第一题：拟合

1.

是的，我们认为与的关系是合理的。

a.

|  |  |  |
| --- | --- | --- |
| 地址编号 | log P | log V |
| 1 | 12.74 | 1.57 |
| 2 | 13.90 | 1.77 |
| 3 | 8.61 | 1.20 |
| 4 | 10.81 | 1.59 |
| 5 | 14.11 | 1.73 |
| 6 | 5.90 | 1.02 |
| 7 | 7.82 | 0.82 |
| 8 | 11.27 | 1.35 |
| 9 | 13.67 | 1.65 |
| 10 | 9.55 | 1.31 |
| 11 | 10.07 | 1.18 |
| 12 | 11.17 | 1.46 |
| 13 | 12.63 | 1.49 |
| 14 | 11.84 | 1.48 |
| 15 | 14.77 | 1.62 |

表 1 与

b.

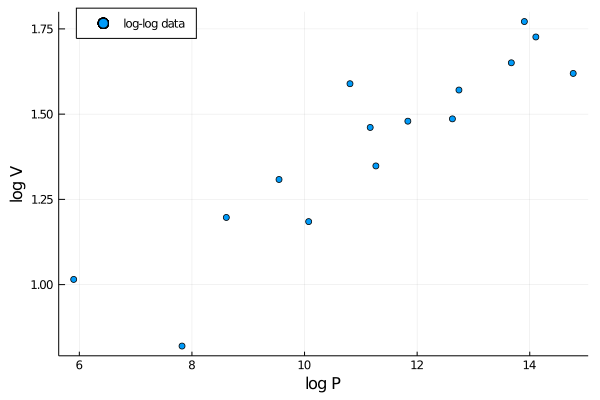


图 1 数据散点图

c.

图二中的蓝色线条是我们目测的拟合线。

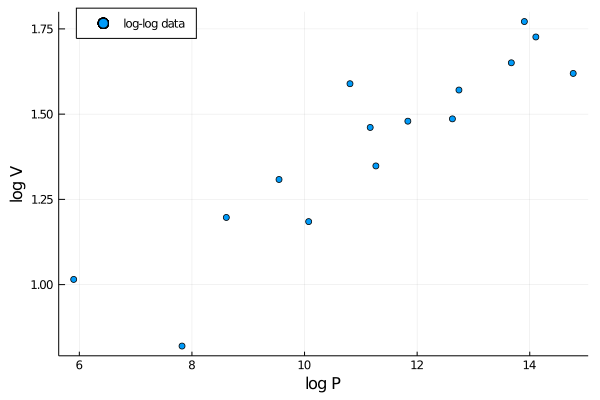


图 2 目测拟合线

d.

因为c中的蓝色线条大致经过点（7.6，1）和（14，1.7），所以斜率大致为0.109。

因此可以得知大致的拟合线方程为 y=0.109x+0.172，所以横截距约为-1.578，纵截距约为0.172。

e.

根据代码所得结果，, 所以。

f.

因为, 所以.

2.

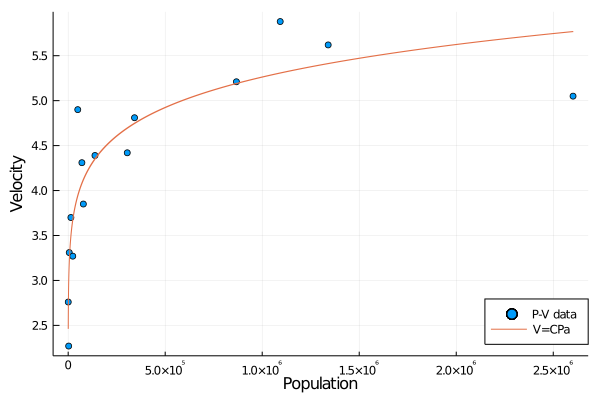


图 3

3.

|  |  |  |
| --- | --- | --- |
| 地址编号 | 实测速度 | 预测速度 |
| 1 | 4.81 | 4.75 |
| 2 | 5.88 | 5.31 |
| 3 | 3.31 | 3.19 |
| 4 | 4.90 | 3.94 |
| 5 | 5.62 | 5.41 |
| 6 | 2.76 | 2.46 |
| 7 | 2.27 | 2.96 |
| 8 | 3.85 | 4.12 |
| 9 | 5.21 | 5.19 |
| 10 | 3.70 | 3.49 |
| 11 | 3.27 | 3.67 |
| 12 | 4.31 | 4.08 |
| 13 | 4.42 | 4.69 |
| 14 | 4.39 | 4.35 |
| 15 | 5.05 | 5.77 |

表 2 预测速度

4.

经计算，伯恩斯坦误差为0.29，结果显示这个模型的优点为预测数据与实际数据十分相近，平均误差小。

第二题：多项式插值

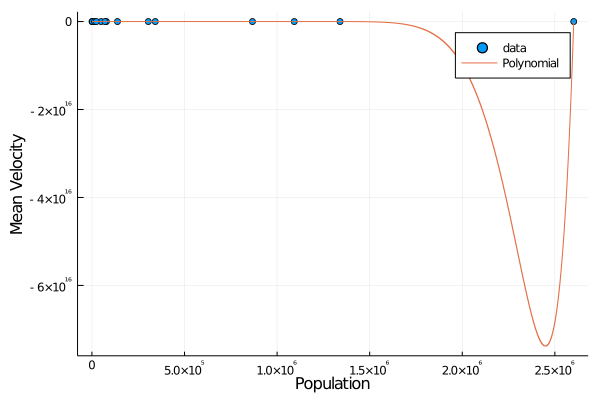


图 4 多项式插值

缺点：当使用拉格朗日插值算法时，多项式次数越多，就会有可能产生龙格现象，即插值结果偏离原函数。对这组数据使用拉格朗日插值算法时，次数达到了14， 相对较高，观察插值后的图像，可发现在左右，预测函数图像突然大幅度变化，速度甚至变成负数，很明显不是一个合理的函数图像。这就是龙格现象。

第三题：三次样条插值

1.

根据三次样条插值多项式得到的在处的导数值为31.494

根据得到的导数值为31.500

相对误差仅为0.019%，两者在处导数值几乎相等。

2.

根据三次样条插值多项式得到的面积为9.485023

根据得到的面积为9.485596

相对误差仅为0.006%，两者在和之间的面积几乎相等。

Julia 代码

第一题

using Plots

using LinearAlgebra  
  
P = [341948, 1092759, 5491, 49375, 1340000, 365, 2500, 78200, 867023, 14000, 23700, 70700, 304500, 138000, 2602000]  
V = [4.81, 5.88, 3.31, 4.90, 5.62, 2.76, 2.27, 3.85, 5.21, 3.70, 3.27, 4.31, 4.42, 4.39, 5.05]  
  
pp = 365:2602000  
vv = zeros(length(pp))  
  
for i in 1:length(pp)  
    vv[i] = 1.397\*(pp[i]^0.096)  
end  
  
scatter(P, V,label="P-V data",legend=:bottomright,xlabel="Population",ylabel="Velocity")  
plot!(pp,vv,label="V=CPa",legend=:bottomright)  
  
savefig("1\_1.png")  
  
logP = log.(P)  
  
logV = log.(V)  
  
scatter(logP, logV,label="log-log data",legend=:topleft,xlabel="log P",ylabel="log V")  
  
savefig("1\_2.png")  
  
z = fill(1.0,(length(P),))  
  
logC = (dot(z,logV)\*dot(logP,logP)-dot(z,logP)\*dot(logP,logV))/(dot(z,z)\*dot(logP,logP)-dot(z,logP)^2)  
  
a = (dot(z,z)\*dot(logP,logV)-dot(z,logP)\*dot(z,logV))/(dot(z,z)\*dot(logP,logP)-dot(z,logP)^2)  
  
f =x-> logC + a\*x  
  
P1,P2 = extrema(logP)  
  
pp = P1:0.1:P2  
  
vv = f.(pp)  
  
plot!(pp,vv,label="least square approximation",legend=:topleft,xlabel="log P",ylabel="log V")  
  
savefig("1\_3.png")

第二题

using Plots  
  
P = [341948, 1092759, 5491, 49375, 1340000, 365, 2500, 78200, 867023, 14000, 23700, 70700, 304500, 138000, 2602000]  
V = [4.81, 5.88, 3.31, 4.90, 5.62, 2.76, 2.27, 3.85, 5.21, 3.70, 3.27, 4.31, 4.42, 4.39, 5.05]  
  
function get\_Lagrange(xx,yy,x)  
    @assert length(xx) == length(yy)  
    y = 0.0  
    N = length(xx)  
    for i in 1:N  
        w = 1.0  
        for j in 1:N  
            if i != j  
                w \*= (x - xx[j])/(xx[i] -  xx[j])  
            end  
        end  
        y += yy[i]\*w  
    end  
    return y  
end  
  
Pmin, Pmax = extrema(P)  
  
every\_pop = Pmin:Pmax  
vel = zeros(length(every\_pop))  
for i in 1:length(every\_pop)  
    vel[i] = get\_Lagrange(P,V,every\_pop[i])  
end  
  
scatter(P,V,label = "data")  
  
savefig("2\_1.png")  
  
p = plot!(every\_pop, vel, label = "P(x)", xlabel = "year", ylabel = "Population")  
  
savefig("2\_2.png")

第三题

using Plots  
  
x = 3.0:0.1:3.9  
  
y = [20.08,22.20,24.53,27.12,29.96,33.11,36.60,40.45,44.70,49.40]  
  
xx = 3.0:0.01:3.9 #xx = x(1):0.01:x(N)  
yy = zero(xx)  
N = length(x)  
  
a = zeros(Float64,N,)  
b = zeros(Float64,N-1,)  
c = zeros(Float64,N,)  
d = zeros(Float64,N-1,)  
h = zeros(Float64,N-1,)  
  
for i in 1:N-1  
    h[i] = x[i+1] - x[i]  
end  
  
a .= y  
  
A = zeros(Float64,N,N)  
B = zeros(Float64,N,1)  
  
A[1,1] = 1.0  
A[N,N] = 1.0  
  
for i in 2:N-1  
    A[i,i-1] = h[i-1]  
    A[i,i] = 2.0\*(h[i-1]+h[i])  
    A[i,i+1] = h[i]  
      
    B[i] = 3.0\*(a[i+1]-a[i])/h[i] - 3.0\*(a[i]-a[i-1])/h[i-1]  
end  
  
c = A\B  
  
for i in 1:N-1  
    b[i] = (a[i+1]-a[i])/h[i] - h[i]/3.0\*(c[i+1]+2\*c[i])  
    d[i] = (c[i+1]-c[i])/3.0/h[i]  
end  
  
function get\_cubic\_data(\_x,a,b,c,d,x)  
    r = a[1]  
    for i in 1:N-1  
        if \_x < x[i+1]  
            r = a[i]+b[i]\*(\_x-x[i])+c[i]\*(\_x-x[i])^2+d[i]\*(\_x-x[i])^3  
            return r  
        end  
    end  
    return a[N]  
end  
  
for i in 1:length(xx)  
    yy[i] = get\_cubic\_data(xx[i],a,b,c,d,x)  
end  
  
scatter(x,y,label="data")  
plot!(xx,yy,label="P(x)",title="Natural spline",xlabel = "x",ylabel = "y")  
  
x\_est = 3.45  
  
N\_est = round(Int,fld((x\_est-x[1]),0.1))+1  
  
y\_der = b[N\_est]+2\*c[N\_est]\*(x\_est-x[N\_est])+3\*d[N\_est]\*(x\_est-x[N\_est])^2  
  
exp(x\_est)  
  
x1 = 3.3  
x2 = 3.6  
  
N1 = round(Int,(x1 - x[1])/0.1)+1  
N2 = round(Int,(x2 - x[1])/0.1)+1  
  
Area = 0  
for i in N1:N2-1  
    P(xi) = a[i]\*xi + 1/2\*b[i]\*(xi-x[i])^2 + 1/3\*c[i]\*(xi-x[i])^3 + 1/4\*d[i]\*(xi-x[i])^4  
    Area = Area + P(x[i+1]) - P(x[i])  
end  
  
Area  
  
Intergration\_exp = exp(3.6)-exp(3.3)