Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ «Информатика и системы управления»

КАФЕДРА ИУ-1 «Системы автоматического управления»

ОТЧЕТ

по лабораторной работе №4

«Интерполяция функций»

по дисциплине

«Методы вычислений»

Выполнил: Шевченко А.Д.

Группа: ИУ1-32Б

Проверил: Бобков А.В.

Работа выполнена: 23.11.2022

Отчет сдан:

Оценка:

Москва 2022

Оглавление

Цель работы

Реализация методов интерполяции функций одной переменной (интерполяция методом ближайшего соседа, линейной интерполяции, интерполяцией молиномом Лагранжа, интерполяцией полиномом Ньютона, интерполяцией кубическим сплайном) и функцией нескольких переменных (методом билинейной интерполяции).

Интерполяция

Интерполяция — это метод нахождения неизвестных промежуточных значений некоторой функции по имеющемуся дискретному набору ее известных значений.

1. Метод ближайшего соседа

Интерполяция методом ближайшего соседа - метод интерполяции, при котором в качестве промежуточного значения выбирается ближайшее известное значение функции.

+ Простота реализации

+ Скорость работы

- Не дифференцируемая функция

- Плохо реализуется физически

2. Линейная интерполяция

Значения функции в точке определяется по следующей формуле:

$$\frac{y - f(x_0)}{f(x_1) - f(x_0)} = \frac{x - x_0}{x_1 - x_0},$$

+ Средняя скорость работы

+ Функция непрерывна

- Негладкая функция

- Не существуют производные высоких порядков

```
function [y0] = linear_interpolatioin(x,y,x0)
%
% linear_interpolatioin(x,y,x0) реализует линейную интерполяцию
% возвращает прогнозируемое значения y0 для точки x0.
% x - вектор значений x
% y - вектор значений y
%
n = length(x);
k = 1;
for i = 2:n-1
    if (x(i) < x0) && (abs(x0 - x(i)) < abs(x0 - x(k)))
        k = i;
    end
end
y0 = (y(k+1) - y(k))/(x(k+1) - x(k)) * (x0 - x(k)) + y(k);
end
```

3. Интерполяция полиномом Лагранж

Строится полином Лагранжа по следующему принципу:

$$L(x) = \sum_{i=0}^{n} y_i * l_i(x).$$

$$l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x_j - x}{x_j - x_i}$$

\+ Гладкая и непрерывная функция

\+ Функция дифференцируема n раз

\- Низкая скорость работы

\- Чувствительность к шуму

```
function [y0] = Lagrange_interpolation(x,y,x0)
%
% Lagrange_interpolation(x,y,x0) реализует интерполяцию методом Лагранжа
% возвращает прогнозируемое значения y0 для точки x0.
% x - вектор значений x
% y - вектор значений y
%
n = length(x);
y0 = 0;
for i = 1:n
    L = 1; % определяем полином Лагранжа
    for j = 1:n
        if j~=i
            L = L * (x(j) - x0) / (x(j) - x(i));
        end
    end
y0 = y0 + L * y(i);
end
end

```

4. Интерполяция полиномом Ньютона

Вычисляется полином Ньютона с помощью разделенных разностей

\+ Гладкая и непрерывная функция

\+ Функция дифференцируема n раз

\- Низкая скорость работы

\- Чувствительность к шуму

\- Сложность в написании

```
function [y0] = Newton_interpolation(x,y,x0)
%
% Newton_interpolation(x,y,x0) реализует интерполяцию методом Ньютона
% возвращает прогнозируемое значения y0 для точки x0.
% x - вектор значений x
% y - вектор значений y
%
n = length(x);
y0 = 0;
p = 1;

function [dy] = rasnost(a,b)
if (a == b)
    dy = y(a);
else
    dy = (rasnost(a+1, b) - rasnost(a, b-1))/(x(b) - x(a));
end
end

for i = 1:n
    y0 = y0 + rasnost(1, i) * p; %y0 = y0 + rasnost(x(1), x(i)) * p;
    p = p * (x0 - x(i));
end
end
```

5. Интерполяция сплайном

На каждом из отрезков (x_i, x_{i+1}) функция f(x) приближается параболой S_i (сплайном 2 порядка), которая удовлетворяет следующим условиям:

1) Сплайн должен проходить через узловые точки:

$$y_i = a_i x^2{}_i + b_i x_i + c_i$$

$$y_{i+1} = a_i x^2{}_{i+1} + b_i x_{i+1} + c_i$$

2) Производные слева и справа должны быть одинаковыми

$$2a_{i+1}x_{i+1} + b_{i+1} = 2a_i x_{i+1} + b_i$$

(Код размещён на следующей странице)

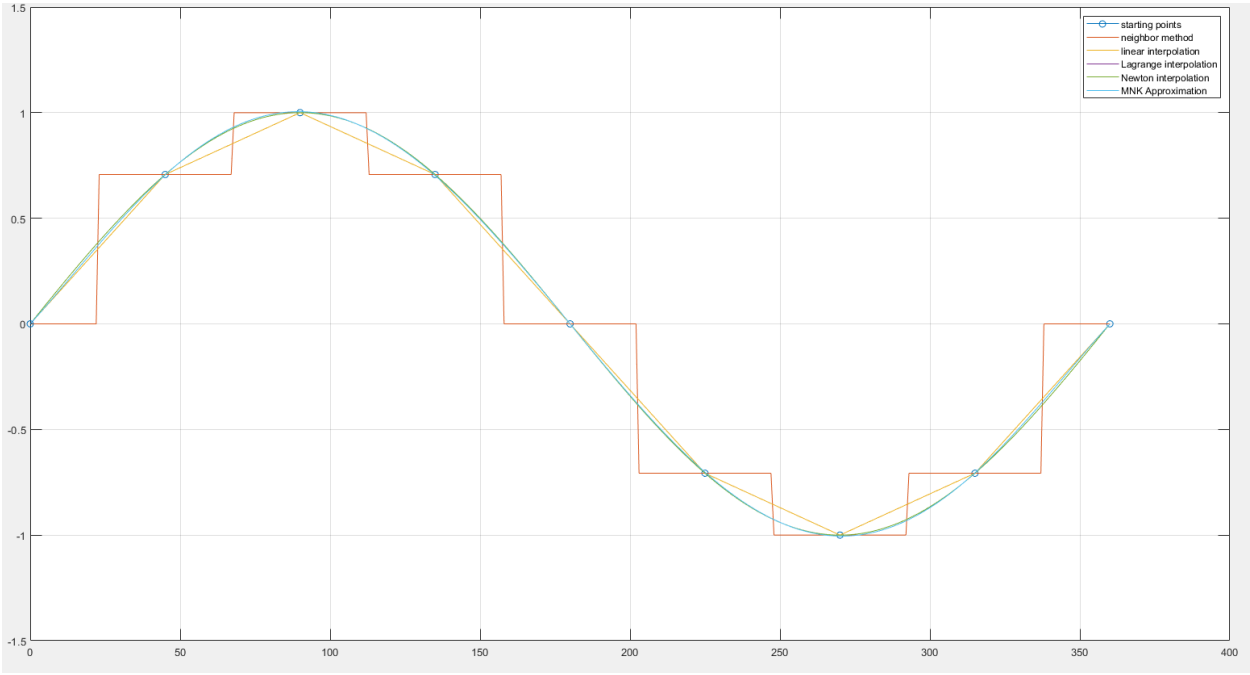
```
function [y0] = spline_interpolation(x,y,x0)
%
% spline_interpolation(x,y,x0) реализует интерполяцию сплайнами 2-го порядка
% x - вектор значений x
% y - вектор значений y
%
n = length(x)-1;
A = zeros(3*n-3,3*n-3);
B = zeros(3*n-3,1);
for i = 1:n-1
    j = i + n - 1;
    k = i + 2*n - 2;
    A(i,i) = x(i)^2;
    A(i,j) = x(i);
    A(i,k) = 1;
    B(i) = y(i);
    A(j,i) = x(i+1)^2;
    A(j,j) = x(i+1);
    A(j,k) = 1;
    B(j) = y(i+1);
    if i < n-1
        A(k,i) = 2*x(i+1); % -2*x(i+1)
        A(k,i+1) = -2*x(i+1); % 2*x(i+1)
        A(k,j) = 1; % -1
        A(k,j+1) = -1; % 1
    else
        A(k,i) = x(n+1)^2;
        A(k,j) = x(n+1);
        A(k,k) = 1;
        B(k) = y(n+1);
    end
end
U = A^(-1)*B;
k = 1;
for i = 2:n-1
    if (abs(x0-x(i)) < abs(x0-x(k)) && (x(i) < x0))
        k = i;
    end
end
a = U(k);
b = U(k+n-1);
c = U(k+2*n-2);
y0 = a*x0^2 + b*x0 + c;
disp(x0), disp(A), disp(B), disp(U);
disp('koef'), disp(a), disp(b), disp(c);
disp('-----')
end
```

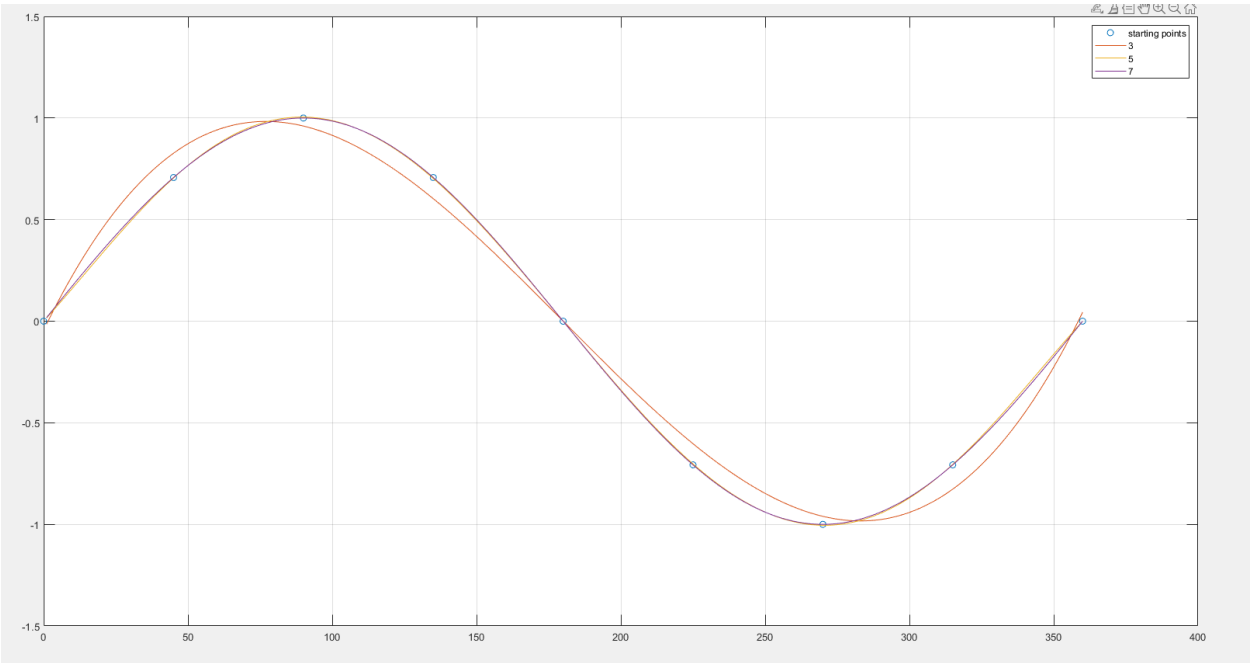
6. Аппроксимация методом наименьших квадратов

```
function [y0] = MNK_approximation(x,y,x0, m)
%
% LSM_approximation(x, y, x0, m)реализует аппроксимацию методом наименьших квадратов,
% m – степень аппроксимируемого полинома
%
m = m + 1;
n = length(x);
A = zeros(m, m);
B = zeros(m, 1);
for i = 1:m
    for j = 1:m
        S = 0;
        for k = 1:n
            S = S + x(k)^(i-1 + j-1);
        end
        A(i,j) = S;
    end
end
for i = 1:m
    S = 0;
    for k = 1:n
        S = S + y(k)*x(k)^(i-1);
    end
    B(i) = S;
end
U = A^(-1) * B;
y0 = 0;
 for i = 1:m
     y0 = y0 + U(i)*x0^(i-1);
 end
 end

```

Сравнение методов на примере синусоиды, заданной на отрезке [0, 360] с шагом 45





Вывод:

Запустим каждый метод 1000 раз и сравним время работы:

```
neighbor_method: 0.000649
linear_interpolatioin: 0.000515
Lagrange_interpolation: 0.000639
Newton_interpolation: 0.020324
spline_interpolation: 0.013377
MNK_approximation: 0.015742
```