Интерполяция и аппроксимация данных

- Для анализа экспериментальных данных, которые представлены в виде таблиц и задают зависимость одних физических величин от других, применяют такие средства, как интерполяция, сглаживание и аппроксимация.
- Если имеются некоторые табличные данные, возникает задача – найти непрерывную кривую, которая наилучшим образом соответствовала бы заданной экспериментальной зависимости. Обычно такие данные удобно интерпретировать в виде полиномиальной функции или сплайна.
- Программа MATLAB содержит встроенные функции для аппроксимации и интерполяции экспериментальных данных.

Полиномиальная аппроксимация

- Построить аппроксимирующий полином заданной степени, который приближает функцию одной переменной, заданную таблицей значений, позволяет функция **polyfit**. Эта функция реализует так называемый метод наименьших квадратов. Она имеет следующий синтаксис:
- **p=polyfit(x,y,n)**,
- где **y** – это вектор значений функции; **x** – вектор значений аргумента, **n** – порядок аппроксимирующего полинома; а **p** – полученный в результате вектор коэффициентов аппроксимирующего полинома длиной **n+1**.

Метод наименьших квадратов

Пусть задана таблица значений функции:

$$y_k = f(x_k),\ \ k = 1, m$$

Найдем коэффициенты полинома по критерию:

$$\min_p F(p) = \sum_{k=1}^{m} \left(\sum_{i=0}^{n} p_i \cdot x_k^i - y_k \right)^2$$

Точка локального минимума удовлетворяет уравнению:

$$\frac{\partial F(p)}{\partial p_j} = 2 \sum_{k=1}^{m} \left(\sum_{i=0}^{n} p_i \cdot x_k^i - y_k \right) \cdot x_k^j,\ \ j = \overline{0, n}$$

Обозначим : $A_{i,j} = \sum_{k=1}^{m} x_k^{i+j},\ B_i = \sum_{k=1}^{m} y_k \cdot x_k^i$ получим

Систему уравнений: $A \cdot p = B$

$$p = A \setminus B$$

Допустим, имеется массив значений аргумента:
x=[1 2 3 4 5 6 7 8 9 10]
и массив соответствующих им значений измеряемой величины:
y=[3 4 6 6.5 7 7.5 9 11 10 9]
Применим к этим данным алгоритм метода наименьших квадратов и сравним результат с функцией **polyfit.**

%Программа метода наименьших квадратов:

```
x=[1 2 3 4 5 6 7 8 9 10];
y=[3 4 6 6.5 7 7.5 9 11 10 9];
n=1;
for j=0:n
    w(j+1,:)=x.^j;
endfor
A=w*w';
B=w*y';
p=A\B
p =

   3.00000
   0.78182
```

%Функция polyfit:

```
x=[1 2 3 4 5 6 7 8 9 10];
y=[3 4 6 6.5 7 7.5 9 11 10 9];
n=1;
p=polyfit(x,y,n);
p =

   0.78182   3.00000
```

Компоненты вектора расположены в порядке убывания степени полинома

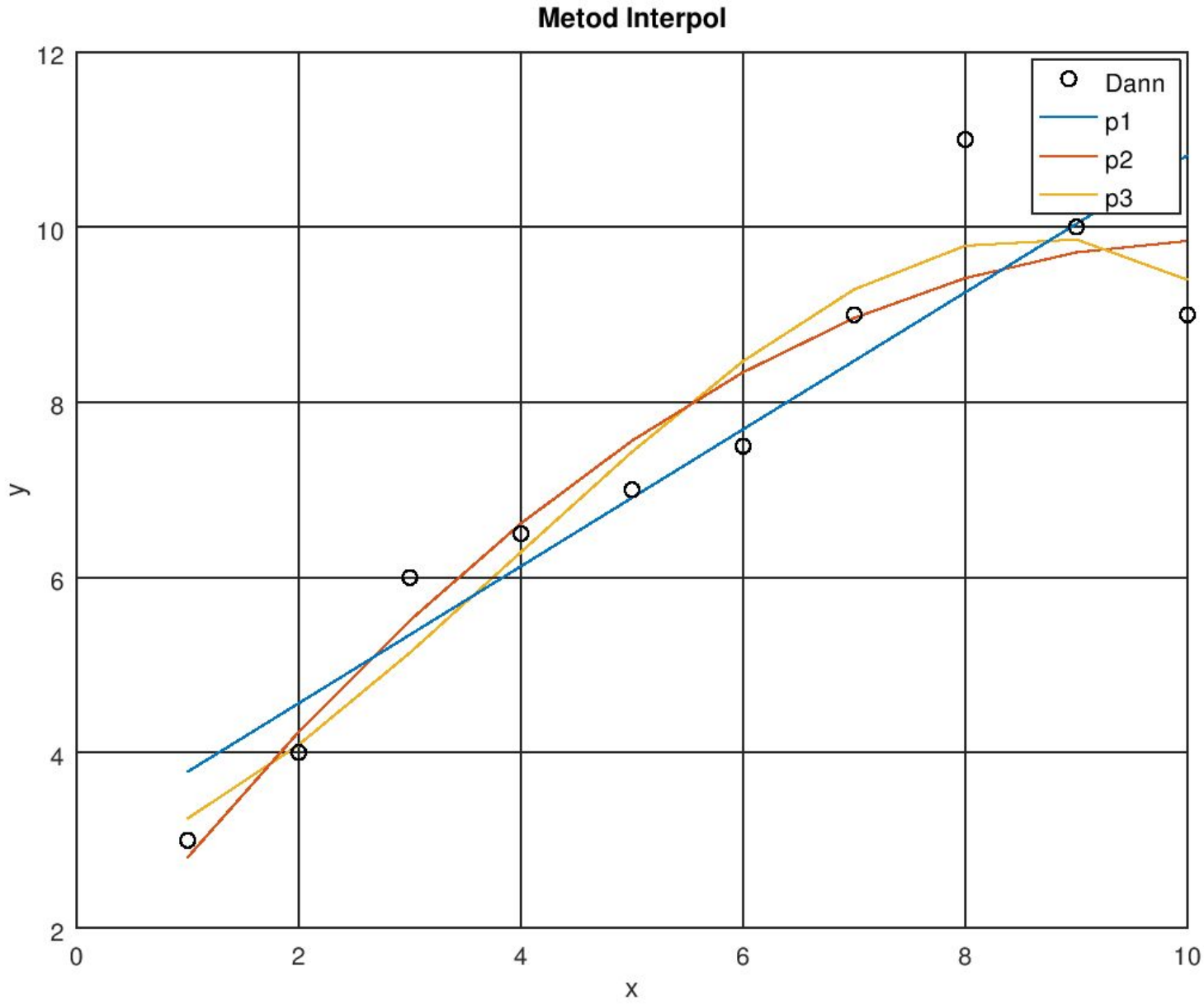
Исправим программу, чтобы получить правильный порядок степеней полинома

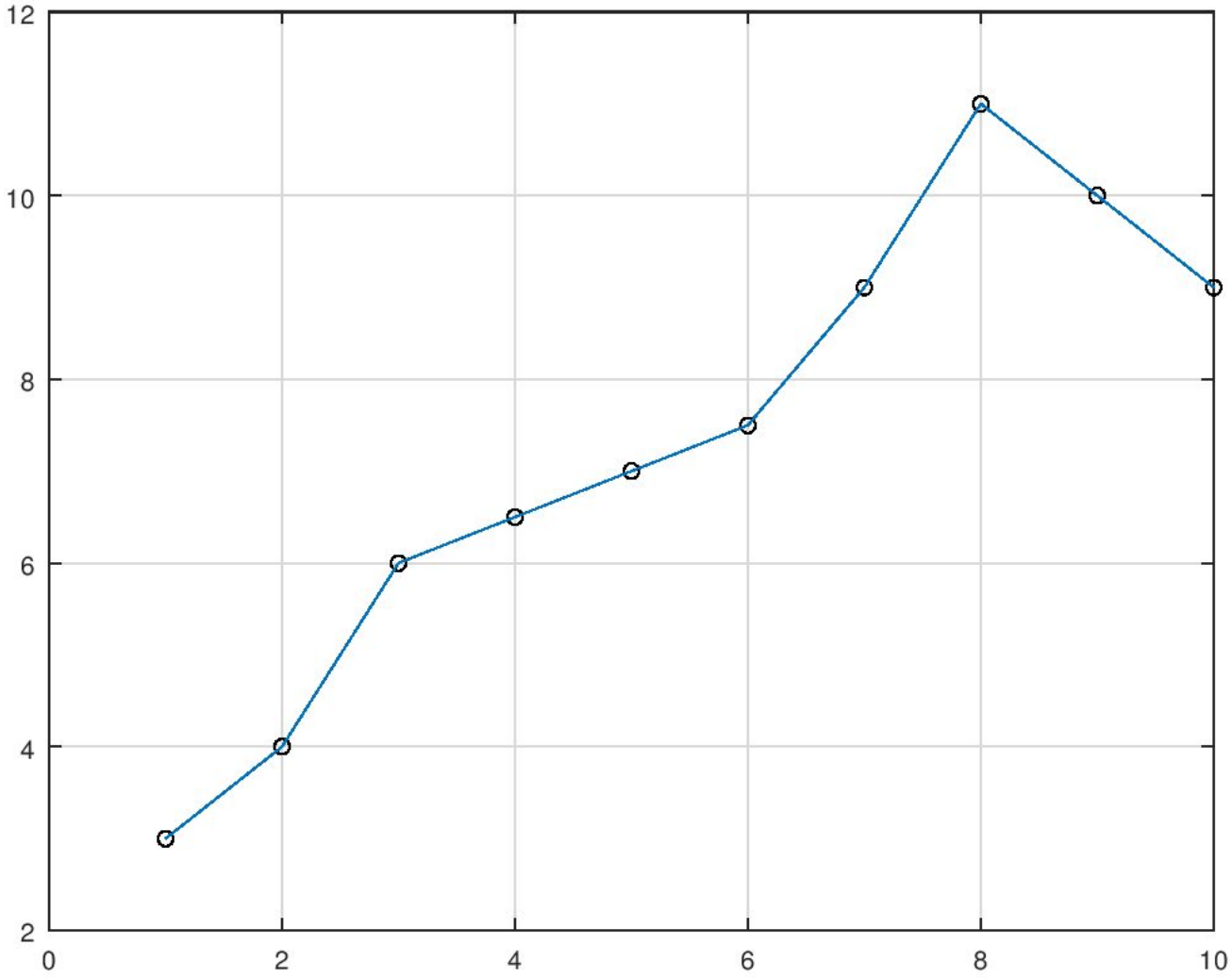
```
 n=1;
for j=0:n
w(j+1,:)=x.^(n-j);
endfor
A=w*w';
B=w*y';
p=A\B
p =

   0.78182
   3.00000
```

Функция полиномиальной аппроксимации

```
function p=poly_inter(x,y,n)
% Аппроксимация методом
наименьших %квадратов
    for j=0:n
     w(j+1,:)=x.^(n-j);
    endfor
    A=w*w';
    B=w*y';
    p=A\B;
endfuction
```

```
x=[1 2 3 4 5 6 7 8 9 10];
y=[3 4 6 6.5 7 7.5 9 11 10 9];
p1=polyfit(x,y,1);  q1=poly_inter(x,y,1);
p2=polyfit(x,y,2);  q2=poly_inter(x,y,2);
p3=polyfit(x,y,3);  q3=poly_inter(x,y,3);
plot(x,y,'ko',x,polyval(p1,x),x,polyval(p2,x),…
x,polyval(p3,x)); grid on
title(‘Metod Interpol');
 xlabel('x'); ylabel('y');
 legend(‘Dann',‘p1',‘p2',‘p3');
```


Metod Interpol
Dann
p1
p2
p3
12
10
8
6
4
2
0
2
4
6
8
10
y
x

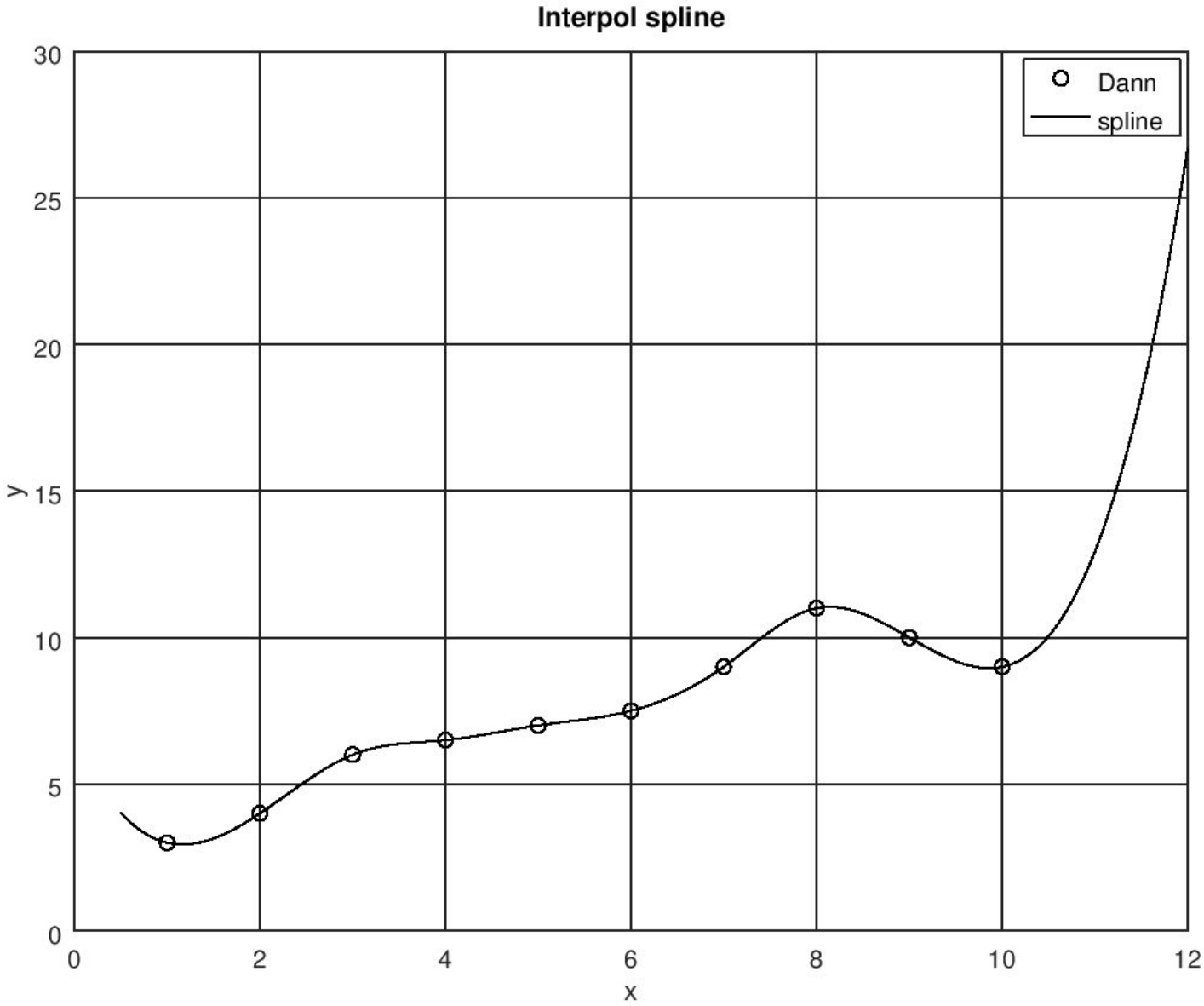



```
pn=polyfit(x,y,length(x));
plot(x,y,'ko',x,polyval(pn,x))
```

Выполнение приближения методом наименьших квадратов не всегда дает хороший результат. При увеличении степени полинома качество приближения может ухудшаться.
Повысить качество аппроксимации экспериментальных данных можно с помощью сплайнов. ***Сплайн*** – это непрерывная гладкая функция, которая на отрезках области определения равна полиномам определённой степени.

- При таком способе интерполяции экспериментальные точки попарно соединяются отрезками полиномов. Обычно используют полиномы третьей степени, поэтому данный метод и получил название ***интерполяция кубическими сплайнами*.**
- Интерполяцию кубическими сплайнами можно выполнить с помощью функции **spline**. Если к этой функции обратиться в форме
- **yy=spline(x,y,xx)**
- **x,y – исходные данные, xx- точки интерполяции, yy – значения функции в точках интерполяции**

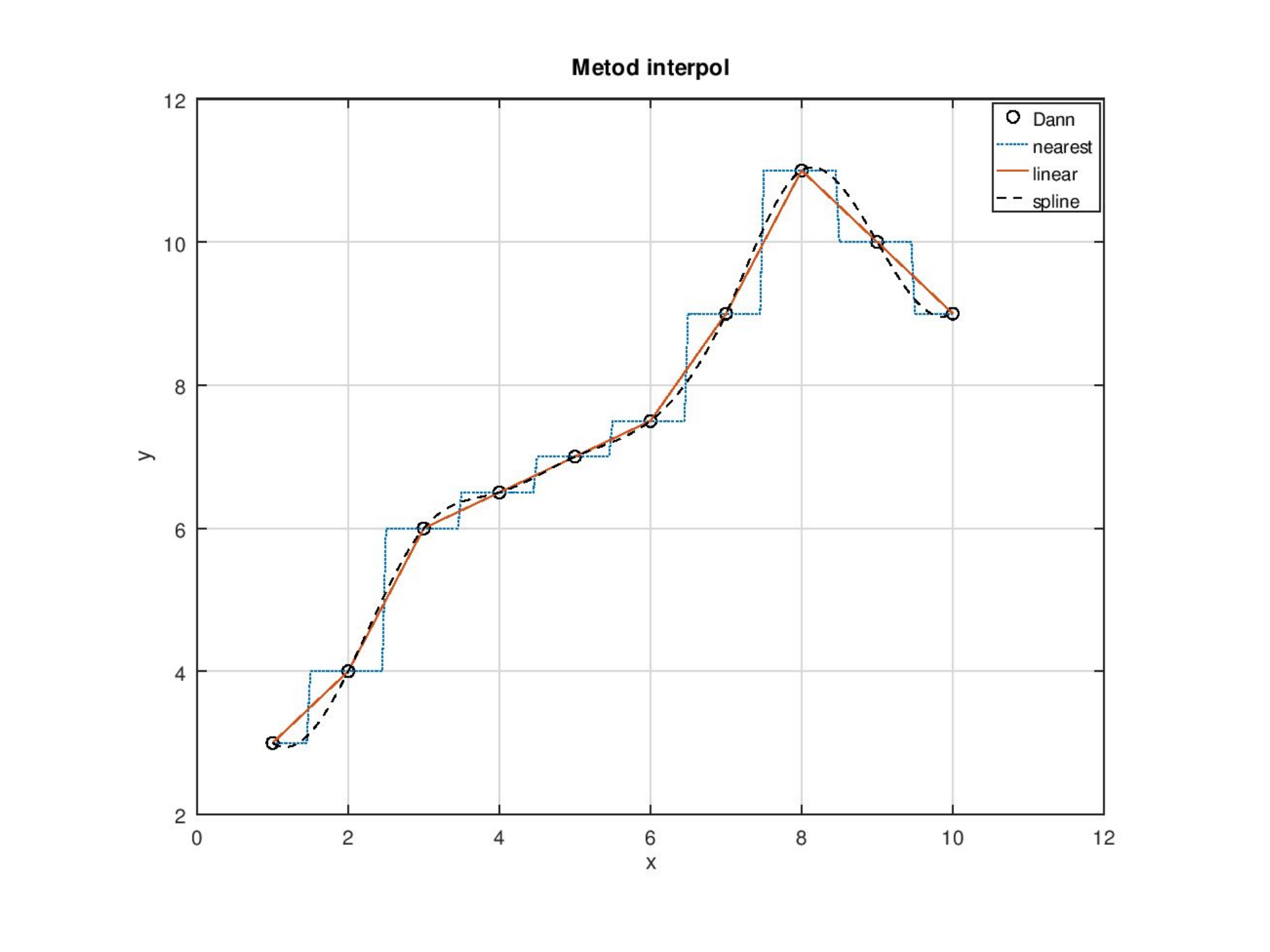
```
x=[1 2 3 4 5 6 7 8 9 10];
y=[3 4 6 6.5 7 7.5 9 11 10 9];
xx=0.5:0.05:12;
yy=spline(x,y,xx);
title(‘Interpol spline');
xlabel('x'); ylabel('y');
plot(x,y,'ko',xx,yy,'k-'),grid on
legend(‘Dann',‘spline')
```


Interpol spline
Dann
spline
30
25
20
15
10
5
0
y
0
2
4
6
8
10
12
x

Для одномерной интерполяции табличных данных в MATLAB имеется функция **interp**l:
yy=interpl(x,y,xx,method)

- В её четвёртом аргументе в виде строки символов задаётся метод интерполяции. Можно задать один из следующих методов:
- **‘nearest’** – ступенчатая интерполяция (когда значение в каждой промежуточной точке принимается равным ближайшему табличному значению);
- **‘linear’** – линейная интерполяция (соединение соседних точек отрезками прямых в соответствии с табличными данными);
- **‘spline’** – интерполяция кубическими сплайнами;
- **‘pchip’** – интерполяция кусочными полиномами Эрмита 3-й степени;
- Если метод не указан, по умолчанию используется **‘linear’**

```
x=[1 2 3 4 5 6 7 8 9 10];
y=[3 4 6 6.5 7 7.5 9 11 10 9];
plot(x,y,'ko')
xx=0.5:0.05:12;
yy1=interp1(x,y,xx,'nearest');
yy2=interp1(x,y,xx,'linear');
yy3=interp1(x,y,xx,'spline');
hold on
plot(xx,yy1,'k:',xx,yy2,'k-',xx,yy3,'k--'),grid on
title(‘Metod Interpol')
xlabel('x'); ylabel('y');
legend(‘Dann','nearest','linear','spline')
```


Metod interpol
Dann
nearest
linear
spline
y
x
12
10
8
6
4
2
0
2
4
6
8
10
12

Решение нелинейных уравнений

- **fzero** *(fun, x0)*
- **fzero** *(fun, x0, options)*
- *[x, fval, info, output] =* **fzero** *(…)*
- Находит нуль одномерной функции.
- fun- указатель на функцию, встроенная функция, или строка, содержащие имя функции.
- x0 должно быть двух элементным вектором, определяющим две точки, которые заключают нуль функции. Другими словами, нуль функции находится между x0(1) и x0(2).

- Если x0 - единственный скаляр, тогда ищутся соседние величины в попытке, чтобы найти отрезок, включающий нуль функции. Если это не удается, то выдается сообщение об ошибке.
- *options* является структурой, определяющей дополнительные опции. К настоящему времени, fzero включает следующие опции: *"FunValCheck"*, *"OutputFcn"*, *"TolX"*, *"MaxIter"*, *"MaxFunEvals"*.
- На выходе, функция возвращает x, приближенный нуль.
- fval- значение функции в этой точке.
- info - выходной флаг, который может иметь значение:
- 1 - алгоритм сходится.
- 0 – достигнуто максимальное число итераций.
- -1 - алгоритм был остановлен из функции пользователя.
- -5 - алгоритм вошел в сингулярную точку.

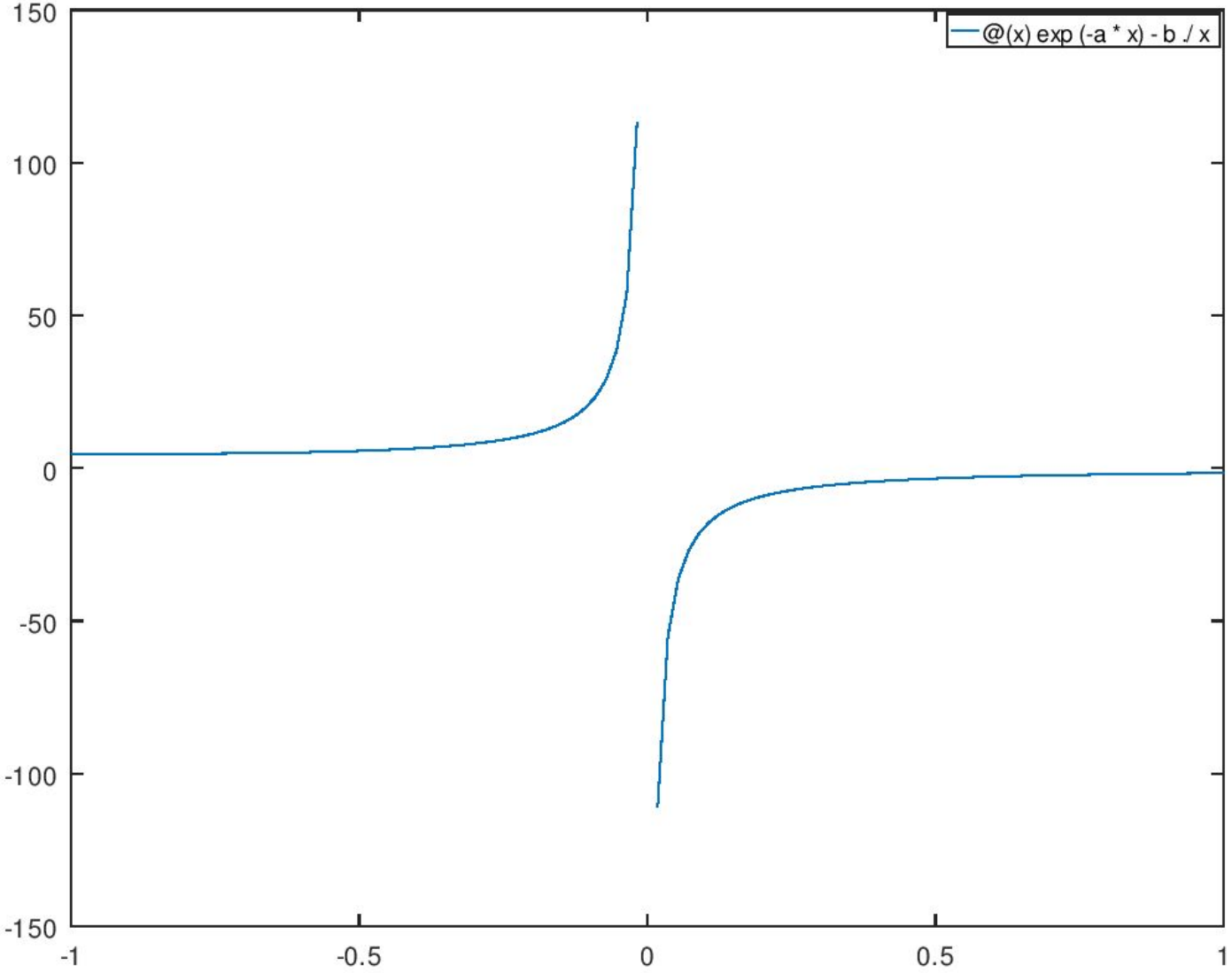
- ***output*** - является структурой, содержащей информацию об алгоритме **fzero** во время прогона.
- Области в структуре:
-
- **iterations** - число итераций.
- **nfev** – число вычислений функции.
- **bracketx** - двух элементный вектор, в координатах которого заключен нуль функции.
- **brackety** - двух элементный вектор, в координатах которого заключено значение функции.
-

- **optimset** *()*
- *options =* **optimset** *()*
- *options =* **optimset** *(par, val, …)*
- *options =* **optimset** *(old, par, val, …)*
- *options =* **optimset** *(old, new)*
-
- Создает структуру *options*.

Пример нахождения корня функции с помощью fzero

```
a=1;b=2;
x0=[-1,1];
f1=@(x)exp(-a*x)-b./x;
[x,y]=fzero(f1,x0)
x =   -3.2415e-16
y =  6170064136931106
```

```
[x,y,info,output]=fzero(f1,x0)
x =   -3.2415e-16
y =  6170064136931106
info = -5
output =

  scalar structure containing the fields:

    iterations =  88
    funcCount =  90
    bracketx =

      -3.2415e-16   1.6429e-18

    brackety =

       6.1701e+15  -1.2174e+18
```


@(x) exp (-a * x) - b ./ x
150
100
50
0
-50
-100
-150
-1
-0.5
0
0.5
1

- **fsolve** *(fcn, x0, options)*
- *[x, fvec, info, output, fjac]* = **fsolve** *(fcn, …)*
- Решает систему нелинейных уравнений определенных функцией fcn.
- fcn должно принять вектор (массив), определяющий неизвестные переменные и возвращать вектор левых сторон уравнений. Правосторонние стороны определены, чтобы быть нулями. Другими словами, эта функция пытается определять вектор x так что fcn (x), дает (приблизительно) все нули.
- x0 определяет начальное значение.

Решить систему уравнений

$$\begin{cases} -2x^2 + 3xy + 4\sin(y) - 6 = 0 \\ 3x^2 - 2xy^2 + 3\cos(x) + 4 = 0 \end{cases}$$

- function y = f 1(x)
- y = zeros (2, 1);
- y(1) = -2*x(1)^2 + 3*x(1)*x(2) + 4*sin(x(2)) - 6;
- y(2) = 3*x(1)^2 - 2*x(1)*x(2)^2 + 3*cos(x(1)) + 4;
- endfunction
- >> x=fsolve (@f1, [1; 2])
- x =

- 0.57983
- 2.54621