**Вычисление корней нелинейного уравнения с заданной точностью**

**Введение**

программный линейный уравнение хорда

В современном производстве значительная роль отводится разработкам на компьютере.

Сейчас невозможно представить себе инженера, занимающегося разработкой новых конструкций без использования ЭВМ. Практически ни одно даже самое мелкое предприятие сейчас не обходится без компьютерной техники. Компьютер является мощнейшим средством для реализации различных проектов. Однако, без необходимого программного обеспечения компьютер не в состоянии сделать ничего. Каждый инженер должен уметь не только пользоваться компьютером, но и составлять для него программы, решая конкретные задачи для реально сложившихся условий.

Все в мире программирования основано на взаимодействии человек - ЭВМ и осуществляется при помощи языков программирования. Однако в последнее время появились и стандартные средства, которые значительно облегчают работу разработчика. Одним из таких пакетов является MathCad. Данный программный продукт предоставляет значительные возможности для разработки программ для решения инженерных задач. Созданные в пакете расчетные модели отличаются простотой и наглядностью, а также легко корректируются.

Цель курсовой работы - вычисление корней нелинейного уравнения с заданной точностью.

# **1. Теоретические сведения к работе**

## **.1 Методика решения нелинейных уравнений**

В общем случае нелинейное уравнение с одним неизвестным можно записать в виде:

, (1)

где  - некоторая непрерывная функция аргумента *x*.

Всякое число , обращающее функцию  в нуль, т.е. при котором , называется корнем уравнения (1).

При численном подходе задача о решении нелинейных уравнений разбивается на два этапа: *локализация* (отделение) корней, т.е. нахождение таких отрезков на оси *x*, в пределах которых содержится один единственный корень, и *уточнение корней*, т.е. вычисление приближенных значений корней с заданной точностью.

Для отделения корней уравнения (1) необходимо иметь критерий, позволяющий убедится, что, во-первых, на рассматриваемом отрезке  имеется корень, а, во-вторых, что этот корень единственный на указанном отрезке. Если функция  непрерывна на отрезке , а на концах отрезка её значения имеют разные знаки , то на этом отрезке расположен, по крайней мере, один корень. Это условие (как видно из рисунка 1) не обеспечивает единственности корня. Достаточным дополнительным условием, обеспечивающем единственность корня на отрезке  является требование монотонности функции на этом отрезке. В качестве признака монотонности функции можно воспользоваться условием знакопостоянства первой производной .



Рисунок 1 - Отделение корней. Функция f(x) не монотонна на отрезке [a, b]

Таким образом, если на отрезке  функция непрерывна и монотонна, а ее значения на концах отрезка имеют разные знаки, то на рассматриваемом отрезке существует один и только один корень. Заметим, что под этот критерий не подпадают кратные корни уравнений, например, очевидный корень уравнения 

Воспользовавшись этим критерием можно отделить корни *аналитическим* способом, находя интервалы монотонности функции.

Отделение корней можно выполнить *графически*, если удается построить график функции [4, стр. 119].

## **1.2 Уточнение корней методом половинного деления**



Рисунок 2 - Метод половинного деления

Считаем, что отделение корней уравнения (1) проведено и на отрезке  расположен один корень, который необходимо уточнить с погрешностью *e*. В качестве начального приближения корня принимаем середину этого отрезка:  (рис. 2). Затем исследуем значение функции  на концах отрезков  и . Тот из отрезков, на концах которого  принимает значения разных знаков, содержит искомый корень; поэтому его принимаем в качестве нового отрезка  (на рис. 2 это отрезок ). Вторую половину отрезка , на которой  не меняет знак, отбрасываем. В качестве следующего приближения корня принимаем середину нового отрезка  и т.д. Таким образом, *k*-е приближение вычисляется как

. (2)

После каждой итерации отрезок, на котором расположен корень, уменьшается вдвое, а после *k* итераций в  раз:

. (3)

Прекратить итерационный процесс следует, когда будет достигнута заданная точность, т.е. при выполнении условия

. (4)

Поскольку корень  принадлежит отрезку , а  - середина этого отрезка, то величина  всегда будет меньше половины длины отрезка  (см. рис. 2), т.е.

. (5)

Следовательно, условие (4) будет выполнено, если

. (6)

Таким образом, итерационный процесс нужно продолжать до тех пор, пока не будет выполнено условие (6).

В отличие от большинства других методов уточнения, метод половинного деления сходится всегда, т.е. обладает безусловной сходимостью. Кроме этого он чрезвычайно прост, поскольку требует лишь вычисления значений функции  и, поэтому применим для решения любых уравнений.

Однако метод половинного деления довольно медленный. С каждым шагом погрешность приближенного значения уменьшается в два раза, т.е.

, (7)

поэтому данный метод является методом с линейной сходимостью.

Вычислим количество итераций *N*, требуемое для достижения заданной точности *e*. Пользуясь выражением (3) можно выяснить для каких значений *k* будет выполнено условие (6), и взять в качестве *N* наименьшее из таких *k*:

, , (8)

где  - целая часть числа *x*. Например, при  и  получим .

Замечание. При реализации метода следует учитывать, что функция  вычисляется с некоторой абсолютной погрешностью . Вблизи корня значения функции  малы по абсолютной величине и могут оказаться сравнимы с погрешностью ее вычисления. Другими словами, при подходе к корню мы можем попасть в полосу шумов  и дальнейшее уточнение корня окажется невозможным. Поэтому целесообразно задать ширину полосы шумов и прекратить итерационный процесс при попадании в нее. Если принять , то итерационный процесс можно завершать, когда значение функции  после *k*-й итерации станет меньшим по модулю **., т.е.

. (9)

Также необходимо иметь ввиду, что при уменьшении интервала  увеличиваются погрешности вычисления его длины  за счет вычитания близких чисел [1, стр. 185].

## **1.3 Уточнение корней методом хорд**



Рисунок 3 - Метод хорд

Рассматриваемый метод так же, как и метод половинного деления, предназначен для уточнения корня на интервале , на концах которого функция  принимает значения разных знаков. Очередное приближение в отличие от метода половинного деления берем не в середине отрезка, а в точке , где пересекает ось абсцисс прямая линия (хорда), проведенная через точки *А* и *В* (рис. 3).

Запишем уравнение прямой, проходящей через точки *А* и *В*:

.

Для точки пересечения прямой с осью абсцисс () получим уравнение

. (10)

В качестве нового интервала для продолжения итерационного процесса выбираем тот из двух  и , на концах которого функция  принимает значения разных знаков. Для рассматриваемого случая (рис. 3) выбираем отрезок , так как . Следующая итерация состоит в определении нового приближения  как точки пересечения хорды  с осью абсцисс и т.д.

Заканчиваем процесс уточнения корня, когда расстояние между очередными приближениями станет меньше заданной точности, т.е.

 (11)

или при выполнении условия (9).



Рисунок 4 - О сходимости метода хорд

Замечание. Метод половинного деления и метод хорд очень похожи, в частности, процедурой проверки знаков функции на концах отрезка. При этом второй их них в ряде случаев дает более быструю сходимость итерационного процесса. Однако в некоторых случаях метод хорд может сходится существенно медленнее метода половинного деления. Такая ситуация показана на рис. 4. Оба рассмотренных метода не требуют знания дополнительной информации о функции . Например, не требуется, чтобы функция была дифференцируема. Даже для разрывных функций рассмотренные методы обладают гарантированной сходимостью. Более сложные методы уточнения корня используют дополнительную информацию о функции , прежде всего свойство дифференцируемости. Как результат они обычно обладают более быстрой сходимостью, но в то же время, применимы для более узкого класса функций, и их сходимость не всегда гарантирована. Примером такого метода служит метод Ньютона.

## **.4 Уточнение корней методом Ньютона**



Рисунок 5 - Метод Ньютона

Пусть нам известно начальное приближение к корню  (вопрос выбора начального приближение будет подробно рассмотрен ниже). Проведем в этой точке касательную к кривой  (рис. 5). Эта касательная пересечет ось абсцисс в точке , которую будем рассматривать в качестве следующего приближения. Значение  легко найти из рисунка:

,

выражая отсюда , получим

.

Аналогично могут быть найдены и следующие приближения. Формула для *k*+1-го приближения имеет вид

,  (12)

Из формулы (12) вытекает условие применимости метода: функция  должна быть дифференцируемой и  в окрестности корня не должна менять знак.

Для окончания итерационного процесса могут быть использованы условия (9) или (11).

## **1.5 Пакет MathCad. Использование программных фрагментов**

Mathcad - это популярная система компьютерной математики, предназначенная для автоматизации решения массовых математических задач в самых различных областях науки, техники и образования. Название системы происходит от двух слов - MATHmatica (математика) и CAD (Computer Aided Design - системы автоматического проектирования, или САПР). Так что вполне правомерно считать Mathcad математическими САПР. [1, стр. 9]

Как интегрированная система Mathcad 2000 содержит следующие основные компоненты:

. Редактор документов - редактор с возможностью вставки математических выражений, шаблонов графиков и текстовых комментариев;

. MathConnex - системный интегратор, обеспечивающий интеграцию Mathcad с рядом иных программных продуктов;

. Центр ресурсов - система управления ресурсами системы;

. Электронные книги - электронные книги с описанием типовых расчетов в различных областях науки и техники;

. Справочная система - система для получения справочных данных по тематическому и индексному каталогу, а также для поиска нужных данных по ключевому слову или фразе;

. Быстрые шпаргалки QuickSheets - короткие примеры с минимальными комментариями, описывающие применение всех встроенных операторов и функций системы;

. Броузер Интернета - собственное средство выхода в Интернет. [1, стр. 15]

Программа в Mathcad есть выражение, в свою очередь, состоящее из других выражений. Программы Mathcad содержат конструкции, во многом подобные программным конструкциям языков программирования: условные передачи управления, операторы циклов, области видимости переменных, использование подпрограмм и рекурсии.

Написание программ в Mathcad позволяет решить такие задачи, которые невозможно или очень трудно решить другим способом.

Программа Mathcad есть частный случай выражения Mathcad. Подобно любому выражению, программа возвращает значение, если за ней следует знак равенства. Точно так же, как переменную или функцию можно определить через выражение, их можно определить и с помощью программы.

Главным различием между программой и выражением является способ задания вычислений. При использовании выражения алгоритм получения ответа должен быть описан одним оператором. В программе может быть использовано столько операторов, сколько нужно. Можно рассматривать программу как «составное выражение.

Условный оператор (if, otherwise)

Действие условного оператора if состоит из двух частей. Сначала проверяется логическое выражение (условие) справа от него. Если оно истинно, выполняется выражение слева от оператора if. Если ложно - ничего не происходит, а выполнение программы продолжается переходом к ее следующей строке. Вставить условный оператор в программу можно следующим образом:

. Если необходимо, введите левую часть выражения и оператор присваивания.

. Создайте новую строку программного кода, нажав на панели Programming (Программирование) кнопку Add Line (Добавить строку).

. Нажмите кнопку условного оператора if.

. Справа от оператора if введите условие. Пользуйтесь логическими операторами, вводя их с панели Boolean (Булевы операторы).

. Выражение, которое должно выполняться, если условие оказывается выполненным, введите слева от оператора if.

. Если в программе предусматриваются дополнительные условия, добавьте в программу еще одну строку нажатием кнопки Add Line и введите их таким же образом, используя оператор if или otherwise.

Оператор otherwise используется совместно с одним или несколькими условными операторами if и указывает на выражение, которое будет выполняться, если ни одно из условий не оказалось истинным.

Операторы цикла (for, while, break, continue)

В языке программирования MathCAD имеются два оператора цикла: for и while. Первый из них дает возможность организовать цикл по некоторой переменной, заставляя ее пробегать некоторый диапазон значений. Второй создает цикл с выходом из него по некоторому логическому условию. Чтобы вставить в программный модуль оператор цикла:

. Создайте в программном модуле новую линию.

. Вставьте один из операторов цикла for или while нажатием одноименной кнопки на панели Programming (Программирование).

. Если выбран оператор for, то вставьте в соответствующие местозаполнители имя переменной и диапазон ее значений, а если while - то логическое выражение, при нарушении которого должен осуществляться выход из цикла.

. В нижний местозаполнитель введите тело цикла, т.е. выражения, которые должны выполняться циклически.

. При необходимости дополните программу другими строками и введите в них нужный код.

Диапазон значений переменной в условии цикла for можно задать как с помощью диапазона ранжированной переменной, так и с помощью вектора.

Оператор цикла for с ранжированной переменной



Оператор цикла for с вектором



Оператор цикла while,



Иногда необходимо досрочно завершить цикл, т.е. не по условию в его заголовке, а в некоторой строке в теле цикла. Для этого предназначен оператор break. Например, в листинге, как только значение переменной цикла i достигает 2, цикл, благодаря оператору break в последней строке программного модуля, прерывается. Соответственно, значение переменной х остается равным 0+1+2=3.

Оператор break внутри цикла for



Оператор break внутри цикла while



Чтобы четче обозначить границы завершения тела цикла, в его конце может использоваться дополнительная строка с оператором continue, который вводится одноименной кнопкой панели. Обычно используется для продолжения выполнения цикла путем возврата в начало тела цикла. Составим программу-функцию, формирующую новый вектор из положительных проекций исходного вектора.



**2. Алгоритмический анализ задачи**

## **.1 Постановка задачи.**

Задание состоит из трех частей.

1. Расчетно - графическая часть

Исходные данные:

· Вид нелинейного уравнения



· Аргумент функции должен быть представлен в виде дискретной переменной. Диапазон изменения переменной выбрать самостоятельно.

Результаты:

· Значения функции, ее первой и второй производных на всем диапазоне значений аргумента функции

· График функции, ее первой и второй производных

· Значения корней уравнения, полученных с помощью

а) трассировки и нанесения маркеров

b) стандартной функции поиска корней уравнения

2. Программная часть

Исходные данные:

· Результаты расчетно - графической части

· Численные методы

a) половинного деления

b) хорд

c) касательных

Результаты:

Значения корней уравнения, полученных каждым численным методом и с разной степенью точности

3. Исследовательская часть

Анализ количества итераций от точности численного метода

Исходные данные:

· Вектор точности вычисления корня уравнения

Результаты:

· Количество итераций вычисления корня уравнения для каждой величины точности

· График зависимостей количества итераций от точности вычислений

## **2.2 Графическая схема алгоритма**

Планируя процесс вычисления корней нелинейного уравнения составляем последовательность действий необходимых расчетов, который изложен в графической схеме алгоритма (рисунок 6).



Рисунок 6. Алгоритм решения задачи



Рисунок 6. Алгоритм решения задачи

При вычислении корней нелинейного уравнения первоначально, в соответствии с поставленной задачей, был разработан алгоритм её решения. В графической схеме алгоритма был отражен процесс решения задачи, это и ввод исходных данных, и выполнение расчетно - графической части, и составление программного фрагмента, и проведение исследований по анализу количества итераций, и конечно вывод результатов.

# **3. Описание документа MathCAD**

## **.1 Описание расчетно-графической части**

1. Определили вид нелинейного уравнения, в виде пользовательской функции.

2. Задали аргумент функции в виде дискретной переменной: от 2.5 до 5 с шагом 0,1.

. Рассчитали значение функции, её первой и второй производных.

. Построили график функции, её первой и второй производных.

. Путем трассировки и нанесения маркеров определили значение корня уравнения на исследуемом участке х=2.8.

. Используя стандартную функцию root, определяем корень уравнения, равный 2.806812.

## **.2 Описание программной части**

1. Согласно исследуемым численным методам, разрабатываем программные фрагменты, реализующие изучаемые численные методы.

. Рассчитали значение корня, используя программные фрагменты с заданной степенью точности, где видно что наиболее точное значение имеют корни, найденные с наименьшими параметрами точности.

**3.3 Описание исследовательской части**

1. Определили вектор точности вычисления корня уравнения (см. приложения В).

2. Используя программные фрагменты, определили количество итераций для каждого метода и построили график зависимости количества итераций от точности вычислений. Из графика видно, что при использовании метода касательных требуется менее всего итераций для точного вычисления корня исследуемой функции (см. приложение В).

В результате проведенных исследований получили следующие результаты:

· определили вектор точности, состоящий из 6 элементов со значениями от 0,1 до 0,000001 (приложение В).

· разработали программные фрагменты, для каждого из используемых численных методов решения нелинейного уравнения.

· Определили вектора для каждого из методов, которые характеризуют количество итераций, необходимых для достижения заданной точности вычисления корня.

· построили график полученных векторов.

Таким образом, можно сделать вывод, что достижение заданной точности наименее затратно при использовании метода касательных.

# **Заключение**

В результате выполнения курсовой работы была составлена графическая схема алгоритма и выполнено вычисление корней нелинейного с заданной точностью в среде Mathcad.

Произведенные расчеты в Mathcad позволили оперативно и точно получить требуемые значения и проанализировать результаты расчетов, как в числовом, так и в графическом виде.

В результате оказалось, что самым эффективным методом является метод касательных, самым неэффективным - метод половинного деления.

# **Список использованной литературы**

1 Воcкобойников Ю.Е., Очков В.Ф. «Программирование и решение задач в пакете Mathcad» Издательство НГАСУ, 2010 - 203 с.

2 Лищенко С.В. «Линейное и нелинейное программирование». - М.: Просвещение, 2011. - 178 с.

3 Практическое руководство к курсовому проектированию по курсу «Информатика» для студентов технических специальностей дневной и заочной форм обучения./ Под ред. Трохова Т.А., Самовендюк Н.В., Романькова Т.Л. - Гомель: Учреждение образования «ГГТУ имени П.О. Сухого», 2009. - с.

 Колдаев В.Д. «Численные методы и программирование». - М.: ИД «Форум», 2009. - 336 с

5 ГОСТ 7.1-84. Библиографическое описание документа. Общие требования и правила составления. - Взамен ГОСТ 7.1-76; Введ. 01.01.86.-М.: Изд-во стандартов, 1984.-78 с. (для стандартов).

 ГОСТ 2.105 - 95. Единая система конструкторской документации. Общие требования к текстовым документам. - М.: Изд-во стандартов, 1995.-36с

 М/УК 2286 Тема: «Основные приемы работы в системе MathCAd, 6.0.», Гомель, ГПИ, 2008.

 М/УК 2453 Тема: Решение систем алгоритмических и дифференциальных уравнений в среде MathCAD Windows, Гомель, ГГТУ 2000.

 М/УК 2564 Тема: «Графические средства пакета MathCAD». - Гомель, ГГТУ, 2001