高压油管的压力控制

摘要

本文中我们先给出了每一题的基本分析和思路，再给出了它们的具体解题过程。

对于问题一，我们假定燃油在油管内压强均匀，高压泵的流量符合题中注 2 所给公式。我们预先根据题目中给的弹性模量和压强的关系，用梯形法数值积分得到了在离散点上燃油密度和压强的对应关系，同时在后面程序使用时通过线性内插得到完整函数关系。我们先通过稳压近似方法，利用长时间内燃油支出平衡给出单向阀开启时长的理论值；再通过将时间离散化，用数值模拟的方法给出压强精确演化规律，并对两种方法的结果进行了交叉验证。

对于问题二，我们建立了一个考虑了油泵和喷油器工作原理的油管演化模型。我们预先利用凸轮的外轮廓形状计算了在给定转角下柱塞的抬升高度。再将燃油在喷油器内的行为近似为了两次扩散过程，并利用已知的针阀升程的变化，通过割线法求解了给定时间和油管压强下的喷油器流量。最后我们将油泵、油管和喷油器三个部件组合到一起，完成系统的模拟演化程序。我们定义关于目标压强的方均根偏差作为压强偏离的量度，得出了最优凸轮角速度为 $\omega = 0.0238\ \mathrm{ms}^{-1}$。

对于问题三，首先我们从上一问找出了压强波动的原因。联系实际情况给出了系统的可变量。我们先对减压阀可连续调节的情况，通过定性半定量分析得出最优的控制方案应满足的条件，再通过数值模拟和调参得到具体的控制方案，该方案能够将压强波动缩小接近 100 倍。然后又考虑了实际应用中可能对阀门控制有所限制，构造了另一种阀门操作频率很低的控制方案，能将压强波动缩小约 4 倍，并从理论上证明了在给定限制条件下该方案的最优性。

在本文的最后，我们对上述建立的模型进行了评价，说明了它的合理性和可拓展性，同时指出了一些近似导致的模型的局限性。

关键字： 高压油管　数值模拟　稳压控制

一、 问题的表述与分析

1.1 引言

高压油管被广泛应用于柴油机等燃油发动机中 [1]，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会导致高压油管内压力的变化，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，管内压强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从而提升工作效率，是当前高压燃油系统亟需解决的技术难题。

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。

1.2 问题 1

1.2.1 问题 1 的表述

已知供油系统 1（以下简称系统 1，下同）的示意图为图 1，由以下三部分组成：

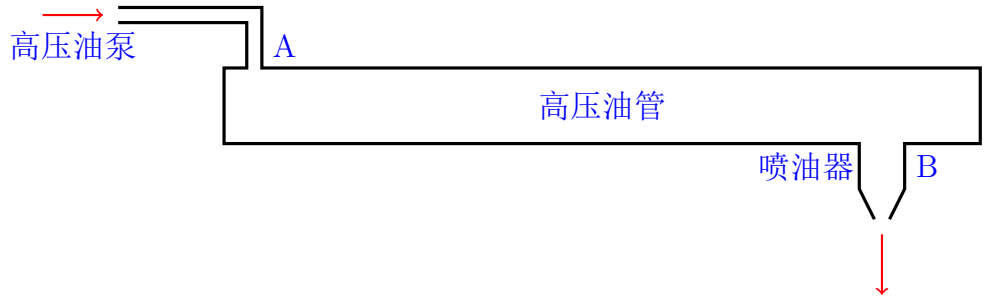

图 1　系统 1 示意图

- 供油器 A：通过单向阀与高压油管（以下简称油管）连接，截面内径为 1.4 mm；单向阀打开时向油管内供油，压强恒为 1.6×10^5 kPa，且每打开一次后就要关闭 10 ms；
- 高压油管：内腔长度为 500 mm，内直径为 10 mm，初始压强为 1×10^5 kPa；
- 喷油器 B：每秒工作 10 次，每次工作时喷油时间为 2.4 ms，工作时向外喷油的速率如图 2 所示。

现有如下问题：

- 单向阀开启时长（以下简称开启时长）为多少时，系统演化过程中油管内的压强关于目标压强 1×10^5 kPa 有最小的偏离？

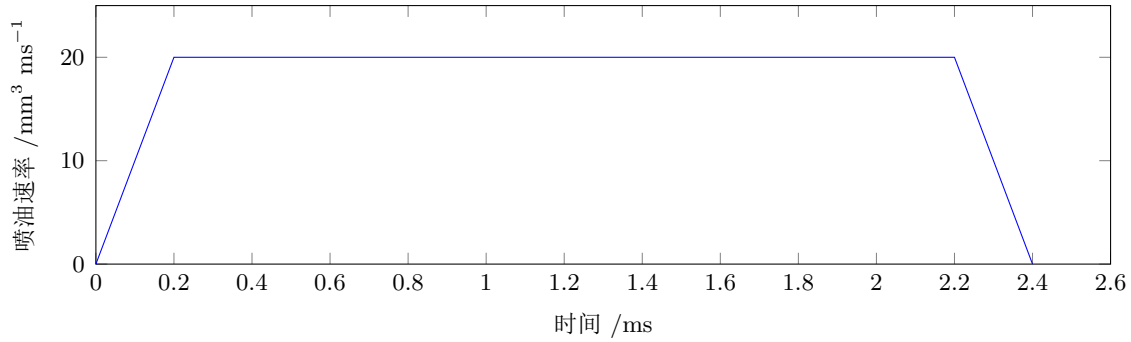

图 **2**　喷油速率示意图

- 开启时长为多少时，油管压强将在约 2000 ms、5000 ms 和 10000 ms 的调整过程后达到 1.5×10^5 kPa？
- 达到 1.5×10^5 kPa 之后，开启时长为多少时，系统演化过程中油管内的压强关于目标压强 1×10^5 kPa 有最小的偏离？

1.2.2 问题 1 的分析

问题 1 的第 1 小问要求我们通过设置开启时长使得油管压强稳定在 1×10^5 kPa。由喷油速率示意图，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，由此我们得出两种解题思路：

稳压近似方法：即近似认为油管压强稳定在 1×10^5 kPa 时，其压强波动可以忽略不计。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。若要求压强稳定，必然有流入质量和流出质量相等，据此可以解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。

数值模拟方法：即不作近似，而是对系统的演化进行精确的数值模拟。在数值模拟中，我们采取较小的时间步长（如 0.01 ms），在每一步计算单向阀流入的流量和喷油嘴流出的流量，并根据出入流量计算油管内燃油的质量变化，进而得到更新后的油管内燃油密度和压强，并计算压强关于 1×10^5 kPa 的偏离大小。

通过数值模拟方法，我们可以验证我们通过稳压近似方法选取的开启时长是否是在精确模拟中使得油管压强偏离最小的开启时长。

对于问题 1 的第 2 小问，对于每一个给定的调整过程长度，我们可以通过数值模拟方法尝试一系列可能的开启时长，对于每一个开启时长计算调整过程过后系统关于目标压强的偏离，从而选取最优的开启时长。

对于问题 1 的第 3 小问，我们可以通过类似于第 1 小问的方法计算出对应于保持 1.5×10^5 kPa 的开启时间。

1.3 问题 2

1.3.1 问题 2 的表述

已知系统 2 的示意图为图 3，由以下三部分组成：

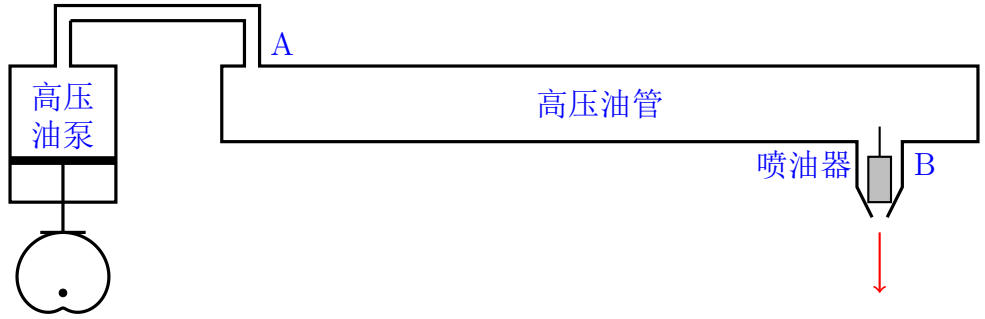

图 **3**　系统 **2** 示意图

- 供油器 A：通过单向阀与油管连接，截面内径为 1.4 mm；燃油来自高压油泵的柱塞腔出口，当柱塞腔内的压强大于高压油管内的压强时，柱塞腔与高压油管连接的单向阀开启，燃油进入高压油管内。柱塞上下运动由凸轮驱动，柱塞腔内径为 5 mm，运动到上止点位置时，柱塞腔残余容积为 20 mm^3；柱塞运动到下止点时，压强为 500 kPa 的低压燃油会充满柱塞腔；
- 高压油管：内腔长度为 500 mm，内直径为 10 mm，初始压强为 1×10^5 kPa；
- 喷油器 B：喷油器喷油器结构如图 4 所示，每秒工作 10 次，喷油量由针阀控制。针阀直径为 2.5 mm、密封座是半角为 $\pi/20$ 的圆锥，最下端喷孔的直径为 1.4 mm。针阀升程为 0 时，针阀关闭；针阀升程大于 0 时，针阀开启，燃油向喷孔流动，通过喷孔喷出。

现有如下问题：

- 凸轮的角速度为多少时，系统演化过程中油管内的压强关于目标压强 1×10^5 kPa 有最小的偏离？

1.3.2 问题 2 的分析

问题 2 相较于问题 1，需要考虑额外的两个问题：

1. 由于系统 2 中高压油泵柱塞由凸轮驱动，我们首先需要将凸轮的边缘曲线转化为凸轮转过不同角度时柱塞的高度；
2. 由于系统 2 中未给出喷油器 B 的流量，需要我们根据压强以及 B 的几何形状求解相应的流量。

在此基础上，我们可以使用和问题 1 中相同的算法，对油泵和油管分别动态地更新压强和密度，实现对系统 2 的数值模拟。通过对不同的角速度下系统的演化进行模拟计

算，我们可以得出使得油管对目标压强 1×10^5 kPa 偏离最小的角速度。

1.4 问题 3

1.4.1 问题 3 的表述

已知系统 3 的示意图为图 3，由以下三部分组成：

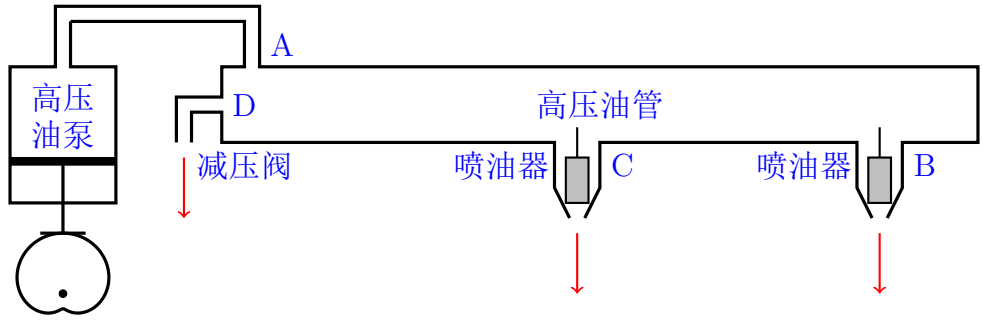

图 4　系统 3 示意图

- 供油器 A：与系统 2 中相同，并假设凸轮仍为匀速转动；
- 高压油管：与系统 1、2 中相同；
- 喷油器 B、C：结构与系统 2 中相同，考虑到实际应用 [2]，B 和 C 能够调节的只有二者喷油的时间间隔，而喷油频率保持不变；
- 单向减压阀 D（以下简称减压阀）：打开减压阀时高压油管内的燃油可以回流到外部低压油路中，使得高压油管内的压强减小 [3]。由于题目并没有明确说明减压阀是否可以连续控制，在本题中我们将按以下两种情况进行分类讨论：

  1. 减压阀可以随时开启和关闭，不受任何限制；
  2. 减压阀每次开启后需要至少 10 ms 的关闭时间，与问题 1 中类似。

  现有如下问题：

- 当减压阀不受限制时，凸轮的角速度为多少、B 和 C 的喷油时间间隔为多少、D 采取何种控制方案时，系统演化过程中油管内的压强关于目标压强 1×10^5 kPa 有最小的偏离？
- 当减压阀受到限制时，凸轮的角速度为多少、B 和 C 的喷油时间间隔为多少、D 采取何种控制方案时，系统演化过程中油管内的压强关于目标压强 1×10^5 kPa 有最小的偏离？

1.4.2 问题 3 的分析

理解此问题的关键在于找到压强波动的原因。由于具体分析需要用到问题 2 的结果，我们将在后文根据问题 2 的结果进行详细分析，在此不再赘述。

二、 模型的建立与求解

2.1 符号说明

现将本文使用的主要符号列于下表：

表 **1** 符号说明

符号	物理量
P_0	大气压
V_1	油泵体积
m_1	油泵中燃油质量
ρ_1	油泵中燃油密度
P_1	油泵中燃油压强
V_2	油管体积
m_2	油管中燃油质量
ρ_2	油管中燃油密度
P_2	油管中燃油压强
P_2^*	油管中燃油的目标压强
ρ_3	喷油器缓冲区中燃油密度
P_3	喷油器缓冲区中燃油压强
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

表 **2** 符号说明（续）

符号	物理量
d_A	单向阀 A 截面直径
S_A	单向阀 A 截面面积
Q_A	单向阀 A 截面流量
τ_A	单向阀 A 周期
τ_A^+	单向阀 A 每次的开启时间
τ_A^-	单向阀 A 每次的关闭时间
d_B	喷油器 B 截面直径
S_B	喷油器 B 截面面积
S_B'	喷油器 B 针阀处等效截面面积
Q_B	喷油器 B 截面流量
ϕ_B	喷油器 B 相位
τ_B	喷油器 B 周期
τ_B'	喷油器 B 和 C 的等效周期
$\tau_B^{(A)}$	喷油器 B 周期内 A 的开启时间
τ_I	调整过程时长
Δt	模拟步长
D	油管中燃油压强的均方根偏差
$D_{\max}$	油管中燃油压强的最大偏差
E	油管中燃油压强的极差

2.2 问题一的求解

2.2.1 数据预处理

为考察系统在不同压强下的行为，首先要建立压强和燃油密度的函数关系。但题目中只给出了压强与弹性模量的关系，因此我们通过数据预处理的步骤将它转化为所需要的数据形式。

由题意，燃油的压强变化量与密度变化量成正比，比例系数为 E/ρ，且已知标准态 $P^* = 1 \times 10^5$ kPa 对应着 $\rho^* = 0.850$ mg mm^{-3}。据此我们可以列出如下的微分方程：

$$\frac{\mathrm{d}P}{\mathrm{d}\rho} = \frac{E}{\rho}$$

由于题目中将 E 表示为 P 的函数，我们进行移项并积分，得到：

$$\int_{P^*}^{P} \frac{\mathrm{d}P'}{E(P')} = \int_{\rho^*}^{\rho} \frac{\mathrm{d}\rho'}{\rho'} = \ln\rho - \ln\rho^*$$

但对于函数 $E(P)$ 我们只知道有限个点的函数值，因此左方的积分必须通过数值积分的方法计算，为此我们首先选取上限 $P = 1.005 \times 10^5$ kPa，将左方的积分近似表示为 $(E(P^*)^{-1} + E(P)^{-1})(P - P^*)/2$，而这两点处的函数值均已知，进而右方的 ρ 可以从上式中解出。依此类推，每次积分 500 kPa 的区间，逐次迭代得到若干个 ρ 的函数值直至 2×10^5 kPa，再向反方向逐次迭代直至 0，即得到所需的函数关系。

2.2.2 基于平均化近似的开启时间计算

首先假设油管的压强近似保持在 $P_2 = P_2^* = 1 \times 10^5$ 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 τ_B 内，从喷油嘴流出的燃油质量 Δm 由流量与密度给出：

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

这些质量应当由从单向阀进入的燃油补充，因此在一个 τ_B 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：

$$\begin{aligned}
\tau_B^{(A)} &= \frac{\Delta m}{Q_A \rho_1} \\
&= \frac{\Delta m}{\rho_1 C S_A \sqrt{2(P_1 - P_2)/\rho_1}} \\
&= 2.80 \text{ ms}
\end{aligned}$$

若我们考虑系统进行较长时间的演化，则 $\tau_B^{(A)}$ 占 τ_B 的比例，应该与它的开启时长 τ_A^+ 占自身周期 τ_A 的比例相同。因此，τ_A^+ 与 τ_A^- 应该满足如下比例式：

$$\frac{\tau_A^+}{\tau_A} = \frac{\tau_B^{(A)}}{\tau_B}$$

代入已知数据，可以求出 $\tau_A = 0.288$ ms。

2.2.3 系统 1 的数值模拟

在数值模拟中，我们设定初始条件为 $P_2 = 1 \times 10^5$ kPa，且时间零点恰好位于单向阀一个工作周期的开始，但可以位于喷油器一个工作周期中的任意一点，该点处相位为 ϕ_B。

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：

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

例如，当取 $\tau_A = 0.284$ ms 时，系统在第一个周期内的压强变化如图5 所示：

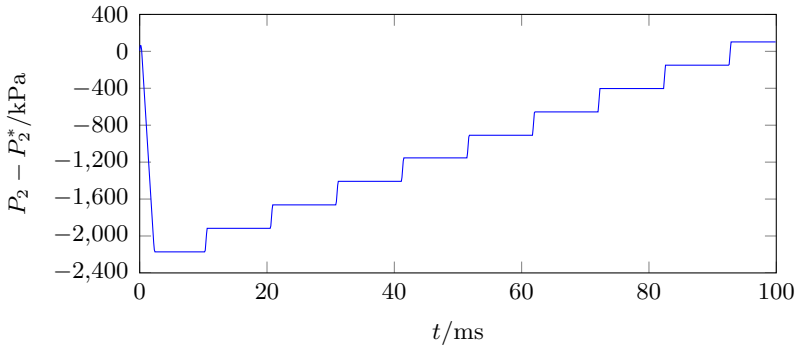

图 5　$\tau_A = 0.288$ ms 时，油管在第一个周期内的压强随时间的变化关系

为了定量刻画油管压强关于目标压强的平均偏离，我们将系统进行 N 个周期的演化，获得各个时刻油管的压强后，定义关于目标压强的均方根偏差 D 作为压强-时间函数的泛函：

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

由于油泵的供油量随开启时长单调递增，对于一个给定的目标压强，油管压强关于目标压强的均方根偏差 D 在开启时长不断增大时一定是先减小后增大的。因此，给定

一系列开启时长 τ_A^+，计算它们对应的均方根偏差 D，就可以从曲线的最低点处找出使得均方根偏差最小的开启时长。图 6 给出了模拟不同数量周期时，D 与 τ_A^+ 的关系：

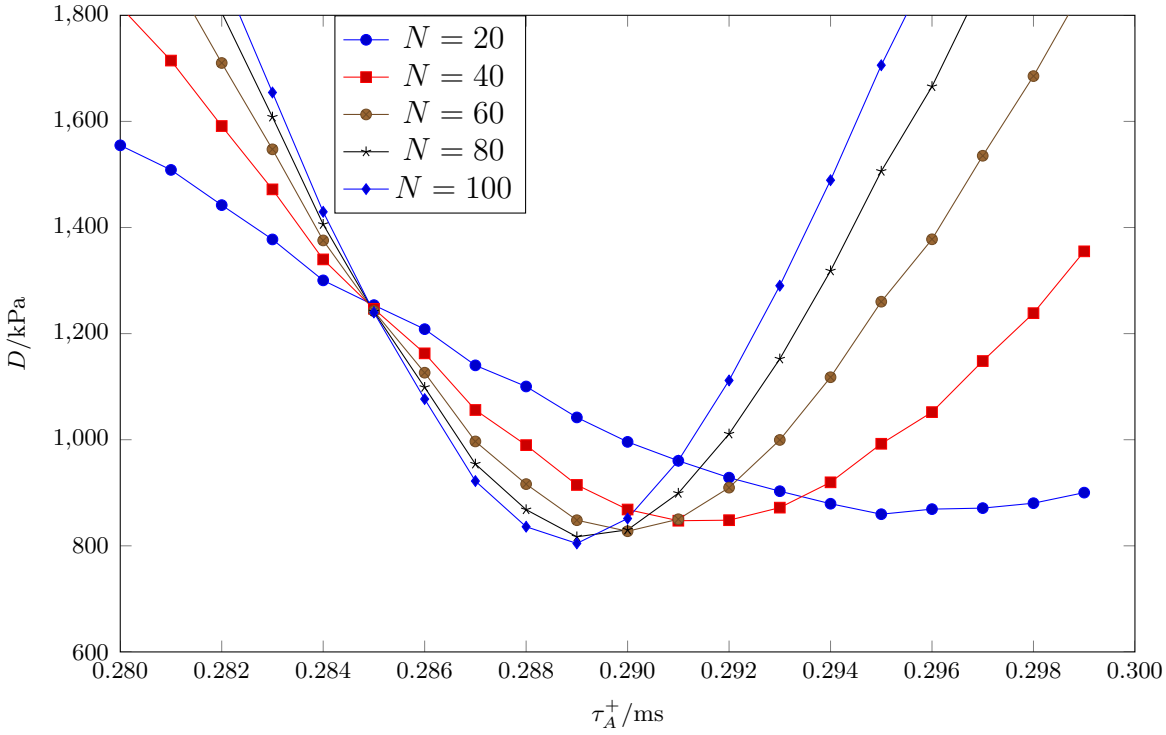

图 **6**　模拟不同周期数时，均方偏差 D 与开启时长 τ_A^+ 的关系

在图 6 中，为什么模拟不同周期数时，最优的开启时长不同呢？这是因为，图 5 中的压强-时间关系表明，油管在大多数时间中对目标压强产生了负偏离，因此周期数少时，开启时长较长（因而进入的油量较多）的方案具有优势；但周期数多时，只有入油量和出油量相等的方案才能在长期保持不偏离 1×10^5 kPa，因此最优的开启时长较短。我们使用几个具有代表性的周期数得到的最优时长如图 7 所示：

从这里我们可以得到两点结论：

- 当 $N \to \infty$ 时，数值模拟得到的最优开启时长就是通过稳压近似得到的最优开启时长，这表明稳压近似能够较好地描述这一系统；
- 当 $N > 100$ 时，这一结果接近于收敛，说明模拟 100 个以上的周期已经足以消除系统的任何暂态效应带来的影响。

2.3 第二小问

我们可以通过与第一小问中相同的方法，来近似计算令油管压强稳定在 1.5×10^5 kPa 所需要的开启时长，结果为 $\tau_A^+ = 0.752$ ms。根据第一小问中的结论，这一时间就是在当模拟时间趋于 $+\infty$ 时，使油管压强关于目标压强的均方根偏差最小的开启时长。

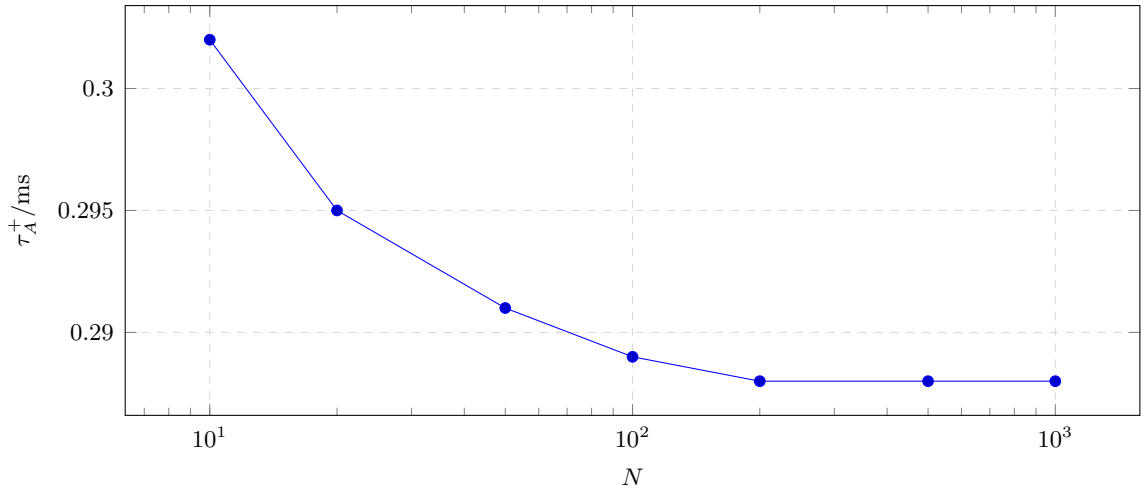

图 **7**　最优开启时长随模拟周期的收敛性

接下来我们考虑分别在 $2000, 5000, 10000$ ms 的调整过程中应该令 τ_A^+ 为多大。由于这一过程中油管压强发生了显著的变化，不能近似认为压强不变，因此难于解析求解。我们仍采用数值模拟的方法，通过扫描一个大致含有最佳 τ_A^+ 的时间范围来确定最佳的 τ_A^+。模拟分为两步：

1. 调整过程中，将 τ_A^+ 设定为我们试探的值，模拟至调整过程结束；
2. 调整过程结束后，令 $\tau_A^+ = 0.752$ ms，继续模拟 $N = 100$ 周期，计算油管压强关于目标压强的均方根偏差。

计算结果如图 8 所示：

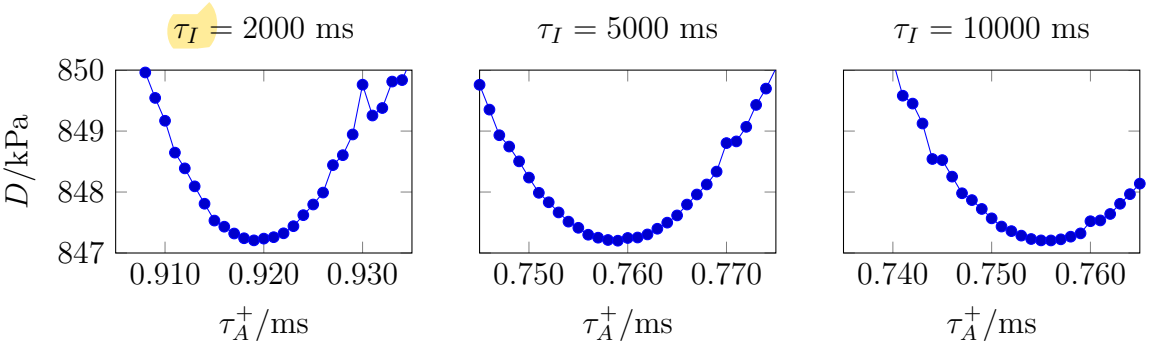

图 **8**　不同弛豫时间下压强的均方根偏离与开启时长的关系

由此可见，调整过程分别为 $2000, 5000, 10000$ ms 时，最佳的 τ_A 分别为 0.919、0.759 和 0.756。

2.4 问题二

2.4.1 数据预处理

由题意，油泵通过柱塞实现泵油，柱塞由凸轮驱动上下运动，且已知凸轮边缘曲线与角度的关系，因此首先要将凸轮边缘曲线转化为柱塞的高度。

当凸轮极轴的转角为 α 时，凸轮上极角 θ 的方向对应的方位角为 $\alpha+\theta$。因此，我们可以选取一系列转角的数据点 α_i，在每个数据点处遍历所有的极角 θ_j，计算对应极径的实际高度并取最大值，作为柱塞的高度：

$$h(\alpha_i) = \max_j (r_j \times \sin(\alpha_i + \theta_j))$$

再对上述数据点处的高度 h 进行线性内插即得到函数 $h(\alpha)$。

2.4.2 锥形喷油器流速的求解 [4]

题目中未给出喷油器的流速，因此需要利用密封座的几何参数进行求解。根据已知信息可以大致画出喷油器（为清楚展示，圆锥半角有所夸大），其中 d_I 为针阀半径，P_3, ρ_3 分别为缓冲区的压强和密度：

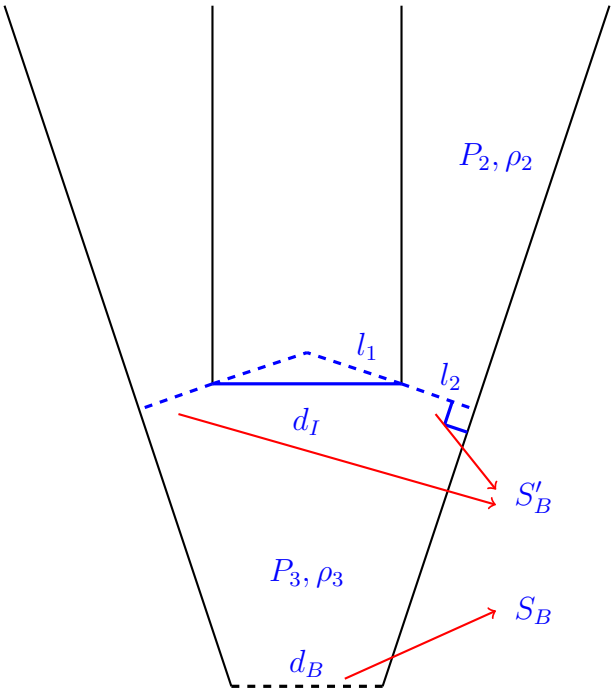

图 9　喷油器示意图

由此看出，在喷油的过程中，针阀以下的部分形成了一个较大的空间，其体积至少为

$$\frac{\pi}{24}\cot\gamma \times \left(d_I^3 - d_B^3\right) \approx 10\ \mathrm{mm}^3$$

而该部分体积与上下均有隔离，因此喷油过程可以用两次扩散过程进行近似。到达稳态时，两次扩散的流速相同，由此对喷油器可以列出以下方程：

$$Q = S'_B\sqrt{\frac{2(P_2 - P_3)}{\rho_2}} = S_B\sqrt{\frac{2(P_3 - P_0)}{\rho_3}}$$

其中等效面积 S'_B 的物理意义是从油管扩散到缓冲区时，所经过的最小截面面积，即针阀边缘到锥面的环形垂面的面积。计算公式为：

$$\frac{1}{2}\pi(d_I + 2l_2\cos\gamma)(l_1 + l_2) - \frac{1}{2}\pi d_I l_1 = \pi^2(z^2\sin\gamma^2\cos\gamma + d_I z\sin\gamma)$$

通过该方程可以求出给定压强 P_2 和给定针阀升程 z 下的流量 Q_B。

2.4.3 系统 2 的数值模拟

在数值模拟中，我们设定初始条件为 $P_2 = 1 \times 10^5$ kPa，且时间零点恰好位于凸轮一个工作周期的开始（即高度下降到最低点），但可以位于喷油器一个工作周期中的任意一点，该点处相位为 ϕ_B。

系统的演化可以用如下迭代关系表示：

$$\begin{aligned}
\alpha &\leftarrow \alpha + \omega \times \mathrm{d}t \\
h &\leftarrow h(\alpha) \\
Q_A &\leftarrow Q_A(t) \\
m_1 &\leftarrow m_1 - Q_A \times \rho_1 \times \mathrm{d}t \\
m_2 &\leftarrow m_2 + Q_A \times \rho_1 \times \mathrm{d}t \\
Q_B &\leftarrow Q_B(t) \\
m_2 &\leftarrow m_2 - Q_B \times \rho_2 \times dt \\
\rho_1 &\leftarrow m_1/V_1 \\
\rho_2 &\leftarrow m_2/V_2 \\
P_1 &\leftarrow P(\rho_1) \\
P_2 &\leftarrow P(\rho_2)
\end{aligned}$$

我们仍以均方根误差 D 为判据，考察不同角速度下关于目标压强的均方根偏离 D。由于油泵的供油量随角速度单调递增，根据与系统 1 的数值模拟中相同的讨论，存在唯一的最优角速度使得 D 达到最小。图 10 给出了一定范围内 D 与 ω 的关系：

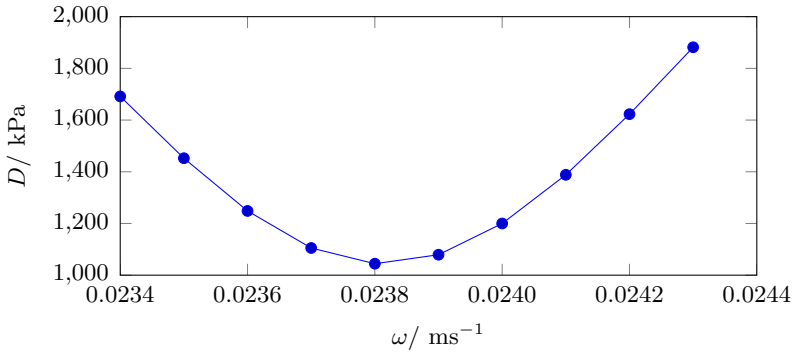

图 **10**　不同角速度下关于目标压强的均方根偏离

由此可见 $\omega = 0.0238\ \text{ms}^{-1}$ 为最优的角速度。在该角速度下，系统的各个物理量随时间变化如图 11 所示：

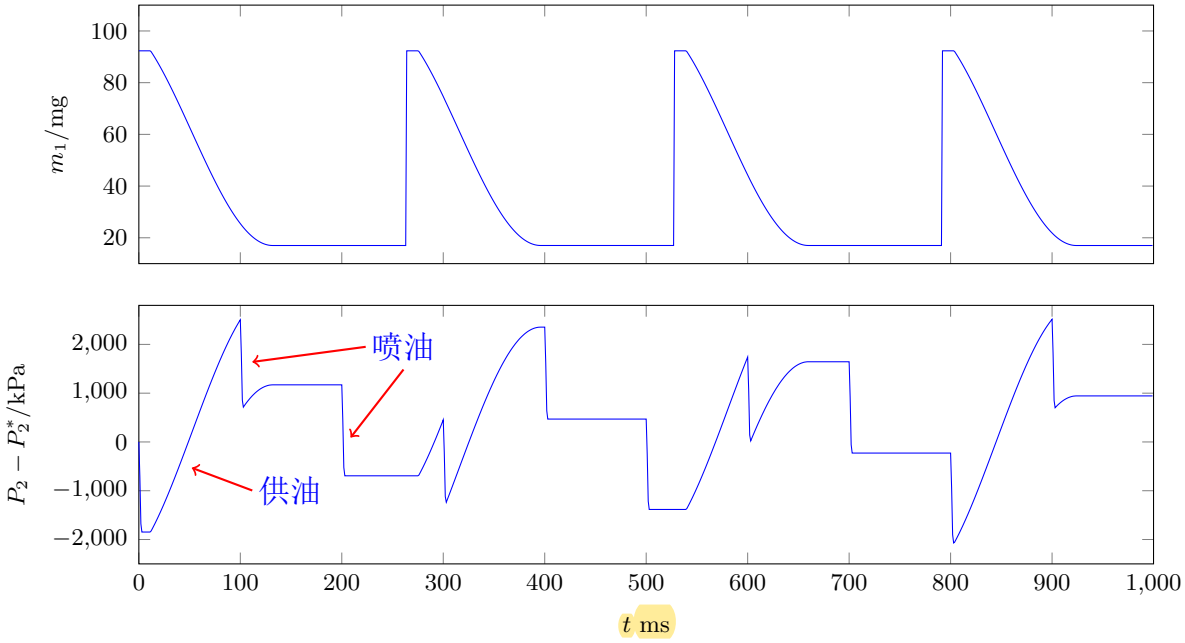

图 **11**　最优角速度 $\omega = 0.0238\ \text{ms}^{-1}$ 下 10 个周期内 m_1 和 P_2 随时间变化的关系

图 11 揭示了压强波动的原因。泵油时，m_1 下降，P_2 上升，其斜率较为平缓；而喷油时，P_2 下降，其斜率较为陡峭。泵油和喷油时间不同，斜率也不同，导致油管内的燃油在某一些时间段过度累积，在另一些时间段过度消耗。即使选取了最优的角速度 ω，也仅能将 D 降到 1044 kPa 左右。

2.5 问题 3

2.5.1 减压阀不受限制情况：定性分析

在问题 2 中我们看到，压强波动是由于喷油和泵油的时间不一致造成的。如果能在喷油的同时泵油，就能使得两者的效应相互抵消，压强波动变小。我们作如下规定：

- 单位时间油泵泵入的燃油质量随时间的变化关系称为「泵油曲线」；
- 单位时间通过 B 和 C 喷油器喷出的燃油质量随时间的变化关系称为「喷油曲线」；
- 单位时间油泵供给的燃油减去减压阀释放的燃油随时间的变化曲线称为「供油曲线」。

我们的目的在于使供油曲线和喷油曲线尽量重合，这样 m_2 在系统演化过程中能够尽量保持在初始值。在最理想的情况下，供油曲线和喷油曲线完全重合，此时 m_2 不变，因而压强也保持不变。

通过减压阀的连续控制，可以利用开启减压阀来抵消油泵的供油，从而将供油曲线向下调整。例如在喷油器没有喷油的情况下，油泵却在泵油，此时若管内压强大于 1×10^5 kPa 则开启减压阀，反之则关闭减压阀，可以保证压强准确维持在标准值附近（即供油曲线向下调整至 0）。当然，测量压强并实时反馈在应用中无法实现，这里仅通过该方法寻找控制策略；一旦找到策略，应用时只需按照预设时间开关减压阀即可，不再需要测量和反馈。

上述调节成立的条件是减压阀的泄流速度大于油泵的供油速度。ω 很大时泵油流量可能会非常大，以至于打开减压阀时油管内压强仍然上升，调节功能失效。因此 ω 有上限值。

为了长时间的精确调节，油泵的周期 $\tau_A = 2\pi/\omega$ 和喷油周期 τ_B 应当是简单整数比的关系。喷油的速率较大且比较恒定，为了使供油曲线贴近喷油曲线，τ_A 应该较小，合理的假设是 $\tau_B = N\tau_A$，N 为整数。在此基础上，为了使得 ω 较大，B 和 C 应该交替喷油，这等效于 $\tau_B' = 50$ ms 的喷油周期。

我们以 $N = 3$ 的情况为例，将上述分析直观展现在图 12 中。上图：泵油曲线，中图：供油曲线，下图：喷油曲线。减压阀能够将泵油曲线调节为供油曲线，使其贴近于喷油曲线。其中，为了使得流入燃油质量等于流出燃油质量，阴影部分应该面积相等。

2.5.2 减压阀不受限制情况：数值模拟

除了喷油周期改变为原来一半外，相较于问题 2 中的程序，此题中数值模拟程序的改变主要体现在 m_2 更新的部分。记 δm 为模拟步长 δt 内从减压阀流出的油量，则 m_2 更新的算法应为：

1. 若 $P_2 < P_2^*$，$m_2 \leftarrow m_2 + Q_A \times \rho_1 \times \mathrm{d}t - Q_B \times \rho_2 \times \mathrm{d}t$
2. 若 $P_2 > P_2^*$，$m_2 \leftarrow m_2 + Q_A \times \rho_1 \times dt - Q_B \times \rho_2 \times \mathrm{d}t - \mathrm{d}m$

供油曲线的阴影部分较喷油曲线的阴影部分更为分散，N 越大供油曲线的阴影越集中，也就更贴近喷油曲线。然而模拟发现 $N \geq 4$ 时减压阀喷油率 $Q_D\rho_2$ 有时小于泵油率 $Q_A\rho_1$，调节功能失效。因此可以肯定 $N = 3$ 为最优方案。

经过调节相位 ϕ_B，得到的最优方案在最开始的 $10 \times \tau_B'$ 内压强与时间的关系如图

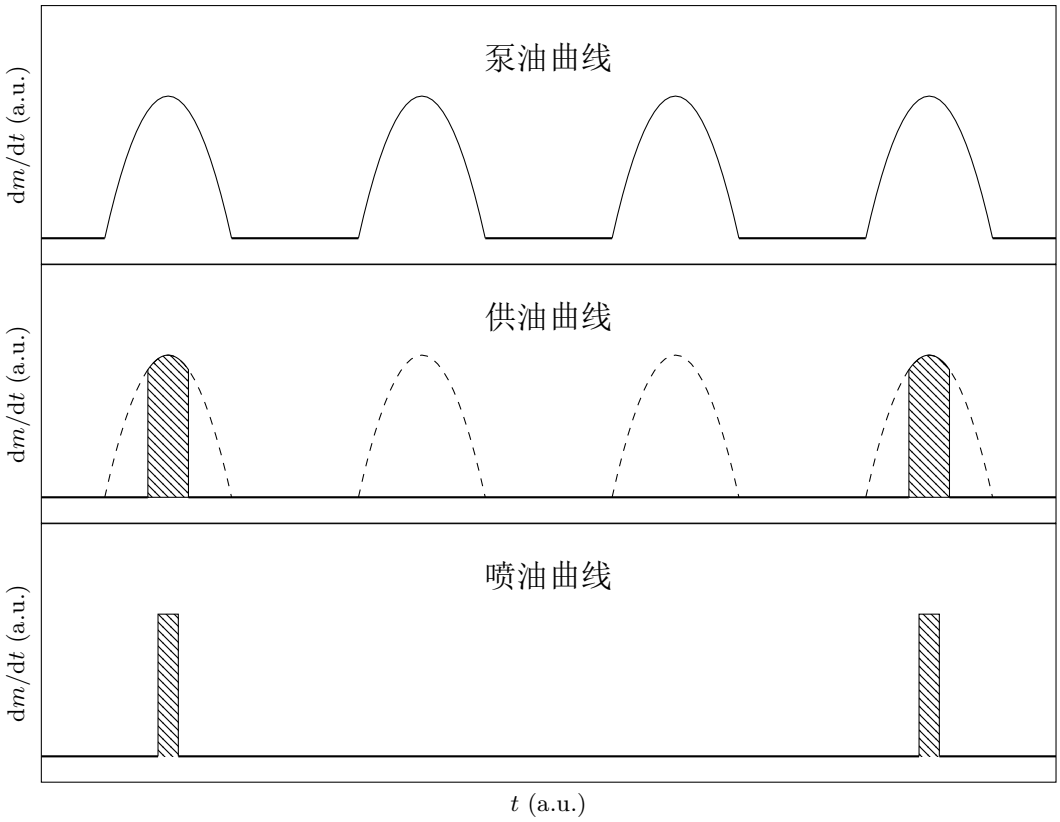

图 **12**　$\tau_A = \tau_B'/3$ 时，泵油、供油和喷油曲线的示意图

13 所示。

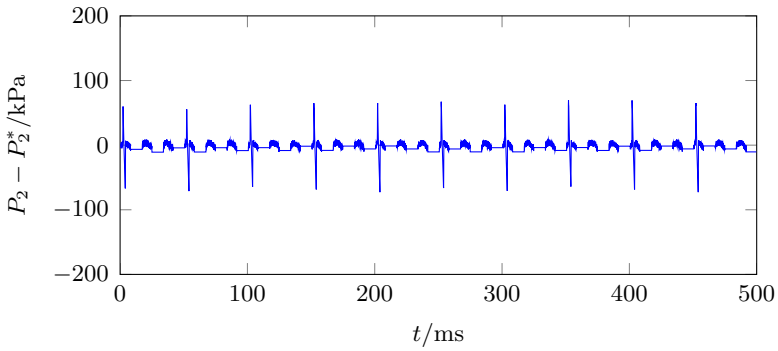

图 **13**　最优连续控制方案下，油管压强随时间的变化关系

由于此时 $Q_A\rho_1 \approx Q_B\rho_2$（比例约为 1.08），油管压强 P_2 的波动很小，关于目标压强的峰值偏差为 $D_{\max} = 70$ kPa，均方根偏差为 $D = 11$ kPa，约为优化前的百分之一。这表明本方案的调控非常有效。

综上，最优控制方案为：

- 单向阀周期 $\tau_A = 50/3$ ms；
- 喷油器 B 和 C 等间隔交替喷油（等效喷油周期 $\tau_B' = 50$ ms）；
- 相位差 $\phi_B = 2.0$ ms；

- 减压阀开启的时间为每个 τ_B' 周期内的 1.95 ms 之前和 4.3 ms 之后。在工作时间外减压阀关闭，在工作时间内（此时喷油器始终关闭）减压阀负责抵消油泵的进油，将 P_2 保持在目标压强 1×10^5 kPa。

2.5.3 减压阀受到限制情况：定性半定量分析

现在我们假设由于实际应用时的限制，减压阀每次开启后至少要关闭 10 ms 的时间。此时分析问题的总体思路与之前一样，即欲使供油曲线与喷油曲线贴合，但由于条件限制具体的策略也会有改变。

关于周期性的讨论仍适用于本题，因此可以设 $\tau_B = N\tau_A$。

由于油管压强变化不大，首先取油管压强为恒为 1×10^5 kPa 做估算，以定性半定量地得出一些有意义的结论。利用这个近似的数值模拟计算可快速得到：

1. 当 $N = 2$ 时，泵油时模拟步长 Δt 时间内高压泵泵入的油量典型大小约为 $\Delta m_A = 0.05$ mg，泵油的持续时间约为 $\tau_A^+ = 18$ ms；
2. 若打开减压阀，Δt 内减压阀流出的油量 $\Delta m_D = 0.170$ mg；
3. Δt 内喷油器喷出燃油质量约为 $\Delta m_B = 0.150$ mg；
4. 由 $E = \rho \mathrm{d}P/\mathrm{d}\rho = m\mathrm{d}P/\mathrm{d}m$，得到油管的压强变化率 $\frac{\mathrm{d}P_2}{\mathrm{d}t} = \frac{E}{m_2}\frac{\mathrm{d}m_2}{\mathrm{d}t}$，取 E 为 1×10^5 kPa 时的估算可得：
   油泵提供的压强增加率约为 $(\frac{\mathrm{d}P}{\mathrm{d}t})_A = 325$ kPa/ms，
   减压阀提供的压强减少率约为 $(\frac{\mathrm{d}P}{\mathrm{d}t})_D = -1107$ kPa/ms，
   喷油器提供的压强减少率为 $(\frac{\mathrm{d}P}{\mathrm{d}t})_B = -975$ kPa/ms；
5. 凸轮转一圈泵油量约为 76 mg，喷油器每次喷油量约为 29 mg，因此若一个周期内同时有油泵泵油和喷油器喷油，剩余 47 mg 的油需要通过减压阀释放。减压阀总开启时间由此算出为 2.8 ms。

根据以上结果，我们可以对 N 作出一定的限制。若 $N = 3$，至少有一个凸轮旋转周期内没有喷油器喷油，则 $\tau_A^+ = 18$ ms 的时间长度内至少有 7.5 ms 是减压阀关闭（减压阀不能连续开启，最好的状态就是在 $\tau_A^+ = 18$ ms 的中间开启减压阀，减压阀总开启时间为 2.8 ms），同时油泵在泵油的状态。根据泵油速度计算，这 7.5 ms 中不可避免增加的油压约为 1500 kPa。当 N 更大时这个压强增加值会更大。为了让油压波动较小，必有 $N < 3$；

另一方面，为了让泵油能更多抵消部分喷油导致的压降，N 越大越好，因此有 $N = 2$。此时，在喷油器完全开启的 2 ms 中，即使油泵也处于开启状态且减压阀关闭，压强也会下降约 1300 kPa。总的压强极差不可能小于这个值。也就是说，我们只要构造了一个压强的极差在 1300 kPa 左右的控制方案，它就是最佳的方案了。

2.5.4 减压阀受到限制情况：方案构造

根据上一节分析，最优方案为 $N = 2$ 时压强极差在 1300 kPa 左右，这个限制是因为喷油器喷油过快导致，无法避免。下面我们就尝试构造这样一种方案。

由于 $N = 2$，且每个凸轮周期内喷油器均需要，所以 B、C 两喷油器应交替喷油，于是可以等效成一个 $\tau'_B = 50$ ms 的喷油器（与上题相同），凸轮周期 $\tau_A = \tau'_B$。我们只需要完成一个周期内的构造，并且让此周期结束时和开始时的压强相等即可。

减压阀每次放油量不应超过喷油器每次喷油量 29 mg，这样压降才不会超过 1700 kPa。由于 $29 \times 2 < 76 < 29 \times 3$，应该有一次喷油器喷油和两次减压阀放油。考虑到减压阀开启有限制，喷油器喷油应在两次放油之间。我们将上述分析展现在示意图 14 中：

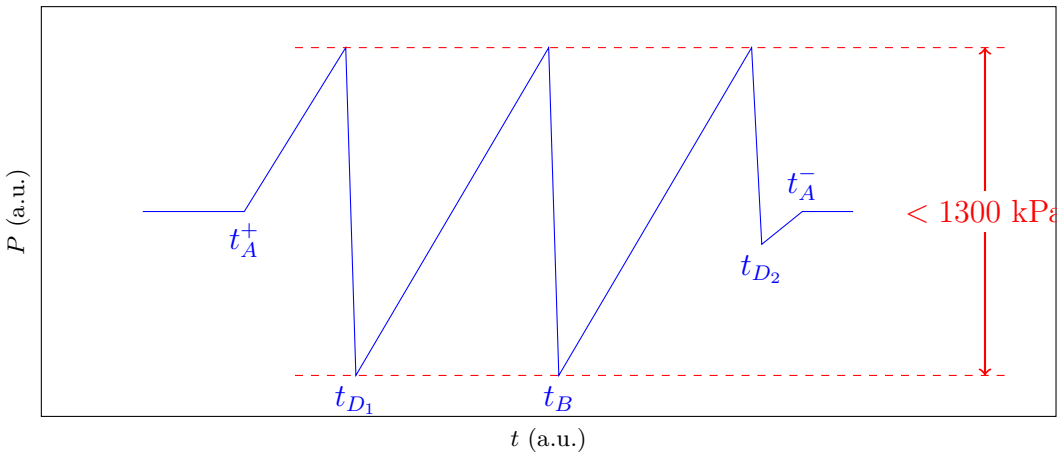

图 **14**　$\tau_A = \tau'_B$ 时，泵油、供油和喷油曲线的示意图

图 14 中，t_A^+ 开始高压油泵中有油泵出，泵油过程直到 t_A^- 才停止。在大部分时间油管内由于高压油泵泵油而压强增大；而在 t_{D_1} 附近减压阀开启压强下降，t_B 附近喷油器喷油导致压强下降，t_{D_2} 附近减压阀第二次开启。

t_B 附近 1300 kPa 的压降无可避免，我们要做的是调节 t_{D_1} 和 t_{D_2} 及减压阀开启时间使得压强的波动范围在这个限度以内，并且最终管内压强能回到标准压强。

2.5.5 减压阀受到限制情况：精确数值模拟

我们沿用之前连续可调情况下的模拟程序，但是由于减压阀开启限制需要做一些调整：

1. 定义布尔变量记录减压阀是否处于锁定的不可开启状态，并记录锁定的剩余时间，只有在减压阀未锁定时可以对其进行操作；
2. 每当减压阀被关闭时，进入 10 ms 的锁定时间；
3. 更新油管总油量 m_2 的代码区域需要做出较大改变。在减压阀锁定时直接按照无减压阀的情况更新 m_2；而在减压阀未锁定时需要根据此刻的时间和系统状态判断在

示意图中哪个部分，以给减压阀正确的开关指令。

最优方案给出的模拟结果如 15 所示：

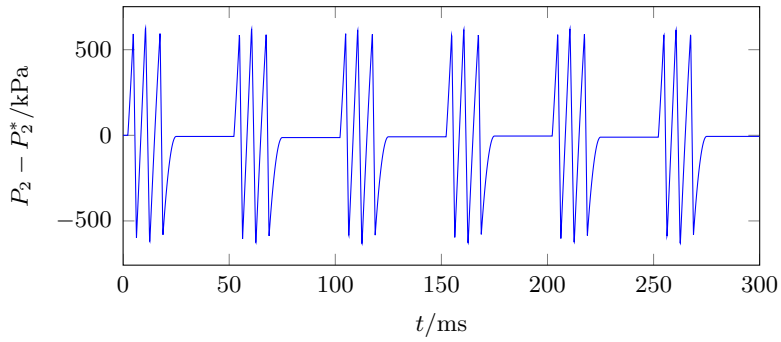

图 **15**　最优不连续控制方案下，油管压强随时间的变化关系

在该条件下，最大偏差 $D_{\max} = 630$ kPa，极差 $E = 1280$ kPa ，符合最优解的要求。计算得到此策略下均方根偏差为 $D = 226$ kPa，小于优化前（即第二题中）的四分之一，说明这一控制方案非常有效。

综上，我们的最终控制方案为：

- 单向阀周期 $\tau_A = 50$ ms；
- 喷油器 B 和 C 等间隔交替喷油；
- 相位差 $\phi_B = 10.35$ ms；
- 减压阀开启的时间为：每个 $\tau_A = 50$ ms 周期内的 $[4.84\text{ ms}, 6.29\text{ ms}]$ 和 $[17.43\text{ ms}, 18.75\text{ ms}]$。

三、 模型的评价

3.1 问题 1

该题中的模型所做的假设是管内压强是均匀的，在单向阀处燃油的流量根据题目中注 2 给出的公式计算。不考虑燃油的粘滞、涡流等复杂液体行为，也没有考虑燃油如何在油管内从一端流动至另一端。由于条件本身也是抽象化的，省略了很多细节，因此模型也简洁而清晰。

当然也正是因为其简洁性，没有太多实际应用的价值。

3.2 问题 2

在问题 2 中，我们将问题 1 的高压泵和喷油器具象化了，考虑了他们的工作原理。这个模型同样认为所有封闭腔内的压强是均匀的，燃油是静止的，忽略了燃油的复杂流体性质。还有一个重要的假设是把喷油器内的流动看做是连续两次扩散过程。

然而实际上喷油器处燃油的流动是非常复杂的，较精确的计算要使用混合多相流模型附加空穴模型 [5]，本题这里的近似较为粗糙，这也是此模型最大的缺陷。

但本模型的优点在于可修改性强，整个程序的核心物理量是管内燃油的质量 m_2，而油泵和喷油器对 m_2 的全部影响都体现为油管压强 P_2 和时间 t 的一个二元函数，这些函数互不干扰。因此后续改进可以是精确计算喷油器的流量模型，然后修改主程序中对应的函数即可，其它部分不受影响。

本模型另一个优点是可拓展性强。比如说加一个喷油器或者加一个阀门，都只要更新 m_2 时加两个函数值即可，这也为问题 3 提供了方便。

3.3 问题 3

问题 3 的模型主要是通过减压阀的控制，实现了更加稳定的油管压强。该模型联系了实际。喷油器是给发动机供油的，而发动机转速在短时间内一般不会剧烈改变，因此模型假定了喷油器的喷油周期是恒定的。凸轮的角速度也很难快速地精确控制，因此假定了凸轮是匀速旋转。

模型最终实现了非常好的稳压效果。若能连续调控减压阀，压强变化量几乎可以忽略。若考虑可能的实际情况，在减压阀调控受限时，通过一个周期内给定时间段仅开启两次减压阀也能实现不俗的效果。并且前文也证明了在这些假定下上述调控方案是最优的。

从实际情况来看，要更好地实现稳压需要更加精细的模型，例如要考虑针阀开启或关闭瞬间由于液体流动行为突然变化导致的压强波动 [6]。而这些实际情况中可能出现的影响压强的因素在本模型中无法计算。可以做的改进是多记录一些物理量，例如管道中的流速等，研究它们对压强的修正。

参考文献

[1] 白云. 高压共轨燃油系统循环喷油量波动特性研究[D]. 哈尔滨工程大学,2017.

[2] 仲志全, 李华宇, 尹琪. 发动机运行工况对机油耗影响的试验研究[J]. 内燃机工程,2004(05):69-71.

[3] 刘学龙, 苏万华, 战强. 高压油管对共轨系统性能影响的研究[J]. 内燃机工程,2010,31(05):47-51+57.

[4] 陶希成. 柴油机锥形孔喷嘴内空穴流动及其对喷雾影响的试验研究[D]. 江苏大学,2016.

[5] 崔慧峰, 罗福强, 董少锋, 梁昱, 周立迎. 柴油机渐缩形喷孔喷嘴流动特性研究[J]. 农业机械学报,2013,44(11):19-25.

[6] 丁晓亮, 张幽彤, 苏海峰. 压电式共轨系统喷油量压力波动修正策略研究[J]. 北京理工大学学报. 2010(09)

附录 A　源代码

1.1　文件结构说明

1. 本文中全部代码使用 Python 3 编写，并使用 3.7.1 版本的解释器解释运行；
2. data 文件夹存放了以下内容：
   - 题目中提供的数据：outline.raw 为附件 1 中的凸轮边缘曲线，movement.raw 为附件 2 中的针阀运动曲线，elasticity.raw 为弹性模量与压力的关系；
   - 处理原始数据生成可用数据的程序：process_outline.py 处理凸轮边缘曲线生成高度与转角的关系并存放在 height.dat 中，process_elasticity.py 处理弹性模量生成压力与密度的关系并存放在 density.dat 中；
   - 函数库程序：lib.py 存放了问题 1 至 3 所共用的一系列函数。
3. 1 文件夹存放了以下内容：
   - 基于稳压近似的估算程序：analysis.py，结果输出到屏幕上；
   - 系统 1 的数值模拟程序：simulation1.py，结果保存到 P-t.dat、D-tauA20/40/60/80/100.dat、tau-N.dat；
   - 系统 1 压力调整的模拟程序：adjustment.py，结果保存到 adjustment_in_2000/5000/10000.dat。
4. 2 文件夹存放了以下内容：
   - 系统 2 的数值模拟程序：simulation2.py，结果保存到 P-t.dat、D-omega.dat、m1-t.dat、QA-t.dat。
5. 3 文件夹存放了以下内容：
   - 在连续情况下，系统 3 的数值模拟程序：simulation3_1.py，结果保存到 t-P-1.dat；
   - 在非连续情况下，系统 3 的数值模拟程序：simulation3_2.py，结果保存到 t-P-2.dat。

1.2　函数库及预处理源代码

Listing 1: 将弹性模量处理为密度数据

```
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]
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*0.5
    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*0.5
    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: 将凸轮轮廓处理为高度数据

```
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()
```

Listing 3: 函数库

```
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
tau_0 = 100
tau_A_close = 10

def get_data_from(path_to_file):
    f =  open(path_to_file, encoding = 'utf-8', mode = 'r')
    l = [[ float(field) for field in line.strip('\n').split('\t')] for line in f]
    f.close()
    return l

def write_data_to(path_to_file, data):
    f =  open(path_to_file, encoding = 'utf-8', mode = 'r')
    for item in data:
        item_s = [ str(field) for field in item]
        f.write(''.join(item_s) + '\n')
    f.close()

def Q(P_high, P_low, rho_high, A):
    return C * A * math.sqrt(2 * (P_high - P_low) / rho_high)

def interpolation(x, x_array, y_array, equally_spaced = False):
    """
    目的: 找到自变量 x 所在分段函数的区间
    输入:
    - x: 待求值的自变量;
    - x_array: 已知数据点中自变量的升序系列
    - y_array: 分别对应于 x_array 中每一个点的因变量值
```

```
    - 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；
    输出：数组 (x_1, x_2, y_1, y_2)，其中 x_1 <= x < x_2
    """

    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')

    imin = 0
    imax =  len(x_array) - 1
    if equally_spaced:
        space = x_array[1] - x_array[0]
        i =  int((x - x_array[0]) / space)
        return y_array[i] + (y_array[i+1] - y_array[i]) * (x - x_array[i]) / space
    else:
        while imax - imin > 1:
            temp =  int((imax + imin)/2)
            if x_array[temp] <= x:
                imin = temp
            else:
                imax = temp
        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)

```

```python
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)
    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
    S_B = math.pi * 0.49  # 圆锥底部截面积
    rho_1 = calc_rho(P_1)  # P_1压强下的煤油密度
    # 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
    # p2 为圆锥中间压强，满足 p2=P_1+delta_P_1
    x = 0
    # 实际待求方程组为：
    # x+y=P_1-P_0
    # return = S_B_prime*math.sqrt(2*x/rho_1) = S_B*math.sqrt(2*(y)/calc_rho(P_1-x))

    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
    x_2 = 0.75 * (P_1 - P_0)  # 这里x_1，x_2的初值其实是任意的
    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
    max_times = 1000  # 设置一个迭代次数上限
    for j in  range(max_times):
        if  abs(x_1 - x_2) < 0.00001:  # 设置允差
            x = x_2
            break
        else:
            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))
            x_1 = x_2 - (x_2 - x_1) * f_2 / (f_2 - f_1)
            # 防止前几次迭代使得x越界
            if x_1 > P_1 - P_0:
```

```
                x_1 = P_1 - P_0
            elif x_1 < 0:
                x_1 = 0
            # 交换下标1和2用于下次迭代
            x_1, x_2 = x_2, x_1
            f_1 = f_2
    # 迭代超过上限就报错
    if x == 0: raise Exception("Secant method doesn't converge! (In yuanzhui())")
    return C * S_B_prime * math.sqrt(2 * x / rho_1)

# 测试函数

def test_interpolation():
    x_array = (1, 2, 3, 4)
    y_array = (1, 4, 9, 16)
    x = 2.5
    print(interpolation(x, x_array, y_array, True))

# test_interpolation()
```

1.3 第一题源代码

Listing 4: 系统 1 的解析计算

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

# 油泵压力为 1.6e5 kPa、油管压力为 1e5 kPa 时，理论的开启时长

rho_1 = calc_rho(P_1)
rho_2 = rho_std
Q_A = Q(P_1, P_2_std, rho_1, S_A)
m_decrease = rho_2 * V_B
m_increase = m_decrease
tau_A = m_increase / Q_A / rho_1
tau_A_open = tau_A_close * tau_A / (tau_0 - tau_A)

print('油泵压力为 1.6e5 kPa、油管压力为 1e5 kPa 时，理论的开启时长为：', tau_A_open)

# 油泵压力为 1.6e5 kPa、油管压力为 1e5 kPa 时，理论的开启时长

P_2 = 1.5e5
rho_2 = calc_rho(P_2)
Q_A = Q(P_1, P_2, rho_1, S_A)
m_decrease = rho_2 * V_B
```

```
m_increase = m_decrease
tau_A = m_increase / Q_A / rho_1
tau_A_open = tau_A_close * tau_A / (tau_0 - tau_A)

print('油泵压力为 1.6e5 kPa、油管压力为 1e5 kPa 时，理论的开启时长为：', tau_A_open)

# # 经过 2、5、10 秒调整时，理论的开启时长
# relax_periods_list = [20, 50, 100]
# rho_increase = rho_2 - rho_std
# m_increase = rho_increase * V_2
# for relax_periods in relax_periods_list:
#     net_m_increase_per_period = m_increase / relax_periods
#     m_decrease_per_period = rho_2 * V_B
#     m_increase_per_period = net_m_increase_per_period + m_decrease_per_period
#     print(net_m_increase_per_period, m_decrease_per_period)
#     Q_A = Q(P_1, P_2, rho_1, S_A)
#     tau_A = m_increase / Q_A / rho_1
#     tau_A_open = tau_A_close * tau_A / (tau_0 - tau_A)
#     print('在 %d 秒调整中，理论的开启时长为：' % relax_periods, tau_A_open)
```

Listing 5: 系统 1 的数值模拟

```
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: [0.294 + i * 0.001 for i in  range(10)],
    20: [0.289 + i * 0.001 for i in  range(10)],
    50: [0.289 + i * 0.001 for i in  range(10)],
    100: [0.287 + i * 0.001 for i in  range(5)],
    200: [0.287 + i * 0.001 for i in  range(5)],
    500: [0.286 + i * 0.001 for i in  range(4)],
    1000: [0.286 + 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(20)] # A 的开启时间
# tau_A_open_list = [0.288 + 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')
    # f.write('%s\t%s\n' % ('tauA', 'D'))
    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
        # f.write('%f\t%f\n' % (tau_A_open, D))
        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.close()
    f.write('%d\t%f\n' % (period_number, best_tau_A_open))
f.close()
```

Listing 6: 系统 1 的压力调整模拟

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

dt = 0.01
relax_period_number_dict = {
    # 对于每一个给定的弛豫时间，要尝试的那些开启时间，现在只是猜想
    20: [0.905 + i * 0.001 for i in  range(31)],
    50: [0.745 + i * 0.001 for i in  range(31)],
    100: [0.735 + i * 0.001 for i in  range(31)]
    }
stable_period_number = 100
period_steps =  int(tau_0 / dt)
stable_steps = stable_period_number * period_steps

# 系统参数
rho_1 = calc_rho(P_1)
```

```
rho_2 = rho_std
P_2_std_new = 1.5e5
tau_A_close = 10
tau_A_open_stable = 0.752
A_initial_phase = 0

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 + A_initial_phase) % 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 % 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

for relax_period_number, tau_A_open_list in relax_period_number_dict.items():
    f = open('adjustment_in_%d.dat' % (relax_period_number * tau_0), encoding = 'utf-8', mode = 'w')
    f.write('%s\t%s\n' % ('tauA', 'D'))
    relax_steps = relax_period_number * period_steps
    count = 0
    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(-relax_steps, stable_steps):
            t = dt * i
            if (t < 0):
```

```
                Q_A = calc_Q_A(t, P_2, tau_A_open)
            else:
                Q_A = calc_Q_A(t, P_2, tau_A_open_stable)
            # 进油
            delta_m = Q_A * rho_1 * dt
            m_2 += delta_m
            Q_B = calc_Q_B(t)
            # 出油
            delta_m = Q_B * rho_2 * dt
            m_2 -= delta_m
            rho_2 = m_2 / V_2
            P_2 = calc_pres(rho_2)
            if (t > 0): D += (P_2 - P_2_std_new)**2
            # if (i == 0): print(tau_A_open, P_2)
        D = math.sqrt(D / stable_steps)
        f.write('%f\t%f\n' % (tau_A_open, D))
        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.close()
```

1.4 第二题源代码

Listing 7: 系统 2 的数值模拟

```
import sys
sys.path.append('..')
from data.lib import *
# import matplotlib.pyplot as plt

# 运行控制参数
# omega_list = [0.0234 + 0.0001*j for j in range(10)]
omega_list = [0.0238]
dt = 0.01
# period_number = 100
period_number = 10
period_steps =  int(tau_0 / dt)
total_steps = period_number * period_steps

# 系统参数
rho_1_std = calc_rho(500)
rho_2 = rho_std
m_1_std = rho_1_std * calc_V_1(min_h)
alpha_0 = 1.5 * pi   # 初始时刻alpha
t_0 = 0     # 初始时刻(决定了B的相位)

def calc_Q_A(P_2, rho_1):
```

```
    """
    输入：时间 t，油管压力 P_2，P_1为高压油泵内压强
    输出：该时间单向阀 A 的流量
    """
    P_1 = calc_pres(rho_1)
    if P_1 > P_2:
        return Q(P_1, P_2, rho_1, S_A)
    else:
        return 0

def calc_Q_B(t, P_2):
    """
    输入：时间 t，油管压强P_2
    输出：该时间喷油嘴 B 的流量
    """
    phase = (t + t_0) % 100
    z = calc_z(phase)
    if z <= 0:
        return 0
    else:
        return solve_Q_B(P_2, z)

# f = open('D-omega.dat', encoding = 'utf-8', mode = 'w')
# f.write('%s\t%s\n' % ('omega', 'D'))
# f = open('P-t.dat', encoding = 'utf-8', mode = 'w')
# f.write('%s\t%s\n' % ('t', 'P'))
# f = open('m1-t.dat', encoding = 'utf-8', mode = 'w')
# f.write('%s\t%s\n' % ('t', 'm1'))
f =  open('QA-t.dat', encoding = 'utf-8', mode = 'w')
f.write('%s\t%s\n' % ('t', 'QA'))
count = 0
for omega in omega_list:
    D = 0
    P_2 = P_2_std
    m_2 = calc_rho(P_2) * V_2
    rho_1 = rho_1_std
    m_1 = m_1_std   # 初始时刻油量
    alpha = alpha_0

    for i in  range(total_steps):
        if  abs(alpha % (2*pi) - 1.5*pi) <= omega * dt: m_1 = m_1_std
        t = i * dt
        V_1 = calc_V_1(calc_height(alpha % (2*pi)))
        rho_1 = m_1/V_1

        # 演化一步
        # 进油
```

```
        Q_A = calc_Q_A(P_2, rho_1)
        delta_m = Q_A * rho_1 * dt
        m_2 += delta_m
        m_1 -= delta_m
        # 出油
        Q_B = calc_Q_B(t, P_2)
        delta_m = Q_B * rho_2 * dt
        m_2 -= delta_m
        # 更新密度和压强
        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
        # if i % 100 == 0: f.write('%f\t%d\n' % (t, int(P_2 - P_2_std)))
        # if i % 100 == 0: f.write('%f\t%f\n' % (t, m_1))
        if i % 100 == 0: f.write('%f\t%f\n' % (t, Q_A))
        alpha += omega * dt
    D = math.sqrt(D / total_steps)  # 均方误差
    # f.write('%f\t%f\n' % (omega, D))
    count += 1
    print('完成了 %d 个角速度的计算，一共 %d 个角速度' % (count,  len(omega_list)))
f.close()
```

1.5 第三题源代码

Listing 8: 系统 3 的数值模拟，连续情况

```
# coding=utf-8
import sys
sys.path.append('..')
from data.lib import *
import matplotlib.pyplot as plt


# 系统参数
rho_1 = calc_rho(P_1)
rho_2 = rho_std
A_initial_phase = 5 # （是不是要改成一个 list？）

def calc_Q_A2(P_2, rho_h):
    """
    输入：时间 t，油管压力 P_2，P_h为高压油泵内压强
    输出：该时间单向阀 A 的流量
    """
    P_h = calc_pres(rho_h)
    if P_h > P_2:
```

```
        return Q(P_h, P_2, rho_h, S_A)
    else:
        return 0

def calc_Q_B(t, P_2):
    """
    输入：时间 t，油管压强P_2
    输出：该时间喷油嘴 B 的流量
    """
    phase = t % 50
    z = calc_z(phase)
    if z <= 0:
        return 0
    else:
        return solve_Q_B(P_2, z)


f =  open('t-P-1.dat', encoding = 'utf-8', mode = 'w')
f.write('t\tP_2\n')

# 运行控制参数
N_list = [3]
dt = 0.01
period_number = 10

rho_h_std = calc_rho(500)
m_h_std = rho_h_std * calc_V_h(min_h)

count = 0
N_len =  len(N_list)
for N in N_list:
    omega = 2*N*pi/50   # rad/ms
    X = []
    Y = []
    temp_Y = []
    tau_temp_Y = 1     # Y的平滑时间

    D = 0
    P_2 = P_2_std
    m_2 = calc_rho(P_2) * V_2
    rho_h = rho_h_std
    period_steps =  int(2 * pi / omega / dt)
    total_steps = period_number * period_steps * N  # period_number 个50ms
    m_h = m_h_std   # 初始时刻油量
    alpha_0 = 1.5*pi   # 初始时刻alpha

    t_0 = -2     # 初始时刻(决定了B的相位)
```

```
67
68      t = t_0
69      alpha = alpha_0
70
71      B_open = False   # True表示此period内存在喷嘴喷油(注意到 period=50/N <=
            50ms,并假设在第一个period内B开启)
72      After_B = False  # 第一个周期中表示B喷嘴是否(喷过或者正在喷)
73
74      for i in  range(total_steps):
75
76          if  abs(alpha % (2*pi) - 1.5*pi) <= omega*dt:
77              m_h = m_h_std
78          t += dt
79          V_h = calc_V_h(calc_height(alpha % (2*pi)))
80          rho_h = m_h/V_h
81
82          # 演化一步
83          # 进油流量
84          Q_A = calc_Q_A2(P_2, rho_h)
85          delta_m_A = Q_A * rho_h * dt
86
87          m_h -= delta_m_A
88          # 出油流量
89          Q_B = calc_Q_B(t, P_2)
90          delta_m_B = Q_B * rho_2 * dt
91
92          delta_m_D_std = Q(P_2, 500, rho_2, S_A) * rho_2 * dt  # 通过减压阀的流量上限
93          # 根据 B_open 判断油管内油量变化
94          B_open = (i/period_steps) % N < 1
95          if not B_open:
96              After_B = False
97              if P_2 > P_2_std:   # 自动稳压
98                  m_2 += delta_m_A - delta_m_D_std
99              else:
100                 m_2 += delta_m_A
101         else:   # 该period内有B喷油
102             phase_h = (i/period_steps) % N    # [0, 1]之间的数，表示处于一个period的哪个位置
103             if After_B == False and Q_B > 16:
104                 After_B = True
105
106             if phase_h < 0.13:
107                 if P_2 > P_2_std:  # 自动稳压
108                     m_2 += delta_m_A - delta_m_B - delta_m_D_std
109                 else:
110                     m_2 += delta_m_A - delta_m_B
111             else:
112                 if After_B and delta_m_B <= delta_m_A:
```

```
                    if P_2 > P_2_std:  # 自动稳压
                        m_2 += delta_m_A - delta_m_B - delta_m_D_std
                    else:
                        m_2 += delta_m_A - delta_m_B
                else:
                    if P_2 > P_2_std + 65:  # 给P_2设定一个上限
                        m_2 += delta_m_A - delta_m_B - delta_m_D_std
                    else:
                        m_2 += delta_m_A - delta_m_B

        # 更新密度和压强
        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
        # f.write(str(t) + '\t' + str(int(P_2)) + '\n')

        if i%10==0:
            temp_Y.append(P_2)  # 在这里添加想要观察的物理量作为Y轴
            if (i/10)%tau_temp_Y == 0:
                X.append(i / 100)
                Y_new =  sum(temp_Y)/ len(temp_Y)
                Y.append(Y_new)
                temp_Y = []

        alpha += omega*dt
        if i%10==0:
            f.write('%f\t%f\n' % (i / 100, P_2-P_2_std))
    D = D / total_steps  # 均方误差



    count += 1
    print('Accomplished(%d/%d)' % (count, N_len))
    print( sum(Y))
    plt.plot(X,Y)
    plt.show()
f.close()
```

Listing 9: 系统 3 的数值模拟，非连续情况

```
# coding=utf-8
import sys
sys.path.append('..')
from data.lib import *
import matplotlib.pyplot as plt

```

```
7
8  # 系统参数
9  rho_1 = calc_rho(P_1)
10 rho_2 = rho_std
11 A_initial_phase = 5 # （是不是要改成一个 list？）
12
13 def calc_Q_A2(P_2, rho_h):
14     """
15     输入：时间 t，油管压力 P_2，P_h为高压油泵内压强
16     输出：该时间单向阀 A 的流量
17     """
18     P_h = calc_pres(rho_h)
19     if P_h > P_2:
20         return Q(P_h, P_2, rho_h, S_A)
21     else:
22         return 0
23
24 def calc_Q_B(t, P_2):
25     """
26     输入：时间 t，油管压强P_2
27     输出：该时间喷油嘴 B 的流量
28     """
29     phase = t % 50
30     z = calc_z(phase)
31     if z <= 0:
32         return 0
33     else:
34         return solve_Q_B(P_2, z)
35
36
37 f =  open('t-P-2.dat', encoding = 'utf-8', mode = 'w')
38 f.write('t\tP_2\n')
39 # 运行控制参数
40 N_list = [1]
41 dt = 0.01
42 period_number = 6
43
44 rho_h_std = calc_rho(500)
45 m_h_std = rho_h_std * calc_V_h(min_h)
46
47
48 omega = 2*pi/50   # rad/ms
49 X = []
50 Y = []
51 temp_Y = []
52 tau_temp_Y = 1     # Y的平滑时间
53
```

```
D = 0
P_2 = P_2_std
m_2 = calc_rho(P_2) * V_2
rho_h = rho_h_std
period_steps =  int(2 * pi / omega / dt)
total_steps = period_number * period_steps  # period_number 个50ms
m_h = m_h_std   # 初始时刻油量
alpha_0 = 1.5*pi   # 初始时刻alpha

t_0 = -10.35     # 初始时刻(决定了B的相位)

t = t_0
alpha = alpha_0

# 记录系统状态
dec_time = 0.0  # 第一个周期中表示第几次降压(减压阀或者喷嘴)

D_lock = False  # True 时D锁定，不能开启，每次锁定10ms
D_lock_time_std = 10.0  # 每次锁定的时间
D_remain_time = 0.0     # 剩余锁定时间
D_P_upper = 600    # 高于标准P_2这个值时自动进入开启状态
D_open = False      # 记录前一时刻D阀门是否打开

for i in  range(total_steps):

    # 更新D_lock状态
    if D_remain_time > 0:
        D_remain_time -= dt
    if D_remain_time <= 0:
        D_lock = False

    if  abs(alpha % (2*pi) - 1.5*pi) <= omega*dt:
        m_h = m_h_std
    t += dt
    V_h = calc_V_h(calc_height(alpha % (2*pi)))
    rho_h = m_h/V_h

    # 演化一步
    # 进油流量
    Q_A = calc_Q_A2(P_2, rho_h)
    delta_m_A = Q_A * rho_h * dt

    m_h -= delta_m_A
    # 出油流量
    Q_B = calc_Q_B(t, P_2)
    delta_m_B = Q_B * rho_2 * dt

```

```
    if D_lock:  # D锁定
        m_2 += delta_m_A - delta_m_B
        if dec_time == 1.5 and Q_B > 10:# 喷嘴开始喷油
            dec_time = 2

        elif dec_time == 2 and Q_B < 5:# 喷嘴喷油结束
            dec_time = 2.5
    else:   # D可以控制
        delta_m_D_std = Q(P_2, 500, rho_2, S_A) * rho_2 * dt  # 通过减压阀的流量

        phase_h = (i/period_steps) % 1    # [0, 1]之间的数，表示处于一个period的哪个位置
        # 更新After_B状态
        if phase_h < 2/period_steps:    # 初始时刻
            dec_time = 0

        if dec_time == 0:
            if P_2 > P_2_std + D_P_upper:   # 第一次打开减压阀
                dec_time = 1
                D_open = True
                m_2 += delta_m_A - delta_m_B - delta_m_D_std
            else:
                m_2 += delta_m_A - delta_m_B

        elif dec_time == 1:    # 第一次减压阀减压结束
            if P_2 < P_2_std - D_P_upper:   # 减压阀关闭
                dec_time = 1.5
                D_open = False
                D_lock = True
                D_remain_time = D_lock_time_std
                m_2 += delta_m_A - delta_m_B
            else:
                m_2 += delta_m_A - delta_m_B - delta_m_D_std

        elif dec_time == 1.5:
            m_2 += delta_m_A - delta_m_B
            if Q_B > 10:  # 喷嘴开始喷油
                dec_time = 2

        elif dec_time == 2:
            m_2 += delta_m_A - delta_m_B
            if Q_B < 5:  # 喷嘴喷油结束
                dec_time = 2.5

        elif dec_time == 2.5:
            if P_2 > P_2_std + D_P_upper:   # 第二次打开减压阀
                dec_time = 3
                D_open = True
```

```
                m_2 += delta_m_A - delta_m_B - delta_m_D_std
            else:
                m_2 += delta_m_A - delta_m_B

        elif dec_time == 3 and D_open:
            if P_2 < P_2_std - 0.973*D_P_upper:   # 减压阀关闭
                dec_time = 3.5
                D_open = False
                D_lock = True
                D_remain_time = D_lock_time_std
                m_2 += delta_m_A - delta_m_B
            else:
                m_2 += delta_m_A - delta_m_B - delta_m_D_std
        else:   # 第二次减压阀减压结束
            # 自动稳压
            if P_2 > P_2_std:
                D_open = True
                m_2 += delta_m_A - delta_m_B - delta_m_D_std
            else:   #压强小于等于标准值
                if D_open:
                    D_open = False
                    D_lock = True
                    D_remain_time = D_lock_time_std
                    m_2 += delta_m_A - delta_m_B
                else:
                    m_2 += delta_m_A - delta_m_B

    # 更新密度和压强
    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
    # f.write(str(t) + '\t' + str(int(P_2)) + '\n')

    temp_Y.append(P_2)  # 在这里添加想要观察的物理量作为Y轴
    if i % tau_temp_Y == 0:
        X.append(i / 100)
        Y_new =  sum(temp_Y)/ len(temp_Y)
        Y.append(Y_new)
        temp_Y = []

    alpha += omega*dt
    if i % 10 == 0:
        f.write('%f\t%f\n' % (i / 100, P_2-P_2_std))
D = D / total_steps  # 均方误差

print('Accomplished')
```

```
195
196 plt.plot(X, Y)
197 plt.show()
198
199 f.close()
```