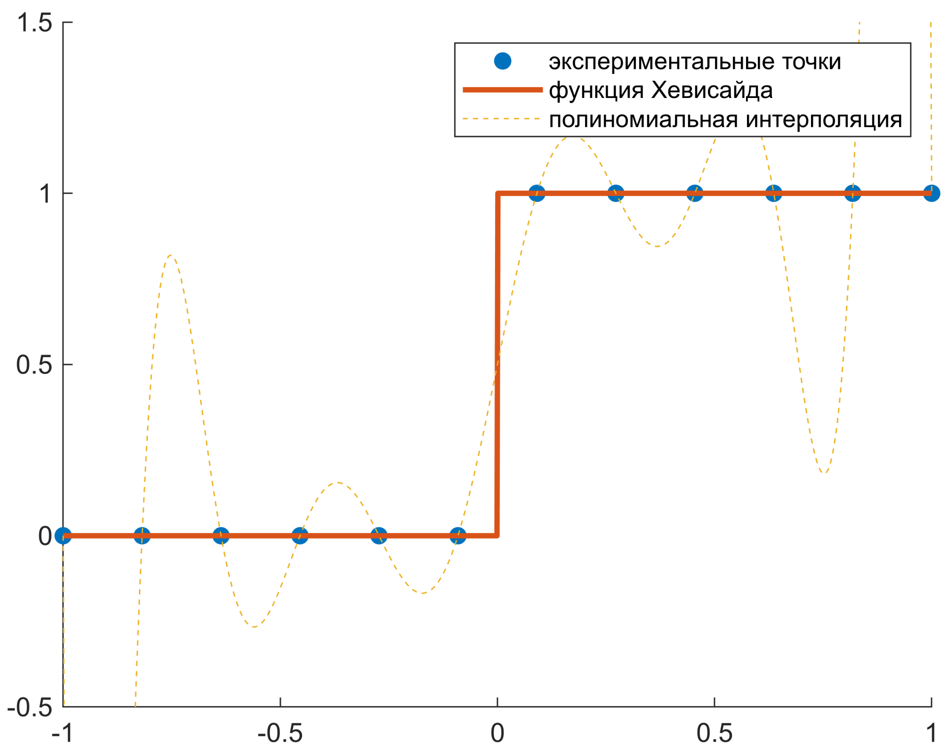
Семинар 19. Интерполяция. Сплайны

Пример (мотивация использовать сплайны)

```
N = 12; % меняем количество "экспериментальных точек"
x = linspace(-1,1,N);
y = heaviside(x);

%полиномиальная интерполяция
p = polyfit(x,y,N-1); % коэффициенты полиномов
yp = polyval(p,linspace(-1,1,1000)); % значение интерполяции


% Рисуем графики
scatter(x,y, DisplayName="экспериментальные точки", MarkerFaceColor="flat")
hold on
plot(linspace(-1,1,1000), heaviside(linspace(-1,1,1000)),DisplayName="функция
Хевисайда", LineWidth=2)
plot(linspace(-1,1,1000), yp, DisplayName="полиномиальная интерполяция",
LineStyle="--")
legend
ylim([-0.5,1.5])
hold off
```



Линейные сплайны

Для n экспериментальных точек $(i = 1, 2, 3, ..., n)$ у нас будет иметься $(n - 1)$ интервалов. У каждого интервала будет своя функция $s_i(x)$. Если мы имеем дело с линейными сплайнами, то функция

$$s_i(x) = a_i + b_i(x - x_i)$$

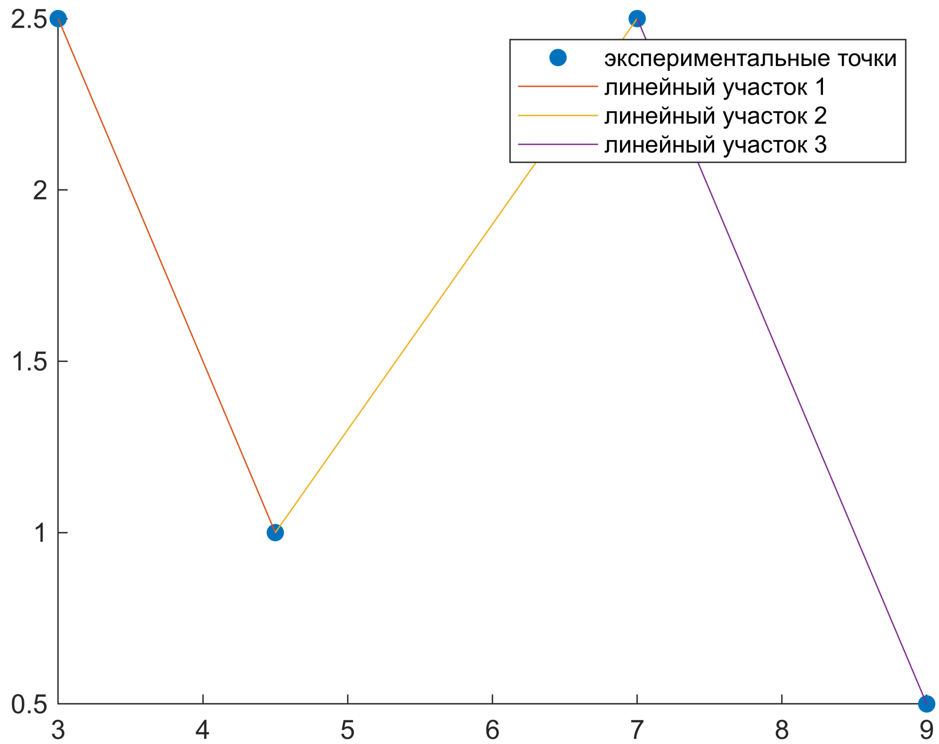
коэффициенты $a_i = f_i$ (пересечение)

коэффициенты $b_i = \frac{f_{i+1} - f_i}{x_{i+1} - x_i}$ (наклон)

следовательно вид функции будет выглядеть следующим образом

$$s_i(x) = a_i + b_i(x - x_i) = f_i + \frac{f_{i+1} - f_i}{x_{i+1} - x_i}(x - x_i)$$

```
% пример реализации линейных сплайнов
clearvars;
x = [3, 4.5, 7.0, 9.0];
y = [2.5, 1.0, 2.5, 0.5];

xp1 = linspace(3,4.5,100);
xp2 = linspace(4.5,7,100);
xp3 = linspace(7.0,9,100);

%построим s_i(x)
s1 = @(xp) y(1)+(y(2)-y(1))/(x(2)-x(1))*(xp-x(1));
s2 = @(xp) y(2)+(y(3)-y(2))/(x(3)-x(2))*(xp-x(2));
s3 = @(xp) y(3)+(y(4)-y(3))/(x(4)-x(3))*(xp-x(3));

%Нарисуем графики
scatter(x,y, MarkerFaceColor="flat", DisplayName="экспериментальные точки")
hold on
plot(xp1,s1(xp1), DisplayName="линейный участок 1")
plot(xp2,s2(xp2), DisplayName="линейный участок 2")
plot(xp3,s3(xp3), DisplayName="линейный участок 3")
legend
hold off
```



Минусы линейных сплайнов состоит в том, что итоговый сплайн не является гладким.

Квадратичные и кубические сплайны

Для простоты рассмотрим уравнения для квадратичных сплайнов

функция для интервала будет выглядеть следующим образом

$s_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2$

найдем коэффициенты.

Для n экспериментальных точек $(i = 1, 2, 3, ...n)$ у нас $n - 1$ интервал и $3(n - 1)$ коэффициентов (a,b,c). То есть нам надо составить $3(n - 1)$ уравнений

- функция должна проходить через все точки (условие непрерывности)

$f_i = a_i + b_i(x_i - x_i) + c_i(x_i - x_i)^2 = a_i$

$a_i = f_i$

Итого мы нашли $(n - 1)$ уравнений. Осталось $2(n - 1)$

- значения соседних функций должны быть равны в узлах

$f_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 = f_{i+1} + b_{i+1}(x_{i+1} - x_{i+1}) + c_{i+1}(x_{i+1} - x_{i+1})^2$

определим шаг $h_i = x_{i+1} - x_i$

перепишем и получим

$f_i + b_i h_i + c_i h_i^2 = f_{i+1}$

таких уравнений мы можем составить $n-1$ штук. Значит нам остается составить $(n-1)$ уравнений

- Первые производные должны быть на внутренних узлах должны быть одинаковы. Это важное условие для того, чтобы соседние сплайны объединялись гладким образом.

$s_i^{'} = b_i + 2c_i(x - x_i)$

По аналогии мы должны составить уравнения для соседних нод i и i+1

$b_i + 2c_i h_i = b_{i+1}$

Таких условий у нас $(n-2)$. То есть остается у нас $n - 1 - (n-2) = n - 1 + n + 2 = 1$. Одно условие

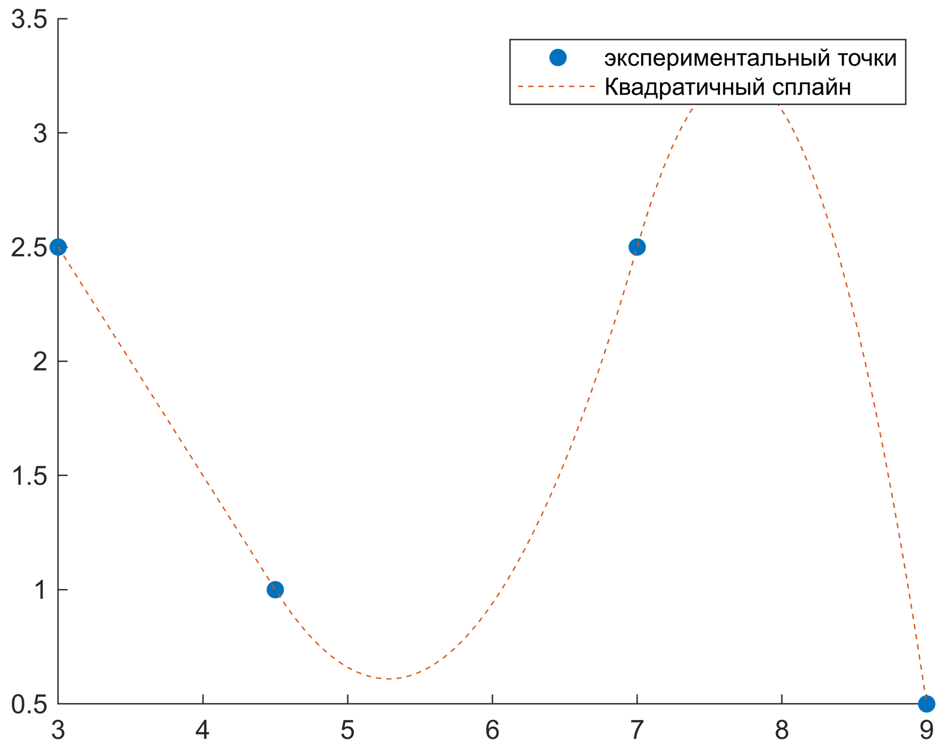
- Предположим, что вторая производная в первой точке равна нулю, получим

$s_i^{''} = 2c_i$

$s_1^{''} = 2c_1 = 0 \Rightarrow c_1 = 0$

Мы определили уравнения для всех коэффициентов.

```
clearvars

x = [3, 4.5, 7.0, 9.0];
y = [2.5, 1.0, 2.5, 0.5];
[coeffs, xplot, yplot] = spline2a(x,y);

scatter(x,y, MarkerFaceColor="flat", DisplayName="экспериментальный точки")
hold on
plot(xplot,yplot, LineStyle="--", DisplayName="Квадратичный сплайн")
legend
hold off
```



Квадратичные сплайны не так часто используются по разным причинам. Одна из них это возможные осциляции как при интерполяции полиномами высокой степени. Кубические сплайны во многом лучше, хоть и имеют особенности реализации

<u>Для кубического сплайна все тоже самое, просто немного сложнее</u>

Наше уравнение

$s_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$

У нас также n экспериментальных точек $(i = 1, 2, ...n)$. Число интервалов $n - 1$. Число уравнений, чтобы найти все коэффициенты $4(n - 1)$

Условия для кубического сплайна

Первое условие аналогично квадратичным сплайнам

- функция должна проходить через все точки (условие непрерывности)

$f(x_i) = a_i + b_i(x_i - x_i) + c_i(x_i - x_i)^2 + d_i(x_i - x_i)^3 = a_i$

$s_i(x) = f_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$

Остается составить 3(n-1) уравнений

- значения соседних функций должны быть равны в узлах

$$f_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = f_{i+1}$$

где $h_i = x_{i+1} - x_i$

- Первые производные должны быть на внутренних узлах должны быть одинаковы. Это важное условие для того, чтобы соседние сплайны объединялись гладким образом.
  $s_i'(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^3$ первая производная

равенство производных для двух интервалов в одном узле запишем так:

$$b_i + 2c_i h_i + 3d_i h_i^2 = b_{i+1}$$

- Вторые производные в узлах должны быть тоже равны

$s_i''(x) = 2c_i + 6d(x - x_i)$ вторая производная

равенство производных для двух интервалов в одном узле запишем так:

$$c_i + 3d_i h_i = c_{i+1}$$

отсюда найдем d_i

$$d_i = \frac{c_{i+1} - c_i}{3h_i}$$

Путем подстановок можно получить следующее уравнение

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = 3(f[x_{i+1}, x_i] - f[x_i, x_{i-1}])$$

Последнее уравнение может быть записано для внутренних узлов, $i = 1, 2, 3, ..., n - 2$ и можно найти коэффициенты $c_1, c_2, c_3 ... c_{n-1}$

Если у нас имеется еще **2 дополнительных условия**, то мы найдем остальные коэффициенты c. Дальше мы найдем коэффиенты b и d

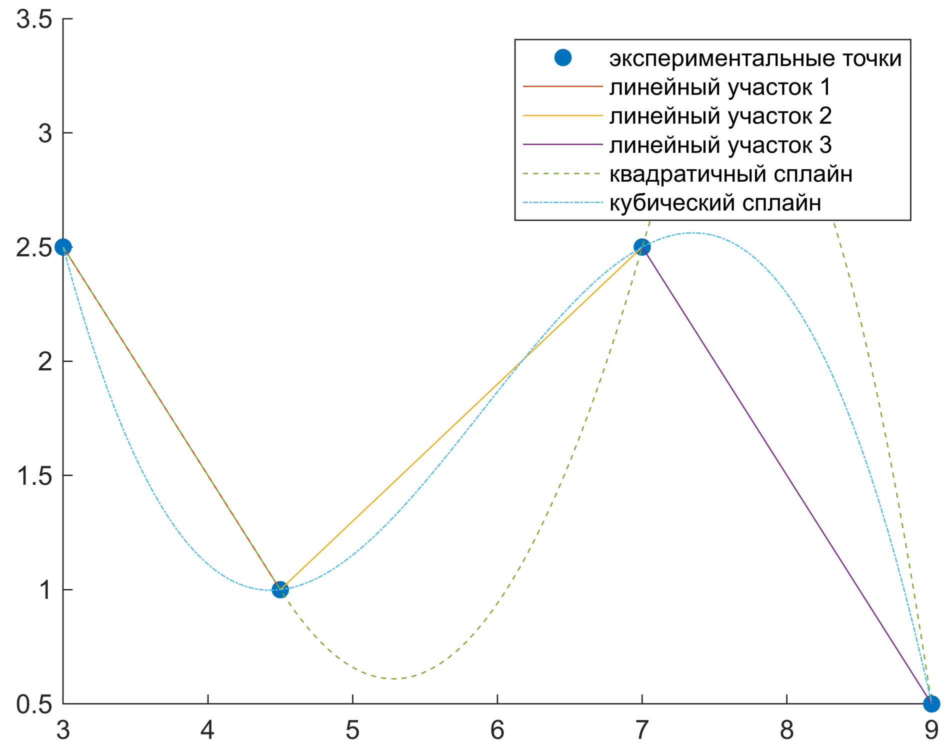
Выбор этих двух условий может быть различным и результирующий сплайн тоже будет различным

Условия	Первое и последнее уравнение	пояснение
natural	$c_1 = 0; c_n = 0$	
clamped	$2h_1c_1 + h_1c_2 = 3f_i[x_2, x_1] - 3f_1'$ 3$h_{n-1}c_{n-1} + 2h_{n-1}c_n = 3f_n' - f[x_n, x_{n-1}]$	указываем значение первых производных (в граничных узлах) Термин clamped значит зажатый (имеется в виду, что мы указываем конкретный наклон конечных сплайнов)

not-a-knot	$h_2c_1-(h_1+h_2)c_2+h_1c_3=0$ $h_{n-1}c_{n-2}-(h_{n-2}+h_{n-1})c_{n-1}+h_{n-2}c_n=0$	непрерывность третьих производных на втором и предпоследнем узле

Сравнение линейного квадратичного и кубического сплайнов

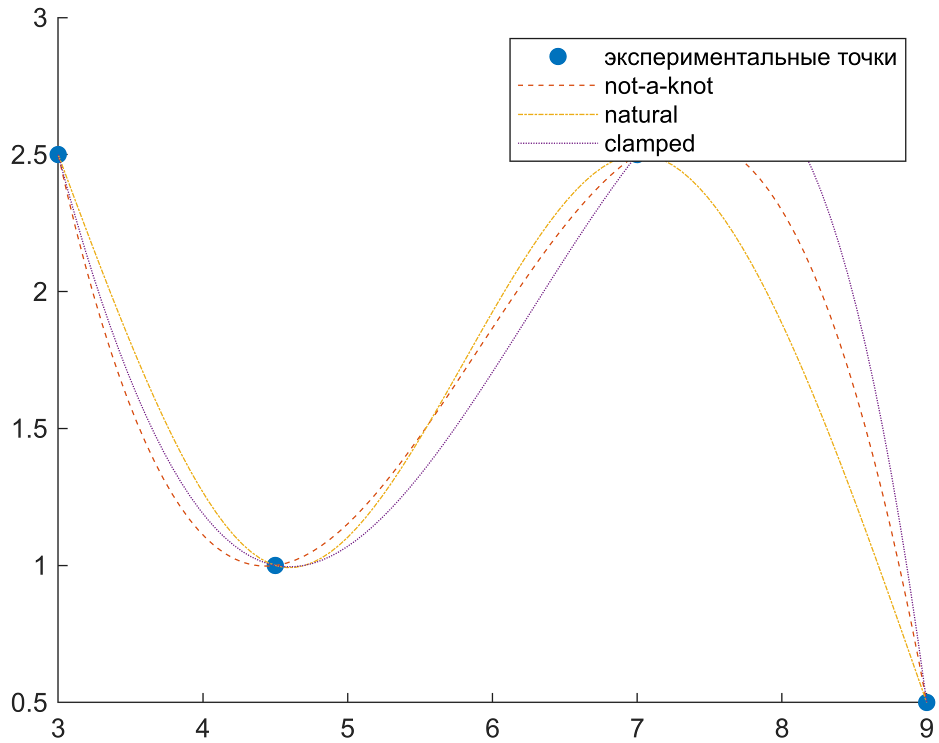
```
clearvars;
%Начальные точки
x = [3, 4.5, 7.0, 9.0];
y = [2.5, 1.0, 2.5, 0.5];

%линейные сплайны
xp1 = linspace(3,4.5,100); %интервал 1
xp2 = linspace(4.5,7,100); %интервал 2
xp3 = linspace(7.0,9,100); %интервал 3

%построим s_i(x)
s1 = @(xp) y(1)+(y(2)-y(1))/(x(2)-x(1))*(xp-x(1));
s2 = @(xp) y(2)+(y(3)-y(2))/(x(3)-x(2))*(xp-x(2));
s3 = @(xp) y(3)+(y(4)-y(3))/(x(4)-x(3))*(xp-x(3));

%Квадратичные сплайны
[coeffs, xplot, yplot] = spline2a(x,y);

%кубические сплайны
xx = linspace(x(1),x(end),100);
yy = spline(x,y, xx);


%Нарисуем графики

%Линейные сплайны
scatter(x,y, MarkerFaceColor="flat", DisplayName="экспериментальные точки")
hold on
plot(xp1,s1(xp1), DisplayName="линейный участок 1")
plot(xp2,s2(xp2), DisplayName="линейный участок 2")
plot(xp3,s3(xp3), DisplayName="линейный участок 3")

%квадратичный сплайн
plot(xplot,yplot, LineStyle="--", DisplayName="квадратичный сплайн")
%кубический сплайн
plot(xx,yy, LineStyle="-.", DisplayName="кубический сплайн")

legend
hold off
```



Сравнение кубических сплайнов

```
clearvars;
%Начальные точки
x = [3, 4.5, 7.0, 9.0];
y = [2.5, 1.0, 2.5, 0.5];
xx = linspace(x(1),x(end),100);

scatter(x,y, MarkerFaceColor="flat", DisplayName="экспериментальные точки")
hold on

%not a knot
yy_knot = spline(x,y, xx);
plot(xx,yy_knot, DisplayName="not-a-knot", LineStyle="--")

pp = csape(x,y, "variational");
yy_natural = fnval(pp,xx);
plot(xx,yy_natural, DisplayName="natural", LineStyle="-.")

% clamped
x = [3, 4.5, 7.0, 9.0];
y = [-2, 2.5, 1.0, 2.5, 0.5, -3.9]; % Принимаются во внимание дополнительные точки
yy_clamped = spline(x,y, xx);
plot(xx,yy_clamped, DisplayName="clamped", LineStyle=":")
legend
```

```
hold off
```



Многомерная интерполяция

билинейная интерполяциия

В двумерном случае у нас есть функция $z = (x_i, y_i)$

Мы хотим найти промежуточные значения функции, зная известные точки (узлы)

Допустим нам известны 4 точки x и соотвтествующие значения функции x:
$f(x_1, y_1), f(x_2, y_1), f(x_2, y_1), f(x_2, y_2)$

Суть заключается в том, чтобы сначала "пройтись по иксам" (линейная интерполяция вдоль направления x), а затем полученные значения использовать для направления y и в конечном счете можно получить итоговую функцию

Запишем уравнения используя форму Лагранжа

$f(x_i, y_1) = \frac{x_i - x_2}{x_1 - x_2} f(x_1, y_1) + \frac{x_i - x_1}{x_2 - x_1} f(x_2, y_1)$ уравнение для (x_i, y_1)

$f(x_i, y_1) = \frac{x_i - x_2}{x_1 - x_2} f(x_1, y_2) + \frac{x_i - x_1}{x_2 - x_1} f(x_2, y_2)$ уравнение для (x_i, y_2)

Эти точки используем для линейной интерполяции вдоль y, чтобы получить финальный результат

$$f(x_i, y_i) = \frac{y_i - y_2}{y_1 - y_2} f(x_i, y_1) + \frac{y_i - y_1}{y_2 - y_1} f(x_i, y_2)$$

Итоговое уравнение

$$f(x_i, y_i) = \frac{x_i - x_2}{x_1 - x_2}\frac{y_i - y_2}{y_1 - y_2} f(x_1, y_1) + \frac{x_i - x_1}{x_2 - x_1}\frac{y_i - y_2}{y_1 - y_2} f(x_1, y_1) + \frac{x_i - x_2}{x_1 - x_2}\frac{y_i - y_1}{y_2 - y_1} f(x_1, y_2) + \frac{x_i - x_1}{x_1 - x_2}\frac{y_i - y_1}{y_2 - y_1} f(x_2, y_2)$$

реализуется функцией interp2

Использование готовых функций для интерполяции

функция spline

Cubic spline data interpolation. (только кубическая интерполяция)

два "режима" работы

мы можем передать либо 2 либо 3 аргумента

1. Мы можем передать 3 аргумента (известное значение **x**, известное значение **y**, **новое** значение **x**). В качестве известных значений необходимо передать вектора. В качестве нового значения можно передать конкретную точку, для расчета функции в этой точке, либо передать вектор, и результат работы функции будет вектор новых значений y
2. Мы можем передать 2 аргумента (известное значение **x**, известное значение **y**). Результатом работы функции будет некая структура. Получить данные y из этой структуры можно с помощью функции fnval(pp,xx) (pp это наша структура)

Также, если length(y)>length(x) на 2 то тогда будут применяться clamped condition. Если вектора равной длины, то not-a-knot condition

Также существуют другие способы кубической интерполяции.

```
%Пример из хелпа
clearvars
x = [1 2 3 4 5 5.5 7 8 9 9.5 10];
y = [0 0 0 0.5 0.4 1.2 1.2 0.1 0 0.3 0.6];
xq = 0.75:0.05:10.25;
yqs = spline(x,y,xq);
yqp = pchip(x,y,xq);
yqm = makima(x,y,xq);
figure
plot(x,y,'ko','LineWidth',2,'MarkerSize',10)
hold on
plot(xq,yqp,'LineWidth',4)
plot(xq,yqs,xq,yqm,'LineWidth',2)
legend('(x,y) data','pchip','spline','makima')
hold off
```

Если функция spline (makima pchip) по сути разные варианты кубической интерполяции, то функция interp1 позволяет провести интерполяцию и другими функциями 'linear'   - (default) linear interpolation

'nearest' - nearest neighbor interpolation

'next'   - next neighbor interpolation

'previous' - previous neighbor interpolation

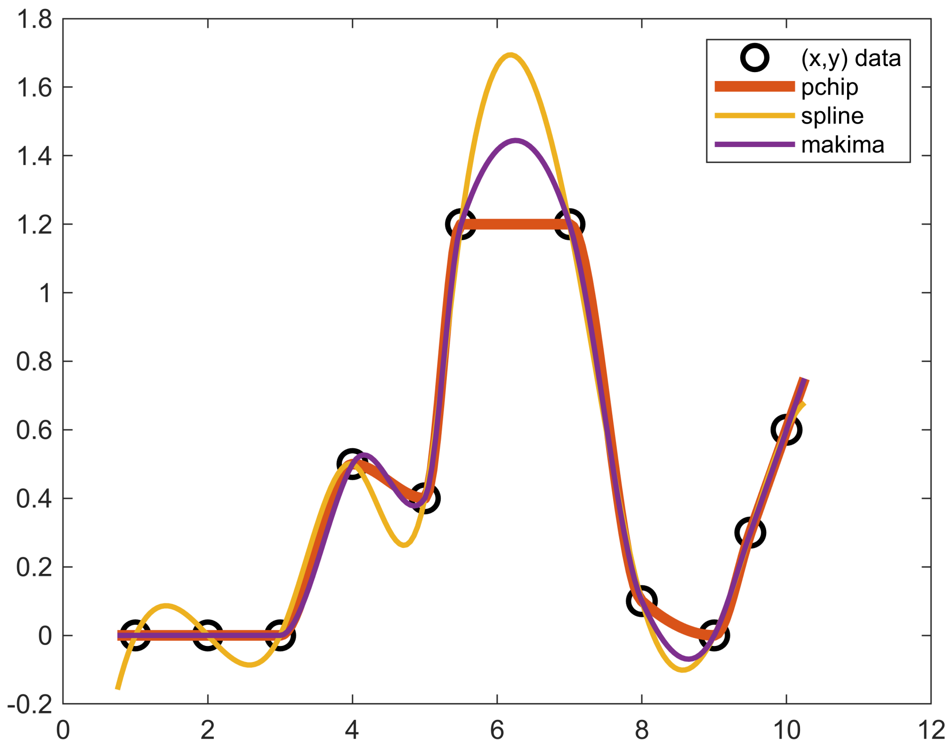
'spline'   - piecewise cubic spline interpolation (SPLINE)

'pchip'   - shape-preserving piecewise cubic interpolation

'cubic'   - cubic convolution interpolation for uniformly-spaced data. This method does not extrapolate and falls back to 'spline' interpolation for irregularly-spaced data. NOTE: 'cubic' changed in R2020b to perform cubic convolution. In previous releases, 'cubic' was the same as 'pchip'. v5cubic' - same as 'cubic' 'makima'   - modified Akima cubic interpolation

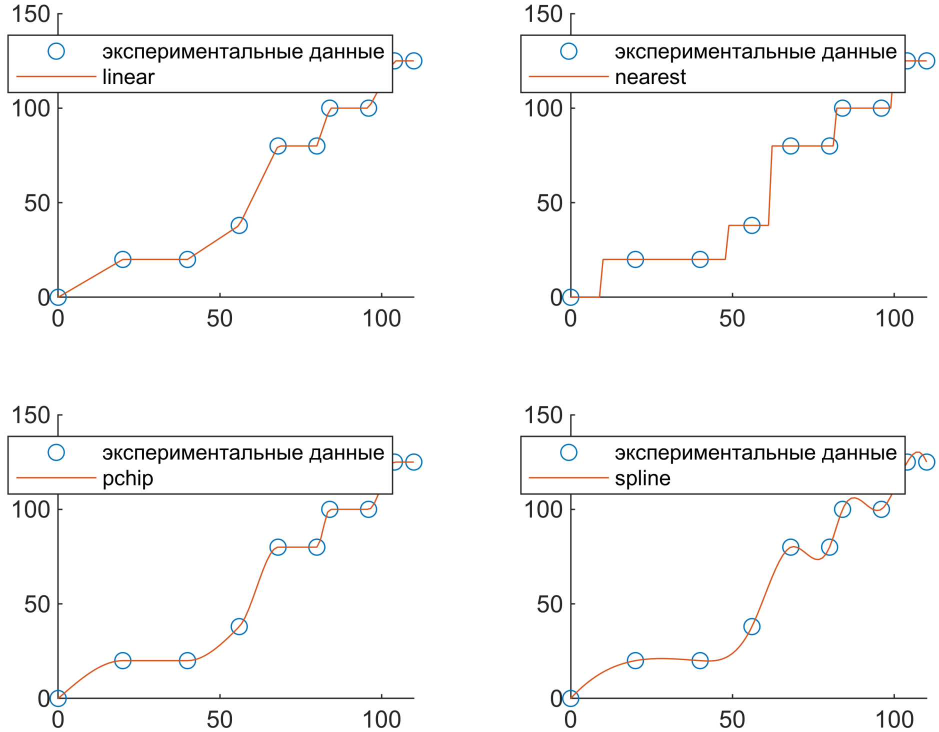
```
clearvars
%Пример
t = [0,20, 40, 56, 68, 80, 84, 96, 104, 110];
v = [0, 20, 20, 38, 80, 80, 100, 100, 125, 125];
tt = linspace(0,110);
vl = interp1(t,v,tt);
vn = interp1(t,v,tt, 'nearest');
vh = interp1(t,v,tt, 'pchip');
vs = interp1(t,v,tt, 'spline');
```

```
tiledlayout(2,2)
```




```
%tile1
nexttile
scatter(t,v,DisplayName="экспериментальные данные")
hold on
plot(tt,vl, DisplayName="linear");
legend
hold off

%tile2
nexttile
scatter(t,v,DisplayName="экспериментальные данные")
hold on
plot(tt,vn, DisplayName="nearest");
legend
hold off

%tile3
nexttile
scatter(t,v,DisplayName="экспериментальные данные")
hold on
plot(tt,vh, DisplayName="pchip");
legend
hold off

%tile4
nexttile
```

```
scatter(t,v,DisplayName="экспериментальные данные")
hold on
plot(tt,vs, DisplayName="spline");
legend
hold off
```



Работа с pp-структурой

Существуют несколько удобных функций, которые позволяют работать с pp-структурой, которая получается при вызове функции spline (и некоторых других)

v = fnval(pp,x)	вычисляет значение
dpp = fnder(pp)	производит дифференцирование (возвращается объект pp)
dirpp = fndir(pp, dir)	производная вдоль направления
ipp = fnint(pp)	интегрирование
fnmin(pp,[a,b])	находит минимум на промежутке a,b
fnplt(pp,[a,b])	строит график на промежутке a,b

```
% пример
clearvars
t = [0,20., 40, 56, 68, 80, 84, 96, 104, 110];
```

```
v = [0, 20, 20, 38, 80, 80, 100, 100, 125, 125];
pp = spline(t,v)
```

```
pp = struct with fields:
      form: 'pp'
    breaks: [0 20 40 56 68 80 84 96 104 110]
     coefs: [9×4 double]
    pieces: 9
     order: 4
       dim: 1
```

```
dpp = fnder(pp)
```

```
dpp = struct with fields:
      form: 'pp'
    breaks: [0 20 40 56 68 80 84 96 104 110]
     coefs: [9×3 double]
    pieces: 9
     order: 3
       dim: 1
```

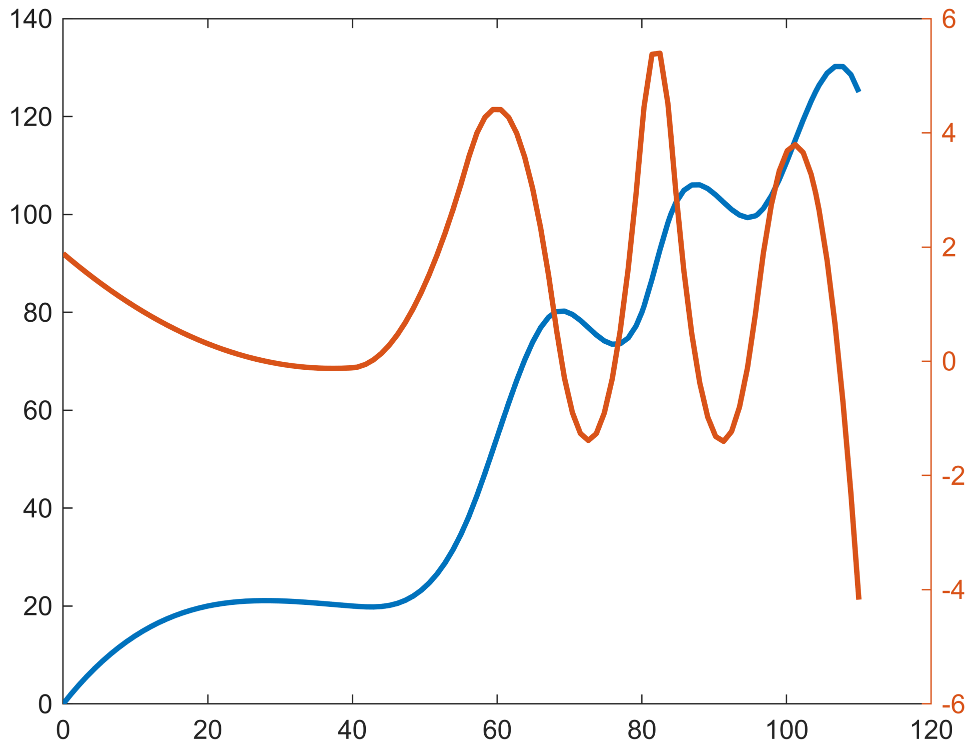
```
ipp = fnint(pp)
```

```
ipp = struct with fields:
      form: 'pp'
    breaks: [0 20 40 56 68 80 84 96 104 110]
     coefs: [9×5 double]
    pieces: 9
     order: 5
       dim: 1
```

```
fnmin(pp,[20,40])
```

```
ans = 20
```

```
figure
fnplt(pp,[0,110])
yyaxis right
fnplt(dpp,[0,110])
```



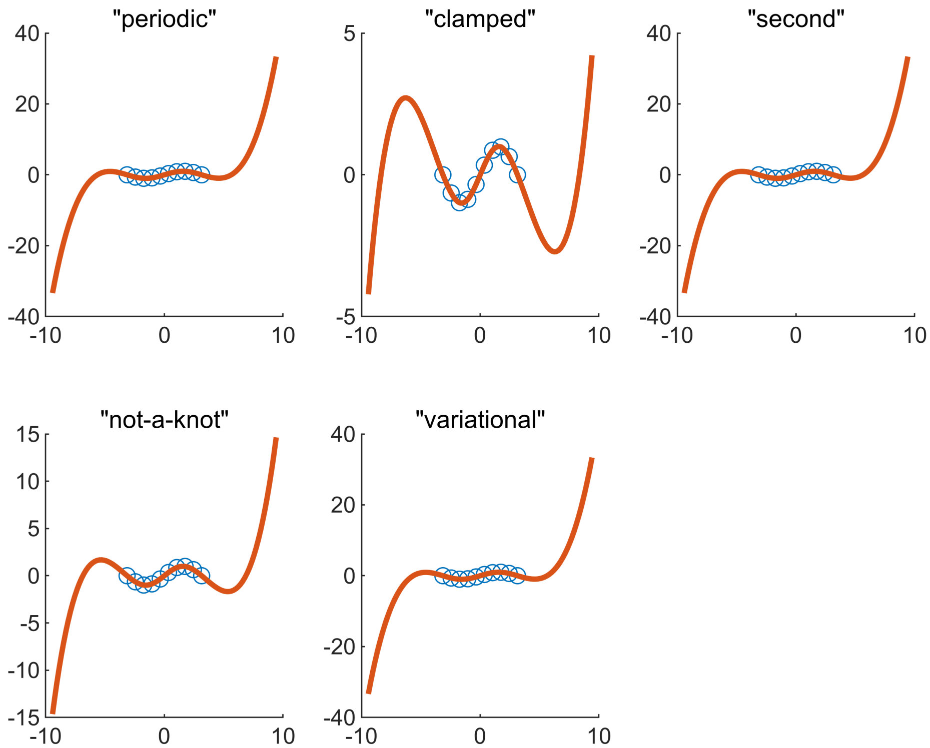
Немного про функцию csape

Она позволяет создавать кубические сплайны с различными конечными условиями. В документации достаточно написано про эту функцию.

Пример с периодическими конечными условиями

```
% Пример
clearvars
t = linspace(-pi,pi,10);
v = sin(t);
Diapason = [-3*pi,3*pi];
ppp = csape(t,v, "periodic");
ppc = csape(t,v, "clamped");
pps = csape(t,v, "second");
ppv = csape(t,v, "variational");
ppk = csape(t,v, "not-a-knot");
figure
tiledlayout(2,3)
%tile 2
nexttile
scatter(t,v,DisplayName="экспериментальные данные")
hold on
fnplt(ppp,Diapason)
hold off
title "periodic"
```

```
%tile 2
nexttile
scatter(t,v,DisplayName="экспериментальные данные")
hold on
fnplt(ppc,Diapason)
hold off
title "clamped"

%tile 3
nexttile
scatter(t,v,DisplayName="экспериментальные данные")
hold on
fnplt(pps,Diapason)
hold off
title "second"

%tile 4
nexttile
scatter(t,v,DisplayName="экспериментальные данные")
hold on
fnplt(ppk,Diapason)
hold off
title "not-a-knot"

%tile 5
nexttile
scatter(t,v,DisplayName="экспериментальные данные")
hold on
fnplt(ppv,Diapason)
hold off
title "variational"
```


"periodic"
"clamped"
"second"
"not-a-knot"
"variational"
40
20
0
-20
-40
5
0
-5
15
10
5
0
-5
-10
-15
-10
0
10