Метод Ньютона для решения систем нелинейных уравнений

Шепрут Илья

20 января 2019 г.

Всё будет описано для размерности $n = 2$, но это легко обобщается на более высокие размерности.

1. Формулировка

Дана Система Нелинейных Уравнений (СНУ) в виде:

$$\begin{cases} F_1(x_1,\ x_2) = 0 \\ F_2(x_1,\ x_2) = 0 \end{cases} \tag{1}$$

Требуется, чтобы она имела непрерывные производные 1 порядка.

Необходимо найти такие $x_1,\ x_2$ при которых система оборачивается в нуль. Это называется решение СНУ.

2. Обозначения

Заменим в (1) аргументы на вектор $\mathbf{x} = (x_1,\ x_2)$:

$$\begin{cases} F_1(\mathbf{x}) = 0 \\ F_2(\mathbf{x}) = 0 \end{cases} \tag{2}$$

Преобразуем (2) к одной вектор-функции $\mathbf{F}$, такой что $\mathbf{F} : \mathbb{R}^2 \to \mathbb{R}^2$:

$$\mathbf{F}(\mathbf{x}) = \begin{pmatrix} F_1(\mathbf{x}) \\ F_2(\mathbf{x}) \end{pmatrix}$$

Тогда (1) примет вид:

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \tag{3}$$

где $\mathbf{0} = (0,\ 0)^T$.

Под **невязкой** понимается вектор $\mathbf{F}(\mathbf{x})$.

3. Преобразования

3.1. Разложение в ряд Тейлора для функции многих переменных

Взято с Википедии.

Пусть дана функция $f(\mathbf{x}) : \mathbb{R}^2 \to \mathbb{R}$, $\mathbf{x} = (x_1,\ x_2)^T$.

Разложение функции $f(\mathbf{x})$ будет произведено в окрестности точки $\mathbf{a} = (a_1,\ a_2)^T$. Требуется, чтобы у функции f имелись непрерывные производные до $(2+1)$-го порядка включительно по всем переменным.

Тогда разложение в ряд Тейлора имеет следующий вид:

$$f(\mathbf{x}) = f(\mathbf{a}) + \sum_{i=1}^{n} \frac{\partial f(\mathbf{a})}{\partial x_i}(x_i - a_i) + \frac{1}{2!}\sum_{i=1}^{2}\sum_{j=1}^{2} \frac{\partial f(\mathbf{a})}{\partial x_i \partial x_j}(x_i - a_i)(x_j - a_j) + R_3(\mathbf{x}) \quad (4)$$

Раскроем сумму первых двух членов предыдущего равенства:

$$f(\mathbf{x}) = f(\mathbf{a}) + \frac{\partial f(\mathbf{a})}{\partial x_1}(x_1 - a_1) + \frac{\partial f(\mathbf{a})}{\partial x_2}(x_2 - a_2) + R_2(\mathbf{x}) \quad (5)$$

Где $R_n(\mathbf{x})$ — остаточный член в форме Лагранжа, $R_n(\mathbf{x}) \xrightarrow{n\to\infty} 0$. (6)

3.2. Разложение в ряд Тейлора СНУ

Обозначим за $\hat{\mathbf{x}}$ искомое решение, за $\mathbf{x}^{[k]}$ решение на k-м шаге и введем ещё обозначение смещения $\Delta\mathbf{x}^{[k]} = \hat{\mathbf{x}} - \mathbf{x}^{[k]}$.

Разложим $F_i(\mathbf{x})$ в окрестности точки $\mathbf{x}^{[k]}$ в ряд Тейлора по формуле (5):

$$F_i(\mathbf{x}) = F_i\left(\mathbf{x}^{[k]}\right) + \frac{\partial F_i\left(\mathbf{x}^{[k]}\right)}{\partial x_1}\left(x_1 - x_1^{[k]}\right) + \frac{\partial F_i\left(\mathbf{x}^{[k]}\right)}{\partial x_2}\left(x_2 - x_2^{[k]}\right) + R_2(\mathbf{x})$$

Подставив в это разложение точку $\hat{\mathbf{x}}$ получаем следующее:

$$F_i(\hat{\mathbf{x}}) = F_i\left(\mathbf{x}^{[k]}\right) + \frac{\partial F_i\left(\mathbf{x}^{[k]}\right)}{\partial x_1}\Delta x_1^{[k]} + \frac{\partial F_i\left(\mathbf{x}^{[k]}\right)}{\partial x_2}\Delta x_2^{[k]} + R_2(\hat{\mathbf{x}})$$

Теперь запишем (3) с учетом предыдущего разложения:

$$\begin{cases} F_1\left(\mathbf{x}^{[k]}\right) + \dfrac{\partial F_1\left(\mathbf{x}^{[k]}\right)}{\partial x_1}\Delta x_1^{[k]} + \dfrac{\partial F_1\left(\mathbf{x}^{[k]}\right)}{\partial x_2}\Delta x_2^{[k]} + R_2^{F_1}(\hat{\mathbf{x}}) = 0 \\ F_2\left(\mathbf{x}^{[k]}\right) + \dfrac{\partial F_2\left(\mathbf{x}^{[k]}\right)}{\partial x_1}\Delta x_1^{[k]} + \dfrac{\partial F_2\left(\mathbf{x}^{[k]}\right)}{\partial x_2}\Delta x_2^{[k]} + R_2^{F_2}(\hat{\mathbf{x}}) = 0 \end{cases}$$

Или в матричном виде:

$$\mathbf{F}\left(\mathbf{x}^{[k]}\right) + \begin{pmatrix} \dfrac{\partial F_1\left(\mathbf{x}^{[k]}\right)}{\partial x_1} & \dfrac{\partial F_1\left(\mathbf{x}^{[k]}\right)}{\partial x_2} \\ \dfrac{\partial F_2\left(\mathbf{x}^{[k]}\right)}{\partial x_1} & \dfrac{\partial F_2\left(\mathbf{x}^{[k]}\right)}{\partial x_2} \end{pmatrix} \cdot \Delta\mathbf{x}^{[k]} + \begin{pmatrix} R_2^{F_1}(\hat{\mathbf{x}}) \\ R_2^{F_2}(\hat{\mathbf{x}}) \end{pmatrix} = \mathbf{0}$$

$$\mathbf{F}\left(\mathbf{x}^{[k]}\right) + \mathbf{J_F}\left(\mathbf{x}^{[k]}\right) \cdot \Delta\mathbf{x}^{[k]} + \mathbf{R}_2(\hat{\mathbf{x}}) = \mathbf{0}$$

$$\mathbf{J_F}\left(\mathbf{x}^{[k]}\right) \cdot \Delta\mathbf{x}^{[k]} = -\mathbf{F}\left(\mathbf{x}^{[k]}\right) - \mathbf{R}_2(\hat{\mathbf{x}})$$

Где $\mathbf{J_F}(\mathbf{x})$ — матрица Якоби функции $\mathbf{F}$ в точке $\mathbf{x}$.

Из этого уравнения можно найти $\Delta\mathbf{x}^{[k]}$, который прибавляется к текущему решению $\mathbf{x}^{[k]}$ чтобы найти $\hat{\mathbf{x}}$.

Мы не не знаем как вычислять $\mathbf{R}_2(\hat{\mathbf{x}})$, но можем им пренебречь, так как он стремится к нулю. Именно из-за этого принебрежения мы не сразу получаем точное решение, а процесс становится итеративным. Итог:

$$\mathbf{J_F}\left(\mathbf{x}^{[k]}\right) \cdot \Delta\mathbf{x}^{[k]} = -\mathbf{F}\left(\mathbf{x}^{[k]}\right) \tag{7}$$

Можно заметить, что это СЛАУ. Её можно легко решить известными методами.

4. Алгоритм

5. Стандартный метод Ньютона

Весь метод заключается в итерационном процессе:

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} + \Delta\mathbf{x}^{[k]} \tag{8}$$

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} - \left(\mathbf{J_F}\left(\mathbf{x}^{[k]}\right)\right)^{-1}\mathbf{F}\left(\mathbf{x}^{[k]}\right) \tag{9}$$

Далее за k обозначается количество итераций; за m максимальное число итераций; за ε_1 максимальная допустимая невязка;

Algorithm 1: Метод Ньютона

Input: ε_1, m, $\mathbf{F}$, $\mathbf{x}^{[0]}$
Result: $\mathbf{x}$

```
1  begin
2      x ← x^[0]
3      k ← 0
4      while ||F(x)|| > ε_1 and k < m do
5          A ← J_F(x)
6          b ← −F(x)
7          Решить СЛАУ A · Δx = b, результат поместить в Δx
8          x ← x + Δx
9          k ← k + 1
10     end
11 end
```

Выход из итерационного процесса производится когда невязка достигает необходимой нижней границы, либо превышается максимальное число итераций.

В $||\mathbf{F}(\mathbf{x})|| > \varepsilon_1$ невязка считается через норму, потому что так проще сравнить вектор с одним числом. Норма выбирается по вашему усмотрению.

Замечание: данный алгоритм не зависит от размерности системы, а, следовательно, может быть применен к любой размерности исходной СНУ.

Замечание: в алгоритме используется матрица Якоби в известной точке, что означает, что знать её аналитическое выражение не нужно, её можно вычислять при помощи численного взятия производной в конкретной точке.

5.1. Улучшенный метод Ньютона

Иногда может получаться такой $\Delta\mathbf{x}$, что невязка в векторе $\mathbf{x}^{[k+1]}$ будет больше, чем в $\mathbf{x}^{[k]}$, поэтому вводится коэффициент $\beta \in [0, 1]$, который домножается на $\Delta\mathbf{x}$, причем β подбирается такой, чтобы невязка строго уменьшалась. Тогда (**??**) будет выглядеть следующим образом:

$$\mathbf{x}^{[k+1]} = \mathbf{x}^{[k]} + \beta \cdot \Delta\mathbf{x}^{[k]}$$

Чтобы найти такой β, при котором невязка будет уменьшаться, используется аналог двоичного поиска: изначально $\beta = 1$, затем, пока невязка нового решения больше, чем предыдущего, β уменьшается вдвое.

Так же, если $\beta \to 0$, то это означает, что невязка не может быть уменьшена на текущей итерации, и это ещё один повод завершить итерационный процесс.

Algorithm 2: Улучшенный Метод Ньютона

Input: ε_1, ε_2, m, $\mathbf{F}$, $\mathbf{x}^{[0]}$
Result: $\mathbf{x}$

begin
  $\mathbf{x} \leftarrow \mathbf{x}^{[0]}$
  $\beta \leftarrow 1$
  $k \leftarrow 0$
  while $||\mathbf{F}(\mathbf{x})|| > \varepsilon_1$ **and** $\beta > \varepsilon_2$ **and** $k < m$ **do**
    $\mathbf{A} \leftarrow \mathbf{J}_{\mathbf{F}}(\mathbf{x})$
    $\mathbf{b} \leftarrow -\mathbf{F}(\mathbf{x})$
    Решить СЛАУ $\mathbf{A} \cdot \Delta\mathbf{x} = \mathbf{b}$, результат поместить в $\Delta\mathbf{x}$
    $\beta \leftarrow 1$
    while $||\mathbf{F}(\mathbf{x} + \beta\Delta\mathbf{x})|| > ||\mathbf{F}(\mathbf{x})||$ **and** $\beta > \varepsilon_2$ **do**
      $\beta \leftarrow 0.5\beta$
    end
    $\mathbf{x} \leftarrow \mathbf{x}^{[k]} + \beta \cdot \Delta\mathbf{x}$
    $k \leftarrow k + 1$
  end
end

5.2. Одномерный случай

В одномерном случае (8) преобразуется к:

$$x^{[k+1]} = x^{[k]} - \left(\frac{df\left(x^{[k]}\right)}{dx}\right)^{-1} f\left(x^{[k]}\right) = x^{[k]} - \frac{f\left(x^{[k]}\right)}{f'\left(x^{[k]}\right)}$$

6. Пример

Например, у нас имеется следующая СНУ:

$$\begin{cases} (x-x_1)^2+(y-y_1)^2=(r_1+r)^2 \\ (x-x_2)^2+(y-y_2)^2=(r_1+r)^2 \\ \dfrac{|ax+by+c|}{\sqrt{a^2+b^2}}=r \end{cases}$$

Решением этой СНУ является окружность с центром в точке $(x,\ y)$ и радиусом r, которая касается внешним образом окружностей $(x_1,\ y_1,\ r_1)$, $(x_2,\ y_2,\ r_2)$ и прямой $ax+by+c=0$.

Для этой системы получим:

$$\mathbf{F}=\begin{pmatrix} (x-x_1)^2+(y-y_1)^2-(r_1+r)^2 \\ (x-x_2)^2+(y-y_2)^2-(r_1+r)^2 \\ \dfrac{|ax+by+c|}{\sqrt{a^2+b^2}}-r \end{pmatrix},\ \mathbf{J_F}=\begin{pmatrix} 2(x-x_1) & 2(y-y_1) & -2(r_1+r) \\ 2(x-x_2) & 2(y-y_2) & -2(r_2+r) \\ \dfrac{|a|}{\sqrt{a^2+b^2}} & \dfrac{|b|}{\sqrt{a^2+b^2}} & -1 \end{pmatrix}$$

Далее остается только применить алгоритм.

7. Когда размерность СНУ не соответствует размерности решения

Пусть дана СНУ:

$$\begin{cases} F_1(x_1,\ x_2,\ \ldots,\ x_n)=0 \\ F_2(x_1,\ x_2,\ \ldots,\ x_n)=0 \\ \ldots \\ F_m(x_1,\ x_2,\ \ldots,\ x_n)=0 \end{cases}$$

Вышеописанный алгоритм работает только когда $m=n$, в случае же $m\neq n$ есть несколько методов приведения к системы квадратному виду.

Следующие методы показывают как надо модифицировать СНУ для конкретной точки, поэтому не получится их применить для того, чтобы один раз вычислить новую СНУ, её придётся считать на каждой итерации заного.

Новая СНУ и соответствующая ей матрица Якоби формируется только для поиска $\Delta\mathbf{x}$ при решении СЛАУ, для расчета невязки используется изначальная СНУ $\mathbf{F}$.

7.1. Расширенный метод Ньютона для $m \neq n$

Algorithm 3: Расширенный Метод Ньютона для $m \neq n$

Input: ε_1, ε_2, m, $\mathbf{F}$, $\mathbf{x}^{[0]}$, $\mathbb{S}$, $\mathbb{J}$
Result: $\mathbf{x}^{[k]}$

```
begin
    x^[k] ← x^[0]
    β ← 1
    k ← 0
    while ||F(x)|| > ε1 u β > ε2 u k < m do
        G ← S(F, x^[k])
        I ← J(F, G, x^[k])
        A ← I(x^[k])
        b ← −G(x^[k])
        Решить СЛАУ A · x = b, результат поместить в x
        β ← 1
        while ||F(x^[k] + βx)|| > ||F(x^[k])|| u β > ε2 do
            β ← 0.5β
        end
        x^[k] ← x^[k] + βx
        k ← k + 1
    end
end
```

$\mathbb{S}$ — это функция, преобразующая $\mathbf{F}$ к размерности n в конкретной точке. О видах этой функции будет написано далее. В случае, когда размерность СНУ совпадает с размерностью решения, $\mathbb{S}(\mathbf{F},\ \mathbf{x}) = \mathbf{F}$.

Аналогично $\mathbb{J}$ — функция возвращающая матрицу Якоби для решения СЛАУ в конкретной точке. Если не указано, как её вычислять, то $\mathbb{J}(\mathbf{F},\ \mathbf{G},\ \mathbf{x}) = \mathbf{J}_{\mathbf{G}}$.

Причем модификации затрагивают только часть, относящуюся к решению СЛАУ, но не затрагивают часть для вычисления нормы (невязки). Так сделано, потому что всё же решается функция $\mathbf{F}$, а не $\mathbf{G}$.

7.2. $m < n$

Это означает, что решений, где СНУ оборачивается в ноль, множество.

Так же это означает, что не хватает уравнений, для того, чтобы построить полный вектор смещения $\Delta\mathbf{x}$. Поэтому делается так, чтобы в итоге некоторые компоненты $\Delta\mathbf{x}$ были равны нулю, предполагается что по этим компонентам уже найдено достаточно хорошее решение, и искать для них смещения не нужно.

Какие именно компоненты занулять — выбирается таким образом, чтобы подтверждалось вышеописанное предположение. В данном случае они выбирается по значениям частных производных, то есть для каждой компоненты под номером j находится число $y_j = \max_i \left|\frac{\partial F_i(\mathbf{x})}{\partial x_j}\right|$, и зануляются те $(n-m)$ компонент, для которых y_j минимально.

Реализовать это можно двумя способами: либо уменьшать матрицу Якоби и вектор $\mathbf{F}(\mathbf{x})$ до размерности m, и затем подставлять найденные решения в необходимые компоненты вектора $\Delta\mathbf{x}$, который имеет размерность n; либо до-

бавлять к исходной СНУ уравнения вида $x_j - a_j = 0$ к системе, которые сделают так, что при решении СЛАУ в точке $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ вектор $\Delta\mathbf{x}$ в необходимых компонентах будет иметь нули.

Следующий алгоритм описывает именно второй вариант.

Новая СНУ $\mathbf{G}(\mathbf{x})$ строится по алгоритму:

Algorithm 4: $m < n,\ \mathbb{S}_1(\mathbf{F},\ \mathbf{a})$

Data: $\mathbf{F}$, $\mathbf{a}$
Result: $\mathbf{G}$

begin
  $\mathbf{A} \leftarrow \mathbf{J_F}(\mathbf{a})$
  $\mathbf{y} \leftarrow \left(\max_i |A_{i,1}|,\ \max_i |A_{i,2}|,\ \ldots,\ \max_i |A_{i,n}|\right)$
  Находятся такие i_j, что: $y_{i_1} \geqslant y_{i_2} \geqslant \cdots \geqslant y_{i_n}$
  for $j \leftarrow 1$ **to** $n - m$ **do**
    $G_j(\mathbf{x}) \leftarrow x_{i_j} - a_{i_j}$
  end
  for $j \leftarrow n - m + 1$ **to** n **do**
    $G_j(\mathbf{x}) \leftarrow F_{i_j}(\mathbf{x})$
  end
end

Обратите внимание, что в итоге $\mathbf{G}(\mathbf{x})$ является функцией $\mathbf{G} : \mathbb{R}^n \to \mathbb{R}^n$ и для каждой различной точки $\mathbf{a}$ функция $\mathbf{G}$ различна.

Такое преобразование делается для того, чтобы при нахождении $\Delta\mathbf{x}$ компоненты с минимальными частными производными были равны нулю.

7.3. $m > n$

Это означает, что решений может не быть, в данном случае метод Ньютона находит точку с минимальной невязкой.

Так же это означает, что уравнений слишком много, чтобы найти вектор смещения $\Delta\mathbf{x}$, поэтому придумываются методы, чтобы либо исключить какие-то уравнения, либо скомбинировать их каким-либо образом. Причем методы должны делать это так, чтобы не потерять всю информацию о исключаемых уравнениях.

7.3.1. Метод исключения

Из системы исключаются $(n-m)$ уравнений, для которых $F_i(\mathbf{x})$ минимальны.

Algorithm 5: $m > n,\ \mathbb{S}_2\,(\mathbf{F},\ \mathbf{a})$

Data: **F**, **a**
Result: **G**
begin
  $\mathbf{y} \leftarrow (|F_1(\mathbf{x})|,\ |F_2(\mathbf{x})|,\ \ldots,\ |F_m(\mathbf{x})|)$
  Находятся такие i_j, что: $y_{i_1} \geqslant y_{i_2} \geqslant \cdots \geqslant y_{i_m}$
  for $j \leftarrow 1$ **to** $n - m$ **do**
    $G_j(\mathbf{x}) \leftarrow F_{i_j}(\mathbf{x})$
  end
end

7.3.2. Метод свертки

Для всех лишних $(n - m + 1)$ уравнений у которых $F_i(\mathbf{x})$ минимальны, проводится свертка. Вместо исключаемых уравнений берется одно уравнение с суммой квадратов всех исключаемых уравнений.

Algorithm 6: $m > n,\ \mathbb{S}_3\,(\mathbf{F},\ \mathbf{a})$

Data: **F**, **a**
Result: **G**
begin
  $\mathbf{y} \leftarrow (|F_1(\mathbf{x})|,\ |F_2(\mathbf{x})|,\ \ldots,\ |F_m(\mathbf{x})|)$
  Находятся такие i_j, что: $y_{i_1} \geqslant y_{i_2} \geqslant \cdots \geqslant y_{i_m}$
  for $j \leftarrow 1$ **to** $n - m - 1$ **do**
    $G_j(\mathbf{x}) \leftarrow F_{i_j}(\mathbf{x})$
  end
  $G_n \leftarrow \sum_{j=n-m}^{n} F_{i_j}^2(\mathbf{x})$
end

7.3.3. Процедура симметризации

Вместо исходной системы должна решаться система:

$$\mathbb{S}_4\left(\mathbf{F},\ \mathbf{x}^{[k]}\right) = \left(\mathbf{J}_{\mathbf{F}}\left(\mathbf{x}^{[k]}\right)\right)^T \cdot \mathbf{F}\left(\mathbf{x}^{[k]}\right)$$

$$\mathbb{J}_4\left(\mathbf{F},\ \mathbf{G},\ \mathbf{x}^{[k]}\right) = \left(\mathbf{J}_{\mathbf{F}}\left(\mathbf{x}^{[k]}\right)\right)^T \cdot \mathbf{J}_{\mathbf{F}}\left(\mathbf{x}^{[k]}\right)$$

Замечание: только ради этой процедуры в аргументах $\mathbb{J}$ имеется **F**, и в принципе только ради этой процедуры введена функция $\mathbb{J}$. Причем **G** здесь не используется.