МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ
Государственное бюджетное профессиональное образовательное учреждение Московской области
«Воскресенский колледж»

Краткое методическое пособие для самостоятельной работы по математике для студентов колледжа по специальностям:

40.02.01 - Право и организация социального обеспечения
27.02.07 -Управление качеством продукции, процессов и услуг (по отраслям)
09.02.07 - Информационные системы и программирование

Элементы численного анализа

ЕН.01 Математика

г. Воскресенск , 2020

Настоящее пособие содержит краткое изложение общих понятий численного и аналитического исследований математических моделей.

Пособие рекомендуется использовать студентам дневного и заочного отделений в процессе изучения разделов «Численные методы интегрирования, дифференцирования, аппроксимация и интерполяция» курса математики и подготовки к текущей аттестации по дисциплинам «Математика» и «Информатика».

Содержание:

стр.

1. АНАЛИТИЧЕСКИЕ И ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ . ОБЩИЕ ПОНЯТИЯ.

Математическое моделирование реальных процессов, происходящих в технических, экономических, социальных, биологических и т.п. системах, предполагает создание математической модели в виде уравнений или систем уравнений (линейных, нелинейных, дифференциальных, интегральных, интегро-дифференциальных и т.п), отражающих взаимосвязь наиболее значимых параметров системы. В качестве неизвестных в математических соотношениях фигурируют значимые параметры вышеуказанных процессов. А в качестве исходных данных или граничных условий используются заранее известные значения параметров или соответствующие математические отношения между ними (в виде равенств или неравенств, например).

Таким образом, по итогам решения уравнений исследователь получает числовые значения неизвестных величин, которые определяют состояние системы в зависимости от времени или изменения исходных (внешних) условий. Проводится своеобразный численный эксперимент, в процессе которого определяются те значения параметров внешних условий, при которых значимые параметры системы могут принимать нужные (оптимальные) значения.

Решение уравнений математической модели может быть проведено с использованием известных правил преобразований исходных соотношений вплоть до получения конечного результата. Такой способ решения называется аналитическим. Поскольку преобразования соотношений в процессе такого решения исходных уравнений носят тождественный характер, решение является точным и его погрешность и достоверность определяется степенью достоверности исходных соотношений и допущений (гипотез) о значимости параметров системы при составлении уравнений математической модели. При этом конечное решение представляется в виде некоторой

функциональной зависимости ранее неизвестного параметра системы от значений исходных параметров.

Во многих случаях решение уравнений математической модели аналитическим способом достаточно громоздко или требует выполнения больших объемов однотипных операций (например, решение систем линейных алгебраических уравнений большой размерности методом Гаусса), что может привести к случайным ошибкам в преобразованиях, а иногда просто не существует аналитического решения уравнений (например, трансцендентные уравнения), кроме того, часто в практическом использовании достаточно иметь не точное решение, а решение с некоторой устраивающей исследователя погрешностью. Например, при оценке средней прибыли предприятия за квартал принципиально на результат не повлияют значения единиц рублей (если прибыль считается в сотнях тысяч). В этих случаях для получения решения используются численные методы. Простейшим аналогом численных методов является известный всем еще со школьной скамьи метод графического решения уравнений, по которому точность решения напрямую зависит от точности построения графиков функций. И практически при решении уравнений данным методом никогда не будет получено точное решение. Однако мы можем неограниченно приближаться к нему, используя более совершенные инструменты при построении графических зависимостей. Численные методы решения уравнений «аналитически» менее громоздки, однако требуют большого количества вычислений (арифметических или алгебраических) В зависимости от типа решаемой задачи и требуемой точности решения количество этих вычислений может достигать десятков и сотен миллионов и даже более. Подобный объем вычислений по силам только электронно – вычислительной машине. Поэтому в практике численных решений уравнений математических моделей практически повсеместно используются расчеты на ЭВМ. В настоящее время в курсе численных методов и вычислительной математики разработаны методы численного решения практически большинства математических задач, сформулированных «аналитически» (т.е. представленных в виде уравнений). Это методы Эйлера , Рунге – Кутта решения дифференциальных уравнений, конечно – разностные методы, методы численного интегрирования и т.п.

Какое решение уравнений математической модели является более предпочтительным, аналитическое или численное, в каждом конкретном случае определяется исследователем с учетом вышеперечисленных факторов и требуемой точности получения результатов.

2. ЭТАПЫ РЕШЕНИЯ ЗАДАЧ НА ЭВМ. ПОНЯТИЕ АЛГОРИТМА И ПРОГРАММЫ.

В процессе использования вычислительной техники при решении задач необходимо четко представлять себе тот факт, что ЭВМ является инструментом для решения задачи, и ни в коей мере не идеологом ее решения. Все действия и вычисления, которые будет совершать ЭВМ, определяются командами и условиями, которые задает программист (пользователь) и, в конечном счете, он является истинным идеологом решения задачи. Поэтому ответственность за результат выполнения команд несет ЭВМ (и, как правило, здесь очень маловероятны ошибки !), а за достоверность результатов решения задачи целиком и полностью отвечает программист. Упрощенно схему решения задачи на ЭВМ можно сформулировать следующим образом: программист без ЭВМ уже точно знает последовательность действий для решения задачи, но количество этих действий настолько велико или они настолько примитивно-однообразны, что для их выполнения целесообразно использовать возможности электронно – вычислительной машины.

Все вышесказанное определяет важность разработки последовательности действий для решения поставленной задачи. Учитывая, что впоследствии эти действия будет совершать ЭВМ практически без какого – либо глубокого логического анализа, «слепо» доверяясь интеллекту программиста, необходимо, чтобы указанная последовательность действий была однозначной и четкой. Необходимо также понимать, что данная последовательность действий должна быть доведена до определенного «микроуровня», когда на каждом шаге ее выполнения уже могут быть использованы известные соотношения арифметических и алгебраических операций, которые ЭВМ в состоянии выполнить самостоятельно. Все это, после постановки задачи, самостоятельно выполняет программист, разрабатывая **алгоритм** решения задачи.

После разработки алгоритма, с точки зрения программиста, задача в принципе решена, осталось только провести расчеты, на которые у него просто физически может не хватить ни времени,

ни памяти. Однако логика алгоритма совершенно непонятна ЭВМ. И в этом случае посредником между программистом и ЭВМ выступает алгоритмический язык программирования, который, грубо говоря, понятен ЭВМ. Перевод алгоритма на алгоритмический язык также является задачей программиста. И здесь от его «словарного запаса» (знание синтаксиса и логики языка) во многом зависит рациональность выполнения решения задачи самой ЭВМ с точки зрения временных затрат и использования ресурсов памяти. Таким образом, **вторым этапом** решения задачи является **написание программы** для ЭВМ по разработанному ранее алгоритму.

Итак, текст программы в виде специальных ключевых слов – операторов алгоритмического языка вводится в оперативную память ЭВМ. Не вдаваясь в процесс перевода программы на язык машинного кода (трансляция или компиляция), будем считать, что далее ЭВМ самостоятельно пытается выполнить программу, параллельно проверяя синтаксис и формальную логику программы. При этом, в случае обнаружения ошибок, выдаются соответствующие сообщения на печать или экран монитора. Действия программиста при этом сводятся к исправлениям допущенных ошибок и повторным запуском программы на исполнение. Этот этап заканчивается выходом программы на счет и выдачей некоторого результата. Полученный результат должен быть проверен на достоверность. Для этого необходимо провести тестирование программы на примерах решения аналогичного типа задач, конечный результат решения которых заранее известен и является заведомо достоверным. После того, когда программа положительно отработала несколько тестовых задач, можно считать, что алгоритм и программа работают правильно.

Нередки ситуации, когда программа не выходит на выдачу конечного результата (виснет, циклится), или неправильно считает тестовую задачу. В этом случае рекомендуется в программу вставлять операторы промежуточного вывода значений изменяющихся в процессе счета параметров, что позволяет выявить неточности алгоритма и программы.

Этот этап решения задачи на ЭВМ называется **отладкой программы**.

Последним этапом решения задачи на ЭВМ является многократное использование отлаженной программы для решения множества задач данного класса при различных входных условиях (**непосредственно счет**), поскольку одним из свойств алгоритма является свойство его **универсальности.**

В заключение рассмотрим на простом примере вариант решения задачи аналитическим и численным способом.

Пусть дано уравнение:

$$\mathbf{2X - 6 = 4} \qquad (2.1)$$

2.1 Аналитическое решение.

Используя известные тождественные преобразования, получаем точное решение уравнения :

$$\mathbf{2X = 4+6 \Leftrightarrow 2X =10 \Leftrightarrow X = 5} \qquad (2.2)$$

В данном примитивном примере аналитическое решение не вызывает никаких трудностей (ни вычислительных, ни громоздкости преобразований). Поэтому, естественно, оно является наиболее предпочтительным.

2.2. Численное решение уравнения.

1. Примем допущение о том, что нас интересует корень уравнения, например, с точностью до 0.001. То есть погрешностью в 0.001 в значении корня уравнения можно пренебречь по каким либо - причинам. Приняв указанное допущение, мы должны отдавать себе отчет в том, что все числа в окрестности 5, отличающиеся от числа 5 менее чем на 0.001, будут являться корнем исходного уравнения.

Однако, в процессе решения мы, естественно, получим один корень.

2. Алгоритм поиска корня уравнения построим основываясь на том, что в графическом представлении функция, переходя через нулевое значение, меняет знак. В случае данного уравнения функция

$$\mathbf{f(x) = 2X - 4 - 6 \Leftrightarrow f(x) = 2X - 10} \qquad (2.3)$$

И общий вид уравнения в нашем случае:

$$\mathbf{f(x) = 0} \qquad (2.4)$$

Допустим, из некоторых априорных (предварительных) сведений о поведении значимой величины **X**, моделируемой

указанным уравнением, нам известно, что корень его находится в интервале (0; 100). Например, уравнение описывает изменение прибыли некоторого безубыточного предприятия и X – есть текущее значение прибыли за квартал в некоторых условных единицах.

Далее, с шагом 0.001, начиная с 0 будем вычислять значения **f(x)** и **f(x+0.001)**, сравнивая знаки полученных числовых результатов. Для этого достаточно рассмотреть значение произведения **Z=f(x)*f(x+0.001)** . Если это произведение меньше нуля, то в интервале **(x; x+0.001)** функция **f(x)** поменяла знак, то есть принимала нулевое значение. А это значит, что в указанном интервале есть корень исходного уравнения. За него можно принять либо **x**, либо **x+0.001**, и вообще любое число из интервала **(x; x+0.001)**. Требования по погрешности в определении корня, как уже говорилось выше, будут выполнены.

Но возможен случай, когда значение **Z** оказывается равным нулю. Это значит, что на очередном шаге по **x**, либо **f(x) =0**, либо **f(x+0.001)=0**. То есть либо **x** , либо **x+0.001** являются точными корнями исходного уравнения. В этом случае достаточно просто проверить какой аргумент функции **x** или **x+0.001** дает ей нулевое значение, и затем принять его в качестве корня уравнения.

3. При кажущейся громоздкости алгоритма программа для ЭВМ, например на алгоритмическом языке Q_BASIC, будет достаточно простой:

```
   X=0
   S=0.001
2  Z=(2*X –10)*(2*(X+S) – 10)
   IF (Z) < 0 THEN  GOTO  3
   IF (Z) = 0 THEN GOTO 4
    X = X + S
    IF(X+S) <=100  THEN GOTO 2
    GOTO 5
3  PRINT “КОРЕНЬ =”,  X +S/2
4  IF (2*X  - 10 ) = 0  THEN  PRINT “КОРЕНЬ =”, X
   IF (2*(X + S)  - 10 ) = 0  THEN  PRINT “КОРЕНЬ =”, (X +S)
5  END
```

Приведенный пример наглядно демонстрирует принципы аналитического и численного решения уравнений. На первый взгляд может показаться, что в данном примере численное решение по сравнению с аналитическим существенно более

трудоемко. Так оно и есть, поскольку в качестве тестового уравнения взято элементарное линейное уравнение школьного курса алгебры.

Однако, если взять более сложное уравнение, например:

$$\log_x (x-\cos(x)) = \sin(x), \qquad (2.5)$$

которое аналитически решить достаточно сложно, численное решение будет вполне оправдано.

В качестве рекомендаций студентам при изучении данной темы следует отметить, что наиболее важно здесь понять принципиальное отличие аналитических и численных методов решения задач, уяснить этапы решения, понятия алгоритма и программы для ЭВМ. При этом, приведенная выше программа для ЭВМ является чисто демонстрационной и приведена для обеспечения полноты изложения материала. Изучение же самого алгоритмического языка программирования составляет отдельную тему последующих занятий.

3. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ НЕКОТОРЫХ ПРИКЛАДНЫХ ЗАДАЧ

3.1. Поиск с заданной точностью корней уравнения на заданном интервале . Метод дихотомии (половинного деления)

Пусть на интервале **[A; B]** задано некоторое уравнение вида:

$$f(x) = 0, \qquad (3.1.)$$

где **f(x)** в общем случае - произвольная непрерывная функция, определенная на заданном интервале. Известно из априорных соображений, что уравнение (3.1.) на данном интервале имеет несколько корней, причем функция **f(x)** ведет себя на этом интервале достаточно гладко в смысле небольших абсолютных значений производной.

Требуется найти все корни данного уравнения на указанном интервале с заданной точностью **ε**.

Алгоритм решения поставленной задачи построим в два этапа:

- на первом этапе на интервале **[A; B]** найдем интервалы **[a_i ; b_i] (i=1,2,3,… n)**, в каждом из которых функция **f(x)** меняет знак только один раз. То есть в каждом из найденных интервалов уравнение (3.1.) имеет только один корень.
- на втором этапе в каждом из найденных интервалов **[a_i ; b_i]** найдем корень исходного уравнения (3.1.) с заданной точностью **ε.**

Рассмотрим графическую иллюстрацию поставленной задачи.

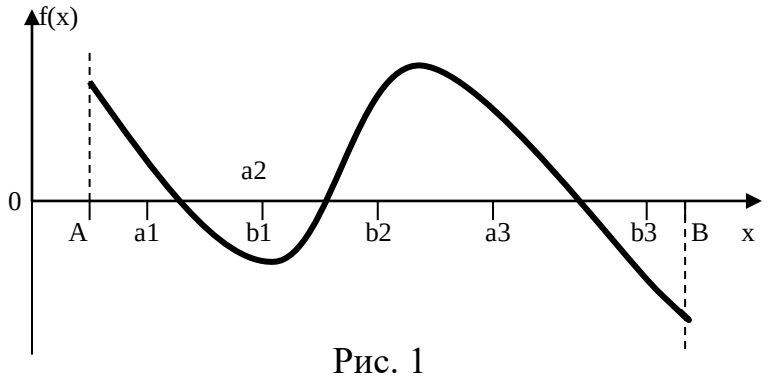

Рис. 1

На рисунке показаны интервалы изменения аргумента **[a_i ; b_i] (i=1,2,3)**, на которых функция один раз меняет знак и общий интервал **[A; B]** поиска корней уравнения.

Поиск интервалов изменения знака функции **f(x)**, где она только один раз меняет знак, будем проводить с использованием алгоритма, рассмотренного выше при численном решении уравнения (1.1). То есть интервал смены знака функции будем определять, анализируя знаки ее значений при продвижении по аргументу с некоторым шагом **δx** слева направо в пределах интервала **[A; B]**.

Начальным значением аргумента является левая граница интервала **x = A**. Следующим значением аргумента является его значение, увеличенное на **δx**, то есть **x = A+δx.** В обоих точках по **x** вычисляем значения функции и смотрим знак произведения **Z = f(x)*f(x+δx)**. В случае положительного знака **Z** (то есть **f(x)** и **f(x+δx)** имеют одинаковые знаки) считаем, что на интервале **[x; x+ δx]** функция не меняет знак. В этом случае далее выполняем аналогичную процедуру, принимая за начальное значение **x = x+ δx** до тех пор, пока не достигнем правой границы заданного интервала **[A; B]**. То есть условием окончания расчетов будет выполнение

условия: **(x+δx) > B** (условие выхода за правую границу исследуемого интервала). В общем случае поведения функции **f(x)** может оказаться, что на некотором I-ом шаге знак произведения **Z** окажется отрицательным. Это свидетельствует о том, что на интервале **[x_i; x_i + δx]** функция **f(x)** сменила знак, то есть прошла через нулевое значение. В этом случае фиксируем интервал смены знака функции **[a_i; b_i]**, где **a_i = xi**, а **b_i = x_i+ δx** . Далее, принимая за начальное значение **x = b_i** , продолжаем вышеописанную процедуру. В результате реализации данного алгоритма мы должны получить сообщение о том, что данная функция на интервале **[A; B]** не меняет знак, или сформировать одномерные массивы **{a_i}** и **{b_i}, i = 1,2,3,….n** границ интервалов, в которых функция меняет знак только один раз.

Следует отметить, что в реализации данного алгоритма шаг **δx** по аргументу выбирается не совсем произвольно, а с учетом некоторых сведений о поведении исследуемой функции, так, чтобы с одной стороны не увеличивать неоправданно количество вычислений (при очень маленьком шаге **δx**) и, с другой стороны, не пропустить интервал изменения ее знака (при слишком большом **δx**).

Итак, будем считать, что первый этап алгоритма решения поставленной задачи реализован. И у нас имеются все интервалы в пределах исходного **[A; B]**, где функция один раз меняет знак.

Проиллюстрируем работу алгоритма на этапе поиска корня исходного уравнения с заданной точностью **ε** на интервале **[a_i; b_i]** графически.

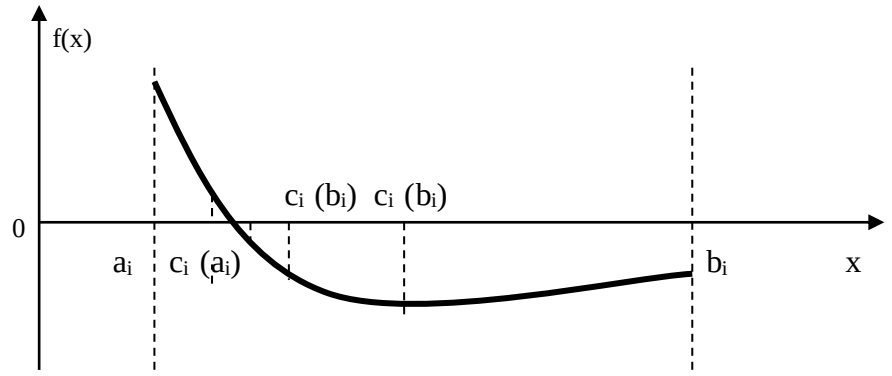

рис. 2

Алгоритм уточнения корня уравнения в заданном интервале строится следующим образом:

- интервал **[a_i; b_i]** делится пополам (находим точку **c_i = (a_i + b_i)/2)**
- определяем знак соотношения **Z = f(a_i)*f(c_i)**

- анализируем знак **Z**: если **Z< 0**, то «сдвигаем» правую границу интервала $b_i = c_i$, если **Z>0**, то «сдвигаем» левую границу интервала $a_i = c_i$
- далее повторяем первый пунк алгоритма
  В результате организованного цикла интервал $[a_i; b_i]$ будет неограниченно сужаться, причем внутри его будет оставаться ноль функции **f(x)**, то есть корень исходного уравнения (3.1).

Цикл расчетов прекращаем по условию, когда интервал вокруг искомого корня сузится настолько, что значения границ интервала и истинное значение корня будут отличаться друг от друга на величину меньшую, чем значение погрешности **ε.** Математическое условие окончания цикла расчетов будет выглядеть следующим образом:

$$ABS (a_i - b_i) <= \varepsilon \quad (3.2),$$

где **ABS** – знак абсолютной величины.

На рис. 2 показаны четыре цикла работы вышеописанного алгоритма. Причем, в скобках после обозначения середины интервала **C** стоит обозначение той границы (a_i или b_i), которая, “перемещаясь” в точку **C**, приводит к требуемому сужению интервала расположения корня уравнения.

Приведенный метод половинного деления (дихотомии) в практическом использовании имеет некоторые ограничения, связанные с поведением функции на исходном интервале **[A; B]** поиска корней уравнения. В первую очередь исходная функция **f(x)** не должна «часто» менять знак на указанном интервале, так, чтобы при определении интервалов одноразовой смены ее знака с шагом **δx** не пропустить смену знака. Это возможно, когда функция **f(x)** на интервале **δx** меняет знак более чем один раз. Во – вторых, функция **f(x)** может вообще не менять знака на интервале **[A; B]**, однако корень исходного уравнения будет присутствовать на указанном интервале. Это случай касания графика функции оси абсцисс.

В первом случае достаточно рационально выбрать шаг **δx** при реализации алгоритма с учетом предварительных сведений о поведении функции. Во втором же случае нужно использовать иной метод численного решения задачи (например, метод хорд или касательных), поскольку предложенный метод может привести к неверному результату.

3.2 Вычисление определенного интеграла . Метод прямоугольников и трапеций.

Пусть требуется вычислить определенный интеграл от функции **f(x)** на интервале **[a;b]** с некоторой точностью **ε**. Функция **f(x)** предполагается непрерывной на заданном интервале.

$$I = \int_a^b (f(x))dx \qquad (3.3)$$

Известно, что значение определенного интеграла численно равно значению площади, ограниченной графиком функции, осью абсцисс и прямыми **x = a ; x = в**. При этом необходимо отметить, что площадь считается положительной, если она расположена выше оси **OX**, и отрицательной, если наоборот.

Построение алгоритма расчетов поясним графически:

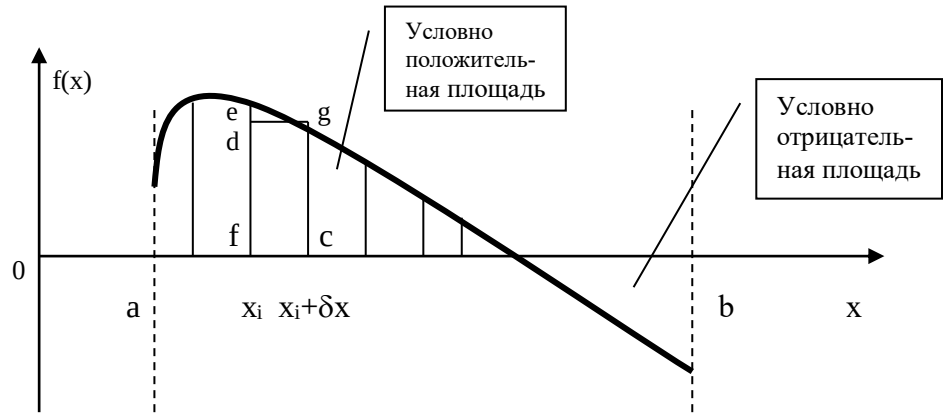

рис. 3

Фактически для вычисления интеграла (3.3) нам необходимо посчитать площадь под кривой **f(x)** над и под осью **OX** , а затем сложить их значения с учетом знаков. Для организации алгоритма расчетов используем следующие очевидные соотношения:

$$S = \lim_{n \to \infty} \sum_{i=1}^{n} (((f(x_i)+f(x_i+\delta x))/2)* \delta x) \qquad (3.4)$$

или

$$S = \lim_{n \to \infty} \sum_{i=1}^{n} (f(x_i)* \delta x) \qquad (3.5)$$

где $f(x_i)$ и $f(x_i+\delta x)$ – значения функции в точках, отстоящих друг от друга на величину шага **δx**.

Соотношение (3.4) соответствует расчету площади путем суммирования площадей элементарных трапеций **fcge**, а соотношение (3.5) - путем суммирования площадей элементарных прямоугольников **fdgc** (рис. 3). Поскольку величина **δx** является положительной, то знак площади при расчете ее по соотношениям (3.4) или (3.5) будет учитываться автоматически по знаку значения функции.

Соотношения (3.4) и (3.5) дают точное значение площади, поскольку при стремлении **n** к бесконечности (**n** здесь определяет количество точек, на которые разбивается интервал интегрирования) неограниченно уменьшается величина **δx** и неограниченно увеличивается количество элементарных трапеций или прямоугольников, площади которых суммируются при вычислении значения интеграла.

Для расчетов на ЭВМ соотношения (3.4) или (3.5) непосредственно использовать нельзя, поскольку "машина не умеет вычислять предел". Поэтому в алгоритме расчетов используются не предельные (точные), а конечные соотношения:

$$S = \sum_{i=1}^{n} \left(\left(f(x_i) + f(x_i + \delta x) \right)/2 \right) * \delta x \qquad (3.6)$$

$$S = \sum_{i=1}^{n} f(x_i) * \delta x \qquad (3.7),$$

которые вычисляют значение площади (интеграла) с некоторой погрешностью, поскольку здесь значение **n** конечно, а значит, и количество элементарных трапеций или прямоугольников тоже является конечным числом. Абсолютная погрешность в данном случае численно равна сумме неучтенных (непросуммированных) площадок. Таких, как треугольник **dge** при вычислении интеграла по соотношению (3.7) например (рис. 3).

При каждом значении **n** соотношения (3.5) или (3.6) дадут некоторое значение площади (интеграла). С увеличением **n** точность его определения будет возрастать, однако погрешность все равно останется, поскольку количество суммируемых площадей трапеций или прямоугольников будет конечным. При этом, как видно из рис. 3, более точным оказывается расчет интеграла по соотношению (3.6) (**метод трапеций**), чем по (3.7) (**метод прямоугольников**).

Таким образом, ясно, что, уменьшая шаг **δx**, (увеличивая количество точек разбиения интервала интегрирования) мы будем неограниченно приближаться к точному значению интеграла. Однако при этом и количество выполняемых операций ЭВМ будет

существенно возрастать, а также и время счета. Кроме того, при работе с существенно – малыми величинами, каким может явиться **δx,** и при большом объеме арифметических операций возрастает общая погрешность результата расчетов за счет так называемой машинной погрешности округления чисел. То есть слишком малый шаг интегрирования заведомо может привести к неверному результату. Возникает вопрос: какой же шаг **δx** взять? Предлагается следующий метод выбора шага интегрирования и получения значения интеграла с заданной точностью **ε:**

1. берем произвольный шаг **δx**, естественно, не превышающий размеры интервала интегрирования
2. вычисляем по соотношениям (3.6) или (3.7) значение интеграла S_1
3. уменьшаем шаг интегрирования в 2 раза
4. вычисляем значение интеграла S_2 с новым шагом интегрирования
5. сравниваем значения S_1 **и** S_2 , используя соотношение **ABS(**S_1 **-** S_2**)**
6. если полученная величина оказывается меньше заданной погрешности **ε** вычисления интеграла, то останавливаем расчеты и выводим на печать полученное значение
7. если же **ABS(**S_1 **-** S_2**)** **>** **ε**, то еще раз уменьшаем шаг интегрирования в 2 раза и повторяем процедуру вычислений п.п. 4, 5, 6, принимая за значение S_1, ранее вычисленное значение S_2
8. повторяем цикл расчетов до тех пор, пока не будет выполнено требование 6.

Смысл вышеописанных вычислений заключается в том, что, вычисляя интеграл с разными шагами интегрирования и уменьшая шаг, мы на каком – то этапе замечаем, что дальнейшее уменьшение шага практически не изменяет результат вычислений. То есть предыдущее значение интеграла, вычисленное с шагом **δx,** ничтожно мало отличается от его последующего значения, вычисленного с шагом **δx/2.** Здесь просматривается аналогия шлифовки поверхности при визуальном контроле качества. Когда на определенном этапе шлифовки последующее шлифование визуально не меняет внешний вид поверхности, шлифование прекращается.

3.3 Численное дифференцирование. Вычисление производной функции в точке.

Пусть дана функция **f(x)**, непрерывная на некотором интервале **[A;B]**.

Необходимо с некоторой точностью **ε** вычислить производную данной функции в точке x_0 , принадлежащей данному интервалу.

Из определения производной :

$$f'(x_0) = \lim_{\delta x \to 0} ((f(x_0 + \delta x) - f(x_0))/ \delta x \quad (3.8)$$

следует, что если в выражении (3.8) опустить знак предела,

$$f'(x_0) = ((f(x_0 + \delta x) - f(x_0))/ \delta x \quad (3.9)$$

то при каждом конечном значении δx, используя соотношение (3.9), мы будем получать некоторое число, которое будет приближаться к значению производной данной функции в точке x_0 при уменьшении шага δx.

Соотношение (3.9) называется конечно – разностным аналогом выражения (3.8).

В определении значения производной функции в точке по конечно – разностному аналогу с заданной точностью ε будем использовать алгоритм, аналогичный определению с заданнной точностью определенного интеграла численным методом.

То есть,

1. В заданной точке x_0 по соотношению (3.9) при первоначально заданном шаге δx_1 определяем значение $f'_1(x_0)$.
2. Затем, уменьшая шаг в два раза ($\delta x_2 = \delta x_1/2$) , вновь определяем значение производной в точке x_0 , $f'_2(x_0)$.
3. Проверяем условие: $ABS(f'_1(x_0) - f'_2(x_0)) < \varepsilon$
4. Если условие (3) выполнимо, то можно считать, что значения $f'_1(x_0)$ или $f'_2(x_0)$ являются искомым значением производной в заданной точке x_0. При этом на печать с целью несущественного снижения погрешности рекомендуется выводить среднее арифметическое $(f'_1(x_0) + f'_2(x_0))/2$.
5. Если же условие (3) не выполняется, то последующими шагами алгоритма будет выполнение пунктов (1), (2), и (3), принимая за первоначальный шаг, шаг, взятый при последующем определении производной ($\delta x_1 = \delta x_2$), до тех пор, пока не будет выполнено соотношение (3).

   В качестве рекомендаций по выбору первоначального шага δx_1 определения производной можно сказать следующее:

   - первоначальный шаг не должен быть настолько большим, чтобы в его пределах функция имела хотя бы один экстремум или точку разрыва;
   - с другой стороны, первоначальный шаг не должен быть настолько малым, чтобы при последующих его уменьшениях (делении пополам), ЭВМ в расчетах не «выскочила» за машинный ноль.

   Вообще говоря, второе ограничение по выбору шага напрямую связано с выбором точности ε определения

производной. То есть при задании очень малой погрешности **ε** вероятность того, что при последующих уменьшениях шага в процессе расчета значения производной, можно получить настолько малое значение шага, что при округлении ЭВМ примет его за ноль, что приведет к «зацикливанию» или остановке в выполнении программы.

Следует заметить, что подход к оценке погрешности вычисления первой производной функции в точке по конечно – разностному аналогу (3.9) может быть иным.

Используем геометрический смысл первой производной, заключающийся в том, что ее значение численно равно тангенсу угла наклона касательной к положительному направлению оси абсцисс, проведенной к графику функции **f(x)** в точке определения производной. При этом очевидно, что значение производной, определенное по конечно – разностному аналогу, численно равно тангенсу угла наклона некоторой секущей к положительному направлению оси абсцисс.

Поясним сказанное графически.

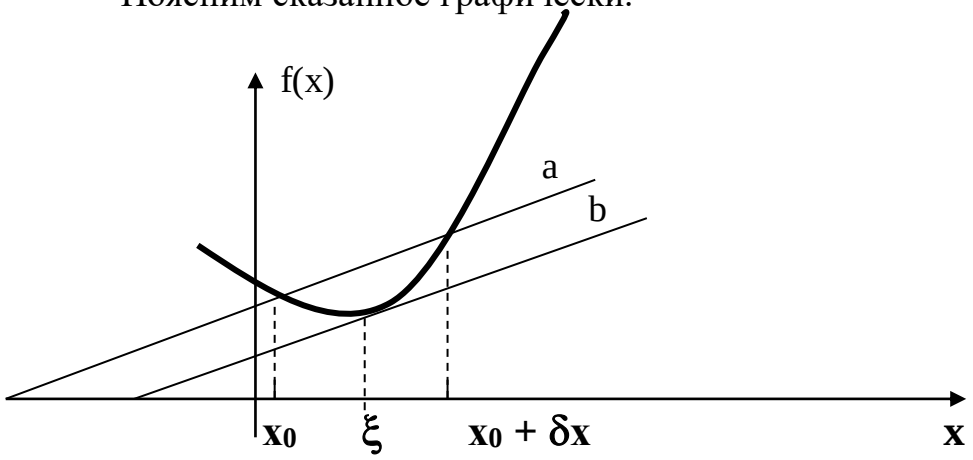

рис. 4

На рис. 4 видно, что внутри интервала **[x0 ; x0 + δx]** существует точка **ξ** , в которой касательная **b**, проведенная к графику функции, параллельна секущей **a** . Этот факт является следствием теоремы Лагранджа для непрерывно – дифференцируемой функции на отрезке. Но если две прямые на плоскости параллельны, то их угловые коэффициенты, а значит, и тангенсы углов наклона к положительному направлению оси абсцисс равны. В нашем случае тангенсы углов наклона этих прямых численно равны значениям производной функции в точке **x0** (при определении ее по конечно – разностному аналогу (3.9)) и точному значению производной данной функции в точке **ξ**.

Таким образом, можно считать, что используя соотношение (3.9), мы точно определяем значение первой производной функции f(x) на отрезке **$[x_0 ; x_0 + \delta x]$**. А точку, в которой определяется значение этой производной, определяем с некоторой погрешностью. Причем мы точно знаем, что эта точка располагается внутри данного интервала. Поэтому можно считать, что максимальное значение абсолютной погрешности в определении точки ξ не превышает длины данного интервала δx..

3.4. Линейная интерполяция функций, заданных таблично

Пусть задана табуляция некоторой функции в виде таблицы ее значений в некоторых точках аргумента **$f(x_i)$, $i = 1,2,3,\ldots,n$**. Ставится задача определить значение данной функции в произвольной точке x_j, принадлежащей интервалу **$[x_1; x_n]$** . При этом априори известно, что исходная функция на любом интервале между точками **$[x_i; x_{i+1}]$, $i=1,2,\ldots n-1$** в поведении близка к линейной функции.

Словесное объяснение алгоритма решения поставленной задачи сводится к следующему. Поскольку на интервале **$[x_i; x_{i+1}]$, $i=1,2,\ldots n-1$** заданная функция ведет себя линейно, принимая в точках $x_i; x_{i+1}$ заданные в таблице значения, то достаточно на каждом интервале составить уравнение прямой, соединяющей точки **$(x_i , f(x_i)); (x_{i+1} ,f(x_{i+1}))$** и затем в одно из этих уравнений (в зависимости от того в какой интервал попадает текущее значение x_j) подставить текущее значение x_j для получения требуемого значения функции **$f(x_j)$**.

Поясним сказанное графически:

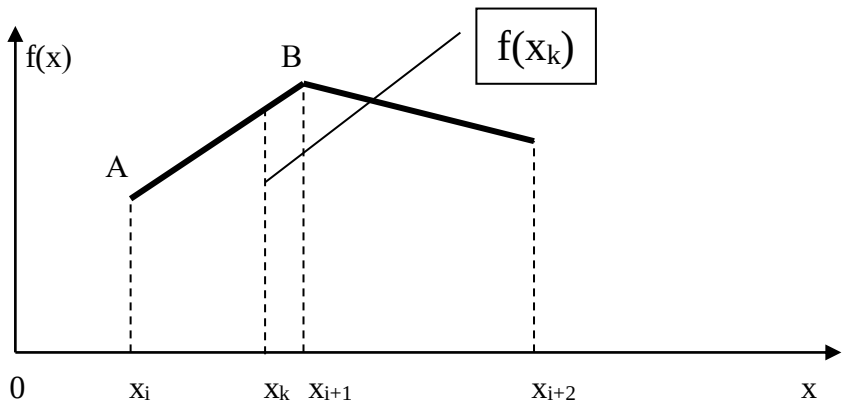

рис. 5

На рис. 5 показан графически принцип линейной интерполяции функции, заданной точками на плоскости **XOY**. В итоге мы получаем непрерывную функцию после соединения заданных точек прямыми. Хотя следует отметить, что уже первая производная полученной функции будет иметь разрывы, то есть полученная функция не является гладкой, поскольку в точках интерполяции функция имеет изломы.

Для записи уравнения прямой, проходящей через две заданные точки, используем стандартное уравнение прямой на плоскости:

$$y = b*x + c \qquad (3.10),$$

где **b** – угловой коэффициент, **c** – свободный член.

Поскольку прямая (3.10) проходит через точки $(x_i , f(x_i)); (x_{i+1} ,f(x_{i+1}))$ (точки **A** и **B**, рис. 4), то координаты этих точек должны удовлетворять уравнению (3.10).

То есть:

$$\begin{aligned} f(x_i) &= b* x_i + c \\ f(x_{i+1}) &= b* x_{i+1} + c \end{aligned} \qquad (3.11)$$

Значения $(x_i , f(x_i)); (x_{i+1} ,f(x_{i+1}))$ являются заданными, и система (3.11) после ее решения позволяет определить неизвестные **b** и **c** в уравнении прямой (3.10).

Решение системы (3.11) дает:

$$\begin{aligned} b &= [f(x_{i+1}) - f(x_i)]/[x_{i+1} - x_i] \\ c &= f(x_i) - [f(x_{i+1}) - f(x_i)]/[x_{i+1} - x_i]* x_i \end{aligned} \qquad (3.12)$$

Таким образом, искомое уравнение прямой (3.10) может быть записано в следующем виде:

$$\begin{aligned} y = {} & [f(x_{i+1}) - f(x_i)]/[x_{i+1} - x_i]*x + f(x_i) \\ & - [f(x_{i+1}) - f(x_i)]/[x_{i+1} - x_i]* x_i \end{aligned} \qquad (3.13)$$

Здесь **x** есть текущее значение аргумента, изменяющегося в интервале $[x_i; x_{i+1}]$, **i=1,2,…n-1**.

Соотношение (3.13) определяет интерполяционную линейную функцию для каждого интервала, заданного абсциссами точек табуляции. Если в выражение (3.13) подставить вместо x некоторое текущее значение аргумента x_j , то мы получим соответствующее значение функции. Однако, следует учесть, что выражение (3.13) определяет значение функции для тех значений x_j , которые попадают в интервал $[x_i; x_{i+1}]$, $i=1,2,\ldots n-1$, где **n** - количество точек интерполяции. То есть после задания текущего значения x_j , необходимо определить, в какой интервал $[x_i; x_{i+1}]$, $i=1,2,\ldots n-1$ это значение попадает. После чего вычислить соответствующие коэффициенты **b** и **c** для соответствующего уравнения прямой (3.13). При этом, естественно, уравнение прямой для каждого интервала будет в общем случае различным. Различие состоит в разных коэффициентах **b** и **c** в уравнениях прямых для каждого интервала.

Рекомендации по разработке алгоритма решения вышепоставленной задачи сводятся к следующему:

1. Организуется ввод массивов табличных значений аргумента **X(I)** и **Y(I)** из исходных данных табуляции функции, **i= 1,2,3,… n**.
2. Вычисляются и формируются массивы коэффициентов **B(I)** и **C(I)** для каждого интервала $[x_i; x_{i+1}]$, $i=1,2,\ldots n-1$ по соотношениям (3.12).
3. Задается текущее значение аргумента x_k .
4. Определяется интервал по оси **OX** в который попадает заданное значение x_j путем проверки в цикле условия:

$$x_i <= x_k >= x_{i+1}\ ,\ i = 1,2,3,\ldots n-1$$

i в данном случае идентифицирует номер интервала при их нумерации слева направо.

5. После определения номера интервала **i** по соотношению (3.13) вычисляется значение функции, соответствующее заданному текущему значению аргумента x_k .