

Национальный исследовательский университет ИТМО

Кафедра программных систем

**Практическая работа 5**

Выполнил: Шебут

Денис Айссаевич

Группа № K3221

Проверил: Иванов С. Е.

Санкт-Петербург

2020

Практическая работа №5

РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ МЕТОДОМ НЬЮТОНА

Для выполнения данной работы расширим возможности ранее созданного класса Matrix. Matrix теперь будет представлять из себя класс шаблон Matrix<T> и иметь наследника - класс NumericMatrix, содержащий в себе методы работы с матрицами чисел. В Matrix<T> добавим два перегруженных метода ConcatHorizontally для удобного объединения матриц (определим такие же методы и в классе наследнике):

Изображение выглядит как снимок экрана, компьютер

Автоматически созданное описание

Рисунок 1 – добавленные методы в классе NumericMatrix

Также внесем небольшие изменения в класс SLAE (system of linear algebraic equations). Изменим тип хранимого единственного решения (double[] single\_solution) с double[] на List<double> и добавим в этот класс новый конструктор, инициирующий СЛАУ по расширенной матрице, а не только с клавиатуры, как это было раньше.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 2 – новый конструктор класса SLAE

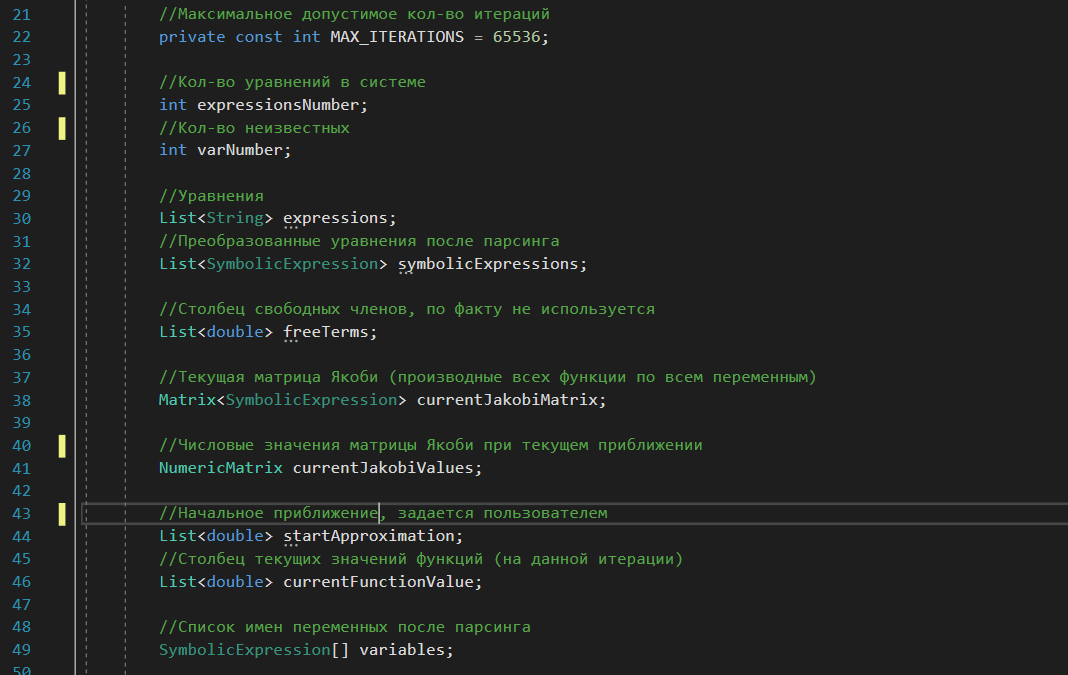
Теперь перейдем непосредственно к системам нелинейных алгебраических уравнений. Программа реализует данный алгоритм решения:

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 3 – алгоритм решения СНАУ

1. Создадим класс SNAE (system of non-linear algebraic equations). Для парсинга введенных с клавиатуры уравнений и их дифференцирования будем использовать библиотеки MathNet.Symbolic и MathNet.Numeric соответственно
   1. Объявим поля класса:



Изображение выглядит как сидит, держит, экран, мужчина

Автоматически созданное описание

Рисунки 4-5 – поля класса SNAE. Класс SymbolicExpression является представлением распарсенного алгебраического выражения в библиотеке MathNet

* 1. Перейдем к методам класса. Напишем конструктор по умолчанию и метод инициализации с клавиатуры:

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 6 – конструктор по умолчанию класса SNAE

Изображение выглядит как снимок экрана, телефон, компьютер

Автоматически созданное описание

Рисунок 7 – метод инициализации СНАУ с клавиатуры. Имена и кол-во переменных определяются по ходу ввода. Парсится только левая часть уравнения с предварительно перенесенным свободным членом. Свободные члены записываются в лист freeTerm (просто, чтобы были)

* 1. Добавим метод инициализации начального приближения. Эта процедура выделена в отдельный метод, чтобы потом организовать подбор множества решений по начальным приближениям:

Изображение выглядит как снимок экрана, сидит, монитор, экран

Автоматически созданное описание

Рисунок 8 – метод инициализации нового начального приближения

* 1. Добавим общий метод построения решения системы (впоследствии в этом методе может быть организован выбор между методами решения):

Изображение выглядит как снимок экрана, сидит, телефон

Автоматически созданное описание

Рисунок 9 – метод построения решений

* 1. Изображение выглядит как снимок экрана, компьютер, экран, телефон

     Автоматически созданное описаниеНапишем метод, решающий систему методом Ньютона:

Изображение выглядит как снимок экрана, телефон, компьютер

Автоматически созданное описание

Изображение выглядит как снимок экрана, сидит, экран, монитор

Автоматически созданное описание

Рисунки 10-12 – метод, находящий ближайшее к стартовому приближению решение системы методом Ньютона

1. Рассмотрим, какие методы класса SNAE использованы для решения системы в пункте 1 e.
   1. Напишем метод нахождения матрицы Якоби для заданных функций и переменных:

Изображение выглядит как снимок экрана, монитор, экран, сидит

Автоматически созданное описание

Рисунок 13 – заданные функции дифференцируются по всем переменным. Использован метод SymbolicExpression.Differentiate() из библиотеки MathNet

* 1. Изображение выглядит как снимок экрана, монитор, сидит, экран

     Автоматически созданное описаниеДобавим метод, подсчитывающий значения матрицы Якоби при текущем приближении:

Рисунок 14 – метод, подсчитывающий значений матрицы Якоби при текущем приближении. Используется метод SymbolicExpression.Evaluate(Dictionary<string, FloatingPoint> variables) для вычисления значений. Класс FloatingPoint – представление чисел с плавающей точкой в библиотеке MathNet

* 1. Добавим метод, подсчитывающий значения заданных функций при текущем приближении:

Изображение выглядит как снимок экрана, сидит, ноутбук, монитор

Автоматически созданное описание

Рисунок 15 - – метод подсчитывающий значений функций при текущем приближении

* 1. Найдем ∆x решив полученную СЛАУ в методе GetLinearSystemSolution():

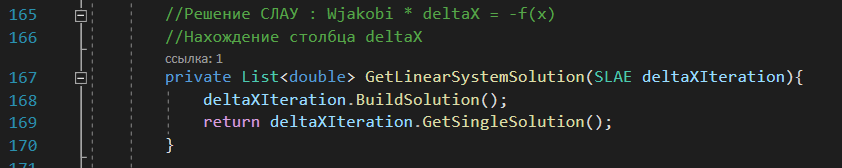


Рисунок 16 – метод, возвращающий решение заданной системы

1. Переопределим метод ToString():

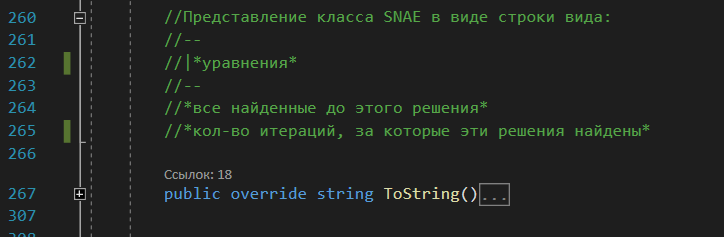


Рисунок 17 – описание реализации метода ToString() класса SNAE

**Вывод:** в ходе выполнения данной практической работы был реализован класс SNAE обеспечивающий не только удобные инструменты работы с системами нелинейных алгебраических уравнений, но и являющийся целостным представлением данных. Был реализован алгоритм поиска решений системы на основе метода Ньютона.