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: 17

2023

Ключевые слова

Аппроксимация, Метод Наименьших квадратов, МНК.

1 Цель работы

Цель работы - найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов. А также был реализован программный модуль, предоставляющий API для аппроксимации вещественнозначных функций с одной переменной.

2 Вычислительная задача

Вычислительная задача была поставлена следующим образом: дана функция $f(x) = \frac{2x}{x^4+17}$, необходимо аппроксимировать ее линейной функцией и параболой на интервале $[0, 2]$, табулируя функцию с шагом 0.2.

Перед решением следует все же ввести читателя в курс дела и рассказать ему про МНК, не так ли? Поэтому отправим заинтересованного читателя ознакамливаться со статьей в Википедии.

На языке программирования C++ был реализован "Случай полиномиальной модели".

```cpp
namespace Mathematica::Functional::Approx::LeastSqare {

using Mathematica::Collection::Array;

template <Abstract::Field F, Algebra::PolynomialDegree D, Count N>
Algebra::Linear::Matrix<F, D, D> buildMatrix(const Array<Point<F>, N>& points
) noexcept {
  const auto sums = Array<F, 2 * D - 1>([&points](auto i) {
    auto sum = F::zero();
    for (const auto point : points) {
      sum += point.x.pow(i);
    }
    return sum;
  });
  return Algebra::Linear::Matrix<F, D, D>([&sums](auto i, auto j) {
    return sums[i + j];
  });
}

template <Abstract::Field F, Algebra::PolynomialDegree D, Count N>
Algebra::Linear::Vector<F, D> buildVector( //
    const Array<Point<F>, N>& points
) {
  return Algebra::Linear::Vector<F, D>([&](auto i) {
    auto value = F::zero();
    for (const auto point : points) {
      value += point.x.pow(i) * point.y;
    }
    return value;
  });
}

template <Abstract::Field F, Algebra::PolynomialDegree D, Count N>
Algebra::Polynomial<F, D> optimalPolynomial( //
    const Array<Point<F>, N>& points
) noexcept {
  const auto matrix = buildMatrix<F, D, N>(points);
  const auto vector = buildVector<F, D, N>(points);
  const auto gauss = Algebra::Linear::Eq::GaussSolver<F, D>();
  const auto coefficients = gauss.solve({matrix, vector}).value;
  return Algebra::Polynomial<F, D>(coefficients);
}

}
```

Листинг 1: Реализация метода наименьших квадратов на языке C++

С использованием данной функциональности напишем простенький тест для получения результатов вычисления:

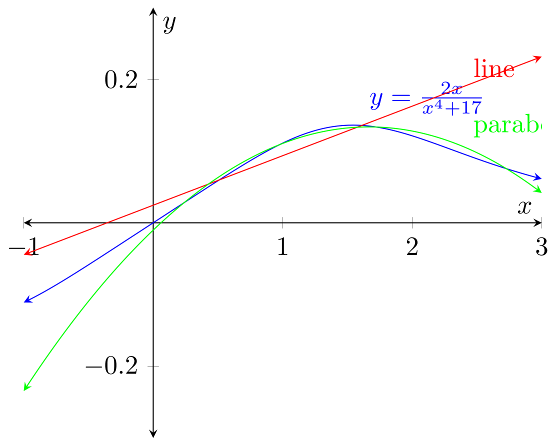
```cpp

TEST(Approx, Playground) { // NOLINT
  using Mathematica::Interop::symbolic;
  using Symatica::Expression::DSL::var;

  using namespace Mathematica; // NOLINT
  using R = Mathematica::Abstract::Float<double>;
  using namespace Mathematica::Functional; // NOLINT
  constexpr auto MIN = 0;
  constexpr auto MAX = 2;
  constexpr auto FINENESS = 0.2;
  constexpr Mathematica::Count N = (MAX - MIN) / FINENESS;

  const auto scope = Interval<R>(MIN, MAX);
  const auto f = [](R x) -> R {              // NOLINT
    return (R(2) * x) / (x.pow(4) + R(17)); // NOLINT
  };

  const auto points = Exploration::Tabulate::trivial<R, N>(f, scope);
  for (const auto& point : points) {
    std::cout << "( " << point.x.asString() //
              << ", " << point.y.asString() //
              << " )" << std::endl;
  }

  const auto line = Approx::LeastSqare::optimalPolynomial<R, 2>(points);
  const auto sline = symbolic(line, var(1));
  const auto parabola = Approx::LeastSqare::optimalPolynomial<R, 3>(points);
  const auto sparabola = symbolic(parabola, var(1));

  const auto lineS = Statistics::standartDeviation(line, points);
  const auto parabolaS = Statistics::standartDeviation(parabola, points);

  const auto lineR2 = Statistics::Score::R2(line, points);
  const auto parabolaR2 = Statistics::Score::R2(parabola, points);

  std::cout << "Line: " + sline->asString() << std::endl;
  std::cout << "Line: STD = " + lineS.asString() << std::endl;
  std::cout << "Line: R2 = " + lineR2.asString() << std::endl;

  std::cout << "Parabola: " + sparabola->asString() << std::endl;
  std::cout << "Parabola: STD = " + parabolaS.asString() << std::endl;
  std::cout << "Parabola: R2 = " + parabolaR2.asString() << std::endl;
}
```

Листинг 2: Тест для решения вычислительной задачи

```
( 0.100000, 0.011765 )
( 0.300000, 0.035277 )
( 0.500000, 0.058608 )
( 0.700000, 0.081206 )
( 0.900000, 0.101948 )
( 1.100000, 0.119150 )
( 1.300000, 0.130942 )
( 1.500000, 0.135977 )
( 1.700000, 0.134111 )
( 1.900000, 0.126531 )
Line: (0.024521 + (0.069030 * $1))
Line: STD = 0.015459
Line: R2 = 0.868067
Parabola: ((-0.010176 + (0.172604 * $1)) + (-0.051787 * ($1 ^ 2)))
Parabola: STD = 0.003526
Parabola: R2 = 0.993138
```

Листинг 3: Результаты вывода программы для решения вычислительной задачи

Итак, мы получили прямую линейного тренда $y = 0.069030 \cdot x + 0.024521$ и параболу $y = -0.051787 \cdot x^2 + 0.172604 \cdot x + -0.010176$. Видно, что парабола имеет меньшее среднеквадратичное отклонение, что толкает нас сделать вывод о том, что кдвадратическая зависимость лучше приближает заданную функцию, в чем нас окончательно убеждают показатели R^2.



3 Программная реализация задачи

4 Вывод

Список литературы

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