

**Источники и классификация погрешностей**

Погрешность – отклонение приближенного решения от точного или истинного.

Различают 3 типа погрешностей:

А) **Неустранимая погрешность** – связана с приближенным характером исходной содержательной модели. (Например параметрами математической модели, которые являются приближенными величинами = ошибки в исходных данных)

Б) **Погрешность метода** – связана со способом решения поставленной математической задачи и она появляется в результате подмены исходной математической модели другой последовательности более простых моделей = замена точных операторов приближенными (производная = разность и тд)

В) **Вычислительная погрешность**: вычисление погрешности обусловлено округлением промежуточных и конечных данных.

**Значащие и верные цифры**

**Значащей** цифрой называют первую слева-направо, отличную от нуля цифру и все последовательности за ней.

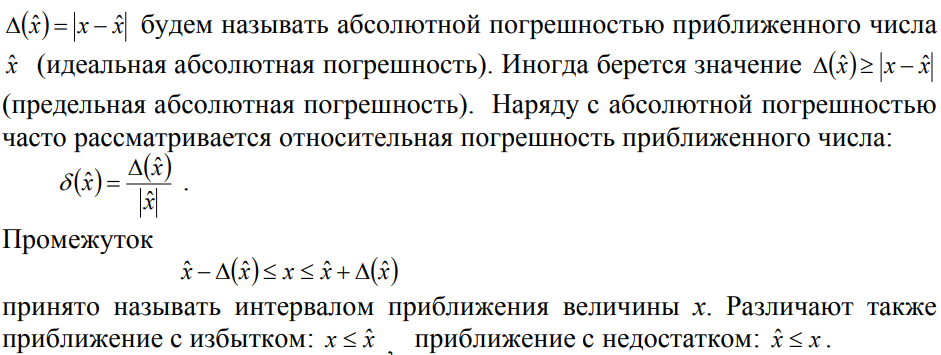
0,07231 – 7,2,3,1 – значащие

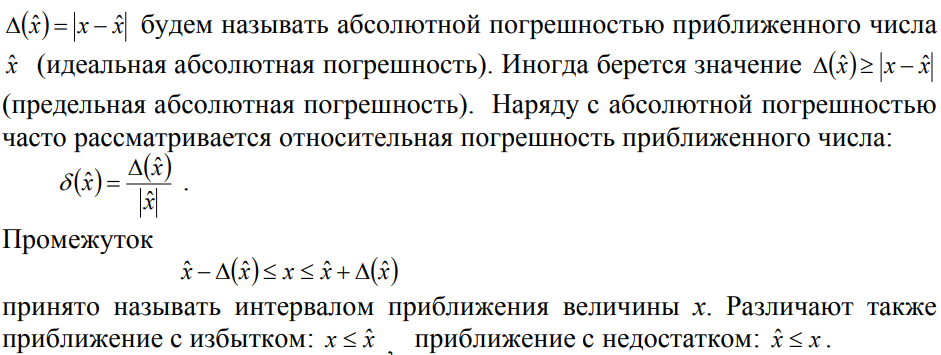
1,23 – 1,2,3 – значащие

Значащую цифру называют **верной**, если погрешность значащего числа не превосходит половину десятичного разряда соответствующей этой цифре.

0,07231 – с погрешностью 0,002 верная цифра 7, а 2,3,1 – сомнительные.

**Абсолютная и относительная погрешность**





**Погрешности арифметических операций**

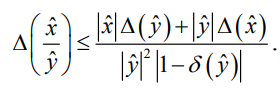
***Погрешности арифметических операций для абсолютной погрешности:***

Сложение и вычитание

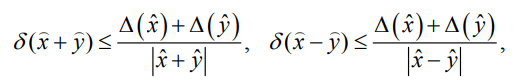


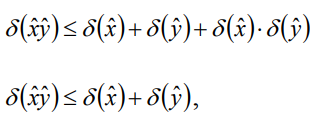


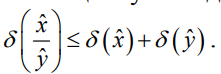
Деление:



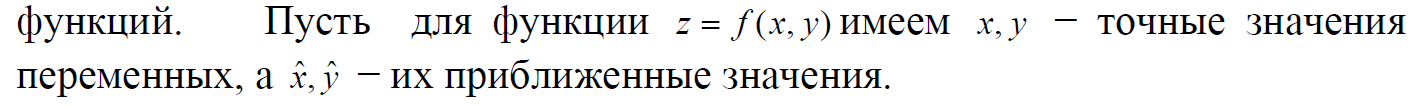
***Погрешности арифметических операций для относительной погрешности:***

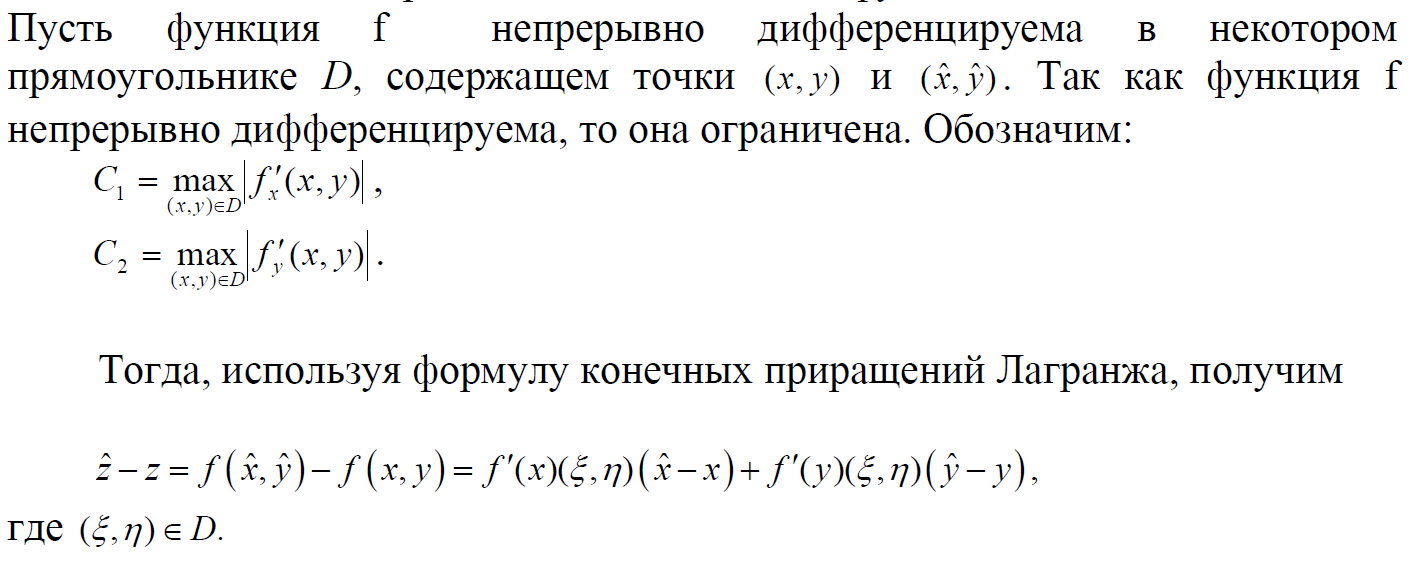
****

****

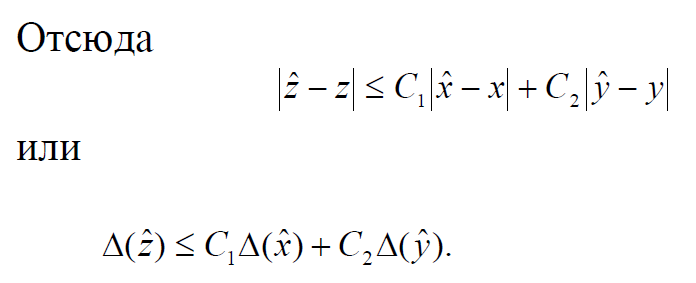
****

***Погрешности функций***

****

****

**f’x(E,N)\*(x^-x)+…**





***Прямая и обратная задачи теории погрешностей***

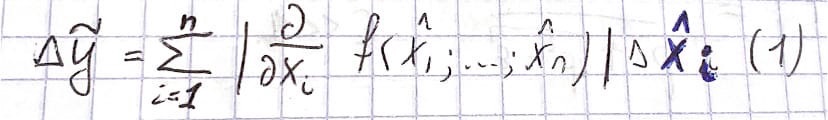
Задача вычисления погрешности функций в случае, когда заданы погрешности аргументов называют прямой задачей теории погрешности (дана погрешность аргументов)

Задача определения допустимой погрешности аргументов по заданной допустимой погрешности функции называют обратной задачей теории погрешности (дана погрешность функции).

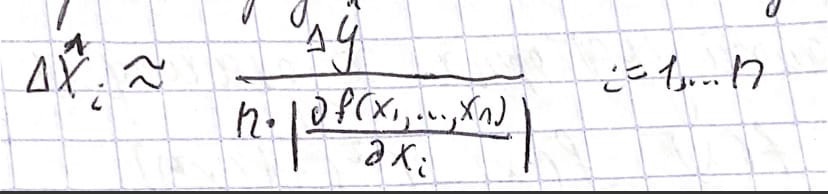
***Методы решения обратных задач теории погрешностей***

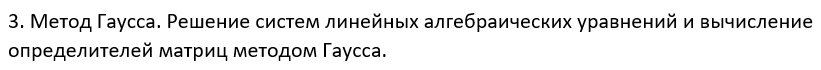
Для функции нескольких переменных y=f(x1,x2,…,x(n)) обратная задача теории погрешностей не имеет общего алгоритма и м.б. решена только при ряде ограничений.

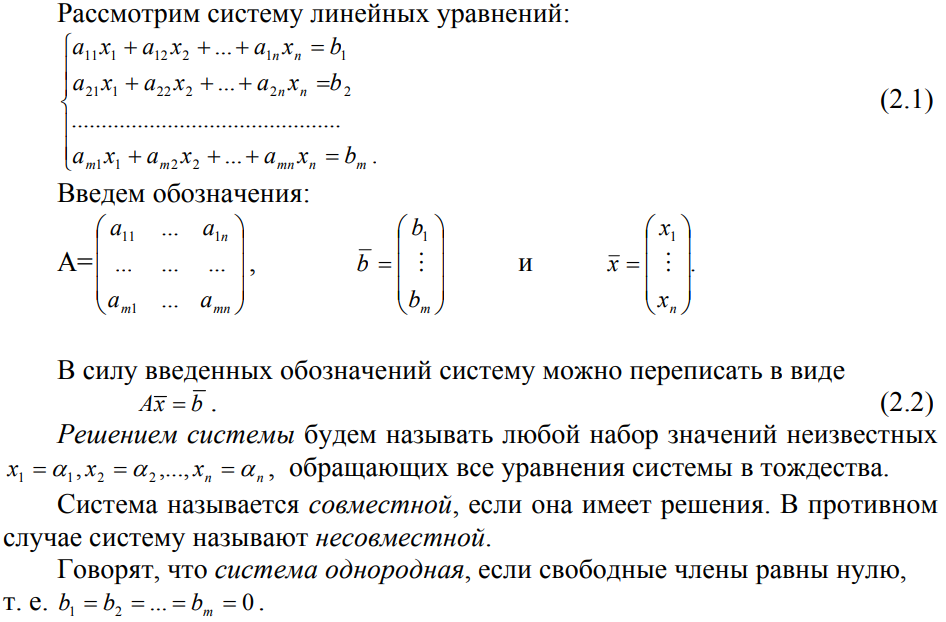
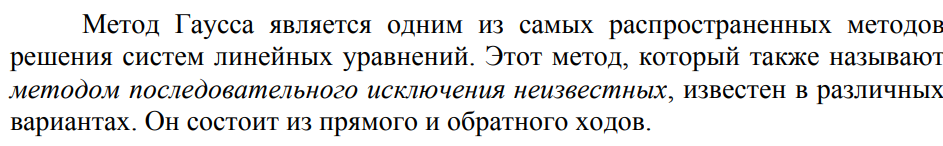
Если значение всех аргументов одинаково легко определить с любой точностью, то применяют принцип равных влияний.

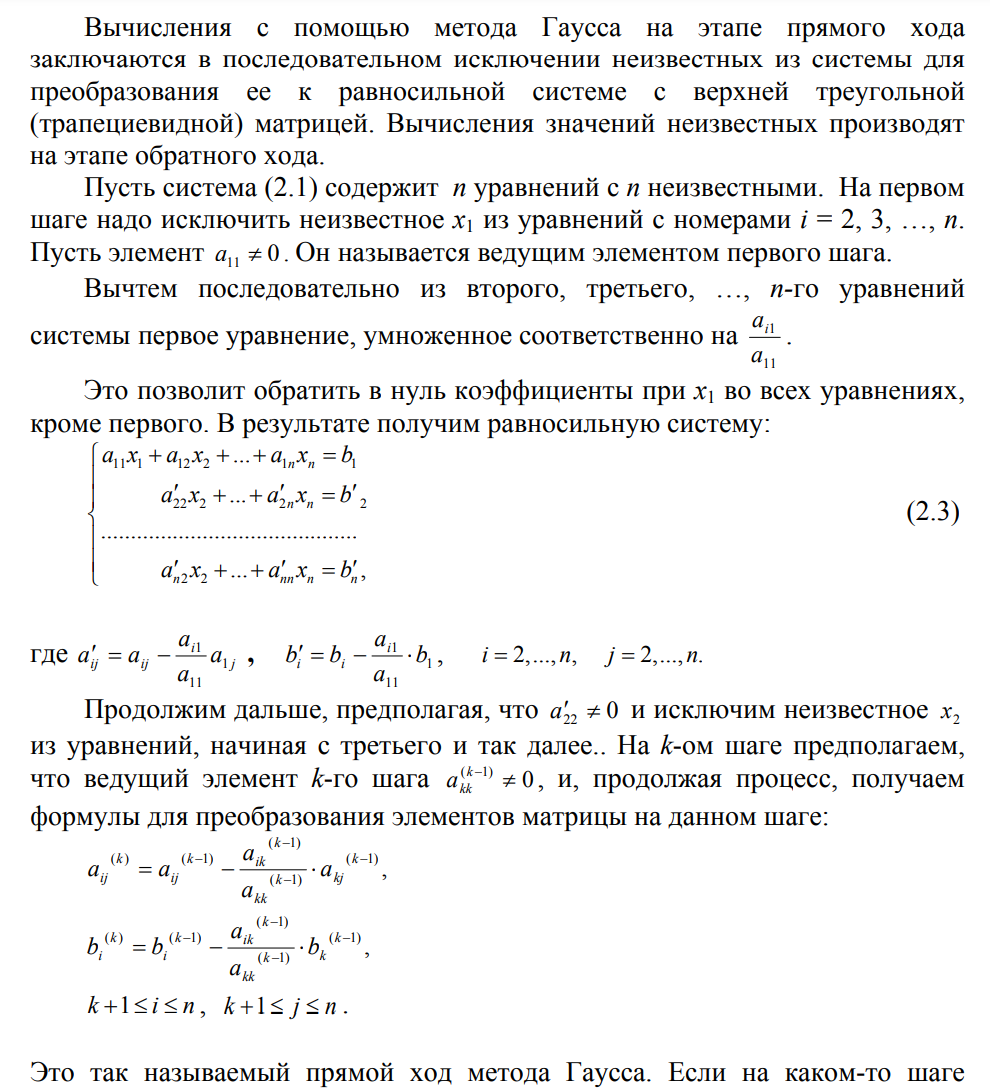


Все слагаемые в ф-ле (1) равны между собой берем dy/n -> одинаковы для всех, в этом случае, допустимые абсолютные погрешности аргументов м.б найдены по ф-лам:

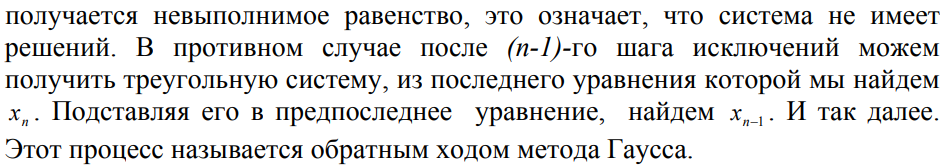




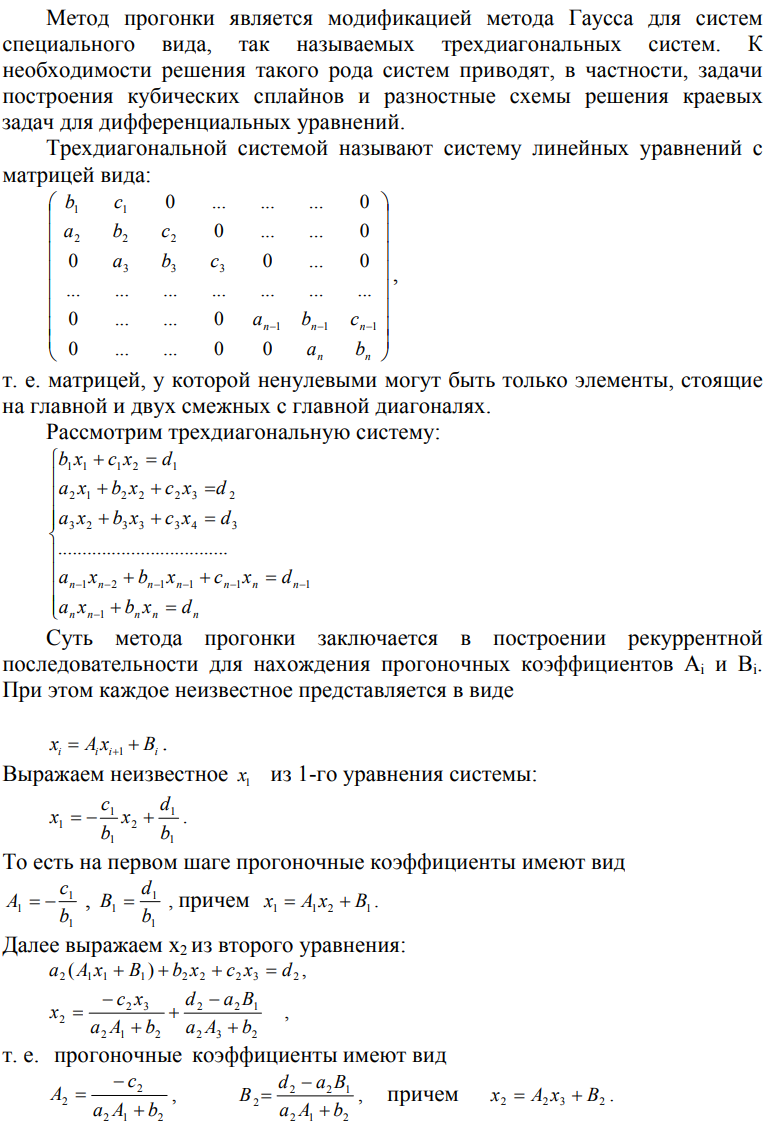
****

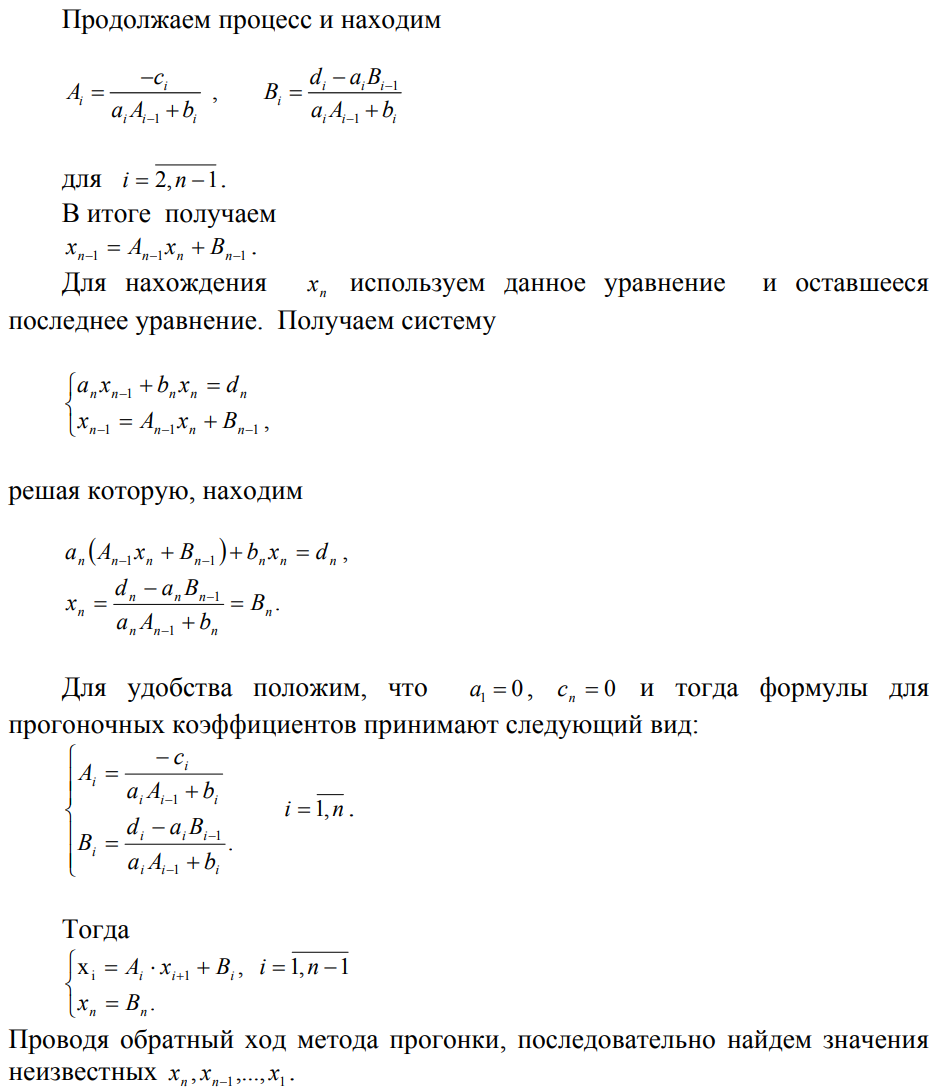
****

****

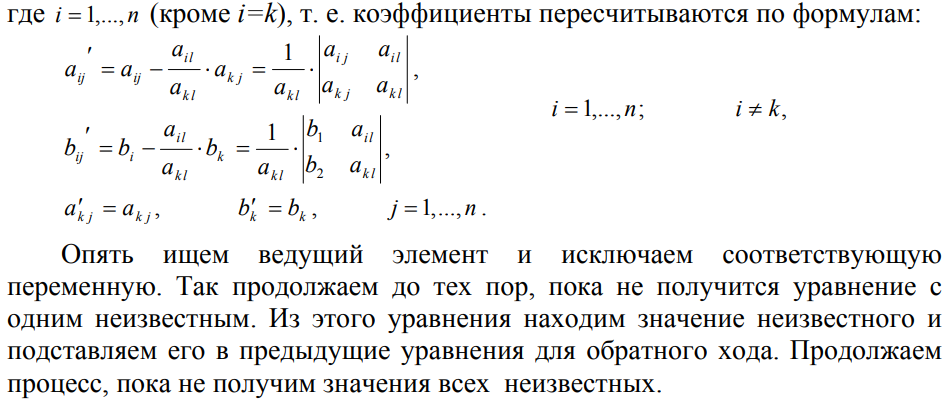
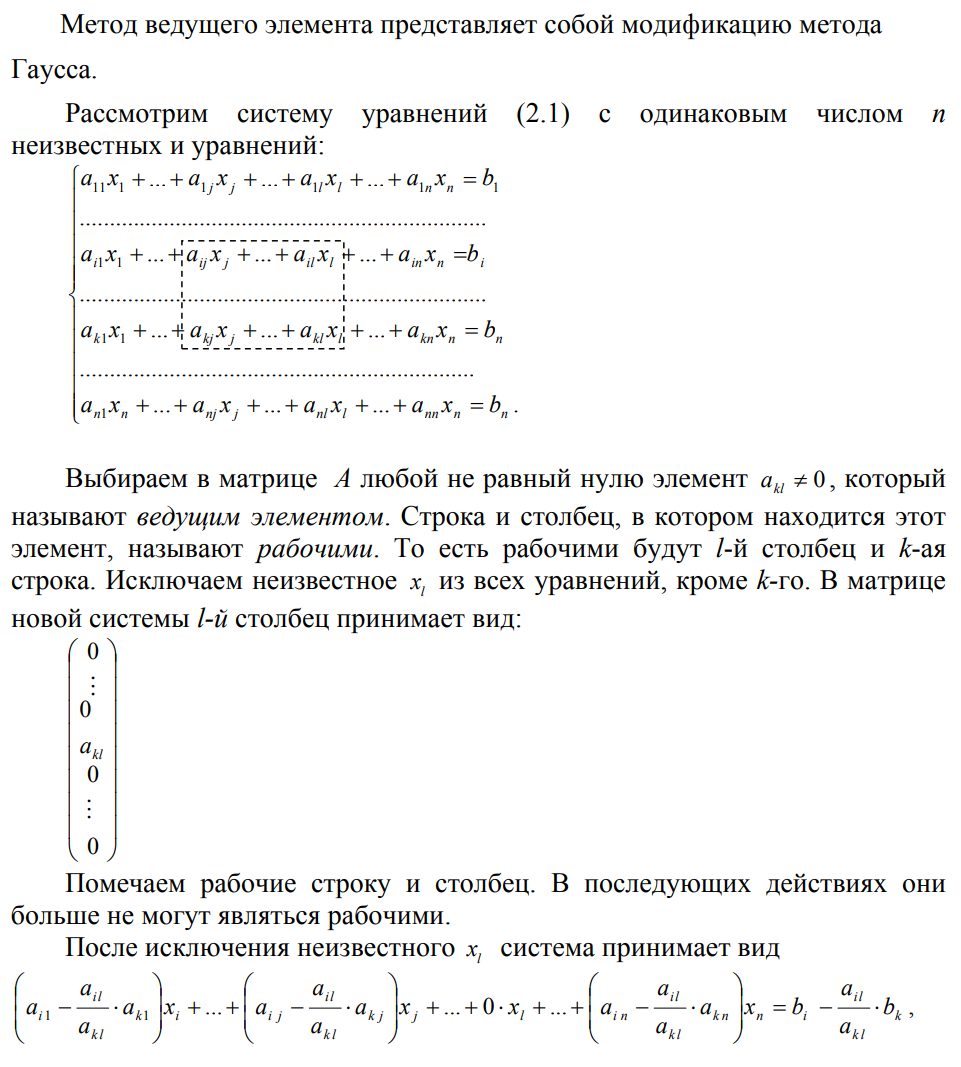
****

****

****

****

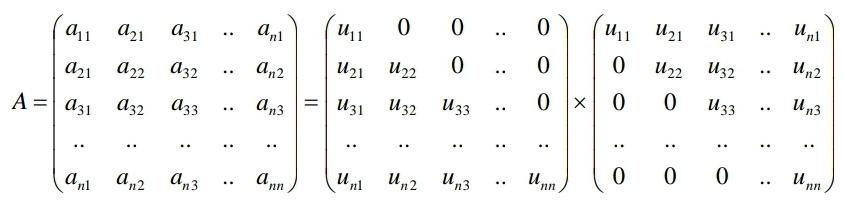




#### **6. Решение систем линейных алгебраических уравнений методом квадратного корня.**

Симметричные матрицы иногда можно представить в виде

произведения двух треугольных: A=U·U\*:

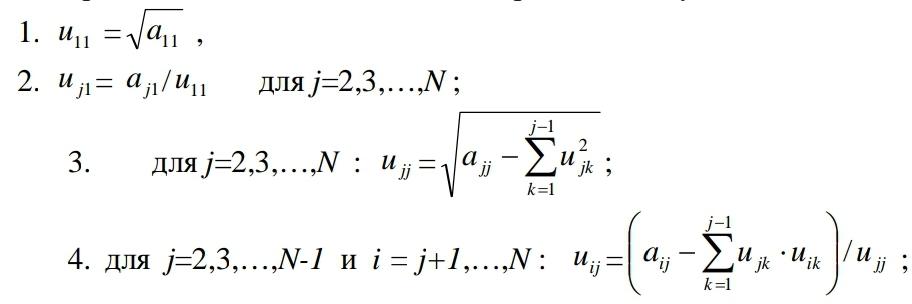


Для этого симметричная матрица должна иметь одно из равносильных

свойств:

* положительно определена;
* все главные миноры должны быть положительными;
* иметь диагональное преобладание и на главной диагонали иметь положительные значения;
* все её собственные числа должны быть положительны.

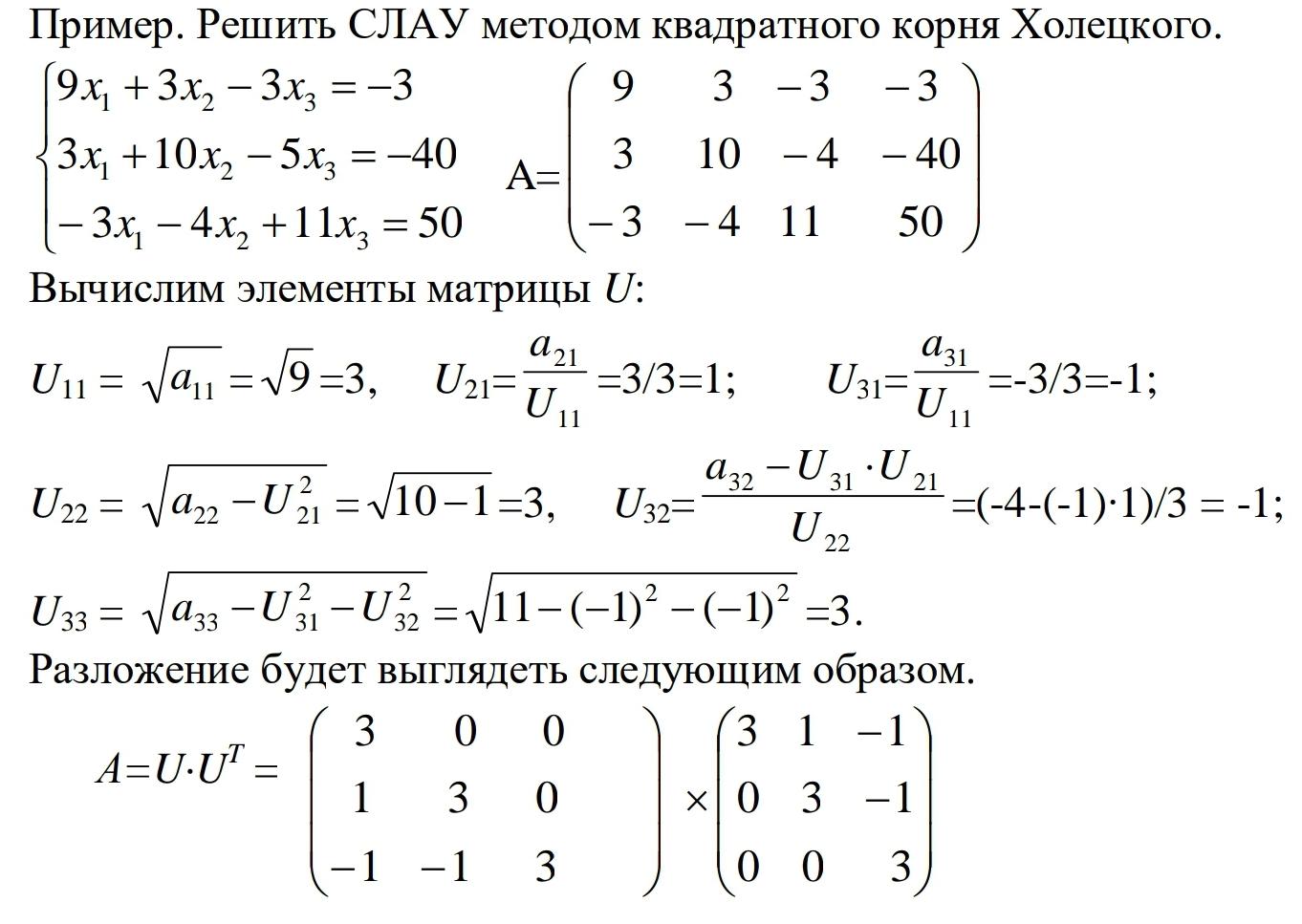
Алгоритм вычисления элементов матрицы U следующий:

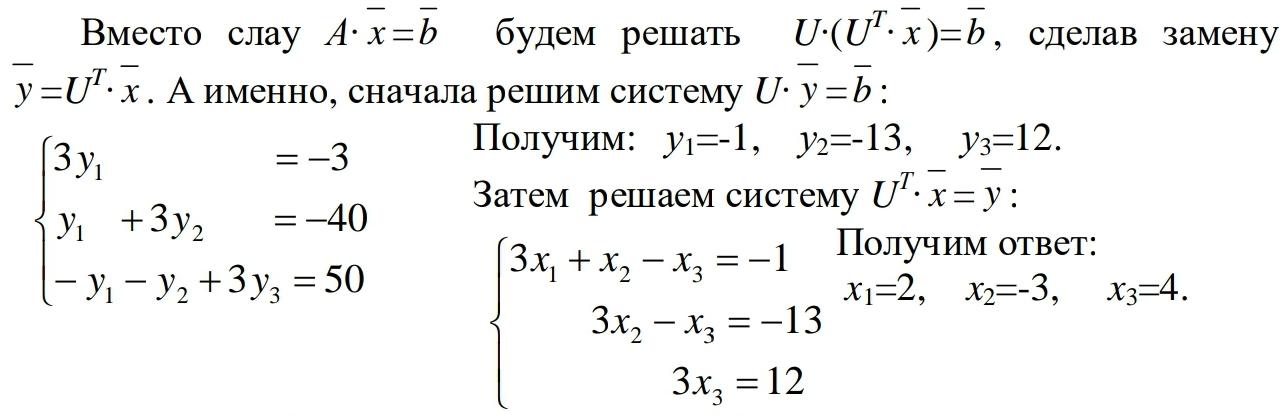


Метод квадратных корней Холецкого быстрее метода Гаусса (по

количеству арифметических операций) почти в два раза. Но применим он не для любых симметричных матриц, а для положительно определенных.

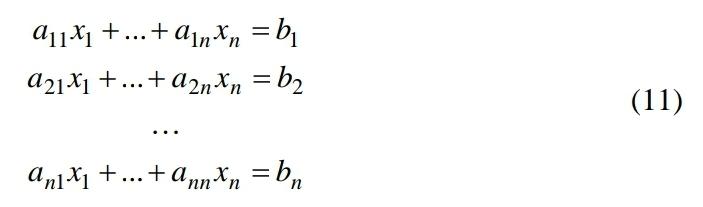
***пример:***





#### **7. Решение систем линейных алгебраических уравнений методом простых итераций. Сходимость метода и оценка погрешности.**

Пусть дана система линейных алгебраических уравнений (СЛАУ) вида:

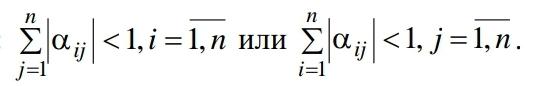


или в матричной форме:

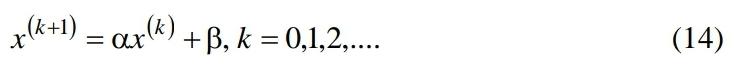


Рассмотрим решение этой системы методом простых итераций. Для применения этого метода необходимо предварительно преобразовать систему (12) к виду

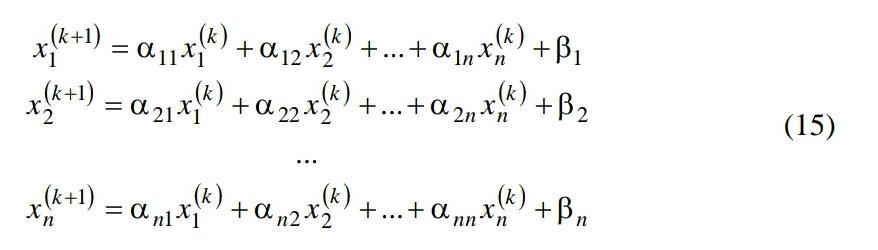


где матрица такова, что выполнены достаточные условия сходимости итерационного процесса : 

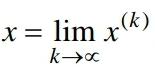
Зададим произвольно начальный вектор приближения  и подставим его в правую часть преобразованной системы уравнений. Получим первое приближение . Аналогично получим  Итак, итерационная формула

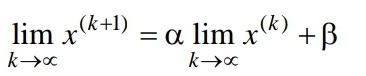


или в координатной форме:

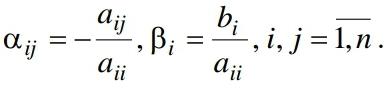


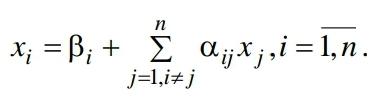
осуществляет итерации по “совокупности координат”. Последовательность векторов  полученных по этой формуле, сходится к решению, если выполнены вышеприведенные достаточные условия сходимости.

Пусть , тогда, переходя к пределу в равенстве (14), имеем:

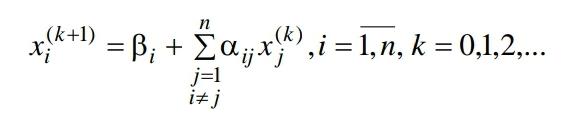


или имеет место формула (13). Следовательно, вектор x - решение системы. Если в исходной системе (12) преобладание диагональных элементов  над остальными коэффициентами значительное, то сходимость итерационного процесса обеспечена. В этом случае переход от исходной системы (12) к виду (13) можно осуществить путем деления каждого уравнения системы (12) на коэффициент , формирования столбца  в левой части и переноса остальных членов в правую часть.

Введем обозначения  Тогда



Рабочая формула итерационного процесса имеет в этом случае следующий вид:



Начальное приближенное решение можно взять произвольно,

например, равным столбцу свободных членов . Далее

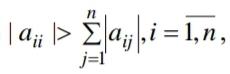
последовательно получаются приближения . Если для

преобразованной системы (13) выполнено по меньшей мере одно из

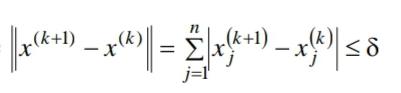
достаточных условий сходимости, то процесс итераций (14) сходится к

единственному решению этой системы, независимо от выбора начального приближения.

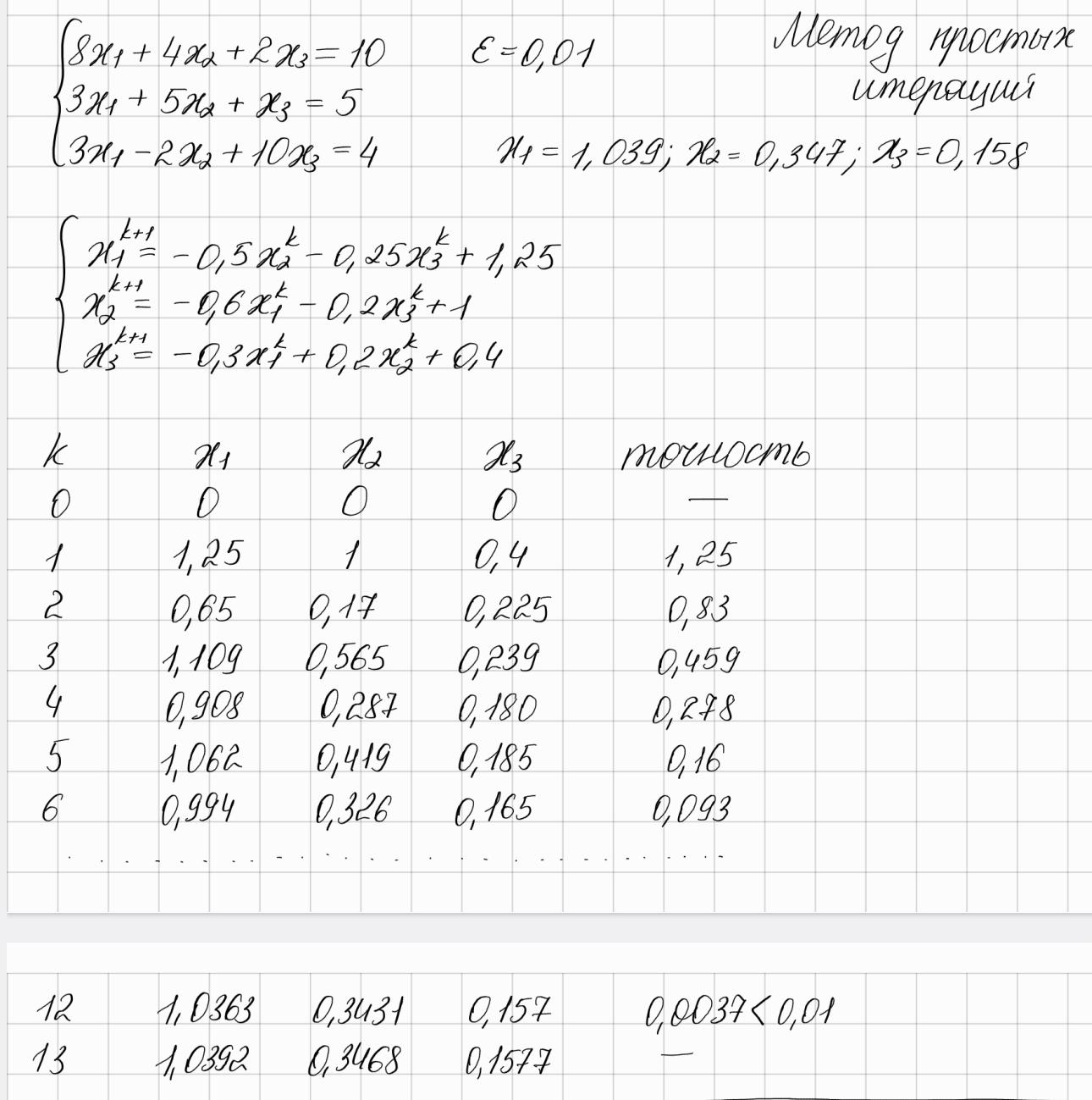
Для системы (12) метод итераций сходится, если выполнены

неравенства  т.е. модули диагональных коэффициентов в

каждом уравнении больше суммы модулей всех остальных коэффициентов (не считая свободных членов).

Итерационный процесс следует закончить, когда два последовательных приближения близки между собой по норме , где  - заданная точность.

***пример:***



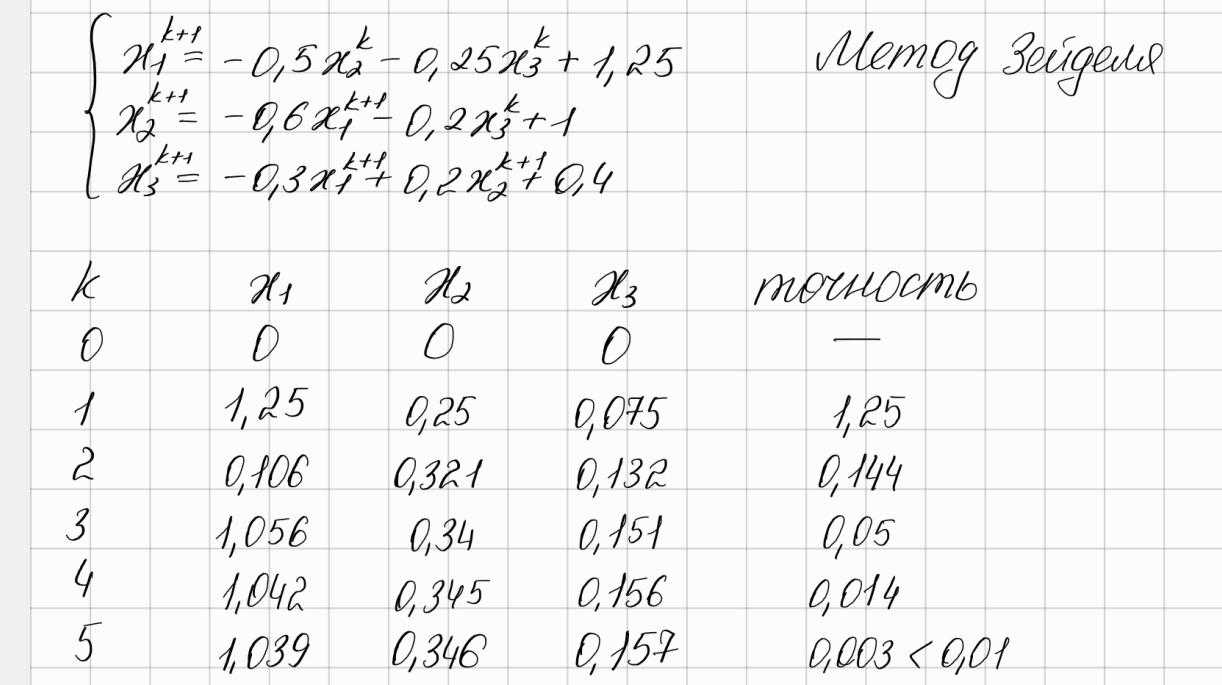
#### **8. Решение систем линейных алгебраических уравнений методом Зейделя. Сходимость метода и оценка погрешности.**

Более быструю сходимость метода простых итераций можно обеспечить, если для каждой i -ой компоненты вектора решения (k + 1) приближения использовать предыдущие компоненты от 1 до i - 1 также (k + 1) приближения, а остальные компоненты от i до n используются от предыдущего k -го приближения. Такая модификация метода простых итераций носит название «метода Зейделя». Запишем рабочие формулы метода Зейделя для каждой компоненты:

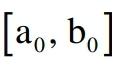
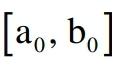


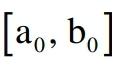
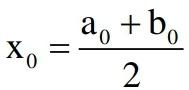
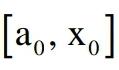
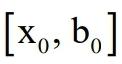
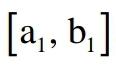
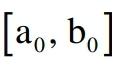
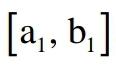
Первое и второе достаточные условия для сходимости метода простых итераций будут одновременно достаточными и для процесса Зейделя.

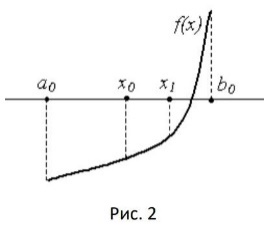
***пример:***

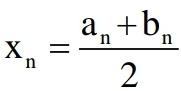
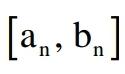
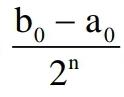
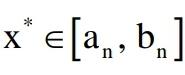


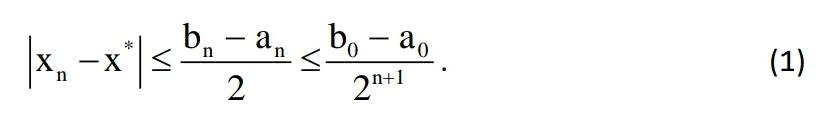
#### **9. Решение нелинейных уравнений. Метод половинного деления. Сходимость метода и оценка погрешности.**

Метод половинного является самым простым и надежным способом решения нелинейного уравнения. Пусть из предварительного анализа известно, что корень уравнения находится на отрезке , т.е. , так что . Пусть функция f (x) непрерывна на отрезке  и принимает на концах отрезка значения разных знаков, т.е. 

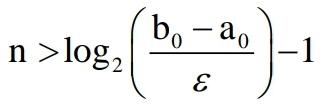
Разделим отрезок  пополам. Получим точку . Вычислим значение функции в этой точке: . Если , то  – искомый корень, и задача решена. Если , то  – число определённого знака:  либо . Тогда либо на концах отрезка , либо на концах отрезка  значения функции f (x) имеют разные знаки. Обозначим такой отрезок . Очевидно, что и длина отрезка  в два раза меньше, чем длина отрезка . Поступим аналогично с отрезком . В результате получим либо корень , либо новый отрезок  и т. д. (рис. 2).



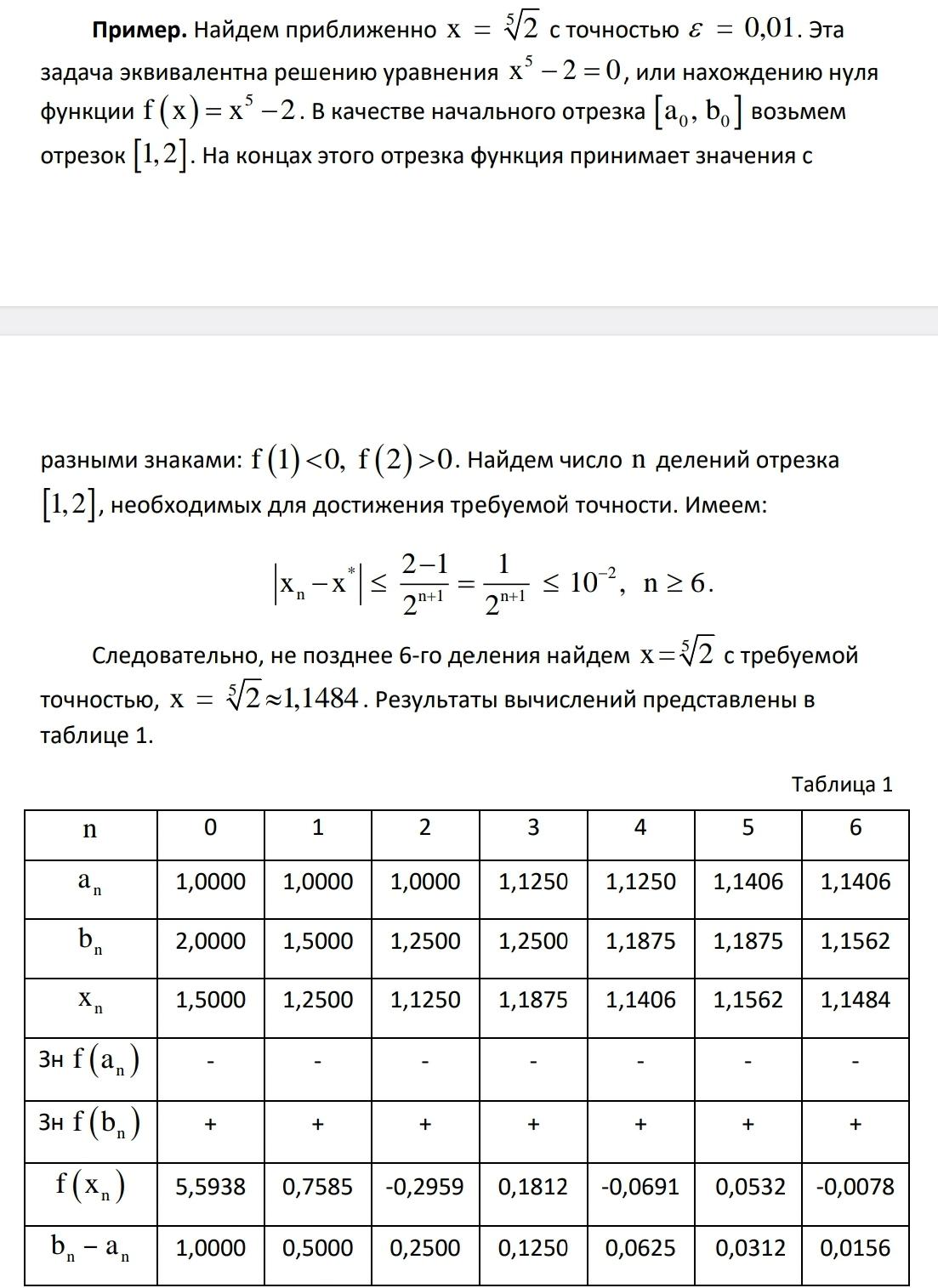
Середина n -го отрезка . Очевидно, что длина отрезка  будет равна , а так как , то:



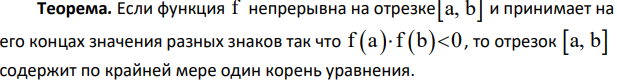
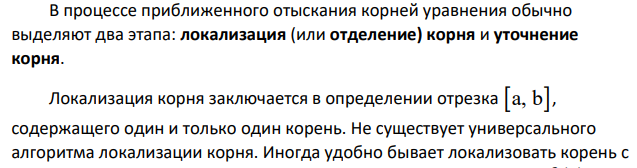
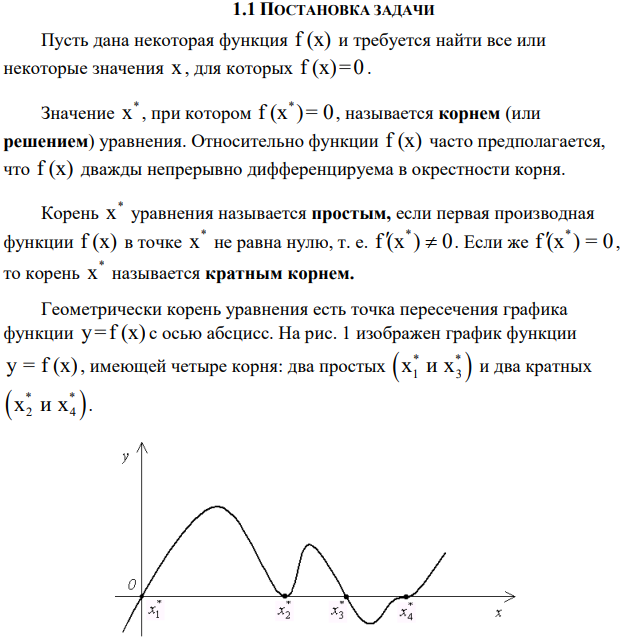
Критерий окончания. Из соотношения (1) следует, что при заданной

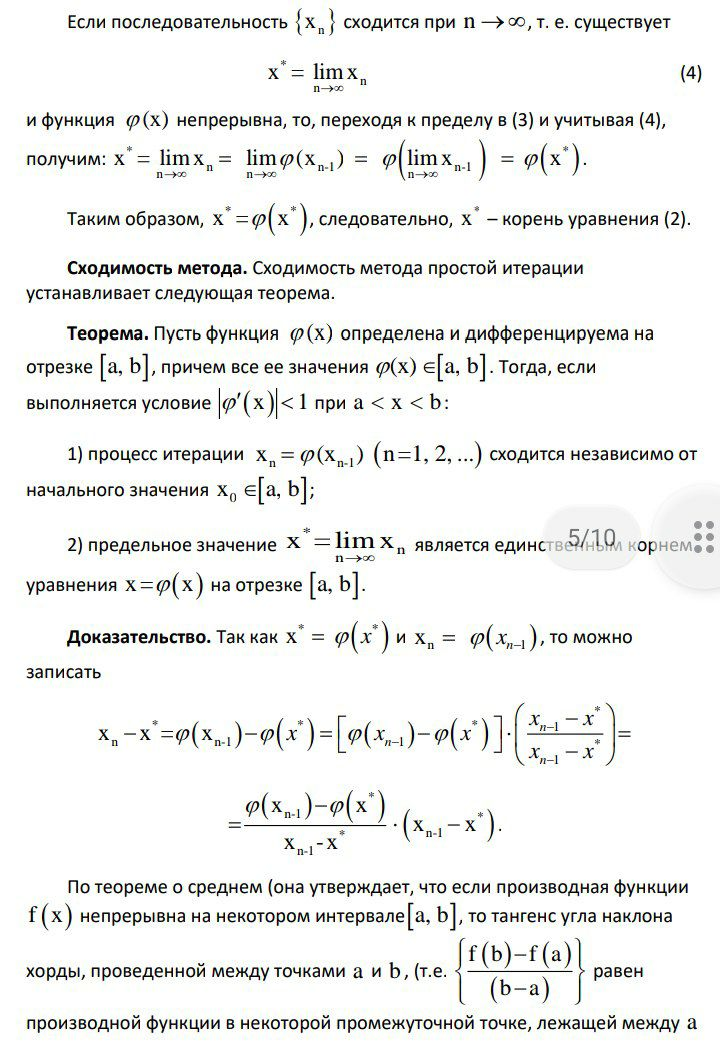
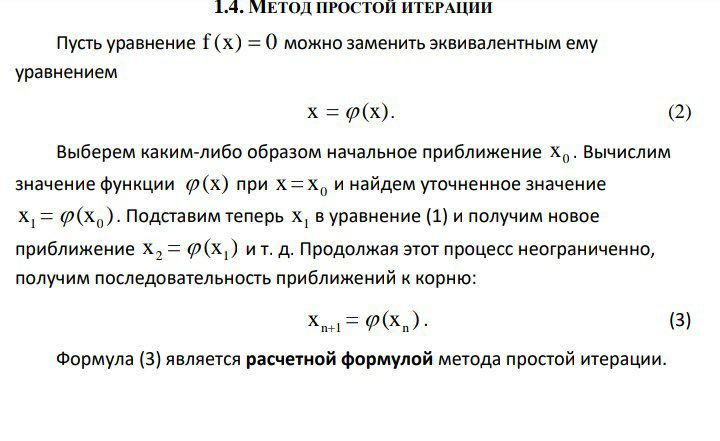
точности приближения  вычисления заканчиваются, когда будет выполнено неравенство  или неравенство . Таким образом, количество итераций можно определить заранее. За приближенное значение корня берется величина .

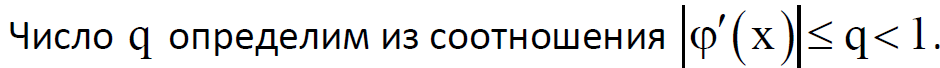
***пример:***

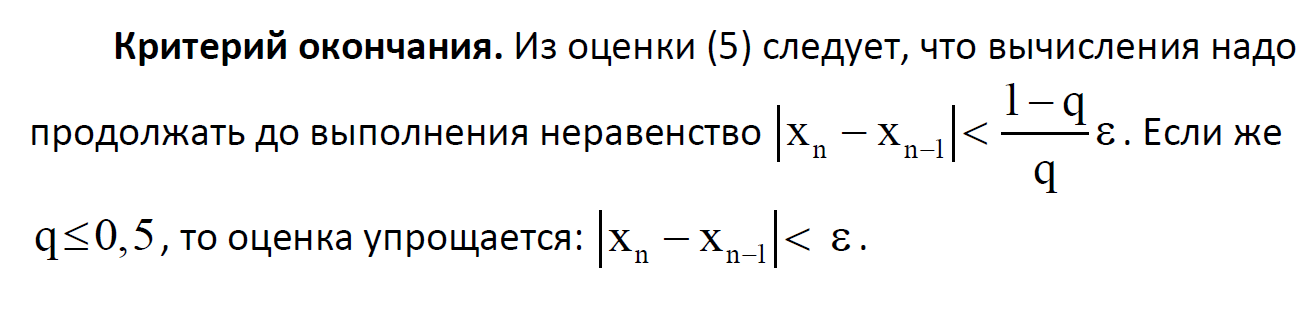




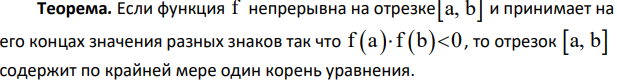
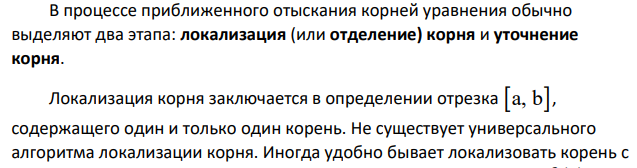
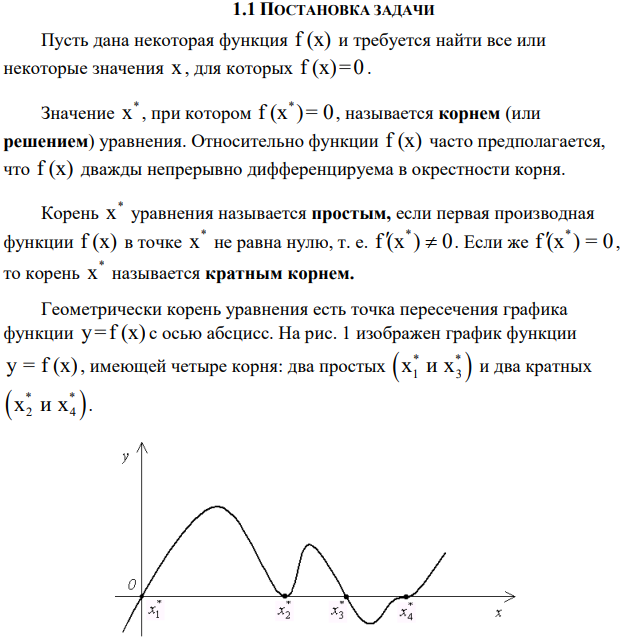


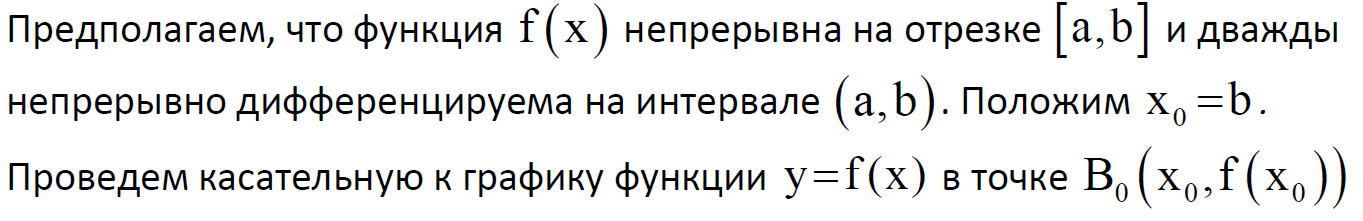
****

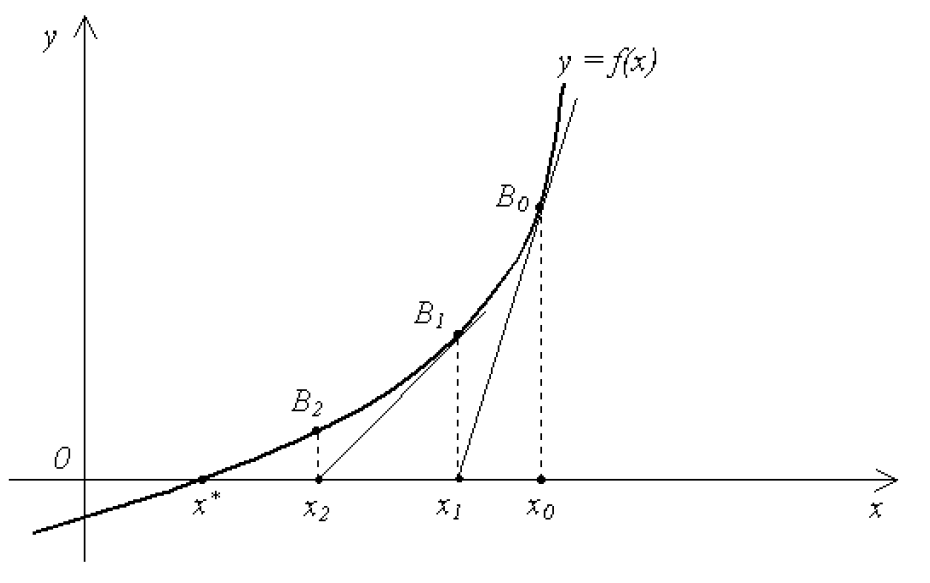


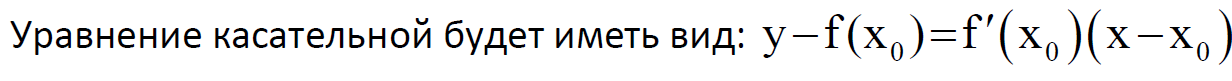


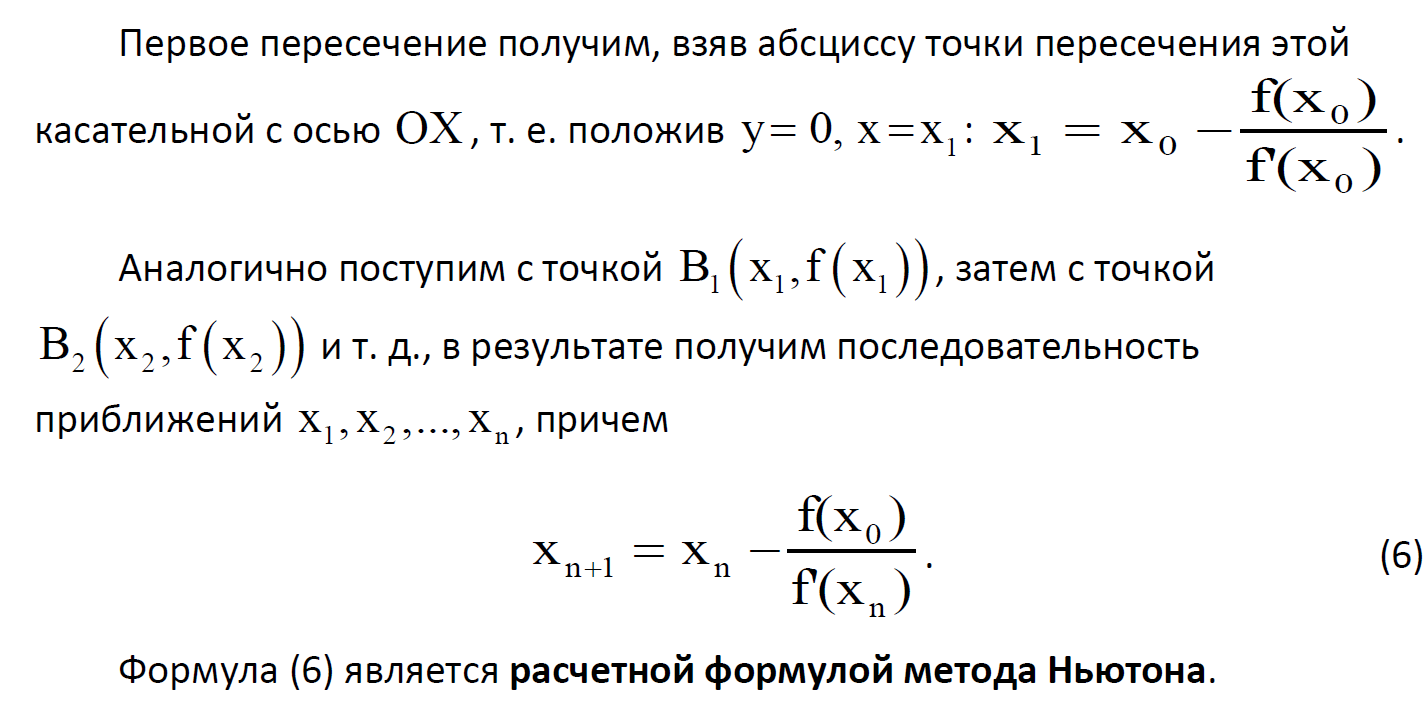


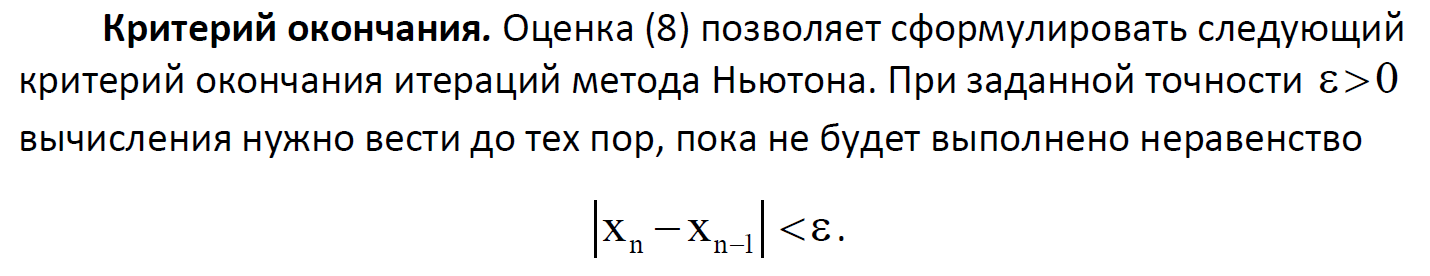




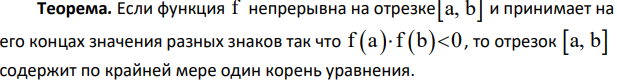
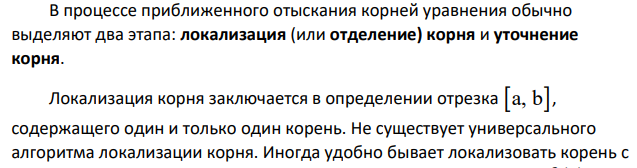
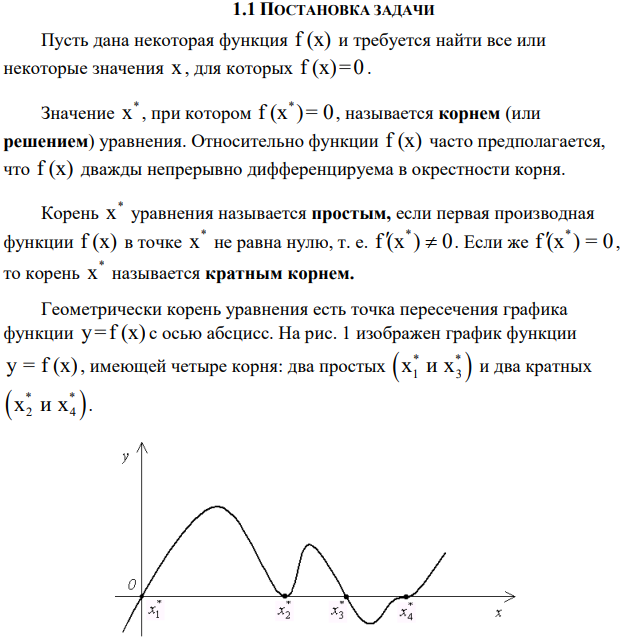


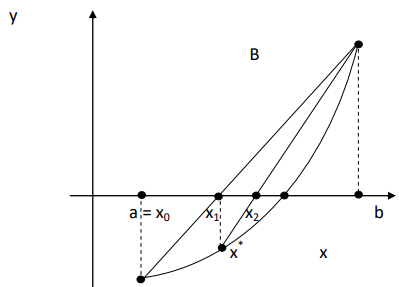
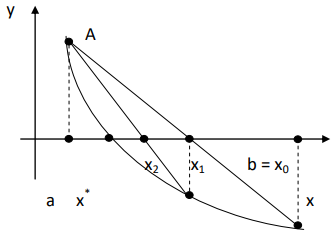
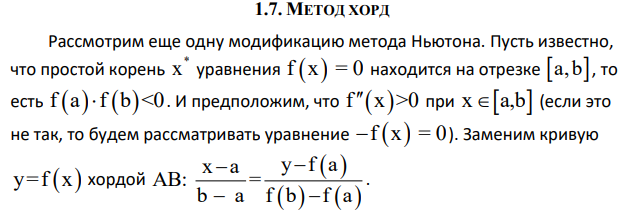


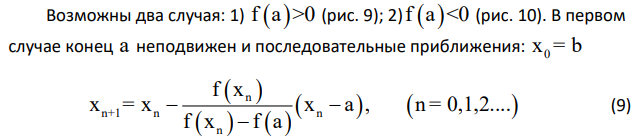


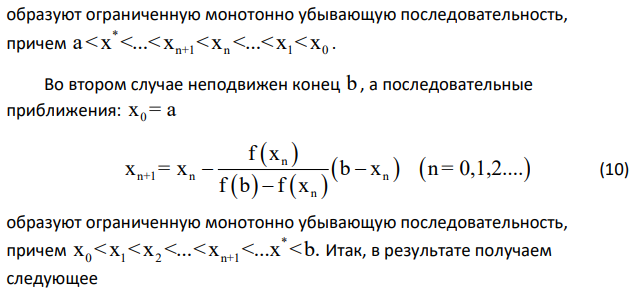


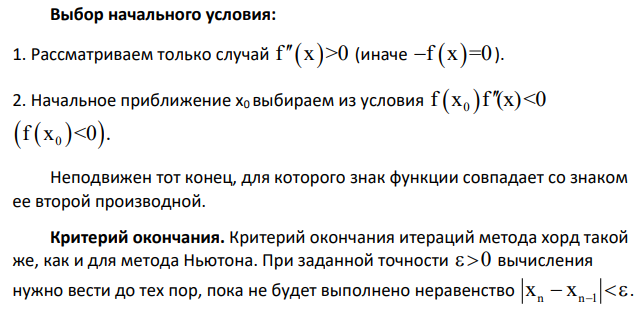




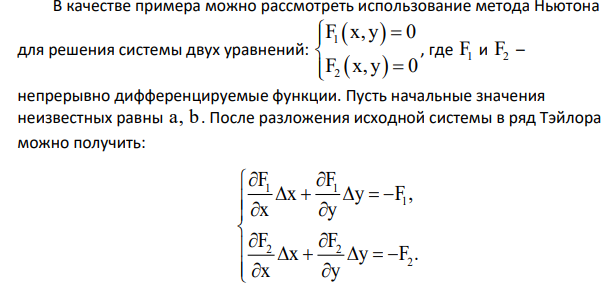


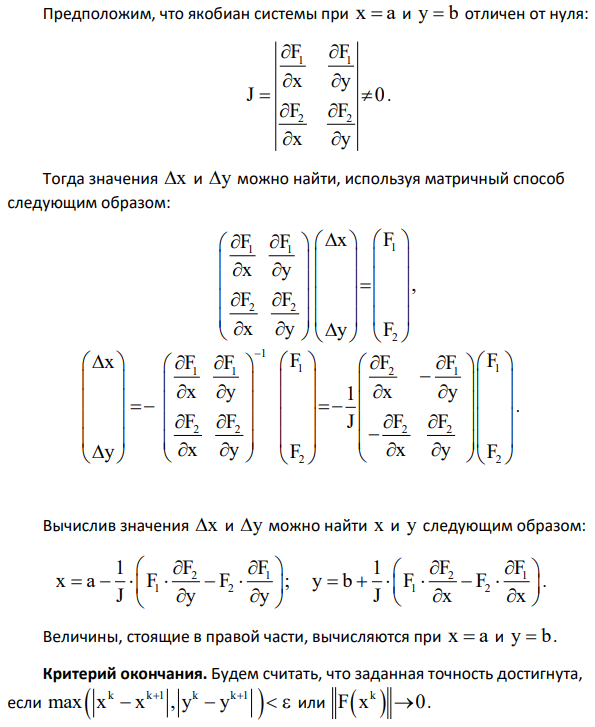


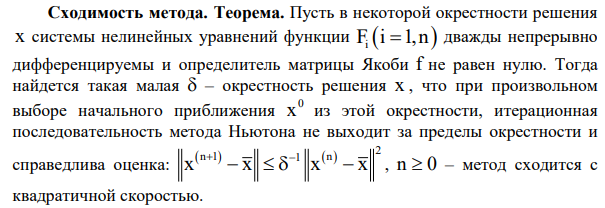






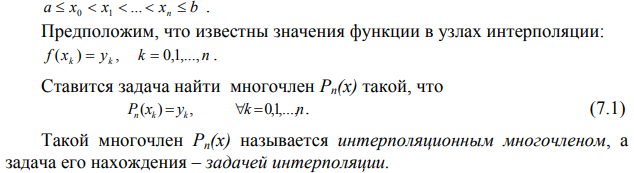


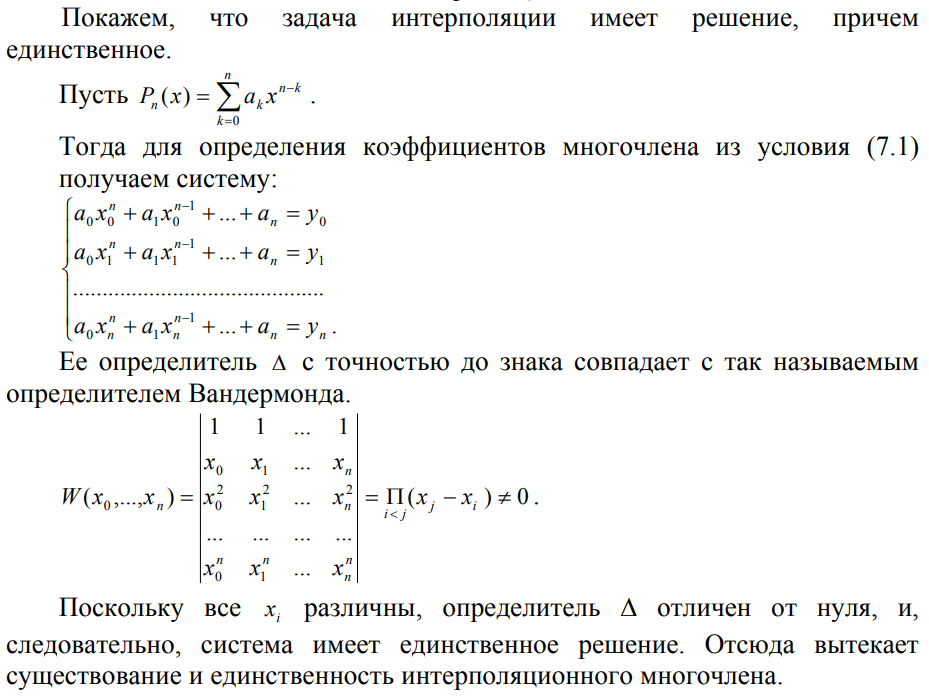






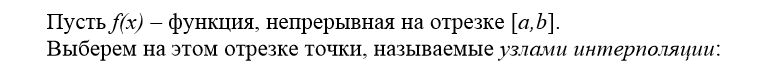


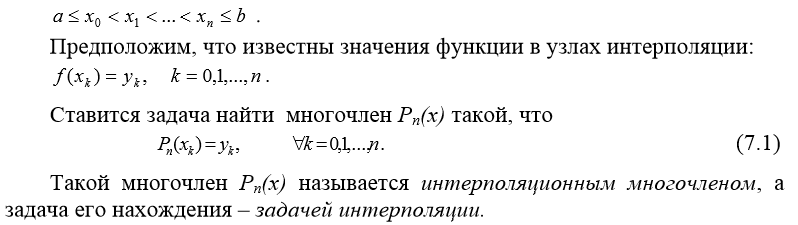






**Не знаешь, что такое интерполяция? Так вот:**





**Интерполяционный многочлен Лагранжа**

Интерполяционный многочлен Лагранжа задаётся формулой:

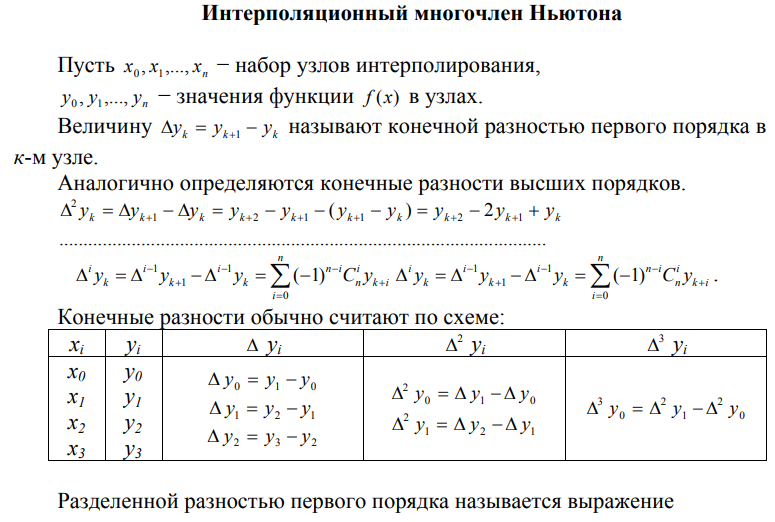
где

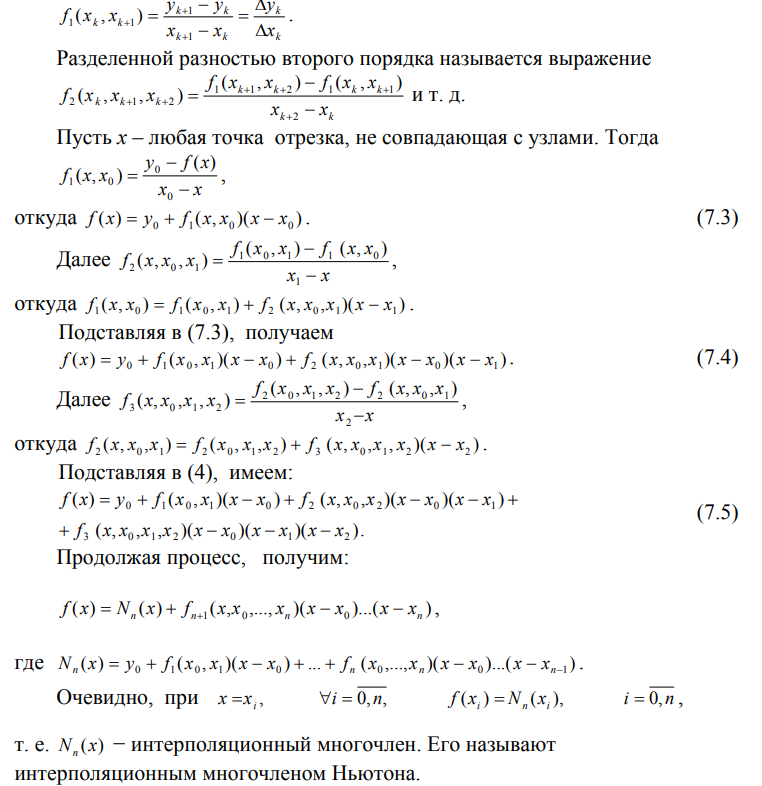
– степень многочлена (или *количество узлов интерполяции - 1*)

К примеру, при:

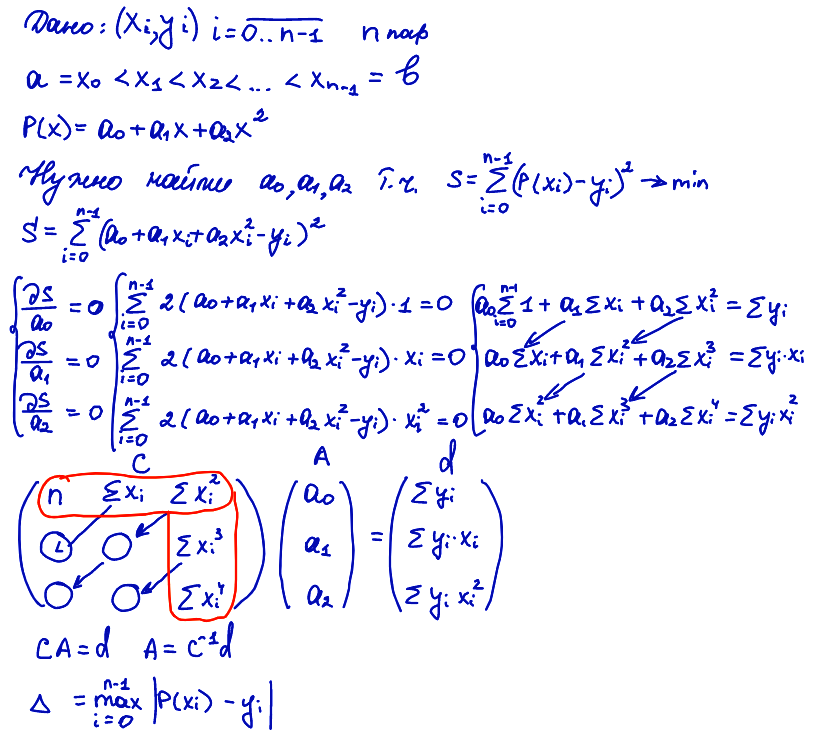
Интерполяционный многочлен Лагранжа имеет минус по сравнению с интерполяционными многочленом Ньютона: при добавлении нового узла нужно заново составлять весь многочлен











### **18. Проблемы интерполяции функции двух переменных.**

### **Проблема выбора узлов**

Заметим что не любое число узлов интерполяции выгодно. Если для одной переменной степень многочлена была взаимно однозначно связана с числом узлов, то для двух переменных многочлен *n*-ой степени P_n(x,y)=\sum_{k+m=0}^na_{km}x^ky^m\  имеет (n+1)(n+2)/2 узлов.

Если число узлов не соответствует этой формуле, то часть коэффициентов при высших степенях должна задаваться принудительно (в частности нулями): для выбора этих коэффициентов редко есть разумные основания.

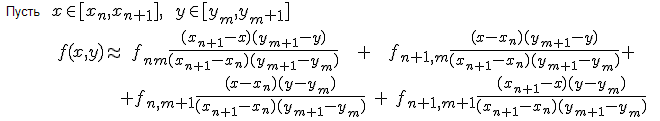
Также не всякое расположение узлов допустимо: в одномерном случае узлы не должны были совпадать. Теперь же для интерполяции многочеленом P_1(x,y) необходимо, чтобы узлы не лежали на прямой в плоскости (x,y). При интерполяции многочленом P_n(x,y) требуется, чтобы узлы не лежали на кривой n-го порядка.

Поэтому для хорошей интерполяции сетка должна быть регулярно построенной, а не представлять собой совокупность беспорядочно расположенных точек. Следующие два примера используют прямоугольную сетку, образованную пересечением прямых x = xn, n = 0, ..., N и y = ym, m = 0, ..., M,

fnm = f(xn, ym) — значение функции в узле {xn, ym }.

**Кусочно-Билинейная интерполяция**

**Билинейной интерполяцией** называют расширение линейной интерполяции для функций двух переменных. Для начала реализуется линейная интерполяция по x на каждой прямой y = ym . Затем при каждом значении x = xn реализуется линейная интерполяция по y с учетом значений функции, полученных на первом шаге.



Результат билинейной интерполяции не зависит от порядка шагов: можно сначала интерполировать вдоль оси абсцисс а затем вдоль оси ординат, так и наоборот, результат будет одним и тем же.

### **19. Построение интерполяционного многочлена для функции двух переменных.**

Рассмотрим только те из выбранных нами узлов, для которых можно построить многочлен степени на 1 меньше.

В точке все слагаемые обращаются в 0 за исключением

Следовательно коэффициент определяется однозначно. В силу единственности интерполяционного многочлена по выбранным нами узлам это и будет единственным значением разности.

- разделенная разность функции при фиксированном

Пусть

Мы получим интерполяционный многочлен относительно , принимающий в точках значения

разделенные разности

Последний интерполяционный полином относительно должен в точках принимать значения Последнее слагаемое при таких превращается в ноль. Значит все члены в правой части кроме последнего дают интерполяционный полином Ньютона степени , принимающие в точках значения

разделенная разность по y

Аналогично то

### **20. Задача численного дифференцирования. Вычисление производных с помощью интерполяционных многочленов.**

Задача численного дифференцирования возникает при нахождении производных от функции *у* = *f{x),* заданной таблично, либо при нахождении производной от аналитической функции, непосредственное дифференцирование которой по каким-либо причинам затруднено.

Пусть задана табличная функция {xi;yi , i=1,...,n}:



Т.е. заданы n точек Мi(xi;yi) на плоскости X0Y, характеризующие значения некоторой неизвестной функции f(x). Требуется найти f `(x0), f ``(x0),... для x0∈[x1;xn]. Для этого построим интерполяционный многочлен y=P(x) и от него вычислим первую производную, вторую производную и т.д. в точке х=х0. Вычисление канонического интерполяционного многочлена легко программируется, ещё проще программируется вычисление производной от многочлена с известными коэффициентами. Часто строят многочлен не по всем узлам, а только по тем, что ближе к х0.

Предположим, что некоторая функция задана таблицей значений *yi* = *f*(*xi*),  с постоянным шагом аргумента *h* = *xi* – *xi*-1 . Для того, чтобы выразить значение производной через значения функции в узлах интерполяции, запишем интерполяционный многочлен Лагранжа степени *m*, удовлетворяющий условиям *Lm*(*xk*) = *yk* = *f*(*xk*), :

,

где лагранжевы коэффициенты вычисляются как

.

Дифференцируя этот многочлен, можно получить приближенные значения производных в узлах интерполирования *xk* .

В частности, для *m* = 1 получим:

;

.

численный дифференцирование производная интерполяционный

Пусть *m* = 2. Тогда



, (5)

, (6)

. (7)

### **21. Вычисление производных с помощью конечных разностей.**

Производная функции  есть 

где 

Если Δ *x* принять равным конечному числу, то



Это соотношение представляет *аппроксимацию производной* с помощью конечных разностей. При этом шаг аргумента Δ *x*, то есть разность между соседними значениями аргумента, – обычно принимают одинаковым и обозначают для краткости одной буквой *h*.

Допустим имеем значения функции с указанным шагом на некотором отрезке , равные  . Запишем выражение для производной в точке *i* (при ). При этом в зависимости от способа вычисления конечных разностей получим разные формулы для вычисления *производной* в одной и той же точке:

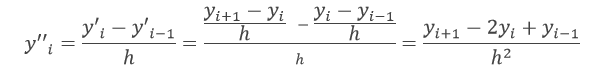
а) для *левой разности* будем иметь:

б) для *правой разности* получим: 

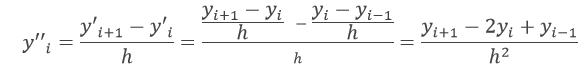
в) для *центральной разности*: 

Учитывая, что для центральной разности берется участок большей длины (2h), то эта разность будет иметь большие погрешности, чем первые две, и соответственно требует разбивки системы на более мелкие участки. Однако она приводит к симметричным уравнениям и поэтому используется довольно часто.

Для *второй производной* в точке *i* будем иметь*:*

a) слева через правые разности:  


б) справа через левые разности:



### **22. Интерполирование сплайнами. Построение кубического сплайна.**

Пусть на  задана непрерывная функция . Введем сетку



и обозначим

Сплайном, соответствующим данной функции  и данным узлам , называется функция , удовлетворяющая следующим условиям:

а) на каждом сегменте  функция  является многочленом третьей степени;

б) функция , а также ее первая и вторая производные непрерывны на ;

в) 

Последнее условие называется условием интерполирования, а сплайн, определяемый условиями а)-в), называется также интерполяционным кубическим сплайном.

На каждом из отрезков , будем искать функцию  в виде многочлена третьей степени

 (10)

где  - коэффициенты, подлежащие определению. Поясним смысл введенных коэффициентов. Имеем

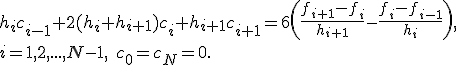


поэтому

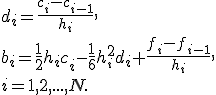


Функция  удовлетворяет условиям , таким образом, естественно требовать, чтобы .

Для определения коэффициентов  получаем систему уравнений

 (11)

В силу диагонального преобладания система (11) имеет единственное решение [3]. Так как матрица системы трехдиагональная, решение можно найти методом прогонки, которая в данном случае устойчива [10]. По найденным коэффициентам  коэффициенты  определяются с помощью явных формул

 (12)

Таким образом, существует единственный кубический сплайн, определяемый условиями а)-в) и граничными условиями . Нужно отметить, что можно рассматривать и другие граничные условия.

*Пример.*

Построить кубический сплайн для функции заданной таблично.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | -1 |  |  | 0 |  |  | 1 |  |  | 2 |
|  |  |  | 0.5 |  |  | 1 |  |  | 2.5 |  |  | 3.5 |

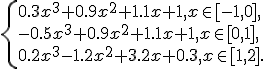
По условиям  Тогда система (11) будет иметь вид



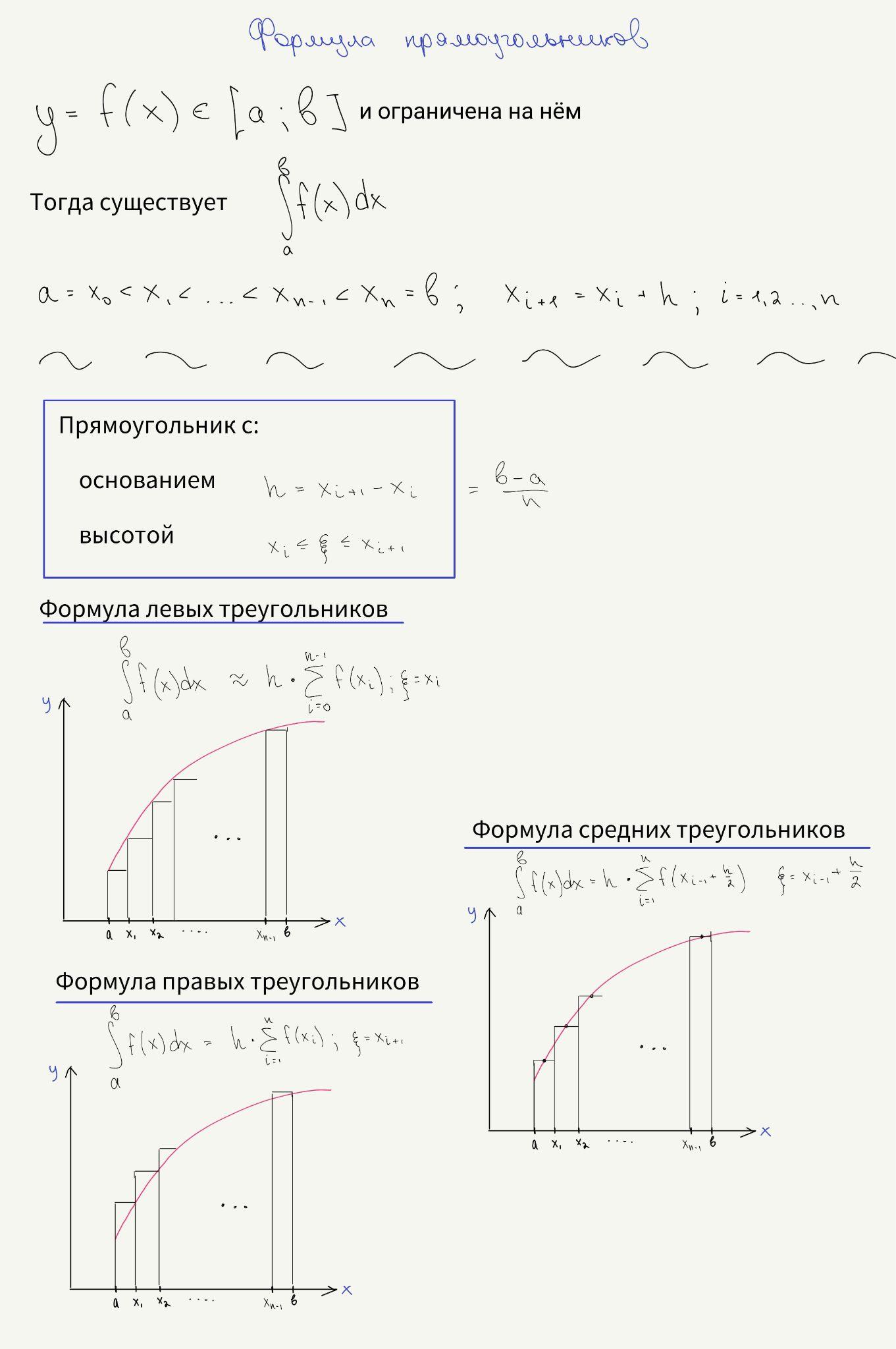
Решая ее получим  Далее из (12) найдем коэффициенты 



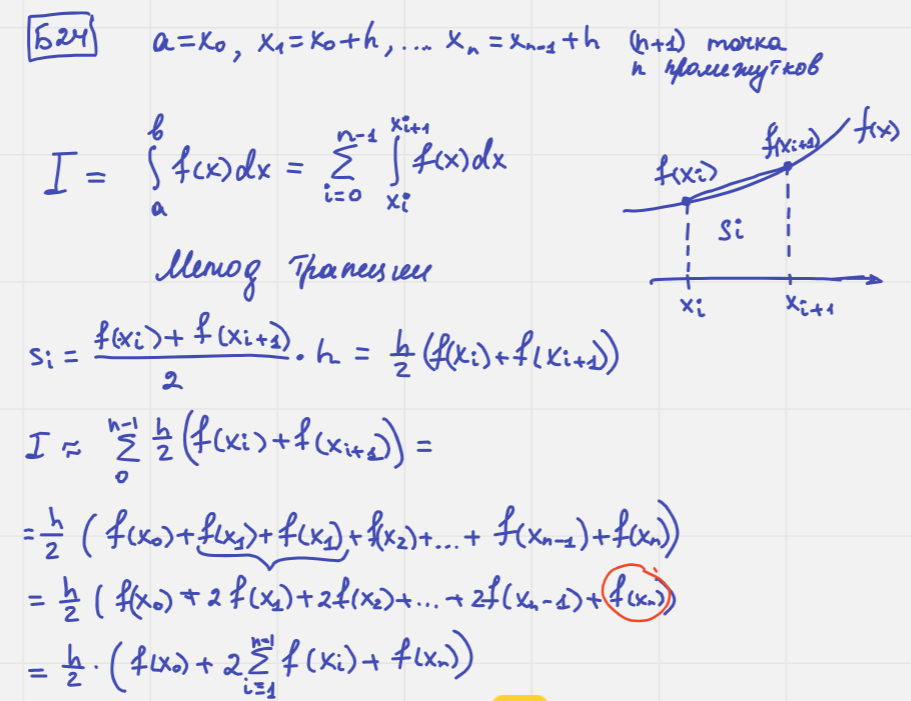
Итак, сплайн построен

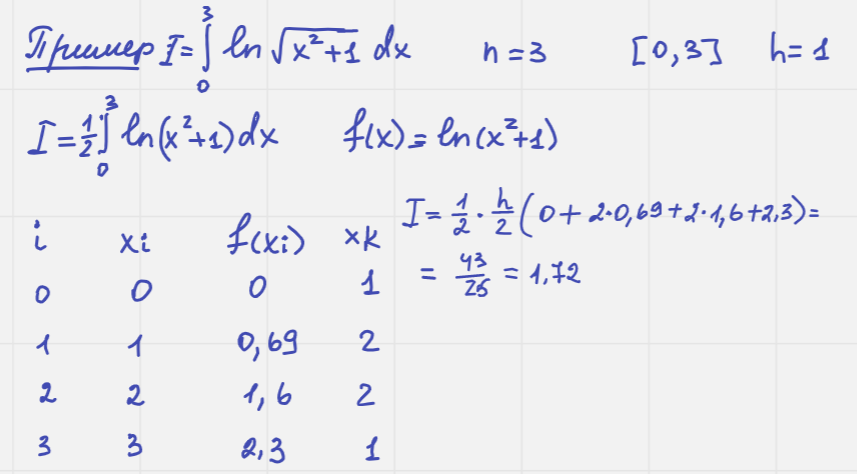


### **23. Численное интегрирование с помощью формулы прямоугольников.**

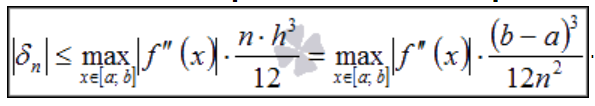
****

### **24. Численное интегрирование с помощью формулы трапеций.**

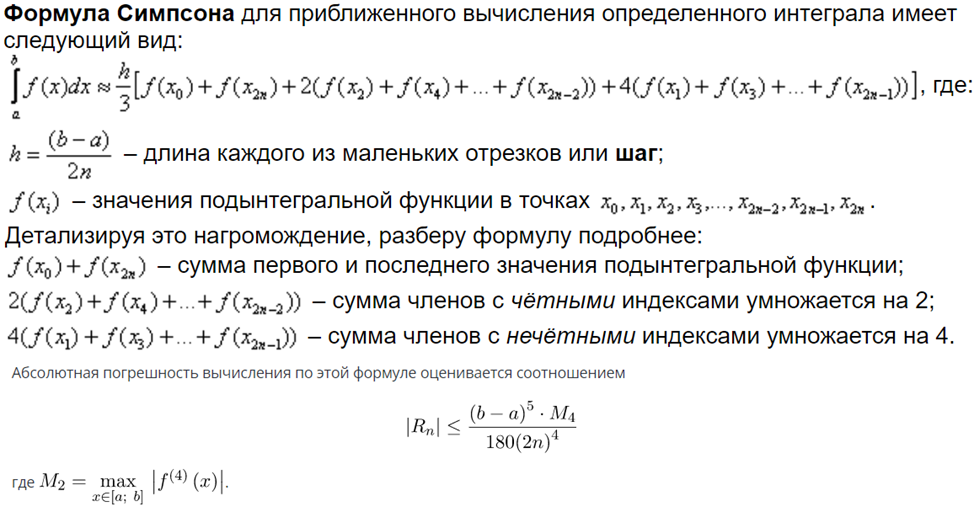




Оценка абсолютной погрешности метода трапеций.



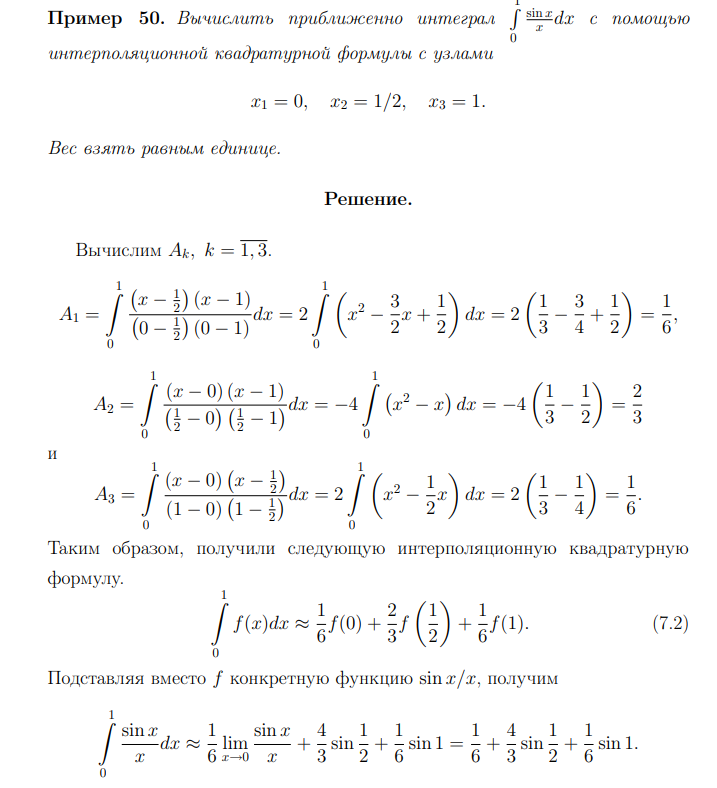
### **25. Численное интегрирование с помощью формулы Симпсона.**

**Суть метода.  
  
Основная инфа.  
**Узлы вычисляются по формуле xi = xi-1 + h, i = 1..2n.  
Если кол-во отрезков, на которое нужно разбивать не указано, то существует простой подход: сначала отрезок интегрирования разбивается на несколько больших отрезков, как правило, на 2-3-4-5. После первичного результата количество отрезков удваивают. Для завершения вычислений сравниваем |In-In-1| < e (если расхождение между двумя последними результатами меньше заданной эпсилон, то заканчиваем, иначе удваиваем кол-во отрезков и считаем дальше).

**Алгоритм решения.**•Выбираем 2n, если не задано (Число 2n понимается как ЕДИНОЕ ЧИСЛО. То есть, НЕЛЬЗЯ сокращать, например 2n=8, на два, получая n=4. Запись 2n лишь обозначает, что количество отрезков чётно. И ни о каких сокращениях речи не идёт).  
•Вычисляем h и x0, x1, …, x2n (x0 это нижняя граница вычисления интеграла, а x2n – верхняя).  
•Вычисляем значения функции в узлах (найденных x0, x1, …, x2n).  
•Подставляем всё в формулу и получаем результат.  
•Удваиваем кол-во отрезков и повторяем все действия для нового значения 2n.  
•Находим модуль разности двух посчитанных результатов и сравниваем с заданным эпсилон. Если разность больше эпсилон, то удваиваем отрезки, считаем и снова сравниваем, но уже 2й и 3й результаты (два последних). Если меньше эпсилон, то записываем ответ.

### **26. Интерполяционная квадратурная формула Лагранжа.**

Формулу  
 , где и   
называют интерполяционной квадратурной формулой Лагранжа.



Главным недостатком ИКФ Лагранжа является проблематичность оценки погрешности.

### **27. Численное решение задачи Коши для ОДУ 1-го порядка методом Эйлера.**

Метод Эйлера необходим для нахождения частного решения ДУ 1-ого порядка на интервале , то есть для решения задачи Коши на данном интервале.

Метод заключается в разбиении интервала на равные промежутки и последующем рекурсивном вычислении приближенных значений функции на каждой из точек.

Пусть и . А также задан интервал и задан некоторый промежуток , такой что , тогда значение ф-ии вычисляется следующим образом:

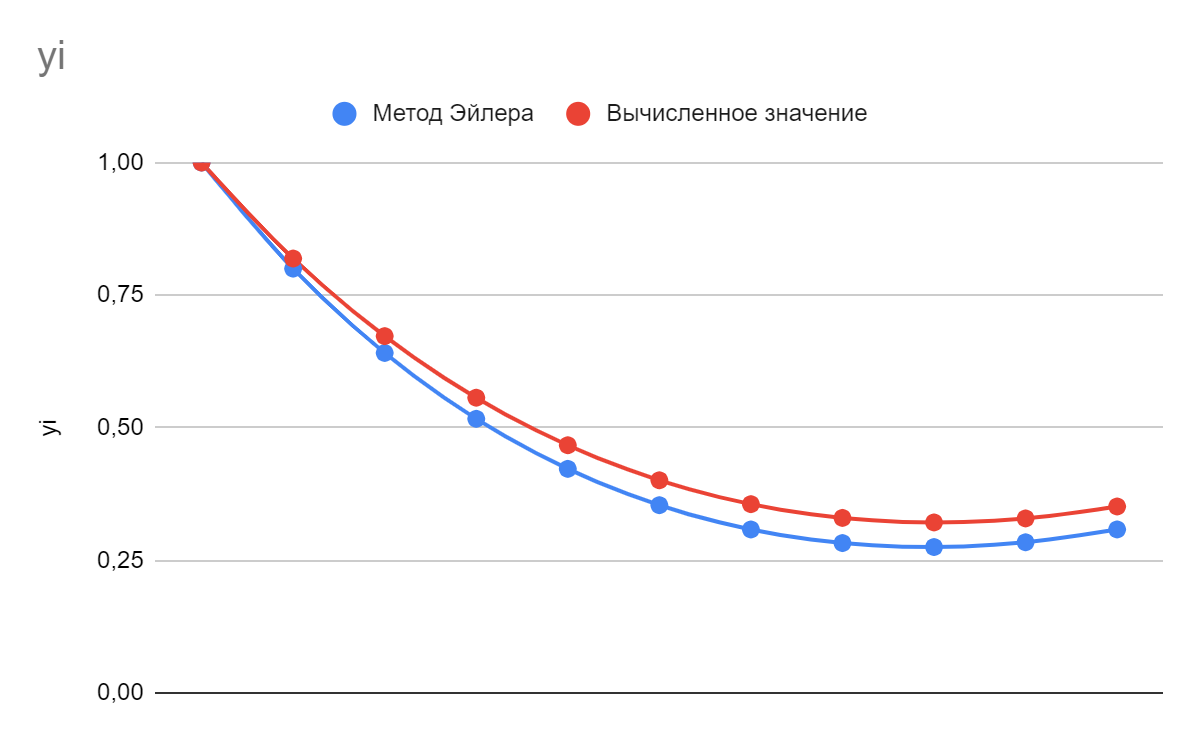
Пример (алгоритм):

;

и т.д.

Вычисления значений удобно записывать в виде таблицы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **i** | **xi** | **yi** | **f(xi, yi)** | **hf(xi; yi)** |
| 0 | 0 | 1, | -2, | -0,2 |
| 1 | 0,1 | 0,8 | -1,59 | -0,159 |
| 2 | 0,2 | 0,641 | -1,242 | -0,1242 |
| 3 | 0,3 | 0,5168 | -0,9436 | -0,09436 |
| 4 | 0,4 | 0,42244 | -0,68488 | -0,068488 |
| 5 | 0,5 | 0,353952 | -0,457904 | -0,04579 |
| 6 | 0,6 | 0,308162 | -0,256323 | -0,025632 |
| 7 | 0,7 | 0,282529 | -0,075059 | -0,007506 |
| 8 | 0,8 | 0,275023 | 0,089953 | 0,008995 |
| 9 | 0,9 | 0,284019 | 0,241963 | 0,024196 |
| 10 | 1 | 0,308215 | 0,38357 | 0,038357 |



По графику видно, что метод Эйлера имеет накапливающуюся погрешность. Чтобы это исправить применяют усовершенствованный метод Эйлера с коррекцией. Суть метода заключается в корректировке вычисления i-того значения:

для повышения точности корректировку можно повторять, подставляя .

Почитать побольше:

* [Метод Эйлера. Усовершенствованный метод Эйлера. Классический метод Рунге-Кутты](http://mathprofi.ru/metody_eilera_i_runge_kutty.html)
* [Метод Эйлера — Википедия](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%AD%D0%B9%D0%BB%D0%B5%D1%80%D0%B0)
* [Метод Рунге — Кутты — Википедия](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%A0%D1%83%D0%BD%D0%B3%D0%B5_%E2%80%94_%D0%9A%D1%83%D1%82%D1%82%D1%8B)