Численные методы на Mathcad’е

# Введение

Сегодня не часто вспоминают о том, что компьютеры были созданы в первую очередь для проведения научных расчетов. До сих пор научные и инженерные расчеты остаются одной из важнейших, хотя, пожалуй, и не самой бросающейся в глаза сфер приложения компьютеров. За многие годы накоплены обширные библиотеки научных подпрограмм, в первую очередь, на языке FORTRAN, предназначенных для решения типовых задач (задачи линейной алгебры, интегрирование, решение дифференциальных уравнений и т. д.). Кроме того, имеется целый ряд различных математических пакетов, реализующих разнообразные численные методы, а так же способных производить аналитические математические преобразования. Пожалуй, наиболее известными сегодня являются следующие пакеты: Mathematica (фирма Wolfram Research), Maple (фирма Waterloo Maple Inc), Matlab (фирма The MathWorks), Mathcad (фирма MathSoft Inc). Первые два фактически являются языками для проведения символических математических преобразований.

Пакета Mathematica, по-видимому, является сегодня наиболее популярным в научных кругах, особенно среди теоретиков. Пакет предоставляет широкие возможности в проведении символических (аналитических) преобразований, однако требует значительных ресурсов компьютера. Система команд пакета во многом напоминает какой-то язык программирования.

Пакет Maple также весьма популярен в научных кругах. Кроме аналитических преобразований пакет в состоянии решать задачи численно. Характерной особенностью пакета является то, что он позволяет конвертировать документы в формат LaTeX – стандартный формат подавляющего большинства научных издательств мирового класса. Кроме того, ряд других программных продуктов используют интегрированный символический процессор Maple. Например, пакет подготовки научных публикаций Scientific WorkPlace (фирма TCI Software Research) позволяет обращаться к символическому процессору Maple, производить аналитические преобразования и встраивать полученные результаты в документ.

Подобно упомянутым выше пакетам, пакет Matlab фактически представляет из себя своеобразный язык программирования высокого уровня, ориентированный на решение научных задач. Характерной особенностью пакета является то, что он позволяет сохранять документы в формате языка программирования С.

Пакет Mathcad популярен, пожалуй, более в инженерной, чем в научной среде. Характерной особенностью пакета является использование привычных стандартных математических обозначений, то есть документ на экране выглядит точно так же обычный математический расчет. Для использования пакета не требуется изучать какую-либо систему команд, как, например, в случае пакетов Mathematica или Maple. Пакет ориентирован в первую очередь на проведение численных расчетов, но имеет встроенный символический процессор Maple, что позволяет выполнять аналитические преобразования. В последних версиях предусмотрена возможность создавать связки документов Mathcad с документами Mathlab. В отличие от упомянутых выше пакетов, Mathcad является средой визуального программирования, то есть не требует знания специфического набора команд. Простота освоения пакета, дружественный интерфейс, относительная непритязательность к возможностям компьютера явились главными причинами того, что именно этот пакет был выбран для обучения студентов численным методам.

В последнее время просматривается тенденция к сближению и интеграции различных пакетов. Например, последние выпуски пакетов Mathematica и Maple имеют хорошие возможности для визуального программирования; в Matlab включена библиотека аналитических преобразований Maple; Mathcad позволяет работать совместно с Matlab.

В настоящем пособии мы рассмотрим на многочисленных примерах, каким образом решаются на Mathcad’e разнообразные задачи численного анализа (решение систем линейных и нелинейных уравнений, решение дифференциальных уравнений, аппроксимация функций и т. д.). Пособие не является ни учебником по численным методам, ни руководством по Mathcad’у. Предполагается, что читатель имеет представление об основных численных методах и умеет пользоваться пакетом Mathcad хотя бы на самом примитивном уровне. Численные методы обсуждаются лишь в том объеме, который необходим для понимания работы соответствующих функций Mathcad′а. Необходимые ссылки как на учебники по численным методам, так и на руководства по пакету Mathcad читатель может найти в списке литературы. Отбор численных методов, включенных в пособие, с одной стороны отражает программу курса «Численные методы», а с другой стороны имеет в виду курс «Математическое моделирование и решение задач на ЭВМ», в котором многие из приведенных методов будут активно использоваться для решения задач компьютерного моделирования. Кроме того, к сожалению, существует настоящая пропасть между теми численными методами, которые описаны в общедоступных учебниках, и теми, которые применяются на практике. В замечательной, хотя и недоступной для большинства студентов книге "Numerical Recipes in C", авторы замечают: "Увы, времена меняются; <…> классические формулы почти абсолютно бесполезны. Они являются музейными экспонатами, хотя и прекрасными". В данном пособии делается попытка перебросить мостик через эту пропасть.

На момент написания данного пособия последней версией Mathcad′а является версия Mathcad 2000 (предыдущая версия – Mathcad 8). Поскольку данная версия еще не получила повсеместного распространения, возможности, реализованные только в этой версии, оговариваются в пособии особо.

# Глава 1. Нахождение корней уравнений

## Введение

Одним из наиболее распространенных методов поиска корней уравнений является ***метод Ньютона*** и его модификации. Пусть требуется решить уравнение. Будем считать, что *x* является решением уравнения. Разложим функцию *f(x*) в ряд в точке *x*0 близкой к точке *x* и ограничимся только первыми двумя членами разложения.



Поскольку *x* – корень уравнения, то . Следовательно,



Таким образом, если нам известно приближенное значение корня уравнения, то полученное уравнение позволяет его уточнить. Понятно, что процесс уточнения можно повторять многократно, до тех пор, пока значение функции не будут отличаться от нуля на величину меньшую, чем заданная точность поиска. Очередное *k*-е приближение находится по формуле



Ограничившись в разложении только первыми двумя членами, мы фактически заменили функцию *f(x)* на прямую линию, касательную в точке *x*0, поэтому метод Ньютона еще называют методом касательных. Далеко не всегда бывает удобно находить аналитическое выражение для производной функции. Однако, в этом и нет особой необходимости: поскольку на каждом шаге мы получаем *приближенное* значение корня, можно для его вычисления использовать *приближенное* значение производной.



В качестве малой величины можно взять, например, заданную точность вычислений , тогда расчетная формула примет вид



(1.1)



С другой стороны, для вычисления производной можно воспользоваться значениями функции, полученными на двух предыдущих шагах,

(1.2)



В таком виде метод называется ***методом секущих (secant method)***. При этом, однако, возникает проблема с вычислением первого приближения. Обычно полагают, что , то есть первый шаг вычислений проводится с использованием формулы (1.1), а все последующие – с использованием формулы (1.2). Именно эта вычислительная схема реализована в пакете ***Mathcad***. Используя метод секущих, мы не можем гарантировать, что корень находится между двумя последними приближениями. Можно, однако, для вычисления очередного приближения использовать границы интервала, на котором функция меняет знак. Такой метод называется ***методом хорд*** (***false position method***).



Идея метода секущих развивается в ***методе Мюллера***. Однако в этом методе для нахождения очередного приближения используются три предыдущие точки. Иными словами, метод использует не линейную, а квадратичную интерполяцию функции. Расчетные формулы метода следующие:

(1.3)



(1.4)



Знак перед корнем выбирается так, чтобы абсолютное значение знаменателя было максимальным.

Поскольку поиск корня заканчивается, когда выполнится условие, то возможно появление ложных корней. Например, для уравнения ложный корень появится в том случае, если точность поиска задана меньше, чем 0,0001. Увеличивая точность поиска, можно избавиться от ложных корней. Однако не для всех уравнений такой подход работает. Например, для уравнения , которое, очевидно, не имеет действительных корней, для любой, сколь угодно малой точности найдется значение *x*, удовлетворяющее критерию окончания поиска. Приведенные примеры показывают, что к результатам компьютерных вычислений всегда нужно относиться критически, анализировать их на правдоподобность. Чтобы избежать "подводных камней" при использовании любого стандартного пакета, реализующего численные методы, нужно иметь хотя бы минимальное представление о том, какой именно численный метод реализован для решения той или иной задачи.



В том случае, когда известен интервал, на котором расположен корень, можно воспользоваться иными методами нахождения решения уравнения.

В ***методе Риддера (Ridder’s method)*** вычисляют значение функции в середине интервала . Затем ищут экспоненциальную функцию такую, что Затем применяют метод хорд, используя значения . Очередное значение вычисляют по формуле



(1.5)



***Метод Брента (Brent method)*** соединяет быстроту метода Риддера и гарантированную сходимость метода деления отрезка пополам. Метод использует обратную квадратичную интерполяцию, то есть ищет *x* как квадратичную функцию *y*. На каждом шаге проверяется локализация корня. Формулы метода достаточно громоздки и мы не будем их приводить.

Особые методы применяют для поиска корней полинома. В этом случае могут быть найдены все корни. После того как один из корней полинома найден, степень полинома может быть понижена, после чего поиск корня повторяется.

***Метод Лагерра (Laguerre’s method)*** основывается на следующих соотношениях для полиномов



Предполагают, что корень *x*1 находится на расстоянии *a* от текущего приближения, в то время как все другие корни находятся на расстоянии *b:* . Тогда



,



откуда



Знак перед корнем выбирают с таким расчетом, чтобы получить наибольшее значение знаменателя.

Еще один метод, который применяют для поиска корней полиномов, – ***метод сопровождающей матрицы (companion matrix)***. Можно доказать, что матрица

,



называемая сопровождающей матрицей для полинома , имеет собственные значения равные корням полинома. Напомним, что собственными значениями матрицы называются такие числа **, для которых выполняется равенство или . Существуют весьма эффективные методы поиска собственных значений, о некоторых из них мы будем говорить далее. Таким образом, задачу поиска корней полинома можно свести к задаче поиска собственных значений сопровождающей матрицы.



## 1.1. Функции произвольного вида

Найдем нули функции  на интервале x=[–2,7], используя ***Mathcad***

Изобразим сначала функцию на графике.





На заданном интервале функция три раза обращается в ноль. Определим нули функции, используя встроенную функцию ***root(f(x),x).*** Первый аргумент – функция, нуль которой необходимо найти, второй – переменная, которую необходимо варьировать. (Вообще говоря, функция ***f*** может быть функцией многих переменных и необходимо указывать, по какой именно переменной мы ищем нуль функции.) Кроме того, необходимо задать начальное приближение поиска. Точность вычислений задается встроенной переменной TOL. По умолчанию ее значение равно 0,001. Это значение можно изменить либо через меню ***Math/Built–In Variables*** или непосредственно в тексте документа: 

Задаем начальное приближение: 

И вычисляем корень: 

Если требуется найти несколько корней, как в нашей задаче, то имеет смысл определить новую функцию: 

Функция r(x) возвращает значение корня ближайшее к *x*, то есть начальное приближение мы задаем через аргумент функции. Задаем вектор начальных приближений x и находим соответствующие им корни X:

   

Для данного примера корни легко могут быть найдены аналитически. Они равны на заданном интервале /2/2 и Полученный численный результат с заданной точностью совпадает с точным решением.

Определение новой функции целесообразно и в том случае, когда мы хотим исследовать зависимость решения от параметра. Пусть функция зависит от параметра a



Первый аргумент функции z задает значение параметра, второй – начальное приближение. Найдем корни уравнения при значениях параметра 1 и 2.

 

Если мы хотим получить комплексный корень, то начальное приближение следует задавать комплексным:



## 1.2. Нахождение корней полиномов

Для нахождения корней полиномов имеется встроенная функция ***polyroots(a)***. Аргументом функции является вектор коэффициентов полинома , то есть для уравнения вектор а имеет вид



  

Если в полиноме отсутствуют некоторые степени, то на соответствующих местах следует писать 0. Пусть требуется найти корни полинома



  

Коэффициенты полинома могут быть и комплексными.

## 1.3. Нахождение корней уравнений путем символических преобразований.

Во многих случаях, ***Mathcad*** позволяет найти аналитическое решение уравнения. Для этого необходимо воспользоваться пунктом ***Solve for Variabl***e из пункта меню ***Symbolic***. Для того чтобы найти решение уравнения необходимо записать выражение и выделить в нем переменную (поставить указатель курсора возле переменной). Это необходимо для того, чтобы показать, какая именно величина является переменной, а какая – фиксированным параметром. После этого выбираем из пункта меню ***Symbolic*** подпункт ***Solve for Variable***

 решение готово ––> 

***Обратите внимание!*** В данном случае был найден только один корень, хотя, очевидно, их бесконечно много.

В случае полинома Mathcad, а точнее – встроенный символический процессор Maple – находит все корни.

 –>  Для этого примера найдено 2 корня, хотя они и вырождены. Пример с комплексными корнями:  ––> 

## 1.4 Поиск корней уравнений в Mathcad 2000.

Mathcad 2000 представляет ряд дополнительных возможностей для поиска корней уравнений. Функция **root(*f*(*var1, var2,* ...),*var*1, [a, b])** имеет теперь два необязательных аргумента a и b, которые определяют границы интервала, на котором следует искать корень. На концах интервала [a,b] функция f должна менять знак (*f(a)f(b)*<0). Задавать начальное приближение для корня не нужно. В данном варианте функция root использует алгоритм Риддера и Брента. Продемонстрируем использование расширенного варианта поиска корней на примере функции



Для оценки местоположения корней построим график этой функции









На интервале [1,8] функция имеет два корня. Mathcad 2000 смог найти только один из них.

Дополнительные возможности появились и для нахождения корней полиномов. Функция ***polyroots*** может использовать два различных алгоритма поиска корней – метод Лагерра и метод сопровождающей матрицы. Переключение методов осуществляется в контекстном меню, которое вызывается нажатием правой кнопки мыши, когда указатель установлен на имя функции.

# Глава 2. Решение систем уравнений и неравенств.

## ВВЕДЕНИЕ

Формально задача поиска решения системы уравнений



может быть записана точно так же, как и задача поиска корня одного уравнения , где . Вблизи точки каждая из функций может быть разложена в ряд Тейлора



или в векторной форме , где **J** – матрица Якоби с элементами



Ограничиваясь только первыми двумя членами разложения и полагая, что , получаем уравнение . Таким образом, мы получаем схему для уточнения решения системы уравнений, аналогичную методу Ньютона для случая одного уравнения



(2.1)



Поскольку вычислять матрицу Якоби на каждом шаге достаточно трудоемко, то обычно ее элементы вычисляют приближенно или используют одни и те же значения на нескольких шагах. Одну из разновидностей метода Ньютона – метод Левенберга-Маркардта – использует Mathcad.

## 2.1. Решение систем линейных и нелинейных уравнений и неравенств.

Системы линейных и нелинейных уравнений и неравенств позволяет решать на ***Mathcad*** блок ***given*** в сочетании с функцией ***Find***.

***Внимание!*** В блоке ***given*** записывается система уравнений и/или неравенств, подлежащих решению.

Система уравнений и/или неравенств должна быть записана после или правее слова ***given.***

При записи уравнений вместо знака ***=*** следует набирать ***Ctrl+=***

Перед словом ***given*** необходимо указывать начальные приближения для всех переменных.

Блок ***given*** не пригоден для поиска индексированных переменных.

Если мы хотим найти комплексный корень, следует задавать комплексное начальное приближение.

Признаком окончания системы служит функция ***Find***, если мы хотим найти точное решение системы, либо функция ***Minerr***, если система не может быть решена точно, и мы хотим найти наилучшее приближение, обеспечивающее минимальную погрешность.

Функции ***Minerr*** и ***Find*** должны иметь столько же или меньше аргументов, сколько уравнений и неравенств содержит блок ***given.*** Если окажется, что блок содержит слишком мало уравнений или неравенств, то его можно дополнить тождествами или повторяющимися выражениями.

В том случае, если решение не может быть найдено при заданном выборе начального приближения, появится сообщение в красной рамке ***Did not find solution*** – решение не найдено.

Зададим начальные приближения и решим систему нелинейных уравнений.

 

   

Если необходимо найти решение при различных начальных приближениях, имеет смысл определить новую функцию

   

***Обратите внимание!*** В этом случае не нужно задавать начальные приближения перед началом блока ***given – Find***. Начальные приближения задаются в качестве аргументов функции *f(x,y)*

 

 

Подобным же образом можно решать системы, зависящие от параметра.

   

 

 

## 2.2. Решение систем линейных уравнений и неравенств.

Совершенно аналогично решаются системы линейных уравнений.

Однако в том случае, когда система линейных уравнений невырождена, то есть ее определитель отличен от нуля, более изящным (хотя и не самым эффективным с точки зрения вычислительной математики) является матричный способ решения.

Решим линейную систему одним и другим методом.

  

   



Решим ту же самую систему матричным методом.

Запишем матрицу системы и вектор свободных членов.

 

Решим систему, умножая слева столбец свободных членов b на матрицу

обратную матрице a.

 

Если Вы работаете с продвинутой версией ***Mathcad***, то для этих же целей можно воспользоваться встроенной функцией ***lsolve***



## 2.3. Символическое решение систем уравнений

Во многих случаях решение системы уравнений может быть найдено не только численно, но и аналитически. Для этого так же используется блок ***given*** и функция ***Find***, но вместо знака равенства после функции следует поставить знак символического преобразования –> (Ctrl+.).









Решение записано в виде матрицы. Каждый столбец соответствует паре (x,y), то есть найдены решения (1,3) и (3,1).

## 2.4. Нахождение экстремумов функций

Для нахождения экстремумов функций многих переменных существует две альтернативные возможности.

Первая заключается в использовании блока ***given*** и функции ***minerr .***

Определим функцию двух переменных



Зададимся целью найти ее экстремум в области x=[–5,5] y=[–5,5]. Оценим по графику положение экстремума.

Заносим в матрицу М значения функции в узловых точках

  



На заданном интервале функция не превосходит 25. Зададим начальные приближения для поиска экстремума

 

Записываем блок уравнений или неравенств. Число уравнений и неравенств в блоке ***given – Find*** должно быть больше и равно числа искомых величин. Если уравнений и неравенств не хватает, то можно просто продублировать одно и то же уравнение или вписать какое–либо тождество, например, 2=2 .

     



Функция ***Minerr*** ищет приближенной решение для системы уравнений и неравенств, записанных в блоке. В данном случае мы получили, что системе уравнений наилучшим образом соответствует точка [0,0]. (Поскольку по умолчанию точность вычислений составляет 0.001, мы округлили результат до 0).

Из графика видно, что значение 26 больше самого большого значения функции в окрестностях точки [1,1], то есть точное решение найти нельзя и функция ***Minerr*** подбирает такое значение x, при котором функция ближе всего к значению 26.

Вторая возможность – поиск нулей первой производной, то есть стандартный математический подход. Для этого можно использовать блок ***given – Find.*** Функция ***Find*** ищет точное решение системы уравнений и неравенств, записанных после слова ***given***.

   

Результаты, полученные различными методами, совпадают, врем счета мало в обоих случаях.

В старших версиях Mathcad’а появилась дополнительная возможность поиска экстремумов с помощью функций ***Minimize*** и ***Maximize***, которые могут быть использованы как сами по себе, так и совместно с блоком given. Аргументы функций: имя функции, экстремум которой ищется, и список ее аргументов.

Определяем функцию двух переменных  и задаем начальные приближения  .

Задаем область поиска максимума внутри блока ***given***







Находим максимум функции в заданной области





В случае функции одной переменной задаем функцию  и начальное приближение 

Находим максимум 

Если мы хотим найти максимальное или минимальное значение функции на некотором интервале, то необходимо определить этот интервал в блоке given







На приведенном ниже графике видно, что первый из найденных максимумов соответствовал случаю, когда производная обращается в ноль; второй максимум лежит на границе интервала.





# Глава 3. Аппроксимация функций

## Введение

***Аппроксимацией*** (приближением) функции называется нахождение такой функции (***аппроксимирующей функции*)**, которая была бы близка заданной. Критерии близости функций и могут быть различные.



В том случае, когда приближение строится на дискретном наборе точек, аппроксимацию называют ***точечной*** или ***дискретной.***

В том случае, когда аппроксимация проводится на непрерывном множестве точек (отрезке), аппроксимация называется ***непрерывной*** или ***интегральной***. Примером такой аппроксимации может служить разложение функции в ряд Тейлора, то есть замена некоторой функции степенным многочленом.

Наиболее часто встречающим видом точечной аппроксимации является ***интерполяция*** (в широком смысле).

Пусть задан дискретный набор точек , называемых ***узлами интерполяции***, причем среди этих точек нет совпадающих, а также значения функции в этих точках. Требуется построить функцию *,* проходящую через все заданные узлы. Таким образом, критерием близости функции является .



В качестве функции обычно выбирается полином, который называют ***интерполяционным полиномом***.



В том случае, когда полином един для всей области интерполяции, говорят, что интерполяция ***глобальная*.**

В тех случаях, когда между различными узлами полиномы различны, говорят о ***кусочной*** или ***локальной интерполяции*.**

Найдя интерполяционный полином, мы можем вычислить значения функции между узлами (провести ***интерполяцию в узком смысле слова***), а также определить значение функции даже за пределами заданного интервала (провести ***экстраполяцию***).



Следует иметь в виду, что точность экстраполяции обычно очень невелика.

## 3.1. ЛОКАЛЬНАЯ ИНТЕРПОЛЯЦИЯ

### 3.1.1. Линейная интерполяция.

Простейшим случаем локальной интерполяции является линейная интерполяция, когда в качестве интерполяционной функции выбирается полином первой степени, то есть узловые точки соединяются прямой линией.

Линейная интерполяция на ***Mathcad***’е осуществляется с помощью встроенной функции ***linterp***.

Пусть требуется провести линейную интерполяцию функции sin(*x*) на интервале [0..6], используя пять узлов интерполяции, и вычислить значения функции в четырех точках *Xk*





Задаем интервал изменения x и число узловых точек

  

Определяем шаг изменения x: 

Вычисляем координаты узлов и значения функции в них:

  

Проводим линейную интерполяцию: 

Вычислим значение интерполяционной функции в заданных точках и сравним их с точными значениями

 

Как видно, результаты интерполяции отличаются от точных значений функции незначительно.

### 3.1.2. Интерполяция сплайнами.

В настоящее время среди методов локальной интерполяции наибольшее распространение получила интерполяция сплайнами (от английского слова ***spline*** – гибкая линейка). При этом строится интерполяционный полином третьей степени, проходящий через все заданные узлы и имеющий непрерывные первую и вторую производные. На каждом интервале интерполирующая функция является полиномом третьей степени



и удовлетворяет условиям

. (3.1)



Если всего *n* узлов, то интервалов – . Значит, требуется определить неизвестных коэффициентов полиномов. Условие (3.1) дает нам *n* уравнений. Условие непрерывности функции и ее первых двух производных во внутренних узлах интервала дает дополнительно уравнений



Всего имеем различных уравнений. Два недостающих уравнения можно получить, задавая условия на краях интервала. В частности, можно потребовать нулевой кривизны функции на краях интервала, то есть .



Задавая различные условия на концах интервала, можно получить разные сплайны.

Решим задачу об интерполяции синуса с помощью сплайнов. Для этого воспользуемся встроенной функцией ***interp(VS,x,y,z)***. Переменные ***x*** и ***y*** задают координаты узловых точек, ***z*** является аргументом функции, ***VS*** определяет тип граничных условий на концах интервала.

Определим интерполяционные функции для трех типов кубического сплайна

 

 

 

Вычисляем значения интерполяционных функций в заданных точках и сравниваем результаты с точными значениями

    

Обратите внимание, что результаты интерполяции различными типами кубических сплайнов практически не отличаются во внутренних точках интервала и совпадают с точными значениями функции. Вблизи краев интервала отличие становится более заметным, а при экстраполяции за пределы заданного интервала различные типы сплайнов дают существенно разные результаты. Для большей наглядности представим результаты на графиках











Убедимся в том, что первые и вторые производные сплайна непрерывны





Но производные более высоких порядков уже не являются непрерывными.

## 3.2. ГЛОБАЛЬНАЯ ИНТЕРПОЛЯЦИЯ

При глобальной интерполяции ищется единый полином для всего интервала. Если среди узлов {*xi,yi*} нет совпадающих, то такой полином будет единственным, и его степень не будет превышать *n*.

Запишем систему уравнений для определения коэффициентов полинома



Определим матрицу коэффициентов системы уравнений

  

Решим систему уравнений матричным методом 

Определим интерполяционный полином 

Представим результаты на графике



Вычислим значения интерполяционного полинома в заданных точках и сравним их с точными значениями

  

Коэффициенты интерполяционного полинома следующие:



***Внимание!*** Из-за накопления вычислительной погрешности (ошибок округления) при большом числе узлов (n>10) возможно резкое ухудшение результатов интерполяции. Кроме того, для целого ряда функций глобальная интерполяция полиномом вообще не дает удовлетворительного результата. Рассмотрим в качестве примера две таких функции. Для этих функций точность интерполяции с ростом числа узлов не увеличивается, а уменьшается. Первым примером является функция .

Построим для нее интерполяционный полином на интервале [–1;1], используя 9 точек.

  



   

Представим результаты на графике.





Второй пример – функция .

Найдем интерполяционный полином, используя заданные выше точки.

  



Убедитесь самостоятельно, что при увеличении числа узлов интерполяции, результаты интерполирования вблизи концов интервала ухудшаются.

## 3.3. МЕТОД НАИМЕНЬШИХ КВАДРАТОВ

### ВВЕДЕНИЕ

Наиболее распространенным методом аппроксимации экспериментальных данных является метод наименьших квадратов. Метод позволяет использовать аппроксимирующие функции произвольного вида и относится к группе глобальных методов. Простейшим вариантом метода наименьших квадратов является аппроксимация прямой линией (полиномом первой степени). Этот вариант метода наименьших квадратов носит также название линейной регрессии.

Критерием близости в методе наименьших квадратов является требование минимальности суммы квадратов отклонений от аппроксимирующей функции до экспериментальных точек:

***.***



Таким образом, не требуется, чтобы аппроксимирующая функция проходила через все заданные точки, что особенно важно при аппроксимации данных, заведомо содержащих погрешности.

Важной особенностью метода является то, что аппроксимирующая функция может быть произвольной. Ее вид определяется особенностями решаемой задачи, например, физическими соображениями, если проводится аппроксимация результатов физического эксперимента. Наиболее часто встречаются аппроксимация прямой линией (линейная регрессия), аппроксимация полиномом (полиномиальная регрессия), аппроксимация линейной комбинацией произвольных функций. Кроме того, часто бывает возможно путем замены переменных свести задачу к линейной (провести линеаризацию). Например, пусть аппроксимирующая функция ищется в виде . Прологарифмируем это выражение и введем обозначения, . Тогда в новых обозначениях задача сводится к отысканию коэффициентов линейной функции .



### 3.3.1. Аппроксимация линейной функцией

Применим метод наименьших квадратов для аппроксимации экспериментальных данных.

Читаем данные из файлов datax и datay

 

При использовании MathCAD 2000 имя файла следует заключать в кавычки и записывать его по правилам MS DOS, например, READPRN("c:\mylib\datax.prn").

Определяем количество прочитанных данных (число экспериментальных точек).

 

Используем встроенные функции ***slope*** и ***intercept*** для определения коэффициентов линейной регрессии (аппроксимация данных прямой линией). Функция ***slope*** определяет угловой коэффициент прямой, а функция ***intercept*** – точку пересечения графика с вертикальной осью.

 

Определяем аппроксимирующую функцию: 

Коэффициенты линейной регрессии –

 

Mathcad 2000 предлагает для этих же целей использовать функцию ***line***



Вычислим стандартное отклонение.

 





### 3.3.2. Аппроксимация полиномами

Теперь попытаемся подобрать полиномы второй и третьей степени, в качестве аппроксимирующей функции. Для этих целей служат встроенные функции ***regress*** и уже знакомая нам функция ***interp***. (Очевидно, что если в качестве аппроксимирующей функции брать полином степени на единицу меньше числа точек, то задача сведется к задаче глобальной интерполяции и полученный полином будет точно проходить через все заданные узлы.)

Вводим степени полиномов:

 

Функция ***regress*** является вспомогательной, она подготавливает данные, необходимые для работы функции ***interp***. Вектор ***vs*** содержит, в том числе, и коэффициенты полинома

 

Функция ***interp*** возвращает значение полинома в точке ***z***. Определив новые функции *f2, f3*, мы получили возможность находить значение полинома в любой заданной точке.

 





Коэффициенты:





 

 

Стандартные отклонения почти не отличают друг от друга, коэффициент при четвертой степени z невелик, поэтому дальнейшее увеличение степени полинома нецелесообразно и достаточно ограничиться только второй степенью.

К сожалению, функция ***regress*** имеется далеко не во всех версиях ***Matcad***'а. Однако, провести полиномиальную регрессию можно и без использования этой функции. Для этого нужно определить коэффициенты нормальной системы и решить полученную систему уравнений, например, матричным методом.

Теперь попытаемся аппроксимировать экспериментальные данные полиномами степени *m* и *m1,* не прибегая к помощи встроенной функции ***regress***.

  

  

Вычисляем элементы матрицы коэффициентов нормальной системы

 

и столбец свободных членов

 

Находим коэффициенты полинома, решая систему матричным методом,

 

Определяем аппроксимирующие функции

 

Коэффициенты полиномов следующие:

 

Вычислим стандартное отклонение

 

 





### 3.3.3. Аппроксимация линейной комбинацией функций

***Mathcad*** предоставляет пользователям встроенную функцию ***linfit*** для аппроксимации данных по методу наименьших квадратов линейной комбинацией произвольных функций.

Функция ***linfit*** имеет три аргумента:

* вектор ***x*** – *x*–координаты заданных точек,
* вектор ***y*** – *y*–координаты заданных точек,
* функция ***F*** – содержит набор функций, который будет использоваться для построения линейной комбинации.

Задаем функцию F (аппроксимирующая функция ищется в виде : 







Определяем аппроксимирующую функцию: 

Вычисляем дисперсию:







### 3.3.4. Аппроксимация функцией произвольного вида

Теперь построим аппроксимирующую функцию дробно–рационального типа *.* Для этого воспользуемся функцией ***genfit***. Функция имеет следующие параметры:



* ***x, y*** – векторы, содержащие координаты заданных точек,
* ***F*** – функция, задающая искомую функциональную *n*–параметрическую зависимость и частные производные этой зависимости по параметрам.
* ***v*** – вектор, задающий начальные приближения для поиска параметров.





 

Поскольку нулевой элемент функции *F* содержит искомую функцию, определяем функцию следующим образом: 

Вычисляем среднее квадратичное отклонение

 



Функция ***genfit*** имеется не во всех реализациях ***Mathcad***'а. Возможно, однако, решить задачу, проведя линеаризацию.

Заданная функциональная зависимость может быть линеаризована введением переменных и . Тогда *.*



Определим матрицы коэффициентов нормальной системы (см. книгу [8] из списка литературы)

 

Находим коэффициенты функции, решая систему матричным методом,

 

Определяем функцию: 

Вычислим стандартное отклонение

 

***Обратите внимание!*** Мы получили другие коэффициенты! Вспомните, задача на нахождение минимума нелинейной функции, особенно нескольких переменных, может иметь несколько решений.

Стандартное отклонение больше, чем в случае аппроксимации полиномами, поэтому следует остановить свой выбор на аппроксимации полиномом.

Представим результаты аппроксимации на графиках



В тех случаях, когда функциональная зависимость оказывается достаточно сложной, может оказаться, что самый простой способ нахождения коэффициентов – минимизация функционала Ф "в лоб".

# Глава 4. Вычисление определенных интегралов

## 4.1. Метод Ромберга

Пусть требуется вычислить определенный интеграл на интервале [a;b].



Далеко не всегда задача может быть решена аналитически. В частности, численное решение требуется в том случае, когда подынтегральная функция задана таблично. Для численного интегрирования подынтегральную функцию аппроксимируют какой-либо более простой функцией, интеграл от которой может быть вычислен. Обычно в качестве аппроксимирующей функции используют полином. В случае полинома нулевой степени метод численного интегрирования называют ***методом прямоугольников***, в случае полинома первой степени – ***методом трапеций***, в случае полинома второй степени – ***методом Симпсона***. Все эти методы являются частными случаями ***квадратурных формул Ньютона-Котеса***.

Итак, в методе трапеций подынтегральную функцию аппроксимируют полиномом первой степени, то есть прямой линией. Это значит, что вместо площади криволинейной трапеции мы будем искать площадь прямоугольной трапеции. Приближенное значение интеграла равно



Погрешность этой формулы равна .



Обозначим , где . Смысл введенного обозначения станет ясен несколько позже.



Оценку значения интеграла можно сделать более точной, если разбить интервал на *n* частей и применить формулу трапеций для каждого такого интервала

.



Если разбить интервал на две части, то есть уменьшит шаг в два раза , то оценка для величины интеграла будет иметь вид



В данном случае суммирование включает только один элемент. Обратите внимание, в новую оценку вошла старая оценка. Нам потребовалось определять значение функции только в новых узлах.

Если имеется 2n подынтервалов, то



Если n=0, то



Если n=1, то



Если n=2, то



Вообще, справедливо рекуррентное соотношение



Полученное соотношение называют рекурсивной формулой трапеций и часто применяют для вычисления определенных интегралов. Преимущество этой формулы состоит в том, что при увеличении числа подынтервалов функцию нужно вычислять только во вновь добавленных точках. К сожалению, с помощью этой формулы нельзя получить сколь угодно точное значение интеграла. Во-первых, при увеличении числа разбиений объем вычислений стремительно возрастает; во-вторых, на каждом шаге накапливается ошибка округлений. Для дальнейшего уточнения значения интеграла можно сделать следующий шаг – ***экстраполировать*** полученную последовательность значений на случай бесконечного числа точек или что то же самое, на случай нулевого шага. Такой подход называется ***методом Ромберга***.

Метод Ромберга заключается в том, что полученные оценки значения интеграла экстраполируют на случай бесконечного числа разбиений (величины шага равной нулю) по рекуррентной формуле

(4.1)



То есть строится следующий треугольник

*R*(1,1)

*R*(2,1) *R*(2,2)

*R*(3,1) *R*(3,2) *R*(3,3)

*R*(4,1) *R*(4,2) *R*(4,3) *R*(4,4)

*R*(5,1) *R*(5,2) *R*(5,3) *R*(5,4) *R*(5,5) ,

в котором первый столбец состоит из значений интеграла, полученных при последовательном удвоении числа интервалов. Второй столбец – результат уточнения значений первого столбца по рекуррентной формуле (4.1). Третий столбец – уточненные значения интеграла на основе второго столбца и т.д.

Формула (4.1) может быть получена различными способами. Можно, например, воспользоваться методом Невиля. Пусть имеется набор точек . Обозначим полином нулевой степени, проходящий через *i*-ю точку. Обозначим полином первой степени, проходящий через точки *i* и *i*+1. Совершенно аналогично будет означать полином *n*–1 степени, проходящий через все *n* точек. Легко убедиться, что



В нашем случае . В качестве выступают . Мы хотим получить значение интеграла в пределе , поэтому .



## 4.2 Вычисление определенных интегралов

Для вычисления определенного интеграла необходимо выбрать знак интеграла из палитры или набрать его нажатием клавиши &. После этого следует вписать пределы интегрирования, подынтегральную функцию и переменную интегрирования. Mathcad успешно справляется с большинством интегралов, в том числе, с несобственными. Точность вычислений регулируется встроенной переменной TOL. По умолчанию ее значение установлено . Ниже приводится несколько примеров успешного вычисления несобственных интегралов, интеграла от быстро осциллирующей функции и интеграла от ступенчатой функции.



 

 

Здесь



Зависимость результата от заданной точности вычислений представлена ниже







Для этого примера результат может быть получен также в символьном виде. Для этого вместо знака равенства необходимо нажать знак символического равенства Ctrl+.



В то же время в некоторых случаях несобственные интегралы вычисляются неправильно.



Хотя очевидно, что



# Глава 5. Решение дифференциальных уравнений

## 5.1. Обыкновенные дифференциальные уравнения

### Введение

Пусть необходимо найти решение уравнения

(5.1)



с начальным условием . Такая задача называется ***задачей Коши***. Разложим искомую функцию в ряд вблизи точки и ограничимся первыми двумя членами разложения . Учтя уравнение (5.1) и обозначив , получаем Эту формулу можно применять многократно, находя значения функции во все новых и новых точках.



(5.2)



Такой метод решения обыкновенных дифференциальных уравнений называется ***методом Эйлера***. Геометрически метод Эйлера означает, что на каждом шаге мы аппроксимируем решение (интегральную кривую) отрезком касательной, проведенной к графику решения в начале интервала. Точность метода невелика и имеет порядок *h*. Говорят, что метод Эйлера – метод первого порядка, то есть его точность растет линейно с уменьшением шага *h*.

Существуют различные модификации метода Эйлера, позволяющие увеличить его точность. Все они основаны на том, что производную, вычисленную в начале интервала, заменяют на среднее значение производной на данном интервале. Среднее значение производной можно получить (конечно же только приближенно) различными способами. Можно, например, оценить значение производной в середине интервала и использовать его для аппроксимации решения на всем интервале



Можно также оценить среднее значение производной на интервале



Такие модификации метода Эйлера имеет уже точность второго порядка.

Оценку значения производной можно улучшить, увеличивая число вспомогательных шагов. На практике наиболее распространенным методом решения обыкновенных дифференциальных уравнений ***является метод Рунге-Кутты*** четвертого порядка. Для оценки значения производной в этом методе используется четыре вспомогательных шага. Формулы метода Рунге-Кутты следующие



Перечисленные методы можно применять и для решения систем дифференциальных уравнений. Поскольку многие дифференциальные уравнения высших порядков могут быть сведены заменой переменных к системе дифференциальных уравнений первого порядка, рассмотренные методы могут быть использованы и для решения дифференциальных уравнений порядка выше первого.

Еще один тип задач, часто встречающихся на практике, – краевые задачи. Пусть имеется дифференциальное уравнение второго порядка . Решение уравнения требуется найти на интервале , причем известно, что Понятно, что произвольный интервал заменой переменных может быть сведен к единичному. Для решения краевой задачи обычно применяют ***метод стрельб***. Пусть где *k* – некоторый параметр. Для некоторого пробного значения *k* может быть решена задача Коши, например, методом Рунге-Кутты. Полученное решение будет зависеть от значения параметра . Мы хотим найти такое значение параметра, чтобы выполнялось условие . Фактически мы свели исходную задачу к задаче решения трансцендентного уравнения с таблично заданной функцией. Если найдены такие значения параметра *k*1 и *k*2, что , то дальнейшее уточнение значения параметра можно проводить методом деления отрезка пополам.



### 5.1.1. Метод Эйлера для дифференциальных уравнений первого порядка

Решим задачу Коши для дифференциального уравнения первого порядка методом Эйлера.



Пусть правая часть уравнения равна 

Зададим границы изменения ***x:***  

Зададим число точек и величину шага:  

Зададим начальные условия:  

Вычислим ***x*** и ***y*** по формулам Эйлера 





Представим результат графически и сравним его с аналитическим решением 

 



Точное аналитическое решение и решение, полученное численно, отличаются в точке x=1 на 

То есть относительная ошибка составляет 

### 5.1.2. Решение систем дифференциальных уравнений.

Для решения дифференциальных уравнений ***Mathcad*** имеет ряд встроенных функций, в частности, функцию ***rkfixed***, реализующую метод Рунге–Кутты четвертого порядка с фиксированным шагом. Фактически эта функция предназначена для решения систем дифференциальных уравнений первого порядка.



Функция ***rkfixed(y, x1, x2, npoints, D)*** возвращает матрицу. Первый столбец этой матрицы содержит точки, в которых получено решение, а остальные столбцы – решения и его первые производные.



Аргументы функции:

* ***y*** – вектор начальных значений (***n*** элементов).
* ***x1*** и ***x2*** – границы интервала, на котором ищется решение дифференциального уравнения.
* ***npoints*** – число точек внутри интервала ***(x1,x2)***, в которых ищется решение. Функция ***rkfixed*** возвращает матрицу, состоящую из ***1+npoints*** строк.
* ***D*** – вектор, состоящий из n элементов, который содержит первые производные искомой функции.

В качестве примера рассмотрим решение системы Вольтерры–Лотки. Эта система описывает динамику численности хищников и жертв на замкнутом ареале и является одной из базовых моделей экологии.



Для решения систем дифференциальных уравнений используются функция ***rkfixed.***

***Внимание!*** В этом примере установлено значение ***ORIGIN=1***, то есть нумерация элементов массива начинается с 1, а не с 0, как это принято в ***Mathcad***'е по умолчанию.

Пусть в начальный момент времени число хищников и число жертв



Задаем вектор начальных значений 

параметры системы  

интервал времени и количество точек, в которых будет вычислено решение  

и вектор правых частей системы. (Поскольку исходная система не зависит явно от времени ***t***, функция ***D*** так же не зависит от времени явно хотя и содержит его в числе своих аргументов.)



Решаем систему с помощью встроенной функции





Представим на графике результаты расчета – зависимость численности популяций от времени



и зависимость числа жертв от числа хищников



Можно использовать обозначения  или  – это одно и то же.

Поскольку дифференциальное уравнение порядка выше первого может быть преобразовано к системе дифференциальных уравнений первого порядка, функция ***rkfixed*** может быть использована и для решения дифференциальных уравнений

### 5.1.3. Решение дифференциальных уравнений методом Рунге–Кутты

Решим еще раз задачу Коши для дифференциального уравнения первого порядка методом Рунге–Кутты.



Зададим границы изменения *x*:  

Зададим число точек внутри интервала 

Зададим начальные условия 

***Обратите внимание на обозначения!*** Поскольку мы решаем только одно дифференциальное уравнение первого порядка, а не систему дифференциальных уравнений, матрица ***y*** содержит только один элемент, однако запись ***y=1*** была бы неправильной. Необходимо явно указывать на то, что величина ***y*** – матрица, то есть писать индекс.

Определим теперь матрицу производных. Эта матрица тоже состоит только из одного элемента. Этот элемент с точностью до обозначений совпадает с правой частью исходного дифференциального уравнения: 

Решаем дифференциальное уравнение 

  



Точное аналитическое решение и решение, полученное численно отличаются в точке на 



Относительная ошибка составляет 

### 5.1.4. Решение дифференциальных уравнений второго порядка

В качестве примера решим задачу о гармоническом осцилляторе, для которого известно аналитическое решение, и легко может быть оценена точность вычислений. Дифференциальное уравнение второго порядка



преобразуем к системе из двух дифференциальных уравнений первого порядка



Пусть декремент затухания 

Пусть циклическая частота 

Зададим начальные условия 

***y0*** соответствует начальной координате, а ***y1*** – начальной скорости. Зададим теперь матрицу ***D***. С учетом того, что искомая величина соответствует нулевому элементу массива ***y***, ее первая производная – первому, а вторая – второму, имеем





Представим результаты расчета на графике и сравним их с аналитическим решением 

 



Для контроля точности вычислений нарисуем фазовую траекторию (зависимость смещения от скорости). Для гармонического осциллятора фазовая траектория должна иметь вид эллипса.



***Примечание***: ***Mathcad*** имеет еще две функции для решения задачи Коши. Это функции ***Rkadapt*** и ***Bulstoer***. Эти функции имеют те же самые аргументы и возвращают решения в такой же форме, что и функция ***rkfixed***. Первая из этих функций использует метод Рунге–Кутты с переменным шагом, что позволяет повысить точность вычислений и сократить их объем, если искомое решение имеет области, где ее значения меняются быстро, и области плавного изменения. Функция ***Rkadapt*** будет варьировать величину шага в зависимости от скорости изменения решения.

Функция ***Bulstoer*** реализует иной численный метод – метод Булирша–Штёра. Ее следует применять, если известно, что решение является гладкой функцией.

### 5.1.5. Решение краевой задачи.

Пусть требуется найти решение дифференциального уравнения при условиях и .



При значениях параметров  

Для решения краевой задачи имеется встроенная функция ***sbval***, реализующая метод стрельб и позволяющая свести краевую задачу к задаче Коши. Функция ***sbval*** имеет следующие параметры:

* ***v***– вектор, содержащий начальные приближения для недостающих начальных условий,
* ***xmin, xmax*** – границы интервала, на котором ищется решение,
* ***D(x,y)*** – вектор–функция, содержащий правые части системы дифференциальных уравнений первого порядка, эквивалентной исходному уравнению, размер вектора ***n*** совпадает со степенью старшей производной дифференциального уравнения
* ***load(xmin,v) –*** вектор–функция, элементы которой соответствуют ***n*** значениям функций на левой границе интервала. Часть этих значений известна, а для части заданы начальные приближения в векторе ***v***. Их уточненные значения будут найдены в процессе вычисления
* ***score(xmax,y)*** – вектор–функция, имеющая то же число элементов, что и ***v***. Каждое значение является разностью между начальными значениями в конечной точке интервала и соответствующей оценки для решения. Этот вектор показывает, на сколько близко найденное решение к истинному.

Наша задача сводится к системе двух дифференциальных уравнений первого порядка:



Поэтому функция D имеет вид 

Задаем граничные условия:  

Задача Коши для дифференциального уравнения второго порядка содержит два начальных условия. Нам известно только одно. Начальное приближение для недостающего значения задаем в векторе ***v***, который в нашем случае состоит только из одного элемента. Несмотря на это, индекс о должен быть обязательно указан, чтобы подчеркнуть векторный характер этой величины: 

На левой границе интервала нам известно значение и задано начальное приближение для . Это значение записано в . Задаем вектор-функцию ***load***. Ее нулевой элемент – начальное значение для , первый – для .











Теперь, когда нам стало известно недостающее начальное условие в задаче Коши, можно воспользоваться, например, функцией ***rkfixed***

  





### 5.1.6. Решение обыкновенных дифференциальных уравнений в Mathcad 2000

Mathcad 2000 предлагает новый способ для решения обыкновенных дифференциальных уравнений, разрешенных относительно старшей производной. Для этих целей служит уже известный нам блок ***given*** совместно с функцией ***odesolve***. Дифференциальное уравнение совместно с начальными или граничными условиями записывается в блоке ***given***. Производные можно обозначать как штрихами (***Ctrl+F7***), так и с помощью знака производной . Пример использования функции для решения задачи Коши приведен ниже.



















**Обратите внимание!** У искомой функции явно указан аргумент, знак производной стоит перед скобкой.

Функция ***odesolve*** имеет три аргумента. Первый аргумент – независимая переменная, вторая – граница интервала, на котором ищется решение, последний аргумент – шаг, с которым ищется решение. Последний аргумент может быть опущен.

Следующий пример демонстрирует решение краевой задачи. Использован другой способ записи производных, используется ***odesolve*** функция с двумя аргументами.















## 5.2. Решение уравнений в частных производных.

### Введение

Одним из методов решения дифференциальных уравнений в частных производных является ***метод сеток***. Идея метода заключается в следующем. Для простоты, ограничимся случаем только функции двух переменных, и будем полагать, что решение уравнения ищется на квадратной области единичного размера. Разобьем область сеткой. Шаг сетки по оси x и по оси y, вообще говоря, может быть разный. По определению частная производная равна



Если рассматривать функцию только в узлах сетки, то частную производную можно записать в форме



где узел соответствует точке . Полученное выражение называется ***правой конечной разностью***. Название связано с тем, что для вычисления производной в точке используются значение функции в этой точке и точке, лежащей правее. Очевидно, что сходное выражение можно было бы получить, используя точку, лежащую слева.



Такое выражение называется ***левой конечной разностью***. Можно получить центральную конечную разность, найдя среднее этих выражений.

Теперь получим выражения для вторых производных.



В данном случае для нахождения производной мы использовали симметричные точки. Однако, очевидно, можно было бы использовать точки с несимметричным расположением.

### 5.2.1. Уравнения гиперболического типа

В качестве примера рассмотрим решение волнового уравнения (уравнения гиперболического типа).



Уравнение будем решать методом сеток. Запишем уравнение в конечных разностях



Полученное уравнение позволяет выразить значение функции u в момент времени через значения функции в предыдущие моменты времени.



Такая разностная схема называется ***явной***, так как искомая величина получается в явном виде. Она устойчива, если.



Зададим начальные условия: смещение струны ***U*** в начальный и последующий моменты времени описывается синусоидальной функцией.

  

 

(Совпадение смещений при j=0 и j=1 соответствует нулевой начальной скорости.)

Зададим граничные условия: на концах струны смещение равно 0 в любой момент времени  

Будем полагать коэффициент  

  

Записываем уравнение в конечных разностях, разрешенное относительно





Представляем результат на графике



### 5.2.2. Уравнения параболического типа.

Еще один пример использования конечных разностей – уравнение диффузии.



Это уравнение параболического типа. Явная разностная схема для этого уравнения имеет вид

(5.3)



Эта разностная схема устойчива, если . Для краткости в дальнейшем мы будем обозначать весь множитель, стоящий перед скобкой, как **.



Задаем коэффициент    
и диапазон изменения пространственной и временной координат:

 

Задаем начальные и граничные условия

   

Уравнение в конечных разностях имеет вид



Представляем результаты на графике. (Для большей наглядности изображена только центральная часть)





Основное достоинство явных методов – их простота – зачастую сводится на нет достаточно жесткими ограничениями на величину шага. Явные схемы обычно устойчивы при столь малых шагах по времени, что они становятся непригодными для практических расчетов. Этого существенного недостатка позволяют избежать неявные схемы. Свое название они получили потому, что значения искомой функции на очередном временном шаге не могут быть явно выражены через значения функции на предыдущем шаге.

Рассмотрим применение неявной схемы на примере уравнения теплопроводности

(5.4)



Запишем неявную разностную схему для этого уравнения

(5.5)



Здесь первый индекс соответствует пространственной, а второй – временной координате. В отличие от явной схемы, для вычисления в правой части уравнения используются значения функции на том же самом временном шаге. Вводя обозначение , уравнение (5.5) можно переписать в виде



(5.6)



или в матричной форме

(5.7)



где .



Задаем количество узлов сетки (в данном случае оно одинаково для обеих переменных) 

   

Задаем значения параметров   

и начальное распределение температуры в области



Формируем матрицы уравнения (5.7)

 

  

   

Находим решение системы 



### 5.2.3. Решение уравнений Лапласа и Пуассона.

Для решения уравнений Пуассона и Лапласа (частный случай, когда ) – уравнений эллиптического типа – предназначена функция ***relax(a, b, c, d, e, f, u, rjac)***, реализующая метод релаксации. Фактически, эту функцию можно использовать для решения эллиптического уравнения общего вида



которое может быть сведено к уравнению в конечных разностях



В частности, для уравнения Пуассона коэффициенты .



Идея метода релаксации заключается в следующем. Если нет источников (уравнение Лапласа), то значение функции в данном узле на текущем шаге определяется как среднее значение функции в ближайших узлах на предыдущем шаге *k*



(5.8)



При наличии источников разностная схема имеет вид (5.9)



Метод релаксации сходится достаточно медленно, так как фактически он использует разностную схему (5.3) с максимально возможным для двумерного случая шагом .



В методе релаксации необходимо задать начальное приближение, то есть значения функции во всех узлах области, а так же граничные условия.

Функция ***relax*** возвращает квадратную матрицу, в которой:

1. расположение элемента в матрице соответствует его положению внутри квадратной области,
2. это значение приближает решение в этой точке.

Эта функция использует метод релаксации для приближения к решению.

Вы должны использовать функцию ***relax,*** если Вы знаете значения искомой функции ***u(x, y)*** на всех четырех сторонах квадратной области.

Аргументы:

***a, b, c, d, e*** – квадратные матрицы одного и того же размера, содержащие коэффициенты дифференциального уравнения.

***f –*** квадратная матрица, содержащая значения правой части уравнения в каждой точке внутри квадрата

***u*** – квадратная матрица, содержащая граничные значения функции на краях области, а также начальное приближение решения во внутренних точках области.

***rjac*** – Параметр, управляющий сходимостью процесса релаксации. Он может быть в диапазоне от 0 до 1, но оптимальное значение зависит от деталей задачи.

  

Задаем правую часть уравнения Пуассона – два точечных источника

  

Задаем значения параметров функции ***relax***

     

Задаем граничные условия и начальное приближение – нули во всех внутренних точках области

    

Находим решение 

и представляем его графически в виде поверхности и линий уровней.





Если граничные условия равны нулю на всех четырех сторонах квадрата, можно использовать функцию ***multigrid***.



Алгоритм метода достаточно громоздкий, поэтому рассматривать его мы не будем.

# Глава 6. Статистические расчеты на Mathcad

## 6.1. Генерация чисел, распределенных равномерно.

***Mathcad*** дает возможность обрабатывать статистическую информацию. Создадим с помощью датчика случайных чисел ***rnd*** последовательность, подчиняющуюся равномерному распределению.

  

Представим последовательность чисел графически



Вычислим среднее значение  ,

стандартное отклонение  ,

и дисперсию  

Проверим, насколько хорошо подчиняются наши числа закону равномерного распределения, построив гистограмму. Определим интервалы для группировки данных:  . И подсчитаем, сколько раз попадают числа из нашей последовательности в каждый из интервалов



 



Числа fi показывают, сколько раз числа из массива x попадают в каждый из определенных нами интервалов 0–2, 2–4, 4–6, 6–8, 8–10

***Внимание!*** В массиве ***f*** элементов на 1 меньше, чем в массиве ***interval***.

Определим границы для построения гистограммы.



 



***Внимание!*** При построении гистограммы следует использовать трассировку типа ***bar,* п**ри этом столбцы центрируются относительно начала интервала. Для центрирования столбцов относительно середины интервала используйте трассировку ***step.***

Повторим построение гистограммы, используя большее число интервалов.

 

 

 

 





Числа fi показывают, сколько раз числа из массива x попадают в каждый из определенных нами интервалов 0–1, 1–2, 2–3, 3–4, 4–5, 5–6, 6–7, 7–8, 8–9, 9–10

## 6.2. Генерация случайных чисел, распределенных по нормальному закону.

Для преобразования равномерно распределенных случайных чисел в числа, распределенные по нормальному закону, воспользуемся преобразованием:

   



Представим числа на графике

 





Вычислим среднее значение M1:  

И стандартное отклонение S1:  

Определим функцию g(x) – функцию плотности нормального распределения с математическим ожиданием M1 и стандартным отклонением S1





Построим гистограмму, разбив весь интервал на подынтервалы шириной 5 единиц.

 

 

 

 

 



Сравним предельное распределение g(x) с реальным распределением.

Для этого вновь воспользуемся функцией ***hist***

  

 

Подсчитаем плотность вероятности f1:

 

 

 

Представим результаты графически:



Определим вероятность W того, что некоторое число попадет в интервал [1,3], воспользовавшись функцией ***cnorm.***







И сравним полученный результат с результатом, который дает функция ***hist***

 

## 6.3. Вычисление коэффициента корреляции.

Занесем в массив x оценки по высшей математике в сессию в некоторой студенческой группе, а в массив y – оценки по физике. Определим, существует ли какая–либо связь между успеваемостью по физике и высшей математике.



 

Вычислим средние оценки и стандартное отклонение

 

 

 

Для вычисления коэффициента корреляции имеется встроенная функция ***corr***

 

Близкий к 1 коэффициент корреляции свидетельствует о высокой зависимости между успеваемостью по этим предметам.

## 6.4. Применение метода Монте-Карло для вычисления кратных интегралов

Хотя ***Mathcad*** позволяет вычислять кратные интегралы непосредственно, однако в большинстве случаев при кратности интегралов 3 и более применение метода Монте-Карло предпочтительнее. Дело в том, при одинаковой точности метод Монте-Карло дает существенный выигрыш во времени (в десятки и сотни раз), особенно при большой кратности интегралов. Идея метода состоит в том, что интеграл заменяется величиной Fср.·V, где V – объем области интегрирования, Fср. –среднее значение подынтегральной функции, вычисленное по нескольким случайно выбранным точкам.

Определим подынтегральную функцию.



И вычислим интеграл обычным способом (***обратите внимание на время счета!)***



А теперь вычислит тот же интеграл методом Монте-Карло

 

  

 

Поскольку в нашем случае объем области интегрирования равен 1, полученное среднее значение совпадает со значением интеграла. При относительной погрешности в 0.001% время вычисления интеграла по методу Монте-Карло существенно меньше.

Интеграл можно вычислить и другим способом. Заключим область интегрирования внутрь прямоугольной области, "набросаем" внутрь полученной области N случайных точек. Тогда интеграл найдем из соотношения ,где N – общее число точек, n – число точек, лежащих внутри области интегрирования, V – объем области, включающей область интегрирования.



Максимальное значение подынтегральной функции в области интегрирования не превосходит 125, следовательно, мы может заключить всю область интегрирования внутрь четырехмерного цилиндроида высотой 125 и объемом V=125. Сгенерируем N четверок случайных чисел и подсчитаем, сколько из них лежит под поверхностью f(x,y,z).

 

 

# Глава 7. Анализ и синтез сигналов с помощью преобразования Фурье.

Определим функцию, задающую так называемый пилообразный сигнал 

и изобразим ее на графике





Заполним массив s:  

Проводим прямое преобразование Фурье: 

***Внимание!*** В том случае, когда в массиве ***s*** содержится элементов, причем все числа действительные, следует использовать функцию ***fft***. Во всех остальных случаях – функцию ***cfft***. Массив ***g*** содержит комплексные коэффициенты дискретного преобразования Фурье.



Размер массива ***f –***



Для анализа вклада отдельных гармоник в исходный сигнал изобразим на графике модули и аргументы гармоник

 

 

Проводим обратное преобразование Фурье, исключив гармоники с малым вкладом. Будем учитывать только гармоники с амплитудой не менее 0.3. Для отсечения слагаемых с малым вкладом воспользуемся функцией единичного скачка – функцией Хевисайда **Ф**.

 

Для обратного преобразования Фурье используется функция ***ifft,*** если прямое преобразование осуществлялось с помощью ***fft,*** и ***cifft***, если прямое преобразование осуществлялось с помощью ***cfft.***





Повторим преобразование Фурье, учтя слагаемые с амплитудой до 0.1.

  



Учет дополнительных гармоник существенно улучшил результат синтеза сигналов.

# Литература

1. MATHCAD 6.0 PLUS/ Финансовые, инженерные и научные расчеты в среде Windows 95./ Пер. с англ. – М.: Информационно-издательский дом "Филинъ", 1996. – 712 с.
2. Плис А.И., Сливина Н.А. Mathcad: математический практикум. – М.: Финансы и Статистика. – 1999.
3. Очков В.Ф. Mathcad 8 Pro для студентов и инженеров. – М.: КомпьютерПресс, 1999.
4. Очков В.Ф.. MathCad 7 Pro для студентов и инженеров. – М.: КомпьютерПресс, 1998. – 384 с.
5. Дьяконов В.П. Справочник по MathCAD PLUS 6.0 PRO. – М.: CK Пресс, 1997. – 336 с.
6. Дьяконов В.П. Справочник по MathCAD PLUS 7.0 PRO. – М.: CK Пресс, 1998. – 352 c.
7. Дьяконов В. П., Абраменкова И. В. MathCAD 7 в математике, в физике и в Internet. – М.: Нолидж.- 1998. – 352 с.
8. Шелест А. Е. Микрокалькуляторы в физике. – М.: Наука. Гл. ред. физ.-мат. лит., 1988. – 272 с.
9. Шикин Е.В., Плис А.И. Кривые и поверхности на экране компьютера. Руководство по сплайнам для пользователей. – М.: ДИАЛОГ–МИФИ, 1996. – 240 с.
10. Тейлор Дж. Введение в теорию ошибок. Пер. с англ. – М.: Мир, 1985. – 272 с., ил.
11. Ортега Дж., Пул У. Введение в численные методы решения дифференциальных уравнений/ Пер. с англ.; Под ред. А. А. Абрамова. – М.: Наука. Гл. физ.-мат. лит., 1986. – 288 с.
12. Заварыкин В. М., Житомирский В. Г., Лапчик М. П. Численные методы: Учеб. пособие для студентов физ.-мат. спец. пед. ин-тов. – М.; Просвещение, 1990. – 176 с.
13. Демидович Б. П., Марон И. А. Основы вычислительной математики. – М.: Наука. Гл. ред. физ.-мат. лит., 1966.
14. Демидович Б. П., Марон И. А., Шувалова Э. З. Численные методы анализа. – М.: Наука. Физматгиз, 1962.

Содержание

Введение 3

Глава 1. Нахождение корней уравнений 5

Введение 5

1.1. Функции произвольного вида 8

1.2. Нахождение корней полиномов 10

1.3. Нахождение корней уравнений путем символических преобразований. 10

1.4 Поиск корней уравнений в Mathcad 2000. 11

Глава 2. Решение систем уравнений и неравенств. 12

ВВЕДЕНИЕ 12

2.1. Решение систем линейных и нелинейных уравнений и неравенств. 12

2.2. Решение систем линейных уравнений и неравенств. 14

2.3. Символическое решение систем уравнений 14

2.4. Нахождение экстремумов функций 15

Глава 3. Аппроксимация функций 18

Введение 18

3.1. ЛОКАЛЬНАЯ ИНТЕРПОЛЯЦИЯ 18

3.1.1. Линейная интерполяция. 18

3.1.2. Интерполяция сплайнами. 19

3.2. ГЛОБАЛЬНАЯ ИНТЕРПОЛЯЦИЯ 24

3.3. МЕТОД НАИМЕНЬШИХ КВАДРАТОВ 27

ВВЕДЕНИЕ 27

3.3.1. Аппроксимация линейной функцией 27

3.3.2. Аппроксимация полиномами 29

3.3.3. Аппроксимация линейной комбинацией функций 31

3.3.4. Аппроксимация функцией произвольного вида 33

Глава 4. Вычисление определенных интегралов 36

4.1. Метод Ромберга 36

4.2 Вычисление определенных интегралов 38

Глава 5. Решение дифференциальных уравнений 40

5.1. Обыкновенные дифференциальные уравнения 40

Введение 40

5.1.1. Метод Эйлера для дифференциальных уравнений первого порядка 42

5.1.2. Решение систем дифференциальных уравнений. 43

5.1.3. Решение дифференциальных уравнений методом Рунге–Кутты 45

5.1.4. Решение дифференциальных уравнений второго порядка 46

5.1.5. Решение краевой задачи. 47

5.1.6. Решение обыкновенных дифференциальных уравнений в Mathcad 2000 49

5.2. Решение уравнений в частных производных. 51

Введение 51

5.2.1. Уравнения гиперболического типа 51

5.2.2. Уравнения параболического типа. 53

5.2.3. Решение уравнений Лапласа и Пуассона. 55

Глава 6. Статистические расчеты на Mathcad 58

6.1. Генерация чисел, распределенных равномерно. 58

6.2. Генерация случайных чисел, распределенных по нормальному закону. 60

6.3. Вычисление коэффициента корреляции. 63

6.4. Применение метода Монте-Карло для вычисления кратных интегралов 64

Глава 7. Анализ и синтез сигналов с помощью преобразования Фурье. 66

Литература 68