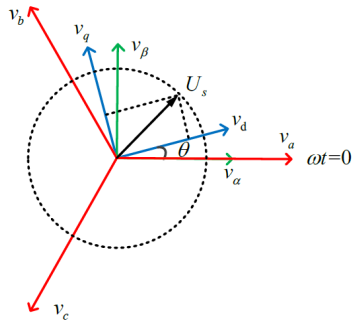


云台控制 -- Yaw轴系统辨识与控制器设计

1.云台电机

在了解云台系统之前，我们得先弄清楚手头上的云台伺服电机的运作机理，这一点是我们建模与控制的基础。

直流无刷电机(BLDC)模型：



$$\begin{cases} u_d = R_s i_d + p L_d i_d - \omega_e L_q i_q \\ u_q = R_s i_q + p L_q i_q + \omega_e L_d i_d + \omega_e \psi_f \end{cases}$$
$$T_e = \frac{3}{2} p_n [\psi_f i_q + (L_d - L_q) i_d i_q]$$

https://blog.csdn.net/qq_41478484

上式是BLDC的dq轴电压方程以及力矩的方程。我们在后续的建模分析中，都是以力矩作为输入为基础的。因此，我们得知，我们是如何控制电机力矩大小的。

1.1 6020云台电机

6020作为一款伺服电机，其有关力矩电压或力矩电流的控制已通过电调帮我们做好了。

使用

电机支持 CAN、PWM 两种控制模式，并且可以识别两种控制信号自动切换模式。此外，电机还可以通过 USB 转串口工具接入电脑，实现参数配置和固件升级。

在 CAN 控制模式下，驱动器根据用户的输入的目标值对转矩电压或转矩电流进行闭环控制。用户可以根据电机反馈的速度、位置等信息，在外部实现速度和位置闭环控制。

电机升级最新固件（版本 $\geq 1.0.11.2$ ）后，您可以通过 RoboMaster Assistant 调参软件（版本 ≥ 2.7 ）在参数设置选项中打开电流环开关以对电机进行转矩电流控制。

在 PWM 控制模式下，电机支持速度和位置两种闭环控制模式，用户可以根据选择进行切换。

（注：文中提到的“扭矩、力矩、转矩”是同一个意思）

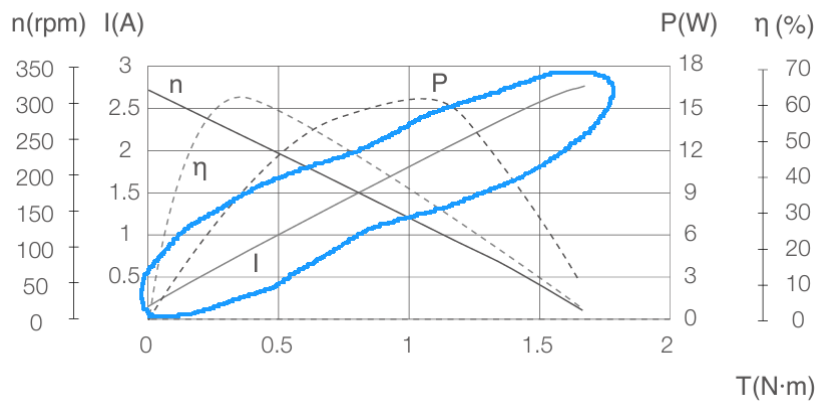
而我们目前需要弄清楚的一点是，这个力矩电压或力矩电流的闭环控制，跟实际的电机力矩存在些什么关系？

首先，这里所说的“力矩电压、力矩电流”指的就是上式电压平衡方程中的 U_q 和 I_q 。而对于一款伺服电机驱动器（电调）而言，它的控制策略是将 I_d 控制为0， I_q 或 U_q 由用户发值指定。因此，上边的力矩方程可变为：

$$T_e = \frac{2}{3} * p n * \Psi_f * i_q$$

（其中， $p n$ 为电机的极对数， Ψ_f 为电机转子的磁链常数）

因此，我们不难看出，**力矩 T_e 是正比于力矩电流 I_q 的**，这一点在6020说明书后面的电机特性曲线也可以佐证：



由于， U_q 并非正比于 I_q ，甚至是非线性的关系（若是电调中有做前馈解耦的操作，那么 U_q 跟 I_q 就是一阶线性关系，因此 U_q 跟力矩最多是个一阶线性关系），所以用 U_q 控制力矩显然不如用 I_q 控制力矩来的直接。

1.2 云台系统模型

Yaw轴云台的转角关于输入力矩是一个二阶的模型

$$J * \ddot{\theta} + b * \dot{\theta} = T$$

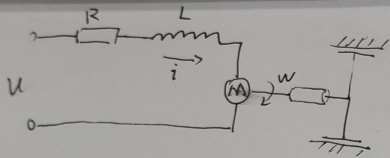
而Yaw轴云台的角速度关于输入力矩就是一个一阶的模型（大家记住这条微分方程，待会建模要用到）

$$J * \dot{\omega} + b * \omega = T$$

（其中 J 是云台转动惯量， b 是阻尼系数，这二者可认为是常数，而 T 是我们控制输入的力矩）

由于我们的控制的 I_q 正比于 T ，所以我们电控发的值与Yaw轴云台的旋转角度毫无疑问是一个二阶的关系。

扩展补充：若我们使用力矩电压模式（控制 U_q ），那么电控发的值关于Yaw轴云台的转角就是一个三阶的关系，证明如下：



取状态向量 $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} i \\ \theta \\ \dot{\theta} \end{bmatrix}$ 输入 $\vec{u} = u$.

$$\begin{cases} u = iR + L \cdot i' + k_b \cdot \omega \\ J \ddot{\theta} + b \dot{\theta} = i \cdot k_a \end{cases}$$

即
$$\begin{cases} \dot{x}_1 = -\frac{R}{L} x_1 - \frac{k_b}{L} x_3 + \frac{1}{L} u \\ \dot{x}_2 = x_3 \\ \dot{x}_3 = \frac{k_a}{J} x_1 - \frac{b}{J} x_3 \end{cases}$$

写成矩阵
$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & 0 & -\frac{k_b}{L} \\ 0 & 1 & 0 \\ \frac{k_a}{J} & 0 & -\frac{b}{J} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \\ 0 \end{bmatrix} u$$

取输出为 θ , $y = [0, 1, 0] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

故
$$A = \begin{bmatrix} -\frac{R}{L} & 0 & -\frac{k_b}{L} \\ 0 & 1 & 0 \\ \frac{k_a}{J} & 0 & -\frac{b}{J} \end{bmatrix} \quad B = \begin{bmatrix} \frac{1}{L} \\ 0 \\ 0 \end{bmatrix}$$

$$C = [0, 1, 0] \quad D = 0$$

$\therefore H(s) = (C \cdot sI - A)^{-1} \cdot B + D$

由 matlab 运算

matlab运行结果显示:

Hs =

$$K_b / (K_a \cdot K_b \cdot s + R \cdot b \cdot s + J \cdot L \cdot s^3 + J \cdot R \cdot s^2 + L \cdot b \cdot s^2)$$

2.云台系统辨识

在进入正题之前，我们先弄明白一个问题：**为什么我们要做云台系统的辨识？**

原因很简单，因为我们想得到系统的传递函数。有了系统传递函数，我们就可以设计控制器来控制系统表现。

2.1 系统建模

还记得刚才那个力矩关于云台转速的一阶微分方程吗，我们接下来就拿它来做文章。

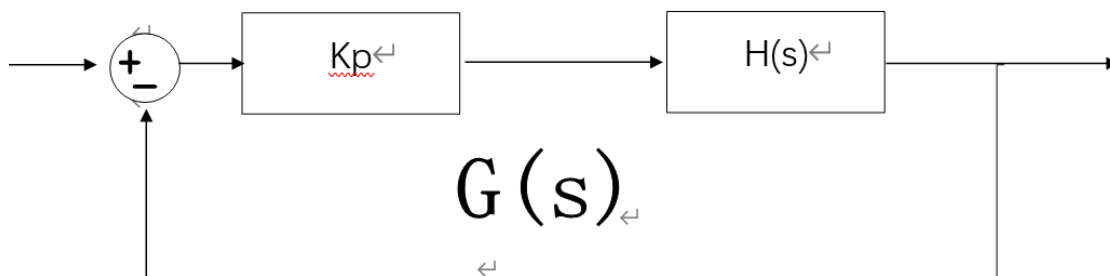
对微分方程

$$J \cdot \dot{\omega} + b \cdot \omega = T$$

做拉普拉斯变换，得到输入、输出传递函数：（这个传递函数是速度开环传递函数）

$$H(s) = \frac{1}{(J*s + b)}$$

那么我们对其进行如下的闭环控制，并称整个闭环传递函数为G(s)：



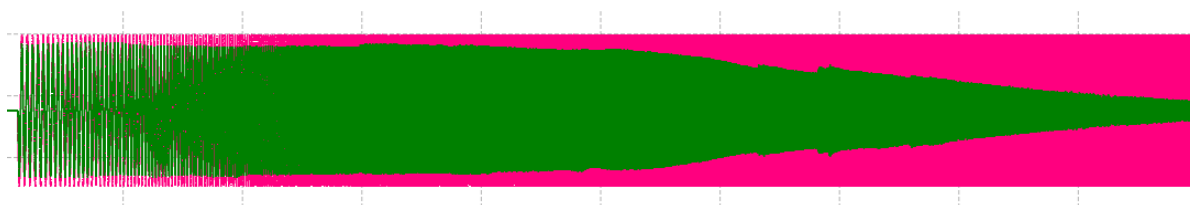
$$\text{有 } G(s) = \frac{Kp * H(s)}{1 + Kp * H(s)} = \frac{Kp}{J * s + b + Kp}$$

由此可见，仅有Kp控制的速度闭环G(s)是一个一阶（1极点、0零点）的系统。

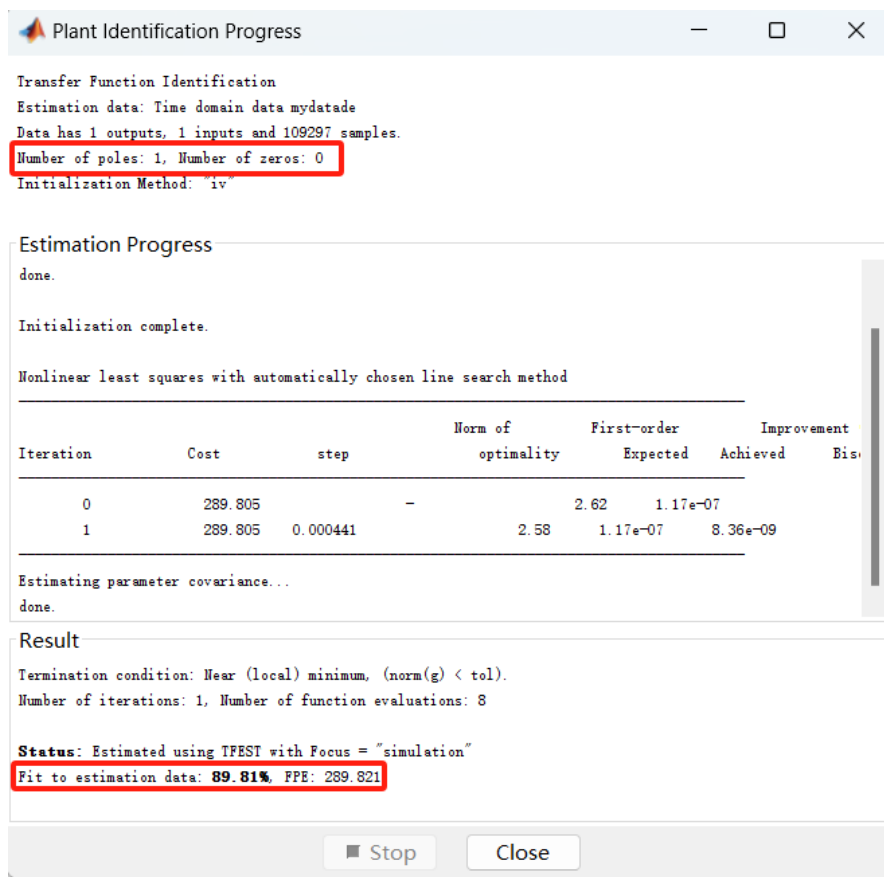
2.2 扫频辨识系统

实验所用电机为电流模式的GM6020云台电机。

我们给速度环输入扫频激励信号（1~80Hz 线性扫频，每个频点重复5个周期），得到的速度输出响应如下图所示：



我们将结果导入到matlab，并利用System Identification工具箱对输入输出数据进行系统辨识，选择辨识系统为1阶系统（1极点、0 零点），辨识结果如下所示：



61.36

辨识的系统传递函数: $G(s) = \frac{\text{-----}}{s + 74.39}$

结果显示，速度环系统的辨识度为89.81%，有较好的辨识度（一般认为辨识度大于80%就为良好）。因此，我们便可基于上述结果 设计前馈环节。

看到这里，可能你们会问，为什么不直接辨识云台的开环系统而是选择辨识云台的速度闭环系统？

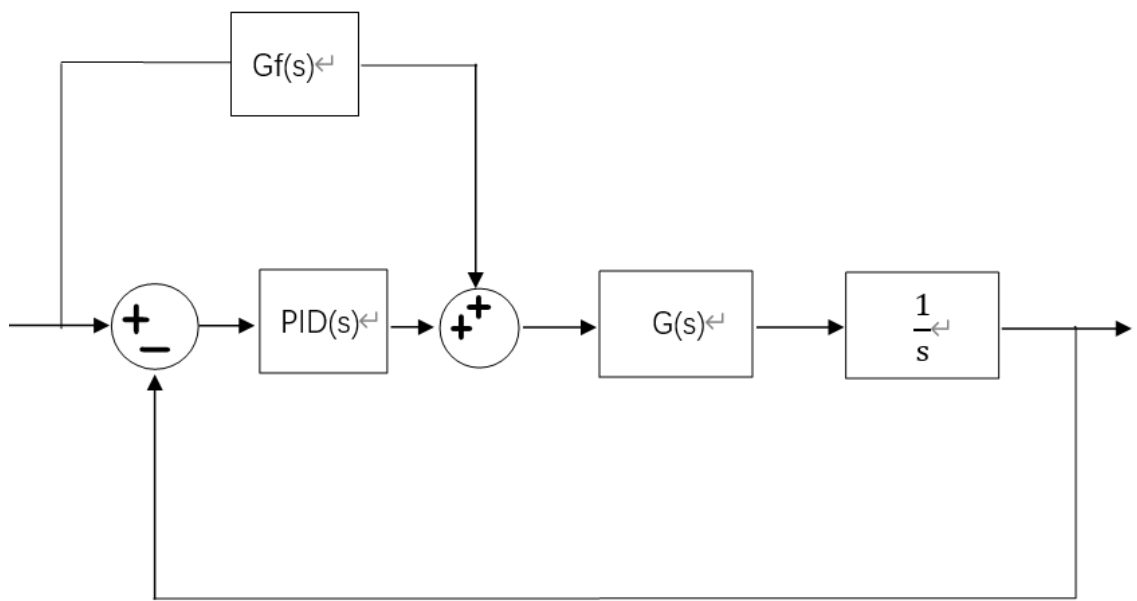
原因有两点：

其一：云台开环系统的激励信号无法确知，很可能你给了一个正弦的电流激励，但实际上由于电机运动方程的约束，其实际的力矩 电流并非等于你所给的激励信号。

其二：速度闭环的带宽要宽于速度开环的带宽，这样更有利于辨识。

2.3 前馈设计

我们最终控制的是电机的位置环，根据双环PID的控制框图，我们可以基于给定的角度信号设计前馈，总的控制框图如下所示：

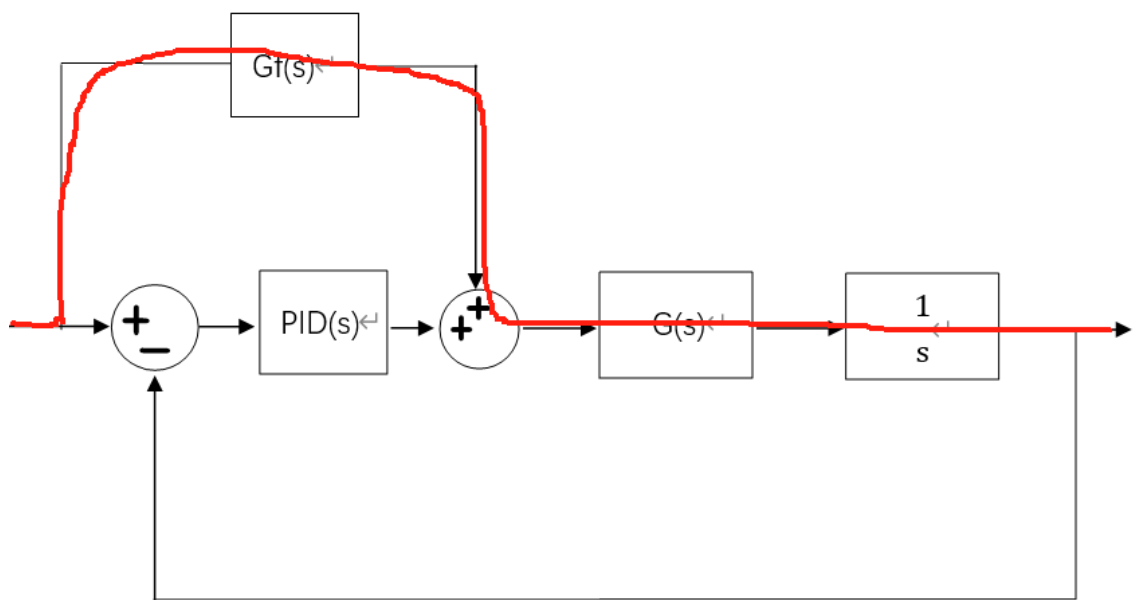


其中 $G(s)$ 为速度环， $PID(s)$ 为外环PID对应的传递函数， $G_f(s)$ 是前馈传递函数。

前馈 $G_f(s)$ 设计原理

我们令 $G_f(s) = \frac{1}{G(s) * \frac{1}{s}}$ ，则标红色的这条支路中，传递函数为

$G_f(s) * G(s) * \frac{1}{s} = 1$ ，这就意味着上边的支路成了一根导线，所有的角度输出将立马等于期望角度的输入。



当然，这是理想状态下的前馈，实际上，我们的电机受物理特性的影响，以及速度环的离散化控制的影响， $G(s)$ 并非完全为线性系统，同时 $G(s)$ 辨识的误差也会导致前馈设计的不完美。尽管如此，前馈的引入也能给系统带来响应速度的提升。

实车验证

我们对步兵Yaw轴云台进行速度环辨识，按照上述方法得到前馈 $G_f(s)$ ，用脉冲不变法将其转换为差分方程，并转换为C代码：

```
float FFC_OUT(float x_n)
{
    static float ts = 0.003;
    static float x_n_1 = 0.0f;
    static float x_n_2 = 0.0f;
    static float a1 = 96.89f/89.38f;
    static float a2 = 1/89.38f;
    float x_dot_2 = 0.0f;
    float x_dot_1 = 0.0f;
    float y_n;

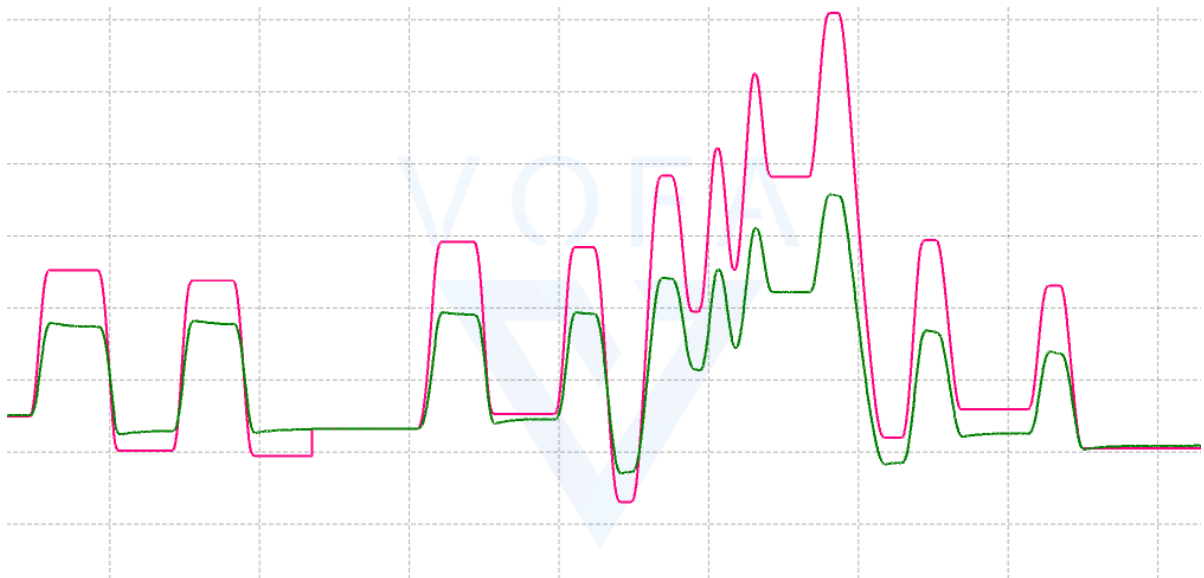
    x_dot_1 = (x_n - x_n_1)/ts;
    x_dot_2 = (x_n - 2*x_n_1 + x_n_2)/(ts*ts);

    y_n = a1*x_dot_1 + a2*x_dot_2;

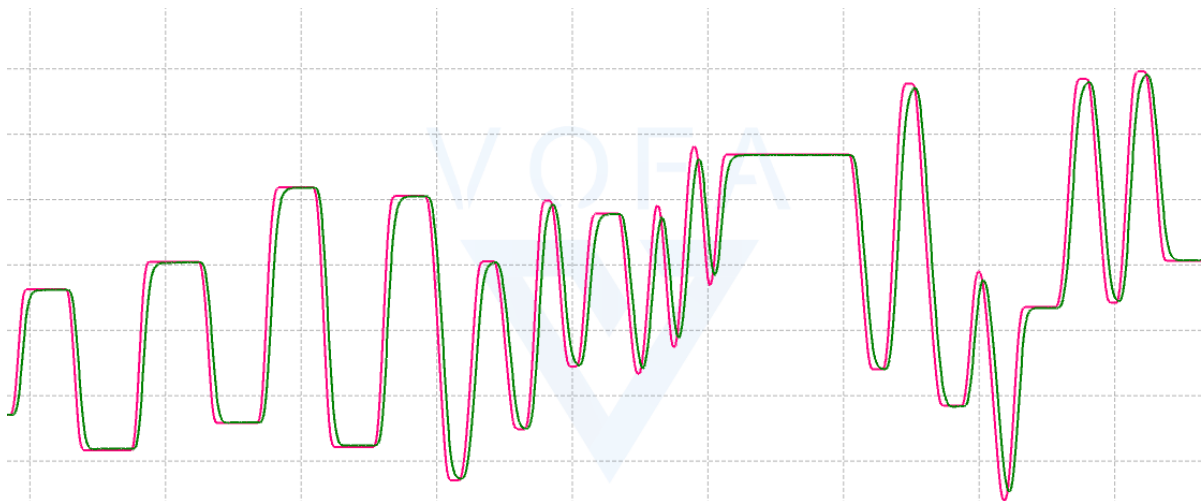
    x_n_2 = x_n_1;
    x_n_1 = x_n;
    return y_n;
}
```

打印出输入输出信号验证前馈设计效果：（红线为输入的期望角度，绿线为输出的反馈角度）

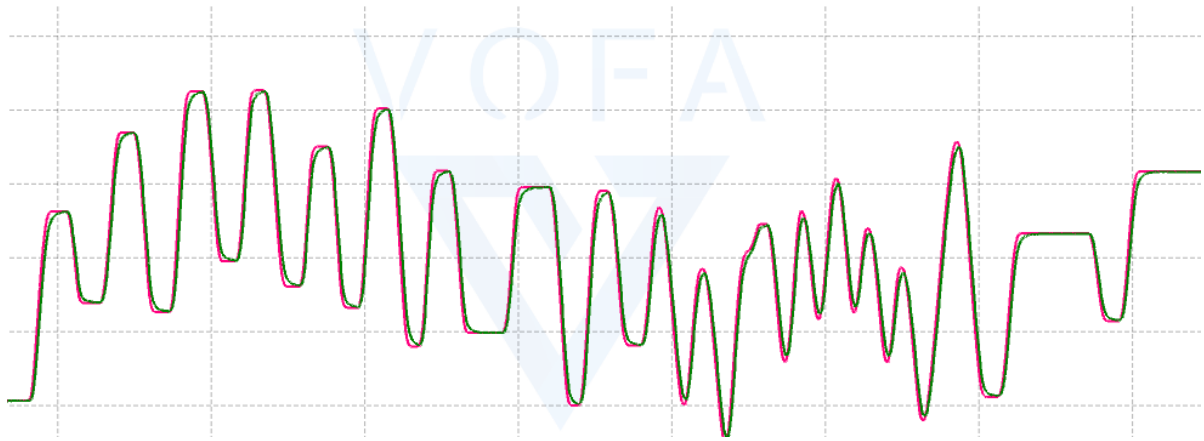
纯前馈（无外环PID）下的输出响应：



纯PID下的输出响应：



前馈+PID的输出响应：



2.4 根轨迹设计法

得到速度闭环的系统传递函数，除了可以用于前馈设计之外，还可以用于指导外环PID的参数设计。

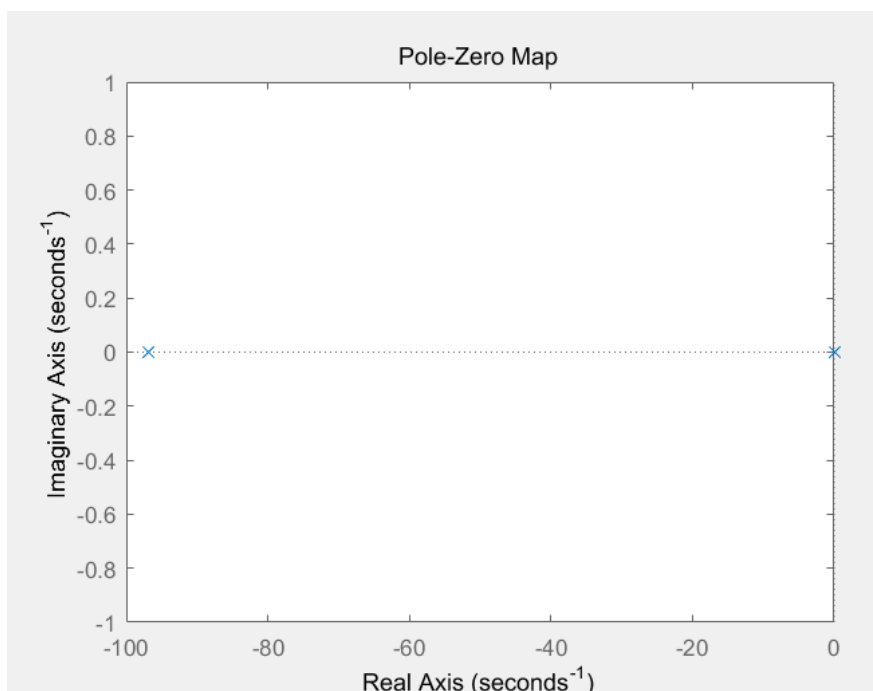
接下来介绍如何通过根轨迹来设计PID参数。

89.38

步兵实车的速度闭环传递函数为 $G(s) = \frac{89.38}{s + 96.89}$

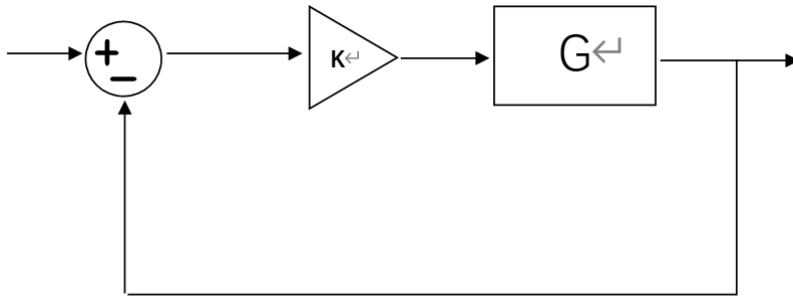
89.38

故内环传递函数为 $G(s) * \frac{1}{s} = \frac{89.38}{s^2 + 96.89s}$ ，其对应的零、极点图如下

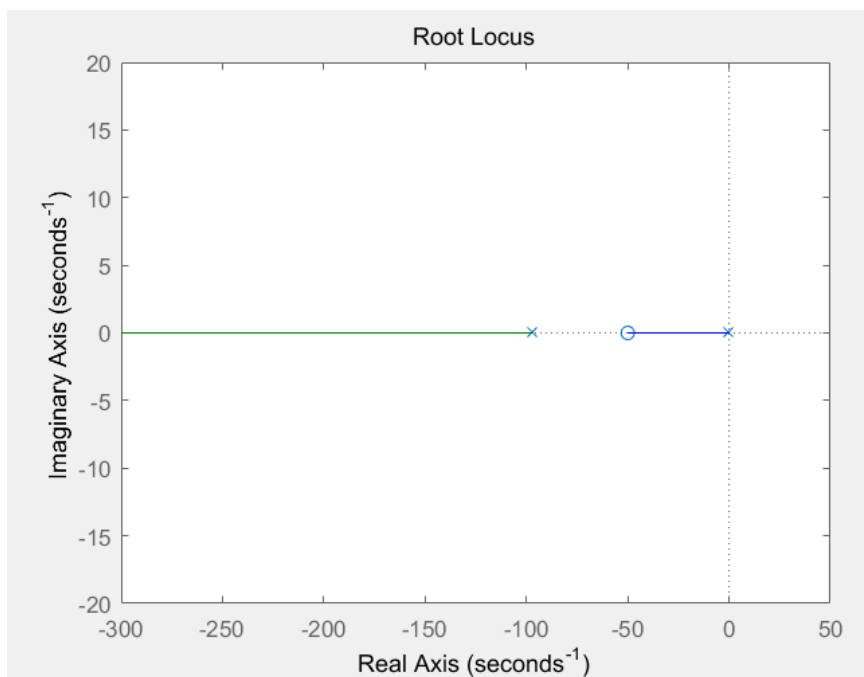


由图或公式可知，在实轴的 -96.89 和 0 处各有一个极点。因此，我们可以设置一个零点在实轴的 -50 处，

由此可确定参数 $\frac{K_p}{K_d} = 50$ ，我们大可以令 $K_p=50$ ，则 $K_d = K_p/50 = 1$ ，（注意，代码中的 K_d 需要再除以PID运行周期 T ，因为代码的 $dout$ 没有除以时间 T ）。



此时给原PID控制框图的误差值乘上一个增益 K ，调节增益 K ，系统的两个极点将会左移，即系统带宽扩大，且增益 K 越大，系统的带宽就越大，意味着系统的响应速度就越快。（设一个零点在 -50 时的根轨迹）



实车验证

我们调节PD参数，令零点在实轴 -50 处。（实验平台的PID运行周期为3ms）

```
float yaw_pid[6] = {0.2f, 0.0f, 1.32f, 30000, 0, 0}
```

修改原先的 `pid_calc` 函数，给每次的误差值都乘上一个增益。

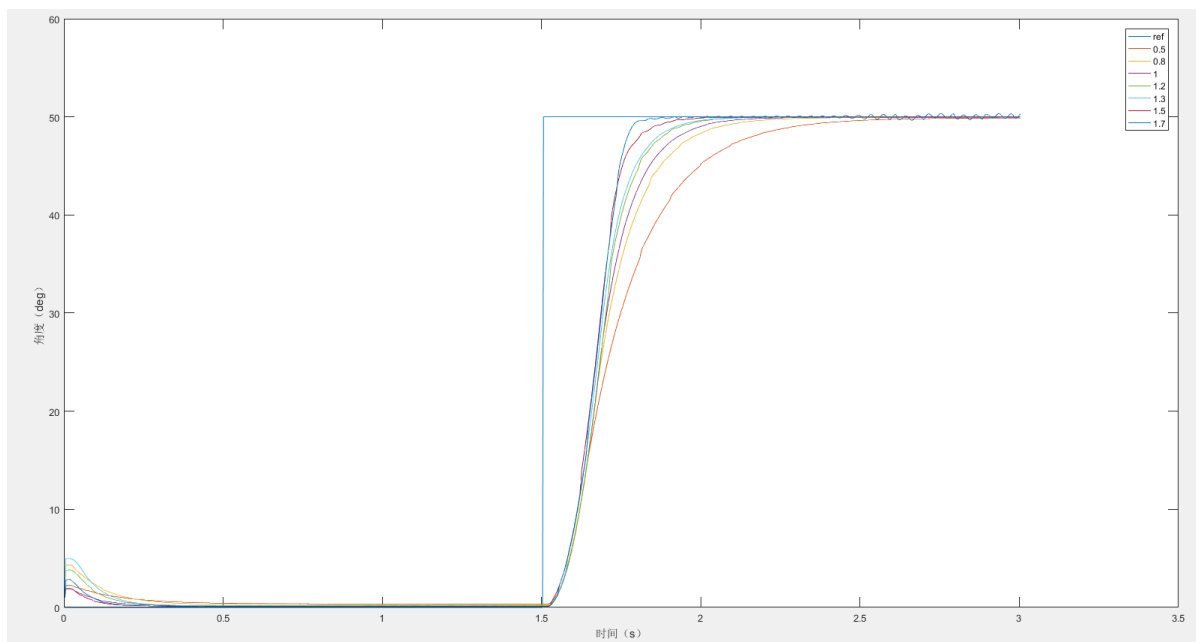
```

float gain_pid_calc(pid_t *pid, float get, float set, float gain)
{
    pid->get = get;
    pid->set = set;
    pid->error[NOW_ERR] = (set - get)*gain;

    #if (PID_MODE == POSITION_PID)
        pid->pout = pid->kp * pid->error[NOW_ERR]; // Kp总增益
        // pid->iout += pid->ki * pid->error[NOW_ERR];
        pid->iterm = pid->ki * pid->error[NOW_ERR]; //Ki ITerm项
        pid->iout = pid->iout + pid->iterm; //Ki 总增益
        pid->dout = pid->kd * (pid->error[NOW_ERR] - pid->error[LAST_ERR]); //Kd 总增益
    #endif
}

```

此时观察不同K增益下云台电机的阶跃响应：



由此可见，K增益越大，云台系统的响应就越快，也就是系统的带宽就越宽。当K增益大到一定值时，过宽的带宽就会引入高频噪声的影响（如图在增益=1.7时云台开始抖动）。

3. 总结

本章文档介绍了云台电机模型、Yaw轴云台系统传递函数，并根据其系统传递函数给出了前馈控制器的设计方法及基于根轨迹的参数调节方法。文档中介绍的这些方法都是经过验证，实际可行的方法，并非纸上谈兵。故希望大家看完文档后可以动手实操一下，这样才算是有所收获了。

未完待续.....(Pitch轴云台系统控制篇，如果有机会的话?)