In [1]:

```
#参考资料：https://blog.csdn.net/weixin_46382984/article/details/107727958
#导入库
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

见PPT49页

例 15.2　北方某城市 1986～1992 年道路交通噪声平均声级数据见表 15.1。

表 15.1　城市交通噪声数据[dB(A)]

序号	年份	L_{eq}	序号	年份	L_{eq}
1	1986	71.1	5	1990	71.4
2	1987	72.4	6	1991	72.0
3	1988	72.4	7	1992	71.6
4	1989	72.1			

In [2]:

```
data = np.array([71.1,72.4,72.4,72.1,71.4,72.0,71.6])
```

公式

(1) 求级比$\lambda(k)$

$$\lambda(k)=\frac{x^{(0)}(k-1)}{x^{(0)}(k)},$$

$$\lambda=(\lambda(2),\lambda(3),\cdots,\lambda(7))$$

In [3]:

```
# 求级比
rankData  =  np.array([ data[i]/data[i+1] for i in range(len(data)-1)])
rankData
```

Out[3]:

```
array([0.9820442 , 1.        , 1.00416089, 1.00980392, 0.99166667,
       1.00558659])
```

In [4]:

```
print(f"级比为范围为 {rankData.min()}~{rankData.max()}")
```

级比为范围为 0.9820441988950275~1.0098039215686272

In [5]:

```
#对数据进行一次求和
sumData = data.cumsum()
sumData
```

Out[5]:

```
array([ 71.1, 143.5, 215.9, 288. , 359.4, 431.4, 503. ])
```

公式

$$(2)\ \text{构造数据矩阵}B\text{及数据向量}Y$$

$$B=\begin{bmatrix} -\frac{1}{2}(x^{(1)}(1)+x^{(1)}(2)) & 1 \\ -\frac{1}{2}(x^{(1)}(2)+x^{(1)}(3)) & 1 \\ \vdots & \vdots \\ -\frac{1}{2}(x^{(1)}(6)+x^{(1)}(7)) & 1 \end{bmatrix},\quad Y=\begin{bmatrix} x^{(0)}(2) \\ x^{(0)}(3) \\ \vdots \\ x^{(0)}(7) \end{bmatrix}.$$

In [6]:

```
#构建矩阵B
BFRAME = np.array([[(-1/2)*(sumData[i]+sumData[i+1]),1] for i in range(len(sumData)-1)])
BFRAME
```

Out[6]:

```
array([[-107.3 ,    1.  ],
       [-179.7 ,    1.  ],
       [-251.95,    1.  ],
       [-323.7 ,    1.  ],
       [-395.4 ,    1.  ],
       [-467.2 ,    1.  ]])
```

In [7]:

```
#构建矩阵Y
YFRAME = data[1:].T
YFRAME
```

Out[7]:

```
array([72.4, 72.4, 72.1, 71.4, 72. , 71.6])
```

公式3

(3) 计算

$$\hat{u}=\begin{bmatrix}\hat{a}\\ \hat{b}\end{bmatrix}=(B^TB)^{-1}B^TY=\begin{bmatrix}0.0023\\ 72.6573\end{bmatrix},$$

In [8]:

```
#按最小二乘法求出系数 a  b
a,b = np.linalg.inv(BFRAME.T.dot(BFRAME)).dot(BFRAME.T).dot(YFRAME)
a,b
```

Out[8]:

```
(0.0023437864785236517, 72.65726960367881)
```

公式

(4) 建立模型

$$\frac{dx^{(1)}}{dt}+\hat{a}x^{(1)}=\hat{b},$$

求解得

$$\hat{x}^{(1)}(k+1)=\left(x^{(0)}(1)-\frac{\hat{b}}{\hat{a}}\right)e^{-\hat{a}k}+\frac{\hat{b}}{\hat{a}}\quad .\qquad (15.7)$$

$$=-30929e^{-0.0023k}+31000$$

In [9]:

```
#求出预测值
def SloveY(k):
    return (data[0] - (b/a))*np.exp(-a*k) + b/a
PredictData = [SloveY(i) for i in range(len(data))]
PredictData
```

Out[9]:

```
[71.09999999999854,
 143.50574144042912,
 215.74197800232287,
 287.8091065037079,
 359.70752283365437,
 431.4376219544247,
 502.999797903678]
```

In [10]:

```
PredictBuild = np.zeros(len(PredictData))
PredictBuild[0] = PredictData[0]

for i in range(1, len(PredictBuild)):
    PredictBuild[i] = PredictData[i] - PredictData[i-1]
PredictBuild
```

Out[10]:

```
array([71.1       , 72.40574144, 72.23623656, 72.0671285 , 71.89841633,
       71.73009912, 71.56217595])
```

残差公式

残差检验

绝对残差: $\varepsilon(k) = x^{(0)}(k) - \hat{x}^{(0)}(k),\ k = 2,3,\cdots,n$

相对残差: $\varepsilon_r(k) = \dfrac{|x^{(0)}(k) - \hat{x}^{(0)}(k)|}{x^{(0)}(k)} \times 100\%,\ k = 2,3,\cdots,n$

平均相对残差: $\overline{\varepsilon}_r = \dfrac{1}{n-1}\sum_{k=2}^{n}|\varepsilon_r(k)|$

如果$\overline{\varepsilon}_r < 20\%$，则认为GM(1,1)对原数据的拟合达到一般要求。

如果$\overline{\varepsilon}_r < 10\%$，则认为GM(1,1)对原数据的拟合效果非常不错。

In [11]:

```
#原数据 data 与 预测数据 PredictBuild 的残差
#将PredictBuild进行一次累减
# CanChaData0 = np.diff(PredictBuild)
# canres = data - CanChaData0
residuals = (data - PredictBuild)
residuals
```

Out[11]:

```
array([ 1.44950718e-12, -5.74144043e-03,  1.63763438e-01,  3.28714986e-02,
       -4.98416330e-01,  2.69900879e-01,  3.78240507e-02])
```

相对误差公式

计算相对误差

$$rel(k)=\frac{e(k)}{x^{(0)}(k)}\times 100\%, k=1,2,\cdots,n$$

In [12]:

```
#相对误差
relativeError = residuals/data
relativeError
```

Out[12]:

```
array([ 2.03868802e-14, -7.93016634e-05,  2.26192594e-03,  4.55915376e-04,
       -6.98062087e-03,  3.74862332e-03,  5.28268865e-04])
```

级比偏差公式

级比偏差检验

首先由$x^{(0)}(k-1)$和$x^{(0)}(k)$计算出原始数据的级比$\sigma(k)$:

$$\sigma(k)=\frac{x^{(0)}(k)}{x^{(0)}(k-1)}(k=2,3,\cdots,n)$$

再根据预测出来的发展系数$(-\hat{a})$计算出相应的级比偏差和平均级比偏差:

$$\eta(k)=\left|1-\frac{1-0.5\hat{a}}{1+0.5\hat{a}}\frac{1}{\sigma(k)}\right|\ (\eta读[eta]),\ \overline{\eta}=\sum_{k=2}^{n}\eta(k)/(n-1)$$

如果$\overline{\eta}<0.2$，则认为GM(1,1)对原数据的拟合达到一般要求。

如果$\overline{\eta}<0.1$，则认为GM(1,1)对原数据的拟合效果非常不错。

In [13]:

```
#级比偏差
rankError = np.array([ 1-((1-0.5*a)/(1+0.5*a))*(1/rankData[i]) for i in range(len(rankData))])
rankError
```

Out[13]:

```
array([-0.01590026,  0.00234104,  0.00647499,  0.01202705, -0.00604265,
        0.00788359])
```

In [14]:

```
#平均级比偏差
rankError.mean() #我们在之前的RankData中已经往前进行移动故不需要再从k=2进行处理（k=2仅数学表达上
```

Out[14]:

```
0.0011306285729361838
```

In [15]:

```
#最终统计表
yearData = [ i for i in range(1986,1993)]
newRankError  = np.insert(rankError, 0, 0)
AllIn = np.array([yearData,data, PredictBuild, residuals, relativeError,newRankError]).T
AllIn
```

Out[15]:

```
array([[ 1.98600000e+03,  7.11000000e+01,  7.11000000e+01,
         1.44950718e-12,  2.03868802e-14,  0.00000000e+00],
       [ 1.98700000e+03,  7.24000000e+01,  7.24057414e+01,
        -5.74144043e-03, -7.93016634e-05, -1.59002600e-02],
       [ 1.98800000e+03,  7.24000000e+01,  7.22362366e+01,
         1.63763438e-01,  2.26192594e-03,  2.34104303e-03],
       [ 1.98900000e+03,  7.21000000e+01,  7.20671285e+01,
         3.28714986e-02,  4.55915376e-04,  6.47498898e-03],
       [ 1.99000000e+03,  7.14000000e+01,  7.18984163e+01,
        -4.98416330e-01, -6.98062087e-03,  1.20270523e-02],
       [ 1.99100000e+03,  7.20000000e+01,  7.17300991e+01,
         2.69900879e-01,  3.74862332e-03, -6.04264569e-03],
       [ 1.99200000e+03,  7.16000000e+01,  7.15621759e+01,
         3.78240507e-02,  5.28268865e-04,  7.88359279e-03]])
```

In [16]:

```
#转换为Pandas
yearData  = []
df = pd.DataFrame(AllIn, columns=["年份","原数据", "预测数据", "残差", "相对误差", "级比偏差"])
df
```

Out[16]:

	年份	原数据	预测数据	残差	相对误差	级比偏差
0	1986.0	71.1	71.100000	1.449507e-12	2.038688e-14	0.000000
1	1987.0	72.4	72.405741	-5.741440e-03	-7.930166e-05	-0.015900
2	1988.0	72.4	72.236237	1.637634e-01	2.261926e-03	0.002341
3	1989.0	72.1	72.067129	3.287150e-02	4.559154e-04	0.006475
4	1990.0	71.4	71.898416	-4.984163e-01	-6.980621e-03	0.012027
5	1991.0	72.0	71.730099	2.699009e-01	3.748623e-03	-0.006043
6	1992.0	71.6	71.562176	3.782405e-02	5.282689e-04	0.007884

In [17]:

```
plt.scatter(data,PredictBuild)  #如果在 x-y = 0  (x=y)直线上，说明预测值与真实值误差很小
#绘制出散点的拟合曲线
plt.plot(data,data,'r')
plt.xlabel('y_test')
plt.ylabel('predictions')
plt.show()
```