**Решение алгебраических уравнений методом половинного деления в Excel**

В рамках подготовки курса для бакалавров МФТИ я понял, что в моем блоге не так много заметок по использованию Excel в математике и физике. Каково же было мое удивление, когда я обнаружил, что книг по этой теме на русском языке буквально единицы. Ранее я опубликовал [Вильям Дж. Орвис. Excel для ученых, инженеров и студентов](https://baguzin.ru/wp/?p=23847). Представлено четыре варианта решения алгебраических уравнений методом половинного деления: итерационная таблица на листе Excel, процедура VBA, функция VBA, пользовательская функция Excel на основе LAMBDA. В настоящей заметке использованы материалы книги Алексея Васильева «Числовые расчеты в Excel». Бумажная и электронная версии книги доступны на сайте [издательства](https://lanbook.com/catalog/informatika/chislovye-raschety-v-excel/).



Рис. 1. Метод половинного деления: цифрами и штрихованными линиями со стрелкой показана последовательность перемещения границ интервала поиска решения

### Алгоритм

Метод половинного деления используется для поиска решения уравнения вида f(x) = 0 на интервале х1 < х < х2 при условии, что функция f(x) на этом интервале непрерывна и принимает на границах интервала значения разных знаков. Если все условия выполнены, то с помощью метода половинного деления можно найти решение уравнения с любой заданной (отличной от нуля) точностью за конечное количество итераций.

Если функция на интервале непрерывна и на границах интервала принимает значения разных знаков, то на этом интервале у функции имеется хотя бы один нуль (т. е. хотя бы в одной точке функция принимает нулевое значение). Но таких нулей может быть и больше одного. Другими словами, необходимые для применения метода половинного деления условия гарантируют, что корень есть, но не гарантируют, что он один. И если корней несколько, то какой именно будет найден — сказать достаточно сложно (все зависит от начальных границ интервала поиска решения). Локализация корней уравнения — отдельная сложная задача, решение которой выходит за рамки книги. В данном конкретном случае, чтобы гарантировать единственность решения на указанном интервале на функцию, определяющую уравнение, можно наложить условие неизменности знака производной (на всем интервале поиска решения). Таким образом, если мы решаем уравнение f(x) = 0 и функция f(x) на интервале поиска решения непрерывна, монотонна (монотонно возрастает или монотонно убывает) и на границах интервала принимает значения разных знаков, то уравнение имеет решение, и это решение единственно.

Алгоритм поиска решения алгебраического уравнения методом половинного деления (см. рис. 1):

* вычисляется значение функции f(x) в центральной точке интервала;
* если функция в центральной точке меньше нуля, в центральную точку смещается та граница интервала поиска решения, на которой функция меньше нуля;
* если функция в центральной точке больше нуля, в центральную точку смещается та граница интервала поиска решения, на которой функция больше нуля;
* в результате интервал поиска решения уменьшается вдвое, а задача, фактически, сводится к предыдущей (с поправкой на измененный интервал) — нужно найти корень уравнения f(x) = 0 на интервале, на границах которого функция f(x) принимает значения разных знаков;
* процесс продолжается до тех пор, пока ширина интервала не станет достаточно малой для того, чтобы обеспечить необходимую точность вычисления корня.

### Решение на основе встроенных функций Excel

В Excel метод половинного деления можно реализовывать несколькими способами. Начнем с метода, основанного на встроенных функциях (т.е., без кода VBA). Для конкретики рассмотрим кубическое уравнение х3 - 8х2 + х + 42 = 0, которое имеет три решения:



Рис. 2. График функции f(x) = х3 - 8х2 + х + 42

Начнем с поиска второго корня х2 = 3. Исходный интервал 0 < х < 5.



Рис. 3. Исходные данные и несколько итераций (формулы см. прикрепленный Excel-файл)

Структура данных:

* столбец А – номер итерации;
* столбец В – левая граница интервала поиска решения;
* столбец С – правая граница интервала;
* столбец D – середина интервала;
* столбцы Е, F и G – значения функции на границах интервала и в его середине.

На рис. 3 видно, что десяти итераций было достаточно, чтобы получить значение корня с погрешностью ε < 0,001. Точному решению будет соответствовать ноль в столбце G.

### Процедура VBA

Создадим модель, в которой вид функции и интервал поиска решения будут задаваться на листе Excel, а вычисления выполняться с помощью пользовательской процедуры, написанной на VBA. В одну ячейку листа введем «пробную» точку для аргумента функции (например, центральную точку интервала поиска решения), а в другую ячейку введем формулу, которая на основе ячейки с аргументом вычисляет значение функции.

Напрашивается желание создать не процедуру VBA, а функцию. Эта функция в качестве результата могла бы возвращать значение корня уравнения, и у нее могло бы быть, например, пять аргументов:

* ссылка на ячейку с аргументом функции;
* ссылка на ячейку со значением функции;
* два аргумента, определяющие значения для границ интервала поиска решения;
* аргумент, определяющий точность, с которой должен быть вычислен корень уравнения.

Однако, из пользовательской функции нельзя изменить значение в ячейке. Процедура позволяет это сделать. Нам потребуются четыре ячейки на листе Excel: нижняя граница интервала поиска решения, верхняя граница, аргумент функции и значение функции. Расположим их следующим образом:



Рис. 4. Структура листа Excel для процедуры VBA

Точность вычисления корня зададим внутри процедуры. На момент запуска процедуры активной должна быть ячейка с формулой (В2). Вычисленное значение корня уравнения вернется в ячейку с аргументом функции (С2).

Фактически, нам нужна процедура с аргументами. Однако удобнее пользоваться макросом — процедурой без аргументов. Поэтому аргументы процедуре мы передадим неявно, через структуру ячеек листа, отталкиваясь от той ячейки, которая активна на момент запуска макроса.

#### Код процедуры

Sub Дихотомия()

' Погрешность вычисления корня

Dim epsilon As Double

epsilon = 0.000001

' Запоминаем активную на момент запуска макроса ячейку

Dim cellF As Range

Set cellF = ActiveCell.Range("A1")

' Запоминаем ячейку справа от активной

Dim cellX As Range

Set cellX = cellF.Offset(0, 1)

' Переменные для начальных значений границ интервала поиска решения

Dim a As Double, b As Double

' Считываем значения из ячеек рабочего листа

a = cellF.Offset(1, 0).Value

b = cellF.Offset(1, 1).Value

' Переменные для значений функции на границах интервала

Dim Fa As Double, Fb As Double

' Переменная для запоминания начального значения

' ячейки с аргументом функции

Dim start As Variant

' Запоминаем начальное значение ячейки с аргументом функции

start = cellX.FormulaLocal

' Меняем содержимое ячейки с аргументом на значение левой границы

cellX.Value = a

' Запоминаем значение функции на левой границе

Fa = cellF.Value

' Меняем содержимое ячейки с аргументом на значение правой границы

cellX.Value = b

' Запоминаем значение функции на правой границе

Fb = cellF.Value

' Если на границах интервала функция принимает

' значения одинаковых знаков

If Fa \* Fb > 0 Then

' Значение ячейки-аргумента возвращается в исходное состояние

 cellX.FormulaLocal = start

' Отображается диалоговое окно с сообщением

 MsgBox "Указан неверный диапазон поиска корня!"

' Завершается работа процедуры

 Exit Sub

' Другие варианты

Else

' Если на левой границе у функции нулевое значение

 If Fa = 0 Then

' В ячейку-аргумент записывается значение левой границы

 cellX.Value = a

' Завершается работа процедуры

 Exit Sub

' Если на левой границе функция не обращается в ноль

 Else

' Если на правой границе у функции нулевое значение

 If Fb = 0 Then

' В ячейку-аргумент записывается значение правой границы

 cellX.Value = b

' Завершается работа процедуры

 Exit Sub

 End If

 End If

End If

' Переменные для запоминания центральной точки интервала и

' значения функции в этой точке

Dim х As Double, Fx As Double

' Оператор цикла - вычисляем корень уравнения

Do

' Точка в центре интервала поиска корня уравнения

 х = (a + b) / 2

' Записываем значение в ячейку с аргументом

 cellX.Value = х

' Запоминаем значение функции в центральной точке

 Fx = cellF.Value

' Если знаки функции на левой границе и в центре совпадают

 If Fx \* Fa > 0 Then

' Левая граница интервала переносится в центр

 a = х

' Значение функции на левой границе

 Fa = Fx

' Другие варианты

 Else

' Если знаки функции на правой границе и в центре совпадают

 If Fx \* Fb > 0 Then

' Правая граница переносится в центр

 b = x

' Значение функции на правой границе

 Fb = Fx

' Если значение функции в центральной точке интервала

' равно нулю

 Else

' Завершение работы процедуры

 Exit Sub

 End If

 End If

' Циклы выполняется до тех пор, пока половинная

' ширина интервала поиска не станет меньше погрешности

Loop Until (Abs(b - a) / 2 < epsilon)

End Sub

#### Как работает код

Этот макрос – процедура без аргументов. Dim epsilon As Double объявляется числовая переменная epsilon, значение которой определяет погрешность, с которой вычисляется корень уровнения. epsilon = 0,000 001. Далее командой Set cellF = ActiveCell.Range("A1") ссылка на активную ячейку записываем в переменную cellF. Переменная cellF описана с типом объект Range (команда Dim cellF As Range). В активной ячейке содержится значение функции. Значение в ячейке справа от активной присваивается переменной cellX = cellF.Offset (0,1). В этой ячейке записан аргумент. В числовые переменные *а* и *b* первоначально записываются значения левой и правой границ интервала поиска решения: а = cellF.Offset(1,0).Value; b = cellF.Offset(1,1).Value. Чтобы определить значения функции на границах этого интервала, нам необходимо последовательно записать значение границ в ячейку с аргументом функции cellX и запомнить, какое при этом будет значение в активной ячейке (ячейка функции cellF). Значения функции на границах интервала записываются в переменные Fa и Fb.

Если на границах интервала поиска решения значения функции f(x) имеют различные знак, будет запущен процесс вычисления корня. Если значения одного знака, мы вернём в ячейку с аргументом исходное значение (до запуска макроса). Поскольку в ячейке могло быть не только число, но и, например, формула. Начальное значение ячейки мы записываем в переменную start, которая объявлена с типом Variant. Значение присваивается командой start = cellX.FormulaLocal.

Команда cellX.Value = а меняет содержимое ячейки cellX на значение левой границы. Значение функции на левой границе записываем в переменную Fa с помощью команды Fa = celIF.Value. Аналогичная процедура проделывается для правой границы интервала поиска решения. После этого «в игру» вступает условный оператор (а точнее, несколько вложенных условных операторов). С его помощью проверяется необходимое условие для применения метода половинного деления.

Если функция на границах имеет значения одинаковых знаков, то произведение значений должно быть больше нуля (условие Fa\*Fb>0). В этом случае командой cellX.FormulaLocal = start значение ячейки cellX возвращается в исходное состояние, отображается диалоговое окно с сообщением о том, что метод для данного диапазона неприменим (команда MsgBox "Указан неверный диапазон поиска корня! ").

Если на левой границе функция равна нулю (условие Fa = 0), в ячейку cellX в качестве результата записывается значение левой границы (команда сеllX.Value = а). Работа макроса завершается. Если на левой границе значение у функции ненулевое, проверяется правая граница. Если значение функции равно нулю, значение правой границы интервала становится решением уравнения и заносится в ячейку cellX.

Если на границах интервала функция принимает значения разных знаков, используем метод половинного деления. Именно здесь выполняется основная часть кода. Вводится две переменные: х – для запоминания текущего приближения для корня уравнения и Fx – значение функции в этой точке.

Вычисление приближенного значения корня уравнения выполняется внутри цикла Do…Until. Цикл выполняется, пока не станет истинным условие Abs(b-a)/2<epsilon. Х получает значение центральной точки интервала: х = (а + b)/2). Далее это значение командой сеllХ.Value = х записывается в ячейку cellX. Далее командой Fx = cellF.Value в переменную Fx записывается значение функции в центральной точке интервала. С помощью условного оператора проверяется условие Fx\*Fa>0, которое означает, что на левой границе и в центре интервала функция уравнения принимает ненулевые значения одинаковых знаков. В этом случае левая граница переносится в центр интервала (команда а = х), причем новое значение Fx получает и переменная Fa, поскольку она должна содержать значение функции уравнения на левой границе.

Если знаки функции на правой границе и в центре совпадают (условие Fx\*Fb>0), аналогичные действия выполняются по отношению к правой границе интервала поиска решения. Возможен также вариант, когда оба указанных условия не выполняются. Это означает, что в центре интервала поиска решения функция принимает нулевое значение. Следовательно, решение найдено, и выполнение макроса прекращается.

Поэкспериментируйте со значениями границ диапазона поиска корня уравнения.

### Функция VBA

Хотя я ранее указал на проблемы с использованием функции (а не процедуры), такой подход является более универсальным. Мы создадим две функции VBA: функцию уравнения и функцию, реализующую метод половинного деления. Вторая функция будет получать первую, как аргумент. При этом функция для решения уравнения создается в обычном модуле, а функция, определяющая уравнение – в модуле класса.

Чтобы создать модуль класса, в окне редактора VBA в меню Insert выберите команду Class Module.



Рис. 5. Создание модуля класса

В результате в проект добавляется модуль класса, который отображается в окне проекта Project в группе Class Modules и по умолчанию имеет название Class1. Щелкните на этом названии и переименуйте в Equations.



Рис. 6. В проекте создан модуль класса с названием по умолчанию Class1

В окно кода модуля класса введите код:

Function eqn(x As Double) As Double

 'Значение функции, определяющей уравнение

 eqn = x^(3) - 8 \* x^(2) + x + 42

End Function

Для добавления обычного модуля в редакторе VBA пройдите *Insert* –> *Module*. Функции для решения дадим кириллическое название чтобы она не выделялась на фоне прочих функций русифицированной версии Excel. Функция НАЙТИКОРЕНЬ() будет иметь три аргумента: ссылка на функцию, определяющую уравнение, левую границу диапазона поиска корня, правую границу.

Подход, основанный на передаче аргументом функции, определяющей уравнение, весьма удобен, поскольку позволяет использовать одну и ту же функцию для решения разных уравнений. Проблема в том, что понятия *ссылка на функцию* в VBA нет. Поэтому первый аргумент функции НАЙТИКОРЕНЬ() – это текстовое значение. Второй и третий аргумент мы хотим использовать для передачи не только чисел, но и ссылок на ячейки листа, поэтому тип явно не указываем (VBA по умолчанию присвоит тип Variant).

Алгоритм выполнения функции НАЙТИКОРЕНЬ() сначала проверяет граничные условия: равенство нулю функции на границе и применимость метода половинного деления. Если предпосылки для применения метода есть, вычисляется корень уравнения. При этом используется рекурсия: в теле функции НАЙТИКОРЕНЬ() вызывается эта же функция НАЙТИКОРЕНЬ(), но с измененными аргументами (уменьшен вдвое диапазон поиска корня).

#### Код функции VBA

(см. также код в Module2 приложенного Excel-файла)

Function НАЙТИКОРЕНЬ(F As String, x1, x2)

' Переменная для запоминания корня уравнения

Dim x As Double

' Переменная для запоминания значения функции уравнения

' в предполагаемой точке корня уравнения

Dim Fx As Double

' Переменная для запоминания значения

' левой границы интервала поиска корня

Dim a As Double

' Переменная для запоминания значения

' правой границы интервала поиска корня

Dim b As Double

' Переменная для запоминания значения функции уравнения

' на левой границе интервала поиска корня

Dim Fa As Double

' Переменная для запоминания значения функции уравнения

' на правой границе интервала поиска корня

Dim Fb As Double

' Переменная для определения погрешности вычисления корня

Dim eps As Double

' Значение погрешности вычисления корня

eps = 0.000001

' Значение левой границы интервала поиска корня уравнения

a = x1

' Значение правой границы интервала поиска корня уравнения

b = x2

' Объектная переменная класса Equations

Dim m As Equations

' Создаем объект класса Equations

Set m = New Equations

' Вычисляем значение функции уравнения на левой границе

Fa = CallByName(m, F, VbMethod, a)

' Если на левой границе функция уравнения принимает нулевое значение

If Fa = 0 Then

' Значение левой границы - корень уравнения

НАЙТИКОРЕНЬ = a

' Завершение выполнения функции

Exit Function

End If

' Вычисление значения функции на правой границе

Fb = CallByName(m, F, VbMethod, b)

' Если на правой границе функция уравнения равна нулю

If Fb = 0 Then

' Значение правой границы - корень уравнения

НАЙТИКОРЕНЬ = b

' Завершение выполнения функции

Exit Function

End If

' Если на границах интервала поиска корня функция уравнения

' принимает значения одинаковых знаков

If Fa \* Fb > 0 Then

' Функция возвращает ошибку

НАЙТИКОРЕНЬ = CVErr(xlErrValue)

' Завершение выполнения функции

Exit Function

End If

' Вычисляем центральную точку интервала поиска корня

x = (a + b) / 2

' Вычисляем значение функции в центральной точке интервала

' поиска корня

Fx = CallByName(m, F, VbMethod, x)

' Если корень локализован с достаточной точностью или

' функция уравнения в центральной точке интервала равна нулю

If (Abs(b - a) < 2 \* eps) Or (Fx = 0) Then

' Центральная точка интервала поиска корня - решение уравнения

НАЙТИКОРЕНЬ = x

' Если корень не найден

Else

' Если в центре интервала и на левой границе функция уравнения

' принимает значения одинаковых знаков

If Fa \* Fx > 0 Then

' Рекурсивный вызов функции поиска корня уравнения.

' Левая граница интервала поиска корня смещена в центр

НАЙТИКОРЕНЬ = НАЙТИКОРЕНЬ(F, x, b)

' Если в центре интервала и на правой границе функция уравнения

' принимает значения одинаковых знаков

Else

' Рекурсивный вызов функции поиска корня уравнения.

' Правая граница интервала поиска корня смещена в центр

НАЙТИКОРЕНЬ = НАЙТИКОРЕНЬ(F, a, x)

End If

End If

End Function

#### Описание работы функции НАЙТИКОРЕНЬ()

Через F обозначено имя функции, которая определяет решаемое уравнение. Предполагается, что у этой функции один аргумент и в качестве значения она возвращает число. Аргументы х1 и х2 обозначают границы поиска решения. Это могут быть как числовые значения, так и адреса ячеек. Ячейки, в свою очередь, должны содержать числовые значения. Хотя идеологически разница не очень большая, числа и ссылки в программном коде обрабатываются по-разному. Чтобы избежать неоднозначности, мы в программном коде функции вводим две переменные а и b типа Double и в качестве значений им присваиваем х1 и х2. Такой прием срабатывает благодаря тому, что для ссылок на ячейки имеется «свойство по умолчанию»: значение ячейки. Если формально числовой переменной присваивается ссылка на ячейку, то на самом деле будет использовано свойство Value соответствующей ячейки, т. е. ее значение. В дальнейшем мы будем «оперировать» именно переменными а и b.

Точность вычисления корня уравнения определяется локальной переменной eps.

Для вызова функции уравнения воспользуемся встроенной функцией VBA CallByName. Для этого вызываемая функция должна быть методом объекта. Чтобы создать объект, мы сначала создаем класс (на основе которого потом создаем объект). Технически создание класса состоит в том, что мы добавляем модуль класса и меняем его название на *Equations*. Команда Dim m As Equations объявляется объектная переменная класса Equations. Для создания объекта класса Equations мы используем команду New Equations. Результат этой команды (ссылка на созданный объект) присваивается в качестве значения переменной *m*. Все вместе выглядит как Set m = New Equations. Объект *m* будет использован для передачи первым аргументом функции CallByName().

Функции CallByName() передаются такие аргументы:

* ссылка на объект, из которого вызывается метод;
* имя вызываемого метода (функции);
* константа, определяющая режим вызова метода;
* аргументы для передачи методу (если есть).

Для вычисления значения функции уравнения на левой границе мы используем команду Fa = CallByName(m,F,VbMethod,a). В переменную Fa записывается результат вызова функции/метода с именем F, который вызывается из объекта *m* с аргументом *а*. Константа VbMethod, переданная третьим аргументом функции CallByName(), определяет режим вызова метода.

В общем случае могут использоваться следующие константы: VbGet (вызывается Get-процедура свойства), VbSet (вызывается Set-процедура свойства), VbLet (вызывается Let-процедура свойства) и VbMethod (вызывается метод).

Сразу после того, как вычислено значение функции уравнения на левой границе, проверяется, равна или нет функция уравнения нулю на левой границе. Если на левой границе функция уравнения принимает нулевое значение (условие Fa = 0), то значение левой границы – корень уравнения. Выполняются команды НАЙТИКОРЕНЬ = а (значение, возвращаемое функцией) и Exit Function. В случае, когда на левой границе функция уравнения принимает ненулевое значение, командой Fb = Call By Name(m,F,VbMethod,b) вычисляется значение функции уравнения на правой границе, и выполняется аналогичная проверка.

В случае, когда на обеих границах функция уравнения принимает ненулевые значения, проверяется условие применимости (точнее, условие неприменимости) метода половинного деления. Если знаки функции уравнения на границах интервала поиска корня одинаковые (условие Fa\*Fb>0), в качестве результата функцией НАЙТИКОРЕНЬ() возвращается ошибка. Соответствующая команда выглядит как НАЙТИКОРЕНЬ = CVErr(xlErrValue). Здесь для «генерирования» ошибки использована встроенная функция VBA CVErr(), аргументом которой передается константа xlErrValue, соответствующая ошибке #ЗНАЧ!.

Константы для типов ошибок: xlErrNA (ошибка #Н/Д), xlErrRef (ошибка #ССЫЛКА!), xlErrDiv0 (ошибка #ДЕЛ/0!), xlErrValue (ошибка #ЗНАЧ!), xlErrName (ошибка #ИМЯ?), xlErrNum (ошибка #ЧИСЛО!) и xlErrNull (ошибка #ПУСТО!).

Если метод половинного деления применим, командой х = (а + b)/2 определяем центральную точку интервала поиска решения уравнения и вычисляем значение функции уравнения в этой точке Fx = CallByName(m,F,VbMethod,x). Если ширина интервала меньше ε (Abs(b-a)<2\*eps) или мы случайно угадали корень (Fx = 0), возвращается корень. Общее условие записано как (Abs(b-a)<2\*eps) Or (Fx = 0). Если ни одно из условий не выполняется, продолжаем половинное деление. Здесь также возможны два варианта:

* если значение функции уравнения в центре совпадает со значением функции уравнения на левой границе, левая граница сдвигается в центр, и по отношению к этому новому интервалу применяется процедура половинного деления – команда НАЙТИКОРЕНЬ = НАЙТИКОРЕНЬ(F,х,b);
* если значение функции уравнения в центре совпадает со значением функции уравнения на правой границе, правая граница сдвигается в центр, и по отношению к этому новому интервалу применяется процедура половинного деления – команда НАЙТИКОРЕНЬ = НАЙТИКОРЕНЬ(F,а,х).

Вот как работа функции выглядит на листе Excel:



Рис. 7. Функция НАЙТИКОРЕНЬ() в действии

Найдены все три корня: -2; 3; 7. Для интервала 8…10 корней нет.

Чтобы решить другое уравнение, достаточно в модуле класса Equations описать функцию для него и указать ее имя первым аргументом при вызове функции НАЙТИКОРЕНЬ().



Рис. 8. Нахождение корней функции f(x) = x4 - 4x3 + 0,5x2 – 3x + 4

### Пользовательская функция Excel на основе LAMBDA

В декабре 2020 года Microsoft [анонсировал](https://www.microsoft.com/en-us/research/blog/lambda-the-ultimatae-excel-worksheet-function/) функцию LAMBDA, которая позволяет определять новые функции, написанные на языке формул Excel. Функция LAMBDA используется на рабочих листах Excel, а не в коде VBA. LAMBDA поддерживает рекурсию, т.е. может вызывать сама себя. Это позволяет организовать циклы. В течение 2021 г. постепенно у всех пользователей Excel после плановых обновлений Office 365 появилась функция LAMBDA. Я недавно [описал](https://baguzin.ru/wp/?p=24759) работу функции. Её можно использовать и для решения уравнений методом половинного деления.[[1]](#footnote-1)

Чтобы воспользоваться функцией, скопируйте код в буфер обмена, в Excel пройдите *Формулы* –> *Диспетчер имен* –> *Создать*. В окне *Создание имени* введите имя *НКОРЕНЬ*, а в поле *Диапазон* вставьте скопированный код.



Рис. 9. Имя для пользовательской функции

#### Код функции LAMBDA

=LAMBDA(

 left;

 right;

 LET(

 mid; (left+right)/2;

 eps; 0,000001;

 Fleft; left^3-8\*left^2+left+42;

 Fright; right^3-8\*right^2+right+42;

 Fmid; mid^3-8\*mid^2+mid+42;

 ЕСЛИ(

 Fleft = 0;

 left;

 ЕСЛИ(

 Fright = 0;

 right;

 ЕСЛИ(

 Fleft\*Fright>0;

 "не выполнено условие";

 ЕСЛИ(

 ИЛИ(ABS(right - left) < 2 \* eps;Fmid = 0);

 mid;

 ЕСЛИ(

 Fleft\*Fmid>0;

 НКОРЕНЬ(mid;right);

 НКОРЕНЬ(left;mid)

 )

 )

 )

 )

 )

 )

)

#### Описание работы функции НКОРЕНЬ() на основе LAMBDA

Функция НКОРЕНЬ() может использоваться на листе Excel, как обычная функция. У нее два параметра: значения левой и правой границ интервала поиска решения:



Рис. 10. Работа функции НКОРЕНЬ()

Использован тот же алгоритм, что и в функции VBA НАЙТИКОРЕНЬ(). Функция НКОРЕНЬ() на основе LAMBDA получает два аргумента…

=LAMBDA(

 left;

 right;

… и выполняет расчет внутри функции LET. Если вы ранее не сталкивались с функцией LET, рекомендую [Упрощение формул Excel путем именования фрагментов с помощью функции LET](https://baguzin.ru/wp/?p=24081). Функция LET присваивает имена константам и значениям формул…

 LET(

 mid; (left+right)/2;

 eps; 0,000001;

 Fleft; left^3-8\*left^2+left+42;

 Fright; right^3-8\*right^2+right+42;

 Fmid; mid^3-8\*mid^2+mid+42;

… и выполняет расчет внутри нескольких вложенных ЕСЛИ. Так, например, переменной mid присвоено значение (left+right)/2.

Первое ЕСЛИ…

 ЕСЛИ(

 Fleft = 0;

 left;

… проверяет, не равно ли значение функции на левой границе нулю. Если равно, то функция НКОРЕНЬ() возвращает значение левой границы – left. Если не равно, запускается вторая проверка…

 ЕСЛИ(

 Fright = 0;

 right;

Если значение функции на правой границе равно нулю, функция НКОРЕНЬ() возвращает значение правой границы – right. Если не равно, запускается третья проверка…

 ЕСЛИ(

 Fleft\*Fright>0;

 "не выполнено условие";

Напомню, что метод половинного деления может быть использован для поиска решения уравнений при условии, что функция f(x) на интервале непрерывна и принимает на границах *значения разных знаков*. Если знаки одинаковые, НКОРЕНЬ() вернет сообщение о нарушении условия. Если знаки разные запускается очередная проверка…

 ЕСЛИ(

 ИЛИ(ABS(right - left) < 2 \* eps;Fmid = 0);

 mid;

Если разница между правой и левой границами по модулю менее двух эпсилон или мы случайно угадали корень (Fmid = 0), НКОРЕНЬ() вернет значение середины между границами – mid. Если нет, то запускается последняя проверка.

 ЕСЛИ(

 Fleft\*Fmid>0;

 НКОРЕНЬ(mid;right);

 НКОРЕНЬ(left;mid)

Если функция на левой границе и в середине принимает значения одного знака, мы меняем левую границу на середину и рекурсивно (оставаясь внутри функции НКОРЕНЬ) снова запускаем НКОРЕНЬ с измененными параметрами – НКОРЕНЬ(mid;right). Если функция на левой границе и в середине принимает значения разного знака, мы рекурсивно запускаем НКОРЕНЬ с другими параметрами – НКОРЕНЬ(left;mid).

Вуаля!

Чтобы применить функцию НКОРЕНЬ для решения иного уравнения, следует переопределить три переменные в функции LET:

 Fleft; left^3-8\*left^2+left+42;

 Fright; right^3-8\*right^2+right+42;

 Fmid; mid^3-8\*mid^2+mid+42;

Как вариант, можно использовать LAMBDA, чтобы поименовать уравнение:



Рис. 11. Имя для кубического уравнения

=LAMBDA(

 x;

 x^3-8\*x^2+x+42

)

Тогда фрагмент кода следует изменить на:

 Fleft; Ур1(left);

 Fright; Ур1(right);

 Fmid; Ур1(mid);

1. Этот метод в книге Алексея Васильева «Числовые расчеты в Excel» не описан. [↑](#footnote-ref-1)