Apollo纵向控制模块梳理

1 PID控制器

PID主程序流程

1. 判断步长是否小于零，若小于零则警告，并返回上一输出`previous_output_`；
2. 判断是否是`first_hit_`，若是则将`first_hist_`置为false，否则计算

$$diff = (error - error_{previous})/dt$$

3. 判断积分器使能`integrator_enabled_`，假则将`integral_`置零，真则计算`integral_`，在积分前应用 Ki 以避免在稳态时改变 Ki 时出现阶跃；

$$integral+ = error * dt * ki$$

4. 积分幅值判断：高于积分饱和上限，则等于积分饱和上限，并将积分饱和状态置1；低于积分饱和下限，则等于积分饱和下限，并将积分饱和状态置-1；其余情况积分饱和状态置0；
5. 保留这一时刻误差，`previous_error_ = error`；输出u幅值判断：高于输出饱和上限，则等于输出饱和上限，并将输出饱和状态置1；低于输出饱和下限，则等于输出饱和下限，并将输出饱和状态置-1；其余情况输出饱和状态置0；
6. 计算输出值，如下

$$output = error * kp + integral + diff * kd$$

7. 保留这一时刻输出，`previous_output_ = output`

2 油门刹车标定表

在Apollo系统中，控制模块会请求加速度量值。通过车辆标定表，控制模块便能找到准确产生所需加速度量值对应的油门、刹车踏板开合度控制命令，之后下发给车辆底盘。车辆标定表提供一个描述车辆速度、油门／刹车踏板开合度、加速度量之间关系的映射表。油门刹车标定过程便是生成车辆标定表的过程。

3 lead/lag补偿器

3.1 leg/leg补偿器及其参数

$$H(s) = \beta\frac{\tau s + 1}{\tau\alpha s + 1}$$

其中，α为滞后系数；β为超前系数；τ为给定的时间系数

采用双线性变换，T为采样周期

$$s = \frac{2}{T}\frac{z-1}{z+1}$$

带入s进入传递函数，可得

$$H(z) = \beta\frac{2\tau(z-1) + T(z+1)}{2\tau\alpha(z-1) + T(z+1)}$$

$$H(z) = \frac{(T\beta - 2\beta\tau) + (2\beta\tau + T\beta)z}{(T - 2\alpha\tau) + (2\alpha\tau + T)z}$$

$$H(z) = \frac{kn_0 + kn_1 * z}{kd_0 + kd_1 * z}$$

对应程序中的代码，如下

$$a1 = \alpha\tau, a0 = 1.00,\ b1 = \beta\tau, b0 = \beta;$$

$$kn1 = 2\beta\tau + T\beta, kn0 = T\beta - 2\beta\tau, kd_1 = 2\alpha\tau + T,\ kd0 = T - 2\alpha\tau$$

```
void LeadlagController::TransformC2d(const double dt) {
  if (dt <= 0.0) {
    AWARN << "dt <= 0, continuous-discrete transformation failed, dt: " << dt;
    transfromc2d_enabled_ = false;
```

```
  } else {
    // 公式?
    double a1 = alpha_ * tau_;
    double a0 = 1.00;
    double b1 = beta_ * tau_;
    double b0 = beta_;
    Ts_ = dt;
    // 带入默认参数数值后，dt，dt，dt，dt
    kn1_ = 2 * b1 + Ts_ * b0;
    kn0_ = Ts_ * b0 - 2 * b1;
    kd1_ = 2 * a1 + Ts_ * a0;
    kd0_ = Ts_ * a0 - 2 * a1;
    if (kd1_ <= 0.0) {
      AWARN << "kd1 <= 0, continuous-discrete transformation failed, kd1: "
            << kd1_;
      transfromc2d_enabled_ = false;
    } else {
      transfromc2d_enabled_ = true;
    }
  }
}
```

3.2 lead/leg补偿器原理

$lead/lag$主要是超前或滞后(不能同时)，本质就是讨论零点 $\frac{1}{\tau}$和极点 $\frac{1}{\alpha\tau}$ 的前后位置，如下：

- 若零点$\frac{1}{\tau}$极点$\frac{1}{\alpha\tau}$，即 $\alpha < 1$，此时为超前校正，作用是提高响应速度，避免引入高频震荡；
- 若零点 $\frac{1}{\tau}$ <极点 $\frac{1}{\alpha\tau}$，即 $\alpha > 1$ ，此时为滞后校正，作用是提高稳态精度，减少稳态误差，但暂态响应将变慢

关于超前/滞后补偿器的原理分析推荐:

滞后补偿器：https://www.bilibili.com/video/BV1W7411n7G8?spm_id_from=333.337.search-card.all.click

超前补偿器：https://www.bilibili.com/video/BV1JJ411i7ph?spm_id_from=333.999.0.0

对于增益系数 β，本质上就是讨论$lead/lag$模块串入系统后对系统开环放大系数的影响，有如下情况：

- 若$0 < \beta < 1$，会使得系统的开环放大系数下降，即幅值衰减；
- 若$\beta > 1$，会使得系统的开环放大系数上升，即幅值增加

增益系数 β 的分析: https://www.bilibili.com/video/BV14J411A7M2?spm_id_from=333.999.0.0

3.3 lead/leg补偿器程序流程

1. 判断连续转离散(`transfromc2d_enabled_`)是否成功，失败则重新进行`TransformC2d`，如果再次失败，则发出警告`C2d transform failed; will work as a unity compensator`，并返回错误；
2. 检查步长`dt`是否小于等于零。如果小于等于零，则发出警告`dt <= 0, will use the last output`，返回上一时刻输出`previous_output_`；
3. 计算内部状态`innerstate_`，计算公式如下

$$s_{inner}(k) = \frac{e(k) - s_{inner}(k-1) * kd_0}{kd_1}$$

移位可得公式

$$s_{inner}(k-1) * kd_0 + s_{inner}(k) * kd_1 = e(k)$$

上述公式不能直观理解，需要进行转换

观察离散传递函数，如下

$$\frac{u(z)}{y(z)} = \frac{kn_0 + kn_1 z}{kd_0 + kd_1 z}$$

$$y(z) = \frac{kn_0 + kn_1 z}{kd_0 + kd_1 z} u(z)$$

令

$$x(z) = \frac{1}{kd_0 + kd_1 z} u(z)$$

即

$$kd_0 x(k) + kd_1 x(k+1) = u(k)$$

$$x(k+1)=\frac{u(k)-kd_0x(k)}{kd_1}$$

传递函数变为

$$y(z)=(kn_0+kn_1z)x(z)$$

从而得到

$$y(k)=kn_0x(k)+kn_1x(k+1)$$

其中，$y(k)$为$output$，$x(k)$为$innerstate$，$u(k)$为$error$(即系统输出与参考值的偏差)，并按照$x(k+1)=\frac{u(k)-kd_0x(k)}{kd_1}$计算$innerstate$的值

4. 进行`innerstate`幅值判断：高于状态饱和上限，则等于状态饱和上限，并将状态饱和状态置1；低于状态饱和下限，则等于状态饱和下限，并将状态饱和状态置-1；其余情况状态饱和状态置0；
5. 计算`output`
6. 保存`innerstate`和`output`变量，即为`previous_innerstate`和`previous_output`