Университет ИТМО

Факультет программной инженерии и компьютерной техники

Вычислительная математика. Лабораторная работа №4. Аппроксимация функций методом наименьших квадратов

Группа: P32131
Студент: Смирнов Виктор Игоревич
Вариант: 16

2023

Ключевые слова

Интерполяция, Многочлен Лагранжа, Многочлен Ньютона.

1 Цель работы

Цель работы - решить задачу интерполяции, найти значения функции при заданных значениях аргумента, отличных от узловых точек, используя методы Лагранжа и Гаусса.

2 Вычислительная задача

2.1 Описание

Поставлена была ответственная задача: даны узлы интерполяции, даны точки, значения многочлена в которых нужно интерполировать. Для решения поставленной задачи воспользуемся библиотеками уже родными библиотеками Mathematica и Sematica, разработанными мною в процессе прохождения курса вычислительной математики в Университете ИТМО (исходный код: `https://github.com/vityaman-edu/math-tool`).

2.2 Программа для исследования

```cpp
#include "Mathematica/Abstract/Float.hpp"
#include "Mathematica/Collection/Array.hpp"
#include "Mathematica/Common/Point.hpp"
#include "Mathematica/Functional/Interpolation/Lagrange.hpp"
#include "Mathematica/Functional/Interpolation/Newton.hpp"
#include "Symatica/Expression/DSL/Literals.hpp"
#include "Symatica/Symbol/TreeTable.hpp"
#include "Symatica/Visitor/Evaluator.hpp"
#include <gtest/gtest.h>
#include <iomanip>

using Mathematica::Point;
using Mathematica::Collection::Array;
using Mathematica::Functional::Interpolation::dividedDifference;
using R = Mathematica::Abstract::Float<double>;

TEST(MathematicaFunctionalInterpolation, Playground) { // NOLINT
  constexpr auto N = 7;
  // clang-format off
  const auto points = Array<Point<R>, N>({{
    { 0.25, 1.2557 },
    { 0.30, 2.1764 },
    { 0.35, 3.1218 },
    { 0.40, 4.0482 },
    { 0.45, 5.9875 },
    { 0.50, 6.9195 },
    { 0.55, 7.8359 },
  }});
  const auto testX = Array<R, N + N>({{
    0.251, 0.512, 0.255, 0.534, 0.272,
    0.551, 0.294, 0.402, 0.372, 0.405,
    0.384, 0.445, 0.351, 0.437
  }});
  // clang-format on

  constexpr auto X = 1;
  const auto x = Symatica::Expression::DSL::var(X);
  auto table = Symatica::Symbol::TreeTable({});
  auto eval = Symatica::Visitor::Evaluator(table);

  const auto lagrange
      = Mathematica::Functional::Interpolation::symbolicLagrange(points, x);
  const auto newton
      = Mathematica::Functional::Interpolation::symbolicNewton(points, x);

  std::cout << "Interpolation::symbolicLagrange:" << std::endl;
```

```cpp
  std::cout << lagrange->asString() << std::endl << std::endl;

  std::cout << "Interpolation::symbolicNewton:" << std::endl;
  std::cout << newton->asString() << std::endl;

  const auto lagrangeAt = [&table, &eval, &lagrange](double x) {
    table.put(X, Symatica::Expression::DSL::l(x));
    return eval.valueOf(lagrange);
  };

  const auto newtonAt = [&table, &eval, &newton](double x) {
    table.put(X, Symatica::Expression::DSL::l(x));
    return eval.valueOf(newton);
  };

  std::cout << "Interpolation::Newton::dividedDifferences: " << std::endl;
  for (auto i = 0; i < N; i++) {
    std::cout << i << ": " << dividedDifference(points, i).value << std::endl;
  }

  for (const auto& x : testX) {
    std::cout << "lagrange(" << x << ") = " << lagrangeAt(x.value) << std::endl;
    std::cout << "newton(" << x << ") = " << newtonAt(x.value) << std::endl;
  }
}
```

Листинг 1: Тест для решения вычислительной задачи

2.3 Результаты измерений

Посмотрим на результаты:

```
Interpolation::symbolicLagrange:
(((((((0 + ((((((1.25 * ((x - 0.30) / -0.05)) * ((x - 0.35) / -0.10)) * ((x - 0.40) / -0.15)) * ((x - 0.45) / -0.20)) * ((x - 0.50) / -0.25)) * ((x - 0.55) / -0.30))) + ((((((2.17 * ((x - 0.25) / 0.05)) * ((x - 0.35) / -0.05)) * ((x - 0.40) / -0.10)) * ((x - 0.45) / -0.15)) * ((x - 0.50) / -0.20)) * ((x - 0.55) / -0.25))) + ((((((3.12 * ((x - 0.25) / 0.10)) * ((x - 0.30) / 0.05)) * ((x - 0.40) / -0.05)) * ((x - 0.45) / -0.10)) * ((x - 0.50) / -0.15)) * ((x - 0.55) / -0.20))) + ((((((4.04 * ((x - 0.25) / 0.15)) * ((x - 0.30) / 0.10)) * ((x - 0.35) / 0.05)) * ((x - 0.45) / -0.05)) * ((x - 0.50) / -0.10)) * ((x - 0.55) / -0.15))) + ((((((5.98 * ((x - 0.25) / 0.20)) * ((x - 0.30) / 0.15)) * ((x - 0.35) / 0.10)) * ((x - 0.40) / 0.05)) * ((x - 0.50) / -0.05)) * ((x - 0.55) / -0.10))) + ((((((6.91 * ((x - 0.25) / 0.25)) * ((x - 0.30) / 0.20)) * ((x - 0.35) / 0.15)) * ((x - 0.40) / 0.10)) * ((x - 0.45) / 0.05)) * ((x - 0.55) / -0.05))) + ((((((7.83 * ((x - 0.25) / 0.30)) * ((x - 0.30) / 0.25)) * ((x - 0.35) / 0.20)) * ((x - 0.40) / 0.15)) * ((x - 0.45) / 0.10)) * ((x - 0.50) / 0.05)))

Interpolation::symbolicNewton:
(((((((0 + (1.25 * 1)) + (18.41 * (1 * (x - 0.25)))) + (4.94 * ((1 * (x - 0.25)) * (x - 0.30)))) + (-58.26 * (((1 * (x - 0.25)) * (x - 0.30)) * (x - 0.35)))) + (7170.66 * ((((1 * (x - 0.25)) * (x - 0.30)) * (x - 0.35)) * (x - 0.40)))) + (-110072.00 * (((((1 * (x - 0.25)) * (x - 0.30)) * (x - 0.35)) * (x - 0.40)) * (x - 0.45)))) + (905928.88 * ((((((1 * (x - 0.25)) * (x - 0.30)) * (x - 0.35)) * (x - 0.40)) * (x - 0.45)) * (x - 0.50))))
Interpolation::Newton::dividedDifferences:
0: 1.2557
1: 18.414
2: 4.94
3: -58.2667
4: 7170.67
5: -110072
6: 905929
lagrange(0.251) = 1.22013
newton(0.251) = 1.22013
lagrange(0.512) = 6.85843
newton(0.512) = 6.85843
lagrange(0.255) = 1.12252
newton(0.255) = 1.12252
lagrange(0.534) = 6.93994
newton(0.534) = 6.93994
lagrange(0.272) = 1.26425
newton(0.272) = 1.26425
lagrange(0.551) = 7.93733
```

```
newton(0.551) = 7.93733
lagrange(0.294) = 1.97998
newton(0.294) = 1.97998
lagrange(0.402) = 4.1112
newton(0.402) = 4.1112
lagrange(0.372) = 3.40014
newton(0.372) = 3.40014
lagrange(0.405) = 4.20971
newton(0.405) = 4.20971
lagrange(0.384) = 3.62562
newton(0.384) = 3.62562
lagrange(0.445) = 5.79299
newton(0.445) = 5.79299
lagrange(0.351) = 3.13273
newton(0.351) = 3.13273
lagrange(0.437) = 5.46635
newton(0.437) = 5.46635
```

Листинг 2: Результаты решения поставленной задачи (коэффициенты были обрезаны до 2 знаков после запятой)

2.4 Анализ результатов

Изобразим графики построенных многочленов. Получается, что Лагранж и Ньютон построили одинаковые многочлены.

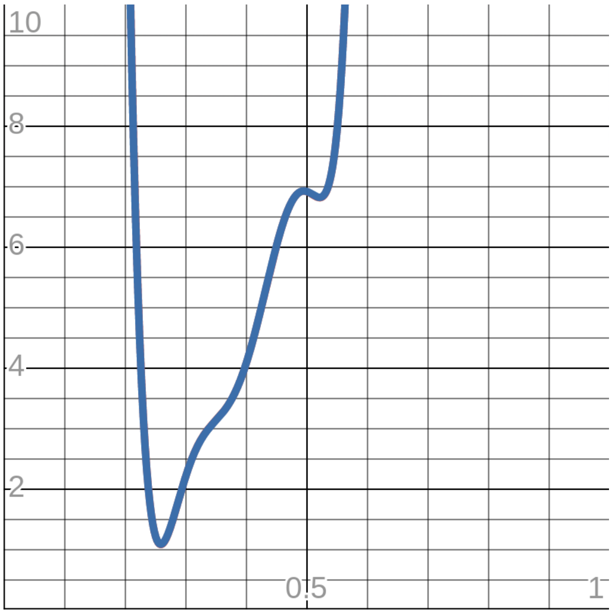

Рис. 1: График построенного многочлена

3 Программная реализация задачи

3.1 Метод Лагранжа

```
template <Count N>
Ptr<Expression> symbolicLagrange(
    const Array<Point<F>, N>& points, const Ptr<Variable>& x
) noexcept {
  const auto X
      = Array<double, N>([&points](auto i) { return points[i].x.value; });
  const auto Y
      = Array<double, N>([&points](auto i) { return points[i].y.value; });
  Ptr<Expression> poly = l(0);
  for (auto i = 0; i < N; i++) {
```

```
    Ptr<Expression> summand = l(Y[i]);
    for (auto j = 0; j < N; j++) {
      if (i != j) {
        summand = summand * ((x - l(X[j])) / l(X[i] - X[j]));
      }
    }
    poly = poly + summand;
  }
  return poly;
}
```

Листинг 3: Метод Лагранжа

3.2 Метод Ньютона

```
template <Abstract::Real F, Size N>
F dividedDifference(const Array<Point<F>, N>& points, Index n) noexcept {
  assert(n < N);
  auto sum = F::zero();
  for (auto j = 0; j < n + 1; j++) {
    auto numerator = points[j].y;
    auto denominator = F::unit();
    for (auto i = 0; i < n + 1; i++) {
      if (i != j) {
        denominator *= (points[j].x - points[i].x);
      }
    }
    sum += numerator / denominator;
  }
  return sum;
}

// Reference: https://www.youtube.com/watch?v=Bi4KzPGtdQU
template <Count N>
Ptr<Expression> symbolicNewton(
    const Array<Point<F>, N>& points, const Ptr<Variable>& x
) noexcept {
  Ptr<Expression> poly = l(0);
  for (auto k = 0; k < N; k++) {
    Ptr<Expression> tail = l(1);
    for (auto i = 0; i < k; i++) {
      tail = tail * (x - l(points[i].x.value));
    }
    poly = poly + l(dividedDifference(points, k).value) * tail;
  }
  return poly;
}
```

Листинг 4: Метод Ньютона

4 Вывод

Выполнив данную лабораторную работу я познакомился с базовыми методами построения интерполяционных многочленов. В отличие от задачи аппроксимации, интерполяция требует точного совпадения значений многочлена в заданных точках, но ничего особо не говорит про значения в других точках. Хотя на практике получается, что интерполяционные многочлены очень даже неплохо приближают функцию, однако при небольших степенях. На данном их свойстве основан метод интерполяции функции сплайнами: при таком подходе мы делим область определения функции на отрезки, на каждом отрезке интерполируем ее многочленом (часто степени 3), а потом кусочно задаем интерполяционную функцию, склеивая многочлены.

Список литературы

[1] Б.П. Демидович, И.А. Марон Основы вычислительной математики: учебное пособие — 1966 год.

[2] Лекции Татьяны Алексеевны Малышевой