线性拟合与插值

Author: [dh768154]

Link: [https://zhuanlan.zhihu.com/p/698543986]

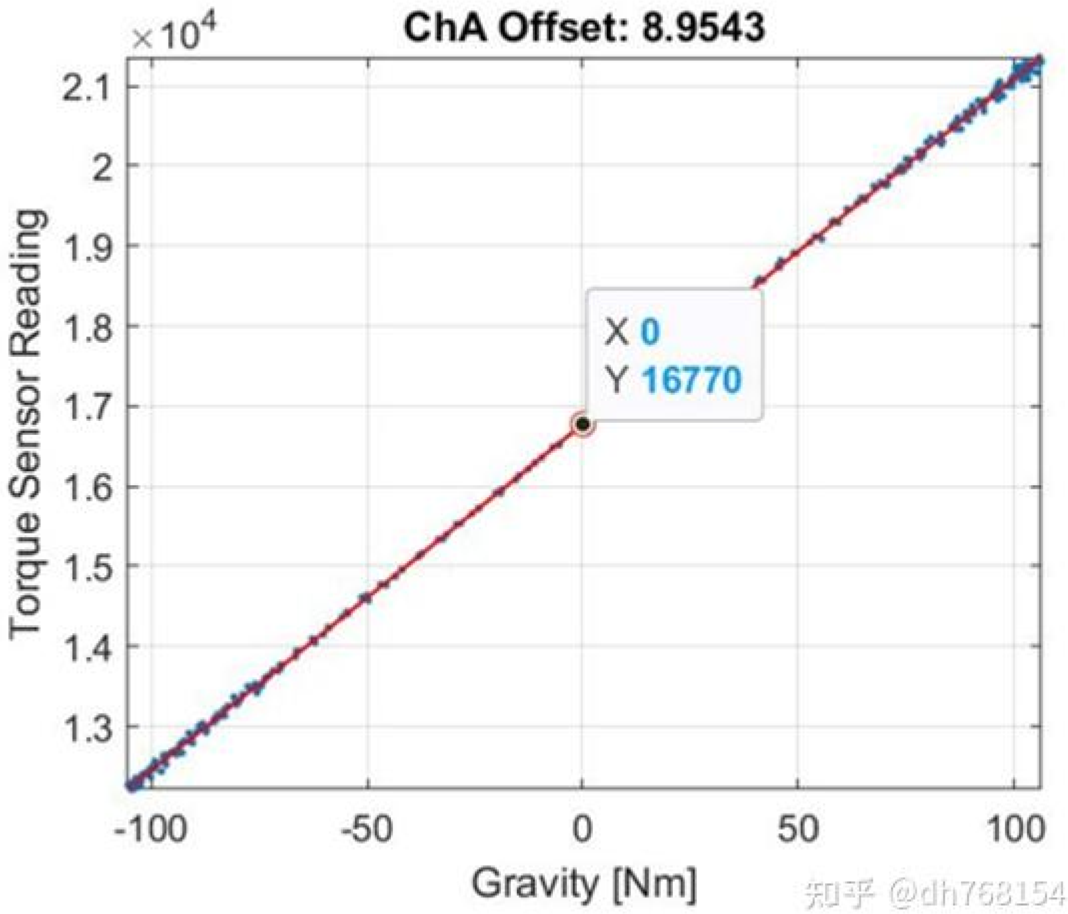
Demo

> "Talk is cheap. Show me the code." - Linus Torvalds

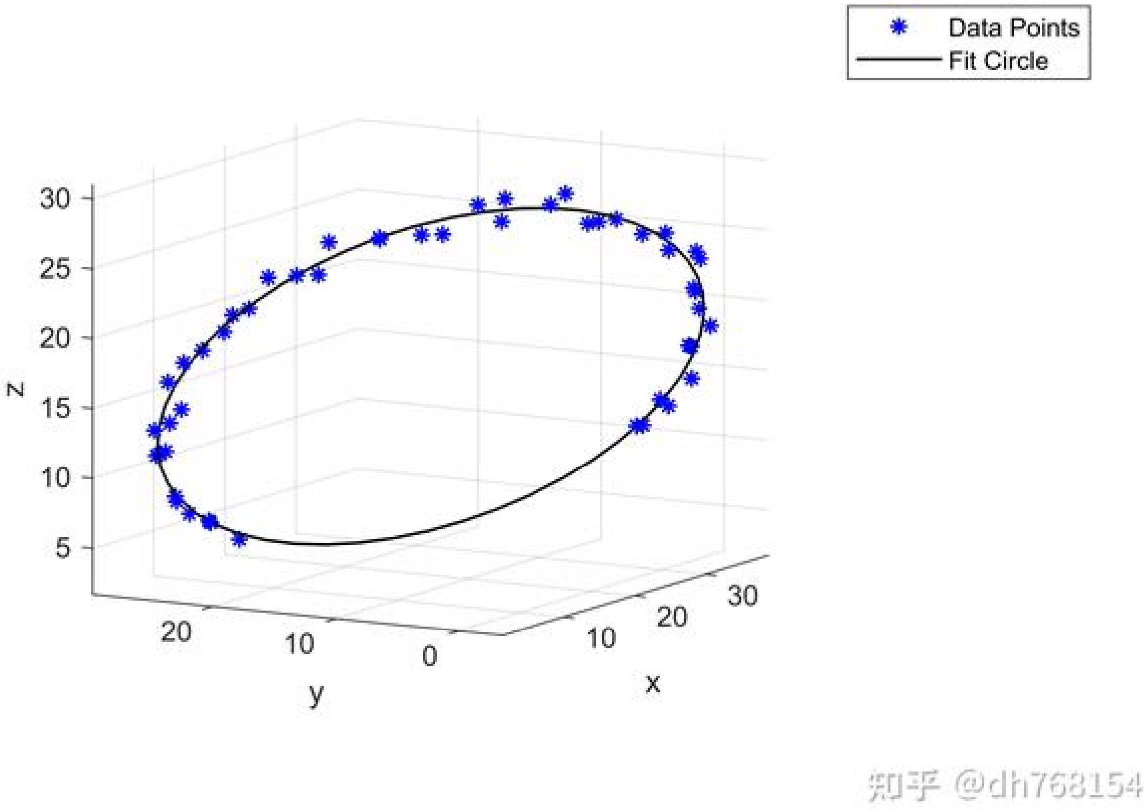
https://www.mathworks.com/matlabcentral/fileexchange/166646-linear-fit

---

线性拟合和插值在机器人标定中用得非常多，包括扭矩传感器标定，参数标定，计算精度等等。



扭矩传感器标定



拟合空间圆，计算精度

线性插值不是指只能在直线上插值，而是指可以写成 $y = basis \cdot c$ 的形式的函数。

比如多项式：

$y\left(x\right) = c_0 \cdot x^0 + c_1 \cdot x^1 + c_2 \cdot x^2 \cdots c_n \cdot x^n$

写成 $y = basis \cdot c$ 的形式：

$$y\left(x_i\right) = \begin{bmatrix} {x_i}^0 & {x_i}^1 & {x_i}^2 & \cdots & {x_i}^n \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

再比如2个频率构成的三角函数：

$y\left(x\right) = A \cdot \sin\left(\omega_1\ x + \phi_1\right) + B \cdot \sin\left(\omega_2\ x + \phi_2\right) + C$

等价于：

$y\left(x\right) = c_0 + c_1 \cdot \sin\left(\omega_1\ x\right) + c_2 \cdot \cos\left(\omega_1\ x\right) + c_3 \cdot \sin\left(\omega_2\ x\right) + c_4 \cdot \cos\left(\omega_2\ x\right)$

写成 $y = basis \cdot c$ 的形式：

$$y\left(x_i\right)=\left[1 \quad \sin\left(\omega_1\ x_i\right) \quad \cos\left(\omega_1\ x_i\right) \quad \sin\left(\omega_2\ x_i\right) \quad \cos\left(\omega_2\ x_i\right)\right]\cdot\begin{bmatrix}c_0\\c_1\\c_2\\c_3\\c_4\end{bmatrix}$$

一维拟合与插值

在2维空间中插值，给一组x和y用于参数拟合，要求的是一组未知的x对应的y值。

以多项式插值为例，一组数据有许多个数据点x和y，那之前的等式可以写成：

$$\vec{y}=\left[\vec{x}^0 \quad \vec{x}^1 \quad \vec{x}^2 \quad \cdots \quad \vec{x}^n\right]\cdot\begin{bmatrix}c_0\\c_1\\c_2\\\vdots\\c_n\end{bmatrix}$$

写得详细些就是：

$$\begin{bmatrix}y_1\\y_2\\\vdots\\y_{\text{end}}\end{bmatrix}=\begin{bmatrix}{x_1}^0 & {x_1}^1 & {x_1}^2 & \cdots & {x_1}^n\\{x_2}^0 & {x_2}^1 & {x_2}^2 & \cdots & {x_2}^n\\\vdots & \vdots & \vdots & \ddots & \vdots\\{x_{\text{end}}}^0 & {x_{\text{end}}}^1 & {x_{\text{end}}}^2 & \cdots & {x_{\text{end}}}^n\end{bmatrix}\cdot\begin{bmatrix}c_0\\c_1\\c_2\\\vdots\\c_n\end{bmatrix}$$

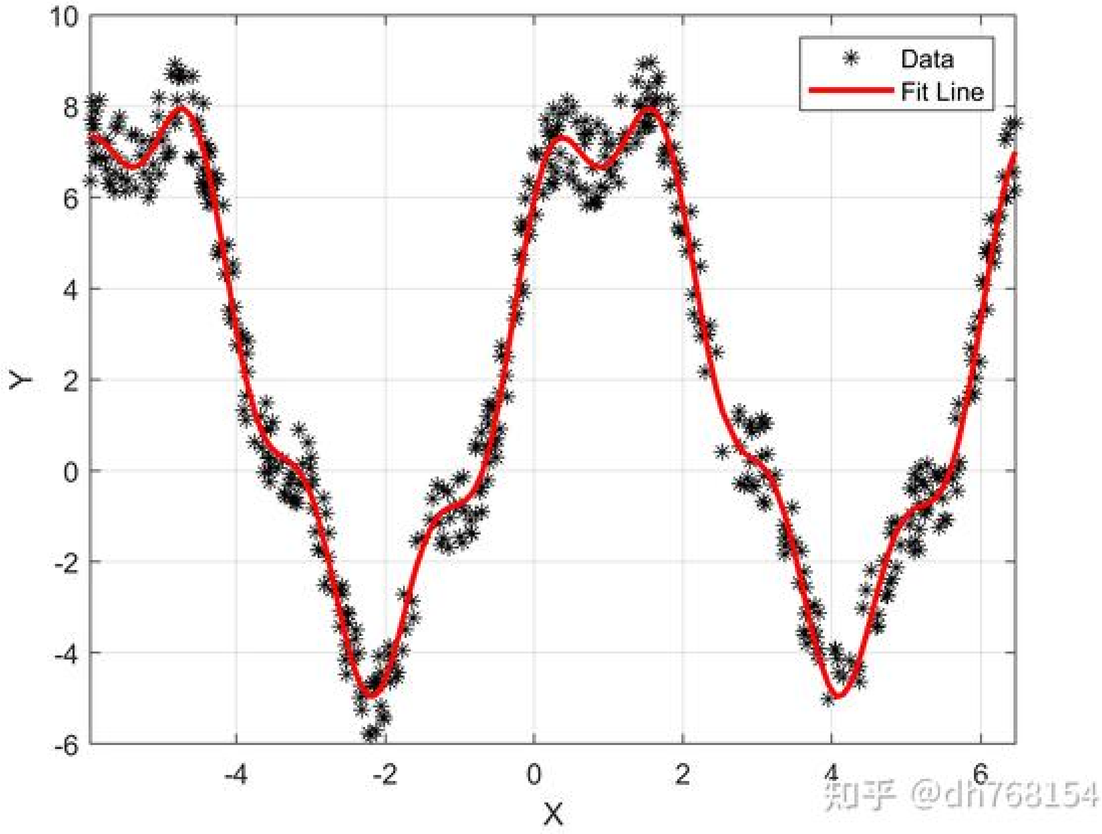
其中 $\left[\vec{x}^0 \quad \vec{x}^1 \quad \vec{x}^2 \quad \cdots \quad \vec{x}^n\right]$ 被称为basis。

同样道理，三角函数拟合的话，basis就是：

$$\begin{aligned}&\left[1 \quad \sin\left(\omega_1\ \vec{x}\right) \quad \cos\left(\omega_1\ \vec{x}\right) \quad \sin\left(\omega_2\ \vec{x}\right) \quad \cos\left(\omega_2\ \vec{x}\right)\right]\\=&\begin{bmatrix}1 & \sin\left(\omega_1\ x_1\right) & \cos\left(\omega_1\ x_1\right) & \sin\left(\omega_2\ x_1\right) & \cos\left(\omega_2\ x_1\right)\\1 & \sin\left(\omega_1\ x_2\right) & \cos\left(\omega_1\ x_2\right) & \sin\left(\omega_2\ x_2\right) & \cos\left(\omega_2\ x_2\right)\\\vdots & \vdots & \vdots & \vdots & \vdots\\1 & \sin\left(\omega_1\ x_{\text{end}}\right) & \cos\left(\omega_1\ x_{\text{end}}\right) & \sin\left(\omega_2\ x_{\text{end}}\right) & \cos\left(\omega_2\ x_{\text{end}}\right)\end{bmatrix}\end{aligned}$$

要得到未知参数，可以用Matlab“\”或者pinv函数：

```
A = [sin(omega(1)*x),cos(omega(1)*x),...
sin(omega(2)*x),cos(omega(2)*x),ones(length(x),1)];
b = y;
P_fit = A\b
```



2维3维拟合与插值

二维拟合插值道理是一样的，用于3维数据，给出x和y的值，需要知道z的值。

如果x方向拟合次数是m，y方向拟合次数是n，等式可以写成：

$$z_i = \begin{bmatrix}1 & x_i & x_i^2 & y_i & x_i\,y_i & x_i^2\,y_i & y_i^2 & x_i\,y_i^2 & \cdots & x_i^m\,y_i^n\end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{(m+1)\times(n+1)} \end{bmatrix}$$

这里的basis其实就是x和y的所有可能的组合，可以通过Matlab符号工具箱得到所有的组合：

```
syms x y real
fit_order = [2,3];
A = x.^(0:fit_order(1));
B = y.^(0:fit_order(2));
basis = reshape(A'*B,1,[]);
disp('Basis : ');...
disp(basis);
```

```
Basis :
```

$$\left(1 \quad x \quad x^2 \quad y \quad x\,y \quad x^2\,y \quad y^2 \quad x\,y^2 \quad x^2\,y^2 \quad y^3 \quad x\,y^3 \quad x^2\,y^3\right)$$

拟合的话和1维一样，用"\"或者pinv函数：

```
A = xs.^(0:fit_order(1));
B = ys.^(0:fit_order(2));

pt_num = length(xs);
basis_c = arrayfun(@(i) reshape(A(i,:)'* B(i,:),1,[]),  1:pt_num,
'UniformOutput', false);
basis = cell2mat(basis_c');
```

这里用了arrayfun，其实用循环也一样。

```
basis = zeros(pt_num,prod(fit_order+1));
for i = 1:pt_num
    basis(i,:) = reshape(A(i,:)'*B(i,:),1,[]);
end
```

这里写的是一个通用方法，计算效率不高。如果是确定次数，建议用matlabFunction函数把符号工具箱计算出的basis导出成函数，然后就可以向量化计算了。这样避免循环和arrayfun

插值

有了参数，插值的话只需要根据给的新的x，y值构建basis，然后矩阵相乘就行了：

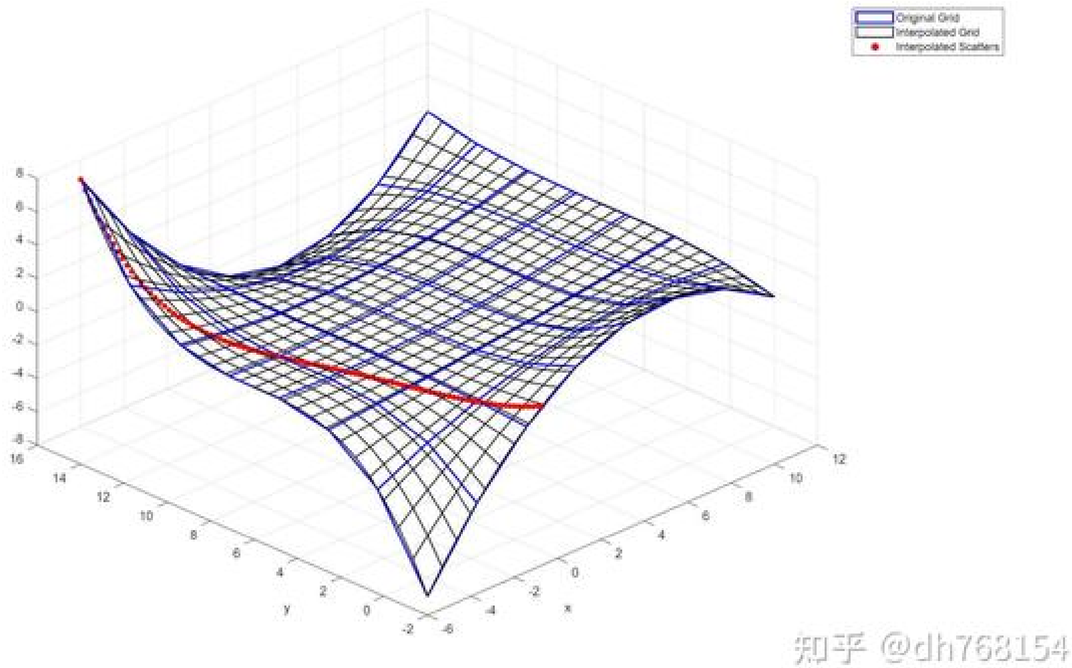
```
A = x1.^(0:fit_order(1));
B = y1.^(0:fit_order(2));
basis_c = arrayfun(@(i) reshape(A(i,:)'* B(i,:),1,[]), 1:numpt, 'UniformOutput',
false);
basis = cell2mat(basis_c');

z = basis*coeff;
```

如果需要插值的是一个曲面，可以用以下公式：

$$z_{\mathrm{ij}}=\begin{bmatrix}x_i^0 & x_i^1 & \cdots & x_i^m\end{bmatrix}\cdot\begin{bmatrix}c_{00} & c_{01} & \cdots & c_{0n}\\ c_{10} & c_{11} & \cdots & c_{1n}\\ \vdots & \vdots & \ddots & \vdots\\ c_{\mathrm{m}0} & c_{\mathrm{m}1} & \cdots & c_{\mathrm{mn}}\end{bmatrix}\cdot\begin{bmatrix}y_j^0\\ y_j^1\\ \vdots\\ y_j^n\end{bmatrix}$$

```
coeff= reshape(coeff,fit_order+1);
A = x1.^(0:order_fit(1));
B = y1.^(0:order_fit(2));
zz = (A*coeff*B')';
```



曲面插值与3维散点插值

3维拟合与插值

比如给地图坐标与海拔高度，插值温度。做法和2维几乎完全一样。

唯一的区别就是怎么得到x,y,z所有的组合：

```
syms z real
fit_order = [2,3,2];
C = z.^(0:fit_order(3));
basis = reshape(reshape(A'*B,1,[])'*C,1,[]);
disp('Basis (1:20) : ');...
disp(basis(1:20));...
disp('Basis (21:36) : ');...
disp(basis(21:36))
```

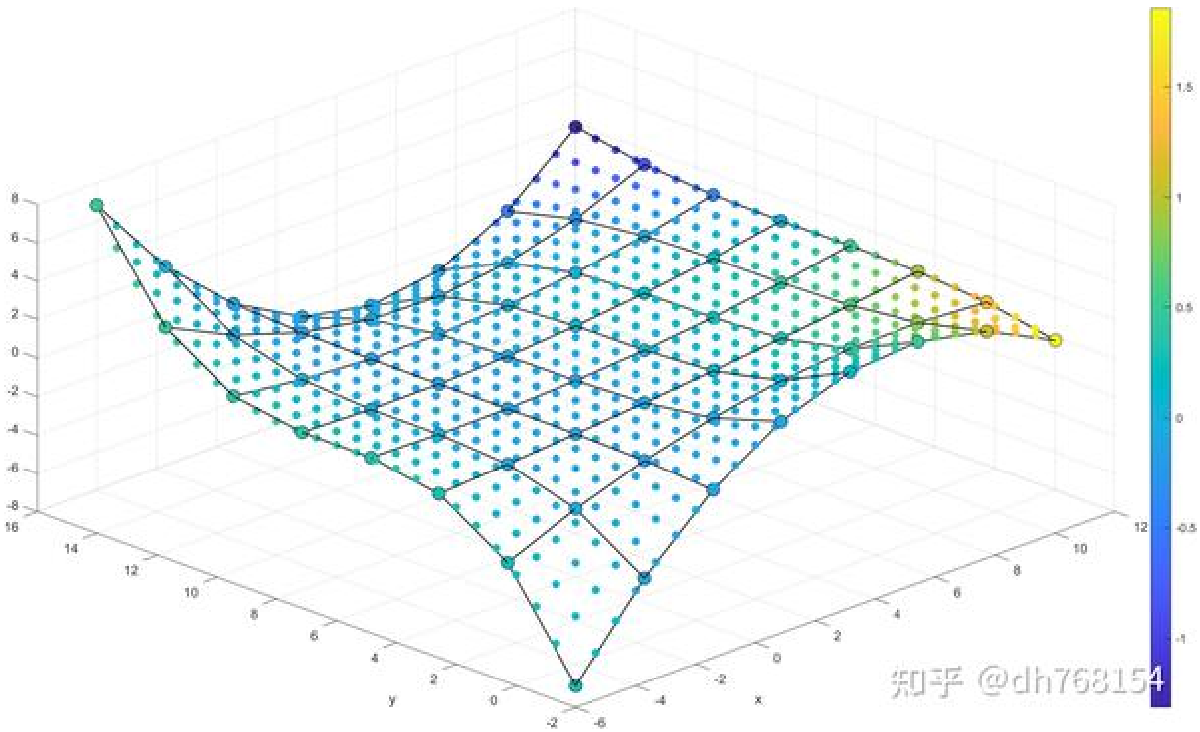
```
Basis (1:20) :
```

$$\left(1 \quad x \quad x^2 \quad y \quad x\,y \quad x^2\,y \quad y^2 \quad x\,y^2 \quad x^2\,y^2 \quad y^3 \quad x\,y^3 \quad x^2\,y^3 \quad z \quad x\,z \quad x^2\,z \quad y\,z \quad x\,y\,z \quad x^2\,y\,z \quad y^2\,z \quad x\,y^2\,z\right)$$

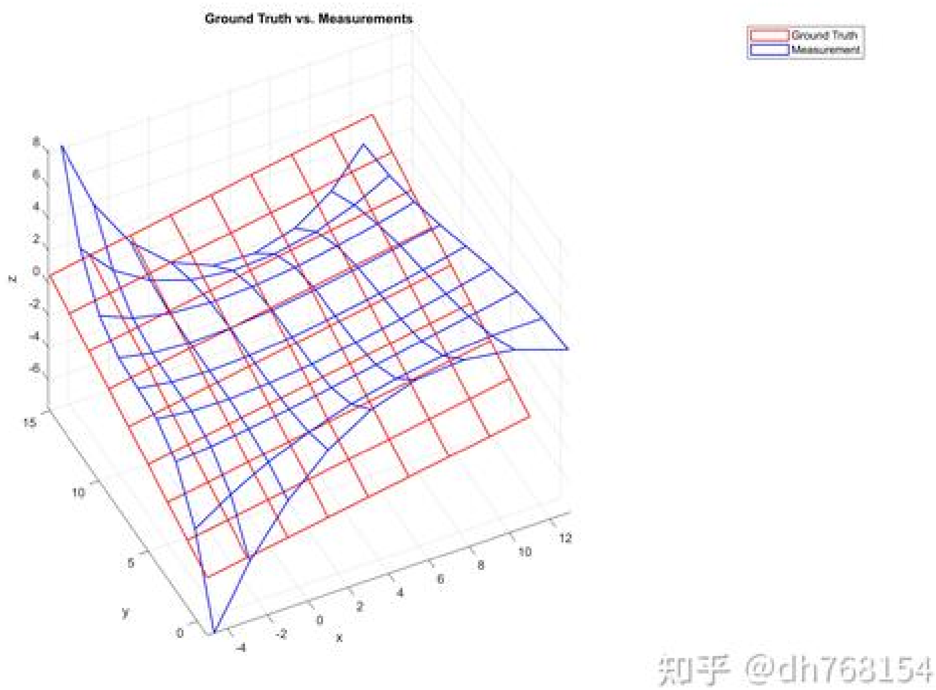
```
Basis (21:36) :
```

$$\left(x^2\,y^2\,z \quad y^3\,z \quad x\,y^3\,z \quad x^2\,y^3\,z \quad z^2 \quad x\,z^2 \quad x^2\,z^2 \quad y\,z^2 \quad x\,y\,z^2 \quad x^2\,y\,z^2 \quad y^2\,z^2 \quad x\,y^2\,z^2 \quad x^2\,y^2\,z^2 \quad y^3\,z^2 \quad x\,y^3\,z^2 \quad x^2\,y^3\,z^2\right)$$

如下图，黑色网格和黑色网格上点的颜色是原始数据，先用2维插值（3维空间）加密集网格，再在新的坐标上进行3维插值（获取颜色）。



3维插值稍微再拓展一下，可以给机器人做基于插值的精度校准，以后有机会说说：



Bezier and Bernstein Polynomial

多项式插值在高次数的情况下可能会不稳定，用Bernstein函数代替多项式的basis。

对于1维：

$$y(x)=B_{N,0}(x)\cdot c_0+B_{N,1}(x)\cdot c_1\cdots+B_{N,n}(x)\cdot c_n$$

其中：$B_{N,k}(x)=\binom{N}{k}\,(1-v)^{N-k}\,x^k$

$\binom{N}{k}$ 是N Choose k:

```
function b = nchoosek2(n,k)
% nchoosek, but works on k array
b = factorial(n)./factorial(n-k)./factorial(k);
end
```

这个Bernstein函数插值其实就是把多项式插值的basis $\begin{bmatrix} x^0 & x^1 & \cdots & x^n \end{bmatrix}$ 换成了 $B_{N,k}\left(x\right)$

对于2，3维：

$$Bx = \begin{bmatrix} B_0\left(x\right) & B_1\left(x\right) & B_2\left(x\right) & \cdots & B_n\left(x\right) \end{bmatrix}$$

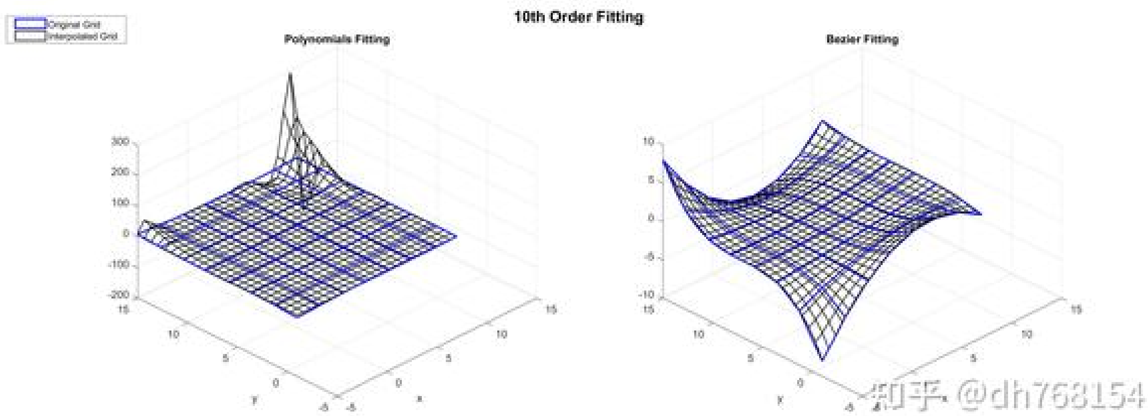
$$By = \begin{bmatrix} B_0\left(y\right) & B_1\left(y\right) & B_2\left(y\right) & \cdots & B_n\left(y\right) \end{bmatrix}$$

$$Bz = \begin{bmatrix} B_0\left(z\right) & B_1\left(z\right) & B_2\left(z\right) & \cdots & B_n\left(z\right) \end{bmatrix}$$

做法和3维多项式一模一样。

值得注意的是，Bernstein函数当自变量在0到1之间时效果比较好，因此在进行拟合前，先要对原数据缩放平移。需要插值的数据也要按照同样的参数缩放平移，完成插值后再变回去。

用10次多项式与10次Bernstein函数插值结果：



另外，新的点位需要在原先用于拟合的数据的范围中。比如用于拟合的数据是0~1之间，需要插值的数据是20就不合适了，如图：