НАХОЖДЕНИЕ СТАЦИОНАРНЫХ ТОЧЕК В ЗАДАЧАХ БЕЗУСЛОВНОЙ ОПТИМИЗАЦИИ В MATLAB*

М. А. Кольцов
kolmax94@gmail.com

А. В. Плоткин
avplotkin@gmail.com

24 ноября 2016 г.

1°. Постановка задачи. Рассмотрим *задачу безусловной оптимизации*

$$f(x) \to \underset{x \in \mathbb{R}^n}{\text{extr}},$$

где $f \in C^1(\mathbb{R}^n)$. Запишем необходимое условие экстремума в точке x_*:

$$g(x_*) := f'(x_*) = \mathbb{O}. \tag{1}$$

Все точки из $\mathbb{R}^n$, удовлетворяющие условию (1), называются *стационарными*. Система уравнений $g(x) = \mathbb{O}$, в общем случае, является нелинейной. Решению таких систем с помощью `MATLAB` и посвящён данный доклад.

2°. Функция fsolve. В среде `MATLAB` имеется функция `fsolve`, предназначенная для решения систем нелинейных уравнений вида $F(x) = \mathbb{O}$, $x \in \mathbb{R}^n$. Решение системы происходит путём минимизации невязки $G(x) = ||F(x)||^2$.

Функция `fsolve` имеет следующий формат вызова:

```
[xval, fval, exitflag] = fsolve(fun, x0, options)
```

Поясним входные параметры функции `fsolve`:

- `fun` — указатель на функцию $F(x)$, задающую левую часть решаемой системы уравнений $\big($в нашем случае — градиент $g(x)\big)$,
- `x0` — начальное приближение к решению,
- `options` — параметр, устанавливающий некоторые дополнительные настройки, в частности, алгоритм решения.

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

Значение параметра `options` задаётся с помощью стандартной функции `optimoptions` следующим образом:

```
options =
optimoptions(@fsolve, Опция 1, Значение 1, Опция 2, Значение 2, ...)
```

Приведём основные доступные настройки:

- ▷ `Algorithm` — используемый алгоритм решения. Возможные значения: «trust-region-dogleg» (по умолчанию), «trust-region», «levenberg-marquardt» (подробнее о данных алгоритмах см. [1, с. 730–731]).
- ▷ `OptimalityTolerance` — мера оптимальности точки (по умолчанию 10^{-6}). Если $||G'(x_k)||_\infty <$ `OptimalityTolerance`, то вычисления прекращаются.
- ▷ `StepTolerance` — минимальный допустимый размер шага (по умолчанию 10^{-6}).
- ▷ `FunctionTolerance` — минимальное допустимое уменьшение невязки $G(x)$ за одну итерацию (по умолчанию 10^{-6}).
- ▷ `MaxIterations` — максимальное число итераций (по умолчанию 400).
- ▷ `MaxFunctionEvaluations` — допустимое количество вычислений функции F (по умолчанию $100 \cdot n$).
- ▷ `SpecifyObjectiveGradient` — использовать или нет (`true` или `false`) второй выходной параметр функции `fun` — матрицу Якоби функции F в точке x. Если `false` (по умолчанию), то матрица приближается конечными разностями. В нашем случае матрицей Якоби является матрица вторых производных функции $f(x)$.
- ▷ `OutputFcn` — указатель на дополнительную функцию, которая будет выполняться после каждой итерации алгоритма.

Полный список доступных настроек приведён в руководстве [1, с. 722–728].

Перейдём к описанию выходных параметров функции `fsolve`:

- `xval` — найденное решение,
- `fval` — значение $F($`xval`$)$,
- `exitflag` — значение, сигнализирующее об окончании вычислений.

Поясним возможные значения r параметра `exitflag`:
$r > 0$ — конечное значение невязки мало $\left(G(\texttt{xval}) < \sqrt{\texttt{FunctionTolerance}}\right)$,
$r = 0$ — превышено максимальное число итераций или количество вычислений функции F (см. `MaxIterations` и `MaxFunctionEvaluations`),
$r < 0$ — конечное значение невязки велико $\left(G(\texttt{xval}) \geqslant \sqrt{\texttt{FunctionTolerance}}\right)$.

3°. Информация о процессе вычислений. Из описания выходных параметров функции ясно, что `fsolve` предоставляет информацию только о конечном значении x и $F(x)$. Однако зачастую бывает интересна информация о всём процессе вычислений (к примеру, для построения графиков). Для получения данной информации может быть использована опция `OutputFcn`.

Функция, указатель на которую передаётся в опцию `OutputFcn`, должна принимать от `fsolve` три аргумента:

- `x` — текущее значение x,
- `optimValues` — структура, содержащая информацию о текущей итерации,
- `state` — стадия выполнения `fsolve`.

Функция может вернуть «`true`», если требуется прекратить вычисления, и «`false`» в противном случае.

Приведём пример простейшей реализации такой функции:

```
function stop = outfun(x, optimValues, state)
  stop = false;                             % не прерываем вычисления
  if isequal(state, 'iter')                 % если конец итерации
    xhist = [xhist; x];                     % сохраняем значение x
    ghist = [ghist; optimValues.fval'];     % сохраняем значение g(x)
  end
end
```

По окончании работы `fsolve` в переменных `xhist`, `ghist` будут содержаться значения x и $g(x)$ на каждой итерации.

Перечень доступной в структуре `optimValues` информации, а также примеры более сложных функций приведены в руководстве [1, с. 166–172, 568–577].

4°. Пример 1: обобщённая трёхдиагональная функция. Обратимся к банку тестовых функций [2] и опробуем `fsolve` на обобщённой трёхдиагональной функции (Generalized Tridiagonal function):

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

Запишем градиент данной функции:

$$\begin{aligned}
\frac{\partial f}{\partial x_1} &= 2\,(x_1 + x_2 - 3) + 4\,(x_1 - x_2 + 1)^3, \\
\frac{\partial f}{\partial x_i} &= 2\,(x_i + x_{i+1} - 3) + 4\,(x_i - x_{i+1} + 1)^3 + \\
&\quad + 2\,(x_{i-1} + x_i - 3) - 4\,(x_{i-1} - x_i + 1)^3, \quad i \in 2 : n-1, \\
\frac{\partial f}{\partial x_n} &= 2\,(x_{n-1} + x_n - 3) - 4\,(x_{n-1} - x_n + 1)^3.
\end{aligned}$$

Пусть $n = 5$. Банк тестовых функций предлагает в качестве начального приближения $x_0 = [2, \ldots, 2]$. Запустим `fsolve` с этим начальным приближением и после 6 итераций получим ответ: $x_* \approx [1.037, 1.370, 1.500, 1.630, 1.963]$, $f(x_*) \approx 2.27875$ (см. рис. 1).

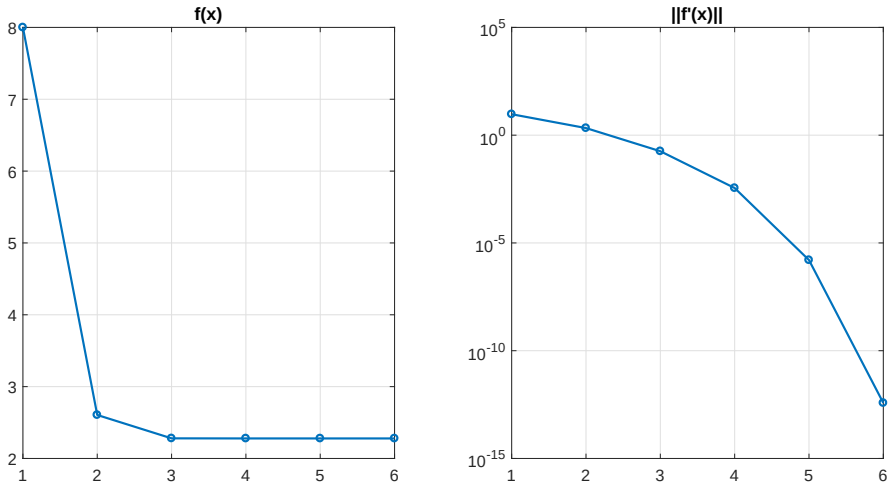

Рис. 1. Поведение обобщённой трёхдиагональной функции по итерациям

5°. Пример 2: функция EG2. Рассмотрим теперь функцию EG2 из того же банка тестовых функций:

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

Запишем градиент данной функции:

$$\frac{\partial f}{\partial x_1} = (1 + 2x_1) \cos(x_1 + x_1^2 - 1) + \sum_{i=2}^{n-1} \cos(x_1 + x_i^2 - 1),$$
$$\frac{\partial f}{\partial x_i} = 2\, x_i \cos(x_1 + x_i^2 - 1), \quad i \in 2 : n-1,$$
$$\frac{\partial f}{\partial x_n} = x_n \cos x_n^2.$$

Также вычислим матрицу вторых производных:

$$\frac{\partial^2 f}{\partial x_1^2} = -(1 + 2x_1)^2 \sin(x_1 + x_1^2 - 1) + 2\, \cos(x_1 + x_1^2 - 1) - \sum_{i=2}^{n-1} \sin(x_1 + x_i^2 - 1),$$
$$\frac{\partial^2 f}{\partial x_i^2} = -4\, x_i^2 \sin(x_1 + x_i^2 - 1) + 2\, \cos(x_1 + x_i^2 - 1), \quad i \in 2 : n-1,$$
$$\frac{\partial^2 f}{\partial x_n^2} = -2\, x_n^2 \sin x_n^2 + \cos x_n^2,$$
$$\frac{\partial^2 f}{\partial x_1 \partial x_i} = -2\, x_i \sin(x_1 + x_i^2 - 1), \quad i \in 2 : n-1.$$

Пусть $n = 5$. Банк тестовых функций предлагает в качестве начального приближения $x_0 = [1, \ldots, 1]$. Запустим `fsolve` с этим начальным приближением и следующими настройками: `SpecifyObjectiveGradient = true`, `OptimalityTolerance` $= 10^{-12}$. После 7 итераций получим ответ: $x_* \approx$ $\approx [1.180, 1.180, 1.180, 1.180, 1.253]$, $f(x_*) \approx 4.5$ (см. рис. 2).

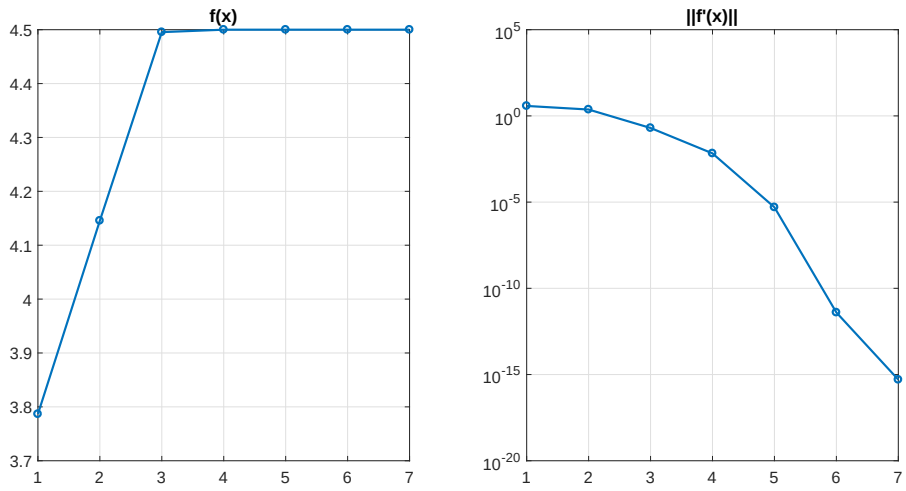

Рис. 2. Поведение функции `EG2` по итерациям

Найденное значение x_* является точкой *локального максимума* (матрица вторых производных отрицательно определена в этой точке).

Интересно заметить, что задача *минимизации* функции `EG2` может быть решена аналитически (решение представлено Даниилом Быковым). Действительно, функция является суммой синусов. Если бы в какой-то точке все синусы принимали значение -1, то эта 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 выходит в другие слагаемые. Далее, минимизируем первое слагаемое:

$$\begin{aligned}
&\sin(x_1 + x_1^2 - 1) = -1,\\
&x_1 + x_1^2 - 1 = \frac{3\pi}{2},\\
&x_1 = \frac{-1 \pm \sqrt{5 + 6\pi}}{2}.
\end{aligned}$$

Теперь можно найти остальные переменные:

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

Значение целевой функции в найденной точке равно $-n + \dfrac{1}{2}$.

6°. Пример 3: обобщённая функция Розенброка. Рассмотрим теперь обобщённую функцию Розенброка (Generalized Rosenbrock function):

$$f(x) = \sum_{i=1}^{n-1} \left[100\,(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right].$$

Запишем градиент данной функции:

$$\begin{aligned}
\frac{\partial f}{\partial x_1} &= -400\,x_1(x_2 - x_1^2) - 2\,(1 - x_1),\\
\frac{\partial f}{\partial x_i} &= -400\,x_i(x_{i+1} - x_i^2) - 2\,(1 - x_i) + 200\,(x_i - x_{i-1}^2), \quad i \in 2 : n-1,\\
\frac{\partial f}{\partial x_n} &= 200\,(x_n - x_{n-1}^2).
\end{aligned}$$

Частным случаем (при $n = 2$) является обычная функция Розенброка, которая подробно изучена. Опробуем `fsolve` сначала на этой двумерной функции. В качестве начального приближения возьмём $x_0 = [-1.2, 1]$. В ответ получим следующее сообщение от `MATLAB`:

```
fsolve stopped because it exceeded the function evaluation limit,
options.MaxFunctionEvaluations = 200 (the default value).
```

Поведение функции Розенброка по итерациям приведено на рис. 3 и рис. 4.

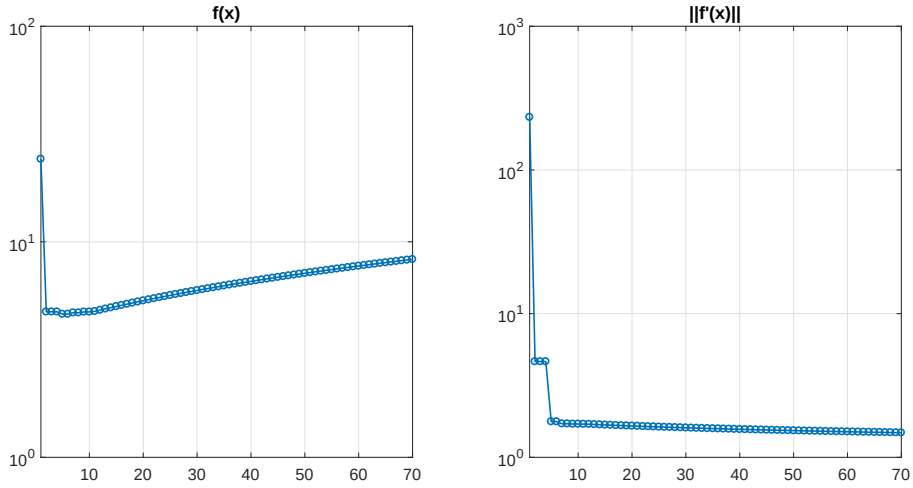

Рис. 3. Поведение функции Розенброка по итерациям при $x_0 = [-1.2, 1]$

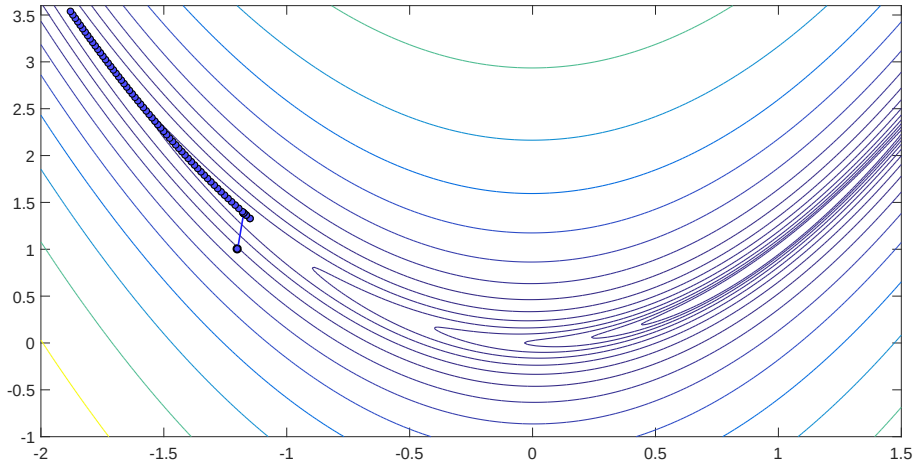

Рис. 4. Поведение аргумента по итерациям при $x_0 = [-1.2, 1]$

Увеличение параметра `MaxFunctionEvaluations` в данном случае не приводит к успеху.

Попробуем другое начальное приближение: $x_0 = [-0.5, 1.5]$. После 29 итераций получим ответ: $x_* \approx [1, 1]$, $f(x_*) \approx 0$ (см. рис. 5 и рис. 6).

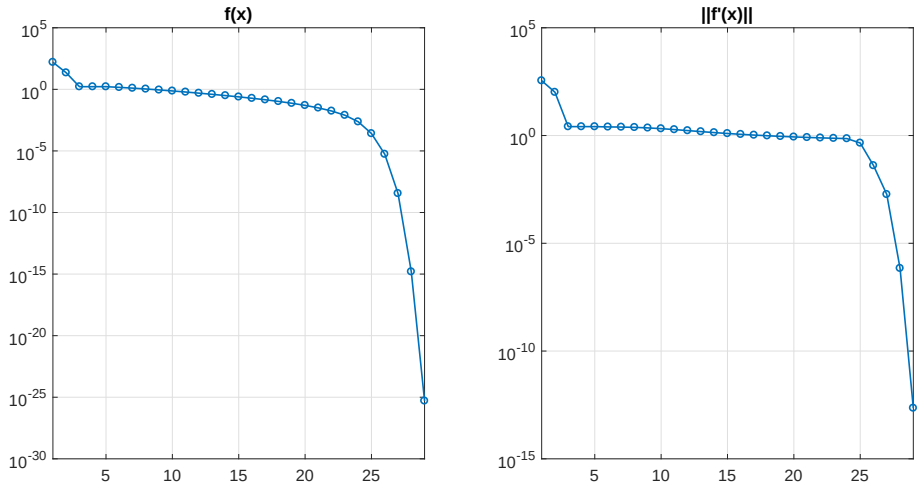

Рис. 5. Поведение функции Розенброка по итерациям при $x_0 = [-0.5, 1.5]$

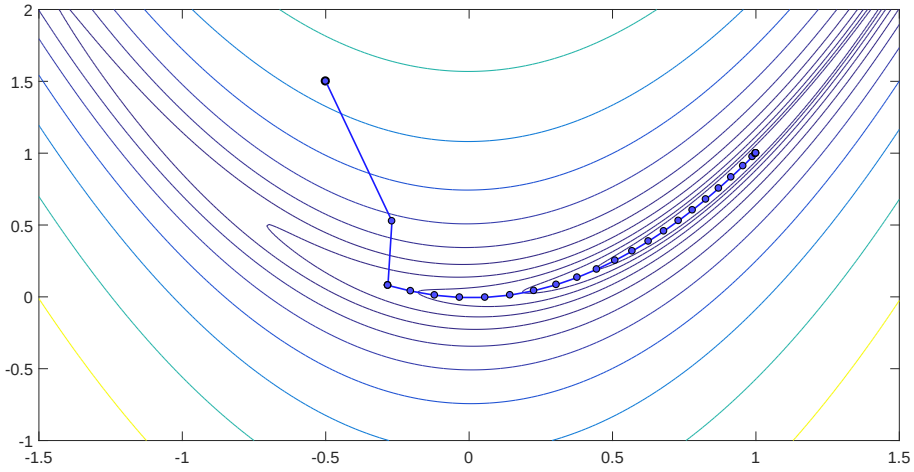

Рис. 6. Поведение аргумента по итерациям при $x_0 = [-0.5, 1.5]$

Найденное значение x_* является точкой глобального минимума функции Розенброка.

Вернёмся к обобщённой функции. Пусть $n = 5$. Банк тестовых функций предлагает в качестве начального приближения $x_0 = [-1.2, 1, -1.2, 1, -1.2]$.

Запустим `fsolve` с этим начальным приближением и после 43 итераций получим ответ: $x_* \approx [-0.962, 0.936, 0.881, 0.778, 0.605]$, $f(x_*) \approx 3.93084$ (см. рис. 7).

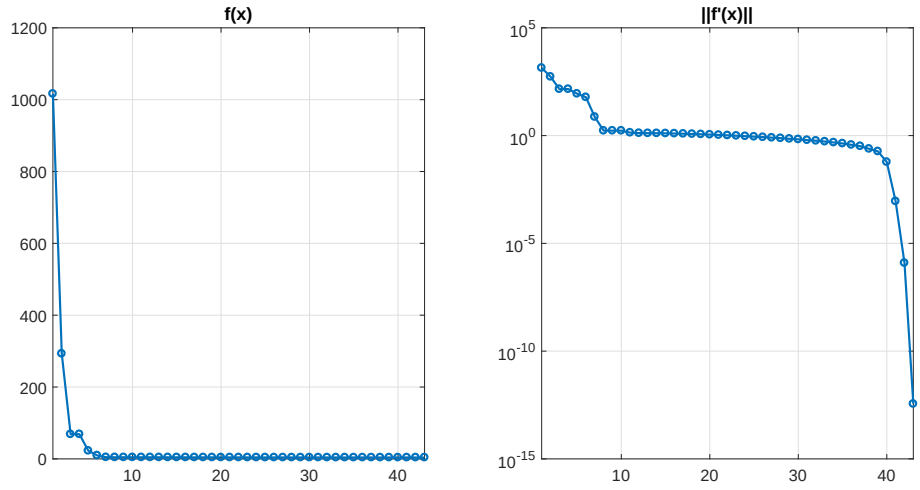

Рис. 7. Поведение обобщённой функции Розенброка по итерациям

Подробное исследование обобщённой функции Розенброка проведено в статье [3].

7°. Если `fsolve` не находит решение. Приведём несколько рекомендаций на случай, когда функция `fsolve` не смогла найти решение.

1) Попробуйте изменить начальное приближение. Задавая различные начальные приближения, вы увеличиваете шанс на успех.

2) Проверьте, является ли функция $F(x)$ гладкой — `fsolve` может плохо сходиться для функций, которые не являются гладкими.

3) Попробуйте изменить настройки точности, в особенности это касается параметров `OptimalityTolerance` и `StepTolerance`.

ЛИТЕРАТУРА

1. The MathWorks. *Optimization Toolbox User's Guide R2016b.* (`http://www.mathworks.com/help/pdf_doc/optim/optim_tb.pdf`)

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

3. Kok S., Sandrock C. *Locating and characterizing the stationary points of the extended rosenbrock function.* Evolutionary Computation. 2009. Vol. 17, No. 1, pp. 437–453.