大纲

近一段时间来，学习到的数学建模中用到的算法、可视化的内容

1. **规划类**问题
    1. 线性规划（普通线性规划、整数规划、0-1规划）
    2. 非线性规划
2. 求解特定的**数学问题**
    1. 微积分
    2. 线性代数
    3. 画图
3. **层次分析法**
4. **灰度系统**理论
    1. 关联性分析
    2. 灰度预测
5. 插值与拟合
    1. 插值
    2. 拟合
        1. 多项式拟合
        2. 自定义曲线拟合
        3. `cft` 工具箱
6. 可视化
    1. `matplotlib`
    2. `matlab` 自带画图函数

可视化还有一个非常不错的库 `seabon` ，停留在知晓阶段。

规划类问题

线性规划

> 目标函数和约束条件均为决策变量的**线性函数**。

普通线性规划

可以使用 `matlab` 或者 `python` 中的 `scipy` 库或者 `pulp` 库求解线性规划相关的问题。

使用 `scipy` 时，对应的线性规划标准模型如下：

$$
\begin{aligned}
&minc^Tx\\
S.t.&\begin{cases}
Ax\leq b\\
Aeq\times x=beq\\
lb\leq x\leq ub.
\end{cases}
\end{aligned}
$$

求解问题：

$$\max z = 2x_1 + 3x_2 - 5x_3$$
$$\text{s.t.} \begin{cases} x_1 + x_2 + x_3 = 7 \\ 2x_1 - 5x_2 + x_3 \geq 10 \\ x_1 + 3x_2 + x_3 \leq 12 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

由于此线性规划模型不满足`scipy`库规定的线性规划标准模型，所以需要做相应的变换。（等式或者不等式左右两侧乘`-1`）

代码：

```
# 线性规划
from scipy.optimize import linprog
import numpy as np

# 参数信息
# 目标函数
c = np.array([2, 3, -5])

# 不等式约束
A = np.array([[-2, 5, -1], [1, 3, 1]])
b = np.array([-10, 12])

# 等式约束
Aeq = np.array([[1, 1, 1]])
beq = np.array([7])

# 库函数求解 需要了解库函数具体的参数信息 查阅官方文档即可
res = linprog(c, A, b, Aeq, beq, bounds=((0, None), (0, None), (0, None)))

# res的更多信息查阅官方文档
print("目标函数最优", res.fun)
print("最优解", res.x)
```

求解下列问题：

$$\max z = 2x_1 + 3x_2 + x_3$$
$$\text{s.t.} \begin{cases} x_1 + 2x_2 + 4x_3 = 101 \\ x_1 + 4x_2 + 2x_3 \geq 8 \\ 3x_1 + 2x_2 \geq 6 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

```
# pulp求解线性规划
import pulp
import numpy as np

# 目标函数
z = np.array([2, 3, 1])

# 不等式约束
A = np.array([[1, 4, 2], [1, 2, 0]])
b = np.array([8, 6])

# 等式约束
Aeq = [[1, 2, 4]]
beq = [101]
# 确定问题 如果求的是最小值可以更改sense参数或者修改目标函数
```

```
m = pulp.LpProblem(sense=pulp.LpMaximize)

# 定义三个变量放到列表中
x = [pulp.LpVariable(f'x{i}', lowBound=0) for i in [1, 2, 3]]  # 列表推导结果[x1, x2, x3]

# 定义目标函数
m += pulp.lpDot(x, z)  # 按位相乘再相加 ⟶ 目标函数

# 不等于约束
for i in range(len(A)):
    m += (pulp.lpDot(A[i], x) >= b[i])

# 等式约束
for i in range(len(Aeq)):
    m += (pulp.lpDot(Aeq[i], x) == beq[i])

# 求解
m.solve()
# 输出结果
# 优化结果
print(pulp.value(m.objective))
# 决策变量
print([pulp.value(var) for var in x])
```

对比这两个库提供的`api`，可以发现`pulp`库更加符合人的直觉。

整数规划

> 相对于普通线性规划，限制了决策变量的解只能是整数。

1. 整数规划对应的松弛模型无解，整数规划无解
2. 松弛模型有解，整数规划不一定有解

可以使用分支定界、割平面求解整数规划。不过`python`有**成熟的库**可以直接调用

求解如下的问题

$$\begin{aligned}&\min z=40x_1+90x_2,\\&\text{s.t.}\begin{cases}9x_1+7x_2\leq 56,\\7x_1+20x_2\geq 70,\\x_1,x_2\geq 0\text{ 为整数.}\end{cases}\end{aligned}$$

可以使用`cvxpy`

```python
import cvxpy as cp
from numpy import array

c = array([40, 90])  # 定义目标向量
a = array([[9, 7], [-7, -20]])  # 定义约束矩阵
b = array([56, -70])  # 定义约束条件的右边向量
x = cp.Variable(2, integer=True)  # 定义两个整数决策变量
obj = cp.Minimize(c * x)  # 构造目标函数
cons = [a * x <= b, x >= 0]  # 构造约束条件
prob = cp.Problem(obj, cons)  # 构建问题模型
prob.solve(solver='GLPK_MI', verbose=True)  # 求解问题
print("最优值为:", prob.value)
print("最优解为: \n", x.value)
```

0-1 规划

> 限制了决策变量只能取0或者1

可以使用匈牙利算法实现，但是 python 丰富的生态提供了开箱即用的库。可以使用 scipy

这种问题最经典的就是**指派问题**。

参考博客

模型：

$$\min z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$s.t = \begin{cases} \sum_{i=1}^{n} x_{ij} = 1, & i = 1,2,\cdots,n \\ \sum_{j=1}^{n} x_{ij} = 1, & j = 1,2,\cdots,n \\ x_{ij} = 0 \text{ 或 } 1, & i,j = 1,2,\cdots,n \end{cases}$$

求解下列指派问题

课程	线代	高数	大物	工图
A	18	5	7	6
B	10	16	6	5
C	11	6	4	7
D	13	12	9	11

安排老师讲课，使得讲课效果最好。

```python
import numpy as np
from scipy.optimize import linear_sum_assignment


# 教师与课程一样多
# 各个教师对各个课的擅长程度矩阵
goodAt = np.array([[18, 5, 7, 16], [10, 16, 6, 5],
                   [11, 6, 4, 7], [13, 12, 9, 11]])
# 转化问题
weakAt = 20-goodAt
row_ind, col_ind = linear_sum_assignment(weakAt)
print(row_ind)
for i in range(len(row_ind)):
```

```
    print("第" + str(row_ind[i]) + "老师" + "教" + "第" + str(col_ind[i]) + "门课")
```

非线性规划

计算 $1/x + x$ 的最小值

```
# 非线性规划
from scipy.optimize import minimize
import numpy as np

def fun(args):
    a = args
    v = lambda x:a/x[0] + x[0]
    return v

args = 1

x0 = np.asarray(2)
res = minimize(fun(args), x0, method='SLSQP')

print(res)
```

求解如下问题：

$$\min f(x) = x_1^2 + x_2^2 + x_3^2 + 8$$
$$\text{s.t.} \begin{cases} x_1^2 - x_2 + x_3^2 \geq 0 \\ x_1 + x_2^2 + x_3^2 \leq 20 \\ -x_1 - x_2^2 + 2 = 0 \\ x_2 + 2x_3^2 = 3 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

```
from scipy.optimize import minimize
import numpy as np


def func(x):
    x1, x2, x3 = x
    return x1 ** 2 + x2 ** 2 + x3**2 + 8


cons = ({'type': 'ineq', 'fun': lambda x: x[0] ** - x[1] + x[2]**2},
        {'type': 'ineq', 'fun': lambda x: -x[0] - x[1] ** 2 - x[2] ** 2 + 20},
        {'type': 'eq', 'fun': lambda x: -x[0] - x[1] ** 2 + 2},
        {'type': 'eq', 'fun': lambda x: x[0] + x[2] ** 2})

res = minimize(func, np.ones(3), constraints=cons,
               bounds=((0, None), (0, None), (0, None)))

print(res.fun)
print(res.x)
```

这里更推荐使用 `matlab` 提供的 `api` 求解非线性规划相关的问题：

同样，为了使用此 `api`，必须把线性规划模型转化成如下的形式：

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \leq 0 \\ ceq(x) = 0 \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub, \end{cases}$$

使用 `fmincon` 求解上述使用 `python` 求解的非线性规划

文档

```
function f = fun1(x)
f = x(1).^2 + x(2).^2 + x(3).^2 + 8;
end


function [g,h] = fun2(x)
g(1) = - x(1).^2 + x(2) - x(3).^2;
g(2) = x(1) + x(2).^2 + x(3).^3 - 20;
% g代表不等式约束，Matlab中默认g≤0，所以这里取相反数
h(1) = - x(1).^2 - x(2).^2 + 2;
h(2) = x(2) + 2 * x(3).^2 - 3;
% h代表等式约束
end


[x, y] = fmincon('fun1', rand(3, 1), [], [], [], [], zeros(3, 1), [], 'fun2');
% 'fun1'代表目标函数，rand(3, 1)随机给了x初值，zeros(3, 1)代表下限为0，即x1, x2, x3≥0,
'fun2'即刚才写的约束条件
```

对于其他的模型， `matlab` 也提供了对应的函数。如下：

```
普通线性规划 linprog
整数规划 intlinprog
```

`api` 的使用不需要死记硬背，用到时查阅文档即可

求解特定的数学问题

`python` 强大的社区生态提供了丰富的库来实现数学问题的求解。

简单介绍下求解的方法，特定的求解问题时可以翻阅文档解决。

使用 `sympy` ，一个基于符号的库。非常符合人的直觉。

参考博客1

参考博客2

文档

更多用法可以参考上述资料。

微积分

> 极限 微分方程 级数等都可以使用 `python` 求解。

微分方程

$$\frac{d^2x(t)}{dt^2} + 2\gamma\omega_0\frac{dx(t)}{dt} + \omega_0^2 x(t) = 0$$

求解上述问题的代码如下：

```
from sympy import *

### symbols函数用于创建符号
t, gamma, omega0 = symbols('x gamma omega0')
x = Function('x')

# 构造微分方程
eq = x(t).diff(t, 2) + 2 * gamma * omega0 * x(t).diff(t, 1) + omega0 ** 2  * x(t)

ans = dsolve(eq, x(t))
pprint(ans)
```

极限

$$\lim_{x\to 0}\frac{\sin x}{x} = 1, \lim_{x\to+\infty}\left(1+\frac{1}{x}\right)^x = e$$

求解如下：

```
from sympy import *

x = symbols('x')

print(limit(sin(x) / x, x, 0))

# oo表示无穷大
print(limit((1 + 1 / x) ** x, x, oo))
```

级数求和

$$\sum_{k=1}^{\infty}\frac{1}{k^2} = \frac{\pi^2}{6}$$

```
from sympy import *
k = symbols('k')

print(summation(1 / (k ** 2), (k, 1, oo)))
```

线性代数

> 求行列式 矩阵的秩 转置 特征向量 特征值等。

这些问题在 `sympy` 中都提供了函数解决，用到时可以查阅文档。

$$\begin{cases} 2x_1 + x_2 - 5x_3 + x_4 = 8 \\ x_1 - 3x_2 - 6x_4 = 6 \\ 2x_2 - x_3 + 2x_4 = -2 \\ x_1 + 4x_2 - 7x_3 + 6x_4 = 2 \end{cases}$$

```
import sympy as sp
A=sp.Matrix([[2, 1, -5, 1],[1, -3, 0, -6],[0, 2, -1, 2],[1, 4, -7, 6]])
b=sp.Matrix([8, 6, -2, 2]); b.transpose()
print("系数矩阵A的秩为:",A.rank()) # r(A) == 4
# A ⟶ 对矩阵求逆矩阵
# x = A^-1 *  b
print("线性方程组的唯一解为:",A.inv()*b)
```

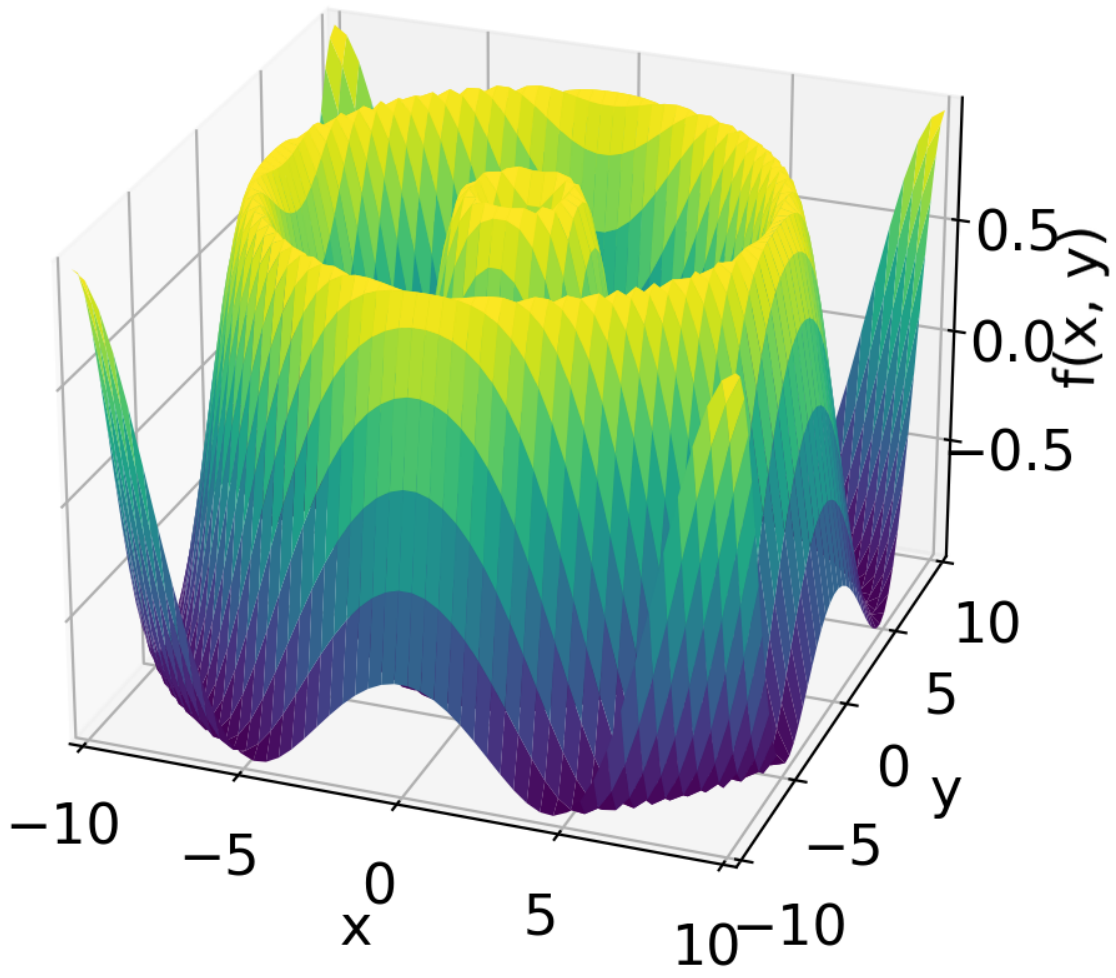
还有很多办法求解数学问题，推荐：

wolframalpha 大部分问题都能求解，还能给出求解的过程。

画图

> 应该是 sympy 库最实用的功能。

```
from pylab import rc  #pylab为matplotlib的接口
from sympy.plotting import plot3d
from sympy.abc import x,y   #引进符号变量x,y
from sympy.functions import sin,sqrt
rc('font',size=16); rc('text')
plot3d(sin(sqrt(x**2+y**2)),(x,-10,10),(y,-10,10),xlabel='x',ylabel='y')
```



层次分析法

> 层次分析法可以用于最佳方案的选取 评价类问题等等。

步骤

1. 建立层次结构模型
2. 构造判断矩阵
3. 层次单排序 一致性检验
4. 层次总排序 一致性检验

给出求解函数

```
%解决层次化分析
function [CR, w, m] = ahp(A)
    RI = [0 0 0.58 0.90 1.12 1.24 1.32 1.41 1.45 1.49 1.51];

    A = [1 1/2 4 3 3; 2 1 7 5 5; 1/4 1/7 1 1/2 1/3; 1/3 1/5 2 1 1; 1/3 1/5 3 1 1];
    [n, n] = size(A)
    [V, D] = eig(A);

    % 寻找最大特征值
    pos = 1;
    max = D(1, 1);
```

```
    for i = 1:n
        if (D(i, i) > max)
            max = D(i, i);
            pos = i
        end
    end

    %最大特征值对应的特征向量
    w = V(:, i);
    %归一化求权向量
    w = w / sum(w)

    %一致性检验
    CI = (max - n) / (n - 1);

    CR = CI / RI(n); # 需要对CR做一致性检验，必须满足要求
    m = max; # 最大特征值
end
```

灰度系统理论

> 灰度系统理论可以用于预测 关联性分析。

通过一个简单的例子来看。

以下为某地区总收入、养猪、养兔收入的变化数据。

表 1　收入数据

	1977	1978	1979	1980	1981	1982	1983
总收入	18	20	22	40	44	48	60
养　猪	10	15	16	24	38	40	50
养　兔	3	2	12	10	22	18	20

关联性分析

量化养猪收入、养兔收入和总收入的关联

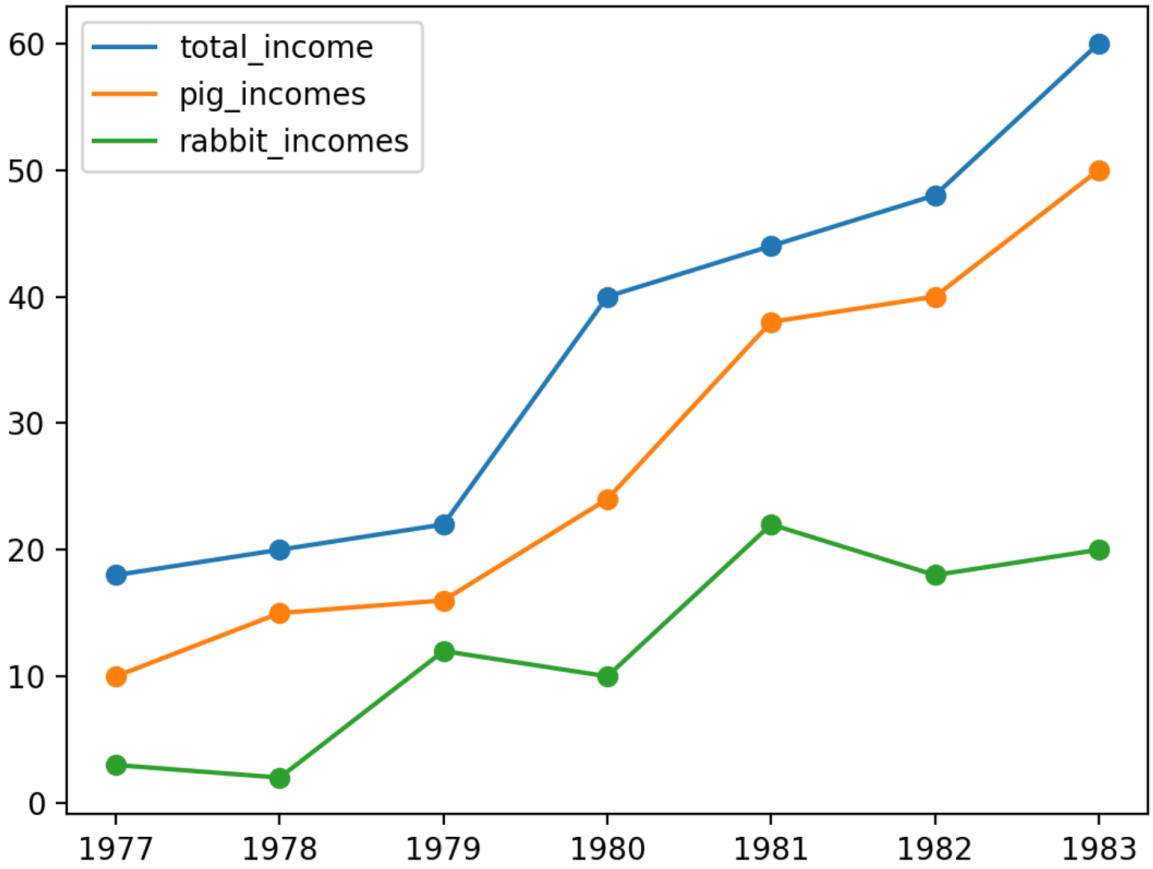
先做出**折线图**

```
import numpy as np
import matplotlib.pyplot as plt
years = np.linspace(1977, 1983, 7, dtype=np.int16)

total_incomes = np.array([18, 20, 22, 40, 44, 48, 60])
pig_incomes = np.array([10, 15, 16, 24, 38, 40, 50])
rabbit_incomes = np.array([3, 2, 12, 10, 22, 18, 20])


plt.figure()
plt.scatter(years, total_incomes)
plt.scatter(years, pig_incomes)
plt.scatter(years, rabbit_incomes)
plt.plot(years, total_incomes, label="total_income")
plt.plot(years, pig_incomes, label="pig_incomes")
plt.plot(years, rabbit_incomes, label="rabbit_incomes")
```

```
plt.legend()
plt.show()
```



通过折线图可以发现，养猪的变化和总收入变化类似。可以直观得感受到，养猪的收入和总收入之间的关联性更大。

也可以采用灰色关联分析:

```
# 关联性分析
import numpy as np
# 参考数列
X0 = np.array([18, 20, 22, 40, 44, 48, 60])
# 比较数列
X = np.array([
    [10, 15, 16, 24, 38, 40, 50],
    [3, 2, 12, 10, 22, 18, 20]
])

minus = []
for i in range(len(X)):
  minus.append(np.abs(X[i] - X0))

minus = np.array(minus)

row_len = minus.shape[0]
col_len = minus.shape[1]
# 最大差 最小差
min_tmp = []
max_tmp = []
for row_idx in range(row_len):
  min_tmp.append(np.min(minus[row_idx]))
  max_tmp.append(np.max(minus[row_idx]))
```

```
min = np.min(np.array(min_tmp))
max = np.max(np.array(max_tmp))

# 分辨系数
rho = 0.5

tmp = []

for i in range(row_len):
  t = []
  for k in range(col_len):
    t.append((min + rho * max) / (abs(X0[k] - X[i][k]) + rho * max))
  tmp.append(t)


ans = []

for i in range(row_len):
  ans.append(sum(tmp[i]))

print(ans)
```

根据得出来的结果，也验证了我们的直觉。

上面的代码就是对模型的一个简单**模拟**。

预测

使用灰度系统对总收入进行预测

一定要进行级比检验

下面的代码也是对模型的简单**模拟**

```
# GM(1, 1)
import numpy as np
import math
import matplotlib.pyplot as plt


X0 = np.array([18, 20, 22, 40, 44, 48, 60])
years = np.linspace(1977, 1983, 7, dtype=np.int16)
# 最开始应该检验数列的级比，这里忽略
length = X0.shape[0]

X1 = []

for i in range(length):
    if(i == 0):
        X1.append(X0[i])
    else:
        X1.append(X0[i - 1] + X0[i])

Z = []
rho = 0.5
for i in range(length):
    if(i == 0):
        Z.append(0)
```

```
    else:
        Z.append(rho * X1[i] + (1 - rho) * X1[i - 1])

Y = np.array(X1[1:])
Y = Y.T

Z = np.array(Z)
B = []

for i in range(Z.shape[0]):
    if(i != 0):
        B.append([-Z[i], 1])


B = np.array(B)

# a b参数
t1 = np.matmul(B.T, B)
# 矩阵求逆矩阵
t1 = np.linalg.inv(t1)

t1 = np.matmul(t1, B.T)
ans = np.matmul(t1, Y)


a = ans[0]
b = ans[1]

# 预测
Xhat1 = []

years1 = np.linspace(1977, 1986, 10)
for i in range(10):
    Xhat1.append((X0[1] - b / a) * pow(math.e, -a * (i - 1)) + b / a)


Xhat0 = []
Xhat0.append(0)

# 需要对预测值Xhat0检验 ⟶ 残差 ...
for i in range(10):
    if(i != 0):
        Xhat0.append(Xhat1[i] - Xhat1[i - 1])
```

也可以对结果进行**可视化分析**。

灰度预测出来的结果感觉不太准确?

插值与拟合

> 用于进行数值分析。

插值

> 插值就是：已知函数在某区间内的若干值，求函数在此区间内的其他值。

插值的方法有：

1. 拉格朗日插值
    1. 高次会出现龙格现象
    2. 把插值区间分成若干小区间，在小区间内进行低次插值避免龙格现象
2. 牛顿插值

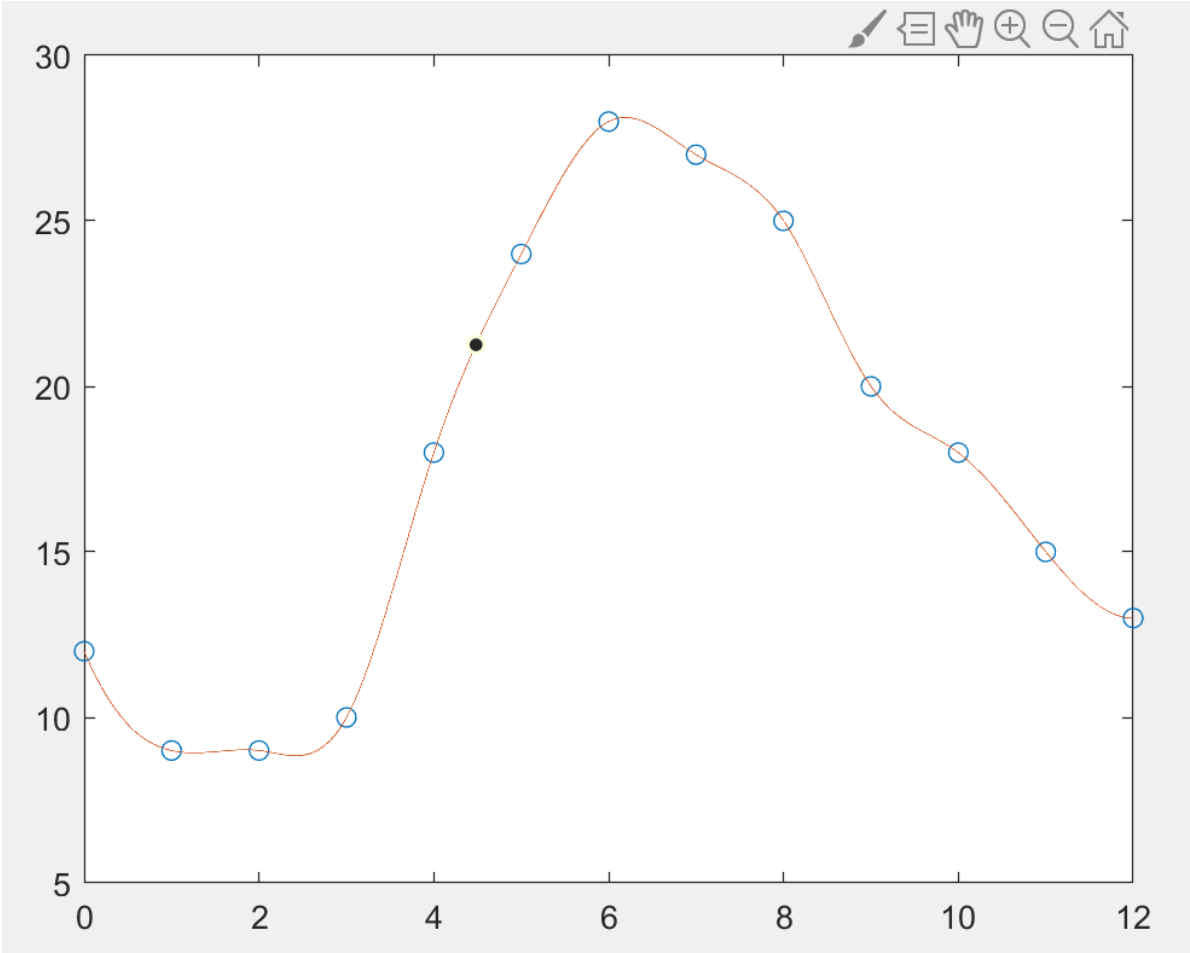
一维插值

在`matlab`中，可以使用`interp1`进行一维插值。

`yi = interp1(x, y, xi, 'method')`

- x y 为插值点
- xi yi为被插值点和插值结果
- `method`表示插值方法 默认为线性插值
    - `nearest` 最邻近插值
    - `linear` 线性插值
    - `spline` 三次样条插值
    - `cubic` 立方插值

```
x = 0:1:12;
y = [12 9 9 10 18 24 28 27 25 20 18 15 13];
xi = 0:1/3600:12;
yi = interp1(x, y, xi, 'spline');
plot(x, y, 'o'); hold on;
plot(xi, yi);
```



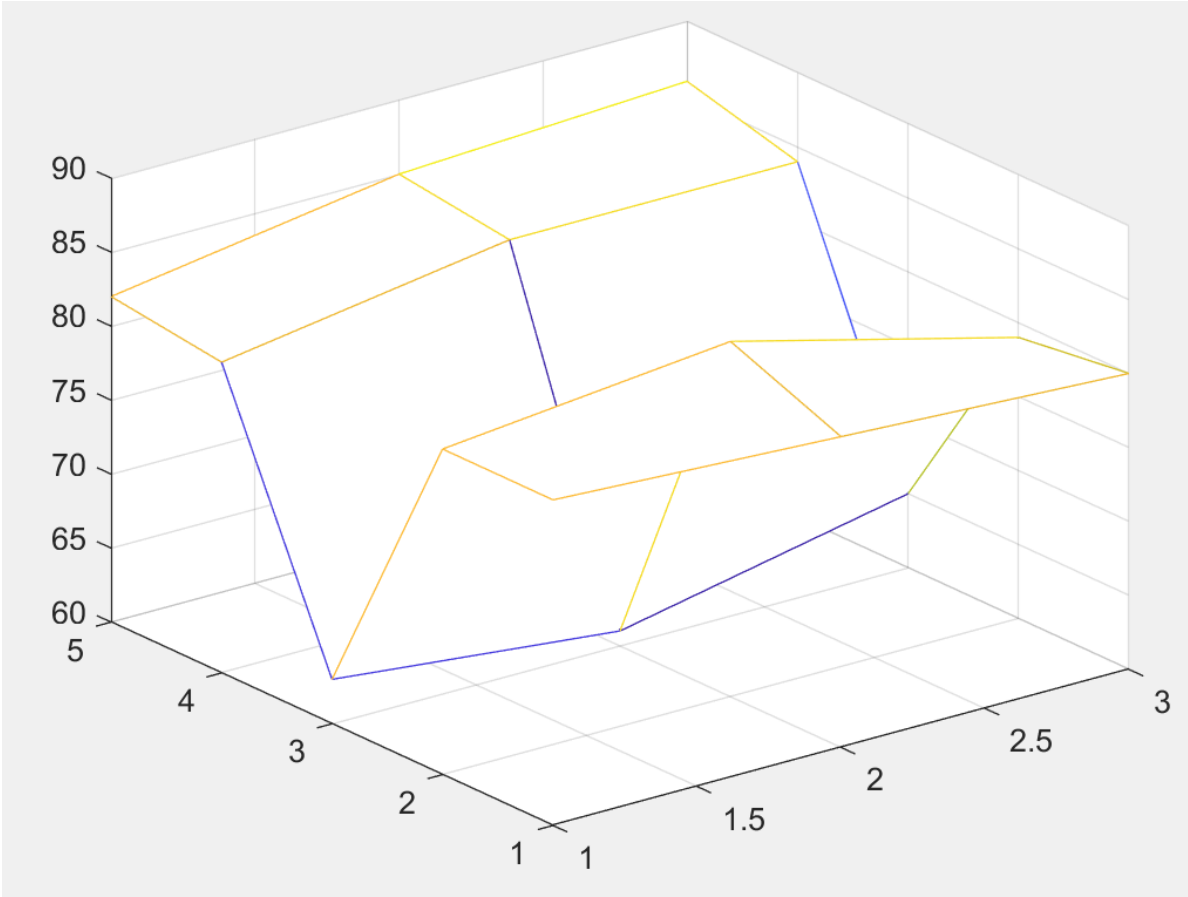
二维插值

二维插值可以使用 `matlab` 中的函数

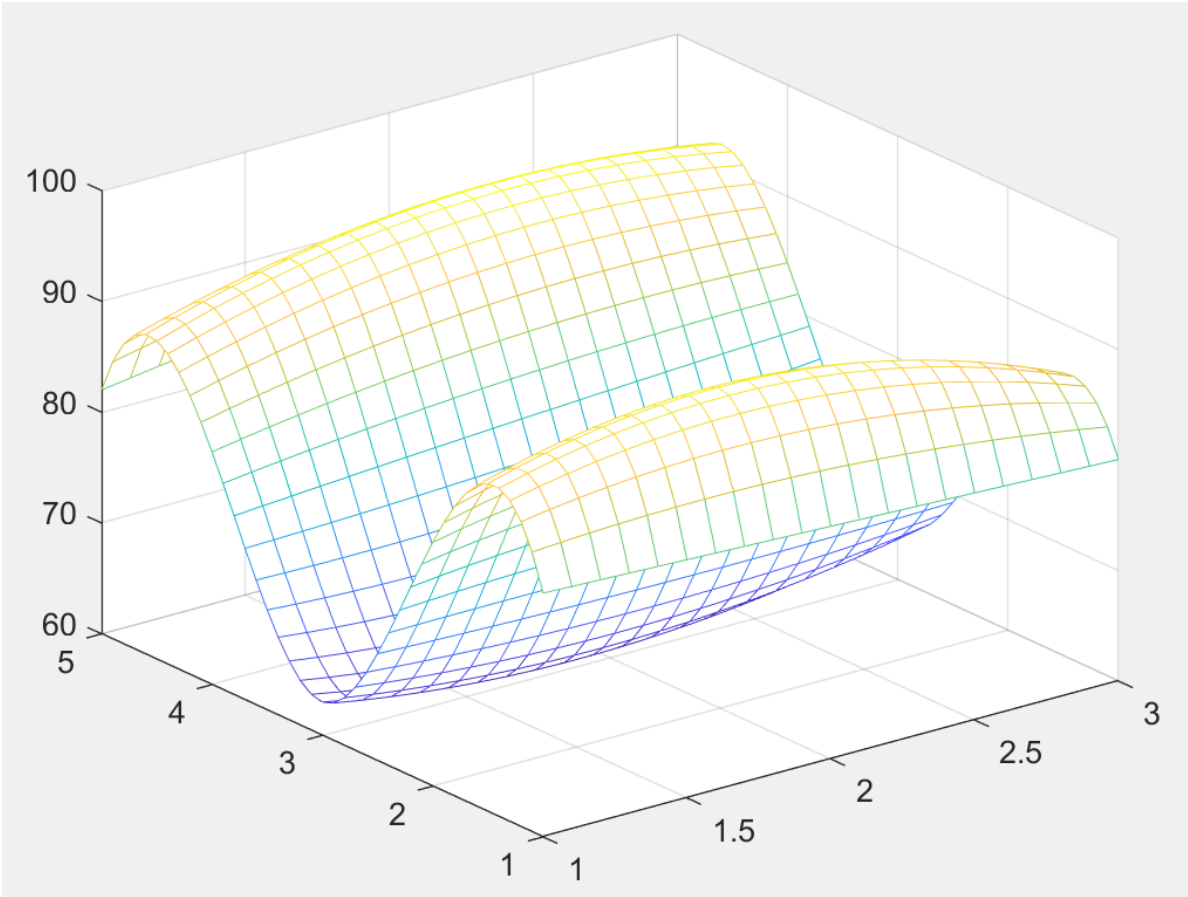
```
zi = interp2(x,y,xi,yi,'method')
```

参数类似一维插值的情况。

```
x = 1:3;
y = 1:5;
z = [82 81 80; 82 84 79; 63 61 65; 81 84 84;82 85 86];
figure(1);
mesh(x, y, z);
```




```
xi = 1:0.1:3
yi =  1:0.1:5
% 需要保持xi yi行列不同
zi = interp2(x, y, z , xi,yi', 'spline');
figure(2)
mesh(xi, yi, zi)
```



有时需要应对散乱点插值，可以使用`griddata`

拟合

拟合的两个步骤

1. 曲线的选择
2. 参数的求解

方法

1. 线性拟合 --> 最小二乘法
2. 非线性拟合 --> 牛顿法

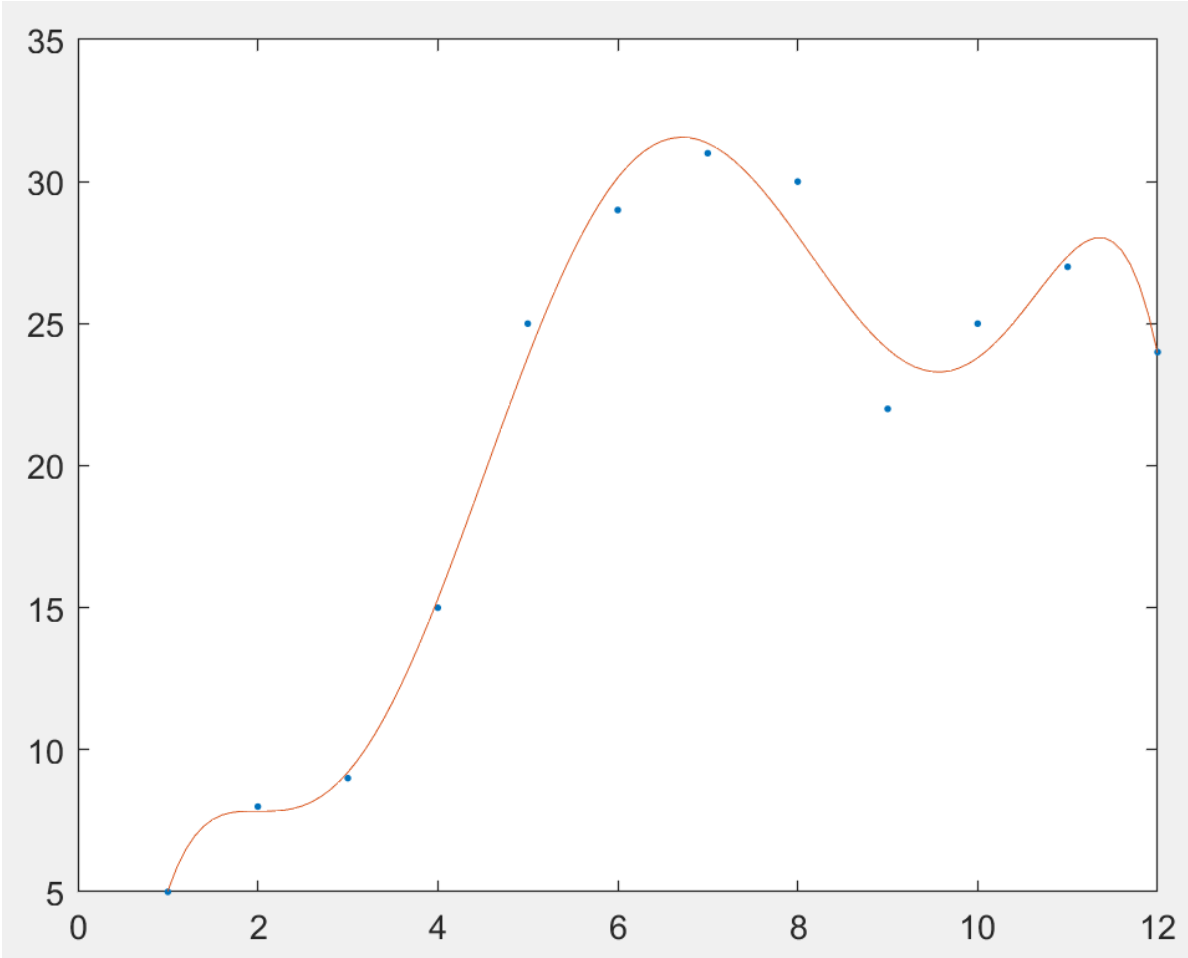
多项式拟合

可以使用`matlab`的`a = polyfit(x, y, n)`进行拟合

1. x y是待拟合的数据
2. n是多项式的次数
3. a是求解得到的多项式系数信息

```
x = 1:12;
y = [5 8 9 15 25 29 31 30 22 25 27 24];
a = polyfit(x, y, 6);

xi = 1:0.1:12;
yi = polyval(a, xi);
plot(x, y, '.', xi, yi);
```



自定义函数拟合

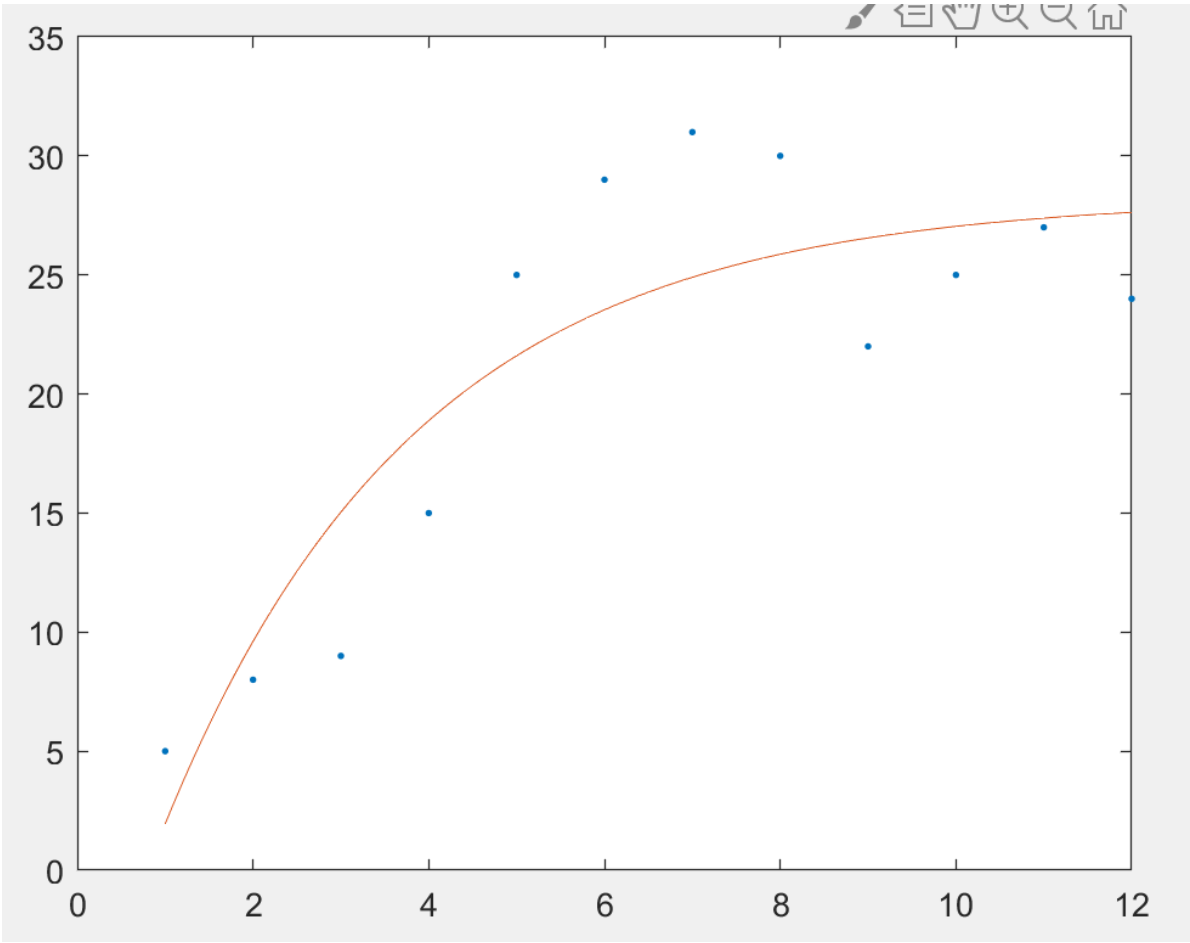
也可以自己设置拟合的曲线。

```
a = nlinfit(x, y, fun, a0);
```

```
x = 1:12;
y = [5 8 9 15 25 29 31 30 22 25 27 24];

a0 = [1 1 1 1 1];
fun = @(a, x) a(1) * exp(-x/a(2)) + a(3) * exp(-x/a(4)) + a(5);
a = nlinfit(x, y, fun, a0);

xi = 1:0.1:12;
yi = fun(a, xi);
plot(x, y, '.', xi, yi);
```



拟合效果不是很好，可以修改拟合函数让拟合做得更好。

`cft` 工具箱

如果不想写代码，可以试试 `cft` 工具箱，只需要点点点，就能得到想要得结果，非常**方便快捷**。