Реализация итерационного метода решения нелинейных СЛАУ (2.3.2м – метод Ньютона-Рафсона: модифицированный)

1. Суть метода и алгоритм решения:

Дано уравнение $\mathbf{f}(\mathbf{x}) = \mathbf{0} \dot{\Leftrightarrow} \begin{cases} f_1(x_1, \dots, x_s) = 0 \\ \dots \dots \dots \\ f_s(x_1, \dots, x_s) = 0 \end{cases}$

Искомым решением является вектор $x = (x_1, \dots, x_s)^T$, который строится с помощью итерационного метода.

В основе данного метода лежит разложение векторной функции $f(x)$ ряд Тейлора. То есть: $f(x) = f(x_k) + f'(x_k)(x - x_k) + O((x - x_k)^2)$, из чего следует, что каждое новое итерационное решение $x_{k+1} = x_k - f_k'^{-1} f_k$, где x_0 – начальное приближение, $f_k = f(x_k)$, $f_k'^{-1} = f'^{-1}(x_k), f'$ – матрица Якоби.

Также рассматривается модифицированный метод Ньютона-Рафсона, который будет взят за основу будущей реализации. Основное отличие в том, что в итерационной формуле f_k' заменяется на постоянную $f_0' = f'(x_0)$

Существуют 3 основных критерия прекращения итерационного процесса:

1) По числу итераций. $k < K_{max}$, где K_{max} задается пользователем
2) По близости к решению нормы разности $x_{k+1} - x_k$, которая должна быть меньше заданного δ
3) По малости нормы значения функции в текущем итерационном методе. Такая норма должна быть меньше некоторого ε, которое задается пользователем

Для корректности работы стоит использовать как минимум два критерия прекращения итерационного процесса. Мною выбран 1 и 3 критерий.

2. Программная реализация:

Входные данные: система уравнений $\begin{cases} f_1(x_1, \dots, x_s) = 0 \\ \dots \dots \dots \\ f_s(x_1, \dots, x_s) = 0 \end{cases}$ и начальное приближение x_0

Выходные данные: найденный вектор $x = (x_1, \dots, x_s)^T$.

Модифицированный метод Ньютона-Рафсона реализован с помощью Wolfram Mathematica:

```
jacInv = Inverse@Transpose@{Flatten@D[f, {x}], Flatten@D[f, {y}]};
jacInv0 = jacInv /. Thread[{x, y} → Flatten@x0];
fK = f /. Thread[{x, y} → Flatten@#] &;

res = NestWhileList[# - jacInv0.fK[#] &, x0, Norm[fK[#]] > e &, 1, kmax, 0];
{Flatten@Last@res, Length@res - 1}
```

Параметр $kmax$ задается пользователем и определяет максимальное количество итераций. Параметр e также задается пользователем и определяет число, норма значения исходной функции от текущего итерационного решения которого должна быть меньше этого числа, чтобы остановить итерационный процес.

3. Результат вычисления на предоставленных входных данных и оценка точности решения:

Для оценки точности решения рассмотрим входные данные, тип данных входящих систем – матрица, где каждая строка - нелинейное уравнение.

Входные данные:

1) $f(x) = \begin{pmatrix} x_1^2 - x_2^2 - 1 \\ x_1 x_2^3 - x_2 - 1 \end{pmatrix}, x_0 = \begin{pmatrix} 1.5 \\ 1.5 \end{pmatrix}$

Рассмотрим полученное решение с помощью написанной мною функции и сравним ее точность со встроенной функцией.

1. Собственная реализация при $kmax = 500, \varepsilon = 1.\times 10^{-8}$

Полученные данные:

```
{{1.50284, 1.12185}, 27}

Norm@fK[Last@res]

5.99679×10^-9
```

Поскольку моя реализация также записывает k, на котором итерационный процесс прервался несложно увидеть, что собственная реализация считает довольно быстро и точно. Ей было достаточно 27 итераций из возможных 500, чтобы получить норму меньшую заданному ξ

2. Решение с помощью встроенной функции $NSolve$:

```
x = {q, w} /. NSolve[f[q, w] == 0, {q, w}, Reals] // First
```

{1.50284, 1.12185}

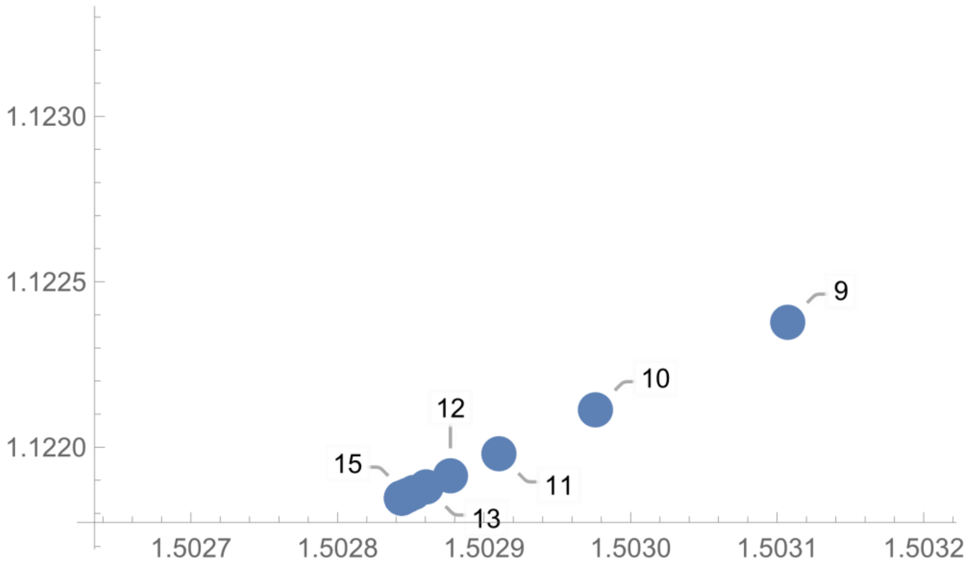
```
Norm@f[Sequence @@ Flatten@x]
```

3.33067×10^{-15}

Нормы векторов невязки очень малы, что означает, что написанная мною функция работает довольно таки точно

Также для еще одного подтвержения правильности собственной реализации рассмотрим диаграмму разброса данных:

На данном графике видно, что с повышением числа k найденное решение на k итерации становится все ближе и ближе к искомому решению и начиная с некоторого номера они практически не отличаются.



Рассмотрим еще один пример:

$$2)\ f(x) = \begin{pmatrix} sinx_1 - x_2 - 1.32 \\ cosx_2 - x_1 + 0.35 \end{pmatrix}, x_0 = \begin{pmatrix} 1.8 \\ -0.3 \end{pmatrix}$$

1. Собственная реализация при $kmax = 200, \varepsilon = 1.\times 10^{-10}$

Полученные данные:

{{1.28578, -0.360344}, 15}

```
Norm@fK[Last@res]
```

8.90613×10^{-11}

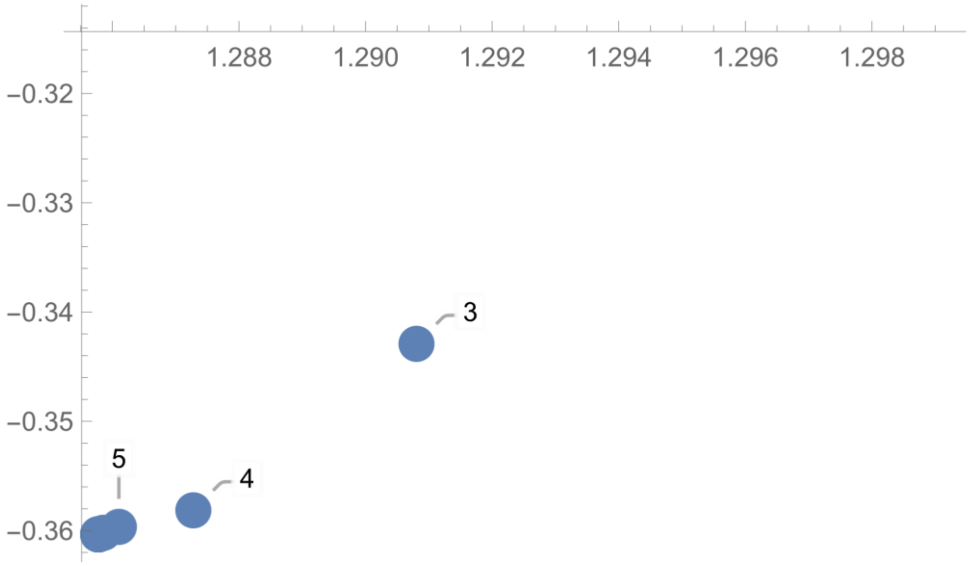
Несложно увидеть, что собственная реализация снова посчитала довольно быстро и точно. Ей было достаточно 15 итераций из возможных 200, чтобы получить норму меньшую заданному ξ

2. Решение с помощью встроенной функции $NSolve$:

```
x = NSolve[{Sin@x - y - 1.32 == 0, Cos@y - x + 0.35 == 0}, {x, y}]

{{1.28578}, {-0.360344}}

Norm@ f[x]

3.23178 × 10^-16
```

Нормы векторов невязки очень малы, что означает, что написанная мною функция работает довольно таки точно

Также для еще одного подтверждения правильности собственной реализации снова рассмотрим диаграмму разброса данных:



На данном графике также видно, что с повышением числа k найденное решение на k итерации становится все ближе и ближе к искомому решению и начиная с некоторого номера они практически не отличаются.