# 有刷直流电机 PID 参数自整定研究

马骋远 [1]

[1] 上海外国语大学附属外国语学校

**摘要**

有刷直流电机作为应用广泛的常见作动机构，常使用 PID 算法进行其位置，转速的控制。为了解决 PID 算法调参依赖经验与试错，耗时长的问题，基于将调参过程中的“试错”离线化、自动化的技术路线，本文提出了一种 PID 参数自整定的算法，通过软件程序对受控对象的分部数学建模与未知参数拟合，使用离线模拟代替在线试错， 制定量化指标代替人工观察，应用启发式粒子群优化算法代替经验调优，并使用 C++对算法进行高效率多线程实现，在有刷直流电机驱动的四轮小车实验平台上取得了良好的效果。与其他方法相比，本方法简单直观，不需要计算受控对象的传递函数或状态空间，也不需要大量的在线运算，还可以拓展到其他控制器与受控对象的参数整定，适用范围更广，更为实用。

**关键字**: PID; 自整定; 粒子群算法

# 目录

# 1 引言

## 1.1 背景

比例-积分-微分控制算法（PID）算法作为上世纪中叶提出的闭环控制算法，由于其简单，直观，效果好，即使是在 21 世纪控制理论蓬勃发展的当下在工业界仍然受到青睐，PID 的核心控制律如下：

$$u(t) = K_p e(t) + K_i \int_0^t e(x)dx + K_d \frac{de(t)}{dt} \tag{1}$$

其中 $e(t)$ 为受控对象在 $t$ 时刻的误差，$u(t)$ 为控制器在 $t$ 时刻的输出。等式右边的三项依次为比例项，积分项和微分项，$K_p$,$K_i$,$K_d$ 分别为比例系数，积分系数与微分系数，也是常规 PID 算法仅有的三个参数。在大多数情况下，理论上只要技术人员适当地选取以上三个参数，PID 算法就可以取得较为满意的控制效果。然而在事实上，正是因为 PID 不依赖于受控对象的物理模型，以上三个参数的整定选取往往只能依赖于调参人员的经验与技巧，而即使在经验指导下参数的整定通常也都是一个试错的过程。虽然人们已经提出了很多 PID 自整定的方案，但是那些方法更多依赖于受控对象的传递函数，状态空间等，而现实条件下，由于受控对象模型的复杂性，这些往往都是难以计算的，因此这些方法大多停留在理论层面，缺乏可操作性。

有刷直流电机在机器人当中有着广泛的应用，为了完成对于其转速，角度的精确控制，类似 PID 的控制算法不可或缺。我们在参与机器人赛事时就多次被 PID 的调参问题所困扰，因此本研究聚焦于有刷直流电机 PID 参数的自整定方法，实现 PID 参数的初步自整定。

## 1.2 国内外研究现状

在 1942 年，John G. Ziegler 和 Nathaniel B. Nichols 发展了一套基于系统震荡周期的启发式 PID 整定方法称为 Ziegler-Nichols 法 [1]，然而由于其通用性其在某些特定情境下的应用往往不尽如人意。

随后得益于控制理论的发展，大量理论上的 PID 自动整定方法被提出，包括但不限于临界比例度法 [2]，阶跃响应法等，大部分以上的整定方法大量依赖于现代控制理论当中的传递函数，状态空间的计算等，因而这些方法依赖于精确的系统建模，过于理想化，因而实用性有所不足。即使存在不过于依赖模型的在线整定算法如基于神经网络的 PID 自整定 [3]，基于模糊逻辑的 PID 自整定 [4]，其往往计算量巨大并不适用于微控制器。

# 2 主体

## 2.1 实验平台

我们拟采用基于 Cortex M3 的 STM32 作为微控制器，7.4 V 镍氢电池作为电源，使用 1250 Hz 的 PWM 控制器驱动一个由四个同规格的有刷直流电机组成的小车底盘作为受控对象，为了模拟实际过程中的载荷，底盘上外加配重。在数据采集上，使用直接安装在驱动轴上的两个精度均为 1° 的光电编码器来测量小车的位置信息。PMW 控制器将 $[-127, 127]$ 内的整数指令映射到占空比。

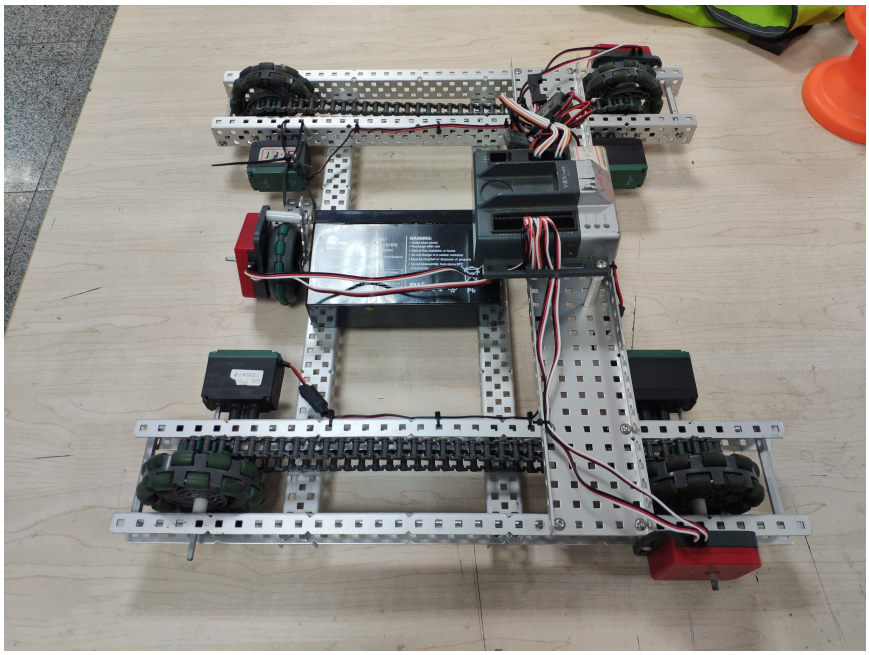


图 1: 试验用小车

在本课题中，我们主要研究小车位置控制 PID 算法的参数自整定。

## 2.2 技术路线

为了进行初步 PID 的参数整定，我们首先对受控对象进行了粗糙的带有未知参数的物理建模。具体来说，受控对象的模型可以分为如下两部分：

1. 直流电机马达：即直流电机马达本身。由于我们能够测量的变量只有小车的速度（微分后即马达转速），以及系统的电压。因此我们建立马达电压，转速与电流，转矩的模型。由于

直流电机马达内部结构相对清晰固定，厂家提供的内部参数较为完备。因此马达本身的建模可以较为精确地完成。

2. 传动：马达从的输出转矩到小车车身的运动经过了传动轴与车轮的传动。整体来看，小车除了动力还受到地面摩擦阻力，轴的转动阻力，惯性等复杂因素的影响，难以精确测得精确建模所需要的各项参数。因此我们将建立一个带有未知参数的较为通用的运动学模型。

接下来，我们试图使用建立的模型拟合实地采集的真实数据，并从中解算出未知参数的值。

在模型参数明确之后，我们将在电脑上编写仿真程序依据建立的模型对受控对象进行离线模拟，以避免在受控对象本身上进行大量的试验。在模拟程序完成之后，将其与 PID 算法本身进行结合，实现在给定 PID 参数与初始条件的情况下对于 PID 运行情况的快速离线模拟。

之后，基于模拟的结果建立评价一组指标函数，考虑的因素有超调量，响应时间，峰值电流等，指标函数将指示一组给定 PID 参数的优劣。

至此，PID 参数的整定问题被转化为了多个指标函数的最优化问题。我们将使用启发式的最优化算法（如粒子群优化，模拟退火等）寻找令指标函数（局部）最优的参数值，作为整定的结果。

最后，将计算出来的参数值应用在实际受控对象上进行验证，观察控制效果。

## 2.3 直流电机马达模型的建立

我们首先建立对于有刷直流电机电压，转速与电流的模型 [5]。

基于 PWM 控制的有刷直流电机的电路图如下（忽略换向）：

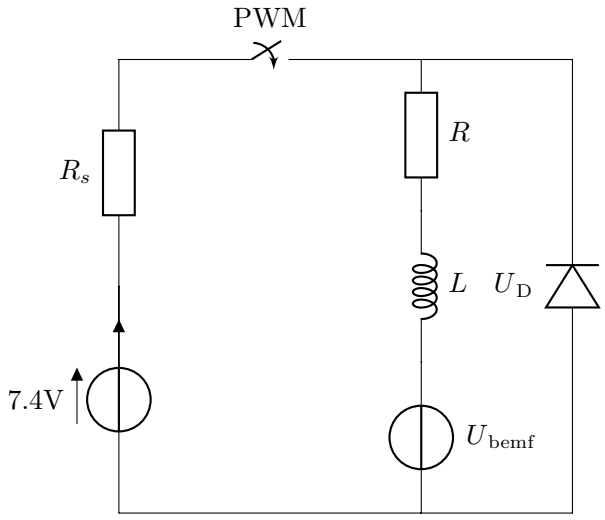



图 2: 有刷直流电机的内部电路

在本节的建模过程中，使用到的符号以及部分参数的测量值如下：

表 1: 马达建模中使用的符号与部分参数测量值

| 符号 | 解释 |
|---|---|
| $f_{\mathrm{pwm}} \approx 1250\,\mathrm{Hz}$ | PWM 频率 |
| $t_{\mathrm{pwm}} = (f_{\mathrm{pwm}})^{-1}$ | PWM 单周期长 |
| $t_{\mathrm{on}}$ | PWM 单周期高电平时长 |
| $t_{\mathrm{off}}$ | PWM 单周期低电平导通时长 |
| $I(t)$ | PWM 单周期内 $t$ 时刻的电流 |
| $I_0$ | PWM 周期开始时的电流 |
| $I_{\max}$ | PWM 峰值电流 |
| $I_{\mathrm{f}}$ | PWM 周期结束时的电流 |
| $U_{\mathrm{b}} \approx 7.4\,\mathrm{V}$ | 电池电压 |
| $U_{\mathrm{bemf}}$ | 反电动势 |
| $U_{\mathrm{L}}$ | 电感电压 |
| $U_{\mathrm{D}}$ | 二极管压降 |
| $L \approx 650\,\mu\mathrm{H}$ | 电感 |
| $R \approx 1.6\,\Omega$ | 马达内阻 |
| $R_{\mathrm{s}} \approx 0.28\,\Omega$ | 系统内阻 |

PWM 分为高电平与低电平两个阶段，我们将分别对两个阶段进行建模。

### 2.3.1 高电平段建模

在 PWM 的高电平段，电流沿图 (2) 的左半部分顺时针流动，由基尔霍夫电流定律，有

$$U_{\mathrm{b}} - I(t)(R + R_{\mathrm{s}}) - U_{\mathrm{L}} - U_{\mathrm{bemf}} = 0 \tag{2}$$

由 $U_{\mathrm{L}} = L\frac{dI(t)}{dt}$，得

$$U_{\mathrm{b}} = L\frac{dI(t)}{dt} + I(t)(R + R_{\mathrm{s}}) + U_{\mathrm{bemf}} \tag{3}$$

以初始条件 $I(0) = I_0$ 解得

$$I(t) = e^{-\frac{t(R+R_{\mathrm{S}})}{L}}\left(I_0 - \frac{U_{\mathrm{b}} - U_{\mathrm{bemf}}}{R + R_{\mathrm{s}}}\right) + \frac{U_{\mathrm{b}} - U_{\mathrm{bemf}}}{R + R_{\mathrm{s}}} \tag{4}$$

$I(t)$ 显然在高电平段单调增，则取 $t = t_{\mathrm{on}}$，得

$$I_{\max} = e^{-\frac{t_{\mathrm{on}}(R+R_{\mathrm{S}})}{L}}\left(I_0 - \frac{U_{\mathrm{b}} - U_{\mathrm{bemf}}}{R + R_{\mathrm{s}}}\right) + \frac{U_{\mathrm{b}} - U_{\mathrm{bemf}}}{R + R_{\mathrm{s}}} \tag{5}$$

令

$$c_{\mathrm{on}} = -\frac{t_{\mathrm{on}}(R+R_{\mathrm{s}})}{L}, e_{\mathrm{on}} = e^{c_{\mathrm{on}}}, I_{\mathrm{on}} = \frac{U_{\mathrm{b}} - U_{\mathrm{bemf}}}{R+R_{\mathrm{s}}} \tag{6}$$

则

$$I_{\max} = e_{\mathrm{on}}\left(I_0 - I_{\mathrm{on}}\right) + I_{\mathrm{on}} \tag{7}$$

最后，为之后计算平均电流做准备，我们计算此时 $I(t)$ 的积分

$$\int_0^{t_{\mathrm{on}}} I(t)dt = t_{\mathrm{on}}\left[I_{\mathrm{on}} - \frac{\left(1-e_{\mathrm{on}}\right)\left(I_0 - I_{\mathrm{on}}\right)}{c_{\mathrm{on}}}\right] \tag{8}$$

### 2.3.2 低电平段建模

在 PWM 的低电平段，电流沿图 (2) 的右半部分逆时针流动，类似地，有

$$I(t)R + L\frac{dI(t)}{dt} + U_{\mathrm{bemf}} + U_{\mathrm{D}} = 0 \tag{9}$$

以初始条件 $I(0) = I_{\max}$ 解得

$$I(t) = e^{-\frac{tR}{L}}\left(I_{\max} + \frac{U_{\mathrm{bemf}} + U_{\mathrm{D}}}{R}\right) - \frac{U_{\mathrm{bemf}} + U_{\mathrm{D}}}{R} \tag{10}$$

取 $t = t_{\mathrm{off}}$，且令

$$c_{\mathrm{off}} = -\frac{tR}{L}, e_{\mathrm{off}} = e^{c_{\mathrm{off}}}, I_{\mathrm{off}} = \frac{U_{\mathrm{bemf}} + U_{\mathrm{D}}}{R} \tag{11}$$

则有

$$I_{\mathrm{f}} = e_{\mathrm{off}}\left(I_{\max} + I_{\mathrm{off}}\right) - I_{\mathrm{off}} \tag{12}$$

为计算平均电流做准备，计算 $I(t)$ 积分

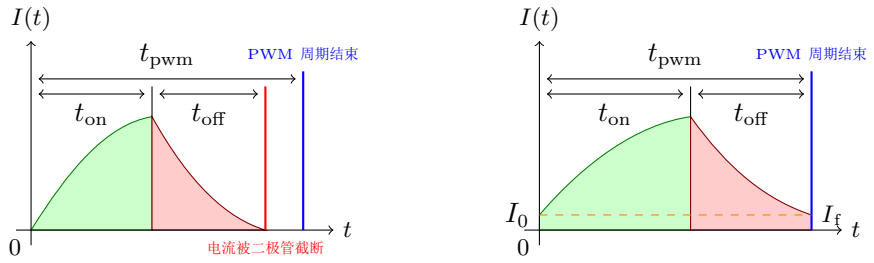
$$\int_0^{t_{\mathrm{off}}} I(t)dt = t_{\mathrm{off}}\left[I_{\mathrm{off}} - \frac{\left(1-e_{\mathrm{off}}\right)\left(I_{\max} - I_{\mathrm{off}}\right)}{c_{\mathrm{off}}}\right] \tag{13}$$

### 2.3.3 平均电流的计算

接下来我们把以上两段的模型进行合并，以计算整个 PWM 周期的平均电流。显然，基于前文计算的积分（公式 (8) 与公式 (13)），平均电流的表达式如下：

$$\begin{aligned}\overline{I} = f_{\mathrm{pwm}}\Bigg\{&t_{\mathrm{on}}\left[I_{\mathrm{on}} - \frac{\left(1-e_{\mathrm{on}}\right)\left(I_0 - I_{\mathrm{on}}\right)}{c_{\mathrm{on}}}\right] \\ &+ t_{\mathrm{off}}\left[I_{\mathrm{off}} - \frac{\left(1-e_{\mathrm{off}}\right)\left(I_{\max} - I_{\mathrm{off}}\right)}{c_{\mathrm{off}}}\right]\Bigg\}\end{aligned} \tag{14}$$

观察上式，上式中的大部分参数都已测得或可以计算，未确定的参数只有 $t_{\text{on}}$，$t_{\text{off}}$, $I_0$ 与 $I_{\text{f}}$。其中 $t_{\text{on}}$ 的计算最为简单，若占空比为 $d$，则有 $t_{\text{on}} = dt_{\text{pwm}}$，相反，$t_{\text{off}} = (1-d)t_{\text{pwm}}$ 却不总是成立，原因是因为图 (2) 中二极管的存在会截断电流。如图




(a) 电流被二极管切断　　(b) 电流连续

图 3: 电流大小在 PWM 单周期内的两种变化情况

**电流被切断的情况**　由图，此时一定有 $I_0 = I_{\text{f}} = 0$，带入式 (12) 反解得

$$t_{\text{off}} = -\frac{L}{R}\ln\left(-\frac{I_{\text{off}}}{I_{\max} - I_{\text{off}}}\right) \tag{15}$$

**电流连续的情况**　由图，此时 $t_{\text{on}} + t_{\text{off}} = t_{\text{pwm}}$，由 $I_0 = I_{\text{f}}$ 对式 (7) 和式 (12) 展开可得

$$I_0 = I_{\text{f}} = e_{\text{off}}\left[e_{\text{on}}I_0 + (1-e_{\text{on}})\,I_{\text{on}}\right] + (1-e_{\text{off}})\,I_{\text{off}} \tag{16}$$

解得

$$I_0 = I_{\text{f}} = \frac{e_{\text{off}}\left[I_{\text{off}} + (e_{\text{on}}-1)\,I_{\text{on}}\right] - I_{\text{off}}}{e_{\text{on}}e_{\text{off}} - 1} \tag{17}$$

在实际操作当中，我们先假定电流连续，解算出 $I_0$, $I_{\text{f}}$，再通过其符号判断电流是否被截断。至此，式 (14) 中的所有参数都已近有了数值，可以通过马达的转速与 PWM 占空比解算马达的电流，进而推出电机的转矩，达成该段建模目标。

## 2.4 传动部分动力学模型的建立

在完成对马达自身的建模之后，我们在马达电流转速以及小车的加速度，位置等参数之间建立动力学模型，这一模型表示了马达动力在小车底盘上传动的状况。

首先，已知理想状态下马达的转矩正比于其输出电流，这一关系使我们把电学量转化为动力学量，设

$$T = K_{\text{T}}I \tag{18}$$

其中 $T$ 为扭矩，$K_{\mathrm{T}}$ 为系数，$I$ 为上节马达模型解算出来的电流。

再接下来，假设在传动过程中轴以及车轮收到粘滞阻力和滑动摩擦的作用，令总的阻力 $f$ 为转速 $\omega$ 的线性函数：

$$f = k_{\mathrm{V}}\omega + f_{\mathrm{s}}\,\mathrm{sgn}(\omega) \tag{19}$$

其中 $k_{\mathrm{V}}$ 为粘滞阻力系数，$f_{\mathrm{s}}$ 为滑动摩擦大小（假设压力始终相同），$\mathrm{sgn}(\cdot)$ 为符号函数。则小车所受的合外力为

$$F = rT - f \tag{20}$$

其中 $r$ 为轮距。所受加速度为

$$a = \frac{F}{m} = \frac{rT - f}{m} \tag{21}$$

其中 $m$ 表示小车以及载荷的重量。对加速度积分两次即可得到小车的位置。

与上节中对马达建立的模型不同，得益于马达电路的相对固定，我们可以较为精确地对于马达自身的特性进行建模，而之后的传动部分由于未知因素过多，不便于测量，因此模型较为简单（以线性假设为主）且包含未知的，随实际场景而变化的参数，这些参数给予了这个模型较好的通用性。

## 2.5 模型的拟合与未知参数解算

在本节中，我们将用上两节建立模型对从我们小车采集而来的真实数据进行拟合，以解算在我们的模型在实验平台上未知参数的值。

在数据采集上，我们的程序以 200 Hz 的采样频率记录小车在手动操作的直线运动下连续 5 秒时间内的马达指令，转速以及传感器的位置信息。整理解码后的数据如下图：

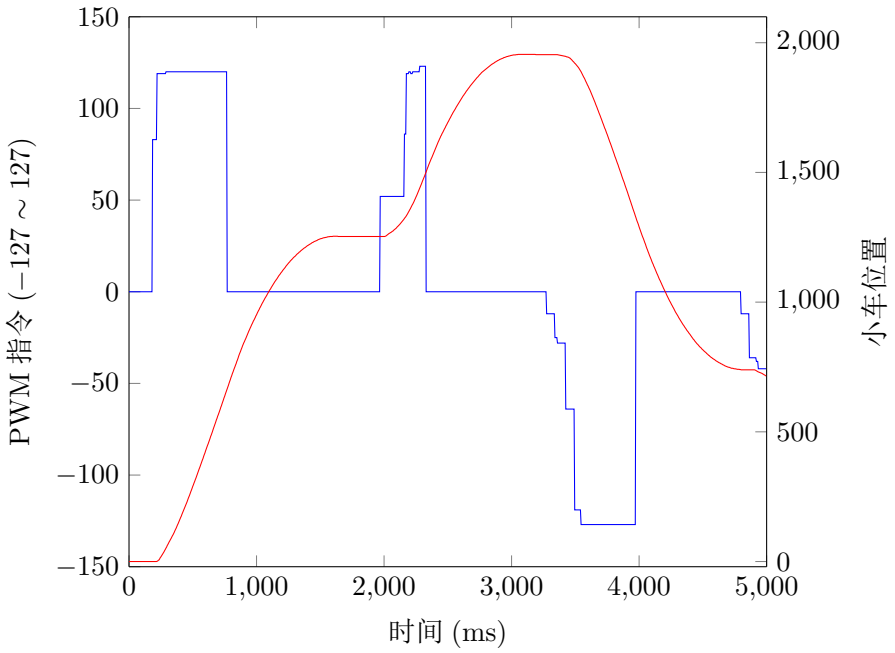



图 4: 从小车上采集到的时间-位置-指令数据

在理论上，处理以上数据较为有效的方法是：

1. 将数据中的位置信息进行微分，求出速度以及加速度。
2. 再使用节 (2.3) 中建立的马达模型结合数据当中的 PWM 指令信息解算出马达的输出电流以及输出扭矩。
3. 由于节 (2.4) 中建立的模型是线性的，可使用对加速度以及马达电流进行线性回归，从而解算模型的各个参数，也可以绘图确认建模所使用的假设是否正确。

而实际上，由于使用传感器精度原因（1°），位置数据在第一次微分之后就出现了大量噪声，难以用滤波进行消除，也无法进行第二次微分计算加速度。这迫使我们寻找其他方法将整个模型作为一个整体来进行拟合。

为了减少未知参数的数量，便于求解，我们在之后的过程中令节 (2.4) 中模型的 $k_{\mathrm{T}} = 1$ 且 $r = 1$，由于模型是线性的因此这样做只会使模型的结果线性缩放常数倍。

对于一组已知的参数，我们可以在特定的初始条件下（初速度以及位置都为 0）使用模型基于采集而来的指令数据对受控对象的位置进行离线模拟，假设 $t$ 时刻的真实位置数据为 $x_t$，模拟

得到的位置为 $y_t$，令：

$$f(x,y)=\sum_t (y_t-kx_t)^2 \tag{22}$$

表示模拟结果与现实数据之间的偏差，我们使用一个常数 $k$ 来补偿省略 $k_\mathrm{T}$, $r$ 后对模型造成的线性影响，$k$ 定义为：

$$k=\frac{\sum_t y_t}{\sum_t x_t} \tag{23}$$

则拟合以及参数的求解变为：

$$\arg\min_{k_\mathrm{V},f_\mathrm{s},m} f(x,y) \tag{24}$$

这是一个多元函数最优化问题，可以使用下一节 (2.6) 介绍的粒子群算法解决。

使用粒子群算法在种群数量为 3000，进化次数为 300 轮的情况下使用模型进行拟合的结果如下：

表 2: 使用粒子群算法拟合的位置参数值

| 参数 | 拟合值 |
|---|---|
| $m$(式 (21)) | 0.00420906 |
| $f_\mathrm{s}$(式 (19)) | 0.21199884 |
| $k_\mathrm{V}$(式 (19)) | 0.00482517 |
| $k$(式 (23)) | 15.5173 |

使用拟合后的模型在相同 PWM 指令下模拟得到的位置-时间曲线与真实的位置-时间曲线与误差如下图：

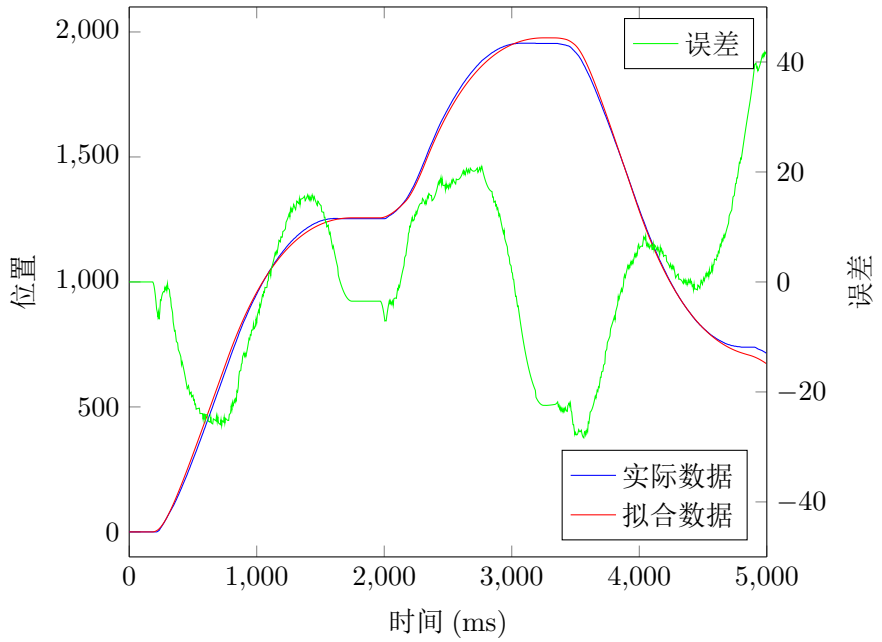



图 5: 实际数据与模型拟合数据对比

可以看到，基于节 (2.3) 与节 (2.4) 建立的模型以及本节拟合得到的参数，我们已经建立了能以较高精度模拟受控对象运行的模型。

## 2.6 粒子群优化算法

在上一节的拟合中，我们将拟合问题规约为最优化问题并进行求解，在下节的 PID 整定中，我们也将把 PID 参数的整定化为在一定指标下的最优化问题。在这两个最优化问题当中，指标函数复杂且非凸，无法使用纯数学的方法解决，因此我们都应用了名为粒子群算法的启发式进化算法进行求解。在本节当中，我们简单介绍粒子群优化算法，并说明其大致的流程与思想。

粒子群优化算法（Particle Swarm Optimization）是上世纪末由 Kennedy 与 Eberhart[6] 提出的原用于研究社会行为的算法，原算法形式化地模拟了鸟群与鱼群之类的群体移动的过程，并在经过简化之后作为一种出色的优化算法沿用至今。

基本的粒子群算法在同一时刻保存着大量备选解向量（粒子）的集合（称为粒子群），每一次迭代中，粒子群中的粒子在简单公式的指导下进行移动。具体来说，粒子移动的速度方向取决于粒子上一时刻的速度（动能），该粒子本身在移动过程中的最优解（即个体最优），与整个粒子群算法在数轮迭代下来的最优解（即全局最优），每个粒子将在每一轮迭代当中曲折接近个体最优与全局最优解，并被寄希望于在移动的过程当中找到更优的解，对于大多数问题，在数轮迭代

过后，大多数粒子都可以收敛到最优或极优解的附近，从而达成优化的目标。

基本粒子群算法的伪代码如下，首先，算法会在整个搜索空间中均匀地初始化一定数量的粒子作为粒子群，并赋予其初始速度：

**算法 1** 粒子群优化算法–初始化

**参数** 优化维度 $D$, 搜索上下界 $b_{\text{low}} < b_{\text{up}} \in \mathbb{R}^D$, 优化函数 $f : \mathbb{R}^D \mapsto \mathbb{R}$, 粒子数目 $N$

1: **function** ParticleSwarmOptimization
2: **for** $i \leftarrow 1$ **to** $N$ **do** ▷ 初始化
3: $x_i \sim U(b_{\text{low}}, b_{\text{up}})$ ▷ 在搜索空间内均匀随机初始化粒子的位置
4: $v_i \sim U(-|b_{\text{up}} - b_{\text{low}}|, |b_{\text{up}} - b_{\text{low}}|)$ ▷ 随机初始化粒子的速度
5: $p_i \leftarrow x_i$ ▷ 初始化个体最优解
6: **if** $f(x_i) < f(g)$ **then**
7: $g \leftarrow x_i$ ▷ 更新全局最优解
8: **end if**
9: **end for**

在初始化过程完毕之后，算法开始进行迭代，在每一轮迭代的开始算法将先对于每个粒子依据动量，个体与全局最优解分维度计算其速度，并更新粒子的位置。然后重新对于每一个粒子计算其函数值，并依据此更新个体最优解与全局最优解的值。

**算法 2** 粒子群优化算法–迭代

**参数** 进化轮数 $G$, 动量项 $\omega$, 全局/局部加速项 $\varphi_g/\varphi_p$

**for** $j \leftarrow 1$ **to** $G$ **do** ▷ 开始进化
  **for** $i \leftarrow 1$ **to** $N$ **do**
    **for** $d \leftarrow 1$ **to** $D$ **do** ▷ 更新粒子的速度
      $r_p, r_g \sim U(0,1)$
      $v_{i,d} \leftarrow \omega v_{i,d} + \varphi_p r_p(p_{i,d} - x_{i,d}) + \varphi_g r_g(g - x_{i,d})$
    **end for**
    $x_i \leftarrow x_i + v_i$ ▷ 更新粒子的位置
    **if** $f(x_i) < f(p_i)$ **then**
      $p_i \leftarrow x_i$ ▷ 更新局部最优解
      **if** $f(x_i) < f(g)$ **then**
        $g \leftarrow x_i$ ▷ 更新全局最优解
      **end if**
    **end if**
  **end for**
  **return** $g$
**end function**

可以看到，参数 $\omega, \varphi_g, \varphi_p$ 在粒子速度更新的时候起了至关重要的作用，其大小直接决定了算法的表现，$\varphi_p, \varphi_g$ 越大，粒子就越快接近当前最优解，算法就越快收敛，但是有可能错过更优解，而 $\omega$ 越大，粒子动量越大，算法收敛变慢，但可以覆盖更大的搜索空间。而在这些参数相同的情况下，粒子群大小越大，迭代次数越多，优化效果越好。

粒子群作为一个启发式算法，其本身并没有对待优化的问题作出任何假设或者限制，且能够在较大的搜索空间中给出快速收敛到最优或极优解。其本身不依赖于目标函数的梯度，因此相较于梯度下降与牛顿法等传统优化方法有更大的适用范围。在连续的实数优化问题上与遗传算法相比，其避免了复杂的个体编解码过程，流程更为简单直观。因此我们在本研究中选用粒子群算法作为优化算法，并使用 C++对其进行了高效的并行实现，取得了良好的效果。

## 2.7 PID 参数自整定

在前几节中，我们已经成功地建立了受控对象的模型。因此，我们已经可以在电脑上离线快速模拟受控对象的运行。在本节中，我们将使用建立的模型对于 PID 的三个参数进行自整定。

在本例中，为了使问题简单化，我们假定目标位置不变。我们使用的 PID 算法实现如下：

**算法 3** 本研究使用的 PID 控制算法

**参数** 系数 $K_p, K_i, K_d$, 循环时间 $\Delta t$, 积分阈值 $e_{\max}$, 设定目标点 $p_{\text{set}}$

**procedure** PIDControlLoop
  $e_{\text{last}} \leftarrow 0$ ▷ 初始化上一次的误差为 0
  $\Sigma e \leftarrow 0$ ▷ 初始化积分为 0
  **loop**
    $p_{\text{now}} \leftarrow$ OpticalEncoder ▷ 获取当前位置
    $e_{\text{now}} \leftarrow p_{\text{set}} - p_{\text{now}}$ ▷ 计算误差
    $\Delta e \leftarrow e_{\text{now}} - e_{\text{last}}$ ▷ 误差变化率
    **if** $|e_{\text{now}}| \le e_{\max}$ **then**
      $\Sigma e \leftarrow \Sigma e + e_{\text{now}} \Delta t$ ▷ 防止积分饱和
    **else**
      $\Sigma e \leftarrow 0$
    **end if**
    $u \leftarrow K_p e_{\text{now}} + K_i \Sigma e + K_d \frac{\Delta e}{\Delta t}$ ▷ 计算输出量
    Drive($u$) ▷ 驱动小车
    $e_{\text{last}} \leftarrow e_{\text{now}}$
    Wait($\Delta t$) ▷ 等待 $\Delta t$
  **end loop**
**end procedure**

可以看到，这是标准 PID 算法的一个变体，我们引入了积分阈值 $e_{\max}$，使算法只对连续绝对值不大于 $e_{\max}$ 的误差进行积分，避免了积分饱和造成的震荡，时滞，提升了算法的灵敏度。考虑到采集数据的量级，在本研究中，我们统一设定 $e_{\max} = 50$。同时，考虑到微控制器的运算速度与传感器的精度，设置 $\Delta t = 20\,\text{ms}$。

我们将沿用在节 (2.5) 中模型拟合的方法，对于一组特定的 PID 参数 $(K_p, K_i, K_d)$，建立衡量其优劣程度的指标函数，将参数整定问题转变为多元函数最优化问题，使用粒子群算法进行求解。

### 2.7.1 指标的选取

为了建立指标函数，我们在模拟 PID 算法的过程中额外收集马达的电流，小车的位置数据，并提出了衡量 PID 参数优劣的三个主要指标：

1. **最大超调量**：定义为 PID 算法过程中算法的最大超调量，设 $t$ 时刻小车在 $p(t)$ 位置，且 $t_0$ 为小车第一次达到设定点 $p_{\text{set}}$ 的时刻，那么超调量 $p_{\text{os}}$ 可以定义为：

$$p_{\text{os}} = \max_{t>t_0} |p(t) - p_{\text{set}}| \tag{25}$$

一般来说，超调量小或为零，说明控制算法没有过多耗费时间与能量在超过目标的折返，算法越优秀。

2. **响应时间**：定义为 PID 算法过程从运行开始到误差小于一定阈值且误差变化量小于一定阈值（即系统稳定）的时间。如设这两个阈值为 $\varepsilon_{\text{e}}$，$\varepsilon_{\text{ec}}$，则响应时间表示为：

$$t_{\text{res}} = \min \left\{ t \,\middle|\, |p(t) - p_{\text{set}}| \le \varepsilon_{\text{e}}, \left| \frac{dp(t)}{dt} \right| \le \varepsilon_{\text{ec}} \right\} \tag{26}$$

响应时间越短，PID 算法越优秀。

3. **累积电荷量**：定义累积电荷量的表达式为：

$$Q_{\text{acc}} = \int_0^{t_{\text{res}}} |I(t)| dt \tag{27}$$

其中 $I(t)$ 为通过节 (2.3) 中模型计算出来的 $t$ 时刻的电流。

过大的电流会带来较高的发热，过大的瞬时电流更是会导致对马达内部结构的机械性损坏（如打齿）。因此，一个优秀的 PID 算法应该在前两个指标优秀的情况下使过程中通过马达的累积电荷量尽可能小，从而减少对系统的损耗。同时，小的累积电荷量可以说明算法更节省能源，更“温和”。

可以看到，这三个指标都是非常直观的，同时这三个指标的计算并不需要额外的代价，可以简单地嵌入算法 (3) 的步骤当中。

在接下来的整定中，取 $p_{\text{set}} = 1000$，$\varepsilon_{\text{e}} = 10$, $\varepsilon_{\text{ec}} = 5$。

### 2.7.2 优化求解

基于上一小节建立的三个指标，PID 参数的整定已经转化为了三个指标的最优化问题，需要通过启发式最优化算法进行求解。

节 (2.6) 基本的粒子群算法只能处理单目标优化的情形，因此我们首先试图将多个优化目标合并为一个元优化目标进行求解，一般来说，常用的做法是取优化目标的加权平均值作为总优化目标，即：

$$f(K_p, K_i, K_d) = w_{\text{os}} p_{\text{os}} + w_{\text{res}} t_{\text{res}} + w_{\text{acc}} Q_{\text{acc}} \tag{28}$$

但是，由于三个指标的量纲各不相同（分别为距离，时间，电荷量，且数量级也各有差异），因此在权值的选取时仍然需要依靠调参人员对于系统的经验。即我们虽然实现了三个参数的自整定，但是又额外引入了更多的元参数，这与我们自整定的初衷是相违背的。

同时，我们观察到，我们提出的三个指标并不是同时可以取到最小的，例如，响应时间小意味着算法快速，累积电荷量少意味着算法耗能少，比较温和，而最快速的算法必然包含了急停等需要大电流的过程，而温和的算法需要的时间可能较长，即鱼和熊掌不可兼得。这说明在三个指标下，两组 PID 参数之间未必存在严格的优于/劣于关系。

因此，我们选择使用 Coello 提出的 MOPSO 算法 [7]（即多目标粒子群算法），该算法是节 (2.6) 的粒子群算法的一个拓展。MOPSO 算法的运行结果不是一组最优解，而是问题的 Pareto 最优边界。

Pareto 最优边界定义为一组解的集合，其中对于任意两组解，它们当中任何一个都至少有一个指标优于对方，即两两之间没有严格的优劣关系。与之相对的称为 Pareto 占优，即两组解之间存在一组解，其任何指标都优于另一组解，$A$Pareto 占优（严格优于）$B$ 记作：

$$A \preceq B \Leftrightarrow \begin{cases} \forall i \in \{1 \cdots k\}, f(A)_i \leq f(B)_i \\ \exists i \in \{1 \cdots k\}, f(A)_i < f(B)_i \end{cases} \tag{29}$$

其中 $k$ 为指标数

为了使粒子群算法能够返回一个 Pareto 最优边界，我们拓展算法当中“全局最优解”的概念，将其从维护一个解转化为维护一个一定大小的存档，是其在任何时刻保存的一组解都构成 Pareto 边界。每一次迭代过后，我们从粒子群中选出不被 Pareto 占优的粒子加入存档。并从中弹出存档中被该粒子 Pareto 占优的粒子。其算法示例如算法 4所示。

**算法 4** 将新粒子 $p$ 加入存档

```
1: procedure PushArchive(p)
2:     for i ← 1 to N do
3:         if x_i ⪯ p then
4:             return                          ▷ 粒子群中有比 p 严格更优的
5:         end if
6:     end for
7:     for q ∈ A do
8:         if q ⪯ p then
9:             return                          ▷ 存档中有比 p 严格更优的
10:        else if p ⪯ q then
11:            Pop(A, q)                       ▷ 弹出存档中比 p 严格更劣的
12:        end if
13:    end for
14:    Push(A, p)                              ▷ 加入存档
15: end procedure
```

同时，在存档内部维护一个以存档中解的指标向量作为坐标的，具有固定大小的自适应网格系统，维护每一个网格中解的数量，称为该网格的密度。在迭代更新粒子位置需要一个全局优解作为指导时，优先从低密度的网格当中选择。当存档容量过大时，优先从高密度的网格中弹出多余的解。以上的选择可以通过轮盘赌算法完成。这样的选择策略鼓励粒子靠近存档中更加“稀缺”的优解，有助于拓展解的多样性。

我们同时采用了 MOPSO 论文 [7] 当中提及的变异算子，进一步增加了解的多样性。

粒子群算法给出的 Pareto 最优边界如下。

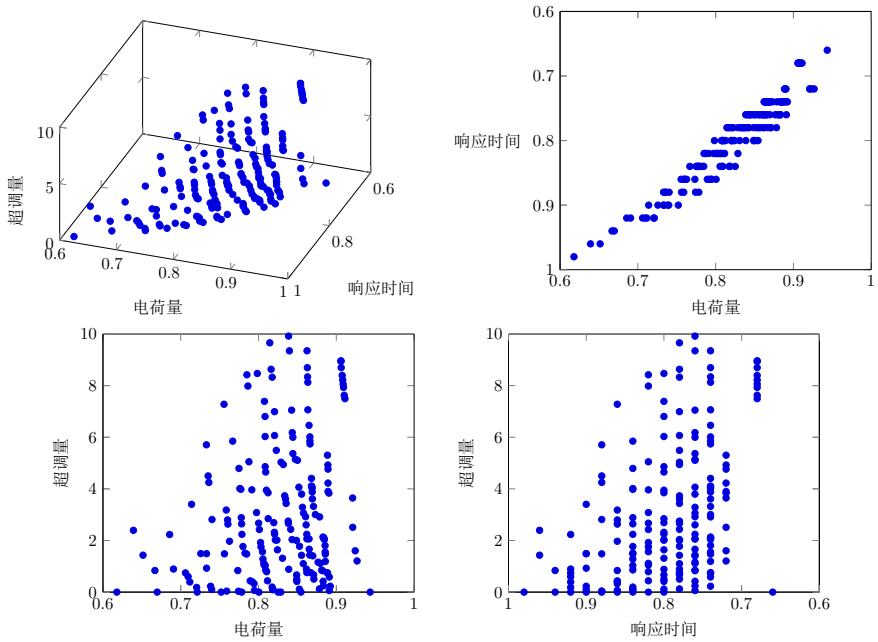



图 6: PID 参数在三个指标下的 Pareto 最优边界

从上图的响应时间-电荷量图可以看出，确实响应时间越短，PID 算法越快，算法就越“粗暴”，累积的电荷量越大。这时就需要在速度与损耗之间做出权衡。而从下方的两图可以观察到，超调量似乎无论电荷量与响应时间的取值都可以取到 0，而就分布而言，算法越“暴力”，越快速，越容易产生一定的“超调”。这与我们的直觉是相一致的。

我们接着选取 Pareto 最优边界背后的 PID 参数，观察它们之间内在的联系：

表 3: 整定得到的部分解的指标值与参数值

| 响应时间 | 电荷量 | 超调量 | $K_p$ | $K_d$ |
|---|---|---|---|---|
| 0.66 | 0.943485 | 0 | 4.309063 | 0.156651 |
| 0.74 | 0.883842 | 2.082238 | 1.969935 | 0.083134 |
| 0.78 | 0.814482 | 9.658462 | 0.759525 | 0.036789 |
| 0.98 | 0.617873 | 0 | 0.342893 | 0.040670 |

上表中不加列 $K_i$ 一项的原因是因为由于积分阈值的存在，$K_i$ 对于整个系统的影响相较 $K_p$,$K_i$ 要小很多，不够典型。

上表中最快与最慢算法的位置-时间图像如下：

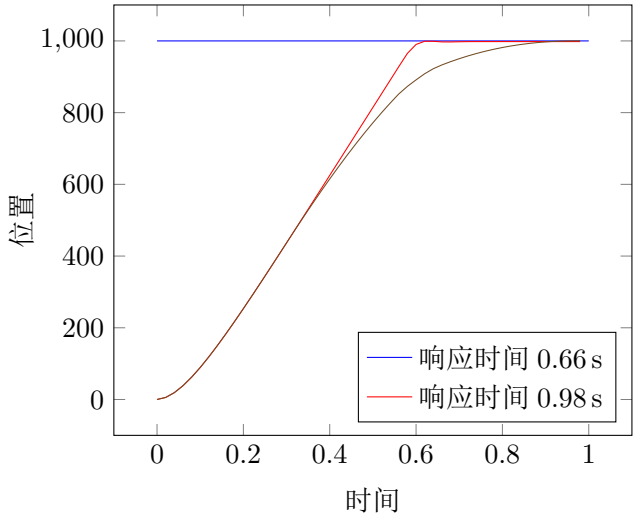



图 7: 最快与较慢 PID 参数的位置-时间图像

总体来看，随着响应时间越短，算法运行越快，$K_p$ 和 $K_d$ 就越大，以“P 油门，D 刹车”的口诀来看，就是“油门”与“刹车”踩得越猛, 算法越快，耗能越多，反之亦然，这同样与经验是一致的。可以说，粒子群算法给出的 Pareto 边界使用数据很好地证明了经验口诀的正确性，又在同时给出了具体的参数。从模拟上看，PID 自整定的算法是成功的。

### 2.7.3 实际验证

上一小节当中，我们使用多目标粒子群优化算法求解了 PID 参数的 Pareto 最优边界，并对其进行了分析。而分析中用到的 PID 参数指标全部都是基于对受控对象的离线模拟得出的预期值，对于整定出来的 PID 参数在受控对象上的实际表现如何仍然未进行试验。在本小节中，我们将整定得出的 PID 参数应用在受控对象四轮小车的实际运行当中，并观察其运行情况。

我们从上一节的 Pareto 最优边界中随机选取一组参数进行实验，选取的参数以及离线模拟得出的指标如下：

表 4: 用做试验的 PID 参数及其预计指标

| | |
|---|---|
| $K_p$ | 0.438823 |
| $K_i$ | 2.53019 |
| $K_d$ | 0.0292212 |
| 响应时间 | 0.84 s |
| 累积电荷量 | 0.80029 |
| 最大超调量 | 8.492539 |

我们将参数及使用 PID 算法 (算法 (3)) 的程序传入小车当中，进行运行，采集运行过程中的 PWM 指令，位置等数据，与离线模拟解算出的预计 PWM 指令，位置进行对比。为进一步作为参考，我们还加入模型在实验中的实际 PWM 指令下预测的位置信息，一并制成下图：

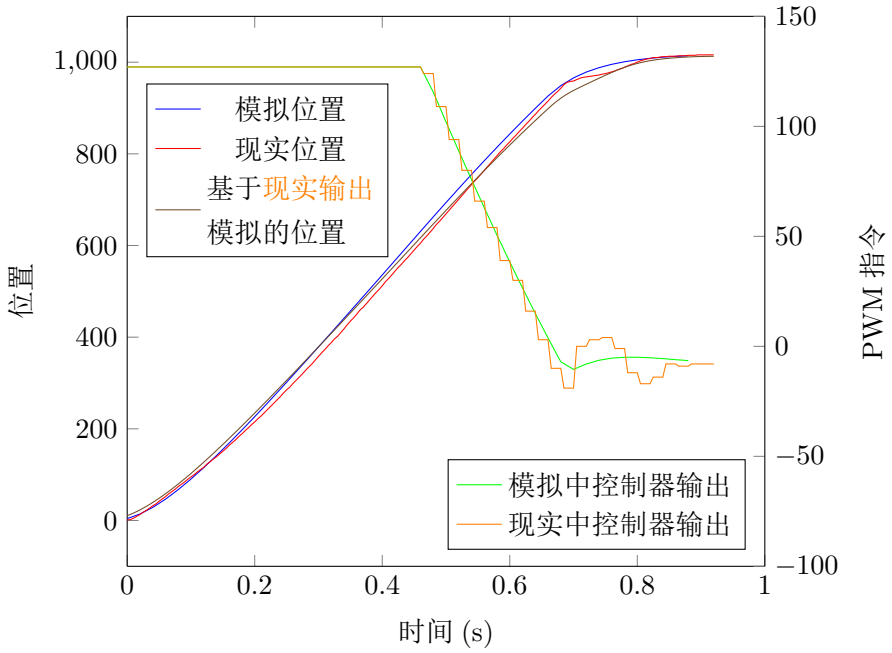



图 8: 现实试验中的算法表现与预测表现对比

观察上图，我们发现模型模拟 PID 算法过程得到的预测位置，基于现实输出模拟的预测位置以及现实中算法运行的位置轨迹的图像非常接近，几乎重合。这首先证明了我们建立的模型的准确性。更为重要的是，在实际试验中，受控对象以表 (4) 为参数的 PID 控制下无超调且稳定平缓地达到预设目标点，指标与离线模拟预测得到的一致，说明我们建立的 PID 自整定算法取得

了成功，是实用的，可操作的。

同时，我们观察到现实中控制器的输出在特定段内偏近阶梯状下降而非模型预测的连续下降，导致位置轨迹在 0.68 s 左右时略微滞后于预期并在 0.74 s 略微超前。这是由于传感器精度与时间粒度的限制，导致 PID 算法计算微分也是离散，不连续的整数值，致使现实中控制器的输出不连续。这是我们建模之初没有考虑到的。这可以通过改进微分算法，亦或是在离线模拟 PID 时加入取整解决。

# 3 结论

## 3.1 总结

本文首先通过对受控对象进行简单建模并使用粒子群算法进行缺失参数的求取，结果表明建立的模型可以在相当的精度内对受控对象进行仿真，说明建立的模型的正确性。后通过模型完成对受控对象的离线模拟，并与 PID 控制算法进行结合设计了 PID 算法运行效果的三个量化指标：超调量，响应时间与累积电荷量，并使用改进后的粒子群算法求得在多目标下 PID 参数的 Pareto 最优边界与对应的整定参数值。在实际验证过程中，我们算法的表现与离线预测的相一致，符合预期，证明了我们算法的正确性，实用性与可操作性。

在节 (2.4) 中，我们建立的动力学模型并没有对被控对象的实际结构做出过多的假设，因此该方法也可以稍加改动快速应用到其他结构的类似被控对象上（而非局限于小车）。

同时，在节 (2.7) 中，我们建立的三个指标也没有依赖于 PID 算法自身的特点，因此，只要稍加改动就可以实现其他控制器在电脑上的离线模拟，并用一样的指标与一样的粒子群优化算法完成其参数的自整定，这体现了我们方法极大的灵活性与可拓展性，是依赖于状态空间，传递函数的传统整定方法所难以达到的。

总体而言，我们的算法从推广调参过程中“试错”的过程入手，以离线模拟代替在线试错，以量化指标代替试验当中的人工观察，以启发式的优化算法代替试错之后的参数调优。因为“试错”这一调参的方式能应用于大多数的受控对象与控制算法。因此作为“试错”的自动化拓展，我们能够比基于控制理论的，针对某一模型/控制器设计整定算法更为灵活，适用于更多的情境。

## 3.2 展望

在节 (2.4) 与节 (2.3) 中，我们在建模当中忽略了实际受控对象运行过程中的噪声，使得后文所依赖的离线模拟是在较为理想的情况下进行的，导致整定得到的 PID 参数可能在鲁棒性上有所欠缺。如果有更多时间，我们期望能够探究在我们算法中引入噪声的方式以及对应指标的设计，增强我们算法给出参数的鲁棒性。

在节 (2.7) 中，我们假定 PID 的目标设定点不变，而很多情况下 PID 控制器的任务是在变目标情况下的轨迹跟踪，这是我们的算法所没有考虑到的。如果有更多的时间，我们将预期就动态目标/多目标情况下指标的设计与自整定进行探究。

### 3.3 致谢

最后，感谢上外附中和中科院技物所的老师们给我提供了这样一个体验科研的平台，在进行本课题的过程中中科院技物所郑伟波老师和袁永春老师给了我诸多指导，让我获益良多。同时我也要特别感谢虹口区青少年活动中心的饶志刚老师，他在实验平台的搭建与试验用的场地上给予了我极大的帮助，使得本课题可以顺利进行。

## 参考文献

[1] Ziegler, J.G & Nichols, N. B. (1942). Optimum settings for automatic controllers[J]. Transactions of the ASME. 64: 759-768.

[2] 孙跃光, 林怀蔚, 周华茂, 杨小玲. 基于临界比例度法整定 PID 控制器参数的仿真研究 [J]. 现代电子技术, 2012, 08, 192-194.

[3] 廖芳芳, 肖建. 基于 BP 神经网络 PID 参数自整定的研究 [J]. 系统仿真学报, 2005, 5, 1711-1713.

[4] A. Visioli. Tuning of PID controllers with fuzzy logic[J]. IEEE Proceedings - Control Theory and Applications, 2001, 148, 1-8.

[5] Pearman (2012), Derivation of formulas to estimate H bridge controller current[OL]. `https://vamfun.wordpress.com/2012/07/21/derivation-of-formulas-to-estimate-h-bridge-controller-current-vex-jaguarvictor-draft/`.

[6] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization[C]. Proceedings of ICNN'95 - International Conference on Neural Networks. doi:10.1109/icnn.1995.488968

[7] Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization[J]. IEEE Transactions on Evolutionary Computation, 8(3), 256 – 279. doi:10.1109/tevc.2004.826067