§ 1. Введение.

Развитие новых технологий и широкое внедрение математических методов в инженерные исследования, а также рост числа выпускаемой вычислительной техники и повышение её качества (производительности), привели к широкому использованию персональных компьютеров во многих областях производства.

В настоящее время успешное разрешение большинства научно-технических проблем или проектно-технологических заданий в значительной степени зависит от умения инженера (проектировщика) оперативно и квалифицированно применять ЭВМ и соответствующее программное обеспечение.

Для решения многих таких задач уже разработан огромный арсенал численных методов, подкрепленный соответствующим математическим анализом, позволяющим оценить точность получаемых решений и их адекватность изучаемым процессам.

Однако наличие и использование ЭВМ не снимает всех проблем, возникающих в ходе моделирования и решения различного рода задач.

Это связано, прежде всего, с тем, что процесс решения научно-исследовательской задачи проходит целый ряд стадий или этапов, каждый из которых имеет свою специфику и оказывает свое влияние на достоверность окончательного результата.

Решение практической задачи начинается с формализации и описания исходных данных, а также формулировки целей задачи на языке строго определенных математических понятий – это ***математическая постановка задачи***.

Выделяя наиболее существенные свойства (характерные черты) реального физического процесса, исследователь описывает его с помощью математических соотношений, моделирующих законы природы.

Этот этап решения называется построением ***математической модели***.

После этого, в рамках уже имеющейся математической модели, осуществляется поиск метода решения задачи и строится его алгоритм.

Этап поиска и разработки алгоритма решения называют ***алгоритмизацией***.

Здесь используются любые формы представления алгоритмов: словесные описания, математические формулы и блок-схемы.

Во многих случаях вслед за построением алгоритма выполняют так называемый контрольный просчет – грубую предварительную оценку ожидаемых результатов, которые используются затем для анализа полученного решения.

На следующем этапе алгоритм решения задачи записывается на языке, понятном компьютеру и предназначенному для вычислений. Это – этап ***программирования***.

В простейших случаях может оказаться, что на этом этапе вовсе и не составляется принципиально новая программа для ЭВМ, а дело сводится к использованию какого-

либо имеющегося математического обеспечения специального назначения, например, программной оболочки **MathCad**.

Далее идут этапы ***отладки и исполнения программы*** на компьютере и получение результатов расчетов в виде наборов числовых данных, графиков, диаграмм и т.п.

Время, требуемое на успешное прохождение этого этапа работ, зависит в первую очередь от уровня квалификации программиста, а также от объема необходимых вычислений и быстродействия используемого компьютера.

Завершающий этап решения задачи – ***анализ*** (или интерпретация) ***результатов***.

Здесь происходит осмысление полученных результатов, сопоставление их с результатами контрольного (тестового) просчета, а также с данными, полученными экспериментальным путем (если таковые вообще имеются).

При этом одни результаты могут оказаться приемлемыми, а другие – противоречащими физическому смыслу реальной задачи (их следует отбросить).

Высшим критерием пригодности полученных результатов, в конечном счете, является практика (натурный эксперимент, опыт).

Наиболее сложным и ответственным этапом решения является построение физико-математической модели рассматриваемого явления или процесса.

Если выбранная математическая модель слишком грубо отражает взаимосвязи изучаемого процесса, то, какие бы изощренные математические методы решения вслед за этим не применялись, найденные значения не будут отвечать условиям реальной задачи и окажутся, вообще говоря, бесполезными.

Математическая модель может иметь вид уравнения, системы уравнений или быть выраженной в форме иных, как угодно сложных, математических структур или соотношений самой различной природы.

Математические модели, в частности, могут быть непрерывными или дискретными, в зависимости от того, какими числовыми функциями – непрерывными или дискретными – они были описаны (заданы).

Своеобразные трудности вызывает также этап разработки алгоритма, суть которых – в поиске приемлемого метода решения задачи.

Дело в том, что уже даже для достаточно простых моделей иногда не удается получить результат решения в аналитической форме.

Пусть, например, исследуемая задача свелась к решению уравнения с одной переменной $2x - \cos(3x) = 0$.

При всей тривиальности этой задачи выразить корни данного уравнения путем алгебраических преобразований не удается, а графический метод необходимой точности результата, очевидно, не дает.

В таких случаях приходится использовать численные методы, позволяющие получать результаты путем реализации последовательности вычислений.

В условиях использования компьютера численные методы выступают как мощное математическое средство решения практических задач.

При этом важно иметь в виду, что фактор использования компьютера не упрощает, а в некотором смысле даже усложняет решение вопросов оценки точности и адекватности получаемых результатов (в виду резкого возрастания количества выполняемых арифметических операций).

Суть возникающих здесь проблем подмечена в известном принципе: «***Компьютер многократно увеличивает некомпетентность вычислителя***».

Из этого остроумного замечания следует, что, используя для решения задачи компьютер, вычислитель не столько должен полагаться на мощность его процессора, сколько помнить о необходимости собственного понимания сущности того, что в конечном итоге он получает от компьютера «на выходе».

Помимо отмеченных выше субъективных факторов, влияющих на качество конечного результата вычислений, в теории выделяются и объективные причины, приводящие к потере точности.

Общая погрешность решения задачи складывается из нескольких факторов.

Отметим основные из них, пользуясь рассмотрением общего хода решения задачи – от построения математической модели и до производства вычислений.

Пусть R – точное значение результата решения некоторой задачи (виртуальный идеальный результат).

Из-за несоответствия выбранной математической модели для реального физического процесса, а также по причине неточности (приближенности) исходных данных, вместо результата R будет получен другой результат R_1.

Образовавшаяся таким образом погрешность $\varepsilon_1 = R - R_1$ уже не может быть устранена в ходе последующих вычислений или математических преобразований.

Эту погрешность так и называют – ***неустранимая погрешность***.

Приступив к решению задачи в рамках математической модели, мы выбираем приближенный (например, численный) метод и, еще не приступив к вычислениям, допускаем новую погрешность, приводящую к получению результата R_2 (вместо R_1).

Погрешность этого этапа $\varepsilon_2 = R_1 - R_2$ называют ***погрешностью метода***.

И, наконец, неизбежность округления данных компьютером (его процессором) приводит к получению результата R_3, отличающегося от результата R_2 на величину ***вычислительной погрешности*** $\varepsilon_3 = R_2 - R_3$.

Округление производится либо путем отбрасывания цифр младших разрядов, либо путем «правильного округления» результата.

Тот или иной способ округления зависит от архитектуры конкретного процессора и реализованной в нем двоичной арифметики.

Полная погрешность ε получается как «сумма» всех погрешностей:

$$\varepsilon = \varepsilon_1 + \varepsilon_2 + \varepsilon_3.$$

Если вместо погрешностей ε_1, ε_2 и ε_3 удается получить лишь их абсолютные верхние оценки Δ_1, Δ_2, Δ_3, то приходится довольствоваться оценочным представлением общей погрешности в виде $\varepsilon \le \Delta_1 + \Delta_2 + \Delta_3$.

При решении конкретных задач те или иные виды погрешностей могут либо отсутствовать совсем, либо влиять на окончательный результат незначительно.

Тем не менее, для исчерпывающего представления о точности окончательного результата в каждом случае необходим полный анализ погрешностей всех видов.

Это в полной мере относится и к неустранимой погрешности – погрешности математической модели.

Располагая несовершенной математической моделью, вычислитель все-таки должен любым способом составить представление о величине неустранимой погрешности.

Понятно, что в условиях слишком грубой модели не имеет смысла проведение уточненного анализа вычислительных ошибок.

Отсюда следует, что оценка величины неустранимой погрешности может послужить удобным поводом для понижения требований к точности последующих вычислений и упрощения используемых числовых алгоритмов.

Как правило, качество и корректность разработанных численных алгоритмов проверяется при помощи так называемых тестовых задач, для которых имеется точное аналитическое решение, истинность которого не вызывает сомнений.

Этап тестирования обычно совмещается с этапом отладки программного продукта.

Тестирование алгоритма (и его программного кода) считается успешно завершенным, когда полученный результат вычислений совпадает с истинным решением задачи с точностью до заданной погрешности округления действительных чисел.

Этот важный этап разработки численных алгоритмов будет нами детально изучен в ходе выполнения лабораторных работ и контрольных заданий.

Умение разрабатывать тестовые задачи и отлаживать с их помощью численные алгоритмы, безусловно, является одним из основных критериев квалифицированности современного разработчика компьютерных программ.

§ 1.1. Абсолютная и относительная погрешности.

Если ξ – точное значение некоторой величины, а ξ^* – известное приближение к нему, то ***абсолютной погрешностью*** приближения ξ^* называют обычно некоторую величину $\Delta(\xi^*)$, про которую известно, что

$$|\xi^* - \xi| \le \Delta(\xi^*). \tag{1.1}$$

Относительной погрешностью называют некоторую величину $\delta(\xi^*)$, про которую известно, что

$$\frac{|\xi^* - \xi|}{|\xi^*|} \le \delta(\xi^*). \tag{1.2}$$

Относительную погрешность часто выражают в процентах.

Значащими цифрами числа называют все цифры в его записи, начиная с первой ненулевой слева.

Например, в следующих записях у чисел $\xi^* = 0,0\underline{3045}$ и $\xi^* = 0,0\underline{3045000}$ значащими цифрами являются подчеркнутые цифры.

Количество значащих цифр в первом случае равно **4**, во втором **7**.

Значащую цифру называют ***верной***, если абсолютная погрешность числа не превосходит единицы разряда, соответствующего этой цифре.

Например, если $\xi^* = 0,0\underline{3045}$ и $\Delta(\xi^*) = 0,000003$, и если $\xi^* = 0,0\underline{3045}000$ и $\Delta(\xi^*) = 0,0000007$, то подчеркнутые цифры являются верными.

Если все значащие цифры верные, то говорят, что число записано ***со всеми верными цифрами***. При $\xi^* = 0,03045$ и $\Delta(\xi^*) = 0,000003$ говорят, что число ξ^* записано со всеми верными цифрами.

Саму абсолютную $\Delta(\xi^*)$ или относительную погрешность $\delta(\xi^*)$, как правило, записывают в виде числа, содержащего одну или две значащие цифры.

Иногда употребляется термин – ***число верных цифр после запятой (точки)***.

При этом подсчитывается количество цифр от первой цифры числа после десятичной точки и до его последней верной цифры (в последнем рассмотренном примере это число равно **5**).

Очевидно, что абсолютная погрешность приближенного числа вполне характеризуется числом верных цифр после запятой, а относительная погрешность – числом верных значащих цифр.

Довольно часто информация о некоторой величине задается пределами ее измерения: $\xi_1 \le \xi \le \xi_2$, например, $1{,}119 \le \xi \le 1{,}127$.

Принято записывать эти пределы с одинаковым числом знаков после запятой и количеством значащих цифр у разности $\xi_2 - \xi_1$ не более трех.

Информацию о том, что число ξ^* является приближенным значением числа ξ с абсолютной погрешностью $\Delta(\xi^*)$, иногда удобно записывать в виде $\xi = \xi^* \pm \Delta(\xi^*)$, где числа ξ^* и $\Delta(\xi^*)$ принято записывать с одинаковым числом знаков после запятой.

Например, записи $\xi = 1{,}123 \pm 0{,}004$ и $\xi = 1{,}123 \pm 4 \cdot 10^{-3}$ относятся к общепринятым представлениям диапазона точности измерений и означают, что $1{,}123 - 0{,}004 \le \xi \le 1{,}123 + 0{,}004$.

Соответственно, информацию, что ξ^* является приближенным значением числа ξ с относительной погрешностью $\delta(\xi^*)$, записывают в виде $\xi = \xi^*(1 \pm \delta(\xi^*))$.

Например, следующие записи: $\xi = 1{,}123 \cdot (1 \pm 0{,}003)$, $\xi = 1{,}123 \cdot (1 \pm 3 \cdot 10^{-3})$ и $\xi = 1{,}123 \cdot (1 \pm 0{,}3\%)$ эквивалентны представлению в виде неравенства $(1 - 0{,}003) \cdot 1{,}123 \le \xi \le (1 + 0{,}003) \cdot 1{,}123$.

§ 1.2. «Грамотное» программирование вычислений.

Ошибки округления, которые сами по себе кажутся незначительными, могут оказать существенное влияние на конечный результат, если в процессе его достижения выполняется большое количество арифметических операций.

Поэтому желательно алгоритмическим способом минимизировать ошибки в каждой операции или в последовательности операций, уменьшая тем самым их распространение и воздействие на конечный результат.

Рассмотрим, как можно это сделать в случае выполнения, казалось бы, простой операции вычисления среднего арифметического c для двух чисел a и b.

Эта операция лежит в основе итераций по методу половинного деления (дихотомии) при вычислении корня нелинейного уравнения на отрезке $[a, b]$.

Возможны два способа выполнения этой арифметической операции:

$$\text{(а)}\ \ c = \frac{a+b}{2}, \qquad \text{(б)}\ \ c = a + \frac{b-a}{2}. \tag{1.3}$$

Очевидно, что формула (а) требует на одну операцию сложения меньше, чем формула (б). Но с точки зрения точности вычислений это не всегда лучше.

Например, пусть вычисления выполняются в десятичной арифметике с тремя цифрами мантиссы числа и с округлением для значений $a = 0.596$ и $b = 0.600$.

Тогда, по формуле (а) имеем

$$c = \frac{0.596 + 0.600}{2} = \frac{1.19\widehat{6}}{2} = \frac{1.20}{2} = 0.600,$$

хотя правильное значение c равно 0.598.

Если же проводить вычисления по формуле (б), то

$$c = 0.596 + \frac{0.600 - 0.596}{2} = 0.596 + \frac{0.004}{2} = 0.598.$$

Отметим, что в том примере, для которого формула (б) оказалась предпочтительнее, числа a и b имеют одинаковые знаки.

Рассмотрим другой пример для десятичной четырехзначной арифметики, где вместо операции округления применяется отбрасывание лишних разрядов.

Пусть $a = -3.483$ и $b = 8.765$.

Тогда по формуле (а) имеем

$$c = \frac{-3.483 + 8.765}{2} = \frac{5.282}{2} = 2.641,$$

что представляет собой точный результат.

Вычисления же по формуле (б) дают

$$c = -3.483 + \frac{8.765 + 3.483}{2} = -3.483 + \frac{12.24}{2} = -3.483 + 6.120 = 2.637.$$

Даже если здесь выполнить округление, то все равно результат, полученный по формуле (б), отличался бы от точного, поскольку значение c равнялось бы 2.642.

Следовательно, в этом примере, где числа a и b имеют разные знаки, формула (1.3.а) оказалась предпочтительнее (1.3.б).

Отсюда можно сделать вывод, что для достижения наивысшей точности надо применять формулы (а) и (б) в зависимости от знаков a и b.

Поэтому наилучшую «формулу» для вычисления среднего арифметического a и b можно представить следующим образом:

$$\text{если } a \cdot b \le 0, \text{ то } c = \frac{a+b}{2}, \text{ иначе } c = a + \frac{b-a}{2}. \qquad (1.4)$$

Рассмотрим еще один показательный пример – определение корней квадратного уравнения $ax^2 + bx + c = 0, \quad a \neq 0$.

Корни этого уравнения определяются известными формулами:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}. \tag{1.5}$$

Предположим, что вычисления выполняются в десятичной арифметике с четырьмя цифрами мантиссы, с одной запасной цифрой и отбрасыванием младших разрядов.

Пусть $a = 1, \quad b = -320, \quad c = 16$.

Для изображения действительных чисел удобно использовать обычный формат чисел с плавающей точкой $\pm.d_1d_2d_3d_4\mathrm{E}n$, где d_i – значащие цифры числа, а n – его десятичный порядок, т.е. $a = +.1000\mathrm{E}1, \quad b = -.3200\mathrm{E}3, \quad c = +.1600\mathrm{E}2$.

Вычисления по формулам (1.5) дадут следующие значения:

$$x_1 = \frac{.3200\mathrm{E}3 + \sqrt{.1024\mathrm{E}6 - .6400\mathrm{E}2}}{.2000\mathrm{E}1} =$$

$$= \frac{.3200\mathrm{E}3 + .319\overline{8}\mathrm{E}3}{.2000\mathrm{E}1} = \frac{.639\overline{8}\mathrm{E}3}{.2000\mathrm{E}1} = .319\overline{9}\mathrm{E}3 = 319.\overline{9},$$

$$x_2 = \frac{.3200\mathrm{E}3 - .319\overline{8}\mathrm{E}3}{.2000\mathrm{E}1} = \frac{.\overline{2}000\mathrm{E}0}{.2000\mathrm{E}1} = .\overline{1}000\mathrm{E}1 = 0.\overline{1} \quad .$$

Первые неверные цифры, полученные вследствие ошибок округления, выделены чертой сверху.

Правильные значения корней x_1 и x_2 с шестью значащими цифрами равны соответственно

$$x_1 = 319.950 \text{ и } x_2 = 0.0500078.$$

Следовательно, непосредственные вычисления по формулам дали хороший результат для x_1 и плохой для x_2, поскольку относительная ошибка в x_2 оказалась равной 100%.

Легко понять, почему это произошло.

В числителе дроби для x_2 вычитаются два близких по значению числа 320.0 и $319.\overline{8}$. Поэтому при вычитании старшие значащие цифры взаимно уничтожаются, а значение результата определяют последние цифры мантисс этих чисел.

Однако последняя цифра числа $319.\overline{8}$ уже содержит ошибку округления после операций вычитания и взятия квадратного корня.

Вследствие этого результат последней операции вычитания $320.0 - 319.\overline{8}$ бесполезен, поскольку уже его ***первая значащая цифра*** содержит ошибку.

Данная ситуация носит название ***катастрофической потери значащих цифр***.

Она имеет место в тех случаях, когда вычитаются числа, имеющие одинаковые знаки и приблизительно равные по абсолютной величине, что приводит к увеличению общего уровня ошибок в вычислениях.

Можно избежать катастрофической потери значащих цифр, если правильно организовать вычисления.

В случае квадратного уравнения вместо непосредственных вычислений по формулам (1.5) следует применить следующие формулы:

$$x_1 = \frac{-b - sign(b)\cdot\sqrt{b^2 - 4ac}}{2a}, \quad x_2 = \frac{c}{ax_1}. \tag{1.6}$$

В нашем примере имеем по формулам (1.6):

$$x_2 = \frac{.1600\text{E}2}{.3199\overline{9}\text{E}3} = .5002\overline{}(\text{E}-1) = 0.05002\overline{}.$$

Организовав «грамотно» вычисления, мы избежали вычитания чисел с одинаковыми знаками, и это позволило получить лучший результат.

Пусть требуется построить алгоритм вычисления значений экспоненциальной функции $f(x) = e^x$ при любом заданном значении аргумента x.

Предположим, что с этой целью мы решили воспользоваться известным разложением функции e^x в степенной ряд

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{i=0}^{\infty} e_i(x), \quad e_i(x) = \frac{x^i}{i!}, \tag{1.7}$$

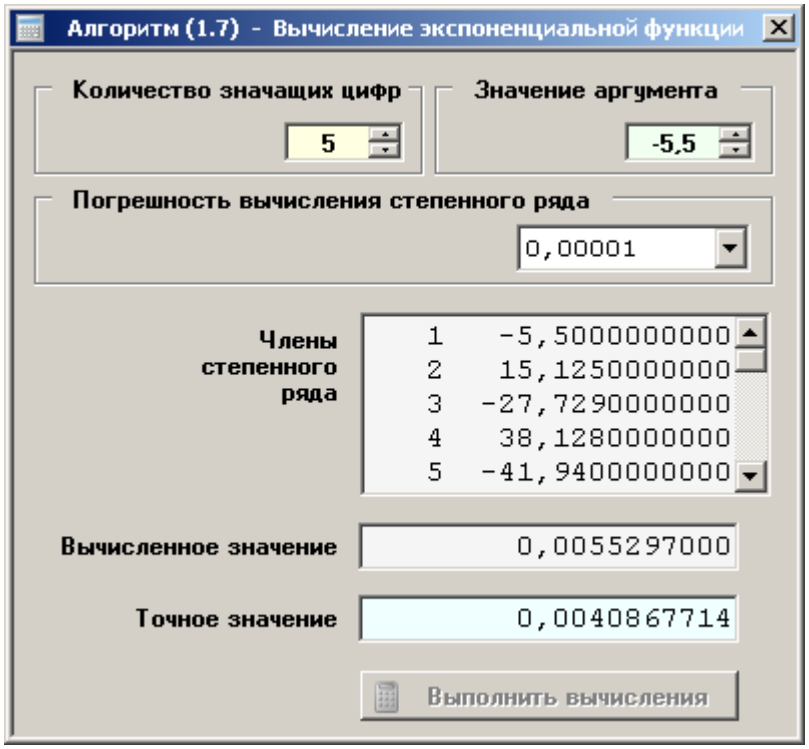
который сходится для любых вещественных x.

Естественно, что в реальных расчетах требуется «обрезать» ряд (1.7) на каком-либо его члене $e_i(x)$, и интерпретировать сумму полученного конечного ряда как сумму бесконечного степенного ряда (1.7). Очевидно, что необходимое количество членов ряда будет зависеть от величины x и требуемой точности ε. Накопление суммы ряда (1.7) можно прекращать, например, при выполнении условия $|e_i(x)| < \varepsilon$.

Пусть мы располагаем «процессором», выполняющим арифметические операции с действительными числами, мантисса которых имеет только пять значащих цифр.

Нам требуется вычислить $e^{-5.5}$ в такой вычислительной системе с абсолютной погрешностью $\varepsilon \sim 10^{-5}$, т.е. суммирование ряда прекращается, когда $|e_i(x)| < 10^{-5}$.

Алгоритм реализован на языке **C#** с эмуляцией округления результата до пятой значащей цифры. Результат работы компьютера приводится ниже:



Результат выполненных вычислений – **+0.00552970.**

При этом ряд (1.7) был «обрезан» после **23**–го члена, равного **-0.0000041291**, поскольку для него было выполнено условие $|e_i(x)| < 10^{-5}$.

Однако точный результат – $e^{-5.5} =$**+0.0040867714**.

Очевидно, что мы имеем катастрофическую потерю значащих цифр.

Существо дела видно из анализа значений первых членов ряда $e_i(x)$: младшая значащая цифра каждого из членов, больших **10** по абсолютной величине, влияет на старшую значащую цифру итогового результата.

А поскольку все члены ряда вычислялись приближенно, то это и обеспечило итоговую ошибку окончательного результата.

Чтобы избежать негативных последствий данной ситуации, вычисление числа $e^{-5.5}$ можно организовать при помощи предварительного определения величины $e^{5.5}$ с последующим вычислением его обратного значения:

$$e^{-5.5} = \frac{1}{e^{5.5}} = \frac{1}{1 + 5.5 + 15.125 + \ldots} \approx 0.0040867714.$$

Данный прием носит чисто иллюстративный характер. Скорость сходимости ряда (1.7) очень мала, вследствие чего необходимо брать достаточно большое количество его членов для достижения приемлемой точности конечного результата.

Заметим, что в современных языках программирования для вычисления e^x и других математических функций используются гораздо более эффективные алгоритмы.

Предыдущие примеры показывают, что численный метод может быть очень чувствителен к небольшим изменениям (возмущениям) в исходных данных задачи.

Оказывается, что существуют задачи, которые сами по себе чувствительны к возмущениям такого рода.

Классическим примером является вычисление корней полинома

$$p(x) = (x-1)(x-2)\dots(x-20) = x^{20} - 210x^{19} + \dots \qquad (1.8)$$

с хорошо отделенными корнями $1, 2, 3, \dots, 20$.

Предположим, что коэффициент при x^{19} слегка возмущен и равен $210 + 10^{-7}$.

Новый полином незначительно отличается от полинома (1.8), но его корни уже существенно отличаются от корней полинома $p(x)$.

Приведем все их значения с пятью знаками после десятичной точки:

1.00000	6.00001	10.09527 ± 0.64350i
2.00000	6.99970	11.79363 ± 1.65232i
3.00000	8.00727	13.99236 ± 2.51883i
4.00000	8.91725	16.73074 ± 2.81262i
5.00000	20.84691	19.50244 ± 1.94033i

Очевидно, что небольшое возмущение данного полинома $p(x)$ привело к значительному изменению корней, некоторые из которых стали даже комплексными, чего не было для полинома $ax^2 + bx + c$ второй степени.

Подчеркнем существенное различие между этими двумя полиномами.

Метод вычисления корней квадратного уравнения (1.5) оказался весьма чувствительным к небольшим возмущениям в коэффициентах, и требовалось только перейти к альтернативному методу (1.6), когда это было необходимо.

В случае же полинома $p(x)$ чувствительна ***сама задача***.

Поэтому независимо от метода поиска корней, операции округления (всегда присутствующие в процессорных вычислениях) обязательно внесут возмущения, которые непременно приведут к значительным искажениям итогового результата.

Когда задача чувствительна, то вряд ли можно сделать что-нибудь другое, кроме попытки идентифицировать ее чувствительность и пересмотреть решаемую проблему в целом, чтобы избежать решения чувствительных задач.

Говорят, что полином ***плохо обусловлен***, если небольшие изменения в его коэффициентах приводят к большим изменениям в его корнях.

Полином $p(x)$ как раз служит примером плохо обусловленных полиномов.

Однако, как это видно из таблицы выше, не все корни многочлена $p(x)$ изменились в равной степени.

Первые шесть корней многочлена $p(x)$ оказались не так чувствительны к возмущениям в коэффициентах как остальные.

В таком случае следует делать вывод об обусловленности каждого корня в отдельности, а не всего полинома $p(x)$ в целом.

Другим примером чувствительной задачи служит система линейных алгебраических уравнений. Степень чувствительности здесь определяется величиной определителя для заданной системы уравнений.

Чем ближе значение определителя к нулю, тем хуже обусловлена система уравнений и выше чувствительность алгоритмов к ошибкам округления.

В завершении краткого обзора проблем, возникающих в результате ошибок округления при проведении приближенных вычислений на ЭВМ, рассмотрим случай ***неустойчивого алгоритма***.

Пусть при решении некоторой задачи (с использованием компьютера) нам предстоит многократно вычислять интегралы следующего вида

$$I_n = \int_0^1 x^n e^{x-1} dx, \quad n = 1, 2, \ldots \tag{1.9}$$

Для интегралов (1.9) известно следующее рекуррентное соотношение:

$$I_n = x^n e^{x-1}\Big|_0^1 - n \cdot \int_0^1 x^{n-1} e^{x-1} dx = 1 - nI_{n-1}, \; n = 2, 3, \ldots, \quad I_1 = \frac{1}{e}. \tag{1.10}$$

Если организовать счет по рекуррентному соотношению (1.10) с использованием чисел «двойной точности», то для $n = 18$ получим $I_{18} \approx -0.02945...$

Единственная ошибка округления в последней, 16-й цифре мантиссы, допускаемая при вычислениях I_n, приводит к тому, что значение I_{18} оказывается отрицательным, хотя на интервале $(0, 1)$ функция $x^{18} e^{x-1}$ положительна и $I_{18} \approx 0.05012...$

Других ошибок в процессе вычислений по рекуррентному соотношению (1.10) внесено не было.

Причиной такого катастрофического накопления ошибок является то, что первоначальная ошибка округления, допущенная при вычислении значения $I_1 = e^{-1}$, при вычислении итогового значения I_n умножается, в конечном итоге, на значение факториала $n!$.

Таким образом, циклический алгоритм (1.10) оказался неустойчивым, поскольку на каждом последующем его шаге первоначальная ошибка округления «умножается» на текущее значение переменной этого цикла.

Если же переписать алгоритм (1.10) в неявном виде

$$I_n = \frac{1 - I_{n+1}}{n+1}, \quad n = \ldots, 3, 2, 1, \tag{1.11}$$

то легко видеть, что на n-м шаге ошибка не только не увеличивается, а даже уменьшается в $n+1$ раз, т.е. данный алгоритм становится уже устойчивым.

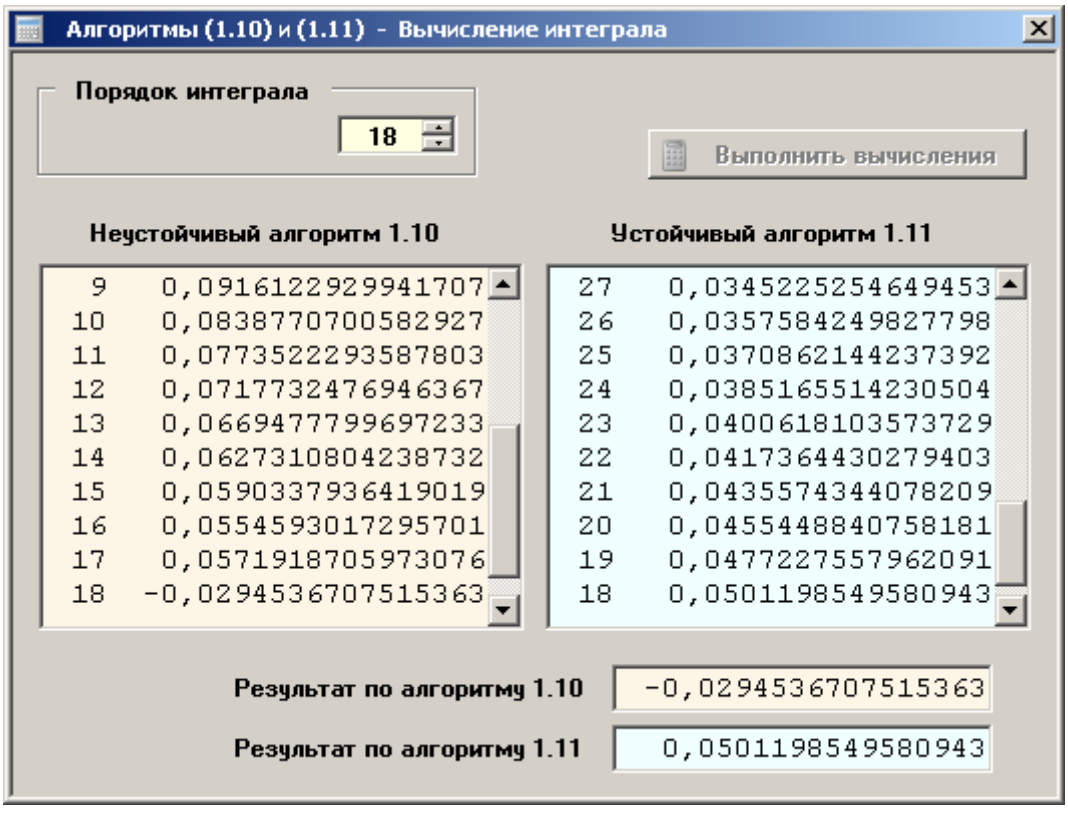
Очевидный его недостаток состоит в том, что нам не известно начальное приближение. Однако, как бы грубо мы ни выбрали начальное приближение при $n >> 1$, начальная и промежуточные ошибки округлений будут быстро уменьшаться на каждом последующем вычислительном шаге.

Приемлемая аналитическая оценка для выбираемого начального приближения может быть получена следующим образом:

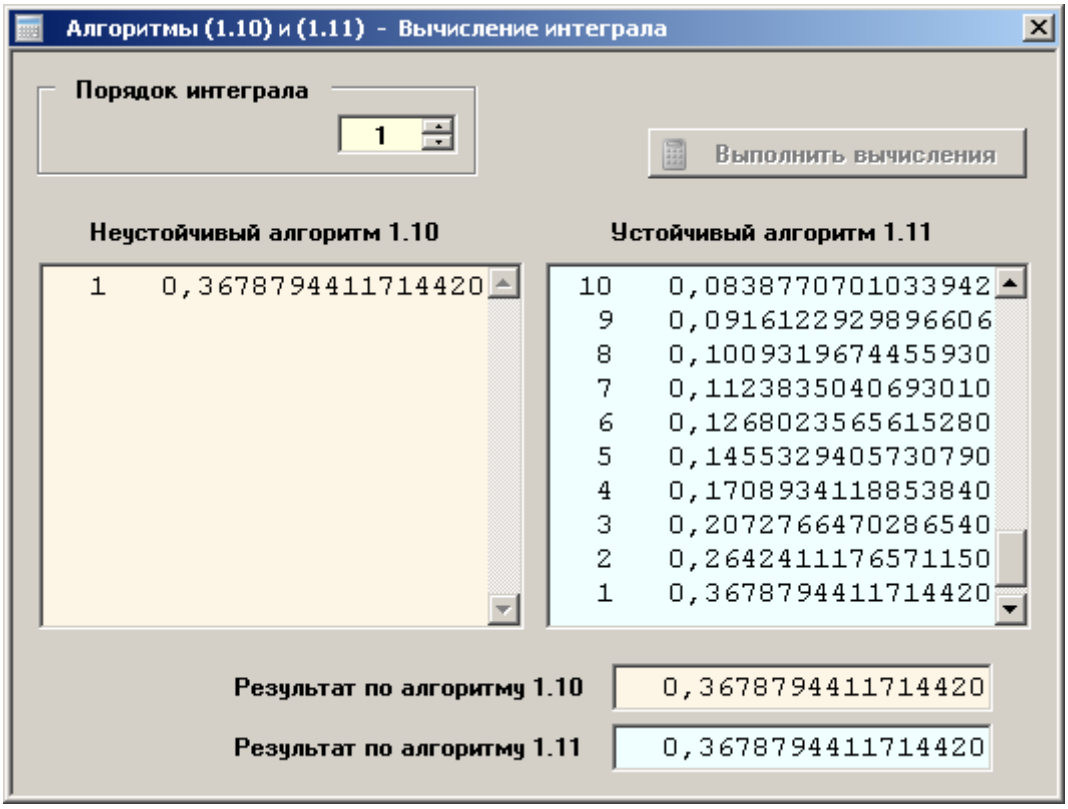
$$I_n = \int_0^1 x^n e^{x-1} dx \le \int_0^1 x^n dx = \left. \frac{x^{n+1}}{n+1} \right|_0^1 = \frac{1}{n+1}. \tag{1.12}$$

Если положить, например, даже $I_{50} \approx 0$, то начальная ошибка не будет превосходить $1/51$. При вычислении I_{49} она уже умножится на $1/50$ и станет равной примерно 0.0004. Когда же мы дойдем до I_{18}, начальная ошибка станет меньше 10^{-16}, т.е. меньше единственной ошибки округления для чисел с «двойной» точностью и окажется совершенно подавленной в силу устойчивости алгоритма (1.11).

Сказанное выше представлено на форме демонстрационного приложения:



Для тестирования алгоритма (1.11) можно вычислить значение $I_1 = e^{-1}$ и сравнить его со значением, вычисляемым по алгоритму (1.10). Они должны быть идентичны.



Вычисления при $n = 10$ позволяют сравнить конечные результаты для обоих алгоритмов (1.10) и (1.11). Мы видим насколько увеличилась ошибка в первой таблице для данного значения индекса n, если за «эталонное» значение выбирать соответствующую величину из второй таблицы.