Реализация решения методом итераций системы нелинейных алгебраических уравнений (Метод секущих. Использование предшествующих приближений)

Задача:1.2.4(а)(1)

Представим, что требуется решить систему нелинейных уравнений:

$\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$ или же $\begin{cases} f_1(x_1, \dots, x_s) = 0 \\ \dots \\ f_s(x_1, \dots, x_s) = 0 \end{cases}$, где $x_k = (x_1, \dots, x_s)^T$ – искомое решение

Искомое решение строится с помощью итерационного метода.

Рассмотрим метод секущих и его подтип, в котором используются предшествующие приближения.

Метод секущих является дискретной модификацией метода Ньютона-Рафсона. Основную суть заимствуется оттуда:

Каждое следующие приближение строится по формуле:

$x_{k+1} = x_k - f_k'^{-1} f_k$, где $f_k = f(x_k), f_k' = f'(x_k)$

Главным отличием метода секущих является тот факт, что f_k' вычисляется иным образом:

$f_k' = f'(x_k)$ приблизительно равно $H_k^{-1}\Gamma_k$, а следовательно, $f_k'^{-1} = \Gamma_k^{-1} H_k$

В общем виде данные матрицы являются аналогами матриц приращения аргумента и функции от данного аругмента.

В данной работе рассматривается метод предшествующих приближений. Его особенность состоит в том, что в начале выбирается не 1 приближение, а s приближений, которые лежат в окресности точки x_0, задаваемое вначале. Далее обе матрицы строятся с применением s предшествующих приближений. То есть для каждого $k > s$

приближения используются $\{x_{k-j}, j = 1,2, \dots, s\}$ предыдущих приближений.

На каждой итерации вычисляются матрицы H_k и Γ_k.

Столбец j матрицы H_k – координаты разности двух векторов x_k и x_{k-j}.

Столбец j матрицы Γ_k – координаты разности $f(x_k)$ и $f(x_{k-j})$.

Поскольку любые x_i и $f(x_i)$ состоят из s компонент, следовательно, полученные матрицы H_k и Γ_k

Квадратные, то есть принцип обратной матрицы не рушится.

Таким образом конечная формула следующая:

$x_{k+1} = x_k - \Gamma_k^{-1} H_k f_k$, где $k = s, s+1, \ldots$

Чтобы любой итерационный метод работал корректно, необходимо определить условия остановки вычислений следующей итерации.

Критерии остановки итерационного процесса следующие:

1) По числу итераций: Пользователь сам выбирает такое число k, по достижению которого процесс вычислений прекращается.
2) По близости к решению. Пользователь задает какое-то малое число, с которым сравнивается на каждой итерации норма разности текущего и нового приближения. И если эта норма меньше заданного числа, то вычисления прекращаются
3) По малости вязки. Пользователь задает какое-то мало число, с которым сравнивается на каждой итерации норма вектора невязки

$$f(x_k)$$

Программная реализация:

Мною была реализована одна функции. Она обладает двумя критериями остановки: основной – 2, вспомогательный - 1.

```
s = Length@x0;
c := RandomReal[{-0.00001, 0.00001}, {1, 2}] // Flatten;
x = {x0};

Do[AppendTo[x, x[[i - 1]] + c], {i, 2, s + 1}];
While[k ≤ kmax && Norm[x[[k]] - x[[k - 1]]] > e,
  h = Table[x[[k]] - x[[k - j]], {j, 1, s}] // Transpose;
  g = Table[f[Sequence @@ x[[k]]] - f[Sequence @@ x[[k - j]]], {j, 1, s}] // Transpose;
  res = Transpose[x[[k]]] - Inverse[g].h.Transpose[f[Sequence @@ x[[k]]]];
  AppendTo[x, Transpose[res]];
  k++
 ];

res
```

Реализация написана с помощью среды $Wolfram\ Mathematica$. Необходимо узнать, насколько точно решает данная функции заданную систему.

Для этого найдем вектор невязки – разность левой и правой части с подставленным найденным с помощью реализации решением – и определим его норму — точность вычисленного решения. Также следует сравнить решение со встроенными функциями среды $Wolfram\ Mathematica$, которые также решает поставленную задачу.

Стоит отметить, что для корректной работы реализуемого метода достаточно, чтобы норма вектора невязки была не больше 10^{-6}

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 $x = (x_1, \dots, x_s)^T$.

Пример. Входные данные:

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

Выходные данные, полученные собственной реализацией при k max = 200 и $e = 0{,}0000001$:

В данном случае норма невязки очень мала, что означает, что полученное решение довольно точное.

```
res
Norm[f[Sequence @@ res]]
```

```
{1.50284, 1.12185}
```

$$9.03465 \times 10^{-9}$$

Сравним со встроенной функцией:

```
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}

Данная функция считает точнее, чем моя реализация, однако, если уменьшить значение e и повысить значение $kmax$ она ничем не будет уступать.

Стоит отметить, что поскольку $f(x) = 0$, достаточно посчитать норму самого $f(x_k)$, что будет нормой невязки.

При довольно больших K и малых E нормы векторов невязки практически не отличаются от норм встроенных функций, а в некоторых случаях могут даже быть меньше них, что подтверждает правильность работы собственной реализации