赛区评阅编号（由赛区组委会填写）：

2019 高教社杯全国大学生数学建模竞赛

承　诺　书

我们仔细阅读了《全国大学生数学建模竞赛章程》和《全国大学生数学建模竞赛参赛规则》（2019 年修订稿，以下简称为“竞赛章程和参赛规则”，可从全国大学生数学建模竞赛网站下载）。

我们完全清楚，在竞赛开始后参赛队员不能以任何方式，包括电话、电子邮件、“贴吧”、QQ 群、微信群等，与队外的任何人（包括指导教师）交流、讨论与赛题有关的问题；无论主动参与讨论还是被动接收讨论信息都是严重违反竞赛纪律的行为。

我们完全清楚，抄袭别人的成果是违反竞赛章程和参赛规则的行为；如果引用别人的成果或资料（包括网上资料），必须按照规定的参考文献的表述方式列出，并在正文引用处予以标注。

我们以中国大学生名誉和诚信郑重承诺，严格遵守竞赛章程和参赛规则，以保证竞赛的公正、公平性。如有违反竞赛章程和参赛规则的行为，我们将受到严肃处理。

我们授权全国大学生数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

我们参赛选择的题号（从 A/B/C/D/E 中选择一项填写）：A

我们的报名参赛队号（12 位数字全国统一编号）：201901001042

参赛学校（完整的学校全称，不含院系名）：北京大学

参赛队员 (打印并签名) ：1. 谭淞宸

2. 曹宇创

3. 李婧宜

指导教师或指导教师组负责人 (打印并签名)：无

（指导教师签名意味着对参赛队的行为和论文的真实性负责）

日期: 2019 年 09 月 13 日

（请勿改动此页内容和格式。此承诺书打印签名后作为纸质论文的封面，注意电子版论文中不得出现此页。以上内容请仔细核对，如填写错误，论文可能被取消评奖资格。）

赛区评阅编号（由赛区组委会填写）：

2019 高教社杯全国大学生数学建模竞赛

编 号 专 用 页

赛区评阅记录（可供赛区评阅时使用）：

评阅人						
备注						

送全国评阅统一编号（由赛区组委会填写）：

全国评阅随机编号（由全国组委会填写）：

（请勿改动此页内容和格式。此编号专用页仅供赛区和全国评阅使用，参赛队打印后装订到纸质论文的第二页上。注意电子版论文中不得出现此页。）

高压油管的压力控制

摘要

高压油管被广泛应用于柴油机等燃油发动机中，燃油经过高压油泵进入高压油管，再由喷口喷出。燃油的周期性进入与喷出直接影响其所匹配燃油机性能的稳定性和工作的可靠性。具体而言，管内压力的波动将导致排气温度不稳定, 降低了催化剂的转换效率, 使燃油机排放一致性下降。因此, 减小管内压力的波动, 是当前高压燃油系统亟需解决的技术难题。

对于问题一：

对于问题二：

对于问题三：

最后，

一、 问题的重述

1.1 引言

燃油进入和喷出高压油管是许多燃油发动机工作的基础，图 1 给出了某高压燃油系统的工作原理，燃油经过高压油泵从 A 处进入高压油管，再由喷口 B 喷出。燃油进入和喷出的间歇性工作过程会导致高压油管内压力的变化，使得所喷出的燃油量出现偏差，从而影响发动机的工作效率。

图 **1**　高压油管示意图

1.2 问题的提出

1.2.1 问题一

某型号高压油管的内腔长度为 500mm，内直径为 10mm，供油入口 A 处小孔的直径为 1.4mm，通过单向阀开关控制供油时间的长短，单向阀每打开一次后就要关闭 10ms。喷油器每秒工作 10 次，每次工作时喷油时间为 2.4ms，喷油器工作时从喷油嘴 B 处向外喷油的速率如图 2 所示。高压油泵在入口 A 处提供的压力恒为 160 MPa，高压油管内的初始压力为 100 MPa。如果要将高压油管内的压力尽可能稳定在 100 MPa 左右，如何设置单向阀每次开启的时长? 如果要将高压油管内的压力从 100 MPa 增加到 150 MPa，且分别经过约 2 s、5 s 和 10 s 的调整过程后稳定在 150 MPa，单向阀开启的时长应如何调整?

二、 问题的分析

2.1 问题一的分析

2.1.1 第一问

问题一的第一问要求我们通过设置单向阀每次开启的时长，来使得高压油管内的压力稳定在 1×10^5 kPa。由题目所给信息，在喷油器的一个工作周期（100 ms）内，喷油器喷出的油量为 44 mm^3，而油管的容积为 3.93×10^4 mm^3，可见油管中的油量变化在 0.1% 上下，可以近似认为压强不变。

在一个工作周期内，喷油器喷出的流量已知；且在压强不变的近似下，单向阀流入的流速也已知，因而我们可以根据质量守恒定律列出一个工作周期内单向阀应当开启的时间，进行等比例转换后即能得到单向阀每次开启的时间。

在上述讨论过程中，忽略了高压油管内的压强变化，并且忽略了同一时间段内单向阀和喷油器的相互影响，这一影响的大小可以通过对体系进行精确的数值模拟来考察。在数值模拟中，我们采取较小的时间步长（如 0.01 ms），在每一步完成以下计算：

1. 检查该时刻单向阀是否处于开启状态；
2. 如果开启，根据两边压强计算单向阀流入的流量；
3. 检查该时刻喷油嘴是否处于开启状态；
4. 如果开启，根据题目所给的流速随时间变化关系计算喷油嘴流出的流量；
5. 将出入流量乘以相应密度换算为高压油管内燃油的质量变化，进而计算得到更新后的油管内燃油密度；
6. 将油管内燃油密度的变化通过弹性模量转化为油管内压强的变化；
7. 将上一步得到的压强继续用于下一步的计算。

通过数值模拟方法，我们可以得到任意时刻油管内的压强，进而计算压强相对于 10^5 kPa 的偏离幅度大小，进而验证我们通过质量守恒定律选取的开启时间是否是使得系统偏离幅度大小最小的开启时间。

2.2 问题二的分析

2.3 问题三的分析

三、 模型的建立与求解

3.1 符号说明

表 **1**　符号说明

符号	物理量
P_1	油泵压力
V_1	油泵体积
ρ_1	油泵中燃油密度
P_2	油管压力
P_2^*	目标油管压力
d_2	油管直径
l_2	油管长度
V_2	油管体积
ρ_2	油管中燃油密度
d_A	A 处小孔直径
S_A	A 处小孔面积
Q_A	A 处小孔流量

表 **2**　符号说明

符号	物理量
d_B	B 处喷孔直径
S_B	B 处喷孔直径
Q_B	B 处喷油嘴流量
τ_0	油管的喷油周期
τ_A	喷油周期内 A 的开启时间
τ_A^+	A 每次的开启时间
τ_A^-	A 每次的关闭时间
$\tau_A^\pm$	A 一次开启/关闭的总时间
θ	凸轮各点极角
α	凸轮极轴方位角
h	凸轮最高点高度
ω	凸轮角速度
γ	圆锥半角
z	针阀高度

本文中，所有长度均以 mm 为单位，所有质量均以 mg 为单位，所有时间均以 ms 为单位。由于压强的量纲为 $\mathrm{ML^{-1}T^{-2}}$，当使用上述三个基本力学量的单位时，压强的自然单位是 kPa，因此本文中所有压强均以 kPa 为单位。

3.2 问题一的求解

3.2.1 第一小问

理论计算 首先假设油管的压力近似保持在 $P_2 = P_2^* = 1.00 \times 10^5$ kPa，因而油管内燃油密度 ρ_2 为常数。在喷油器的一个工作周期内，从喷油嘴流出的燃油质量，由流速对时间的积分与密度的乘积给出：

$$\Delta m = \int_0^{\tau_0} Q_B \rho_2 \mathrm{d}t = 37.4 \text{ mg}$$

这些质量应当由从单向阀进入的燃油补充，因此在喷油器一个工作周期内，单向阀平均应该开启的时间应当由损失的质量和补充燃油质量的速度决定：

$$\begin{aligned} \tau_A &= \frac{\Delta m}{Q_A \rho_1} \\ &= \frac{\Delta m}{\rho_1 C A \sqrt{2(P_1 - P_2)/\rho_1}} \\ &= 2.76 \text{ ms} \end{aligned}$$

若我们考虑系统进行较长时间的演化，则在一个周期内单向阀平均开启的时间应该与相同。因此，单向阀开启的时长 τ_A^+ 与它关闭的时长 τ_A^- 应该满足如下比例式：

$$\frac{\tau_A^+}{\tau_A^-} = \frac{\tau_A}{\tau_0 - \tau_A}$$

代入已知数据，可以求出 $\tau_A = 0.284$ ms。

数值模拟 在数值模拟中，我们设定初始条件为 $P_2 = 1.00 \times 10^5$ kPa，且时间零点恰好位于喷油器一个工作周期的开始，但可以位于单向阀一个工作周期中的任意一点，该点处相位为 t_0。

系统的演化可以用如下关系表示，其中左箭头「$\leftarrow$」是程序设计中的赋值等号：

$$\begin{aligned} Q_A &\leftarrow Q_A(t) \\ m_2 &\leftarrow m_2 + Q_A \times \rho_1 \times dt \\ Q_B &\leftarrow Q_B(t) \\ m_2 &\leftarrow m_2 - Q_B \times \rho_2 \times dt \\ \rho_2 &\leftarrow m_2 / V_2 \\ P_2 &\leftarrow P(\rho_2) \end{aligned}$$

将系统进行 N 个周期的演化，获得各个时刻油管的压强后，我们可以定义系统关于目标压力的均方偏差 D 作为压强-时间函数的泛函：

$$D[P_2(t)] = \frac{1}{N\tau_0}\int_0^{N\tau_0}[P_2(t) - P_2^*]^2\,\mathrm{d}t$$

给定一系列开启时间 τ_A^+，计算它们的均方偏差，就可以从中找出使得均方偏差最小的开启时间。

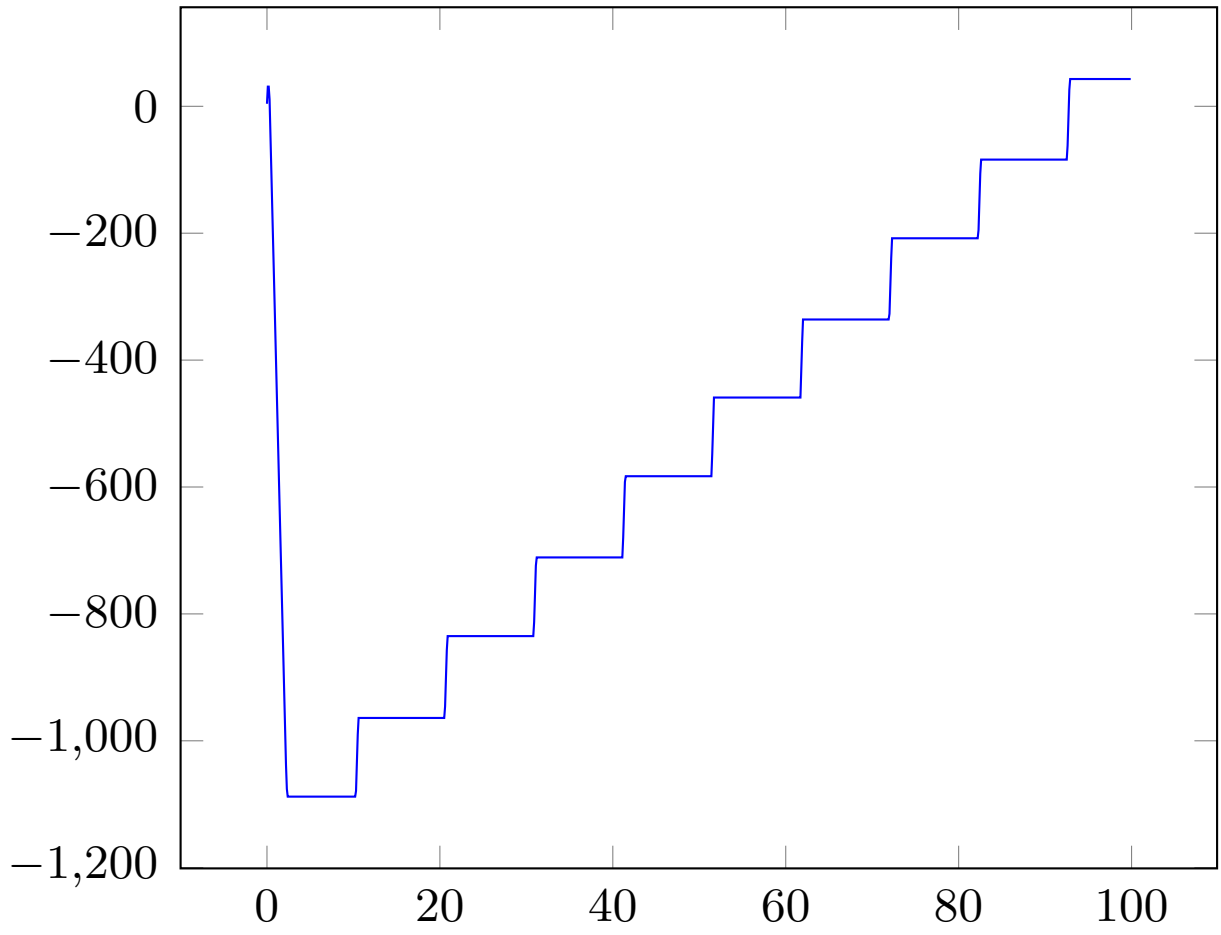


图 **2**　**P-t** 图

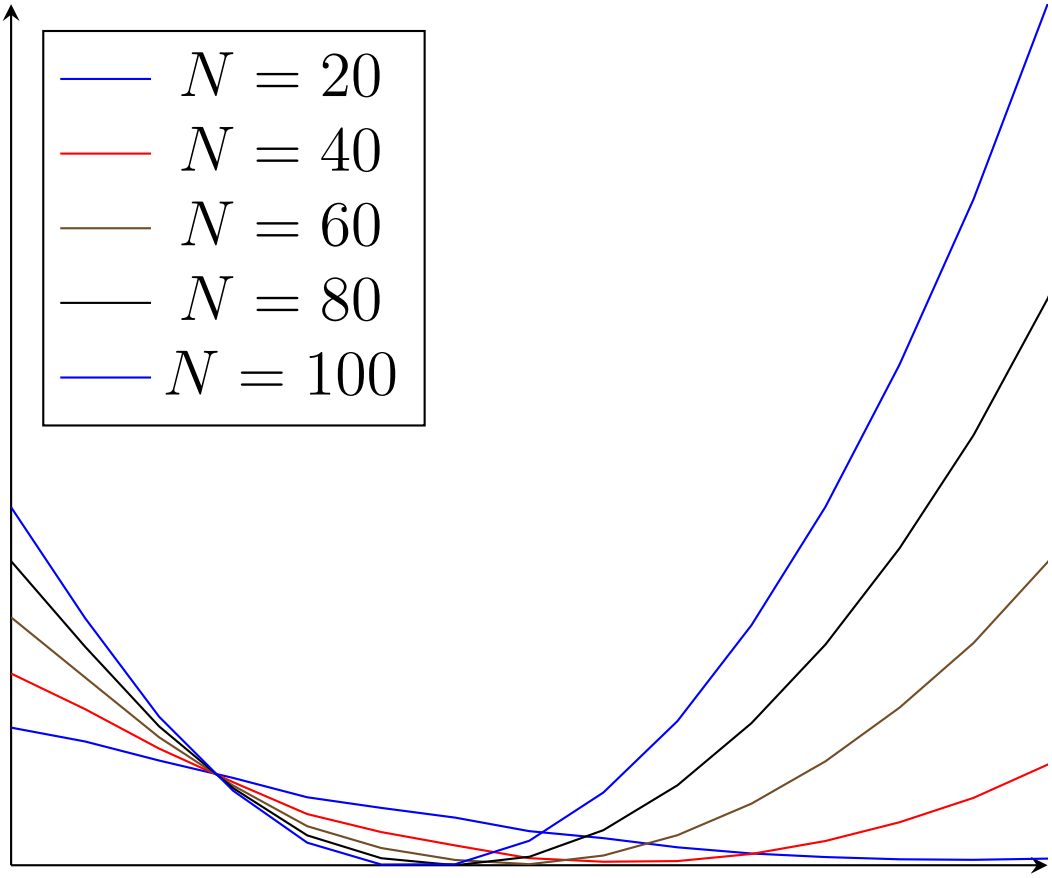


图 **3**　不同周期的 **D-tau** 图

3.3 问题二的求解

3.4 问题三的求解

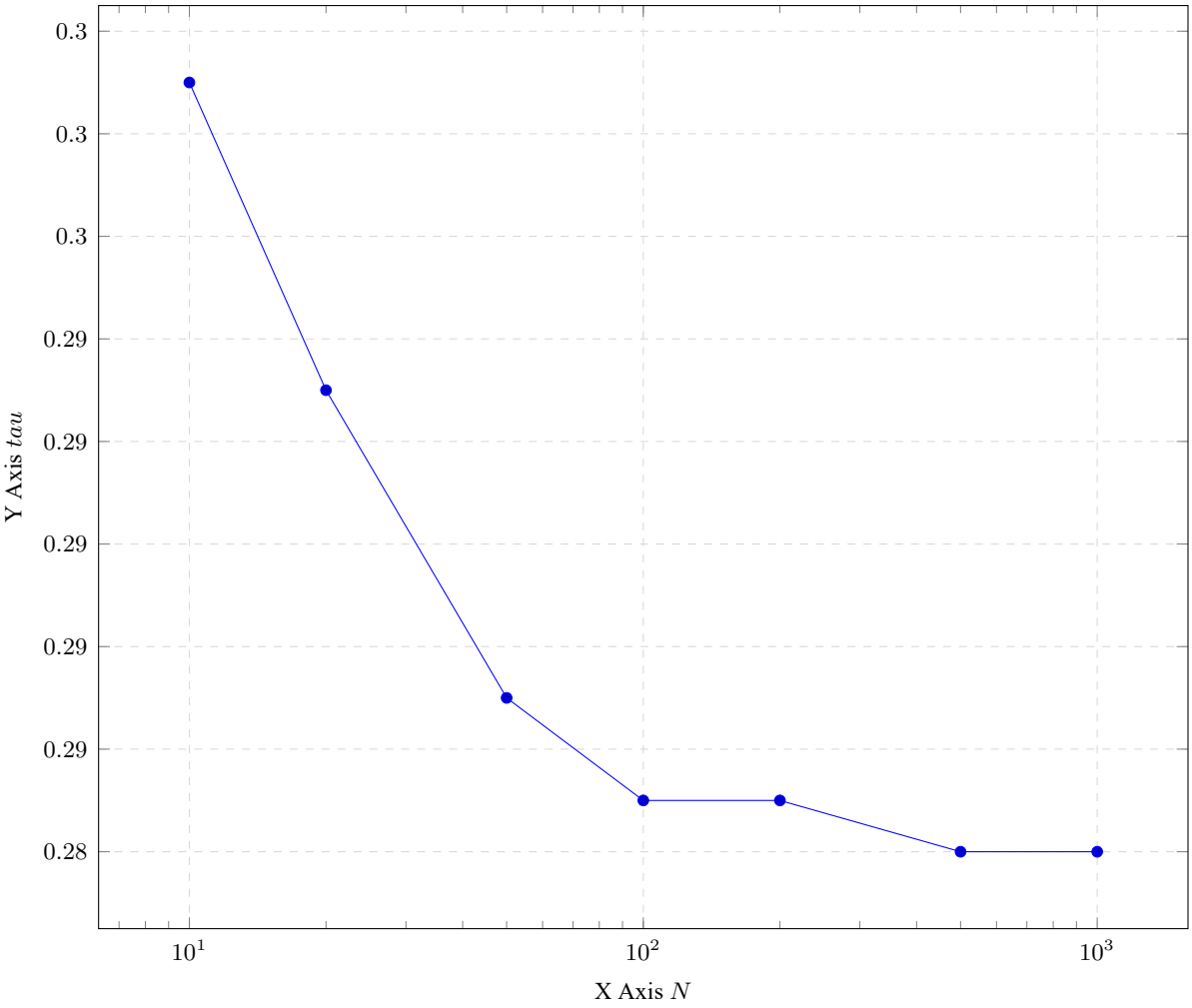


图 **4**　自由开始时间随模拟周期的收敛性

四、 模型的评价

优点、缺点、潜在改进空间和应用范围。

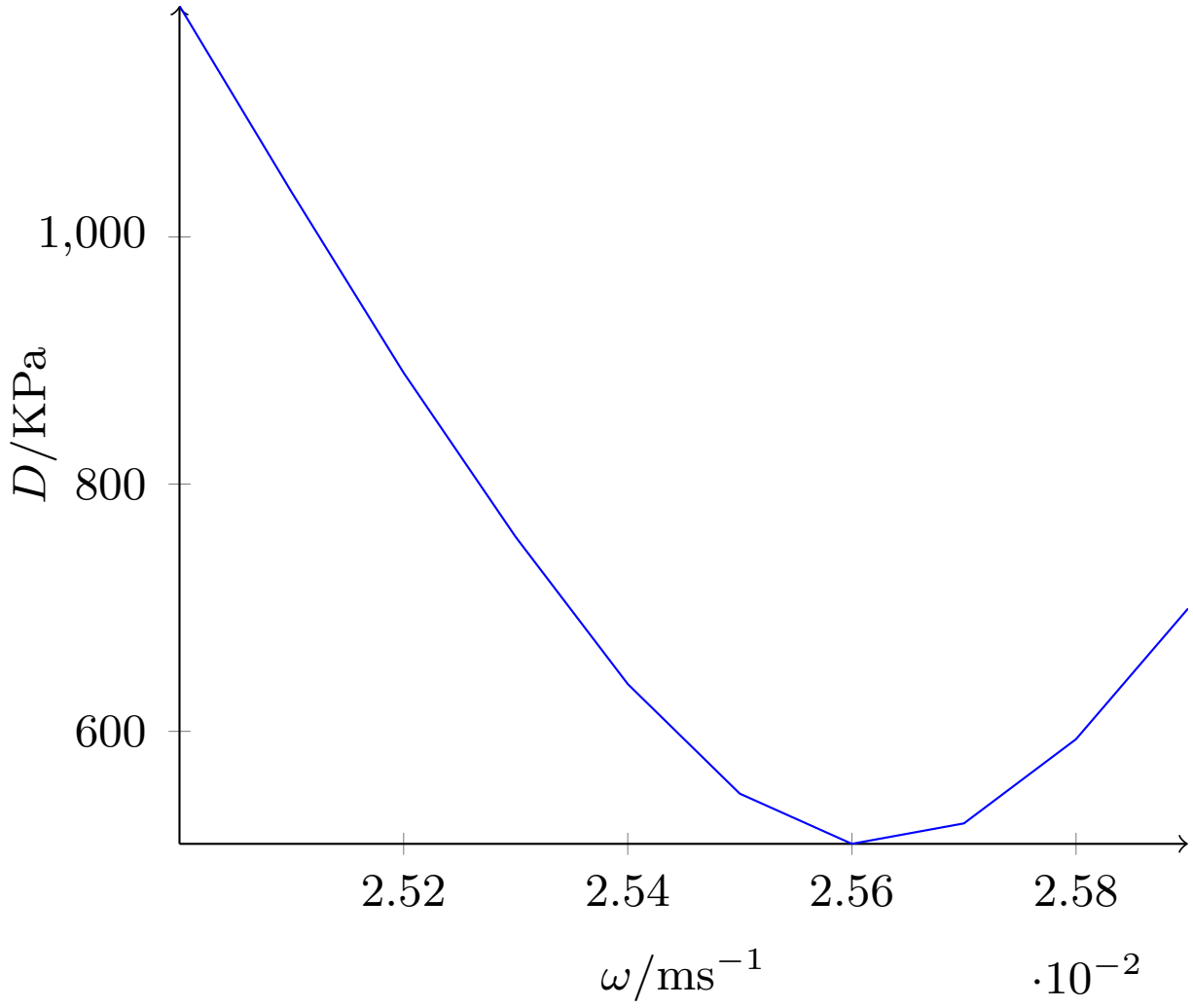


图 5　D-Omega 图

五、 其它小功能

5.1 脚注

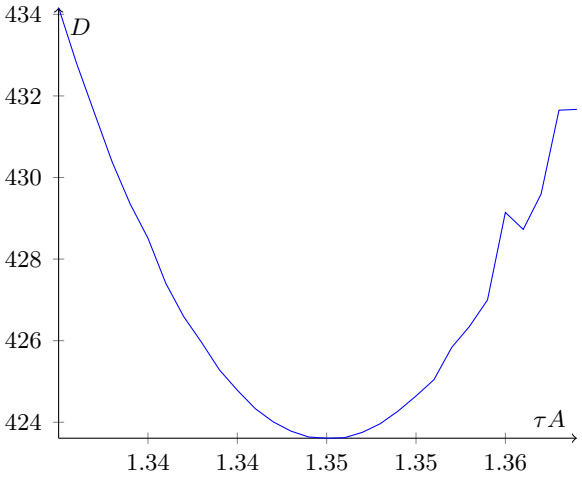
利用 `\footnote{具体内容}` 可以生成脚注[1]。

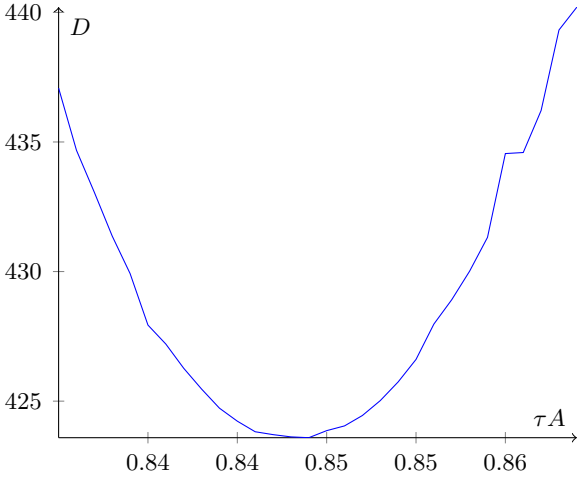
六、 参考文献与引用

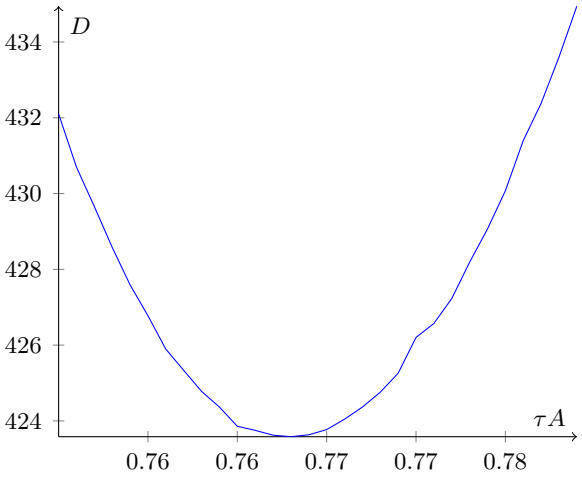
参考文献对于一篇正式的论文来说是必不可少的，在建模中重要的参考文献当然应该列出。LaTeX 在这方面的功能也是十分强大的，下面进介绍一个比较简单的参考文献制作方法。有兴趣的可以学习 `bibtex` 或 `biblatex` 的使用。

LaTeX 的入门书籍可以看《LaTeX 入门》[?]。这是一个简单的引用，用 `\cite{bibkey}` 来完成。要引用成功，当然要维护好 bibitem 了。下面是个简单的例子。

[1] 脚注可以补充说明一些东西







参考文献

[1] 文李明. 双阀电控单体泵燃油系统喷射特性研究[D]. 哈尔滨工程大学,2012.

[2] 丕茂, 张幽彤, 谢立哲. 喷射参数对共轨系统高压油管压力波动幅度的影响[J]. 内燃机学报,2013,31(06):550-556.

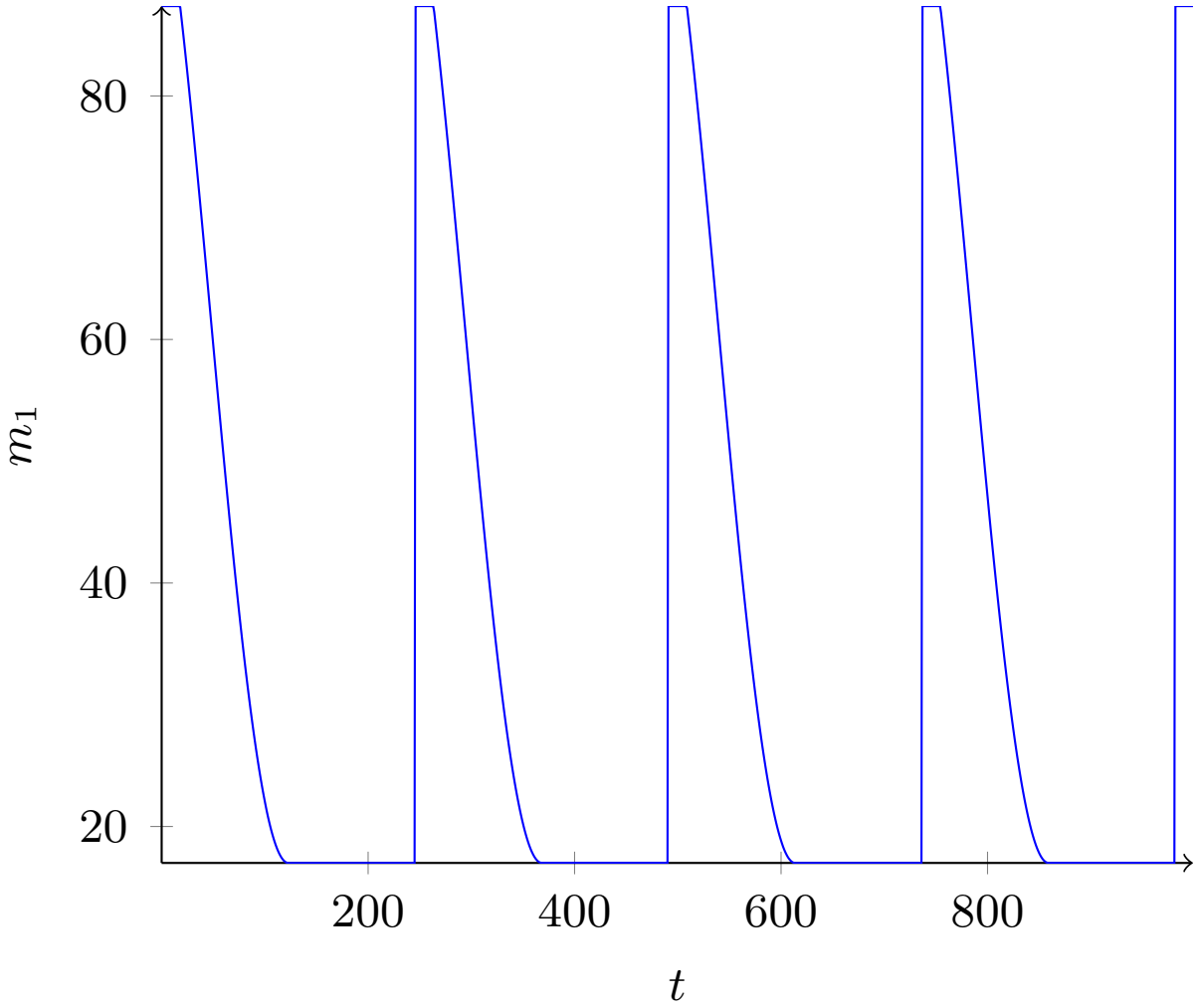


图 **6**　**m1-t** 图

附录 A　源代码

1.1　说明

这里应该注明一些东西：

- 语言版本（如 Python 3.7.4）
- 编译运行环境（越通用越好，比如 Python 最好在自己平时用的集成式开发环境中开发完成后放到自带的 IDLE 中看看能否运行）
- 各个源代码文件的输入、输出、调用关系等等，对于这种 1000 行代码左右的中小型项目来说，比较好的开发方式是一个模块文件 + 几个小题分别调用模块。以 Python 为例，比较好的做法是在 `lib.py` 中封装一个 `class`，然后每个小题 `import lib` 完成问题的求解和输入输出等等。
- 如果运行时间比较长要说明

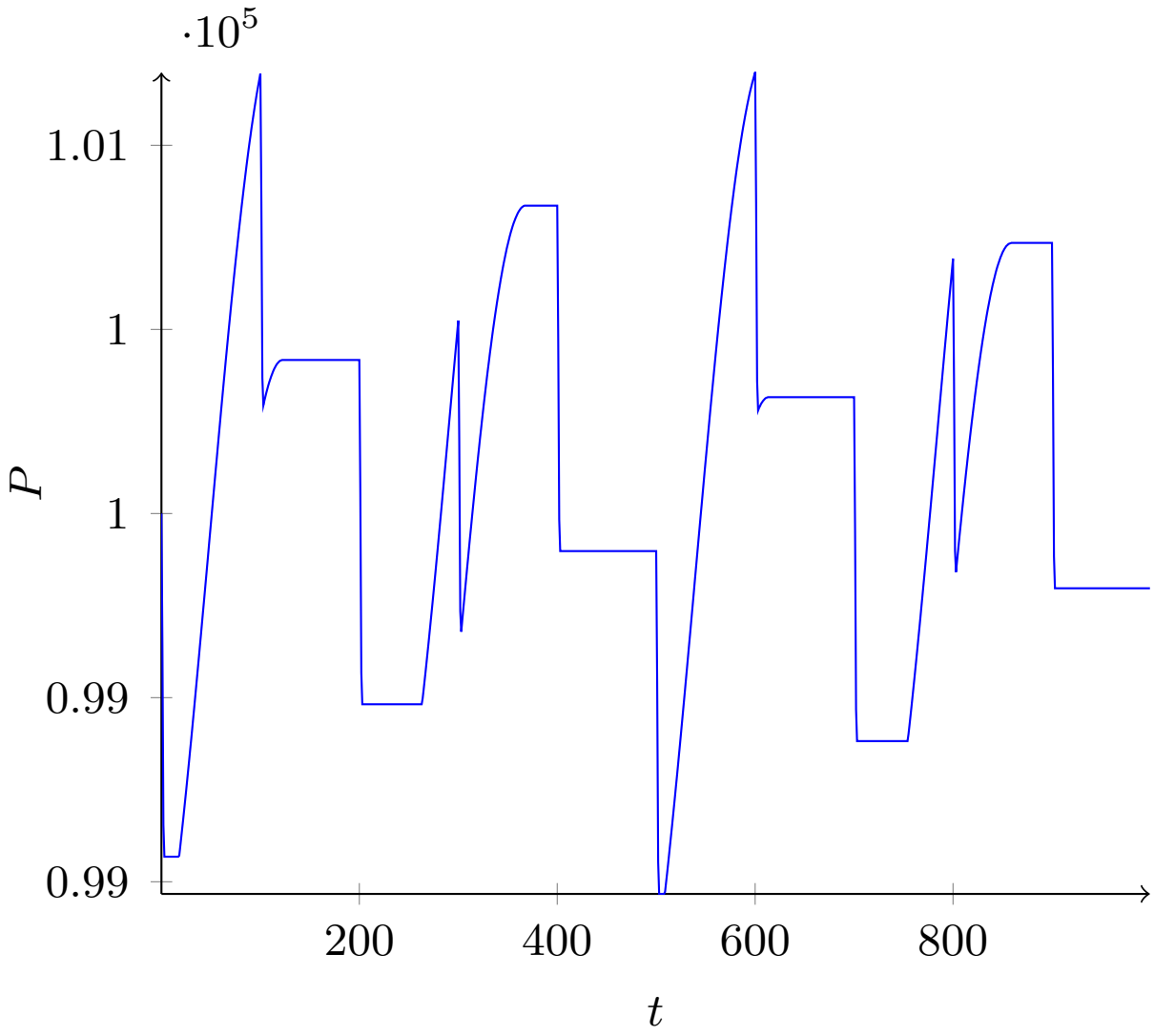


图 7　P-t 图

1.2　数据预处理

这里给出预处理数据的源代码。

Listing 1: *process_elasticity.py*

```
from lib import *
import math

infile = 'elasticity.raw'
outfile = 'density.dat'

f =  open(infile, encoding = 'utf-8', mode = 'r')
elasticity_data = [line.strip().split('\t') for line in f]
f.close()

P_array = [ float(line[0]) for line in elasticity_data]
```

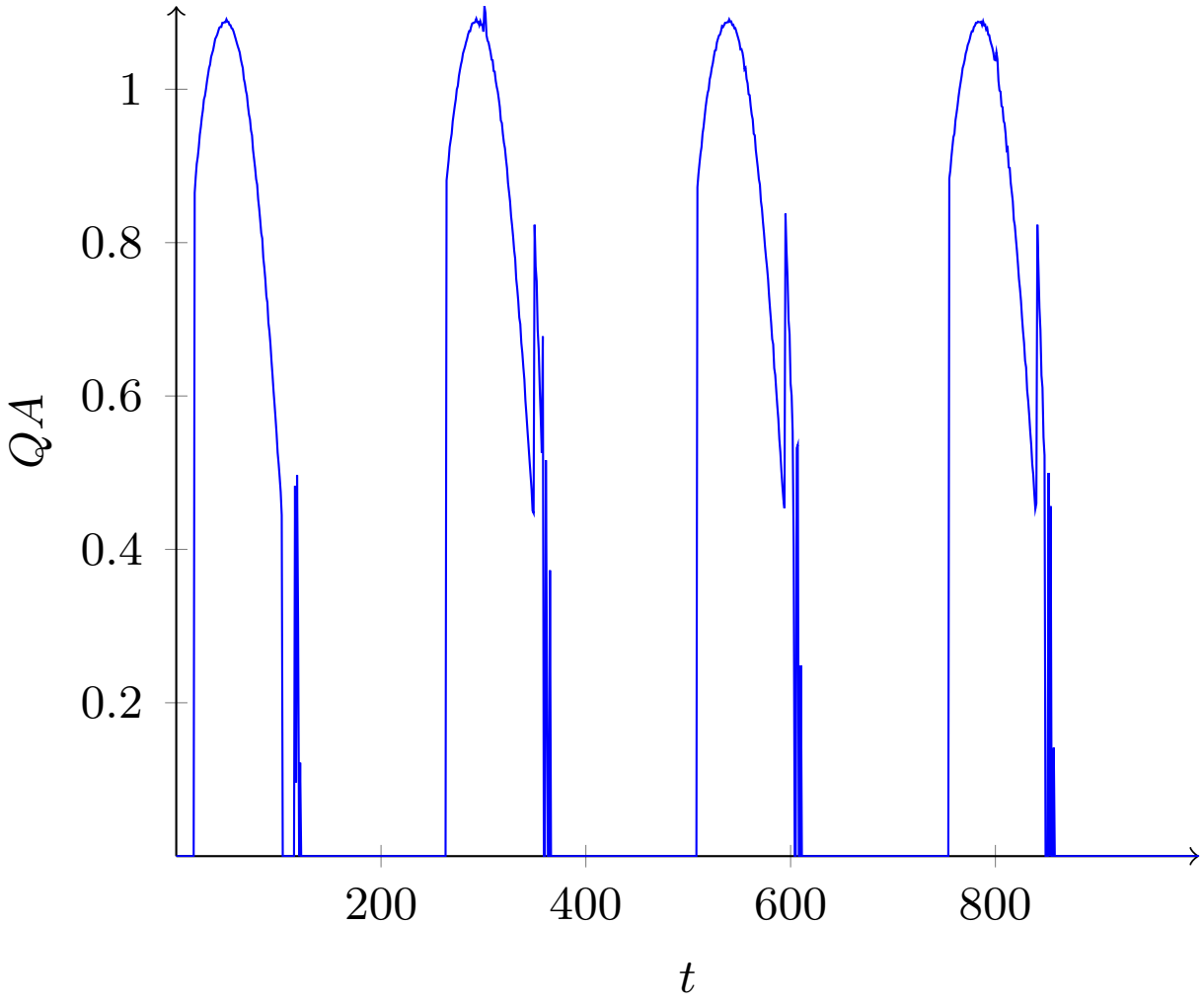


图 **8**　**QA-t** 图

```
E_array = [ float(line[1]) for line in elasticity_data]
rho_array = [0 for i in  range( len(P_array))]
rho_array[200] = rho_std

# 计算积分
for i in  range(200, 400):
    old_rho_log = math.log(rho_array[i])
    integral = (1/E_array[i] + 1/E_array[i+1])/2
    new_rho_log = old_rho_log + integral
    rho_array[i+1] = math.exp(new_rho_log)

for i in  range(200, 0, -1):
    old_rho_log = math.log(rho_array[i])
    integral = (1/E_array[i] + 1/E_array[i-1])/2
    new_rho_log = old_rho_log - integral
    rho_array[i-1] = math.exp(new_rho_log)

f =  open(outfile, encoding = 'utf-8', mode = 'w')
```

```

for i in  range(401):
    f.write( str(rho_array[i]) + '\t' +  str(P_array[i]*1000) + '\n')
f.close()
```

Listing 2: *process_outline.py*

```
from lib import *
import math

infile = 'outline.raw'
outfile = 'height.dat'

f =  open(infile, encoding = 'utf-8', mode = 'r')
outline_data = [line.strip().split('\t') for line in f]
f.close()

theta_array = [ float(line[0]) for line in outline_data]
r_array = [ float(line[1]) for line in outline_data]
alpha_array = theta_array
height_array = [ max(r * math.sin(alpha + theta) for theta, r in
    zip(theta_array, r_array)) for alpha in alpha_array]

f =  open(outfile, encoding = 'utf-8', mode = 'w')
for alpha, height in  zip(alpha_array, height_array):
    f.write( str(alpha) + '\t' +  str(height) + '\n')
f.close()
```

接下来给出本论文定义的函数。

Listing 3: lib.py

```
import math

# 全局常数
pi = math.pi
P_0 = 101.325
P_1 = 1.6e5
P_2_std = 1e5
rho_std = 0.850
rho_2_std = 0.850
C = 0.85 / math.sqrt(1000)
d_2 = 10
l_2 = 500
V_2 = l_2 * pi * d_2**2/4
d_A = 1.4
S_A = pi * d_A**2 / 4
V_B = 44
```

```
17 tau_0 = 100
18 tau_A_close = 10
19
20 def get_data_from(path_to_file):
21     f =  open(path_to_file, encoding = 'utf-8', mode = 'r')
22     l = [[ float(field) for field in line.strip('\n').split('\t')] for line in f]
23     f.close()
24     return l
25
26 def write_data_to(path_to_file, data):
27     f =  open(path_to_file, encoding = 'utf-8', mode = 'r')
28     for item in data:
29         item_s = [ str(field) for field in item]
30         f.write(''.join(item_s) + '\n')
31     f.close()
32
33 def Q(P_high, P_low, rho_high, A):
34     return C * A * math.sqrt(2 * (P_high - P_low) / rho_high)
35
36 def interpolation(x, x_array, y_array, equally_spaced = False):
37     """
38     目的：找到自变量 x 所在分段函数的区间
39     输入：
40     - x：待求值的自变量；
41     - x_array：已知数据点中自变量的升序系列
42     - y_array：分别对应于 x_array 中每一个点的因变量值
43     - length：0 表示 ref 中 x 不是等间隔系列，其他数值表示以 length 为等间隔；
44     输出：数组 (x_1, x_2, y_1, y_2)，其中 x_1 <= x < x_2
45     """
46
47     if x < x_array[0] or x >= x_array[-1]: raise Exception('过小或过大的自变量数值（在 get_interval
       中')
48
49     imin = 0
50     imax =  len(x_array) - 1
51     if equally_spaced:
52         space = x_array[1] - x_array[0]
53         i =  int((x - x_array[0]) / space)
54         return y_array[i] + (y_array[i+1] - y_array[i]) * (x - x_array[i]) / space
55     else:
56         while imax - imin > 1:
57             temp =  int((imax + imin)/2)
58             if x_array[temp] <= x:
59                 imin = temp
60             else:
61                 imax = temp
62         return y_array[imin] + (y_array[imax] - y_array[imin]) * (x - x_array[imin]) /
```

```
            (x_array[imax] - x_array[imin])

# 密度
density_data = get_data_from('../data/density.dat')
rho_array =  tuple(item[0] for item in density_data)
P_array =  tuple(item[1] for item in density_data)
rho_min = rho_array[0]
# 高度
height_data = get_data_from('../data/height.dat')
alpha_array =  tuple(item[0] for item in height_data)
height_array =  tuple(item[1] for item in height_data)
min_h =  min(height_array)
max_h =  max(height_array)
# 针阀高度
movement_data = get_data_from('../data/movement.raw')
t_array =  tuple(item[0] for item in movement_data)
z_array =  tuple(item[1] for item in movement_data)

def calc_rho(P):
    return interpolation(P, P_array, rho_array, equally_spaced = True)

def calc_pres(rho):
    if rho < rho_min:
        return 0
    return interpolation(rho, rho_array, P_array, equally_spaced = False)

def calc_height(alpha):
    return interpolation(alpha, alpha_array, height_array, equally_spaced = True)

def calc_z(t):
    return interpolation(t, t_array, z_array, equally_spaced = False)

def calc_V_1(height):
    return 20 + (max_h-height)*pi*25/4
def calc_V_h(height):
    return 20 + (max_h-height)*pi*25/4

def solve_Q_B(P_1, h):
    """
    目标：计算喷油器 B 的流速
    输入：
    - P_1：高压油管的压强（kPa），要求大于大气压 P_0
    - h：针阀升程（mm）
    输出：流量（mm^3/ms）
    """
    theta = math.pi / 20  # 9°的夹角
    sin_t = math.sin(theta)
```

```
109     S_B_prime = math.pi * (h * h * sin_t * sin_t * math.cos(theta) + 2.5 * h * sin_t)  #
            针阀和圆锥之间等效截面积
110     S_B = math.pi * 0.49  # 圆锥底部截面积
111     rho_1 = calc_rho(P_1)  # P_1压强下的煤油密度
112     # x,y>0 表示 delta_P_1 与 delta_p2, 分别表示在针阀和圆锥底部的压降, 满足 x+y=P_1-P_0
113     # p2 为圆锥中间压强, 满足 p2=P_1+delta_P_1
114     x = 0
115     # 实际待求方程组为:
116     # x+y=P_1-P_0
117     # return = S_B_prime*math.sqrt(2*x/rho_1) = S_B*math.sqrt(2*(y)/calc_rho(P_1-x))
118
119     x_1 = 0.25 * (P_1 - P_0)  # 利用x_1, x_2用割线法求解 f(x)=0, 这里f(x)是严格单调递增函数
120     x_2 = 0.75 * (P_1 - P_0)  # 这里x_1, x_2的初值其实是任意的
121     f_1 = S_B_prime * math.sqrt(2 * x_1 / rho_1) - S_B * math.sqrt(2 * (P_1 - P_0 - x_1) /
            calc_rho(P_1 - x_1))  # 目标为f=0
122     max_times = 1000  # 设置一个迭代次数上限
123     for j in  range(max_times):
124         if  abs(x_1 - x_2) < 0.00001:  # 设置允差
125             x = x_2
126             break
127         else:
128             f_2 = S_B_prime * math.sqrt(2 * x_2 / rho_1) - S_B * math.sqrt(2 * (P_1 - P_0 - x_2) /
                    calc_rho(P_1 - x_2))
129             x_1 = x_2 - (x_2 - x_1) * f_2 / (f_2 - f_1)
130             # 防止前几次迭代使得x越界
131             if x_1 > P_1 - P_0:
132                 x_1 = P_1 - P_0
133             elif x_1 < 0:
134                 x_1 = 0
135             # 交换下标1和2用于下次迭代
136             x_1, x_2 = x_2, x_1
137             f_1 = f_2
138     # 迭代超过上限就报错
139     if x == 0: raise Exception("Secant method doesn't converge! (In yuanzhui())")
140     return C * S_B_prime * math.sqrt(2 * x / rho_1)
141
142 # 测试函数
143
144 def test_interpolation():
145     x_array = (1, 2, 3, 4)
146     y_array = (1, 4, 9, 16)
147     x = 2.5
148     print(interpolation(x, x_array, y_array, True))
149
150 # test_interpolation()
```

1.3 第一题源代码

Listing 4: simulation.py

```python
import sys
sys.path.append('..')
from data.lib import *

# 数值模拟控制参数
dt = 0.01 # 步长
# period_number_list = {10: [0.290 + i * 0.001 for i in range(10)],
#     20: [0.285 + i * 0.001 for i in range(10)],
#     50: [0.285 + i * 0.001 for i in range(10)],
#     100: [0.283 + i * 0.001 for i in range(5)],
#     200: [0.283 + i * 0.001 for i in range(5)],
#     500: [0.282 + i * 0.001 for i in range(4)],
#     1000: [0.282 + i * 0.001 for i in range(3)]
#     }
# period_number_list = [20 * i for i in range(1, 6)] # 周期数列表
period_number_list = [1] # 周期数
period_steps =  int(tau_0 / dt) # 每个周期中的演化步数

# 系统参数
rho_1 = calc_rho(P_1) # 油泵中油的压力
rho_2 = rho_std # 油管中油的压力
tau_A_close = 10 # A 的关闭时间
# tau_A_open_list = [0.280 + i * 0.001 for i in range(30)] # A 的开启时间
tau_A_open_list = [0.284 + i * 0.001 for i in  range(1)] # A 的开启时间
t_0 = 0 # B 的初始相位时间

def calc_Q_A(t, P_2, tau_A_open):
    """
    输入：时间 t，油管压力 P_2
    输出：该时间单向阀 A 的流量
    """
    tau_A_period = tau_A_close + tau_A_open
    phase = t % tau_A_period
    if phase < tau_A_open:
        return Q(P_1, P_2, rho_1, S_A)
    else:
        return 0

def calc_Q_B(t):
    """
    输入：时间 t
    输出：该时间喷油嘴 B 的流量
    """
```

```
    phase = (t + t_0) % 100
    if phase < 2.4:
        if phase < 0.2:
            return 100 * (phase + dt / 2)
        elif phase < 2.2:
            return 20
        else:
            return 20 - 100 * (phase - 2.2 + dt / 2)
    else:
        return 0

f =  open('P-t.dat', encoding = 'utf-8', mode = 'w')
f.write('%s\t%s\n' % ('t', 'P'))
# f = open('tau-N.dat', encoding = 'utf-8', mode = 'w')
# f.write('%s\t%s\n' % ('N', 'tau'))
# for period_number, tau_A_open_list in period_number_list.items():
for period_number in period_number_list:
    count = 0
    total_steps = period_number * period_steps # 总的演化步数
    D_min = 1e8
    best_tau_A_open = 0
    # f = open('D-tauA%d.dat' % period_number, encoding = 'utf-8', mode = 'w')
    for tau_A_open in tau_A_open_list:
        D = 0
        P_2 = P_2_std
        m_2 = calc_rho(P_2) * V_2
        for i in  range(total_steps):
            t = dt * i
            # 演化一步
            # 进油
            Q_A = calc_Q_A(t, P_2, tau_A_open)
            delta_m = Q_A * rho_1 * dt
            m_2 += delta_m
            # 出油
            Q_B = calc_Q_B(t)
            delta_m = Q_B * rho_2 * dt
            m_2 -= delta_m
            # 更新压强
            rho_2 = m_2 / V_2
            P_2 = calc_pres(rho_2)
            # 计算平方误差
            D += (P_2 - P_2_std)**2
            # 该行代码输出 P-t 图
            if i % 10 == 0: f.write('%f\t%d\n' % (t,  int(P_2) - 100000))
        D = math.sqrt(D / total_steps) # 均方误差
        if D < D_min:
            D_min = D
```

```
            best_tau_A_open = tau_A_open
        count += 1
        print('完成了 %d 个时长的计算，一共 %d 个时长' % (count,  len(tau_A_open_list)))
    # f.write('%d\t%f\n' % (period_number, best_tau_A_open))
f.close()
```

1.4 第二题源代码

1.5 第三题源代码