Компьютерные инструменты в образовании, 2025
№ 2: 73–80
УДК: 004.42
http://cte.eltech.ru
doi:10.32603/2071-2340-2025-2-73-80

АВТОМАТИЗАЦИЯ ВЫБОРА НАЧАЛЬНЫХ ПРИБЛИЖЕНИЙ ДЛЯ РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ ЧИСЛЕННЫМИ МЕТОДАМИ

Винокурова Д. В.[1], аспирант, ✉ d.v.vinokurova@gmail.com

[1]Российский государственный педагогический университет им. А. И. Герцена, набережная реки Мойки, д. 48, 191186, Санкт-Петербург, Россия

Аннотация

Выбор начальных приближений для нахождения корней нелинейных уравнений влияет на сходимость методов. В статье представлен алгоритм поиска начальных приближений для вычисления корней нелинейных уравнений для различных численных методов, который применяется для составления учебных задач. Реализация выполнена на языке программирования Python. В статье приводится псевдокод алгоритма. Рассматриваются возможности указанного алгоритма и вспомогательных функций, а также подробно описывается процесс работы программы. Представлены результаты сравнительного анализа количества итераций, необходимых для нахождения корней с применением алгоритма поиска начальных приближений и ручного метода подбора интервалов локализации корня. Предложенный подход продемонстрировал свою эффективность на 520 различных нелинейных уравнениях.

Ключевые слова: *начальные приближения, корни нелинейных уравнений, вычислительная математика, Python, численные методы*

Цитирование: Винокурова Д. В. Автоматизация выбора начальных приближений для решения нелинейных уравнений численными методами // Компьютерные инструменты в образовании. 2025. № 2. С. 73–80. doi:10.32603/2071-2340-2025-2-73-80

1. ВВЕДЕНИЕ

Нахождение корней нелинейных уравнений является важной задачей численного анализа. Существуют различные методы решения данных уравнений, такие как метод половинного деления, метод простой итерации, метод Ньютона, метод хорд и многие другие. Каждый метод имеет свои особенности. Правильный выбор начальной точки или интервала влияет на сходимость метода за определенное количество итераций.

В рассмотренной литературе [1–4] при описании различных методов не делается акцент на том, каким образом получаются начальные приближения для численных методов. Приводимые примеры ограничиваются функциями, которые не содержат асимптоты и существенные ограничения на область допустимых значений [2, 4]. В образовательных целях обучающимся для применения рассмотренных численных методов к решению задач по нелинейным уравнениям преподавателю необходимо предоставлять различные типовые задачи и интервалы локализации корней для схождения методов за

определенное количество итераций при проведении расчетов вручную. Очевидно, что чем лучше будут выбраны интервалы, тем меньше итераций потребуется для сходимости выбранного численного метода. В связи с этим требуется автоматизация процесса поиска начальных приближений для дальнейшего вычисления корней нелинейных уравнений.

В работе предлагается алгоритм поиска начальных приближений для нахождения корней нелинейных уравнений на языке программирования Python. Предложенный подход сокращает количество итераций при поиске корней, учитывая области непрерывности и асимптоты функций. Приводится вспомогательный алгоритм для вычисления множества корней различными методами, опирающимися на функцию root_scalar из библиотеки SciPy [5].

2. АЛГОРИТМ ПОИСКА НАЧАЛЬНЫХ ПРИБЛИЖЕНИЙ

Для поиска начальных приближений у функций необходимо находить их области допустимых значений (ОДЗ), чтобы избежать отрезков, в которых имеются асимптоты.

Реализация алгоритма представлена в модуле roots_nonlinear_eq в функции find_initial_approximations [6]. Псевдокод для функции find_initial_approximations (ПОИСК_НАЧАЛЬНЫХ_ПРИБЛИЖЕНИЙ) представлен в Algorithm 1. Функция содержит один обязательный параметр equation (УРАВНЕНИЕ), который принимает на вход уравнение и четыре дополнительных параметра:

- initial_range (НАЧ_ИНТЕРВАЛ) — интервал для поиска начальных приближений, который задается в виде кортежа (x_min, x_max), в псевдокоде (X_МИН, X_МАКС);
- f_interval (ИНТЕРВАЛ) — флаг наличия интервалов или отсутствия (для некоторых методов начальные приближения могут задаваться в виде одной точки либо в виде интервала);
- num_points (ЧИСЛО_ТОЧЕК) — количество точек для построения сетки;
- ndigits (КОЛИЧЕСТВО_ЗНАКОВ) — количество знаков после запятой (используется для компактного вывода учебных задач).

Алгоритм для функции СОЗДАТЬ_ЗНАЧЕНИЯ_ПО_ОДЗ приведен в Algorithm 2, данный код позволяет создать значения для x по ОДЗ функции.

Функция СОЗДАТЬ_ДОПУСТИМЫЕ_ИНТЕРВАЛЫ формирует список числовых интервалов заменяя бесконечности на конкретные числа, учитывая НАЧ_ИНТЕРВАЛ, заданный в функции ПОИСК_НАЧАЛЬНЫХ_ПРИБЛИЖЕНИЙ. Например, если интервал будет представлен в виде $(-\infty, 2)$, то при условии, что X_МИН = 1, X_МАКС = 3 функция вернет $[(1, 2)]$. Функция ПОЛУЧИТЬ_ИСКЛЮЧЕННЫЕ_ТОЧКИ извлекает точки, которые не должны входить в интервал (с учетом ОДЗ) и возвращает массив с исключенными точками.

Функция СОЗДАТЬ_ЗНАЧЕНИЯ_ПО_ОДЗ (create_domain_values) используется не для всех типов функций, в связи с тем, что:

- тригонометрические функции являются периодическими и в библиотеке Sympy их непрерывные области записываются через лямбда-функции. При наличии сложных функций внутри аргумента усложняется запись непрерывных интервалов и их необходимо раскрывать по определенным алгоритмам. Существует риск пропуска особенностей поведения функции, поэтому проводится проверка на наличие асимптот. Например, ОДЗ в Sympy в функции $cot(5 \cdot x + 6) + (4 \cdot x + 9)/(2 \cdot x + 9)$ записывается в виде: Union(Complement(Interval.open(−∞, -9/2), Union(ImageSet(Lambda(_n,

Algorithm 1:

```
ФУНКЦИЯ ПОИСК_НАЧАЛЬНЫХ_ПРИБЛИЖЕНИЙ(УРАВНЕНИЕ, НАЧ_ИНТЕРВАЛ = (-5, 5), ИНТЕРВАЛ = ИСТИНА,
ЧИСЛО_ТОЧЕК = 1000, КОЛИЧЕСТВО_ЗНАКОВ = 6):

ОДЗ ← ПОЛУЧИТЬ_ОДЗ(УРАВНЕНИЕ, X)
X_МИН ← НАЧ_ИНТЕРВАЛ[0]
X_МАКС ← НАЧ_ИНТЕРВАЛ[1]

ЕСЛИ НЕ ЯВЛЯЕТСЯ_ТРИГОНОМЕТРИЧЕСКИМ(УРАВНЕНИЕ) И
     НЕ (ЯВЛЯЕТСЯ_ДРОБНО_РАЦИОНАЛЬНЫМ(УРАВНЕНИЕ) ИЛИ
         ЯВЛЯЕТСЯ_КОМБИНИРОВАННЫМ_ДРОБНЫМ(УРАВНЕНИЕ)) И ОДЗ ≠ ℝ:
   X_ЗНАЧЕНИЯ  СОЗДАТЬ_ЗНАЧЕНИЯ_ПО_ОДЗ(ОДЗ,
   X_МИН, X_МАКС, ЧИСЛО_ТОЧЕК, КОЛИЧЕСТВО_ЗНАКОВ)
ИНАЧЕ
   X_ЗНАЧЕНИЯ ← РАВНОМЕРНАЯ_СЕТКА(X_МИН, X_МАКС, ЧИСЛО_ТОЧЕК)
   Y_ЗНАЧЕНИЯ ← ПУСТОЙ_СПИСОК
ДЛЯ КАЖДОГО X_ТЕКУЩИЙ ИЗ X_ЗНАЧЕНИЯ:
   ПОПРОБОВАТЬ:
      Y_ТЕКУЩИЙ ← ВЫЧИСЛИТЬ(УРАВНЕНИЕ, X_ТЕКУЩИЙ)
      ЕСЛИ ЯВЛЯЕТСЯ_ВЕЩЕСТВЕННЫМ(Y_ТЕКУЩИЙ):
         ДОБАВИТЬ(Y_ЗНАЧЕНИЯ, Y_ТЕКУЩИЙ)
      ИНАЧЕ:
         ДОБАВИТЬ(Y_ЗНАЧЕНИЯ, NaN)
      ЕСЛИ ОШИБКА:
         ДОБАВИТЬ(Y_ЗНАЧЕНИЯ, NaN)

НАЧАЛЬНЫЕ_ПРИБЛИЖЕНИЯ ← ПУСТОЙ_СПИСОК
ДЛЯ i ОТ 0 ДО ДЛИНА(Y_ЗНАЧЕНИЯ) - 1:
   Y1 ← Y_ЗНАЧЕНИЯ[i]
   Y2 ← Y_ЗНАЧЕНИЯ[i + 1]

   ЕСЛИ Y1 = NaN ИЛИ Y2 = NaN:
      ПРОДОЛЖИТЬ
   ЕСЛИ Y1 * Y2 < 0:
      X1 ← ОКРУГЛИТЬ(X_ЗНАЧЕНИЯ[i], КОЛИЧЕСТВО_ЗНАКОВ)
      X2 ← ОКРУГЛИТЬ(X_ЗНАЧЕНИЯ[i + 1], КОЛИЧЕСТВО_ЗНАКОВ)

ЕСЛИ НЕ ЯВЛЯЕТСЯ_ТРИГОНОМЕТРИЧЕСКИМ(УРАВНЕНИЕ) И
     НЕ (ЯВЛЯЕТСЯ_ДРОБНО_РАЦИОНАЛЬНЫМ(УРАВНЕНИЕ) ИЛИ
         ЯВЛЯЕТСЯ_КОМБИНИРОВАННЫМ_ДРОБНЫМ(УРАВНЕНИЕ)) И ОДЗ ≠ ℝ:
   ЕСЛИ ИНТЕРВАЛ:
      ДОБАВИТЬ(НАЧАЛЬНЫЕ_ПРИБЛИЖЕНИЯ, (X1, X2))
   ИНАЧЕ:
      ДОБАВИТЬ(НАЧАЛЬНЫЕ_ПРИБЛИЖЕНИЯ, X1)
ИНАЧЕ ЕСЛИ НЕ ЯВЛЯЕТСЯ_ТРИГОНОМЕТРИЧЕСКИМ(УРАВНЕНИЕ)
   ЕСЛИ ЗНАЧЕНИЯ_ВХОДЯТ_В_ОДЗ(X1, X2, ОДЗ, КОЛИЧЕСТВО_ЗНАКОВ=4):
      ЕСЛИ ИНТЕРВАЛ:
         ДОБАВИТЬ(НАЧАЛЬНЫЕ_ПРИБЛИЖЕНИЯ, (X1, X2))
      ИНАЧЕ:
         ДОБАВИТЬ(НАЧАЛЬНЫЕ_ПРИБЛИЖЕНИЯ, X1)
   ИНАЧЕ:
      ЕСЛИ АСИМПТОТЫ_НА_ИНТЕРВАЛЕ(УРАВНЕНИЕ, X, ЧИСЛО_ТОЧЕК, X1, X2):
         ЕСЛИ ИНТЕРВАЛ:
            ДОБАВИТЬ(НАЧАЛЬНЫЕ_ПРИБЛИЖЕНИЯ, (X1, X2))
         ИНАЧЕ:
            ДОБАВИТЬ(НАЧАЛЬНЫЕ_ПРИБЛИЖЕНИЯ, X1)

   ВОЗВРАТ: НАЧАЛЬНЫЕ_ПРИБЛИЖЕНИЯ, X_ЗНАЧЕНИЯ, Y_ЗНАЧЕНИЯ
```

Algorithm 2:

```
ФУНКЦИЯ СОЗДАТЬ_ЗНАЧЕНИЯ_ПО_ОДЗ (
ОДЗ,
X_МИН,
X_МАКС,
ЧИСЛО_ТОЧЕК,
КОЛИЧЕСТВО_ЗНАКОВ = 6):

   ДОПУСТИМЫЕ_ИНТЕРВАЛЫ ← СОЗДАТЬ_ДОПУСТИМЫЕ_ИНТЕРВАЛЫ(ОДЗ, X_МИН, X_МАКС, КОЛИЧЕСТВО_ЗНАКОВ)
   ИСКЛЮЧЕННЫЕ_ТОЧКИ ← ПОЛУЧИТЬ_ИСКЛЮЧЕННЫЕ_ТОЧКИ(ОДЗ, КОЛИЧЕСТВО_ЗНАКОВ)

   СПИСОК_СЕТОК ← ПУСТОЙ_СПИСОК
   ДЛЯ КАЖДОГО (НАЧАЛО, КОНЕЦ) ИЗ ДОПУСТИМЫЕ_ИНТЕРВАЛЫ:
      ДЛЯ КАЖДОЙ ТОЧКА ИЗ ИСКЛЮЧЕННЫЕ_ТОЧКИ:
      ЭПСИЛОН ← 10 ^ (- КОЛИЧЕСТВО_ЗНАКОВ)
         ЕСЛИ ЧИСЛА_ПОЧТИ_РАВНЫ(КОНЕЦ, ТОЧКА, ЭПСИЛОН):
            КОНЕЦ ← КОНЕЦ | 10^(-КОЛИЧЕСТВО_ЗНАКОВ)
         ЕСЛИ ЧИСЛА_ПОЧТИ_РАВНЫ(НАЧАЛО, ТОЧКА, ЭПСИЛОН):
            НАЧАЛО ← НАЧАЛО + 10^(-КОЛИЧЕСТВО_ЗНАКОВ)
      ДОБАВИТЬ РАВНОМЕРНУЮ_СЕТКУ(НАЧАЛО, КОНЕЦ, ЧИСЛО_ТОЧЕК / ЧИСЛО_ИНТЕРВАЛОВ,
      БЕЗ_ПОСЛЕДНЕЙ_ТОЧКИ) В СПИСОК_СЕТОК
   ЕСЛИ ДЛИНА(ИСКЛЮЧЕННЫЕ_ТОЧКИ) = 0:
      ДОБАВИТЬ РАВНОМЕРНУЮ_СЕТКУ(НАЧАЛО, КОНЕЦ, ЧИСЛО_ТОЧЕК / ЧИСЛО_ИНТЕРВАЛОВ,
      БЕЗ_ПОСЛЕДНЕЙ_ТОЧКИ) В СПИСОК_СЕТОК
   ОБЩИЙ_МАССИВ ← КОНКАТЕНАЦИЯ(СПИСОК_СЕТОК)
   ВОЗВРАТ: ОБЩИЙ_МАССИВ
```

2*_n*pi/5 - 6/5 + 2*pi/5), Integers), ImageSet(Lambda(_n, 2*_n*pi/5 - 6/5 + 3*pi/5), Integers))), Complement(Interval.open(-9/2, oo), Union(ImageSet(Lambda(_n, 2*_n*pi/5 - 6/5 + 2*pi/5), Integers), ImageSet(Lambda_n, 2*_n*pi/5 - 6/5 + 3*pi/5), Integers))));

— в дробно-рациональных уравнениях и комбинированных уравнениях, включающих комбинацию дробно-рациональных функций с другими типами функций при таком способе формирования значений, могут получаться лишние корни, из-за слишком близких значений вблизи асимптот.

Далее под особыми типами функций в тексте статьи будем подразумевать тригонометрические, дробно-рациональные и комбинированные функции, которые включают дробно-рациональные и другие типы подфункций. Классификация функций осуществляется программой при помощи специальных условий и соответствующих функций проверки.

Нахождение начальных приближений выполняется в цикле, в котором значения выбранного отрезка проверяются на значения NaN. При отсутствии данных значений во втором условии оцениваются концы отрезка на наличие различных знаков. Выполнение условия свидетельствует о существовании корня в данном интервале. Внутри условия содержатся три дополнительных: в первом условии проверяется, содержит ли данное уравнение особые типы функций и не включает ли непрерывная область всю область действительных чисел, во втором проверяется не относится ли функция к классу тригонометрических, третье условие выполняется в случае невыполнения первых двух.

В каждом из трех условий в начальное приближение добавляется интервал либо точка, в зависимости от флага ИНТЕРВАЛ, который задается исходя из применяемого метода. Если функция не является тригонометрической (то есть выполняется второе условие), то

дополнительно проверяется вхождение рассматриваемого интервала в непрерывную область в функции ЗНАЧЕНИЯ_ВХОДЯТ_В_ОДЗ. Если функция включает тригонометрическую функцию, то в начальном приближении проверяется наличие асимптот. Попадание в асимптоту может привести к бесконечным итерациям или лишним корням. Наличие асимптот проверяется в функции АСИМПТОТЫ_НА_ИНТЕРВАЛЕ, где интервал делится на заданное количество точек в зависимости от значения ЧИСЛО_ТОЧЕК. В каждой из этих точек выполняется проверка на асимптоты дополнительной функцией, в которой вычисляются пределы слева и справа, а затем проверяются на наличие резких скачков функции. В качестве резкого скачка функции установлено значение 50. Вернемся к условию в цикле формирования начальных приближений. В условии, в котором функция является тригонометрической, если в интервале отсутствует асимптота, в зависимости от флага ИНТЕРВАЛ, добавляется соответствующее приближение.

Функция ПОИСК_НАЧАЛЬНЫХ_ПРИБЛИЖЕНИЙ предоставляет несколько приближений, если они были найдены на интервале НАЧ_ИНТЕРВАЛ, что позволяет найти все корни. Если корней достаточно много, что, как правило, может возникать в тригонометрических функциях, которые содержат асимптоты, необходимо увеличить значение ЧИСЛО_ТОЧЕК, для устранения больших скачков значений в данных функциях.

3. РЕЗУЛЬТАТЫ ПРИМЕНЕНИЯ АЛГОРИТМА

Рассмотрим результаты применения функции find_initial_approximations на практике. Для рационального нахождения корней нелинейных уравнений в модуле roots_nonlinear_eq была разработана дополнительная функция для вызова численных методов с соответствующими параметрами. Функция find_roots_with_root_scalar вычисляет корни опираясь на алгоритмы, представленные в функции root_scalar модуля SciPy [5]. Предложенные функции позволяют найти все корни у нелинейных уравнений на заданных интервалах. В модуле examples представлены варианты вызова различных типов функций из модуля roots_nonlinear_eq [6] с использованием восьми методов, реализованных в root_scalar.

Опираясь на функцию find_roots_with_root_scalar для поиска корней, была проведена оценка возможностей функции find_initial_approximations на результатах сравнения количества итераций, необходимых для поиска корней с применением алгоритма поиска начальных приближений и ручного метода подбора интервалов локализации корня. Ручной метод основывается на выборе интервалов по эскизу графика функции. Для простоты анализа в случае наличия нескольких корней у функций рассматривался только один корень. Из семи типов уравнений (логарифмических, показательных, тригонометрических, дробно-рациональных, иррациональных, полиномиальных и комбинированных) для анализа было взято по одному уравнению и осуществлены вычисления для каждого из восьми методов [6].

В процессе анализа было выявлено, что использование функции find_initial_approximations намного эффективнее ручного поиска интервала локализации корня. Применяемая функция позволила сократить количество итераций и предоставила интервалы, которые привели к схождению численных методов, в отличие от ручного подбора (рис. 1). Результаты анализа для других функций доступны в Zenodo в файле Excel [6]. Отметим, что в некоторых уравнениях количество итераций достигало 50. Это числовое ограничение по максимальному количеству итераций, которое могло иметь значительно большее значение.

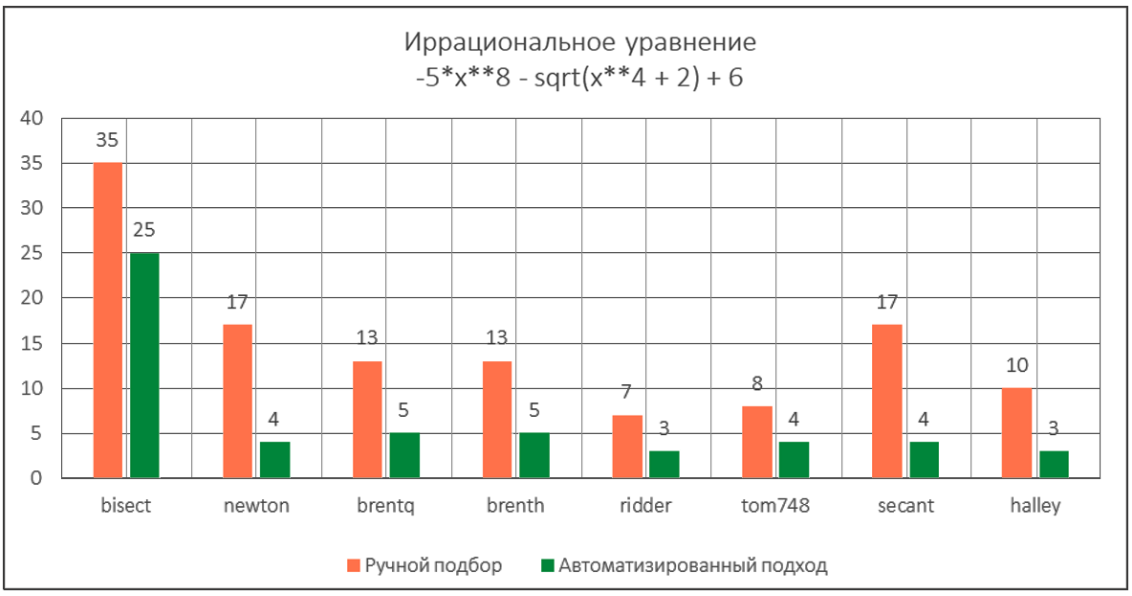

Рис. 1. Сравнительная гистограмма необходимого количества итераций для решения иррационального уравнения различными методами

В ходе функционального тестирования было установлено, что описанный метод поиска начальных приближений позволяет найти множество корней, существующих на выбранном интервале для различных типов функций. В некоторых задачах, в случае недостающих корней, возникает необходимость в увеличении значения ЧИСЛО_ТОЧЕК. Как правило, это происходит в уравнениях, содержащих функции с асимптотами.

При использовании предложенного метода с другими типами функций, выходящими за рамки протестированных случаев, рекомендуется экспериментальная проверка результатов для подтверждения корректности и устойчивости работы алгоритма.

4. ЗАКЛЮЧЕНИЕ

Алгоритм нахождения начальных приближений для нелинейных уравнений показал свою успешность на тестах семи различных типов уравнений, которые в общей сложности составили 520 уравнений, что подтверждает его корректность на использованном множестве функций. Сравнительный анализ количества итераций, необходимых для поиска корня каждым методом, показал, что предложенный подход значительно сокращает поиск корней и обеспечивает сходимость численных методов. Предложенный подход способствует эффективному и рациональному выполнению вычислений в области вычислительной математики. В образовательных целях данный алгоритм позволит обучающимся применять полученные начальные приближения для поиска корней нелинейных уравнений. Это обеспечит нахождение корней за минимальное число итераций. Программа реализована в виде модуля Python. Модуль размещён в открытом репозитории Zenodo [6], откуда его можно скачать и импортировать в собственные проекты.

Список литературы

1. *Зенков А. В.* Вычислительная математика для IT-специальностей : учебное пособие. М., Вологда: Инфра-Инженерия, 2022.
2. *Мицель А. А.* Вычислительные методы : учебное пособие. Томск: Эль Контент, 2013.
3. *Агапова Е. Г.* Вычислительная математика : учебное пособие. Хабаровск: Изд-во Тихоокеан. гос. ун-та, 2017.
4. *Коледин В. В.* Вычислительная математика : учебное пособие. Нижневартовск, 2023.
5. SciPy documentation. Version: 1.14.1. [Электронный ресурс]. URL: https://docs.scipy.org/ (дата обращения: 15.11.24).
6. *Винокурова Д. В.* Программная реализация алгоритма поиска начальных приближений для решения нелинейных уравнений и анализ результатов. Zenodo, 2024. doi:10.5281/zenodo.17350251

Поступила в редакцию 10.05.2025, окончательный вариант — 16.07.2025.

Винокурова Дарья Валентиновна, аспирант, Институт информационных технологий и технологического образования, РГПУ им. А. И. Герцена, ✉ d.v.vinokurova@gmail.com

Computer tools in education, 2025
№ 2: 73–80
http://cte.eltech.ru
doi:10.32603/2071-2340-2025-2-73-80

Automation of Initial Approximations Choice for Solving Nonlinear Equations by Numerical Methods

Vinokurova D. V.[1], Postgraduate, ✉ d.v.vinokurova@gmail.com

[1]Herzen University, 48 Moika river embankment, 191186, Saint Petersburg, Russia

Abstract

The choice of initial approximations for finding the roots of nonlinear equations affects the convergence of methods. This paper presents the algorithm for finding initial approximations for computing the roots of nonlinear equations for various numerical methods. This algorithm is used for compiling educational tasks. The implementation is written in the Python programming language. The article provides pseudocode for the algorithm. The possibilities of the specified algorithm and auxiliary functions are considered, and the process of the program operation is described in detail. The results of a comparative analysis of the number of iterations required to find roots using the initial approximation search algorithm and the manual method of selecting root localization intervals are presented. The proposed approach has demonstrated its efficiency on 520 different nonlinear equations.

Keywords: *initial approximations, roots of nonlinear equations, computational mathematics, Python, numerical methods.*

Citation: D. V. Vinokurova, “Automation of Initial Approximations Choice for Solving Nonlinear Equations by Numerical Methods,” *Computer tools in education*, no. 2, pp. 73–80, 2025 (in Russian); doi:10.32603/2071-2340-2025-2-73-80

References

1. A. V. Zenkov, *Vychislitel’naya matematika dlya IT-spetsial’nostey: uchebnoe posobie* [Computational Mathematics for IT Specialties: A Textbook], Moscow, Vologda, Russia: Infra-Engineering, 2022 (in Russian).
2. A. A. Mitsel, *Vychislitel’nye metody: uchebnoe posobie* [Computational Methods: A Textbook], Tomsk, Russia: El Content, 2013 (in Russian).
3. E. G. Agapova, *Vychislitel’naya matematika: uchebnoe posobie* [Computational Mathematics: A Textbook], Khabarovsk, Russia: Pacific State Institute Publishing House, 2017 (in Russian).
4. V. V. Koledin, *Vychislitel’naya matematika: uchebnoe posobie* [Computational Mathematics: A Textbook], Nizhnevartovsk, Russia, 2023 (in Russian).
5. SciPy Community, *SciPy documentation*, Version 1.14.1, 2024. [Online]. Available: https://docs.scipy.org/.
6. D. V. Vinokurova, “Programmnaya realizaciya algoritma poiska nachal’nyh priblizhenij dlya resheniya nelinejnyh uravnenij i analiz rezul’tatov” [Software Implementation of the Algorithm for Searching Initial Approximations for Solving Nonlinear Equations and Analysis of the Results], *Zenodo*, 2024 (in Russian). [Online]. Available: https://zenodo.org/record/17350251. doi:10.5281/zenodo.17350251

Received 10-05-2025, the final version — 16-07-2025.

Daria Vinokurova, Postgraduate, Institute of Computer Science and Technology Education, Herzen University, ✉ **d.v.vinokurova@gmail.com**