НАХОЖДЕНИЕ СТАЦИОНАРНЫХ ТОЧЕК В ЗАДАЧАХ БЕЗУСЛОВНОЙ ОПТИМИЗАЦИИ В MATLAB*

М. А. Кольцов

kolmax94@gmail.com

17 декабря 2015 г.

1°. Постановка задачи. Будем рассматривать *задачу безусловной оптимизации*:

$$F(x) \to \operatorname*{extr}_{x \in \mathbb{R}^n} \tag{1}$$

Если $F \in C^1(\mathbb{R}^n)$, то можно записать необходимое условие экстремума: если $x^* \in \mathbb{R}^n$ — решение задачи (1), то

$$f := F'(x^*) = \mathbb{O} \tag{2}$$

Все точки из $\mathbb{R}^n$, удовлетворяющие условию (2), называются *стационарными точками*.

В дальнейшем мы покажем, как искать стационарные точки с помощью MATLAB.

2°. Функция `fsolve`. В среде MATLAB имеется функция `fsolve`, предназначенная для решения систем нелинейных уравнений. Формат её вызова таков:

```
[x, f, exit] = fsolve(fun, x0, options)
```

Опишем её входные и выходные параметры:

- `fun` — правая часть системы уравнений, в нашем случае — $f = F'$. Может быть ссылкой на функцию MATLAB («`@f`») или анонимной функцией («`@(x) cos(x)`»). Функция должна принимать на вход аргумент `x` и возвращать число — значение $f(x)$
- `x0` — начальное приближение к решению

*Семинар по конструктивному негладкому анализу и недифференцируемой оптимизации «CNSA & NDO»: `http://www.apmath.spbu.ru/cnsa/`

- `x` — выходной параметр: решение уравнение
- `f` — выходной параметр: значение $f(x)$
- `exit` — выходной параметр: флаг, сигнализирующий о результатах вычислений

Значение флага `exit` могут быть такими:

- 1 — решение найдено
- 2 — изменение x слишком мало
- 3 — изменение функции слишком мало
- 4 — вычисленное направление поиска слишком мало
- 0 — превышено число итераций
- -2 — метод сходится к точке, не являющейся корнем

Также функции `fsolve` можно передать третий параметр — `options` — в котором можно задать дополнительные настройки работы алгоритма. Значение этого параметра следует получать с помощью стандартной функции `optimset` следующим образом:

```
options = optimset(Опция 1, Значение 1, Опция 2, Значение 2, ...)
```

Приведём основные возможные настройки:

- `Algorithm` — используемый алгоритм. Может быть «`trust-region-dogleg`» (по умолчанию), «`trust-region-reflective`» или «`levenberg-marquardt`»
- `TolFun` — чувствительность к изменению значения функции (по умолчанию 10^{-6})
- `TolX` — чувствительность к изменению значения x (по умолчанию 10^{-6})
- `MaxIter` — максимальное число итераций (по умолчанию 400)
- `Jacobian` — использовать (`'on'` или `'off'`) второй выходной параметр функции `fun`: якобиан f в точке x. Если целевая функция не предоставляет якобиан, то `fsolve` приближает его конечными разностями (по умолчанию `'off'`). В нашем случае якобиан f является матрицей вторых производных целевой функции F

Настройки чувствительности означают, что как только соответствующая величина изменится меньше, чем на порог чувствительности, алгоритм закончит свою работу. Этому соответствуют случаи 2 и 3 значений флага `exit`.

3°. Пример использования `fsolve`: обобщённая трёхдиагональная функция. Обратимся к банку тестовых функций [1] и испробуем MATLAB на обобщённой трёхдиагональной функции (Generalized Tridiagonal Function):

$$F(x) = \sum_{i=1}^{n-1} (x_i + x_{i+1} - 3)^2 + (x_i - x_{i+1} + 1)^4$$

Градиент этой функции и матрица вторых производных вычисляются так:

$$\begin{aligned}
\frac{\partial F}{\partial x_1} &= 2(x_1 + x_2 - 3) + 4(x_1 - x_2 + 1)^3 \\
\frac{\partial F}{\partial x_i} &= 2(x_i + x_{i+1} - 3) + 4(x_i - x_{i+1} + 1)^3 \\
&\quad + 2(x_{i-1} + x_i - 3) - 4(x_{i-1} - x_i + 1)^3, \qquad 1 < i < n \\
\frac{\partial F}{\partial x_n} &= 2(x_{n-1} + x_n - 3) - 4(x_{n-1} - x_n + 1)^3 \\
\frac{\partial^2 F}{\partial x_1^2} &= 2 + 12(x_1 - x_2 + 1)^2 \\
\frac{\partial^2 F}{\partial x_i^2} &= 2 + 12(x_i - x_{i+1} + 1)^2 + 2 + 12(x_{i-1} - x_i + 1)^3, \qquad 1 < i < n \\
\frac{\partial^2 F}{\partial x_n^2} &= 2 + 12(x_{n-1} - x_n + 1)^2 \\
\frac{\partial^2 F}{\partial x_1 \partial x_{i-1}} &= 2 - 12(x_{i-1} - x_i + 1)^2, \qquad 1 < i \\
\frac{\partial^2 F}{\partial x_1 \partial x_{i+1}} &= 2 - 12(x_i - x_{i+1} + 1)^2, \qquad i < n
\end{aligned}$$

Пусть $n = 5$. В качестве начального приближения банк тестовых задач предлагает взять $x_0 = [2, \ldots, 2]$. Запустим `fsolve` с такими параметрами: `TolFun = TolX = 1e-10`, `Jacobian = on` и за 6 шагов получим ответ: $x \approx [1.0368, 1.3699, 1.5000, 1.6301, 1.9632]$, $F(x) \approx 2.2787$ с комментарием

```
Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the selected value of the function tolerance, and
the problem appears regular as measured by the gradient.
```

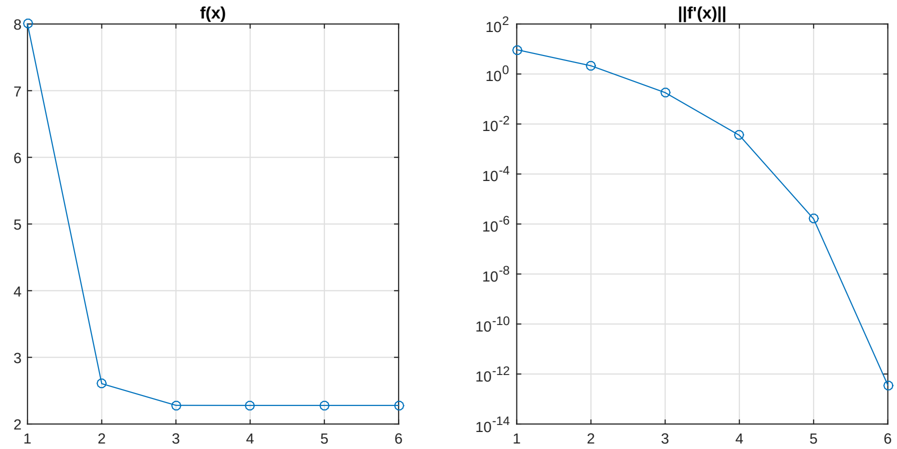

Рис. 1. Поведение функции Generalized Tridiagonal на каждом шаге

4°. Пример использования `fsolve`: функция EG2. Возьмём теперь из банка тестовых функций функцию EG2:

$$F(x) = \sum_{i=1}^{n-1} \sin(x_1 + x_i^2 - 1) + \frac{1}{2}\sin x_n^2$$

с рекомендованным начальным приближением $x_0 = [1, \ldots, 1]$.

Вычислим её градиент и матрицу вторых производных

$$\frac{\partial F}{\partial x_1} = (1 + 2x_1)\cos(x_1 + x_1^2 - 1) + \sum_{i=2}^{n-1} \cos(x_1 + x_i^2 - 1)$$

$$\frac{\partial F}{\partial x_i} = 2x_i \cos(x_1 + x_i^2 - 1), \qquad i \in 2 : (n-1)$$

$$\frac{\partial F}{\partial x_n} = x_n \cos(x_n)^2$$

$$\frac{\partial^2 F}{\partial x_1^2} = -(1 + 2x_1)^2 \sin(x_1 + x_1^2 - 1) + 2\cos(x_1 + x_1^2 - 1) - \sum_{i=1}^{n-1} \sin(x_1 + x_i^2 - 1)$$

$$\frac{\partial^2 F}{\partial x_i \partial x_1} = 2x_i \sin(x_1 + x_i^2 - 1)$$

$$\frac{\partial^2 F}{\partial x_i^2} = -4x_i^2 \sin(x_1 + x_i^2 - 1) + 2\cos(x_1 + x_i^2 - 1)$$

$$\frac{\partial^2 F}{\partial x_n^2} = -2x_n^2 \sin(x_n^2) + \cos(x_n^2)$$

и запустим `fsolve` при $n = 5$ и с теми же параметрами. За 7 шагов получаем ответ $x \approx [1.1795, 1.1795, 1.1795, 1.1795, 1.2533]$, $F(x) \approx 4.5$, комментарий MATLAB при этом не изменился.

Однако, оказывается, что в найденной точке x функция EG2 достигает локального *максимума*, в чем можно убедиться, посмотрев на график её значений (рис. 2) и посчитав собственные числа матрицы вторых производных — оказывается, что она там отрицательно определена, что означает точку локального максимума.

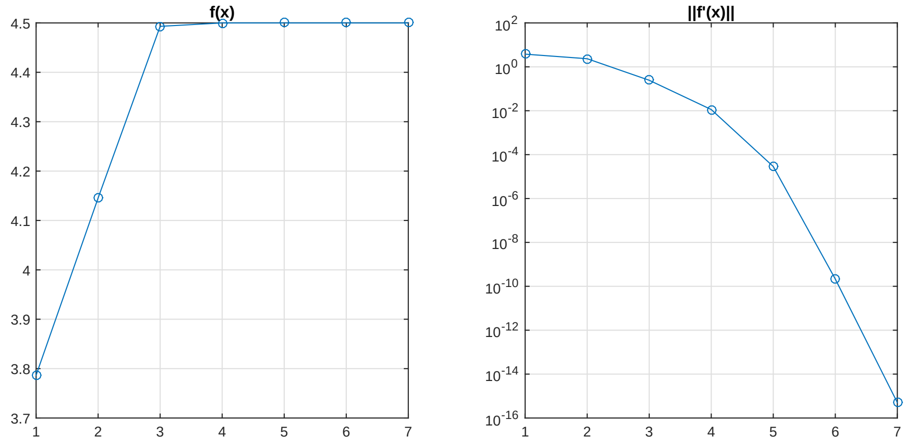

Рис. 2. Поведение функции EG2 на каждом шаге

Интересно заметить, что задача минимизации функция EG2 может быть решена аналитически (решение представлено Даниилом Быковым). Действительно, функция является суммой синусов. Если бы в какой-то точке x^* все синусы принимали значение -1, то это точка была бы точкой глобального минимума. Попробуем этого добиться.

Во-первых, ясно, что надо положить $x_n^* = \sqrt{\dfrac{3\pi}{2}}$, так как x_n не входит в другие слагаемые. Далее, минимизируем первое слагаемое:

$$\sin(x_1^* + x_1^{*2} - 1) = -1$$
$$x_1^* + x_1^{*2} - 1 = \frac{3\pi}{2}$$
$$x_1^* = \frac{-1 \pm \sqrt{1 + 4(1 + \frac{3\pi}{2})}}{2} = \frac{-1 \pm \sqrt{5 + 6\pi}}{2}$$

Теперь можно найти и остальные переменные:

$$x_i^* = \pm\sqrt{1 + \frac{3\pi}{2} - x_1^*}, \quad i \in 2 : (n-1)$$

В точке x^* значение функции равно $-n + \frac{1}{2}$.

ЛИТЕРАТУРА

1. Andrei N. *An unconstrained optimization test functions collection* // Advanced Modeling and Optimization. 2008. Vol. 10, No. 1, pp. 147–161.