Отчет по лабораторной работе №1: Решение СЛАУ численными методами

Чигарев Дмитрий 381807-1

1. Подготовка системы

Исходная СЛАУ, предлагаемая для решения имеет вид:

Для удобства работы с системой, приведем её к виду, в котором преобладают диагональные элементы матрицы (на диагонали стоят максимальный в строке элемент):

|  |
| --- |
| *# Примечание: коды всех используемых функций вынесены в приложении*  >>> [A, b] = transform(A, b)  >>> disp("Преобразованная система с преобладанием диагональных элементов:", [A, b])  "Преобразованная система с преобладанием диагональных элементов:"  2.389 0.273 0.126 0.418 0.144  0.329 2.796 0.179 0.278 0.297  0.186 0.275 2.987 0.316 0.529  0.197 0.219 0.274 3.127 0.869 |

2. Устойчивость задачи

Исследуем устойчивость задачи решения данной СЛАУ — оценим зависимость изменения вектора решения от малых колебаний системы.

Изменение решения при:

Справедливой оценкой для коэффициентов и служит число обусловленности:

Вычислим число обусловленности для нашей системы:

|  |
| --- |
| >>> cond\_n = cond(A)  >>> printf("Число обусловленности равно: %f\n", cond\_n)  Число обусловленности равно: 1.667647 |

Число обусловленности является коэффициентом роста относительной погрешности при возмущении системы. Малыми “возмущениями системы” могут стать ошибки округления при формировании СЛАУ или же ошибки измерения данных, представленных в системе.

3, 4. Оценка погрешностей

По условию задания: *абсолютная погрешность всех членов системы равна 0.001.* Оценим, с какой точностью мы можем получить решение.

Для этого определим величину относительной и абсолютной погрешности решения.

В случае нашей системы оценки примут следующее значение:

|  |
| --- |
| >>> [rel\_err, abs\_err] = get\_err(A, b, err=0.001)  >>> printf("Оценка относительной погрешности решения: %f\n", rel\_err)  Оценка относительной погрешности: 0.002319  >>> printf("Оценка абсолютной погрешности решения: %f\n", abs\_err)  Оценка абсолютной погрешности: 0.000455 |

Из числа обусловленности и оценки относительной погрешности, можно сделать вывод о гарантированной точности решения в 2 знака после запятой.

Соответственно, ответом на вопрос: можно ли гарантировано получить решение с точностью *0.001*, для системы, у которой погрешность членов системы также составляет *0.001,* – нет.

5, 6. Метод Гаусса

Решим СЛАУ методом Гаусса. Для этого приведем систему к треугольному виду, а затем последовательно выразим неизвестные переменные.

|  |
| --- |
| >>> gaus\_x = solve\_with\_gaus(A, b)  >>> disp("Метод Гаусса выдал следующее решение:", gaus\_x)  "Метод Гаусса выдал следующее решение:"  -0.0009828  0.0712863  0.1430468  0.2604372 |

Посчитаем относительную и абсолютную погрешность полученного решения:

|  |
| --- |
| >>> ref\_x = A^-1 \* b *// Точное решение* >>> printf("Абсолютная погрешность: %.e\n", norm(gaus\_x - ref\_x))  Абсолютная погрешность: 2.2708e-17  >>> printf("Относительная погрешность: %.e\n", norm(gaus\_x-ref\_x)/norm(ref\_x))  Относительная погрешность: 6.9389e-18 |

7. Метод простых итераций

Итерационные методы основаны на последовательном приближении начального решения.

Исходная СЛАУ преобразовывается к эквивалентной ей:

Далее строится ряд:

Можно доказать, что при выполнении некоторых условий, накладываемых на матрицу , этот ряд сходится к точному решению :

Отсюда можно вывести оценку абсолютной погрешности решения на шаге итерационного процесса:

Также видно, что, решив следующее неравенство для мы вычислим необходимое кол-во итераций для получения решения с заданной точностью:

Ограничением на матрицу , является:

Поиск собственных чисел матрицы является трудоемкой задачей, и потому, проверка этого условия не является частью работы МПИ. В реализации метода из приложения это условие проверяется лишь для демонстрационных целей.

8. Метод Якоби

Изложенное выше, является определением метода простых итераций. Метод Якоби задает правила по преобразованию исходной системы к виду, пригодному для использования метода МПИ, а также предлагает способ задания начального приближения

Матрица , вектор и начальное приближение строятся следующим правилам:

Решим нашу систему методом Якоби с заданной погрешностью в и :

|  |
| --- |
| >>> x1 = solve\_with\_iter(A, b, eps=0.01, verbose=%T)  Запуск итерационного метода на 4 шагов...  Шаг: 1 | Погрешность: 0.09795509 |  Текущее решение:  -0.009827  0.0601614  0.1341681  0.2511471  Шаг: 2 | Погрешность: 0.02442234 |  Текущее решение:  0.0023823  0.0738191  0.1456046  0.2625515  Шаг: 3 | Погрешность: 0.00678556 |  Текущее решение:  -0.001777  0.0705163  0.1423804  0.2598237  Шаг: 4 | Погрешность: 0.00182003 |  Текущее решение:  -0.0007523  0.0714834  0.1432321  0.2605995  >>> x2 = solve\_with\_iter(A, b, eps=0.01, verbose=%T)  Запуск итерационного метода на 6 шагов...  Шаг: 1 | Погрешность: 0.09795509 |  Текущее решение:  -0.009827  0.0601614  0.1341681  0.2511471  Шаг: 2 | Погрешность: 0.02442234 |  Текущее решение:  0.0023823  0.0738191  0.1456046  0.2625515  Шаг: 3 | Погрешность: 0.00678556 |  Текущее решение:  -0.001777  0.0705163  0.1423804  0.2598237  Шаг: 4 | Погрешность: 0.00182003 |  Текущее решение:  -0.0007523  0.0714834  0.1432321  0.2605995  Шаг: 5 | Погрешность: 0.00049640 |  Текущее решение:  -0.0010435  0.0712311  0.1429972  0.2603926  Шаг: 6 | Погрешность: 0.00013446 |  Текущее решение:  -0.000966  0.071301  0.1430604  0.2604492 |

9. Метод Зейделя

Метод Зейделя основан на методе Якоби. Его отличием является то, что при подсчете компоненты приближенного вектора , учитываются компоненты, уже полученные на этом шаге:

В методе Зейделя будем пользоваться следующим критерием останова (вместо заранее рассчитанного кол-ва шагов метода):

При выполнении этого условия будем останавливать итерационный процесс.

Решим нашу систему методом Зейделя с заданной погрешностью в :

|  |
| --- |
| >>> x3 = solve\_with\_zeidel(A, b, eps=0.01, verbose=%T)  Шаг: 1 | Погрешность: 0.088608.8 |  Текущее решение:  -0.009827  0.0684103  0.1420147  0.2612862  Шаг: 2 | Погрешность: 0.009593.8 |  Текущее решение:  -0.0007482  0.0712403  0.1429466  0.2604344  Шаг: 3 | Погрешность: 0.000250.8 |  Текущее решение:  -0.0009717  0.0712917  0.143046  0.2604362 |

10. Сравнение исследованных методов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод | Решение | Относительная погрешность | Абсолютная погрешность | Сложность |
| Гаусс | -0.0009828  0.0712863  0.1430468  0.2604372 | 2.2708e-17 | 6.9389e-18 |  |
| Якоби | -0.000966  0.071301  0.1430604  0.2604492 | 9.4058e-05 | 2.8741e-05 | где - кол-во итераций |
| Зейдель | -0.0009717  0.0712917  0.143046  0.2604362 | 4.0389e-05 | 1.2342e-05 | где - кол-во итераций |

Приложения

1. Ссылка на репозиторий с исходным кодом: <https://github.com/proxodilka/numerical-analysis-labs/blob/master/lab1_system_linear_equtions/lab1.sce>