2023.02.05

2024.02.08

Требования и рекомендации по заданию 1 (1 курс, 2 семестр)

Первое задание — простейшие вычислительны алгоритмы. Не вникаем глубоко в теорию вычислительных алгоритмов, но понимаем, что вычисления на конкретном процессоре выполняются с некоторой (относительной) погрешностью. Влияние этой погрешности на окончательный результат зависит как от самого математического вычислительного алгоритмя, так и от способа реализации этого алгоритма. То есть, одни и те же вычисления можно провести по разным математически эквивалентным формулам, но результат может иметь разную окончательную погрешность.

При программировании надо иметь в виду, что конкретная задача не всегда может быть решена конкретным математическим вычислительным алгоритмом. Следовательно, программная реализация должна как-то проверять входные данные на корректность и контролировать процесс вычислений с тем, чтобы обнаружить ситуации, когда корректное решение не может быть получено. В таких случаях программа должна выдавать некоторую диагностику и комментарии по поводу возникших ситуаций. Как говорят, программа может отказаться решать задачу, но она не имеет права молчаливо дать неверный ответ (вводя пользователя в заблуждение) либо “упасть”, аварийно прерывая процесс вычислений.

Далее замечания и рекомендации по решению отдельных задач 1 задания.

Описание математических вычислительных алгоритмов

В данной версии документа отсутствуют

Комментарии к задаче 1 — решение нелинейного уравнения.

Основная вычислительная процедура нахождения корня должна быть оформлена в виде функции, получающей на вход указатель на функцию, задающую искомое уравнение, требуемую точность корня, другие параметры, определяющие конкретную задачу и конкретный алгоритм.

На выходе процедура должна выдавать (через возвращаемое значение или параметры-указатели) значение найденного корня и условный код успешности завершения. Основные ситуации описываемые этим кодом завершения — это: корень успешно найден с требуемой точностью, корень найден, но точность не гарантируется, корень не может быть найден, в процессе поиска корня произошла некоторая ошибка (какая?). При анализе реализации преимущества имеют версии, в которых “экономится” количество обращений к функции, определяющей уравнение. Поэтому при разработке программы старайтесь по возможности не вычислять заново уже подсчитанные значения, а использовать их повторно в процессе вычислений.

При тестировании программы нужно проверить работоспособность процедуры для разных уравнений, разных ситуаций с дополнительными данными (границами отрезка поиска решений, точностью и т.п.). Нужно проверить варианты, когда корни есть, когда корней нет, когда точность задана некорректно (например равна 0 или -1), когда существует корень для неотрицательной функции на отрезке и т.п. Для проверки можно использовать уравнения с известными значениями корней, печатая рядом точный ответ и ответ, полученный по вашей программе. Интересно проверить как ваша функция будет реагировать на параметры существеннно разного масштаба, например, найти корни уравнения $e^x - a = 0$ при $a = 1, 10^{-2}, 10^{-10}, 10^{-20}$....

Комментарии к задаче 2 — вычисление определенного интеграла.

Постановка задачи — вычислить определенный интеграл от заданной функции на заданном отрезке. Вычислительная процедура должна быть оформлена в виде функции, получающей на вход указатель на подинтегральную функцию, отрезок интегрирования, требуемую точность вычисления, другие параметры, определяющие конкретную задачу и конкретный алгоритм.

Таким образом, процедура должна сама подбирать необходимый шаг разбиения для составной квадратурной формулы и проводить итерационное уточнение вычисленного интеграла методом удвоения числа узлов разбиения до достижения требуемой точности.

Для тестов следует взять несколько функций, которые можно проинтегрировать в явном виде (по формуле Ньютона–Лейбница). Тогда легко проверить насколько точно будет ваша реализация работать на разных отрезках интегрирования. То есть мы можем сравнить результат вычислений вашей процедуры и результат, полученный по формуле Ньютона–Лейбница. Недостаточно проверять только на многочленах, надо также пытаться проинтегрировать другие функции (экспоненциальные, тригонометрические и т.д.). Интересно проверить процедуру на вычислении интегралов от функции типа $\sin kx$ при разных значениях k (в том числе достаточно больших $k \sim 10000$).

Метод удвоения разбиения может в некоторых случаях (при слишком малой требуемой погрешности) провоцировать разбиение отрезка интегрирования на очень большое число узлов. В таким случаях сама вычислительная прогрешность суммирования множества слагаемых может стать заметной и исказить финальный результат (или "зациклить" вычисления). В процедуре следует отслеживать такую ситуацию. Например, можно следить за монотонным убыванием контрольной разности $|I_n - I_{2n}|$. Если монотонность убывания такой разности нарушается, то это признак того, что погрешность суммирование множества слагаемых интегральной суммы стала сравнимой с погрешностью используемой квадратурной формулы с данным количеством узлов. То есть, дальнейшее увеличение количества узлов разбиения уже не даст эффекта уменьшения погрешности результата из-за значительной ошибки суммирования. Это обстоятельство показывает возможность разных исходов: вычисление интеграла с тре,уемой точностью и вычисление без гарантии точности. Вычислительная процедура должна об этом сообщать через возвращаемый код успешности вычислений.

Здесь также важно минимально количество обращений к подинтегральной функции. Поэтому при удвоении числа разбиений нужно по возможности использовать значения функции, вычисленные на предыдущем разбиении.

Комментарии к задаче 3 — минимизация функции.

Постановка задачи — найти аргумент x, при котором функция $f(x)$ принимает свое минимальное значение. Вычислительная процедура должна быть оформлена в виде функции, получающей на вход указатель на исследуемую функцию, требуемую точность вычисления точки минимума, другие параметры, определяющие конкретную задачу и конкретный алгоритм.

Следует определиться как задавать область поиска точки минимума, т.е. задавать ее отрезком или пытаться найти минимум на всей числовой прямой. В этом последнем случае нужно понять как поступать, если не получается определить точку минимума за достаточно большое число попыток.

Процедура должна на выходе давать значение точки минимума, значение функции в найденой точке минимума и код успешности поиска.

Следует заметить, что представление вещественных чисел в ЭВМ имеет относительную погрешность, и поэтому при поиске минимума реально вычисленная функция будет иметь не единственный минимум, а "плато" — постоянный участок, на котором вычисление функции дает одинаковые значения несмотря на разные аргументы. Примером может служить функция $1 + (x-1)^4$. Эта функция имеет единственную точку минимума $x = 1$. Однако при вычислении этой функции для любого x из интервала $(1 - 10^{-5}, 1 + 10^{-5})$ мы всегда получим значение 1. Таким образом, мы не можем определить точку минимума такой функции точнее, чем $O(10^{-5})$. Аккуратный алгоритм минимизации может попытаться найти левую и правую границы такого "плато", взять в качестве ответа его середину, и выдать диагностику о фактической длине этого плато как характеристику ограничений на поиск погрешностей.

Комментарии к задаче 4 — вычисление элементарных функций.

Решение должно быть оформленно в виде функций, получающих на вход аргумент x и

требуемую точность вычисления ε.

Суммирование рядов Тейлора дает адекватный результат, если аргумент x не сильно превосходит единицу (а лучше, если он по модулю меньше единицы). В таком случае суммирование можно прекращать, если последний добавленный член ряда стал по модулю меньше ε. Поэтому первым делом надо преобразовать аргумент к наименьшему подходящему значению. Для тригонометрических функций можно использовать периодичность и формулы приведения (есть хорошая функция fmod), это позволяет привести аргумент в диапазон $[0, \pi/2]$. Для экспоненты можно воспользоваться разложением аргумента на целую и дробную часть (функция modf). Степень с дробной частью считать суммированием ряда, а степень с целой частью считать быстрым возведением числа (константы e) в целую степень. Для отрицательных аргументов полезно вспомнить про свойство $e^{-x} = 1/e^x$. Для логарифма можно привести аргумент к нужном диапазону делением или умножением на e, а также воспользоваться заменой $x = \frac{1-y}{1+y}$. Далее ряды для $\ln(1-y)$ и $\ln(1+y)$ выписываются по формулам Тейлора.

Для проверки надо сравнить ваш ответ с результатами вычислений по библиотечным функциям. Следует проверить как работает ваша процедура для разных больших и маленьких, положительных и отрицательных аргументов, чтобы оказались задействованы формулы приведения аргумента к необходимому диапазону.

Комментарии к задаче 5 — интеполяция.

В задаче требуется "пересчитать" значения функции, заданой таблично (аргументы x_i, значения y_i), на другую таблицу, определяемую массивом новых аргументов $\bar{x}_i$ и массивом $\bar{y}_i$ для сохранения новых переинтерполированных значений. Решение должно быть оформлено в виде функции, получающей на вход данные массивы аргументов и значений. Исходную таблицу аргументов и значений удобнее задавать в файле, т.е. программа сначала должна прочитать этот файл и создать массивы аргументов и значений.

Следует принять решение как поступать, если аргумент x выходит за границы исходного диапазона табличных аргументов $[x_{i\,\min}, x_{i\,\max}]$. Наверное, наиболее правильное решение — выдавать отказ, но можно принять и другие решения (заполнять некоторым фиксированным значением, крайним значений с исходной сетки и т.п.). Для простоты можно считать, что массивы аргументов x_i и $\bar{x}_i$ уже упорядочены по возрастению, чтобы не заниматься сортировкой внутри интерполирующих процедур.

Для тестов можно брать конкретную функцию, вычислять ее на заданном множестве аргументов x_i, и потом применять интерполяцию, сопоставляя результат с точными значениями исходной функции в данных точках, а также рисуя графики этих функций, например, с помощью gnuplot.

Для интеполяции многочленом Лагранжа интересен тест приближения многочлена k-й степени многочленом степени $n > k$. То есть надо вычислить значения многочлена k-й степени в $n+1$ точке, построить многочлен Лагранжа по этим значениям и потом сравнить значения этих двух многочленов в разнообразных точках.

В задаче среднеквадратического приближения бóльший интерес представляет не столько значение функции, сколько коэффициенты уравнения аппроксимирующей прямой $y = ax + b$. Поэтому вычисляющая функция может выдавать именно эти коэффициенты, и потом их можно использовать для вычисления каких-то значений, если потребуется.

Для наглядности надо уметь рисовать графики исходной и приближенной проинтерполированной функции, например, с помощью программы Gnuplot. Программа gnuplot позволяет легко рисовать ломанные линии, заданные в файле координатами (x, y). Пусть мы имеем файл f1.txt, в котором на каждой строке записаны некоторые значения (x, y). Тогда для рисования нужно выполнить в консоли следующие простые команды.

```
> gnuplot                          [откроется консоль gnuplot]
> plot 'f1.txt' with lines         [будет нарисован график (ломаная) точек из f1.txt]
> quit                             [выход из консоли gnuplot]
```

Можно нарисовать одновременно графики (ломаные) из двух файлов

```
> gnuplot
> plot 'f1.txt' with lines, 'f2.txt' with lines
> quit
```

Если в файл с координатами вставить в некотором месте строку, содержащую только несколько пробелов (не пустую строку, а именно строку с пробелами!), то при рисовании в этом месте у ломаной линии появится разрыв. Т.е. таким образом можно отрисывывать несколько отдельных ломаных линий через данные в одном файле.

Итак, достаточно подготовить два файла с таблицами аргументов-значений исходной и проинтерподированной функции и потом увидеть как эти функции соотносятся друг с другом.