浙江大學

Project

课程名称:	数值分析方法
姓　　名:	张友超
学　　号:	3170100125
指导教师:	余官定

目录

问题 1、

给定采样点（1, 0.1051）,(4, 0.1827) , (7, 0.0511) , (10, 0.0043)，至少采用两种方案对该曲线进行重构。报告中应给出重构的关键步骤以及最终结果

方法 1：牛顿插值法

关键步骤：

```
def Netwon(X,Y):
    n=len(X)
    #先将表格制作出来
    A=np.zeros([n,n])
    for i in range(0,n):
        A[i][0] = Y[i]
    for j in range(1,n):
        for i in range(j,n):
            A[i][j] = (A[i][j-1] - A[i-1][j-1])/(X[i]-X[i-j])
    #创建函数表达式
    P=Y[0]
    x=sympy.Symbol('x')#创建符号变量
    for i in range(1,n):
        w=1
        for j in range(0,i):
            w*=(x-X[j])
        P+=w*A[i][i]
    return P#可以输出表达式
```

$$P(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2)$$

最终结果：

$$P(x) = 0.0018148145\mathrm{X}^3 - 0.0334\mathrm{X}^2 + 0.154755\mathrm{X} - 0.018070370$$

放法 2：拉格朗日插值法

关键步骤：

```
def get_L(xk,x_=[]):#计算 Lnk(x)=(x-x0)...(x-xn)/(xk-x0)...(xk-xn)
    def L_k(x):
        numerator=1.0
        dominator=1.0
        for i in range(len(x_)):
            if x_[i]!=xk:
                dominator*=xk-x_[i]
                numerator*=x-x_[i]
        return numerator/dominator
    return L_k
def get_Lagrange(x_=[],y_=[]):#获得拉格朗日插值函数
```

```
    x=sympy.Symbol('x')#创建符号变量
    result=0.0
    for i in range(len(x_)):
        pass
        result+=y_[i]*get_L(x_[i],x_)(x)
    return result
```

$$L_{4,k}(x)=\frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}$$

$$P(x)=f(x_0)L_{4,0}(x)+\cdots+f(x_n)L_{4,4}(x)$$

最终结果:

$$P(x)=0.0018148145\mathrm{X}^3\ -\ 0.0334\mathrm{X}^2\ +\ 0.154755\mathrm{X}\ -\ 0.018070370$$

方法 3: 三次样条插值法:

关键步骤:

```
def Cubic(x_given,y_given):#直接按照书本上伪代码写
    n=len(x_given)
    a=np.zeros(n,dtype=float)
    b=np.zeros(n,dtype=float)
    c=np.zeros(n,dtype=float)
    d=np.zeros(n,dtype=float)
    s=np.zeros(n-1,dtype=tuple)#分段函数
    h=np.zeros(n-1,dtype=float)
    m=np.zeros(n-1,dtype=float)#α
    m[0]=0
    x=sympy.symbols('x')
    for i in range(0,n):
        a[i]=y_given[i]
    for i in range(0,n-1):
        h[i]=x_given[i+1]-x_given[i]
    for i in range(1,n-1):
        m[i]=3*(a[i+1]-a[i])/h[i]-3*(a[i]-a[i-1])/h[i-1]
    l=np.zeros(n,dtype=float)
    u=np.zeros(n,dtype=float)
    z=np.zeros(n,dtype=float)
    l[0],u[0],z[0]=1,0,0
    for i in range(1,n-1):
        l[i]=2*(x_given[i+1]-x_given[i-1])-h[i-1]*u[i-1]
        u[i]=h[i]/l[i]
        z[i]=(m[i]-h[i-1]*z[i-1])/l[i]
    l[n-1],z[n-1],c[n-1]=1,0,0
    for j in range(n-2,-1,-1):
        c[j]=z[j]-u[j]*c[j+1]
```

```
        b[j]=(a[j+1]-a[j])/h[j]-h[j]*(c[j+1]+2*c[j])/3
        d[j]=(c[j+1]-c[j])/(3*h[j])
    for i in range(0,n-1):
        s[i]=a[i]+b[i]*(x-x_given[i])+c[i]*(x-x_given[i])**2+d[i]*(x-x_given[i])**3
    return s#分段函数
```
最终结果:

$$\begin{cases} S_1(x) = 0.04634667x - 0.002275556(x-1)^3 + 0.058753333, & 1 < x < 4 \\ S_2(x) = 0.0036296x^3 - 0.06403556x^2 + 0.32296889x - 0.31690296, & 4 < x < 7 \\ S_3(x) = -0.0013540741x^3 + 0.04062x^2 - 0.4096356x + 1.3925, & 7 < x < 10 \end{cases}$$

方法 4: 埃米尔特插值法:

关键步骤:(坐标点作为形参)

```
from scipy.interpolate import KroghInterpolator
H = KroghInterpolator(x, y)#埃米尔特插值(坐标点作为形参)
```
最终结果:该方法得不到表达式,但是可以返回一个函数

拟合图像:

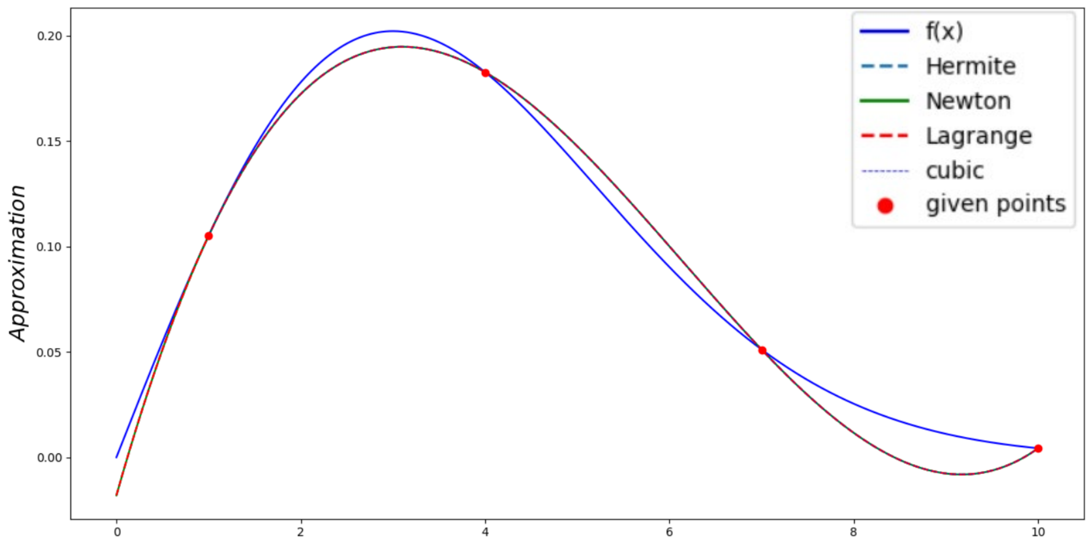


图 1.函数拟合曲线

可以看到,牛顿插值法和拉格朗日插值法得到的表达式相同,这是因为两者都是通过给定 n+1 个互异的插值节点,让你求一条 n 次代数曲线近似地表示待插值的函数曲线.

这就叫做代数插值. Lagrange 插值代数和 Newton 法插值都属于代数插值的范畴.

Lagrange 插值和 Newton 法插值的结果和余项都是一致的,因为都是利用 n 次多项式插值,所以一样.

区别:Lagrange 插值法是通过构造 n+1 个 n 次基本多项式,然后线性组合(结果当然也是 n 次的多项式)而得到的.而 Newton 法插值是通过求各阶差商,递推得到的一个 f(x)=f(x0)+(x-x0)f[x0,x1]+(x-x0)(x-x1)f[x0,x1,x2]....+(x-x0)...(x-x(n-1))f[x0,x1...xn]这样的公式,

代进去就可以得到．

另外，四种拟合方法得到的图像基本相同。

问题 2.

实际上，上述概率密度曲线对应于参数为 3 的瑞利分布，即 $f(x)=\frac{x}{\sigma^2}\exp(-\frac{x^2}{2\sigma^2})$ ，其中 $\sigma=3$ （这里我们限制 $x\in[0,10]$）。计算并比较（1）中结果在 L2 范数(求积分)下的误差。如果采用 L_∞ 范数（差的绝对值最大）计算误差，结果又会怎样?

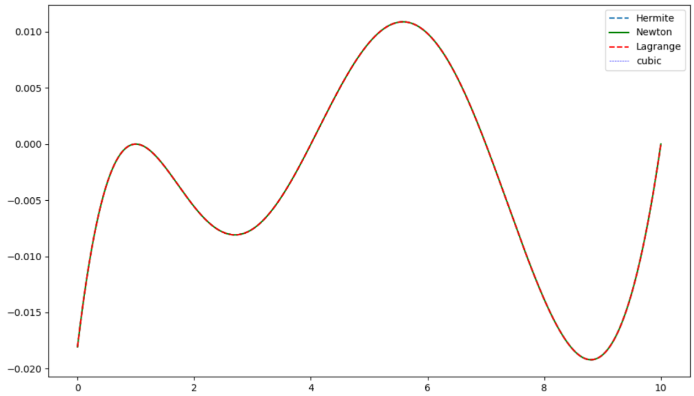
L2 范数:

Lagrange L_2: 0.000878331778560006

Netwon L_2: 0.0008783317785600052

Cubic L_2: 0.0008214155424958413

Hermite L_2: 0.000878331778560006



绝对误差曲线

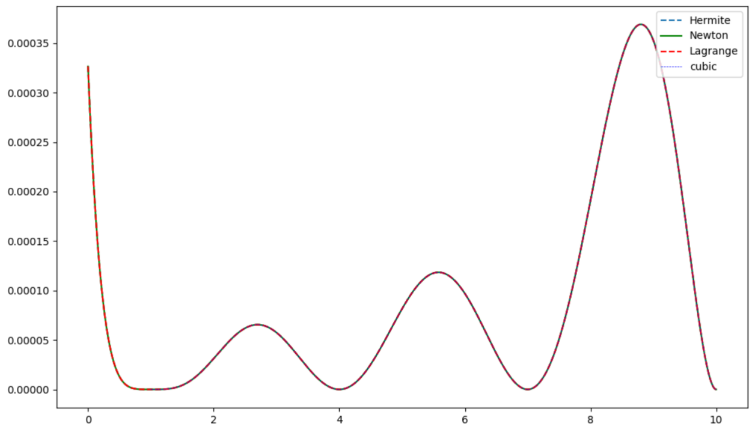
L∞范数:

Lagrange L_∞: 0.019209604129851177

Netwon L_∞: 0.019209604129851163

Cubic L∞: 4.533178363547163e − 06

Hermite L∞: 0.019209604129851177



误差平方曲线

问题 3、

（1）中给出的数据点是最优的吗？如果不是，应该如何选取用于重构曲线的数据点？请给出你的设计思路与解决方案。

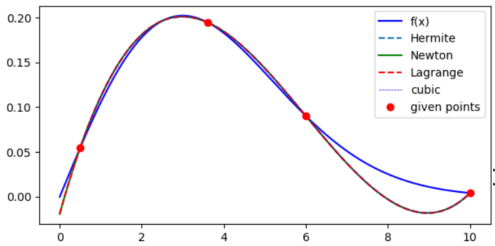
设计思路:

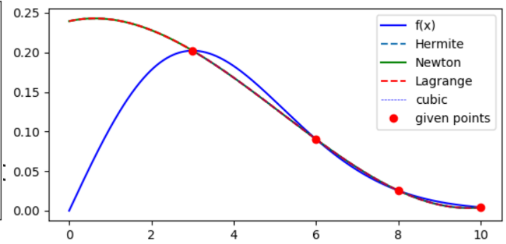
我认为 L_2 范数更能代表函数的拟合效果，对应的是绝对误差的平方图像。

观察误差平方的图像可以发现，误差主要集中在[0,1],[2,3],[5,6],[8,10],区间内部，这是因为所给的采样点（1, 0.1051）,(4, 0.1827) , (7, 0.0511) , (10, 0.0043)不能更好的代表概率密度曲线对应于参数为 3 的瑞利分布函数在[0,10]区间的特征。

1.我认为重要特征包括采样点处原始函数的导数，以及采样点一定邻域区间内导数符号，以及相邻采样点之间的导数关系。

2.另一点是所给的采样点的离散程度的问题，虽然 xi 是均匀离散，但是 yi 不是，这些点的相对离散度共同决定了拟合效果。



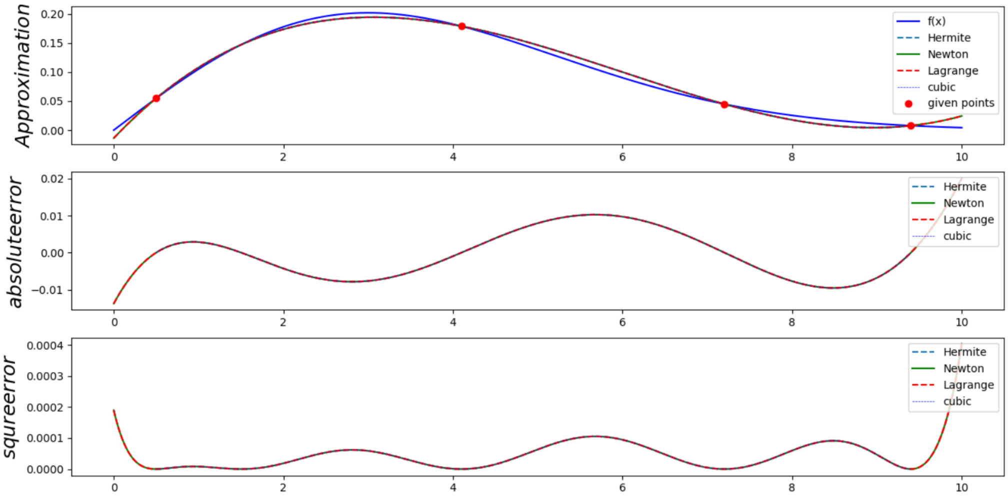


观察这两幅图，第一幅图在[0,6]先增后减区间中误差较小，但是在[6,8]缓和递减区间中误差较大；第二幅图则反之。这很好的反映了如果在某一区间内采样点越多，则误差越小的规律。

然而误差是一定存在的，在[0,10]这个大区间内，我们只有四个采样点，所以采样点的选取必须结合函数的走势，才能确保误差最小。

解决方案:

我新选取的点是(0.5，0.05478)，(4.1,0.1790), (7.2,0.0449),(10,0.0077),如图所示:



这是新的四个采样点的拟合图以及绝对误差与平方误差的走势图, 对比原来采样点可以发现在[0,1],[2,3],[5,6],[8,9]区间内的平方误差都有所减小。

L2 范数:

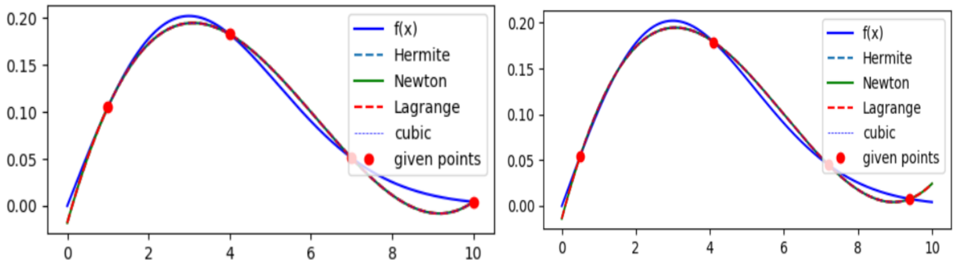
Lagrange L_2: 0.0004422760138568671

Netwon L_2: 0.0004422760138568672

Cubic L_2: 0.00034959175917540983

Hermite L_2: 0.000878331778560006

这是因为新选取的点 9.4 与 10 相比大大减小了[8,10]区间的误差，0.5 与 1 相比减小了[0，1]区间内的误差。这样就大大减小了 L_2 范数。



前后拟合效果对比图

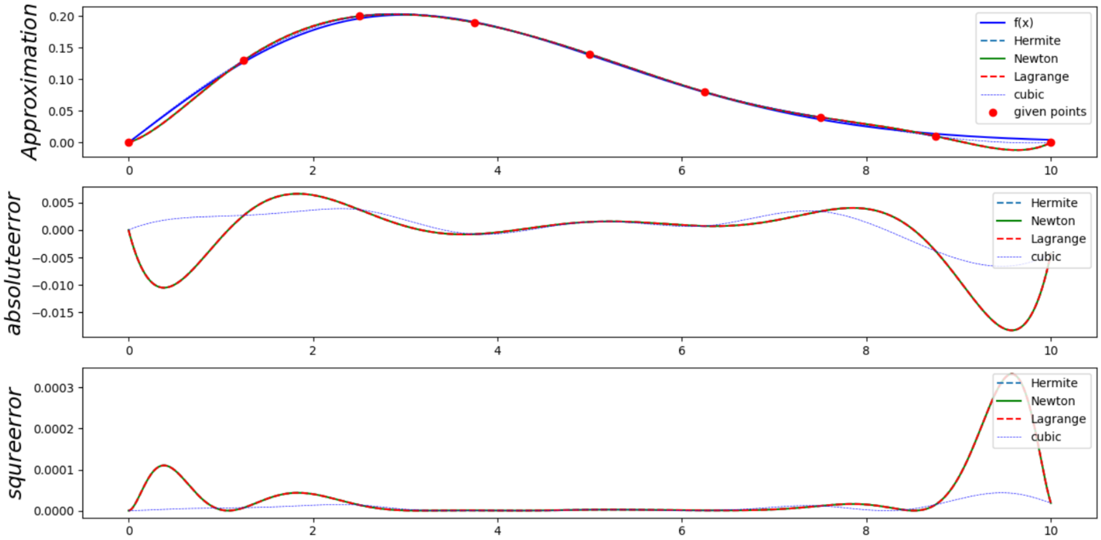
问题 4.

设计传输协议

设计思路:

1.在 60bit 通信容量的限制下，我们需要在采样点的个数与精度之间选择即采样点个数越少，那么采样点的精度越高（有效位数越多）；采样点的个数越多，那么可以用于拟合的点约多，但是精度就越低（有效位数越少）。接下来验证:

取 x=[0,1,2,3,4,5,6,7,8,9,10]，Y 的有效位数为两位小数，对第三位四舍五入



L2 范数：

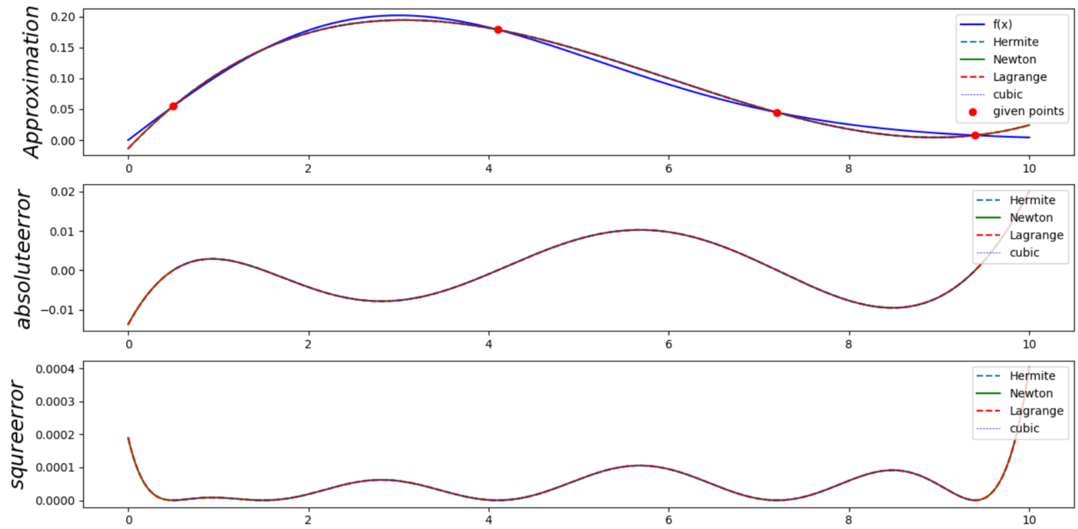
$$\text{Lagrange } L\_2: 0.0003475412481046549$$

$$\text{Netwon } L\_2: 0.00034754124810465784$$

$$\mathbf{Cubic\ L\_2: 8.298500909024016e-05}$$

$$\text{Hermite } L\_2: 0.0003475412481046549$$

取 x = [0.5,4.1,7.2,9.4]，Y 的有效位数为 4 位小数，对第 5 位四舍五入



L2 范数：

$$\text{Lagrange } L\_2: 0.000878331778560006$$

$$\text{Netwon } L\_2: 0.0008783317785600052$$

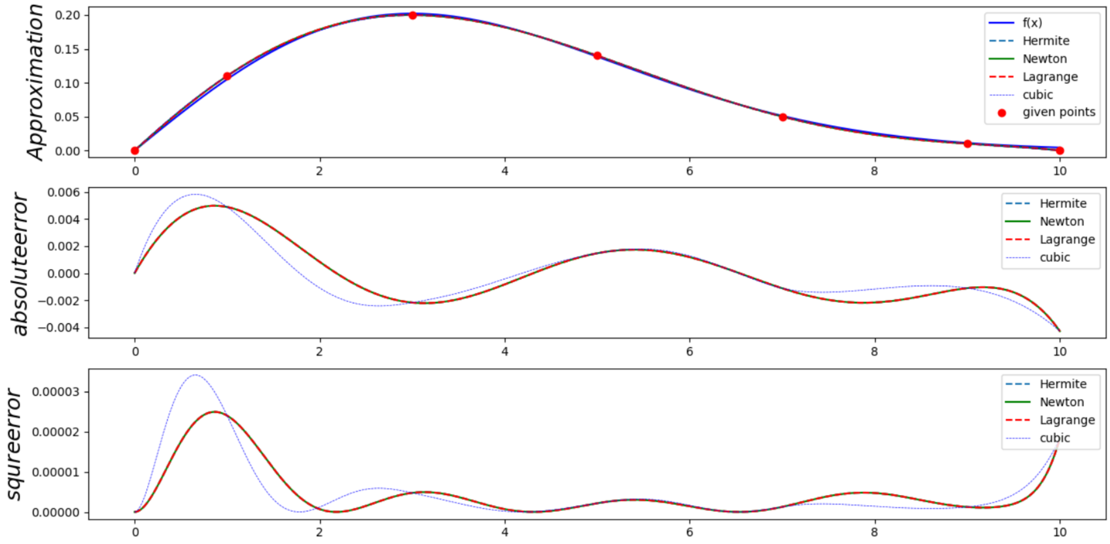
$$\mathbf{Cubic\ L\_2: 0.0008214155424958413}$$

$$\text{Hermite } L\_2: 0.000878331778560006$$

到这里可以证明采样点越多提高拟合精度的能力，比单纯的提高采样点有效位数提高精度的能力要强，特别是对于采用**三次样条分段插值法**（图中蓝色虚线）精度的提高更加明显。

2.进一步思考，不仅是采样点的个数越多拟合效果越好，所选的采样点还必须能够代表原始函数的特征，并且离散度的影响也必须考虑。

经过尝试，可以找到 x=[0, 1, 3, 5, 7.5, 9, 10]，Y 保留两位有效位数



L2 范数:

$$\text{Lagrange L\_2}: 4.6256406201437164e-05$$

$$\text{Netwon } L_2: 4.6256406201437137e-05$$

$$\text{Cubic } L_2: 4.926241388507842e-05$$

$$\text{Hermite L\_2}: 4.6256406201437164e-05$$

可以发现，这似乎是很完美的数据点的选择，所以折中考虑采样点的个数，采样点的精度，与数据离散度，再结合 60bit 的，可得到一个复杂的方案，但是这种方案不具有普适性，因为会加上很多的条件，比如：

条件 / 采样点	**X(x∈0~8)整数部分 3bit**	**Y 小数部分前两位 5bit(默认整数为 0)**
1	0	.00
2	1	.11
3	3	.20
4	5	.14
5	7	.05
	X(x∈9~10)整数部分 4bit	**Y 小数部分前两位 5bit(默认整数为 0)**
6	9	.01
7	10	.00

所需通信容量为：

$$3\times5+4\times2+5\times7=58bit<60$$

发送格式为：

000,00000,001,01011,011,10100,101,01110,111,00101,**00**,1001,00001,1010,0000

| 0 .00 | 1 .11 | 3 .20 | 5 .14 | 7 .05 || 9 .01 | 10 .00|

4.为考虑普适性，选择多采样点，均匀离散的方案，如下：

0,1010,XXXXX,XXXXX,XXXXX,XXXXX,XXXXX,XXXXX,XXXXX,XXXXX,XXXXX,XXXXX,XXXXX
[0, 10], Y_0 Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7 Y_8 Y_9 Y_{10}
解释:

1).前五位传的是 X 的范围，[0,10]只取整数即[0,1,2,3,4,5,6,7,8,9,10]

2).后 55 位传的是 Y（0~10）的值，小数部分前两位（默认整数为 0）

甲发送:

0,1010,00000,01011,10010,01110,10100,10010,01110,01001,00101,00011,00000

乙接收后得到:

X	0	1	2	3	4	5	6	7	8	9	10
Y	.00	.11	.18	.20	.18	.14	.09	.05	.03	.01	.00

重构结果:（三次样条函数）

$$S(0) = x*(0.1178 - 0.00781*x**2)$$
$$S(1) = -0.000959*x**3 - 0.02058*x**2 + 0.13839*x - 0.00686$$
$$S(2) = 0.001609*x**3 - 0.03593*x**2 + 0.1691*x - 0.02733$$
$$S(3) = 0.0045109*x**3 - 0.06210*x**2 + 0.24743*x - 0.10567$$
$$S(4) = 0.000346*x**3 - 0.01207*x**2 + 0.0475*x + 0.1608$$
$$S(5) = 0.004103*x**3 - 0.06842*x**2 + 0.3293*x - 0.3087$$
$$S(6) = 0.00324*x**3 - 0.05289*x**2 + 0.236*x - 0.1223$$
$$S(7) = -0.00706*x**3 + 0.1635*x**2 - 1.2786*x + 3.4121$$
$$S(8) = 0.005017*x**3 - 0.12645*x**2 + 1.0409*x - 2.7735$$
$$S(9) = -0.0030*x**3 + 0.09010*x**2 - 0.9080*x + 3.07335$$

L2 范数:

$$\mathbf{Cubic\ L\_2: 8.298500909024016e-05}$$

图像: