k:]).argmax(), self._n - k)
   max_row += k
   max_col += k
   if max_row != k:
    self._A_resolve[[k, max_row]] = self._A_resolve[[max_row, k]]
    self._b[[k, max_row]] = self._b[[max_row, k]]
   if max_col != k:
     self._A_resolve[:, [k, max_col]] = self._A_resolve[:, [max_col, k]]
     self._perm[k], self._perm[max_col] = self._perm[max_col], self._perm[k]
   for i in range(k + 1, self._n):
    factor = self._A_resolve[i, k] / self._A_resolve[k, k]
    self._A_resolve[i, k:] -= factor * self._A_resolve[k, k:]
    self._b[i] -= factor * self._b[k]
  x = np.zeros(self._n)
  for i in range(self._n - 1, -1, -1):
  x[i] = (self._b[i] - np.dot(self._A_resolve[i, i + 1:], x[i + 1:])) / self._A_resolve[i, i]
  x_final = np.zeros(self._n)
  for i, p in enumerate(self._perm):
  x_final[p] = x[i]
```

3 Заключение

4 Приложение

5 Решения теоретических задач

5.1 Неравенство 1

$$\frac{1}{\sqrt{n}}N(A) \leq ||A||_2 \leq N(A), \text{ где } N(A) = \sqrt{\sum_{i,j} |a_{i,j}|^2}.$$

Рассмотрим левую часть этого двойного неравенства:

$$\frac{1}{\sqrt{n}}N(A) \leq ||A||_2$$

$$||A||_2^2 = \max_i \lambda_i \geq \frac{1}{n}(\lambda_1 + \lambda_2 + \cdots + \lambda_n) = \frac{1}{n}\mathrm{Sp}(A * A) = \frac{1}{n}N(A)^2.$$

Взяв квадратные корни от правой и левой части, получим исходное неравенство.

$$||A||_2 \geq \frac{1}{n}N(A), \tag{1}$$

Для правой части проведём аналогичные рассуждения:

$$||A||_2 \leq N(A)$$

$$||A||_2^2 = \max_i \lambda_i \leq \lambda_1 + \lambda_2 + \cdots + \lambda_n = \mathrm{Sp}(A * A) = N(A)^2.$$

Взяв квадратные корни от правой и левой части, получим исходное неравенство.

$$||A||_2 \leq N(A). \tag{2}$$

Из неравенств (1) и (2) следует целевое двойное неравенство:

$$\frac{1}{\sqrt{n}}N(A) \leq ||A||_2 \leq N(A).$$

5.2 Неравенство 2

$$||A||_2^2 \leq ||A||_1 \cdot ||A||_\infty$$

Рассмотрим квадрат второй матричной нормы:

$$\max_i \lambda(A) \leq \sum_i \lambda_i = \mathrm{Sp}(A * A) = \sum_{i,j} |a_{i,j}|^2$$

Распишем правую сумму через неравенство Коши-Буняковского:

$$\sum_{i,j} |a_{i,j}|^2 = \sum_{i,j} |a_{i,j}| \cdot |a_{i,j}| \leq$$

$$\leq \max_i \sum_j |a_{i,j}| \cdot \max_j \sum_i |a_{i,j}| = ||A||_1 \cdot ||A||_\infty$$

Список литературы