S©MR ISSN 1813-3304

СИБИРСКИЕ ЭЛЕКТРОННЫЕ
МАТЕМАТИЧЕСКИЕ ИЗВЕСТИЯ

Siberian Electronic Mathematical Reports
http://semr.math.nsc.ru

*Том 16, стр. 144–144 (2019)* УДК 519.61
DOI 10.33048/semi.2019.16.xxx MSC 65H10

# ГЛОБАЛЬНОЕ РЕШЕНИЕ ПОКООРДИНАТНО МОНОТОННЫХ СИСТЕМ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Г.В. ГРЕНКИН

Abstract. The paper proposes an algorithm of finding all roots of a system of 3 nonlinear algebraic equations with 3 unknowns that has left-hand sides monotonic in any variable. The algorithm is based on the solution of systems of 2 equations with 2 unknowns at each iteration, and after that it chooses a root with certain index. Thus, the iterative process is subdivided into parts working independently.

**Keywords:** nonlinear equations, global search, numerical method.

## 1. Введение

Задачи решения систем нелинейных уравнений с покоординатно монотонными функциями встречаются при моделировании процессов, относящихся к распределению ресурсов в системе — зависимость количества ресурсов от мощности каждого источника является монотонной. Постановка задачи мотивирована исследованием вопроса единственности решения обратной задачи для уравнений сложного теплообмена, состоящей в отыскании неизвестных интенсивностей источников тепла по данным суммарного количества энергии в области каждого источника либо в каком-то другом наборе подобластей. Требуется решить систему

$$F_j(x_1, x_2, \ldots, x_n) = 0, \quad j = 1, \ldots, n \tag{1}$$

с монотонными функциями $F_j$, вычисляя функции в серии точек $\mathbf{x}^k$. При этом в вычислительный алгоритм должна быть заложена такая логика, которая позволит гарантированно найти все корни системы.

Grenkin, G.V., Global solution of coordinate-wise monotonic systems of nonlinear equations.



*Поступила 1 января 2015 г., опубликована 31 декабря 2015 г.*

Методы решения систем нелинейных уравнений можно разделить на стохастические и детерминированные, причем последние могут гарантировать нахождение корня в выделенной области. Поскольку мы стремимся найти все корни, то отдаем предпочтение детерминированным алгоритмам. Например, в случае монотонного отображения метод Ньютона дает гарантированную монотонную сходимость к единственному корню системы [1], поэтому одним из способов построения алгоритма может служить построение последовательности монотонных приближений, что, однако, работает в двумерном случае, но уже в трехмерном случае затруднительно даже для частично (покоординатно) монотонного отображения. Другой подход состоит в сведении системы уравнений к задаче наименьших квадратов [2]. Наконец, интервальные методы [3] заключаются в последовательном уменьшении области, в которой может находиться корень системы. В работе [4] для полного решения кубической системы $3 \times 3$ применялась нейронная сеть. В [5] корни нелинейной системы находятся последовательно друг за другом при помощи локального поиска.

В настоящей работе предлагается алгоритм, основанный на следующей идее. Для решения системы из $n$ уравнений с $n$ неизвестными предлагается проделать такой круг: зафиксировать по очереди каждую переменную и на каждом шаге улучшить текущее решение переопределенной системы из $n$ уравнений с $(n-1)$ неизвестными — для этого запустить еще один, внутренний, круг: выбрать $(n-1)$ уравнений из $n$ и найти решение этой полученной системы после одного круга такого же $(n-1)$-мерного алгоритма.

## 2. Численный метод

Обсудим гарантирующие полный поиск способы решения системы (1) при $n = 3$. Один из способов решения задачи — метод ветвей и границ [3, с. 307], который берет прямоугольную область, разбивает ее на 8 частей и выбрасывает из рассмотрения те из них, на которых хотя бы одна из функций $F_j$ сохраняет знак. По-видимому, данный алгоритм потребует слишком много действий за счет большого числа не отбрасываемых частей.

Еще один способ, претендующий на надежность — метод наименьших квадратов, который сводит решение системы к минимизации целевой функции $J(\mathbf{x}) = \sum\limits_{j=1}^{n} F_j^2(\mathbf{x})$. Можно запустить из всех углов большого параллелепипеда алгоритм градиентного спуска и затем вырезать из этого параллелепипеда меньший параллелепипед, углами которого являются начальная и конечная точки градиентного метода. Это связано с предположением, что конечная точка с большей разницей в значении целевой функции должна находиться на большем расстоянии от начальной точки — если бы внутри параллелепипеда находилась точка с меньшим значением целевой функции, то расстояние до нее от угла параллелепипеда ожидалось бы больше, чем до дальнего угла параллелепипеда, а это не так. Итак, недостаток этого метода — некоторая неуверенность в нахождении всех корней.

Для решения системы из трех уравнений с покоординатно монотонными функциями трех переменных предлагается следующий метод. Зафиксируем начальное приближение $(x_0, y_0, z_0)$ и проведем через эту точку плоскости, параллельные координатным плоскостям, и рассмотрим линии нулевого уровня $\{F_j = 0\}$ в сечениях $xy$, $yz$ и $zx$. В каждом таком сечении найдем одну из

координат как одну из точек попарного пересечения $i$-й и $j$-й линий нулевого уровня, причем если таких точек несколько, возьмем $k$-ю из них. Таким образом, алгоритм распределяется на несколько нитей, каждая из которых задается девяткой индексов $(i_{xy}, j_{xy}, k_{xy}, i_{yz}, j_{yz}, k_{yz}, i_{zx}, j_{zx}, k_{zx})$ и работает так: зафиксируем поочередно все допустимые значения индексов и для каждой выбранной девятки обновим координаты $x$ (при заданных $y, z$), $y$ (при новом $x$ и заданном $z$) и $z$ (при новых $x, y$), выбрав $k$-ю по порядку точку пересечения линий $\{F_i = 0\}$ и $\{F_j = 0\}$. Ожидается, что в некоторых нитях алгоритм сойдется к корню системы. Начальные нити задаются девятками $(i_1, j_1, 1, i_2, j_2, 1, i_3, j_3, 1)$, где $i_s, j_s \in \{1, 2, 3\}$, $i_s \neq j_s$. Любая нить может создать новую нить, если число точек пересечения в каком-нибудь сечении превзойдет текущее число значений индексов. Отметим, что в частном случае линейной системы точек попарного пересечения в любом сечении будет 3, поэтому новые нити порождаться не будут. В нелинейном случае алгоритм будет ветвиться, но все ветвления будут сгруппированы по номеру точки пересечения линий нулевого уровня. Таким образом, алгоритм генерирует не дерево, а несколько (проиндексированных) последовательностей приближений.

Двумерная задача нахождения всех точек пересечения линий уровня двух монотонных функций $f_1(x, y)$, $f_2(x, y)$ решается методом переменных направлений: при фиксированном $x$ находим $y$ (методом бисекций), при котором $f_1(x, y) = 0$ либо $f_2(x, y) = 0$. Далее аналогичным образом находим $x$. Выбор функции зависит от знаков $f_1$ и $f_2$ в текущем приближении и от выбранного направления движения — $x \uparrow y \downarrow$ либо $x \downarrow y \uparrow$). Таким образом, генерируется монотонная последовательность приближений, и итерационный процесс останавливается, когда по крайней мере одна из координат превзойдет по модулю заданное достаточно большое число.

## 3. Примеры

Описанный алгоритм реализован в среде Octave, с программной реализацией можно ознакомиться по ссылке https://github.com/lapkin25/nonlinear-monot.

В качестве примера рассмотрим линейную систему с левыми частями:

$$F_1(x, y, z) = 3x + 1.5y + z - 10,$$
$$F_2(x, y, z) = 0.5x + 3y + z - 15,$$
$$F_3(x, y, z) = 2x + 0.2y + z - 10.$$

Алгоритм сгенерировал 27 последовательностей приближений, и 11 из них сходятся к корню. Таблица 1 содержит значения минимума по всем нитям величины невязки $\max\limits_j |F_j|$ на каждой итерации алгоритма. Сходимость имеет скорость геометрической прогрессии: величина невязки на каждой итерации уменьшается более чем вдвое.

Далее рассмотрим пример нелинейной системы с левыми частями:

$$F_1(x, y, z) = 3x + 1.5y + z - 10,$$
$$F_2(x, y, z) = 0.5x + 3y + z - 15,$$
$$F_3(x, y, z) = 15((x + 10)^2 \cdot \operatorname{sign}(x + 10)/100 + (y + 10)^2 \cdot \operatorname{sign}(y + 10)/100) + z - 50.$$

| Номер итерации | Невязка |
|---|---|
| 1 | 3.864 |
| 2 | 1.480 |
| 3 | 0.567 |
| 4 | 0.143 |
| 5 | 0.024 |
| 6 | 0.002 |

Таблица 1. Динамика изменения невязки.

На рис. 1 изображены поверхности нулевого уровня функций $F_j$. Можно видеть, что все три поверхности пересекаются одновременно в трех точках: $(0.72, 4.54, 0.99)$, $(-8.08, -10.13, 49.45)$ и $(-18.49, -27.49, 106.72)$. За 5 итераций алгоритм при заданных параметрах точности и начальном приближении $(0, 0, -20)$ нашел лишь первый из перечисленных корней, в то время как за 7 итераций алгоритм нашел 3 корня и в процессе вычислений создал 185 нитей, в 4 из которых невязка оказалась меньше 0.01.

В таблице 2 представлена статистика сходимости алгоритма. В столбце «Невязка» перечислены наилучшие значения невязки для каждого из найденных корней (один и тот же корень может быть найден несколькими нитями). В столбце «Число нитей» указано число нитей по окончании соответствующей итерации. При этом статистика сходимости может зависеть от задаваемых параметров допустимой точности. Увеличение невязки связано с нахождением близких корней в разных нитях, из которых алгоритм оставляет один, невязка на котором не превосходит заданного порога.

| Номер итерации | Невязка | Число нитей |
|---|---|---|
| 1 | | 31 |
| 2 | | 41 |
| 3 | | 64 |
| 4 | 0.004; 0.003 | 83 |
| 5 | 0.005 | 116 |
| 6 | 0.002; 0.002; 0.002 | 158 |
| 7 | 0.004; 0.003; 0.002 | 185 |

Таблица 2. Статистика сходимости.

На рис. 2 показаны линии нулевого уровня функций $F_j$ в сечении $xy$ с начальным приближением $(0, 0, 0)$. Можно видеть, что одна пара линий уровня пересекается в трех точках — это означает, что алгоритм, запущенный только для нитей с $k = 1$, создаст для каждой такой нити еще две нити, поменяв $k$ на $k = 2$ и $k = 3$, и положив новые приближения в этих нитях равным $(x, 0, 0)$, где $x$ — координата точки пересечения линий уровня в сечении $xy$.

Наконец, рассмотрим пример системы с большим числом корней (см. рис. 3):

$$F_1(x, y, z) = x + 0.5 \sin x + 0.5y + 0.25 \sin y + 1.5z + 0.75 \sin z,$$
$$F_2(x, y, z) = x + y + z,$$
$$F_3(x, y, z) = x + 0.3 \cos x + 0.5(y + 0.3 \cos y) + 1.5(z + 0.3 \cos z).$$

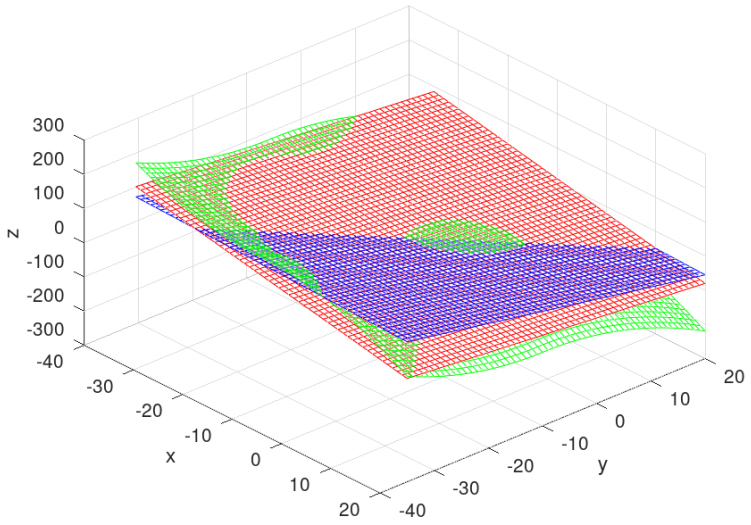


Рис. 1. Поверхности нулевого уровня функций $F_j$.

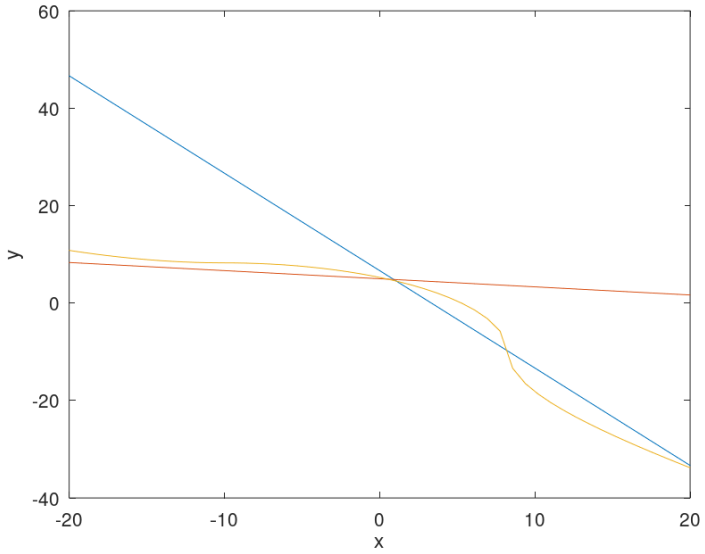


Рис. 2. Линии нулевого уровня функций $F_j$ в сечении $xy$ при $z = 0$.

Результаты вычислений зависят от того, в какой области ищется корень. Так, если задать ограничение на поиск корней $|x_i| \leq 15$, то после 5 итераций с

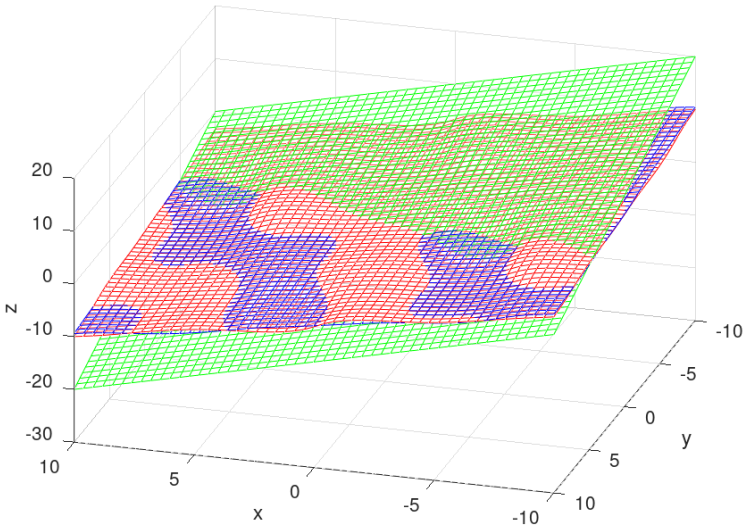


Рис. 3. Поверхности нулевого уровня функций $F_j$.

начальным приближением $(0, 0, 0)$ алгоритм находит 5 корней: $(-1.97, 1.25, 0.71)$, $(6.13, -3.33, -2.79)$, $(-6.42, 2.94, 3.48)$, $(-14.53, 7.52, 7.00)$, $(10.59, -5.02, -5.56)$.

Чтобы проанализировать работу алгоритма, рассмотрим функции $x_k^{ij}(z)$, $y_k^{ij}(x)$ и $z_k^{ij}(y)$, где первая из них — это $x$-координата $k$-й точки пересечения кривых $\{F_i = 0\}$ и $\{F_j = 0\}$ в сечении $xy$ при заданном $z$, вторая функция — $y$-координата точки пересечения тех же кривых в сечении $yz$, и третья — $z$-координата в сечении $zx$. Графики функций $x_{ij}^k$, $y_{ij}^k$, $z_{ij}^k$ для системы из второго примера изображены на рис. 4–6. Каждому корню соответствует точка пересечения трех линий с разной штриховкой, при этом первый корень соответствует значениям индексов $i_{xy} = 1, j_{xy} = 2, k_{xy} = 1$, $i_{yz} = 1, j_{yz} = 3, k_{yz} = 1$, $i_{zx} = 2, j_{zx} = 3, k_{zx} = 1$, второй корень — $i_{xy} = 1, j_{xy} = 2, k_{xy} = 1$, $i_{yz} = 2, j_{yz} = 3, k_{yz} = 2$, $i_{zx} = 1, j_{zx} = 3, k_{zx} = 2$, и третий корень — $i_{xy} = 2, j_{xy} = 3, k_{xy} = 1$, $i_{yz} = 1, j_{yz} = 3, k_{yz} = 1$, $i_{zx} = 1, j_{zx} = 2, k_{zx} = 1$.

Таким образом, наш алгоритм можно рассматривать как метод переменных направлений для функций $x_k^{ij}$, $y_k^{ij}$ и $z_k^{ij}$. Двигаясь в каждом сечении вдоль выбранной кривой, приходим к одному из корней.

По сравнению с методом ветвей и границ, предложенный метод проигрывает в том, что не дает информации об отсутствии корней в выделенной области. Однако новый алгоритм строит итерационную последовательность, которая сходится к корню менее чем за 10 итераций. Дальнейшие исследования затронутого вопроса могут коснуться построения модификаций предложенного алгоритма, учитывающих возможные отсечения лишних вычислительных нитей, и реализации параллельного варианта алгоритма.

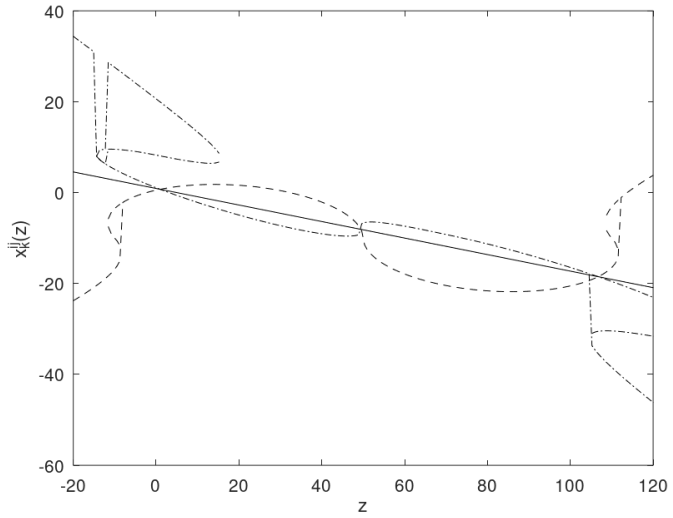


Рис. 4. Графики функций $x_{ij}^k$ при $i = 1, j = 2$ (сплошная линия), $i = 1, j = 3$ (штрих-пунктирная линия) и $i = 2, j = 3$ (пунктирная линия), где $k = 1, 2, 3$.

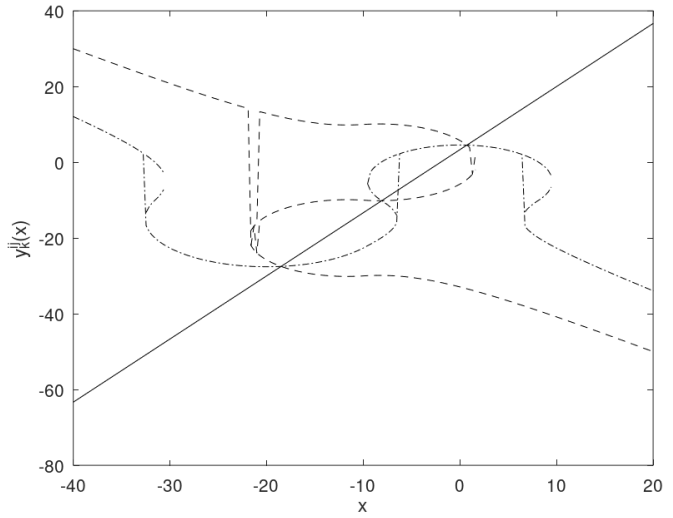


Рис. 5. Графики функций $y_{ij}^k$ при $i = 1, j = 2$ (сплошная линия), $i = 1, j = 3$ (штрих-пунктирная линия) и $i = 2, j = 3$ (пунктирная линия), где $k = 1, 2, 3$.

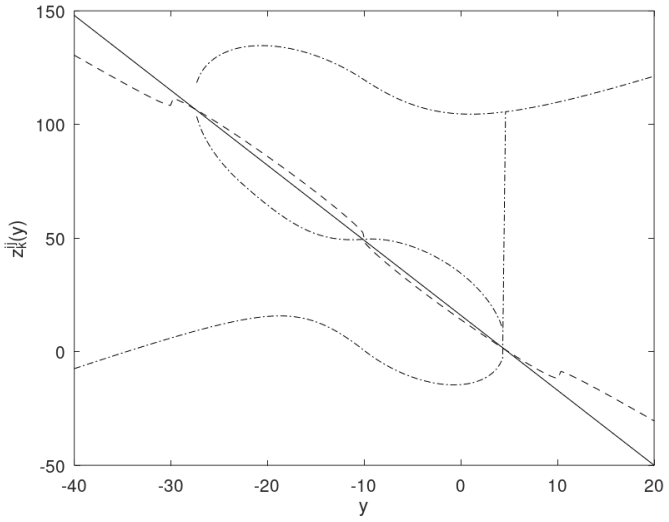


Рис. 6. Графики функций $z_{ij}^k$ при $i = 1, j = 2$ (сплошная линия), $i = 1, j = 3$ (штрих-пунктирная линия) и $i = 2, j = 3$ (пунктирная линия), где $k = 1, 2, 3$.

## References

[1] J. M. Ortega, W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables,* Academic Press, 1970.

[2] J. M. Dennis, R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations,* Prentice-Hall, 1988.

[3] E. Hansen, G. W. Walster, *Global optimization using interval analysis,* Marcel Decker, 2004.

[4] K. Goulianas, A. Margaris, M. Adamopoulos, *Finding all real roots of* $3 \times 3$ *nonlinear algebraic systems using neural networks,* Appl. Math. Comput. 2013. V. 219. P. 4444–4464.

[5] V. P. Bulatov, *Numerical methods for finding all real roots of systems of nonlinear equations,* Comput. Math. Math. Phys. 2000. V. 40. N 3. P. 331–338.

Gleb Vladimirovich Grenkin
Vladivostok State University,
ul. Gogolya, 41,
690014, Vladivostok, Russia
*Email address*: Gleb.Grenkin@vvsu.ru