Санкт-Петербургский Государственный Университет

Вычислительный практикум

Задача обратного интерполирования

Борис Мельников
b.melnikov17@gmail.com

под руководством Алцыбеева Глеба Олеговича
Преподаватель & Ассистент

май, 2023

Содержание

1 Введение

Итак, несколько ранее мы установили, что студенту математического факультета бывает необходимо находить значение сложной функции в заданной точке приближенно. Однако случается, что значения функции в точках уже известны, и куда более важным вопросом уже становится нахождение точки, в которой заданная функция принимает то или иное значение. По аналогии с задачей интерполирования данная проблема носит название "задача обратного интерполирования". Установлено, что у нее есть несколько способов решения, в связи с чем рассмотрим 2 самых популярных.

Целью работы ставилось знакомство с данной задачей и вариантами ее решения, наиболее подходящими для проведения вычислений на компьютере. В ходе работы были сделаны попытки реализовать удобный для использования интерфейс между пользователем и компьютером через встроенное в Windows 11 решение PowerShell.

2 Постановка задачи

Пусть на промежутке $[a,b]$ задана таблица значений вещественной функции $y = f(x)$:

x	$f(x)$
x_0	$f(x_0)$
x_1	$f(x_1)$
$\dots$	$\dots$
x_n	$f(x_n)$

узлы которой предполагаются попарно различными, т.е. $x_i \neq x_j, i \neq j$.

Требуется приближенно найти такое $\bar{x}$, что $f(\bar{x}) \approx \bar{y}$, где $\bar{y}$ известно.

3 Необходимая теория

3.1 Подготовительный этап

Конечно, итогом работы должна стать программа, которая будет находить требуемое значение несколькими методами, которые далее будут описаны. Но для начала нужно сформировать входные данные. Поступим так: поскольку вычислительные мощности современных машин позволяют находить зачения сложных функций с небольшой погрешностью, будем напрямую в ходе исполнения программы составлять таблицу из условия. Создав двумерный вектор `vector <vector <double>> table`, заполним его $m+1$ значением точек из отрезка $[a,b]$, где шаг между самими точками укажем равным $h = (b-a)/m$. В данном случае значения m, a, b становятся переменными, значения которых задаются пользователем в ходе исполнения программы, а 1-ая точка (она же x_0) в массиве будет совпадать с a. Значения функции $f(x) = \cos(x) + 2 \cdot x$ (соответствует моему варианту #10), соответствующие точкам

x_i в объявленном массиве, заполним значениями, которые будут автоматически вычислены при помощи функции `double f(double x)` (в теле функции просто задано выражение `cos(x) + 2 * x`).

Теперь, когда имеется подготовленный массив данных с узлами интерполирования, можем переходить непосредственно к нахождению приближенного значения точки $\bar{x}$, в которой $f(\bar{x}) \approx F$ (значение F будет храниться в переменной `double F`).

3.2 Метод #1

Предостережение: метод применим, если предполагается, что функция $f(x)$ на отрезке $[a,b]$ всюду строго монотонна.

Итак, метод основан на следующей теореме из математического анализа:

Теорема Отображение $f : X \longrightarrow Y$ обратимо (т. е. существует обратное отображение f^{-1}) $\Longleftrightarrow$ f – биективно.

Понятно, что строго монотонная на отрезке функция является на нем биективной. Но тогда исходную задачу можно свести к задаче интерполирования – будем строить интерполяционный многочлен для функции f^{-1}. Делается это так: поскольку у нас есть таблица точек из отрезка $[a,b]$ и соответствующих им значений функции f, можем поменять столбцы местами и получим уже таблицу значений функции f и соответсвующих им точек, где эти значения достигаются (т. е. из таблицы для f получим таблицу для f^{-1}). Далее методом Лагранжа или Ньютона можно построить многочлен, интерполирующий f^{-1}, и найти приближенно искомое значение $\bar{x}$ как $f^{-1}(F)$.

3.3 Метод #2

Теперь поговорим о 2-ом методе. Применяя его, уже нет необходимости требовать строгой монотонности функции $f(x)$ на заданном отрезке $[a,b]$. Теперь мы не будем ставить вопрос о существовании обратной функции, а сразу построим интерполирующий функцию $f(x)$ многочлен $P_n(x)$ одним из определенных ранее способов. Тогда выражение $P_n(x) = F$ задает уравнение, корни которого можно найти одним из способов, которые были описаны в работе #1.

3.4 Преимущества и недостатки методов

#1: Конечно, требование строгой монотонности функции $f(x)$ на отрезке $[a,b]$ сильно сужает класс функций, для которых данным методом можно найти $\bar{x}$ по заданному F. Однако если функция $f(x)$ все же удовлетворяет требуемому условию, то у нас появляется возможность отыскать $\bar{x}$ вне отрезка $[a,b]$. Безусловно, мы констатировали, что поведение интерполирующего многочлена вне отрезка $[a,b]$ непредсказуемо, но в "безвыходной ситуации"подобным образом можно отыскать решение $\tilde{x}$, для которого мы точно будем знать значение невязки $|f(\tilde{x}) - F|$, которая в каких-то случаях может быть невелика. Говоря же о весомых преимуществах, отметим, что, в отличие от 2-го метода, в данном методе требуется лишь вычислить значение интерполирующего многочлена в точке без необходимости последующего нахождения корней уравнения, что может занимать время (конечно, все описанные в работе #1 методы сходятся, но в зависимости от сложности левой части уравнения время сходимости может быть велико).

#2: Основное преимущество данного метода, конечно, – отсутствие требования строгой монотонности функции $f(x)$. Также, в случаях, когда решения, полученные методом #1, не удовлетворяют требуемой степени точности, корни с большей точностью можно найти данным методом (объясняется в [1]). Но, как уже было замечено выше, появляется необходимость производить дополнительный ряд вычислений при подсчете корней уравнения $P_n(x) = F$.

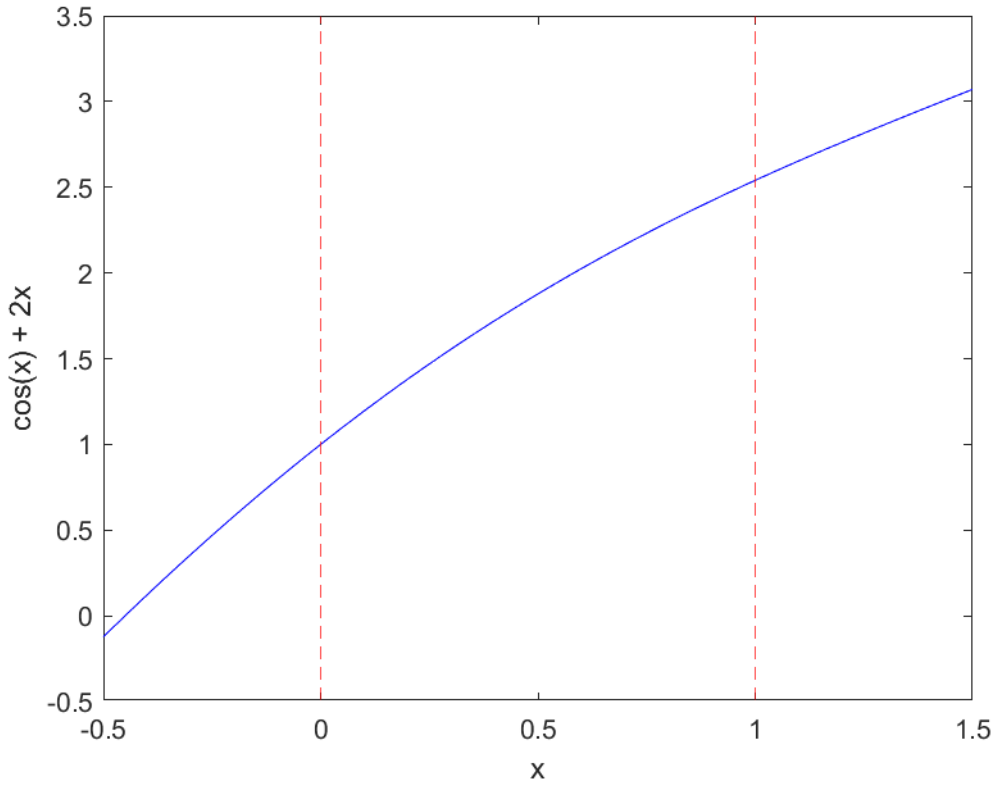

Рис. 1: График $f(x)$ на отрезке $[a,b]$

4 Практическая реализация

Итак, теперь опишем данные для задачи, которую предстоит решить. Будем решать задачу из варианта #10 со следующими начальными данными:
$f(x) = cos(x) + 2 \cdot x$;
$a = 0$;
$b = 1$;
$m = 10$;
$t = -8$;

Поскольку методы используют решения, результаты для которых были получены ранее (например, зависимость точности получаемого значений от наибольшей степени интерполирующего многочлена), не будем заново описывать данные закономерности. Кратко упомянем, что нахождение корней вне заданного отрезка $[a,b]$ приведет к получению непредсказуемых результатов в методе #1, и сообщению об ошибке (`Unable to find the value`) в методе #2. При этом значения $\bar{x}$, лежащие внутри отрезка, полученные методом #2, имеют большую точность, чем значения из алгоритма метода #1. Покажем:

Ex. #1:

```
@@ Interpolation point:
   x = 1.0001

   Degree of interpolation polynomial
   Hint: n <= 10
   n = 10

@@ Method #1
   Lagrange value: 5.000070402e-05
   |f(L_res) - F|: 1.5800183384e-10
```

```
@@ Method #2
   Measurement accuracy (10^t , t < 0)
   t = -8
   Found value: 5.0000625019e-05
   |f(F_found) - F|: 6.6613381478e-15
```

Ex. #2:

```
@@ Interpolation point:
   x = 1.67845

   Degree of interpolation polynomial
   Hint: n <= 10
   n = 10

@@ Method #1
   Lagrange value: 3.7374098654e-01
   |f(L_res) - F|: 5.6427129635e-10

@@ Method #2
   Measurement accuracy (10^t , t < 0)
   t = -8
   Found value: 3.7374098689e-01
   |f(F_found) - F|: 1.8207657604e-14
```

Ex. #3:

```
@@ Interpolation point:
   x = 2.164

   Degree of interpolation polynomial
   Hint: n <= 10
   n = 10

@@ Method #1
   Lagrange value: 6.9937892700e-01
   |f(L_res) - F|: 4.2092995756e-11

@@ Method #2
   Measurement accuracy (10^t , t < 0)
   t = -8
   Found value: 6.9937892703e-01
   |f(F_found) - F|: 4.8849813084e-15
```

Остается привести небольшую справку о коде программы:
1) Для упрощения запуска программы тела функций, прописанные в прошлых работах, продублированы в файле и этой программы (это функции `double Lagrange_r()`, `void secants (double A, double B, double eps)`;
2) Функция `double Lagrange(double x)` реализована через `double Lagrange_r()` путем повторного использования кода, однако их предназначение разное: `Lagrange(double x)` используется функцией `secants(double A, double B, double eps)` для вычисления приближенного значения интерполирующего функцию $f(x)$ многочлена, а функция `double Lagrange_r()` – приближенного значения интерполирующего функцию $f^{-1}(x)$ многочлена (в объяснении метода #1 говорилось, что столбцы в таблице узлов интерполирования нужно поменять местами, однако вместо этого в данной функции происходит захват значений сразу из второго столбца таблицы (элементы вида `table[x][1]`), а в функции `Lagrange(double x)` – первого

(table[x][0]));

Ссылка на репозиторий: https://github.com/BoriskaCat/-CM-Projects/tree/main/Lab%202

```
Output:
@@ #2. Inverse interpolation problem

@@ #ID: 10

@@ Parameters:
   f(x) = cos(x) + 2x
   m = 10
   a (> -1) = 0
   b (> -1) = 1

@@ Table of values:
     x      |       f(x)
   0.000    |   1.00000000000
   0.100    |   1.19500416528
   0.200    |   1.38006657784
   0.300    |   1.55533648913
   0.400    |   1.72106099400
   0.500    |   1.87758256189
   0.600    |   2.02533561491
   0.700    |   2.16484218728
   0.800    |   2.29670670935
   0.900    |   2.42160996827
   1.000    |   2.54030230587


@@ Interpolation point:
   x = 2

   Degree of interpolation polynomial
   Hint: n <= 10
   n = 10

@@ Method #1
   Lagrange value: 0.5824379434
   |f(L_res) - F|: 9.5750163354e-10

@@ Method #2
   Measurement accuracy (10^t , t < 0)
   t = -8
   Found value: 5.8243794277e-01
   |f(F_found) - F|: 1.1102230246e-14

?? Change the value of F, n, eps? [y / n]
n
```

Список литературы

[1] Лебедева А.В., Пакулина А.Н. Практикум по методам вычислений. Часть 1. СПб., СПбГУ. 2021. 156 - стр.