Академический университет

Сравнение метода Зейделя и градиентного спуска

Николай Жидков

15 марта 2018 г.

1 Использование структуры матрциы

Так как наша матрица представляет собой несколько блоков, идущих по центру, а все остальные элементы равны 0, то сразу сохраним матрицу как набор этих блоков. Дальше можно заметить, что оба исследуемых метода используют при пересчете только базовые операции +, -, домножение на константу, и умножение на блочную матрицу A. Из этого можно сделать вывод, что можно сразу решать задачу по блоком, а потом просто соединить ответ. Соответсвенно, в программе везде n - это лист из размерностей блоков, A - список блоков и так далее. Мы для каждого блока делаем свои преобразования и считаем, когда разница между соседними глобальными(!) x_k и x_{k+1} станет меньше eps.

2 Структура программы

Программа разделена на функции, записанные в файле solve.py. Основных функций 3, остальные должны быть понятны из названий

- $read(filename)$, функция чтения:
  - Принимает название файла для чтения данных
  - Возвращает лист размерностей n, лист блоков A, лист желаемых решений x_s и лист желаемых начальных точек x_0.
- $seidel(n, A, x_s, x_0, eps, full\_mode)$, находит решение методом Зейделя:

  - Принимает лист размерностей n, лист блоков A, лист желаемых решений x_s и лист желаемых начальных точек x_0, минимальную норму между соседями eps и флаг дебажного вывода
  - Возвращает лист решений для каждого блока

- $grad(n, A, x_s, x_0, eps, full\_mode)$, находит решение градиентным спуском:
  - Принимает лист размерностей n, лист блоков A, лист желаемых решений x_s и лист желаемых начальных точек x_0, минимальную норму между соседями eps и флаг дебажного вывода
  - Возвращает лист решений для каждого блока

3 Структура файлов исходных данных

Первые n строк входного файла описывают матрицу A. Строка i имеет следующую структуру: $A_{i,0}, A_{i,1}, ..., A_{i,n-1}$. $n+1$-ая строка содержит желаемый вектор решения x^* в формате $x^*_0, ..., x^*_{n-1}$. $n+1$-ая строка содержит желаемый стартовый вектор $x0$ в формате $x0_0, ..., x0_{n-1}$
Пример содержимого файла для системы третьего порядка:

1 2 3
4 5 5
7 8 10
12 3 -5
17 -2 40

4 Примеры вызова из командной строки

- Запуск, выводится только ответ (файл с входными данными обязательно указывать первым параметром!) методом Зейделя (метод тоже обязательно надо указать)

  python3 solve.py input.txt –method==seidel

- Запуск, выводится только ответ (файл с входными данными обязательно указывать первым параметром!) методом градиентного спуска

  python3 solve.py input.txt –method==grad

- Запуск, выводится вся дебаг информация

  python3 solve.py input.txt –method==grad –full

- Запуск с заданным eps

  python3 solve.py input.txt –method==seidel –eps=0.001

5 Численный эксперимент

Везде использовался вектор $x* = [1, 2, -3, -4, 5, 6, -7, -8, 9, -10]$
Далее приведены таблички для разных стартовых точек X_0
$X_0 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$

Заданный eps (%)	Итерации метода Зейделя (%)	Итерации градиентного спуска
1e-2	18	28
1e-3	28	34
1e-4	38	48
1e-5	49	63
1e-6	59	70

$X_0 = [10, -3, 5, 17, -0.1, 0.2, 7, 20, 10, 0]$

Заданный eps (%)	Итерации метода Зейделя (%)	Итерации градиентного спуска
1e-2	22	34
1e-3	32	48
1e-4	43	62
1e-5	53	76
1e-6	63	92

$X_0 = [0, 1, 2, 4, 8, 16, 32, 64, 128, 256]$

Заданный eps (%)	Итерации метода Зейделя (%)	Итерации градиентного спуска
1e-2	15	41
1e-3	25	50
1e-4	35	60
1e-5	46	70
1e-6	56	85

5.1 Выводы

Как видно из результатов при выборе разных X_0 метод Зейделя оказывается неизменно немного лучше метода градиентного спуска для данной матрицы A.