# 第一次作业

**学号**：12115990136  
**姓名**：刘抗非  
**课程**：化工过程模拟及软件应用

## T1.jl

### 问题描述

某公司销售的设备价格方案具有数量折扣特性。具体价格策略如下：购买一台设备的价格为$150；若购买两台设备，第二台的价格为$120；从第三台开始，每台设备的价格均为$110。需要计算购买1至10台设备的总费用，并通过图形展示不同数量购买时的价格变化。

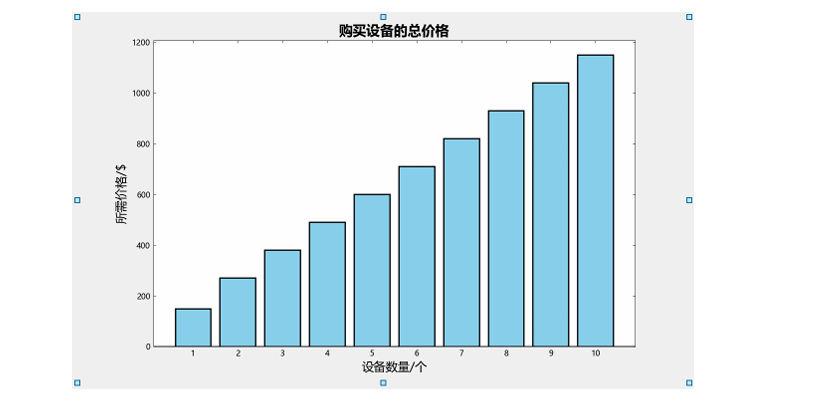
### 求解思路

1. **函数设计**：设计一个函数calculate\_prices(n)，输入参数n代表购买的设备数量，函数返回该数量设备的总价格。
2. **条件判断**：
   * 当n为1时，总价格为$150。
   * 当n为2时，总价格为120。
   * 当n大于等于3时，总价格为120 + ($110 \* (n-2))。
3. **循环计算**：使用循环结构遍历1到10的设备数量，调用calculate\_prices函数计算每个数量的总价格，并存储结果。
4. **绘图展示**：使用条形图展示不同数量设备的购买总价格，以直观反映价格与购买数量的关系。

### MWORKS 程序

using TyBase  
using TyMath  
using TyPlot  
  
# 定义计算费用的函数  
function calculate\_prices(n)  
 if n == 1  
 return 150.0  
 elseif n == 2  
 return 150.0 + 120.0  
 elseif n >= 3  
 return 150.0 + 120.0 + (n-2) \* 110.0  
 end  
end  
  
# 创建数量和对应价格的数组  
n = [1:1:10;];  
prices = Float64[];  
  
# 循环计算购买1到10台的价格并将结果放到prices中  
for i in n  
 push!(prices, calculate\_prices(i))  
end  
  
# 输出结果  
prices\_10 = prices[10];  
print("购买 10 台设备的总价格为: ", prices\_10, "\$")  
  
# 绘制条形统计图  
figure(figsize=(10, 6)) # 设置图形大小  
bar(n, prices, color=:skyblue, edgecolor=:black, linewidth=1.5)  
  
# 添加标签和标题  
xlabel("设备数量/个", fontsize=14)  
ylabel("所需价格/\$", fontsize=14)  
title("购买设备的总价格", fontsize=16, fontweight="bold")  
grid(true) # 添加网格  
xticks(n); # 设置 x 轴刻度

运行后结果如下:

julia> 正在运行 T1.jl  
购买 10 台设备的总价格为: 1150. 0$

### 结果分析

根据程序输出，可以看到随着购买数量的增加，设备的单价逐渐降低，从而使得总价格的增速较为平缓。这种价格策略可能会激励顾客购买更多的设备。条形图清晰地展示了这一价格变动趋势，用户可以直观地感受到不同购买数量对总价格的影响。此外，从程序设计角度来看，通过循环和条件判断的有效结合，实现了对复杂价格策略的精确计算。

## T2.jl

### 问题描述

在化工反应工程的研究中，记录了一种反应物A在体积为4升的容器中进行水解反应的数据。数据包括反应时间（小时）和对应的浓度（mol/L）。需要使用插值方法来估计特定时间点的浓度，并通过曲线拟合方法确定反应的动力学方程。

### 求解思路

#### 插值计算

1. **数据准备**：给定时间点t和浓度cA的原始数据。
2. **目标时间点**：定义需要插值的时间点tq。
3. **插值方法**：
   * 使用线性插值(linear)和样条插值(spline)方法。
   * 对每个目标时间点应用这两种插值方法，计算预测的浓度值。
4. **结果展示**：绘制原始数据和插值结果的散点图，以比较两种插值方法的效果。

#### 曲线拟合

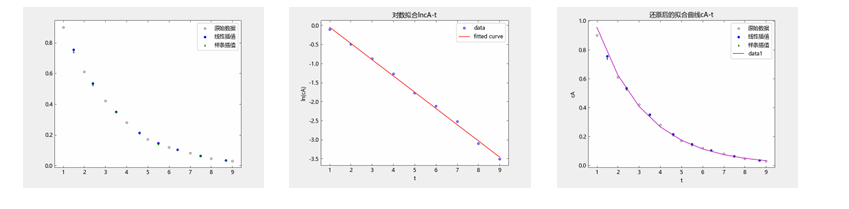
1. **模型假设**：假设反应遵循一阶动力学方程，即-dCA/dt = k\*CA。
2. **对数线性化**：对浓度取自然对数，将方程转化为线性形式。
3. **线性拟合**：使用最小二乘法拟合得到的对数浓度与时间的关系，得到速率常数k和初始浓度的对数C。
4. **动力学方程重建**：根据拟合参数重建动力学方程。

### MWORKS 程序

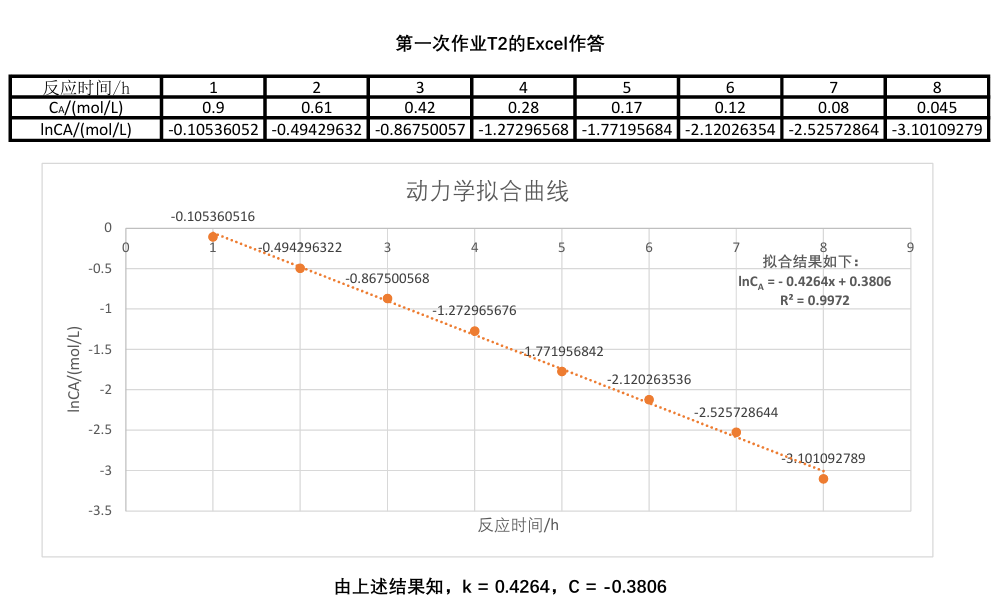
using TyBase  
using TyMath  
using TyPlot  
using TyCurveFitting  
using Printf  
  
t = [1:1:9;]  
cA = vec([0.9 0.61 0.42 0.28 0.17 0.12 0.08 0.045 0.03]);  
  
# 待插值查询的点  
tq = vec([1.5, 2.4, 3.5, 4.6, 5.5, 6.4, 7.5, 8.7]);  
  
# 两种不同的插值方式  
cAq\_linear = interp1(t, cA, tq, "linear");  
cAq\_spline = interp1(t, cA, tq, "spline");  
  
# 输出结果，保留 6 位小数  
println("线性插值所得结果:")  
for value in cAq\_linear  
 @printf("%.6f\n", value)  
end  
  
println("样条插值所得结果:")  
for value in cAq\_spline  
 @printf("%.6f\n", value)  
end  
  
# 另一种格式化输出方式  
# println("线性插值所得结果:")  
# println(join(map(x -> @sprintf("%.6f", x), caq\_linear), "\n"))  
# println("样条插值所得结果:")  
# println(join(map(x -> @sprintf("%.6f", x), caq\_spline), "\n"))  
  
# 第一幅图————纯享数据散点图  
figure()  
hold(true)  
scatter(t, cA, 10, "k", label="原始数据")  
scatter(tq, cAq\_linear, 20, "b", marker="h", filled=true, label="线性插值")  
scatter(tq, cAq\_spline, 10, "g", marker="d", filled=true, label="样条插值")  
legend()  
  
# 准备拟合数据  
ln\_cA = log.(cA)  
  
# 拟合一阶动力学方程，这里用线性回归 ln(cA) = -kt - C  
lin\_fit = fit("poly1", t, ln\_cA)  
params = lin\_fit.params  
  
# 因 ln(cA) = -kt - C，则提取参数并取反得到k和C  
k, C = -params[1], -params[2] # 取反得k, C值  
println("拟合参数:")  
println("k: ", k)  
println("C: ", C)  
  
# 第二幅图————绘制对数拟合曲线  
figure()  
plotfit(lin\_fit, t, ln\_cA)  
title("对数拟合lncA-t")  
xlabel("t")  
ylabel("ln(cA)")  
legend()  
  
# 第三幅图————绘制对数拟合还原后的曲线  
figure()  
hold(true)  
scatter(t, cA, 10, "k", label="原始数据")  
scatter(tq, cAq\_linear, 20, "b", marker="h", filled=true, label="线性插值")  
scatter(tq, cAq\_spline, 10, "g", marker="d", filled=true, label="样条插值")  
fit\_curve = exp.(-k .\* t .- C) # 指数形式 cA = e^(-kt - C)  
plot(t, fit\_curve, color="m", label="拟合曲线")  
title("还原后的拟合曲线cA-t")  
xlabel("t")  
ylabel("cA")  
legend();

运行后结果如下:

julia> 正在运行 T2.jl  
线性插值所得结果:  
0.755000  
0.534000  
0.350000  
0.214000  
0.145000  
0.104000  
0.062500  
0.034500  
样条插值所得结果:  
0.739206  
0.524951  
0.346368  
0.207671  
0.139943  
0.104286  
0.060961  
0.031485  
拟合参数:  
k: 0.4264822157072334  
C: -0.3806641018736966



Excel中拟合结果如下：



### 结果分析

通过插值和曲线拟合，我们能够在未直接观测到的时间点估计浓度，并且确定了反应的动力学模型。线性插值和样条插值的结果差异展示了不同数学方法在处理实际化工数据时的适用性。动力学方程的参数提供了对反应速率的量化描述，有助于进一步的反应设计和优化。

## T3.jl

### 问题描述

给定条件下，将2.03MPa、477K和2.83m³的NH3气体压缩到0.142m³，压缩后温度为448.6K。需要用普遍化关系式计算其最终压力。

### 求解思路

1. **定义常量**：氨气（NH3）的临界压力Pc、临界温度Tc和偏心因子w。
2. **计算初始状态（状态1）**：
   * 计算对比压力Pr1和对比温度Tr1。
   * 使用二维插值得到Z0和Z1，进而计算压缩因子Z。
   * 根据理想气体状态方程推导出物质的量n。
3. **设置终态参数并进行迭代计算（状态2）**：
   * 输入终态的体积V2和温度T2。
   * 计算单位摩尔体积v2。
   * 初始估计P2，使用试差法迭代计算直至满足精度要求。
   * 在每次迭代中，根据当前估计的P2计算Pr2和Tr2，再次使用二维插值得到Z0和Z1，计算新的Z值和新的P2。

### MWORKS 程序

using TyBase  
using TyMath  
using TyPlot  
using Printf  
  
# 常量定义  
const Pc = 11.28e6 # Pa  
const Tc = 405.65 # K  
const w = 0.2526 # 无量纲  
const R = 8.314 # J/(mol·K)  
  
# 定义 Pr 和 Tr 的网格数据（示例数据，请根据实际表格数据替换）  
Pr\_values = [0.2, 0.4, 0.6, 0.8, 1.0] # 示例 Pr 网格  
Tr\_values = [1.0, 1.2, 1.4, 1.6, 1.8] # 示例 Tr 网格  
  
# 示例 Z0 和 Z1 表格数据 (Pr × Tr)，实际应用中，这些数据应来自于 NH₃ 的压缩因子表格  
Z0\_table = [  
 1.1 1.09 1.07 1.05 1.03;  
 1.0 0.99 0.97 0.95 0.93;  
 0.9 0.89 0.87 0.85 0.83;  
 0.8 0.79 0.77 0.75 0.73;  
 0.7 0.69 0.67 0.65 0.63  
]  
  
Z1\_table = [  
 0.05 0.048 0.046 0.044 0.042;  
 0.04 0.038 0.036 0.034 0.032;  
 0.03 0.028 0.026 0.024 0.022;  
 0.02 0.018 0.016 0.014 0.012;  
 0.01 0.008 0.006 0.004 0.002  
]  
  
# 使用 meshgrid2 创建二维网格数据  
Pr\_grid, Tr\_grid = meshgrid2(Pr\_values, Tr\_values)  
  
# 定义二维插值函数  
function get\_Z(Pr, Tr, Pr\_grid, Tr\_grid, Z0\_table, Z1\_table, w)  
 # 确保 Pr 和 Tr 是标量，因为上面运算出来的Pr，Tr是矩阵  
 if !isscalar(Pr) || !isscalar(Tr)  
 error("Pr 和 Tr 必须是标量")  
 end  
   
 # 使用 TyMath.interp2 进行二维插值  
 Z0 = TyMath.interp2(Pr\_grid, Tr\_grid, Z0\_table, Pr, Tr)  
 Z1 = TyMath.interp2(Pr\_grid, Tr\_grid, Z1\_table, Pr, Tr)  
   
 # 确保 Z0 和 Z1 是标量  
 Z0 = isscalar(Z0) ? Z0 : Z0[1]  
 Z1 = isscalar(Z1) ? Z1 : Z1[1] # ? : 是三元运算符，? 表示判断，为真时执行 : 前面的，为假时执行 : 后面的  
   
 Z = Z0 + w \* Z1  
 return Z  
end  
  
# 状态1已知  
P1 = 2.03e6 # Pa  
V1 = 2.83 # m³  
T1 = 477.0 # K  
  
# 对比压力和对比温度  
Pr1 = P1 / Pc  
Tr1 = T1 / Tc  
  
# 获取状态1的 Z  
Z\_1th\_state = get\_Z(Pr1, Tr1, Pr\_grid, Tr\_grid, Z0\_table, Z1\_table, w)  
  
# 计算物质的量 n  
n = (P1 \* V1) / (Z\_1th\_state \* R \* T1)  
  
@printf("使用普维法计算得到的物质的量 n = %.6f mol\n", n)  
  
# 终止条件  
T2 = 448.6 # K  
V2 = 0.142 # m³  
  
# 计算状态2的摩尔体积  
v2 = V2 / n  
  
# 迭代求解状态2的 P2  
# 初始化 P2 的猜测值，假设初始为理想气体情况  
P2 = (R \* T2) / v2  
  
# 设置迭代参数  
tolerance = 1e-6  
max\_iterations = 100  
global iteration = 0  
global abs\_error = 1.0  
  
while abs\_error > tolerance && iteration < max\_iterations  
 global iteration += 1  
 global Pr2 = P2 / Pc  
 global Tr2 = T2 / Tc  
 global Z2 = get\_Z(Pr2, Tr2, Pr\_grid, Tr\_grid, Z0\_table, Z1\_table, w)  
   
 println("Debug: Z2 = ", Z2, " type: ", typeof(Z2))  
   
 P2\_new = (Z2 \* R \* T2) / v2  
 global abs\_error = abs(P2\_new - P2)  
 global P2 = P2\_new  
   
 println("Debug: P2 = ", P2, " type: ", typeof(P2))  
end  
  
if iteration == max\_iterations  
 println("迭代未收敛。")  
else  
 @printf("\n使用普压法迭代试差计算得到的 P2 = %.6f MPa\n", P2 / 1e6)  
 @printf("迭代次数: %d\n", iteration)  
end  
  
# 计算并打印其他参数  
Z2\_final = (P2 \* v2) / (R \* T2)  
Pr2\_final = P2 / Pc  
Tr2\_final = T2 / Tc  
  
@printf("\n其他参数:\n")  
@printf("v2 = %.6f m³/mol\n", v2)  
@printf("Z2 = %.6f\n", Z2\_final)  
@printf("Pr2 = %.6f\n", Pr2\_final)  
@printf("Tr2 = %.6f\n", Tr2\_final)

运行后结果如下:

julia> 正在运行 T3.jl  
使用普维法计算得到的物质的量 n = 1415.336161 mol  
Debug: Z2 = 0.7509759003926045 type: Float64  
Debug: P2 = 2.791685867650969e7 type: Float64  
Debug: Z2 = 0.8351167398465856 type: Float64  
Debug: P2 = 3.1044719267950337e7 type: Float64  
Debug: Z2 = 0.806687038162363 type: Float64  
Debug: P2 = 2.9987870487957727e7 type: Float64  
...（省略部分）  
Debug: Z2 = 0.8138669622319971 type: Float64  
Debug: P2 = 3.025477775549488e7 type: Float64  
Debug: Z2 = 0.8138669622320426 type: Float64  
Debug: P2 = 3.0254777755496576e7 type: Float64  
Debug: Z2 = 0.8138669622320279 type: Float64  
Debug: P2 = 3.0254777755496033e7 type: Float64  
  
使用普压法迭代试差计算得到的 P2 = 30.254778 MPa  
迭代次数: 29  
  
其他参数:  
v2 = 0.000100 m³/mol  
Z2 = 0.813867  
Pr2 = 2.682161  
Tr2 = 1.105879

### 结果分析

* **收敛性**：检查迭代是否能在设定的最大迭代次数内收敛。
* **准确性**：结果的准确性依赖于插值的精度和表格数据的准确性。
* **计算效率**：尽管试差法可能需要多次迭代，但每次迭代的计算量相对较小，因此整体效率是可接受的。

## T4.jl

### 问题描述

在铂催化剂上，乙烯深度氧化的动力学方程可以表示为：

其中 和 分别为乙烯及氧的分压。

### 求解思路

已知在473K等温下的实验数据，需要使用曲线拟合函数 lsqcurvefit 求出动力学方程的参数 和，并绘制拟合前后各点速度 的散点图。

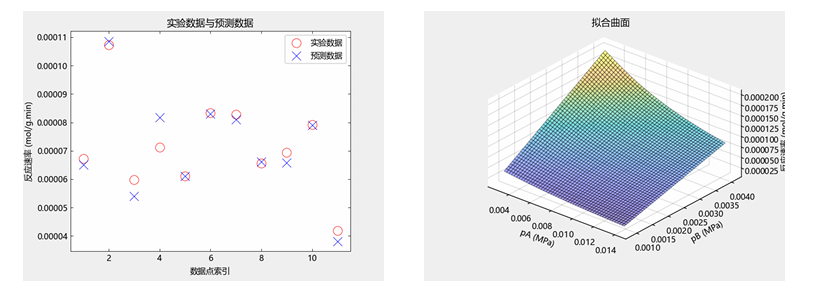
1. **数据准备**：将 和 合并为一个一维数组作为输入数据。
2. **定义动力学模型函数**：根据给定的动力学方程，编写模型函数。
3. **参数估计**：使用最小二乘法 (lsqcurvefit) 对动力学模型进行参数拟合。
4. **结果评估**：计算拟合的决定系数 来评估模型的拟合质量。
5. **结果可视化**：绘制实验数据和拟合曲面的三维散点图，以直观展示拟合效果。

### MWORKS 程序

using TyBase  
using TyMath  
using TyOptimization  
using TyPlot  
using Printf  
  
# 动力学模型函数  
function kinetic\_model(res\_opt, p)  
 k, KB = res\_opt  
 n = length(p) ÷ 2  
 # 分解 p 为 pA 和 pB  
 pA = p[1:n]  
 pB = p[n+1:end]  
 return @. (k \* pA \* pB) / (1 + KB \* pB)^2  
end  
  
# 实验数据  
pA = [8.99, 14.22, 8.86, 8.32, 4.37, 7.75, 7.75, 6.17, 6.13, 6.98, 2.87] .\* 1e-3 # 转换为 MPa  
pB = [3.23, 3.00, 4.08, 2.03, 0.89, 1.74, 1.82, 1.73, 1.73, 1.56, 1.06] .\* 1e-3 # 转换为 MPa  
r = [0.672, 1.072, 0.598, 0.713, 0.610, 0.834, 0.828, 0.656, 0.694, 0.791, 0.418] .\* 1e-4 # 转换为 mol/g.min  
  
# 将 pA 和 pB 合并为一个一维数组  
p = vcat(pA, pB)  
  
# 参数上下界  
lb = [0.0, 0.0]  
ub = [Inf, Inf]  
  
# 网格搜索范围  
k\_values = range(1.0, stop=1000.0, length=100)  
KB\_values = range(1.0, stop=2000.0, length=1000)  
  
# 设定R²阈值  
threshold\_R\_squared = 0.95  
  
# 初始化最佳结果  
global best\_R\_squared = -Inf  
global best\_params = nothing  
global best\_resnorm = Inf  
  
# 网格搜索  
for k\_init in k\_values  
 for KB\_init in KB\_values  
 local initial\_params = [k\_init, KB\_init]  
 local res\_opt, resnorm = lsqcurvefit(kinetic\_model, initial\_params, p, r, lb, ub)  
   
 # 计算 R²  
 local predicted\_r = kinetic\_model(res\_opt, p)  
 local ss\_res = sum((r - predicted\_r).^2)  
 local ss\_tot = sum((r .- mean(r)).^2)  
 local R\_squared = 1 - ss\_res / ss\_tot  
   
 # 更新最佳结果  
 if R\_squared > best\_R\_squared  
 global best\_R\_squared = R\_squared  
 global best\_params = res\_opt  
 global best\_resnorm = resnorm  
 end  
   
 # 如果达到阈值，则停止搜索  
 if best\_R\_squared >= threshold\_R\_squared  
 break  
 end  
 end  
 if best\_R\_squared >= threshold\_R\_squared  
 break  
 end  
end  
  
# 打印最佳结果  
@printf("最佳拟合参数:\n")  
@printf("k = %.6f\n", best\_params[1])  
@printf("KB = %.6f\n", best\_params[2])  
@printf("最佳 R² = %.6f\n", best\_R\_squared)  
@printf("最佳残差范数: %.6e\n", best\_resnorm)  
  
# 预测 r 值  
predicted\_r = kinetic\_model(best\_params, p)  
  
# 绘制实验数据和预测点  
figure()  
scatter(1:length(r), r, label="实验数据", color="red", s=100, marker="o") # 实验数据点，红色圆圈  
hold("on")  
scatter(1:length(predicted\_r), predicted\_r, label="预测数据", color="blue", s=100, marker="x") # 预测数据点，蓝色叉号  
xlabel("数据点索引")  
ylabel("反应速率 (mol/g.min)")  
title("实验数据与预测数据")  
legend()  
  
# 绘制拟合曲面  
figure()  
pA\_range = range(minimum(pA), stop=maximum(pA), length=50)  
pB\_range = range(minimum(pB), stop=maximum(pB), length=50)  
fitted\_surface = [kinetic\_model(best\_params, [pA, pB])[1] for pA in pA\_range, pB in pB\_range]  
surface(pA\_range, pB\_range, fitted\_surface, alpha=0.5, color="blue")  
xlabel("pA (MPa)")  
ylabel("pB (MPa)")  
zlabel("反应速率 (mol/g.min)")  
title("拟合曲面")  
plt\_view(40, 35);

运行后结果如下:

julia> 正在运行 T4.jl  
最佳拟合参数:  
k = 112.000000  
KB = 1877.938939  
最佳 R² = 0.935842  
最佳残差范数: 1.788820e-10

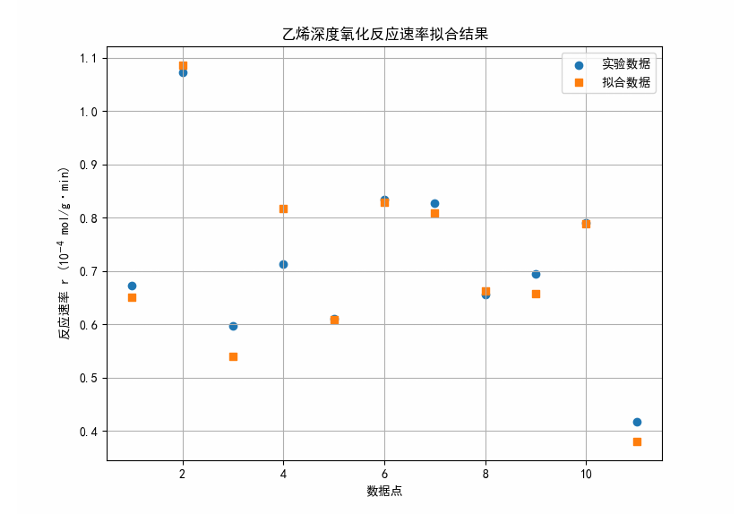


因上述lsqcurvefit运行效果不佳，经常因不同的初值陷入局部最优，所以上述采用了网格搜索，用**python**最优化库中curve\_fit进行相同的操作相应**python**代码如下：

import numpy as np  
import matplotlib.pyplot as plt  
from scipy.optimize import curve\_fit  
  
# 设置中文显示  
plt.rcParams['font.sans-serif'] = ['SimHei'] # 使用黑体  
plt.rcParams['axes.unicode\_minus'] = False # 正常显示负号  
  
# 实验数据  
pA = np.array([8.99, 14.22, 8.86, 8.32, 4.37, 7.75, 7.75, 6.17, 6.13, 6.98, 2.87]) \* 1e-3 # MPa  
pB = np.array([3.23, 3.00, 4.08, 2.03, 0.89, 1.74, 1.82, 1.73, 1.73, 1.56, 1.06]) \* 1e-3 # MPa  
r = np.array([0.672, 1.072, 0.598, 0.713, 0.610, 0.834, 0.828, 0.656, 0.694, 0.791, 0.418]) \* 1e-4 # mol/g·min  
  
# 定义动力学方程  
def kinetics(X, k, KB):  
 pA, pB = X  
 return k \* pA \* pB / (1 + KB \* pB)\*\*2  
  
# 执行曲线拟合  
initial\_guess = [1.0, 1.0]  
params, covariance = curve\_fit(kinetics, (pA, pB), r, p0=initial\_guess)  
  
# 输出拟合结果  
k, KB = params  
print(f"拟合参数:\nk = {k:.6e}\nKB = {KB:.6e}")  
  
# 计算拟合后的速率  
r\_fitted = kinetics((pA, pB), \*params)  
  
# 计算R²  
SS\_tot = np.sum((r - np.mean(r))\*\*2)  
SS\_res = np.sum((r - r\_fitted)\*\*2)  
R\_squared = 1 - SS\_res / SS\_tot  
print(f"R² = {R\_squared:.4f}")  
  
# 绘制散点图  
plt.figure(figsize=(8, 6))  
plt.scatter(range(1, len(r) + 1), r \* 1e4, label="实验数据", marker='o')  
plt.scatter(range(1, len(r\_fitted) + 1), r\_fitted \* 1e4, label="拟合数据", marker='s')  
plt.xlabel("数据点")  
plt.ylabel("反应速率 r (10$^{-4}$ mol/g·min)")  
plt.title("乙烯深度氧化反应速率拟合结果")  
plt.legend()  
plt.grid(True)  
plt.show()

运行后结果如下:

PS E:\LGRepository\Mworks.syslab> cd e:/LGRepository/Mworks.syslab/10.24HomeWork  
PS E:\LGRepository\Mworks.syslab\10.24HomeWork> & C:/Users/Public/TongYuan/.julia/miniforge3/python.exe e:/LGRepository/Mworks.syslab/10.24HomeWork/T4.py  
拟合参数:  
k = 1.110885e+02  
KB = 1.868472e+03  
R² = 0.9359



### 结果分析

* **动力学模型函数**：函数 kinetic\_model 接收参数 和，以及压力数据，计算预测的反应速率。
* **参数估计**：通过网格搜索初始化参数，然后使用 lsqcurvefit 进行参数优化。
* **结果评估**：计算决定系数，若 达到设定阈值（例如0.95），则认为拟合质量良好。
* **结果可视化**：使用散点图和曲面图展示实验数据和拟合模型，以便比较实验值与模型预测值。