Spline interpolation with MATLAB function MAKIMA

17250099 HuZhixiang

Introduction

Interpolation refers to getting a curve that can pass through all the given data. It is a good way to manually draw a primitive curve if you have enough professional knowledge. However, it would take so much time on manual drawing, so it is significant to provide some algorithms for computers to interpolate the given points.

There are various methods of interpolation of measurement data, such as linear interpolation, polynomial interpolation, spline interpolation, pchip interpolation, makima interpolation, akima interpolation, etc. In the mathematical field of numerical analysis, spline interpolation is a form of interpolation using a special piecewise polynomial called spline.

Linear interpolation only considers the influence of two points nearby. In polynomial interpolation, the low-order polynomial has fewer parameters, and the interpolation accuracy is very low, and the use of high-order polynomials will make the solution unstable, and the phenomenon of "Runge" appears, that is, the interpolation function is consistent with the actual data at the interpolation point, however, there is a large deviation outside the interpolation point.

Therefore, people have developed piecewise polynomials based on polynomials, namely spline interpolation. Spline interpolation refers to drawing part of the curve by relative points. Spline interpolation maintains the simple characteristics of polynomial operations, and avoids the shortcomings of numerical instability when the polynomial order is higher, so it has been widely used. But in spline function interpolation, to determine the polynomial on any one area, we must consider the influence of all data points on it. This

not only expands the scope of error propagation but also increases a lot of work. Sometimes only a few data points near the interpolation point are used as control points for interpolation. However, the MAKIMA interpolation method here has unique advantages. The MAKIMA interpolation method considers the effect of the derivative value of the element like the cubic spline function, so the entire interpolation curve obtained is smooth. Cubic spline function interpolation has minimum modulus characteristics, best optimal approximation, and convergence, while the curve obtained by MAKIMA interpolation method is smoother and more natural than spline.

Modified AKIMA piecewise cubic Hermitian interpolation

Actually, MAKIMA is short for modified AKIMA piecewise cubic Hermitian interpolation. It refers to a specific MATLAB modification of AKIMA's derivative formula.

It has the following three main properties:

- 1. It produces undulations which are between 'spline' and 'pchip'.
- 2. The MAKIMA interpolation algorithm solved the negligible difference between slopes of interval if the lower and the higher slopes of line are both equal and di when the lower and the higher slopes are nearly equal of AKIMA.
- 3. The MAKIMA interpolation decreases the undulation compared with AKIMA interpolation if the lower and the higher slopes are both equal ($\delta_{i-2}=\delta_{i-1}$, $\delta_i=\delta_{i+1}$) and the higher slopes are not equal to the lower slopes.

AKIMA interpolation

Before knowing the Modified AKIMA interpolation – 'MAKIMA', it is necessary to understand the AKIMA piecewise cubic Hermitian interpolation.

For each interval $[x_i \ x_{i+1}]$, piecewise cubic Hermitian interpolation gives a cubic polynomial that can both interpolate the values y_i and y_{i+1} at the nodes x_i and x_{i+1} and calculate the specific derivatives d_i and d_{i+1} at the nodes x_i and x_{i+1}.

(derivatives d is the slopes of the curve) The main point is the choice of derivatives di，so AKIMA introduced derivative formula that can avoid excessive local wiggles and get a smoother curve.

Let $\delta_i=(y_{i+1}-y_i)/(\ x_{i+1}-x_i)$ be the slope of the line at x_i .

It is supposed to get the derivative d_i of the node x_i. As it has been talked about that it only uses five local data points [4]. It is reasonable to surmise that the derivative d_i needs to decrease the undulation by approaching slopes of line δ_{i-1}, δ_{i-2} δ_i, δ_{i+1}, and δ_{i+2} .

Then the AKIMA's derivative at x_i is defined as:

$$d_i = \frac{|\delta_{i+1} - \delta_i|\delta_{i-1} + |\delta_{i-1} - \delta_{i-2}|\delta_i}{|\delta_{i+1} - \delta_i| + |\delta_{i-1} - \delta_{i-2}|}$$

Function CODE:

```
function vq = akima(x,v,xq)
%it is easier to find the error of input data
if numel (x) <2
    error('the input data must be more than one'),
end
if numel(x)>numel(v) || numel(x)<numel(v)
    error('the input arrays of x and v must be same'),
end
```

```
    x = x(:).'; v = v(:).';
    dfx = diff(x);
    de = diff(v)./dfx;
    slopes = akimaSlopes(de);
    vq = ppval(pwch(x,v,slopes,dfx,de),xq);
end
function s = akimaSlopes(de)
     n =  numel(de) + 1;
    % for left augment
    de_0  = 2*de(1)   - de(2);
    de_minus1 = 2*de_0    - de(1);
    % for right augment
    de_plus1  = 2*de(n-1) - de(n-2);
    de_plus2 = 2*de_plus1    - de(n-1);
    %slopes
    de = [de_minus1 de_0 de de_plus1 de_plus2];
    weight = abs(diff(de));
    w1 = weight(1:n);
    w2 = weight(3:end);
    de1 = de(2:n+1);
    de2 = de(3:n+2);
    weightssum = w1 + w2;
```

```
    s = (w2./weightssum) .* de1 + (w1./weightssum) .* de2;

    %for edge case

    individual = w1 == 0 & w2 == 0;

    s(individual) = (de1(individual) + de2(individual)) / 2;
end
```

From a set of data, for example, (x_1,y_1)…,(x_n,y_n). It is supposed to use the formula above to calculate the derivative of all the points. For the points (x_n,y_n) and (x_1,y_1), $d_n = \frac{w_1}{w_1+w_2}\delta_{n-1} + \frac{w_2}{w_1+w_2}\delta_n$, and $d_1 = \frac{w_1}{w_1+w_2}\delta_0 + \frac{w_2}{w_1+w_2}\delta_1$. And it could be notice that it requires $\delta_{n+1} = \frac{y_{n+2}-y_{n+1}}{x_{n+2}-x_{n+1}}$, $\delta_{-1} = \frac{y_{-1}-y_0}{x_{-1}-x_0}$. However, the values of $y_{n+2}, x_{n+2}, y_{n+1}, x_{n+1}, y_0, x_0, y_{-1}, x_{-1}$ are not given in the input data. Therefore, AKIMA used edge extrapolation (It is a way which can estimate the extrapolation value from the data) to compute them as

$\delta_0=2\delta_1-\delta_2$, $\delta_{-1}=2\delta_0-\delta_1$ and $\delta_n=2\delta_{n-1}-\delta_{n-2}$, $\delta_{n+1}=2\delta_n-\delta_{n-1}$.

Edge extrapolation [4]:

It is known that there are three point (x_1,y_1), (x_2,y_2), (x_3,y_3). And we have the extrapolation edge point $(x_4,\ y_4)$, (x_5,y_5), (x_{-1},y_{-1}), (x_0,y_0) which can complete following equations

$$x_5 - x_3 = x_4 - x_2 = x_3 - x_1$$
$$\frac{y_5-y_4}{x_5-x_4} - \frac{y_4-y_3}{x_4-x_3} = \frac{y_4-y_3}{x_4-x_3} - \frac{y_3-y_2}{x_3-x_2} = \frac{y_3-y_2}{x_3-x_2} - \frac{y_2-y_1}{x_2-x_1}$$
$$x_3 - x_1 = x_2 - x_0 = x_1 - x_{-1}$$
$$\frac{y_3-y_2}{x_3-x_2} - \frac{y_2-y_1}{x_2-x_1} = \frac{y_2-y_1}{x_2-x_1} - \frac{y_1-y_0}{x_1-x_0} = \frac{y_1-y_0}{x_1-x_0} - \frac{y_0-y_{-1}}{x_0-x_{-1}}$$

And then, because what we want to know is the d_1, d_2, $d_3$which means it needs δ_{-1}, δ_0, δ_1, δ_2, δ_3, δ_4.

We can calculate δ1 and δ2 by $\delta_1 = \frac{y_2 - y_1}{x_2 - x_1}, \delta_2 = \frac{y_3 - y_2}{x_3 - x_2}$

Then we use the equation we construct above, and we can get

$\delta_4 - \delta_3 = \delta_3 - \delta_2 = \delta_2 - \delta_1$
$\delta_2 - \delta_1 = \delta_1 - \delta_0 = \delta_0 - \delta_{-1}$

And it is easy to solve that

$\delta_3 = 2\delta_2 - \delta_1$
$\delta_4 = 2\delta_3 - \delta_2 = 3\delta_2 - 2\delta_1$
$\delta_0 = 2\delta_1 - \delta_2$
$\delta_{-1} = 3\delta_1 - 2\delta_2$

If there are only two points (x_1,y_1), (x_2,y_2), and we can only calculate δ1, and it needs to calculate $\delta_{-1}, \delta_0, \delta_2, \delta_3$.

We can let $\delta_{-1} = \delta_0 = \delta_2 = \delta_3 = \delta_1$

Moreover, notice that $d_i = \frac{w_1}{w_1 + w_2}\delta_{i-1} + \frac{w_2}{w_1 + w_2}\delta_i$, it can find that the numerator and denominator of the point (x_i,y_i) could be 0 if the lower and the higher slopes are both equal ($\delta_{i-2}=\delta_{i-1}$, $\delta_i=\delta_{i+1}$).

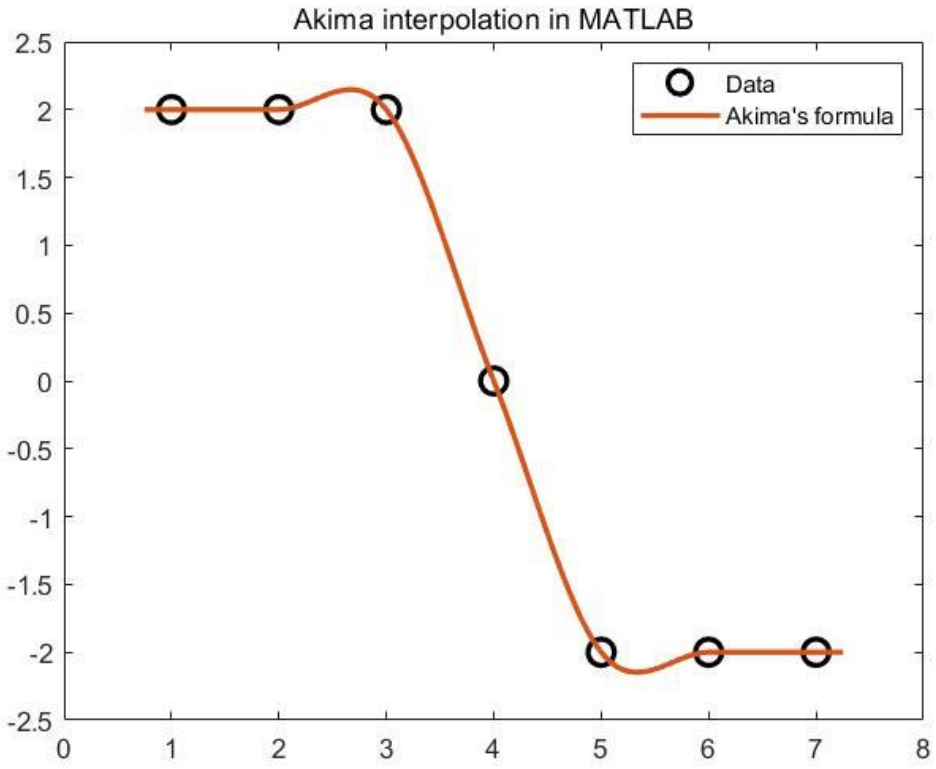
$[d_i = \frac{w_1}{w_1 + w_2}\delta_{i-1} + \frac{w_2}{w_1 + w_2}\delta_i = \frac{|\delta_{i+1} - \delta_i|\delta_{i-1} + |\delta_{i-1} - \delta_{i-2}|\delta_i}{|\delta_{i+1} - \delta_i| + |\delta_{i-1} - \delta_{i-2}|} = \frac{0}{0}]$

The AKIMA's formula would return NaN in this situation. Therefore, AKIMA uses the formula which takes the average of the upper and lower slopes for this kind of edge case: $d_i=(\delta_{i-1}+\delta_i)/2$.

For example, for the data series that (1,2), (2,2), (3,2), (4,0), (5,-2), (6,-2), (7,2), by the calculation of AKIMA algorithm, it has $\delta_3=\delta_4=1$, and $\delta_5=\delta_6=0$. We use the MATLAB code of AKIMA algorithm to plot the graph of it.

Code:

```
x = 1:7;

v = [2 2 2 0 -2 -2 -2 ];

xq = 0.50:0.05:7.50;

vqa = akima(x,v,xq);

figure

plot(x,v,'ko','LineWidth',1,'MarkerSize',8), hold on

plot(xq,vqa,'LineWidth',1)

hold off

legend('Data','AKIMA')

title('AKIMA interpolation in MATLAB')
```



As it can be seen in the above graph, the derivative at x=5 , d_i=(δ_4+δ_5)/2=-1, which means it uses the edge case formula d_i=(δ_{i-1}+δ_i)/2 to replace the NaN derivative.

However, there are two shortcomings of the AKIMA algorithm, which is why we use the modified AKIMA interpolation when analyzing some specific data.

Shortcoming (1): There is a negligible difference between d_i when the lower and the higher slopes are both equal and di when the lower and the higher slopes are nearly equal.

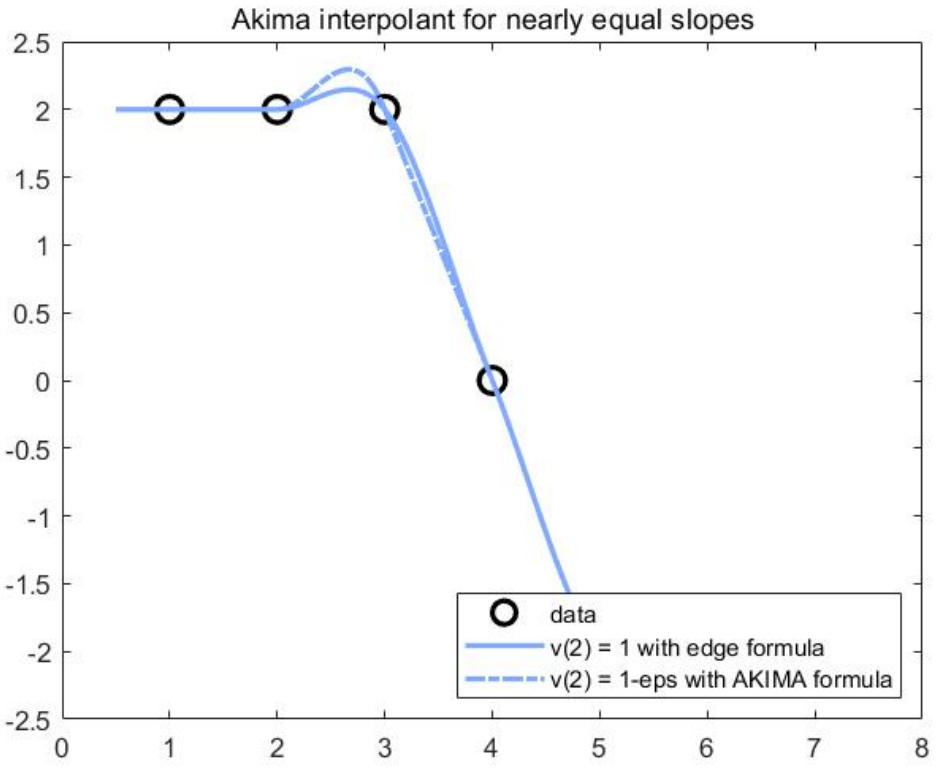
The one of them is that when the lower and the higher slopes are both almost equal, the numerator and denominator are almost zero. As discussed above, that the AKIMA algorithm would use the edge case formula to replace the NaN derivative. The slopes are not zero actually. Therefore, it would not choose the average formula. However, it has a non-negligible difference between the values of derivatives.

For example, using the example above, and let ε=2^(-52) be the difference between equal and almost equal. So let veps2=2-ε, that δeps2=ε and δeps1=−ε. Now, the value of d2 and deps2 for two data can be compared with AKIMA interpolation.

Code:

```
x = 1:7;

v = [2 2 2 0 -2 -2 -2 ];

xq = 0.50:0.05:7.50;

eps=2^(-52)

veps = v; veps(2) = veps(2) - eps;

vqa = akima(x,v,xq);

vqa = akima(x,veps,xq);

    plot(x,v,'ko','LineWidth',1,'MarkerSize',8)



        hold on

    plot(xq,akima(x,v,xq),'Color',[1/2 2/3 1],'LineWidth',4,...

        )

    hold on

    plot(xq,akima(x,veps,xq),'-.','Color',[ 1/2 2/3

1],'LineWidth',2,...

        )

    hold off
```

```
legend('Data',' v(2) = 1 with edge formula ', 'v(2) = 1-eps
with AKIMA formula')

title('Akima interpolant for nearly equal slopes' )
```

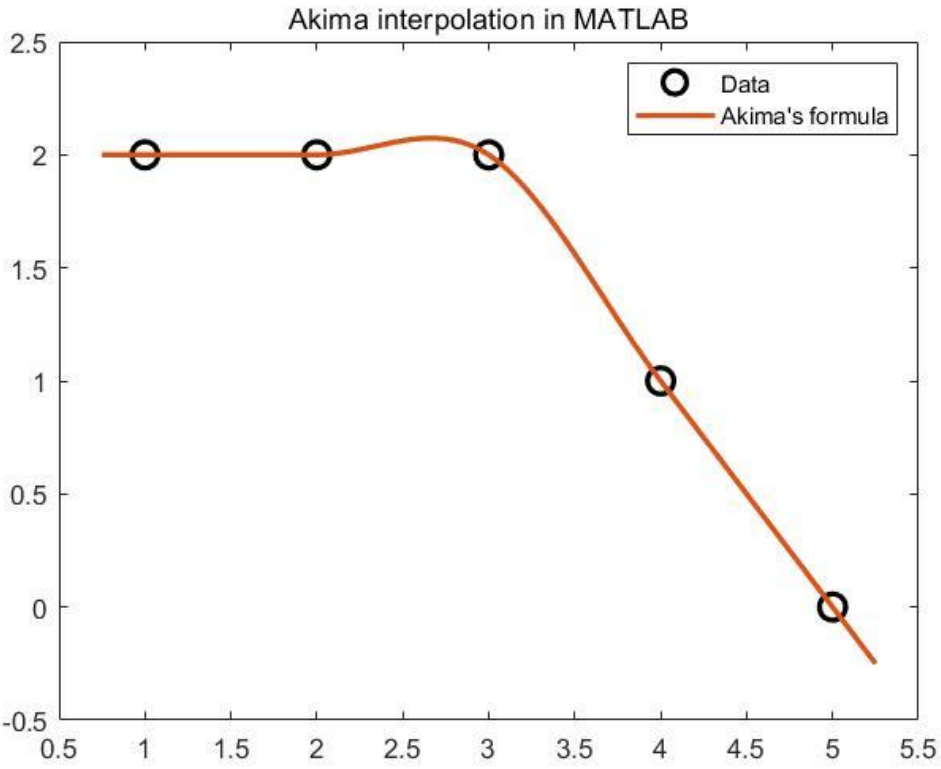


Shortcoming （2） If the lower and the higher slopes are both equal

($\delta_{i-2}=\delta_{i-1}$, $\delta_i=\delta_{i+1}$) and the higher slopes are not equal to the lower slopes, then the AKIMA interpolation produce the undulation.

CODE:

```
v = [ 2 2 2 1 0];

xq = 0.50:0.05:5.50;

vqa = AKIMA(x,v,xq);

figure

plot(x,v,'ko','LineWidth',5,'MarkerSize',10), hold on

plot(xq,vqa,'LineWidth',3)

hold off

legend('Data','AKIMA's formula')

title('AKIMA interpolation in MATLAB')
```



As the above situation, while the edge case formula is used, it might produce the undulation. (edge formula $d_i=(\delta_{i-1}+\delta_i)/2$). And as the second example above, $d_3=(\delta_2+\delta_3)/2$=-0.5, which is not equal to δ_3= -1, this is why there is undulation. Actually, the only case that there is no undulation is that $d_i=(\delta_{i-1}+\delta_i)/2=\delta_i$. So we get $\delta_{i-1}=\delta_i$, which means the upper slope equal to the lower slope. Therefore, it can conclude that if the lower and the higher slopes are both equal ($\delta_{i-2}=\ \delta_{i-1}$, $\delta_i=\delta_{i+1}$) and the higher slopes are not +equal to the lower slopes, then the AKIMA interpolation produces the undulation.

MAKIMA cubic Hermitian interpolation

And these two shortcomings of AKIMA interpolation is the reason why we use the MAKIMA cubic Hermitian interpolation in MATLAB. The modified AKIMA interpolant can solve these two problems well.

Modified AKIMA formula:

$$d_i = \frac{w_1}{w_1 + w_2}\delta_{i-1} + \frac{w_2}{w_1 + w_2}\delta_i$$

$$w_1 = |\delta_{i+1} - \delta_i| + |\frac{\delta_{i+1} + \delta_i}{2}|,\ w_2 = |\delta_{i-1} - \delta_{i-2}| + |\frac{\delta_{i-1} + \delta_{i-2}}{2}|$$

As it can be seen, the formula of derivative is the same as the AKIMA algorithm. Only the weights w_1 and w_2 are changed.

How to solve the negligible difference between di when the lower and the higher slopes are both equal and di when the lower and the higher slopes are nearly equal to AKIMA(1).`

$$d_i = \frac{w_1}{w_1 + w_2}\delta_{i-1} + \frac{w_2}{w_1 + w_2}\delta_i$$

$$= \frac{|\delta_{i+1} - \delta_i| + |\frac{\delta_{i+1} + \delta_i}{2}|}{|\delta_{i+1} - \delta_i| + |\frac{\delta_{i+1} + \delta_i}{2}| + |\delta_{i-1} - \delta_{i-2}| + |\frac{\delta_{i-1} + \delta_{i-2}}{2}|}\delta_{i-1} + \frac{|\delta_{i-1} - \delta_{i-2}| + |\frac{\delta_{i-1} + \delta_{i-2}}{2}|}{|\delta_{i+1} - \delta_i| + \left|\frac{\delta_{i+1} + \delta_i}{2}\right| + |\delta_{i-1} - \delta_{i-2}| + |\frac{\delta_{i-1} + \delta_{i-2}}{2}|}\delta_i$$

we know $\delta_{i-2} = \delta_{i-1}$, and $\delta_i = \delta_{i+1}$

let $\delta_{i-2} = \delta_{i-1} = a$, $\delta_i = \delta_{i+1} = b$

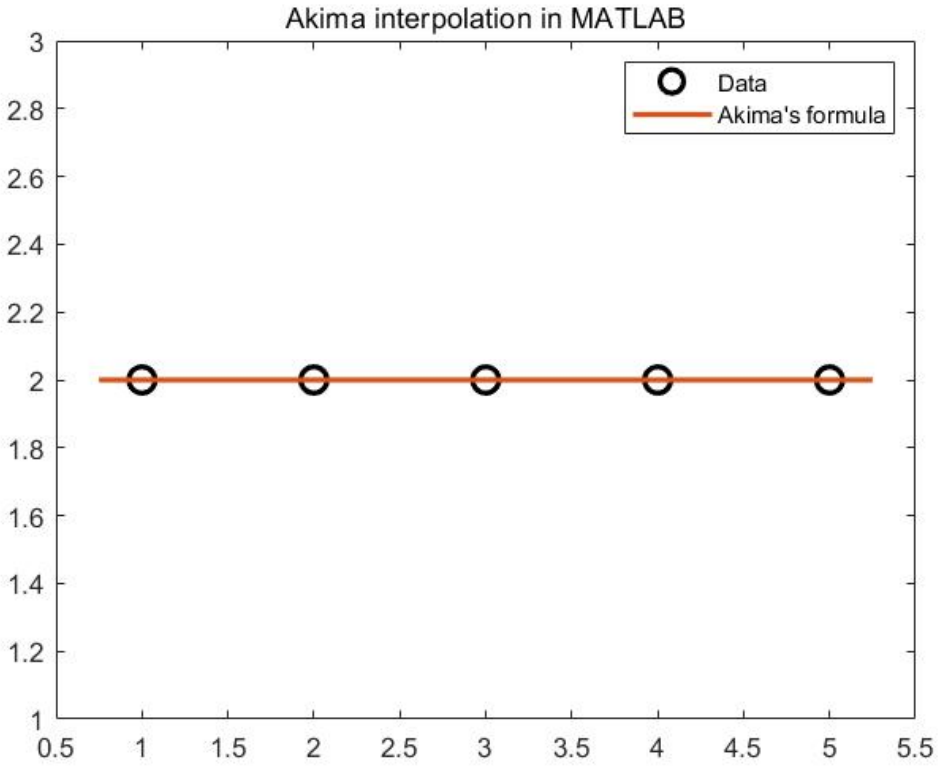
then $d_i = \frac{ba + ab}{a + b}$

$$= \frac{2ab}{a + b}$$

As you can see, the numerator and denominator of d_i is not zero now, so it is not supposed to use the edge formula when the lower and the higher slopes are both equal ($\delta_{i-2} = \delta_{i-1}$, $\delta_i = \delta_{i+1}$). Therefore, when the lower and the higher slopes are nearly equal, there is not any different.

CODE:

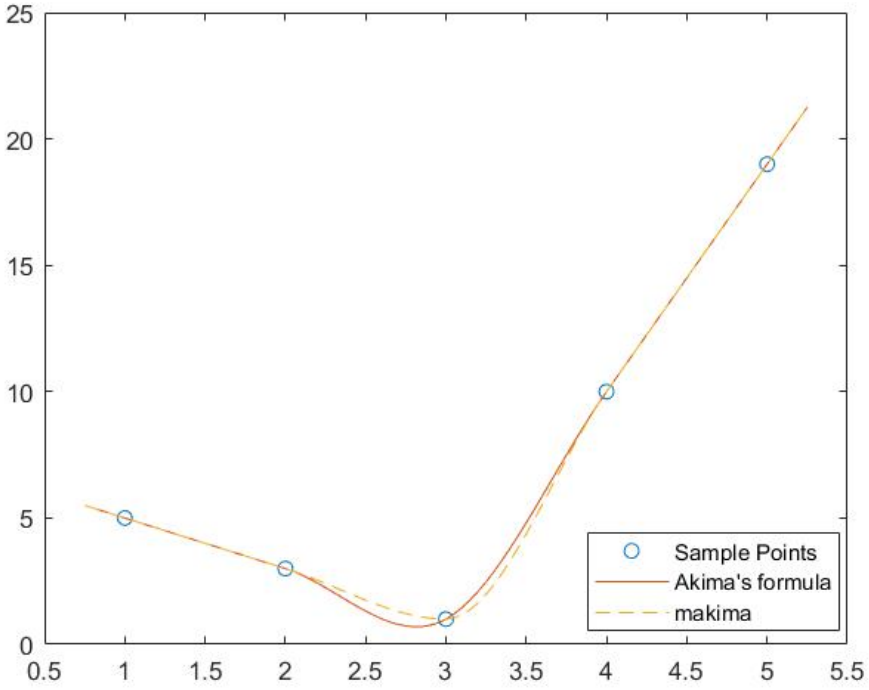
```
x = 1:5;

v = [ 2 2 2 2 2];

xq = 0.50:0.05:5.50;

vqa = AKIMA(x,v,xq);

figure

plot(x,v,'ko','LineWidth',4,'MarkerSize',8), hold on

plot(xq,vqa,'LineWidth',4)

hold off

legend('Data','AKIMA's formula')

title('AKIMA interpolation in MATLAB')
```

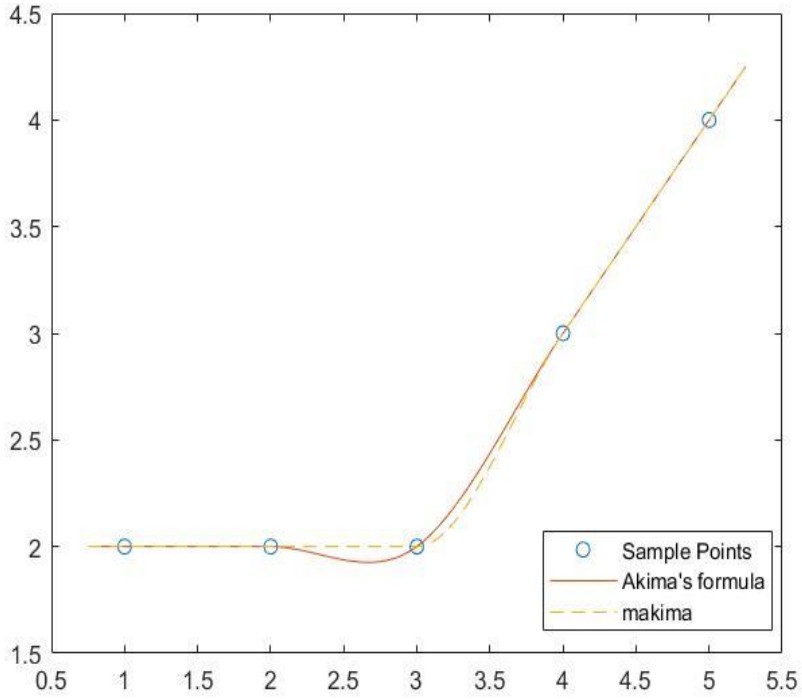


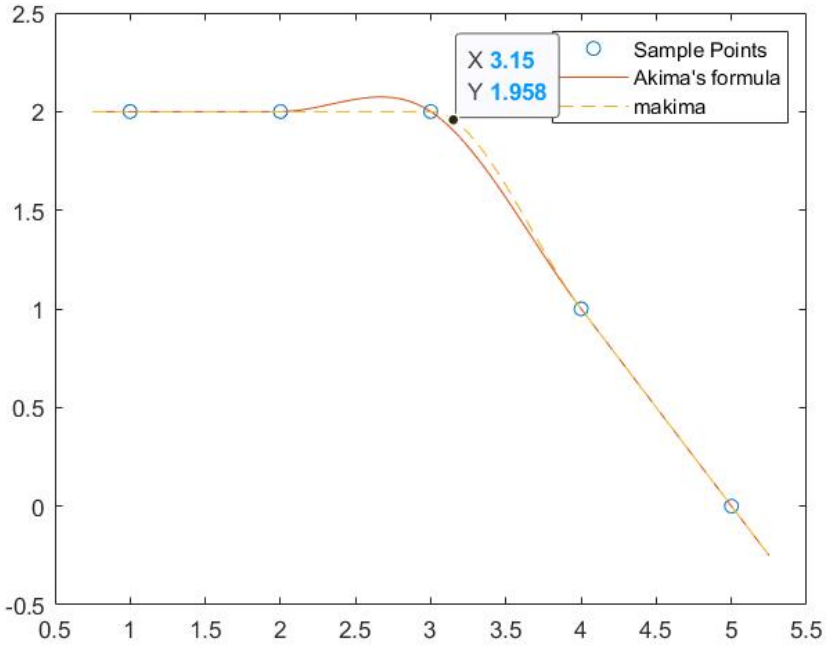
However, there is one case that we would get d_i=NaN. When $y_{n+2=}y_{n+1} = y_n$=$y_{n-1} = y_{n-2}$, then the four slopes would be all equal to zero (δ_{i-2}= δ_{i-1}= δ_i=δ_{i+1}). For this special case, it can be set that d_i=0.

How to solve the overshoot of AKIMA(2):

Now, it is supposed to work out how the MAKIMA formula decreases the undulation. Use some examples to find the rule.


25
20
15
10
5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
Sample Points
Akima's formula
makima





Code:

```
x = [1 2 3 4 5];

v = [2 2 2 1 0 ];
```

```
xq = 0.50:0.05:5.50;

m = MAKIMA(x,v,xq);

vqa = AKIMA(x,v,xq);

plot(x,v,'o',xq,vqa,'-',xq,m,'--')

hold off
legend('Sample Points','AKIMA's formula','MAKIMA')

x = [1 2 3 4 5];

v = [2 2 2 3 4];

xq = 0.50:0.05:5.50;

m = MAKIMA(x,v,xq);

vqa = AKIMA(x,v,xq);

plot(x,v,'o',xq,vqa,'-',xq,m,'--')

hold off

legend('Sample Points','AKIMA''s formula','MAKIMA')



x = [1 2 3 4 5];
```

```
v = [5 3 1 10 19];

xq = 0.50:0.05:5.50;

m = MAKIMA(x,v,xq);

vqa = AKIMA(x,v,xq);

plot(x,v,'o',xq,vqa,'-',xq,m,'--')

hold off

legend('Sample Points','AKIMA''s formula','MAKIMA')
```

As it can be seen in the above formula, we can find that the AKIMA interpolation divides the undulation evenly from the two nodes beside. However, the MAKIMA interpolation divides the undulation more from the slopes which are close to zero, which would directly prevent the undulation. $w_1=|\delta_{i+1}-\delta_i|+|\frac{\delta_{i+1}+\delta_i}{2}|$, $w_2=|\delta_{i-1}-\delta_{i-2}|+|\frac{\delta_{i-1}+\delta_{i-2}}{2}|$

Prove:

$$d_i = \frac{w_1}{w_1+w_2}\delta_{i-1} + \frac{w_2}{w_1+w_2}\delta_i$$

$$=\frac{|\delta_{i+1}-\delta_i|+|\frac{\delta_{i+1}+\delta_i}{2}|}{|\delta_{i+1}-\delta_i|+|\frac{\delta_{i+1}+\delta_i}{2}|+|\delta_{i-1}-\delta_{i-2}|+|\frac{\delta_{i-1}+\delta_{i-2}}{2}|}\delta_{i-1} + \frac{|\delta_{i-1}-\delta_{i-2}|+|\frac{\delta_{i-1}+\delta_{i-2}}{2}|}{|\delta_{i+1}-\delta_i|+\left|\frac{\delta_{i+1}+\delta_i}{2}\right|+|\delta_{i-1}-\delta_{i-2}|+|\frac{\delta_{i-1}+\delta_{i-2}}{2}|}\delta_i$$

we know $\delta_{i-2}=\delta_{i-1}$, and $\delta_i=\delta_{i+1}$

let $\delta_{i-2}=\delta_{i-1}=a$, $\delta_i=\delta_{i+1}=b$

then $d_i=\frac{ba+ab}{a+b}$

$=\frac{2ab}{a+b}$

d_i（edge）=

$=\frac{a+b}{2}$

We want to confirm that $|d_i| \leq |d_i \text{ (edge)}|$,

Then,

It would be $|\frac{2ab}{a+b}| \leq |\frac{a+b}{2}|$

(1) So when ab>0

$a+b \geq 2\sqrt{ab}$

$\frac{a+b}{2} >= \sqrt{ab}$

$\frac{1}{a+b} \leq \frac{1}{2\sqrt{ab}}$

$|\frac{2ab}{a+b}| \leq \frac{2ab}{2\sqrt{ab}} \leq \sqrt{ab} \leq |\frac{a+b}{2}|$

(2) When ab<0

$a+b \geq 2\sqrt{|ab|}$

$\frac{a+b}{2} >= \sqrt{|ab|}$

$\frac{1}{a+b} \leq \frac{1}{2\sqrt{|ab|}}$

$|\frac{2ab}{a+b}| \leq \frac{|2ab|}{2\sqrt{|ab|}} \leq \sqrt{|ab|} \leq |\frac{a+b}{2}|$

It has been proved.

Then it can be concluded that the MAKIMA interpolation would directly prevent the undulation compared with AKIMA interpolation.

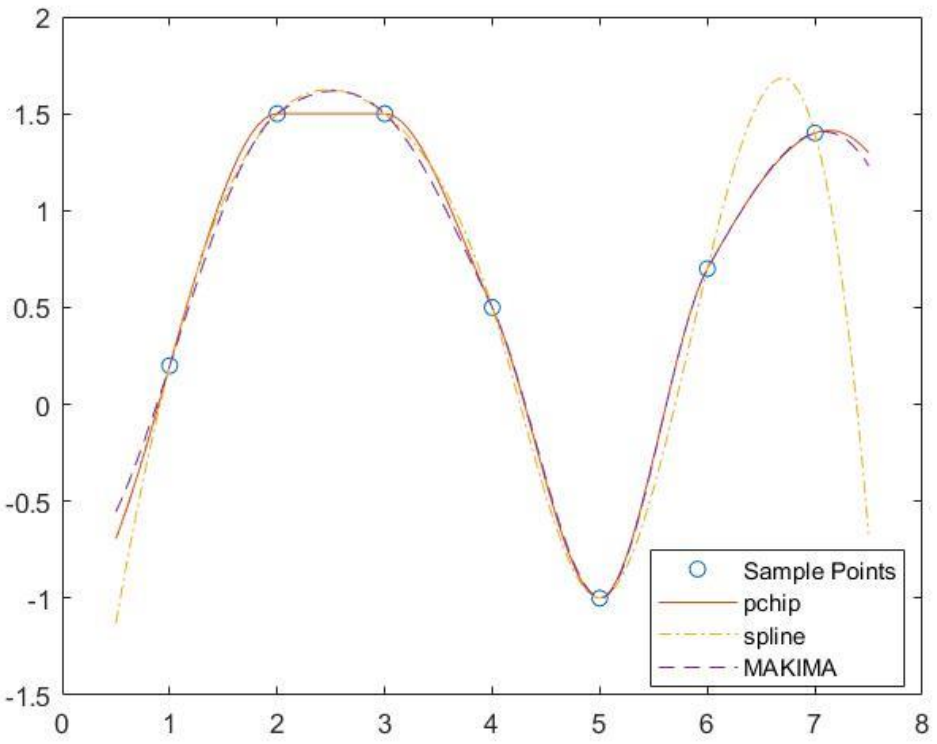
Comparing with Pchip and Spline interpolation, Makima interpolation

The spline curve (which is actually a piecewise polynomial) has slight oscillations on both sides, but it is smoother because it requires the interpolation function to be continuous at the interpolation point. The first-order derivative of the interpolation function is continuous, the second-order derivative of the interpolation function is continuous, and the PCHIP interpolation function is relatively stable, but not as smooth as the spline markings, because it only requires that the interpolation function be continuous at the interpolation point, and the first derivative of the interpolation function is continuous. However, the MAKIMA interpolation produces undulations between the PCHIP and SPLINE. Firstly, the MAKIMA interpolation only requires that the interpolation function be continuous at the interpolation point, and the first derivative of the interpolation function is continuous, which means it produces smaller undulation than the SPLINE. Secondly, the MAKIMA interpolation use five points from two side, however, the PCHIP use four points from one side. Therefore, the MAKIMA interpolation is smoother than the PCHIP.

Here's a representative example comparing these three cubic Hermitian interpolants:[4]

```
x = [1 2 3 4 5 6 7];

y = [0.2 1.5 1.5 0.5 -1 0.7 1.4 ];

xq2 = 0.50:0.05:7.50;

p = pchip(x,y,xq2);

s = spline(x,y,xq2);

m = makima(x,y,xq2);

plot(x,y,'o',xq2,p,'-',xq2,s,'-.',xq2,m,'--')
```

```
hold off

legend('Sample Points','pchip','spline','MAKIMA')
```



- As the figure above, 'MAKIMA' actually produces better results than 'Spline' and 'Pchip'. The first reason is that it has smaller undulations than 'Spline' between the interval [2 3] and [6 7]. The second reason is that it is not as aggressive as 'Pchip' in reducing the undulations, the AKIMA algorithm curve is smoother beside the nodes compared with 'Pchip'.

Use the Lagrange algorithm, Double Hermitian interpolation polynomial, and the MAKIMA algorithm to solve the polynomial with 2 or 3 points.

Lagrange interpolation

For a given point n+1, there is only one Lagrange polynomial whose degree does not exceed n. If a higher degree polynomial is included, there are infinite, because all polynomials that differ from each other satisfy the condition.

For a certain multi-form function, it is known that there are given k + 1 value points:

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

This corresponds to the independent variable's position and corresponds to the value of the function at this position.

Assuming that any two different x_i are different from each other, then the Lagrange interpolation formulas obtained by applying the Lagrange interpolation formula are:

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

Each of them is Lagrange's basic multivariate style (or interpolated basis function), and its expression is:

$$\ell_j(x) := \prod_{i=0, i\neq j}^{k} \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)}$$

[3]

Lagrange's basic multiple formulas characteristic is to take a ratio of 1 at the top and take an average of 0 at other points.

Hermitian interpolation

Hermitian interpolation is another type of interpolation problem. This type of interpolation requires that the interpolation polynomial's function value be the same as the original function value at a given node. At the same time, it is also required that the

derivative value of the first order of the interpolation polynomial up to the specified order at the node is also equal to the corresponding derivative value of the interpolated function. Such interpolation is called Hermitian interpolation. Hermitian interpolation satisfies that it is equal to the given function value at the node, and the derivative value at the node is also equal to the given derivative value. For the case of higher-order derivatives, Hermitian interpolation polynomials are more complicated. In actual situations, it is often encountered that the function value and the first-order derivative are given.

Double Hermitian interpolation polynomial:

If there exists $H_{2n+1}(x) \in P_{2n+1}$, that

$H'_{2n+1}(x_i) = f'(x_i), i = 0,1, \dots, n$

$H_{2n+1}(x_i) = f(x_i)$

Then it can concluded that $H_{2n+1}(x)$ is the double Hermitian interpolation polynomial for f(x).

Now we try to work it out.

LET

$$\ell_j(x) := \prod_{i=0, i\neq j}^{k} \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)}$$

We only consider that the slope and the function value of the nodes are equal

$H_{2n+1}(x) = \sum_{i=0}^{n} H_i(x) y_i + \sum_{i=0}^{n} h_i(x) y'_i$

And because we want

$H'_{2n+1}(x_i) = f'(x_i), i = 0,1, \dots, n$

$H_{2n+1}(x_i) = f(x_i)$

So we let

$$\begin{cases} H_i(x_j) = g_{ij} = \begin{cases} 0, j \neq i \\ 1, j = i \end{cases}, (i = 0,1,2,\cdots,n) \\ H_i'(x_j) = 0 \end{cases}$$

$$\begin{cases} h_i'(x_j) = g_{ij} = \begin{cases} 0, j \neq i \\ 1, j = i \end{cases}, (i = 0,1,2,\cdots,n) \\ h_i(x_j) = 0 \end{cases}$$

And then it is easy to see that

$$\begin{aligned} H_{2n+1}(X_1) &= \sum_{i=0}^{n} H_i(x_i)y_i + \sum_{i=0}^{n} h_i(x_i)y_i' \\ &= \sum_{i=0}^{n} g_{ij}y_i + 0 \\ &= y_i \end{aligned}$$

$$\begin{aligned} H_{2n+1}'(x_i) &= \sum_{i=0}^{n} H_i'(X_i)y_i + \sum_{i=0}^{n} h_i'(x_i)y_i' \\ &= 0 + \sum_{i=0}^{n} g_{ij}y_i' \\ &= y_i' \end{aligned}$$

It is obvious that Hi(x) has the same zero divisor as $\ell_i(x)$, Hi(x)'=0, so it can be confirmed that Hi(x) has the divisor of $\ell_i^2(x)$, and it is easy to find that the degree of $\ell_i^2(x)$ is 2n.

So we let

$H_i(x) = (ax + b)\ell_i^2(x)$

It is known that $\ell_i^2(x_i) = 1$ and $H_i(x_i) = 1$

ax_i + b = 1

$H_i'(x_i) = a\ell_i^2(x_i) + 2(ax_i + b)\ell_i(x_i)\ell_i'(x_i) = a + 2\ell_i'(x_i) = 0$

From $a + 2\ell_i'(x_i) = 0$ and ax_i + b = 1

It can be got that a=$-2\ell_i^2(x_i)$, and b=$1 + 2x_i\ell_i'(x_i)$

And then

$$H_i(x) = [1 + 2x_i\ell_i'(x_i) - 2\ell_i'(x_i)\cdot x]\ell_i^2(x) = [1 - 2(x - x_i)\ell_i'(x_i)]\ell_i^2(x)$$

$$= \left[1 - 2(x - x_i)\sum_{k=0,k\neq i}^{n}\frac{1}{x_i - x_k}\right]\ell_i^2(x)$$

As the same way,

Let $h_i(x) = (cx + d)\ell_i^2(x)$

It is known that $h_i(x_i) = 0$, and $\ell_i(x_i) = 1$

Then cx_i+d=0

$$h_i'(x_i) = c\ell_i^2(x_i) + 2(cx_i + d)\ell_i(x_i)\ell_i'(x_i) = c = 1$$

then it can be got that c=1 and d=-x_i

$$h_i(x) = (x - x_i)\ell_i^2(x)$$

and now we can get the formula

$$H_{2n+1}(x) = \sum_{i=0}^{n} H_i(x)y_i + \sum_{i=0}^{n} h_i(x)y_i'$$

$$= \sum_{i=0}^{n} [1 - 2(x - x_i)\ell_i'(x_i)]\ell_i^2(x)y_i + \sum_{i=0}^{n} (x - x_i)\ell_i^2(x)y_i'$$

Now we only consider two point interpolation

When n=1, $H_3(x_0) = y_0, H_3(x_1) = y_1, H_3'(x_0) = y_0', H_3'(x_1) = y_1{}'$

$$\ell_i(x) := \prod_{i=0,i\neq j}^{k}\frac{x - x_i}{x_i - x_j} = \frac{(x - x_0)}{(x_i - x_0)}\cdots\frac{(x - x_{i-1})}{(x_i - x_{i-1})}\frac{(x - x_{i+1})}{(x_i - x_{i+1})}\cdots\frac{(x - x_k)}{(x_i - x_k)}$$

$$\ell_i'(x) = \sum_{j=0,j\neq i}^{n}\frac{1}{x_i - x_j}$$

$$\ell_i^2(x) = \sum_{j=0,j\neq i}^{n}\left(\frac{x - x_i}{x_i - x_j}\right)^2$$

$$H_3(x) = \sum_{i=0}^{n} [1 - 2(x - x_i)\ell_i'(x_i)]\ell_i^2(x)y_i + \sum_{i=0}^{n} (x - x_i)\ell_i^2(x)y_i'$$

$=\left[\left(1+2\frac{x-x_0}{x_1-x_0}\right)y_0+(x-x_0)y_0'\right]\left(\frac{x-x_1}{x_0-x_1}\right)^2+\left[\left(1+2\frac{x-x_1}{x_0-x_1}\right)y_1+(x-x_1)y_1'\right]\left(\frac{x-x_0}{x_1-x_0}\right)^2$ And

it is known by AKIMA algorithm that

$$d_i=\frac{w_1}{w_1+w_2}\delta_{i-1}+\frac{w_2}{w_1+w_2}\delta_i$$

$w_1=|\delta_{i+1}-\delta_i|+|\frac{\delta_{i+1}+\delta_i}{2}|,\ w_2=|\delta_{i-1}-\delta_{i-2}|+|\frac{\delta_{i-1}+\delta_{i-2}}{2}|$

And we know

$$\delta_0=\frac{y_1-y_0}{x_1-x_0}$$

then we can get that

$$y_0=d_0=\frac{w_1}{w_1+w_2}\delta_{-1}+\frac{w_2}{w_1+w_2}\delta_0$$

$$y_1=d_1=\frac{w_1}{w_1+w_2}\delta_0+\frac{w_2}{w_1+w_2}\delta_1$$

$$\mathrm{d}_0=\frac{\left(|\delta_1-\delta_0|+\frac{|\delta_1+\delta_0|}{2}\right)\delta_{-1}}{|\delta_1-\delta_0|+\frac{|\delta_1+\delta_0|}{2}+|\delta_{-1}-\delta_{-2}|+\frac{|\delta_{-1}+\delta_{-2}|}{2}}$$

$$+\frac{\left(|\delta_{-1}-\delta_{-2}|+\frac{|\delta_{-1}+\delta_{-2}|}{2}\right)\delta_0}{|\delta_1-\delta_0|+\frac{|\delta_1+\delta_0|}{2}+|\delta_{-1}-\delta_{-2}|+\frac{|\delta_{-1}+\delta_{-2}|}{2}}$$

$=\frac{\delta_0^2}{2\delta_0}+\frac{\delta_0^2}{2\delta_0}$

$=\delta_0$

$$\mathrm{d}_1=\frac{\left(|\delta_2-\delta_1|+\frac{|\delta_2+\delta_1|}{2}\right)\delta_0}{|\delta_2-\delta_1|+\frac{|\delta_2+\delta_1|}{2}+|\delta_0-\delta_{-1}|+\frac{|\delta_0+\delta_{-1}|}{2}}$$

$$+\frac{\left(|\delta_0-\delta_{-1}|+\frac{|\delta_0+\delta_{-1}|}{2}\right)\delta_1}{|\delta_2-\delta_1|+\frac{|\delta_2+\delta_1|}{2}+|\delta_0-\delta_{-1}|+\frac{|\delta_0+\delta_{-1}|}{2}}$$

$=\frac{\delta_0^2}{2\delta_0}+\frac{\delta_0^2}{2\delta_0}$

$=\delta_0$

And we can get that

$$H_3(x)=\left[\left(1+2\frac{x-x_0}{x_1-x_0}\right)y_0+(x-x_0)y_0'\right]\left(\frac{x-x_1}{x_0-x_1}\right)^2+\left[\left(1+2\frac{x-x_1}{x_0-x_1}\right)y_1+(x-x_1)y_1'\right]\left(\frac{x-x_0}{x_1-x_0}\right)^2=\left[\left(1+2\frac{x-x_0}{x_1-x_0}\right)y_0+(x-x_0)(\frac{y_1-y_0}{x_1-x_0})\right]\left(\frac{x-x_1}{x_0-x_1}\right)^2+\left[\left(1+2\frac{x-x_1}{x_0-x_1}\right)y_1+(x-x_1)(\frac{y_1-y_0}{x_1-x_0})\right]\left(\frac{x-x_0}{x_1-x_0}\right)^2$$

Now we only consider three point interpolation

When n=2, $H_4(x_0)=y_0, H_4(x_1)=y_1, H_4(x_2)=y_2, H_4'(x_0)=y_0', H_4'(x_1)=y_1', H_4'(x_2)=y_2'$.

$$\ell_i(x):=\prod_{i=0,i\neq j}^{k}\frac{x-x_i}{x_i-x_j}=\frac{(x-x_0)}{(x_i-x_0)}\cdots\frac{(x-x_{i-1})}{(x_i-x_{i-1})}\frac{(x-x_{i+1})}{(x_i-x_{i+1})}\cdots\frac{(x-x_k)}{(x_i-x_k)}$$

$$\ell_i'(x)=\sum_{j=0,j\neq i}^{n}\frac{1}{x_i-x_j}$$

$$\ell_i^2(x)=\sum_{j=0,j\neq i}^{n}\left(\frac{x-x_i}{x_i-x_j}\right)^2$$

$$H_4(x)=\sum_{i=0}^{n}[1-2(x-x_i)\ell_i'(x_i)]\ell_i^2(x)y_i+\sum_{i=0}^{n}(x-x_i)\ell_i^2(x)y_i'$$

When i=0,

$$\ell_i'(x)=\sum_{j=0,j\neq i}^{n}\frac{1}{x_i-x_j}=\frac{1}{x_0-x_1}+\frac{1}{x_0-x_2}$$

$$\ell_i^2(x)=\sum_{j=0,j\neq i}^{n}\left(\frac{x-x_i}{x_i-x_j}\right)^2=\left(\frac{x-x_0}{x_0-x_1}\right)^2+\left(\frac{x-x_0}{x_0-x_2}\right)^2$$

When i=1,

$$\ell_i'(x)=\sum_{j=0,j\neq i}^{n}\frac{1}{x_i-x_j}=\frac{1}{x_1-x_0}+\frac{1}{x_1-x_2}$$

$$\ell_i^2(x)=\sum_{j=0,j\neq i}^{n}\left(\frac{x-x_i}{x_i-x_j}\right)^2=\left(\frac{x-x_1}{x_1-x_0}\right)^2+\left(\frac{x-x_1}{x_1-x_2}\right)^2$$

When i=2,

$$\ell_i'(x) = \sum_{j=0, j\neq i}^{n} \frac{1}{x_i - x_j} = \frac{1}{x_2 - x_0} + \frac{1}{x_2 - x_1}$$

$$\ell_i^2(x) = \sum_{j=0, j\neq i}^{n} \left(\frac{x - x_i}{x_i - x_j}\right)^2 = \left(\frac{x - x_2}{x_2 - x_0}\right)^2 + \left(\frac{x - x_2}{x_2 - x_1}\right)^2$$

$$H_4(x)=\left[\left(1 - 2(x - x_0)\left(\frac{1}{x_0 - x_1} + \frac{1}{x_0 - x_2}\right)\right)\left[\left(\frac{x - x_0}{x_0 - x_1}\right)^2 + \left(\frac{x - x_0}{x_0 - x_2}\right)^2\right]y_0 + (x - x_0)\left[\left(\frac{x - x_0}{x_0 - x_1}\right)^2 + \left(\frac{x - x_0}{x_0 - x_2}\right)^2\right]y_0' + \left[\left(1 - 2(x - x_1)\left(\frac{1}{x_1 - x_0} + \frac{1}{x_1 - x_2}\right)\right)\left[\left(\frac{x - x_1}{x_1 - x_0}\right)^2 + \left(\frac{x - x_1}{x_1 - x_2}\right)^2\right]y_1 + (x - x_1)\left[\left(\frac{x - x_1}{x_1 - x_0}\right)^2 + \left(\frac{x - x_1}{x_1 - x_2}\right)^2\right]y_1' + \left[\left(1 - 2(x - x_2)\left(\frac{1}{x_2 - x_0} + \frac{1}{x_2 - x_1}\right)\right)\left[\left(\frac{x - x_2}{x_2 - x_0}\right)^2 + \left(\frac{x - x_2}{x_2 - x_1}\right)^2\right]y_2 + (x - x_2)\left[\left(\frac{x - x_2}{x_2 - x_0}\right)^2 + \left(\frac{x - x_2}{x_2 - x_1}\right)^2\right]y_2'$$

And it is known by AKIMA algorithm that

$$d_i = \frac{w_1}{w_1 + w_2}\delta_{i-1} + \frac{w_2}{w_1 + w_2}\delta_i$$

$$w_1=|\delta_{i+1} - \delta_i| + \left|\frac{\delta_{i+1} + \delta_i}{2}\right|,\ w_2=|\delta_{i-1} - \delta_{i-2}| + \left|\frac{\delta_{i-1} + \delta_{i-2}}{2}\right|$$

And we know δ_0 and δ_1

$$\delta_0 = \frac{y_1 - y_0}{x_1 - x_0}$$

$$\delta_1 = \frac{y_2 - y_1}{x_2 - x_1}$$

$$\delta_2 = 2\delta_1 - \delta_0$$

$$\delta_3 = 3\delta_1 - 2\delta_0$$

$$\delta_{-1} = 2\delta_0 - \delta_1$$

$$\delta_{-2} = 3\delta_0 - 2\delta_1$$

then we can get that

$$y_0 = d_0 = \frac{w_1}{w_1 + w_2}\delta_{-1} + \frac{w_2}{w_1 + w_2}\delta_0$$

$$y_1 = d_1 = \frac{w_1}{w_1 + w_2}\delta_0 + \frac{w_2}{w_1 + w_2}\delta_1$$

$$y_2 = d_2 = \frac{w_1}{w_1 + w_2}\delta_1 + \frac{w_2}{w_1 + w_2}\delta_2$$

$$d_0 = \frac{\left(|\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2}\right)\delta_{-1}}{|\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2} + |\delta_{-1} - \delta_{-2}| + \frac{|\delta_{-1} + \delta_{-2}|}{2}}$$

$$+ \frac{\left(|\delta_{-1} - \delta_{-2}| + \frac{|\delta_{-1} + \delta_{-2}|}{2}\right)\delta_0}{|\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2} + |\delta_{-1} - \delta_{-2}| + \frac{|\delta_{-1} + \delta_{-2}|}{2}}$$

$$= \frac{\left(|\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2}\right)\delta_{-1}}{|\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2} + |2\delta_0 - \delta_1 - 3\delta_0 - 2\delta_1| + \frac{|2\delta_0 - \delta_1 + 3\delta_0 - 2\delta_1|}{2}} +$$
$$\frac{\left(|2\delta_0 - \delta_1 - 3\delta_0 - 2\delta_1| + \frac{|2\delta_0 - \delta_1 + 3\delta_0 - 2\delta_1|}{2}\right)\delta_0}{|\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2} + |2\delta_0 - \delta_1 - 3\delta_0 - 2\delta_1| + \frac{|2\delta_0 - \delta_1 + 3\delta_0 - 2\delta_1|}{2}}$$
$$= \frac{\left(|\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2}\right)(2\delta_0 - \delta_1)}{|\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2} + |-\delta_0 - 3\delta_1| + \frac{|5\delta_0 - 3\delta_1|}{2}} + \frac{\left(|-\delta_0 - 3\delta_1| + \frac{|5\delta_0 - 3\delta_1|}{2}\right)\delta_0}{|\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2} + |-\delta_0 - 3\delta_1| + \frac{|5\delta_0 - 3\delta_1|}{2}}$$

.

$$d_1 =$$

$$\frac{\left(|2\delta_1 - \delta_0 - \delta_1| + \frac{|2\delta_1 - \delta_0 + \delta_1|}{2}\right)\delta_0}{|2\delta_1 - \delta_0 - \delta_1| + \frac{|2\delta_1 - \delta_0 + \delta_1|}{2} + |\delta_0 - 2\delta_0 - \delta_1| + \frac{|\delta_0 + 2\delta_0 - \delta_1|}{2}}$$

$$+ \frac{\left(|\delta_0 - 2\delta_0 - \delta_1| + \frac{|\delta_0 + 2\delta_0 - \delta_1|}{2}\right)\delta_1}{|2\delta_1 - \delta_0 - \delta_1| + \frac{|2\delta_1 - \delta_0 + \delta_1|}{2} + |\delta_0 - 2\delta_0 - \delta_1| + \frac{|\delta_0 + 2\delta_0 - \delta_1|}{2}}$$

$$= \frac{\left(|\delta_1 - \delta_0| + \frac{|3\delta_1 - \delta_0|}{2}\right)\delta_0}{|\delta_1 - \delta_0| + \frac{|3\delta_1 - \delta_0|}{2} + |-\delta_0 - \delta_1| + \frac{|3\delta_0 - \delta_1|}{2}} + \frac{(|-\delta_0 - \delta_1| + \frac{|3\delta_0 - \delta_1|}{2})\delta_1}{|\delta_1 - \delta_0| + \frac{|3\delta_1 - \delta_0|}{2} + |-\delta_0 - \delta_1| + \frac{|3\delta_0 - \delta_1|}{2}}$$

$$d_2 = \frac{\left(|\delta_3 - \delta_2| + \frac{|\delta_3 + \delta_2|}{2}\right)\delta_1}{|\delta_3 - \delta_2| + \frac{|\delta_3 + \delta_2|}{2} + |\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2}}$$

$$+ \frac{\left(|\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2}\right)\delta_2}{|\delta_3 - \delta_2| + \frac{|\delta_3 + \delta_2|}{2} + |\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2}}$$

$$= \frac{\left(|3\delta_1 - 2\delta_0 - 2\delta_1 - \delta_0| + \frac{|3\delta_1 - 2\delta_0 + 2\delta_1 - \delta_0|}{2}\right)\delta_1}{|3\delta_1 - 2\delta_0 - 2\delta_1 - \delta_0| + \frac{|3\delta_1 - 2\delta_0 + 2\delta_1 - \delta_0|}{2} + |\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2}} +$$
$$\frac{\left(|\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2}\right)(2\delta_1 - \delta_0)}{|3\delta_1 - 2\delta_0 - 2\delta_1 - \delta_0| + \frac{|3\delta_1 - 2\delta_0 + 2\delta_1 - \delta_0|}{2} + |\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2}}$$

$$= \frac{\left(|\delta_1 - 3\delta_0| + \frac{|5\delta_1 - 3\delta_0|}{2}\right)\delta_1}{\left(|\delta_1 - 3\delta_0| + \frac{|5\delta_1 - 3\delta_0|}{2}\right) + |\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2}} + \frac{\left(|\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2}\right)(2\delta_1 - \delta_0)}{\left(|\delta_1 - 3\delta_0| + \frac{|5\delta_1 - 3\delta_0|}{2}\right) + |\delta_1 - \delta_0| + \frac{|\delta_1 + \delta_0|}{2}}$$

And we can get that

$$H_4(x)=\left[\left(1-2(x-x_0)\left(\frac{1}{x_0-x_1}+\frac{1}{x_0-x_2}\right)\right)\left[\left(\frac{x-x_0}{x_0-x_1}\right)^2+\left(\frac{x-x_0}{x_0-x_2}\right)^2\right]y_0+(x-x_0)\left[\left(\frac{x-x_0}{x_0-x_1}\right)^2+\left(\frac{x-x_0}{x_0-x_2}\right)^2\right]\left[\frac{\left(|\delta_1-\delta_0|+\frac{|\delta_1+\delta_0|}{2}\right)(2\delta_0-\delta_1)}{|\delta_1-\delta_0|+\frac{|\delta_1+\delta_0|}{2}+|-\delta_0-3\delta_1|+\frac{|5\delta_0-3\delta_1|}{2}}+\frac{\left(|-\delta_0-3\delta_1|+\frac{|5\delta_0-3\delta_1|}{2}\right)\delta_0}{|\delta_1-\delta_0|+\frac{|\delta_1+\delta_0|}{2}+|-\delta_0-3\delta_1|+\frac{|5\delta_0-3\delta_1|}{2}}\right]+\left[\left(1-2(x-x_1)\left(\frac{1}{x_1-x_0}+\frac{1}{x_1-x_2}\right)\right)\left[\left(\frac{x-x_1}{x_1-x_0}\right)^2+\left(\frac{x-x_1}{x_1-x_2}\right)^2\right]y_1+(x-x_1)\left[\left(\frac{x-x_1}{x_1-x_0}\right)^2+\left(\frac{x-x_1}{x_1-x_2}\right)^2\right]\left[\frac{\left(|\delta_1-\delta_0|+\frac{|3\delta_1-\delta_0|}{2}\right)\delta_0}{|\delta_1-\delta_0|+\frac{|3\delta_1-\delta_0|}{2}+|-\delta_0-\delta_1|+\frac{|3\delta_0-\delta_1|}{2}}+\frac{\left(|-\delta_0-\delta_1|+\frac{|3\delta_0-\delta_1|}{2}\right)\delta_1}{|\delta_1-\delta_0|+\frac{|3\delta_1-\delta_0|}{2}+|-\delta_0-\delta_1|+\frac{|3\delta_0-\delta_1|}{2}}\right]+\left[\left(1-2(x-x_2)\left(\frac{1}{x_2-x_0}+\frac{1}{x_2-x_1}\right)\right)\left[\left(\frac{x-x_2}{x_2-x_0}\right)^2+\left(\frac{x-x_2}{x_2-x_1}\right)^2\right]y_2+(x-x_2)\left[\left(\frac{x-x_2}{x_2-x_0}\right)^2+\left(\frac{x-x_2}{x_2-x_1}\right)^2\right]\left[\frac{\left(|\delta_1-3\delta_0|+\frac{|5\delta_1-3\delta_0|}{2}\right)\delta_1}{\left(|\delta_1-3\delta_0|+\frac{|5\delta_1-3\delta_0|}{2}\right)+|\delta_1-\delta_0|+\frac{|\delta_1+\delta_0|}{2}}+\frac{\left(|\delta_1-\delta_0|+\frac{|\delta_1+\delta_0|}{2}\right)(2\delta_1-\delta_0)}{\left(|\delta_1-3\delta_0|+\frac{|5\delta_1-3\delta_0|}{2}\right)+|\delta_1-\delta_0|+\frac{|\delta_1+\delta_0|}{2}}\right]$$

with

$$\delta_0=\frac{y_1-y_0}{x_1-x_0}$$

$$\delta_1=\frac{y_2-y_1}{x_2-x_1}$$

As it can be seen, it has been too difficult to calculate, so this project only takes two points and three points into consideration of getting polynomial.

Conclusion

To sum up, the MAKIMA interpolation mainly has three advantages:

- 1. Compared with pchip, it produces undulations which are not aggressive as pchip from decreasing undulation, and compared with spline it also prevents part of undulations.
    - 2. Compared with AKIMA interpolation, the MAKIMA interpolation algorithm solved the negligible difference between slopes of the interval if the lower and the higher slopes of the line are equal and di when the lower and the higher slopes are nearly equal of AKIMA.

- 3. The MAKIMA interpolation decreases the undulation compared with AKIMA interpolation if the lower and the higher slopes are both equal ($\delta_{i-2}=\delta_{i-1}$, $\delta_i=\delta_{i+1}$) and the higher slopes are not equal to the lower slopes.

And it is not a good way to use the choice of derivatives (slope of the curve) in makima algorithm, because it is too complicated when the data is more than two points.

Reference List：

1. E. Waring. Problems Concerning Interpolations. Philosophical Transactions of the Royal Society of London. 1779, 69: 59–67.
2. E. Meijering. *A chronology of interpolation: From ancient astronomy to modern signal and image processing,*. Proceedings of the IEEE: 323.
3. Julius Orion Smith III. Lagrange Interpolation. Center for Computer Research in Music and Acoustics (CCRMA), *Stanford University*.
4. *H. Akima, "A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures", JACM, v. 17-4, p.589-602, 1970.*
5. Retrieved from: Mathworks.com/help/matlab/ref/makima.html.

   Retrieved from: https://blogs.mathworks.com/cleve/2019/04/29/makima-piecewise-cubic-interpolation/#53876a13-2ec0-4332-b281-370e04da1d6b