სეკანტის მეთოდი

ნიუტონის მეთოდის მინუსი: ფუნქციის ანალიზური წარმოებულის არსებობა (ცოდნა).

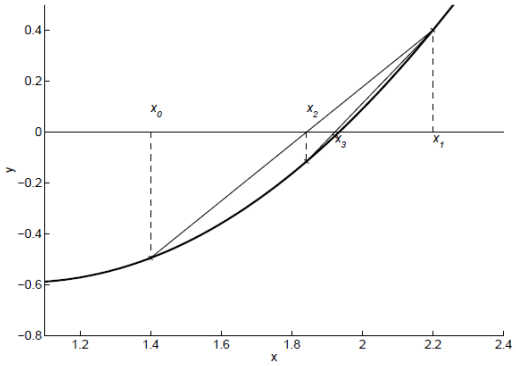
შემთხვევა: *ფუნქციის წარმოებული არ ვიცით, ან მისი გამოთვლა შესაძლებელია მნიშვნელოვანი გამოთლითი რესურსის დანახარჯით.*

სეკანტის მეთოდში წარმოებულის შეფასება ხდება დისკრეტულად:

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

$$x_{n+1} = x_n + h_n,$$

$$h_n = -f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})},$$



მაგალითი:

$$f(x) = (x/2)^2 - \sin x = 0$$

$$x_0 = 1.5, x_1 = 2.$$

n	x_n	$f(x_n)$	h_n
0	1.5	−0.434994986604	
1	2.0	+0.090702573174	−0.086268 778965
2	**1.91**3731 221035	−0.026180060742	+0.019322 989205
3	**1.933**054 210240	−0.000924399645	+0.000707 253882
4	**1.933761** 464122	+0.000010180519	−0.000007 704220
5	**1.933753 759**902	−0.000000003867	+0.000000 002925
6	**1.933753 762827**		

სკალარული ფუნქციის მინიმუმის პოვნა

ერთგანზომილებიანი მინიმიზაციის პრობლემა (ოპტიმიზაციის ამოცანები).
თუკი ფუნქცია დიფერენცირებადია მინიმუმში:

$$f'(x^*) = 0.$$

ფუნქციის წარმოებულის ნულთან ტოლობა (მეორე წარმოებული დადებითია, ან საზღვრები).

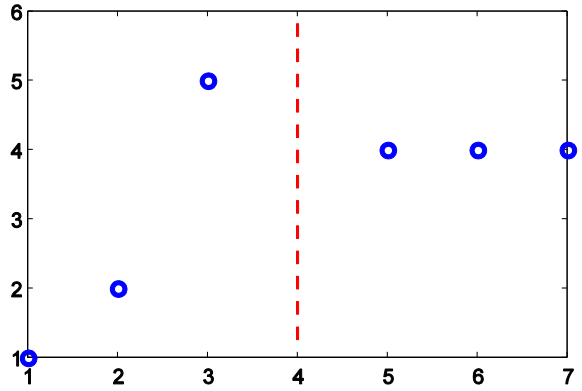
ფუნქციის ლოკალური მინიმუმები.

რიცხვითი ინტერპოლაცია

არარსებული ინფორმაციის აღდგენა სხვა მოსაზრებებიდან.
(ფუნქციის უწყვეტობა, გლუვობა, ფორმა, ტრაექტორის ჩაკეტილობა ...)

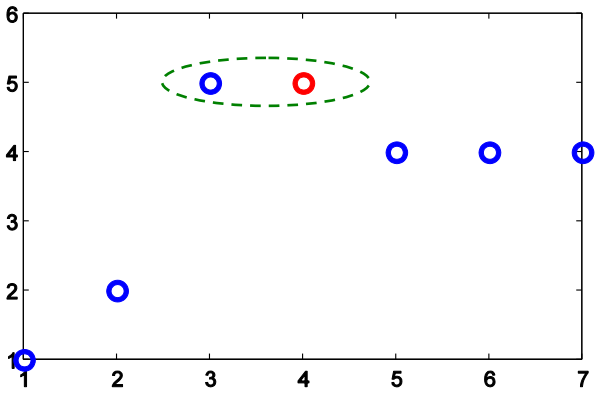
პირდაპირი (გულუბრყვილო) ინტერპოლაციის მეთოდი:

F(1) = 1;
F(2) = 2;
F(3) = 3;
F(4) = . . .
F(5) = 4;
F(6) = 4;
F(7) = 4;



ნულოვანი რიგის ინტერპოლაცია

F(k) = F(k-1);



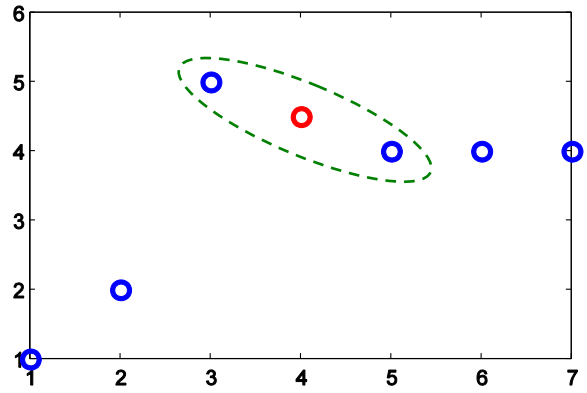
1 რიგის ინტერპოლაცია

Piecewise Linear interpolation:

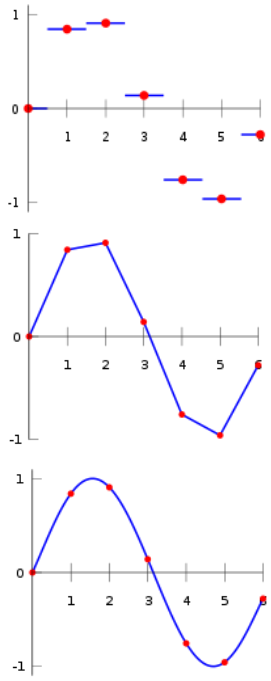
F(k-1) = a*x(k-1) + b
F(k+1) = a*x(k+1) + b

Find: a,b;

F(k) = a*X(k) +b



მაღალი რიგის ინტერპოლაცია



Piecewise Quadratic interpolation:

F(k) = a*x(k)^2 + bx(k) + c

Find: a,b,c

Forward scheme: F(k-2), F(k-1), F(k+1)
Backward scheme: F(k-1), F(k+1), F(k+2)

Check:

polar

დავალება

```
phi = (0:pi/39.5:2*pi);
omega = 4;
```

*F(phi) = 2 + sin(omega*phi - pi/8);*

F1(phi) = F'(phi) 2 point central derivative
F2(phi) = F'(phi) 5 point derivative

L1(1:40) = (Interpolate missing pixels in F1, 0th order interpolation)
L2(1:40) = (Interpolate missing pixels in F2, 0th order interpolation)

G1(1:40) = (Interpolate missing pixels in F1, linear interpolation)
G2(1:40) = (Interpolate missing pixels in F2, linear interpolation)

Polar plot: L1, L2
Polar plot: G1, G2

H1(phi) = F''(phi) 2 point central derivative
H2(phi) = F''(phi) 5 point derivative

Polar plot: H1, H2

ინტერპოლაცია ლაგრანჟის პოლინომებით

ინტერპოლაციის მიზანია K წერტილიდან აღვადგინოთ N წერტილი, სადაც N >> K (!).

მეთოდი: აღვადგინოთ ანალიზური ფუნქცია (fit), ანალიზური ფუნქციიდან აღვადგინოთ N დისკრეტული წერტილი.

მოცემულია k+1 წერტილი რომელთაგანაც არცერთი არ მეორდება:

$$(x_0, y_0), \ldots, (x_j, y_j), \ldots, (x_k, y_k)$$

ლაგრანჟის პოლინომი:

$$L(x) := \sum_{j=0}^{k} y_j \ell_j(x)$$

ლაგრანჟის საბაზისო პოლინომები:

$$\ell_j(x) := \prod_{\substack{0 \le m \le k \\ m \ne j}} \frac{x - x_m}{x_j - x_m} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)},$$

წერტილები არ მეორდება: $x_j - x_m \neq 0$

<u>მაგალითი 1:</u> f(x) = x^2;

$$\begin{array}{ll} x_0 = 1 & f(x_0) = 1 \\ x_1 = 2 & f(x_1) = 4 \\ x_2 = 3 & f(x_2) = 9. \end{array} \quad L(x) = 1 \cdot \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} + 4 \cdot \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} + 9 \cdot \frac{x-1}{3-1} \cdot \frac{x-2}{3-2} = x^2.$$

<u>მაგალითი 2:</u> f(x) = x^3;

$$\begin{array}{ll} x_0 = 1 & f(x_0) = 1 \\ x_1 = 2 & f(x_1) = 8 \\ x_2 = 3 & f(x_2) = 27 \end{array} \quad L(x) = 1 \cdot \frac{x-2}{1-2} \cdot \frac{x-3}{1-3} + 8 \cdot \frac{x-1}{2-1} \cdot \frac{x-3}{2-3} + 27 \cdot \frac{x-1}{3-1} \cdot \frac{x-2}{3-2} = 6x^2 - 11x + 6.$$

ლაგრანჟის ინტერპოლაციის რეალიზაცია მატლაბში

$$P(x_k) = y_k, \;\; k = 1, \ldots, n.$$

$$P(x) = \sum_k \left(\prod_{j \neq k} \frac{x - x_j}{x_k - x_j} \right) y_k.$$

$$P(x) = c_1 x^{n-1} + c_2 x^{n-2} + \cdots + c_{n-1} x + c_n,$$

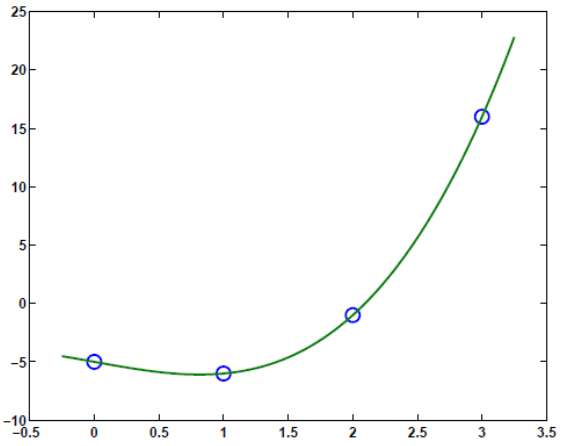
ამოცანის ანალიზური ფორმა:

$$\begin{pmatrix} x_1^{n-1} & x_1^{n-2} & \cdots & x_1 & 1 \\ x_2^{n-1} & x_2^{n-2} & \cdots & x_2 & 1 \\ \cdots & \cdots & \cdots & \cdots & 1 \\ x_n^{n-1} & x_n^{n-2} & \cdots & x_n & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

```
function v = polyinterp(x,y,u)

Nx = length(x);
Nu = length(u);

v = zeros(1,Nu);

for k = 1:Nx
     w = ones(1,Nu);
     for j = [1:k-1 k+1:Nx]
          w = (u-x(j))./(x(k)-x(j)).*w;
          end
          v = v + w*y(k);
end
```

გამოყენება:

```
v = polyinterp(x,y,u);
plot(x,y,'o',u,v,'-');
```

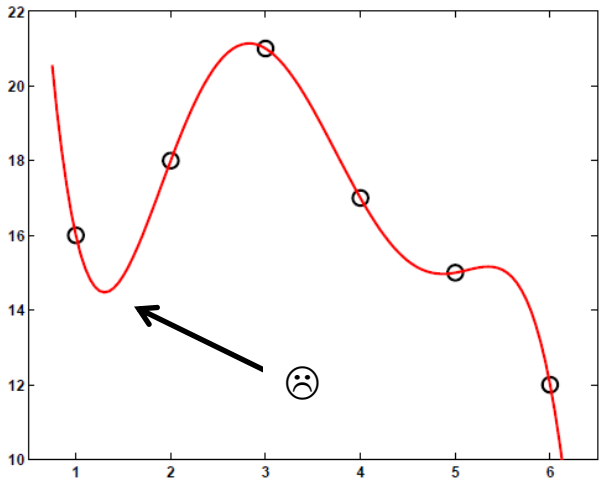


დავალება

`Polyinterp` ფუნქციის გამოყენებით მოახდინეთ `(-1,1)` ინტერვალში 5 თქვენს მიერ შერჩეული წერტილიდან ლაგრანჟის პოლინომიალური ინტერპოლაციავექტორში: `u = (-1:0.1:1);`

<u>უწყვეტი ინტერპოლაციის პრობლემები - მაგალითი:</u>

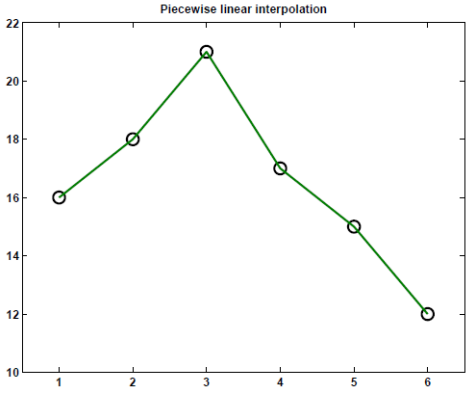
```
x = 1:6;
y = [16 18 21 17 15 12];

1     2     3     4     5     6
16    18    21    17    15    12

u = .75:.05:6.25;
v = polyinterp(x,y,u);
plot(x,y,'o',u,v,'r-');
```



უბან-უბან წრფივი ინტერპოლაცია

ნაწყვეტებისაგან შედგენილი ფუნქცია - spline

ნაწყვეტ-ნაწყვეტ ანალიზური ფუნქცია, რომლითაც შესაძლებელია ინტე რვალში ნებისმიერი წერტილის აღდგენა;



$$x_k \le x < x_{k+1}$$

$$L(x) = y_k + (x - x_k)\frac{y_{k+1} - y_k}{x_{k+1} - x_k}$$

უბან-უბან კუბური ერმიტული ინტერპოლაცია (pchip)

ინტერპოლანტი წერტილები: (x_k, y_k)

უბან-უბან კუბური ინტერპოლაციური ფუნქცია: $y = P(x)$

ინტერვალი ორ მოცემულ წერტილს შორის:

$$h_k = x_{k+1} - x_k.$$

ცვლადი ინტერვალს შიგნით:

$$s = x - x_k$$

ინტერპოლანტის დახრის კუთხის შეფასება:

$$d_k = P'(x_k).$$

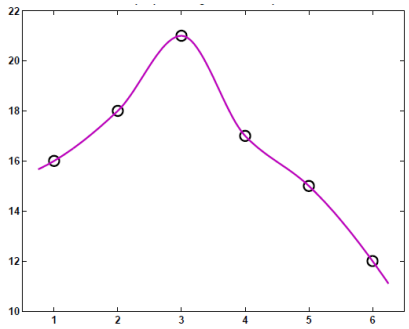
ორ ინტერპოლანტ წერტილს შორის ინტერვალში კუბური პოლინომიალი (spline):

$$P(x) = \frac{3hs^2 - 2s^3}{h^3}y_{k+1} + \frac{h^3 - 3hs^2 + 2s^3}{h^3}y_k + \frac{s^2(s-h)}{h^2}d_{k+1} + \frac{s(s-h)^2}{h^2}d_k.$$

ინტერპოლაციური ფუნქციის მნიშვნელობა ემთხვევა ინტერპოლანტ წერტილების მნიშვნელობებს:

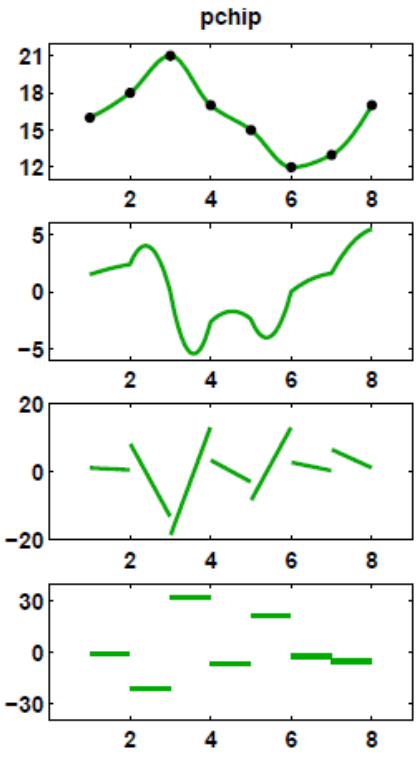
$$P(x_k) = y_k,\ P(x_{k+1}) = y_{k+1},$$
$$P'(x_k) = d_k,\ P'(x_{k+1}) = d_{k+1}.$$

არის კუბური პოლინომი (s-ის, ანუ x-ის მიმართ).



შესაძლებელია სხვადასხვა იმპლემენტაცია. მიდგომები: დახრის კუთხის შეფასება დისკრეტულ ინტერპოლანტში (d_k).

$$P(x) = y_k + sd_k + s^2c_k + s^3b_k,$$
$$c_k = \frac{3\delta_k - 2d_k - d_{k+1}}{h},$$
$$b_k = \frac{d_k - 2\delta_k + d_{k+1}}{h^2}.$$



pchip და მისი სამი ანალიზური წარმოებული.

დავალება:

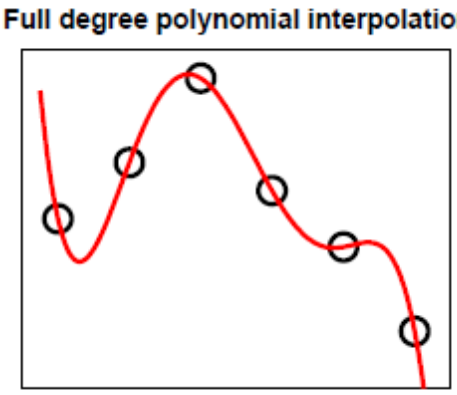
შეადარეთ ნახაზზე მოყვანილ წერტილებზე უბან-უბან კუბური ინტერპოლაციის 1,2 და 3 რიგის ანალიზური წარმოებულები კუბური ინტერპოლაციური ფუნქციის რიცხვით ცენტრალურილური წარმოებულის მეთოდით აღებულ 3 წარმოებულს.

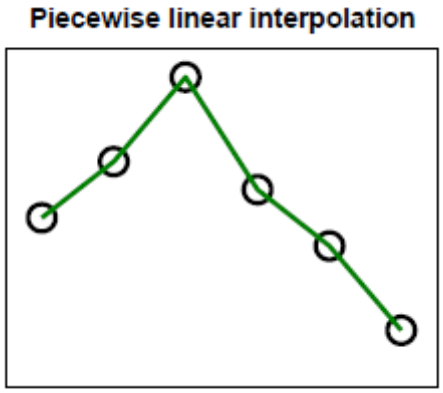
```
function v = pchiptx(x,y,u)
%PCHIPTX piecewise cubic Hermite interpolation.

h = diff(x);
delta = diff(y)./h;
d = pchipslopes(h,delta);

% Piecewise polynomial coefficients
n = length(x);
c = (3*delta - 2*d(1:n-1) - d(2:n))./h;
b = (d(1:n-1) - 2*delta + d(2:n))./h.^2;

% Find subinterval indices k so that x(k) <= u < x(k+1)
k = ones(size(u));
for j = 2:n-1
  k(x(j) <= u) = j;
end

% Evaluate interpolant
s = u - x(k);
v = y(k) + s.*(d(k) + s.*(c(k) + s.*b(k)));
```

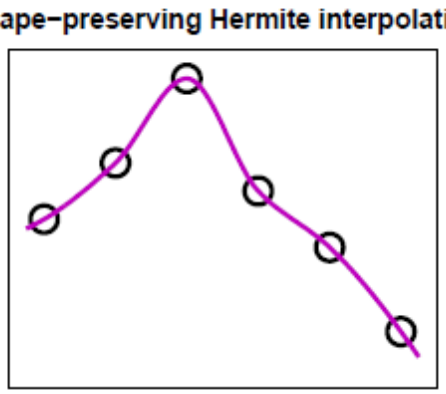
```
function d = pchipslopes(h,delta)
% PCHIPSLOPES Slopes for shape-preserving Hermite cubic
% pchipslopes(h,delta) computes d(k) = P'(x(k)).

% Slopes at interior points
% delta = diff(y)./diff(x).
% d(k) = 0 if delta(k-1) and delta(k) have opposites
% signs or either is zero.
% d(k) = weighted harmonic mean of delta(k-1) and
% delta(k) if they have the same sign.

n = length(h)+1;
d = zeros(size(h));
k = find(sign(delta(1:n-2)).*sign(delta(2:n-1))>0)+1;
w1 = 2*h(k)+h(k-1);
w2 = h(k)+2*h(k-1);
d(k) = (w1+w2)./(w1./delta(k-1) + w2./delta(k));

% Slopes at endpoints
d(1) = pchipend(h(1),h(2),delta(1),delta(2));
d(n) = pchipend(h(n-1),h(n-2),delta(n-1),delta(n-2));
```

```
function d = pchipend(h1,h2,del1,del2)
% Noncentered, shape-preserving, three-point formula.

d = ((2*h1+h2)*del1 - h1*del2)/(h1+h2);

if sign(d) ~= sign(del1)
  d = 0;
elseif (sign(del1)~=sign(del2))&(abs(d)>abs(3*del1))
  d = 3*del1;
end
```

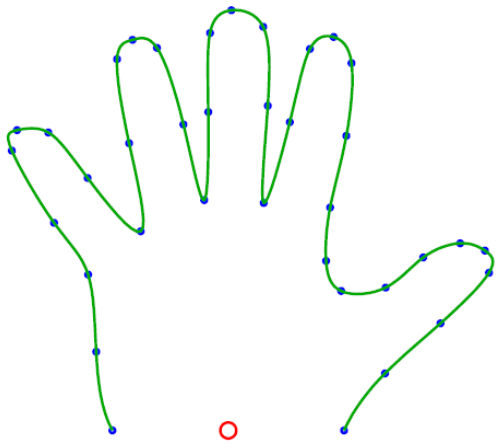
6 წერტილიანი ინტერპოლანტის ინტერპოლაციის შედეგები







დავალება:



ააგეთ თქვენი ხელის ფუნქცია კუბური ინტერპოლაციის გამოყენებით. გამოიყენეთ მაქსიმუმ 40 საინტერპოლაციო წერტილი. მიღებული მრუდი უნდა შეიცავდეს 1000 წერტილს.

გამოიყენეთ პარამეტრული ინტერპოლაცია:
x=x(t), y=y(t),

Matlab-ში წერტილების ინტერაქტიული შეყვანის კოდი:

```
figure('position',get(0,'screensize'));
axes('position',[0 0 1 1]);
[x,y] = ginput;
```