

## ЛАБ. РАБОТА №7. Аппроксимация таблично заданной функции.

### Постановка задачи

Пусть некоторую функцию  $F(x)$ , представленную табличными значениями, на аргументах  $(X_0, \dots, X_m, \dots, X_M)$  в области  $(X_b, X_e)$ , необходимо наилучшим образом (то есть наиболее точным и, по возможности, наиболее компактным образом) представить в аналитическом виде.

Такая задача получила наименование – «задача аппроксимации таблично заданной функции».

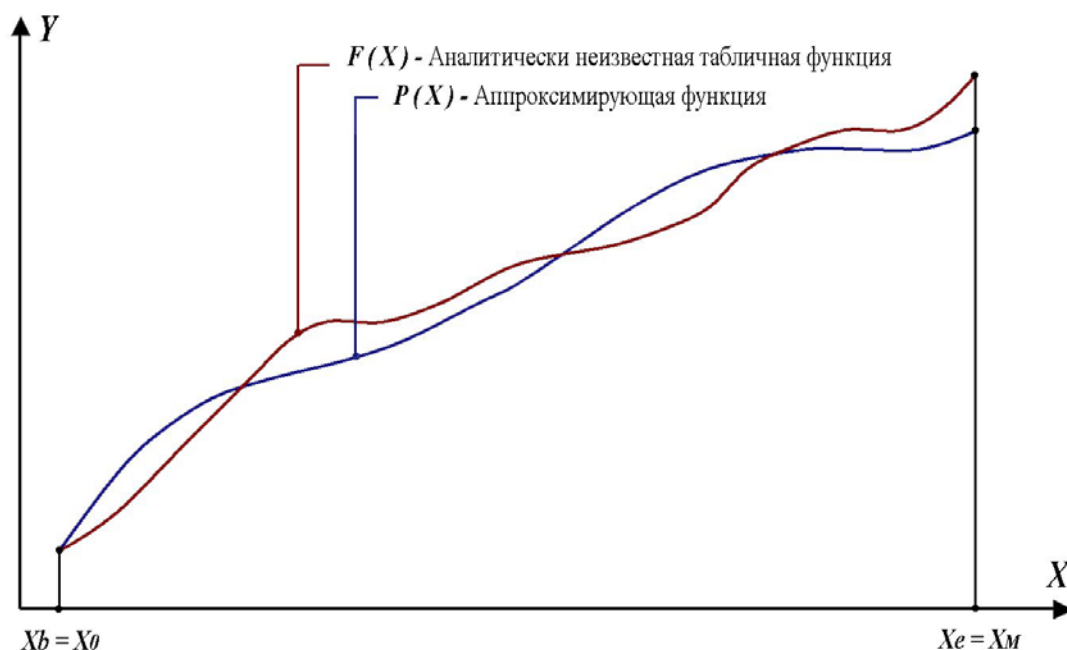


Рис.1. Постановка задачи аппроксимации таблично заданной функции.

Эта задача, как правило, решается с помощью *аппроксимирующей* функции  $P(x)$  путем определения ее коэффициентов  $(a_0, \dots, a_i, \dots, a_n)$  таким образом, чтобы достигнуть минимального различия между исходной таблично заданной функцией  $F(x)$  и ее аналитическим эквивалентом, то есть:

$$F(x) \cong P(x) = \sum_{n=0}^N a_n v_n(x)$$

Где:

$v_n(x)$  - является одной из заранее выбранных аналитических функций, которые называют базисными функциями;

$a_n$  - является искомыми коэффициентом при этой функции.

Как вы уже догадались, для того, чтобы найти решение поставленной задачи, как минимум, необходимо математически определить условия, при которых достигается равенство:

$$F(x) \cong P(x)$$

Следует также подчеркнуть, что успех решения задачи аппроксимации зависит также от того, будут ли такие условия конструктивны с точки зрения достижения поставленной цели.

## Задача аппроксимации в дискретной форме

Классическим примером условий, которые позволяют решить задачу аппроксимации, являются условия, сформулированные в методе наименьших квадратов. Построение таких условий мы рассмотрим ниже.

### Построение условий для решения задачи аппроксимации в дискретной форме.

Для начала, исходя из предположения о существовании аналитического вида функций, графически представим функцию  $F(x)$ , а также некоторую аппроксимирующую функцию  $P(x)$ , которая пока еще не вполне удовлетворяет решению нашей задачи.

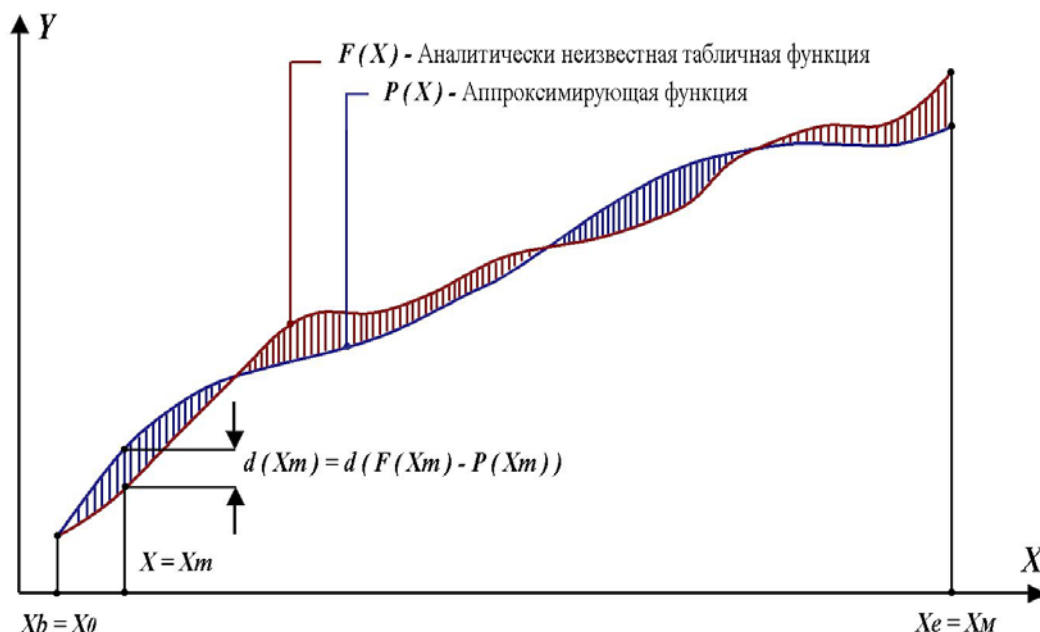


Рис.2. Отклонения между аппроксимируемой и аппроксимирующей функциями.

### Шаг 1. Дискретные отклонения и их сумма

Выводы, которые непосредственно доступны из графического представления функций  $F(x)$  и  $P(x)$  можно сформулировать следующим образом:

- **Вывод 1.** Если бы нам сразу удалось достигнуть равенства функций, то для каждого значения независимой переменной  $x$ , отклонение  $d$  между соответствующими значениями этих функций было бы нулевым.
- **Вывод 2.** Сумма всех отклонений будет тем ближе к нулю, чем ближе к нулю будет каждое из составляющих сумму отклонений. Однако, данное утверждение будет справедливым только в том случае, если все отклонения  $d$  мы будем рассматривать без учета их знака.
- **Следствие** – Если сумма всех отклонений (без учета их знака) стремится к нулю,

$$\sum_{m=0}^M d(x_m) \rightarrow 0$$

то к нулю будут стремиться все входящие в нее отклонения. Таким образом, если мы построим формальный метод, вычисляющий коэффициенты  $(a_0, \dots, a_i, \dots, a_K)$ , при этом значения этих коэффициентов позволит наилучшим образом приблизить такую сумму к нулевому значению, то это будет равнозначно решению задачи аппроксимации:

$$F(x) \cong P(x)$$

## Шаг 2. Отклонения в форме модуля

Для начала, определим отклонение  $d(x)$  между функциями  $F(x)$  и  $P(x)$  как простую разность между этими функциями:

$$d(x) = F(x) - P(x)$$

К сожалению, простые разности, полученные при различных значениях  $x$ , могут иметь как положительные, так и отрицательные значения. В итоге, суммируя все положительные и отрицательные отклонения между функциями  $F(x)$  и  $P(x)$ , вполне возможно получить нулевой результат при очевидном неравенстве этих функций. Для этого достаточно, чтобы сумма всех положительных отклонений случайным образом оказалась равна сумме всех отрицательных отклонений.

На первый взгляд, этот недостаток легко устранить - достаточно разность между функциями записать в форме модуля, и проблема будет устранена:

$$d_{\text{mod}}(x_m) = |F(x_m) - P(x_m)|$$

Однако применение модуля вносит в такое выражение особые точки, то есть точки, в которых производные, взятые слева и справа от точки не равны между собой. Появление таких особых точек (а их количество явно непредсказуемо) сразу вносит существенные сложности в методы поиска минимумов, которые нам далее придется использовать.

## Шаг 3. Отклонения в квадратичной форме

Проблема особых точек, обнаруженная нами на шаге 2, ставит перед нами простой вопрос – что для нас более важно, удобная и надежная оценка равенства функций  $F(x)$  и  $P(x)$  или привязка конструируемой оценки к отклонению в виде *простой разницы* между этими функциями?

Ответ очевиден – *важно получить однозначную, всюду положительную, непрерывную функцию оценки, избавленную от особых точек*. Самый простой способ получить такую функцию оценки, это представить отклонение как квадрат разности функций  $F(x)$  и  $P(x)$  :

$$d_{sq}(x_m) = (F(x_m) - P(x_m))^2$$

В этом случае, сумма всех отклонений определенная как:

$$\sum_{m=0}^M d_{sq}(x_m) \rightarrow 0$$

будет располагать всем, необходимым для нас, набором свойств:

- Сумма будет гарантировано положительна и ограничена снизу нулевым значением, которое является индикатором равенства функций  $F(x)$  и  $P(x)$
- Операции, которые использовались при конструировании суммы, не будут вносить особых точек, препятствующих применению методов минимизации.

#### **Шаг 4. Формальное условие для решения задачи аппроксимации**

На основании результатов шага 3, запишем окончательное выражение для формальной оценки успешности решения задачи аппроксимации можно записать следующим образом:

$$D = \sum_{m=0}^M ((F(x_m) - P(x_m)))^2 \rightarrow 0$$

Или:

$$D = \sum_{m=0}^M ((F(x_m) - \sum_{n=0}^N a_n v_n(x_m)))^2 \rightarrow 0$$

#### **Решение задачи аппроксимации в дискретной форме**

В рамках лабораторной работы мы опустим все шаги, которые последовательно приводят нас к построению следующей системы линейных уравнений относительно неизвестных  $a_n$ , решение которой позволяет найти аппроксимирующую функцию  $P(x)$ :

$$\begin{cases} \sum_{n=0}^N a_n \cdot C_{0,n} = B_0 \\ \dots \\ \sum_{n=0}^N a_n \cdot C_{N,n} = B_N \end{cases}$$

где:

$$B_k = \sum_{m=0}^M F(x_m) \cdot v_k(x_m)$$

$$C_{k,n} = \sum_{m=0}^M v_k(x_m) \cdot v_n(x_m)$$

Легко заметить, что в эти суммы входят функции, которые, либо даны нам как табличные значения для всех своих аргументов  $x$ , либо, учитывая известный аналитический вид функций  $v(x)$ , могут быть вычислены как значения для такого же набора аргументов  $x$ . Таким образом, рассматриваемые суммы  $B_k$  и  $C_{k,n}$ , можно рассматривать как константы, значения которых зависят только от индексов  $k$  и  $n$ .

## **РЕАЛИЗАЦИИ МЕТОДОВ НАИМЕНЬШИХ КВАДРАТОВ** (Least Squares\_Method)

### **МЕТОД НАИМЕНЬШИХ КВАДРАТОВ ДЛЯ БАЗИСНЫХ ФУНКЦИЙ $x^n$**

```
% Least Squares_Method
% МЕТОД НАИМЕНЬШИХ КВАДРАТОВ ДЛЯ БАЗИСНЫХ ФУНКЦИЙ  $x^n$ 
% Синтаксис вызова:
%                               [A]=LSM01(F,nV)
%
% F - Исходная матрица табличная функция вида:
%
%   x1 x2 x3 x4 .... xM
%   y1 y2 y3 y4 .... yM
%
% nV - Длина рапроксимирующего степенного ряда
%
% Например (подготовка тестовых данных):
%
%   x=[0:pi/100:2*pi];
%   y=eval('x+sin(x)-log(x+1)');
%
%   F=[x;y];
%
% Вызов метода:
%   [A]=LSM01(F,10)
%
% Результат:
% Максимальная абсолютная ошибка : 0.00028535
% A =
%   -0.0003
%    1.0068
%    0.4619
%   -0.4028
%    0.1001
%   -0.0232
%    0.0070
%   -0.0013
%    0.0001
%   -0.0000
%
function [A]=LSM01(F,nV)
if isempty(F)
    disp('Матрица [F] не задана');
    A=NaN;
    return
end;
% Минимальный входной контроль
if (size(F,1) ~= 2)
    disp('Размеры матрицы [F] заданы неправильно');
    A=NaN;
    return
end;
if (size(F,2) < nV)
    disp('Недостаточно точек в [F] для определения коэффициентов для [V]');
end;
```

```

        A=NaN;
        return
    end;
    % Подготовка рабочих матриц
    C=zeros(nV,nV);
    B=zeros(nV,1);
    % Построим систему линейных уравнений для аппроксимации
    % Для всех строк матрицы 'C'
    for row=1:nV
        % Для всех 'x' матрицы 'F'
        B(row,1)=0;
        for k=1:size(F,2)
            x=F(1,k);
            B(row,1)=B(row,1)+ (F(1,k)^(row-1))*F(2,k);
        end;
        % Для всех колонок матрицы 'C'
        for col=1:nV
            % Для всех 'x' матрицы 'F'
            C(row,col)=0;
            for k=1:size(F,2)
                x=F(1,k);
                C(row,col)=C(row,col)+(F(1,k)^(row-1))*(F(1,k)^(col-1));
            end;
        end;
    end;
    % Вычислим коэффициенты для аппроксимирующего выражения
    ab=cat(2,C,B);
    A = fgauss01(ab);
    % Ниже представлены варианты, которые иногда выдают предупреждения
    % A=inv(C)*B; % A=C\B;

    % Вычислим значения функции по аппроксимирующему выражению
    % Для всех 'x' матрицы 'F'
    for k=1:size(F,2)
        x=F(1,k);
        y(1,k)=0;
        % Для всех строк матрицы 'V'
        for row=1:nV
            y(1,k)=y(1,k)+A(row,1)*(F(1,k)^(row-1));
        end;
    end;

    % Построим график исходной табличной функции
    % plot(F(1,:),F(2,:));
    % Построим график по результатам аппроксимации
    % plot(F(1,:),y);

    % Построим график абсолютной ошибки
    delta=F(2,:)-y;
    maxdelta=max(abs(delta));
    disp(strcat('Максимальная абсолютная ошибка : ',
    num2str(maxdelta)));
    plot(F(1,:),delta);

```

### **(1-1) ТЕСТ МЕТОДА НАИМЕНЬШИХ КВАДРАТОВ ДЛЯ БАЗИСНЫХ ФУНКЦИЙ $x^n$**

```

% SCRIPT Test01_LSM

```

```

disp('-----');
disp('ПОДГОТОВКА ТЕСТОВОГО ПОЛИНОМА');
x=[0:0.1:2];
c=[0.5 2 -1 0.3 1.7];
disp('Коэффициенты перечисляются от старших степеней к младшим');
y=polyval(c,x);
plot(x,y);
replay=input('Для продолжения нажмите Enter...');

disp('-----');
disp('АППРОКСИМАЦИЯ СТЕПЕННЫМ РЯДОМ');
F=[x;y];
[A1]=LSM01(F,5);
A1'
disp('Коэффициенты перечисляются от младших степеней к старшим');
replay=input('Для продолжения нажмите Enter...');

disp('-----');
disp('НАЛОЖЕНИЕ ШУМА С МАКСИМАЛЬНОЙ АМПЛИТУДОЙ');
y=polyval(c,x);
ns=rand(1,length(x));
maxns=max(abs(ns))
y=y+ns;
F=[x;y];
disp('АППРОКСИМАЦИЯ СТЕПЕННЫМ РЯДОМ');
[A2]=LSM01(F,5);
A2'
disp('Коэффициенты перечисляются от младших степеней к старшим');

```

## (2) МЕТОД НАИМЕНЬШИХ КВАДРАТОВ ДЛЯ СВОБОДНО ОПРЕДЕЛЯЕМЫХ БАЗИСНЫХ ФУНКЦИЙ

```

% Least Squares_Method
% МЕТОД НАИМЕНЬШИХ КВАДРАТОВ ДЛЯ СВОБОДНО ОПРЕДЕЛЯЕМЫХ БАЗИСНЫХ
% ФУНКЦИЙ
% Синтаксис вызова:
%
%                               [A]=LSM01Free(F,V)
%
% F - Исходная матрица табличная функция вида:
%
%   x1 x2 x3 x4 .... xM
%   y1 y2 y3 y4 .... yM
%
% V - Литеральный вектор столбец описывающий базисные функции :
%
%   v1
%   v2
%   .....
%   vN
%
% Например (подготовка тестовых данных):
%
%   x=[0:pi/100:2*pi];
%   y=eval('x+sin(x)-log(x+1)');
%
%   F=[x;y];
%   V = ['x^0      '; 'x      '; 'sin(x)  '; 'log(x+1)']

```

```

%
% ВНИМАНИЕ! Длина строк всех литералов должна быть строго
одинаковой и
% при необходимости расширяться пробелами справа.
%
% Вызов метода:
% [A]=LSM01(F,V)
%
% Результат:
% Максимальная абсолютная ошибка : 3.5527e-014
% A =
%    0.0000
%    1.0000
%    1.0000
%   -1.0000
%
function [A]=LSM01Free(F,V)
if isempty(F) || isempty(V)
    disp('Одна из входных матриц не задана');
    A=NaN;
    return
end;
% Минимальный входной контроль
if (size(F,1) ~= 2)
    disp('Размеры матрицы [F] заданы неправильно');
    A=NaN;
    return
end;
nV=size(V,1);
if (size(F,2) < nV)
    disp('Недостаточно точек в [F] для определения коэффициентов для
[V]');
    A=NaN;
    return
end;
% Подготовка рабочих матриц
C=zeros(nV,nV);
B=zeros(nV,1);
% Построим систему линейных уравнений для аппроксимации
% Для всех строк матрицы 'C'
for row=1:nV
    % Для всех 'x' матрицы 'F'
    B(row,1)=0;
    for k=1:size(F,2)
        x=F(1,k);
        B(row,1)=B(row,1)+eval(V(row,:))*F(2,k);
    end;
    % Для всех колонок матрицы 'C'
    for col=1:nV
        % Для всех 'x' матрицы 'F'
        C(row,col)=0;
        for k=1:size(F,2)
            x=F(1,k);
            C(row,col)=C(row,col)+eval(V(row,:))*eval(V(col,:));
        end;
    end;
end;
end;
% Вычислим коэффициенты для аппроксимирующего выражения

```



```

A=C\B;

% Вычислим значения функции по аппроксимирующему выражению
% Для всех 'x' матрицы 'F'
for k=1:size(F,2)
    x=F(1,k);
    y(1,k)=0;
    % Для всех строк матрицы 'V'
    for row=1:nV
        y(1,k)=y(1,k)+A(row,1)*eval(V(row,:));
    end;
end;

% Построим график исходной табличной функции
% plot(F(1,:),F(2,:));
% Построим график по результатам аппроксимации
% plot(F(1,:),y);

% Построим график абсолютной ошибки
delta=F(2,:)-y;
maxdelta=max(abs(delta));
disp(strcat('Максимальная абсолютная ошибка : ',
num2str(maxdelta)));
if max(abs(delta)) > 1e-16
    plot(F(1,:),delta);
end;

```

## (2-1) ТЕСТ ДЛЯ МЕТОДА НАИМЕНЬШИХ КВАДРАТОВ ДЛЯ СВОБОДНО ОПРЕДЕЛЯЕМЫХ БАЗИСНЫХ ФУНКЦИЙ

```

% SCRIPT Test02_LSM
hold on;
x=[0:pi/100:2*pi];
y=eval('x+sin(x)-3*log(x+1)');
% plot(x,y);
% ИСХОДНЫЕ ДАННЫЕ
F=[x;y];
% ТЕСТ1
V = ['x^0      '; 'x      '; 'sin(x)  '; 'log(x+1)'];
[A2]=LSM01Free(F,V)
replay=input('Для продолжения нажмите Enter...');
% ТЕСТ2
[A1]=LSM01(F,20)

```

---

## ПРАКТИЧЕСКАЯ РАБОТА.

1. Выполните все примеры из теоретической части лабораторной работы.
2. Восстановите блок – алгоритмическую схему метода наименьших квадратов для базисных функций  $x^n$ .
3. Восстановите блок – алгоритмическую схему метода наименьших для свободно определяемых базисных функций.

---

Финальная редакция материала 23.08.2018г.  
Воронов С.И.