轮足机器人速度跟踪探究

之前的方法

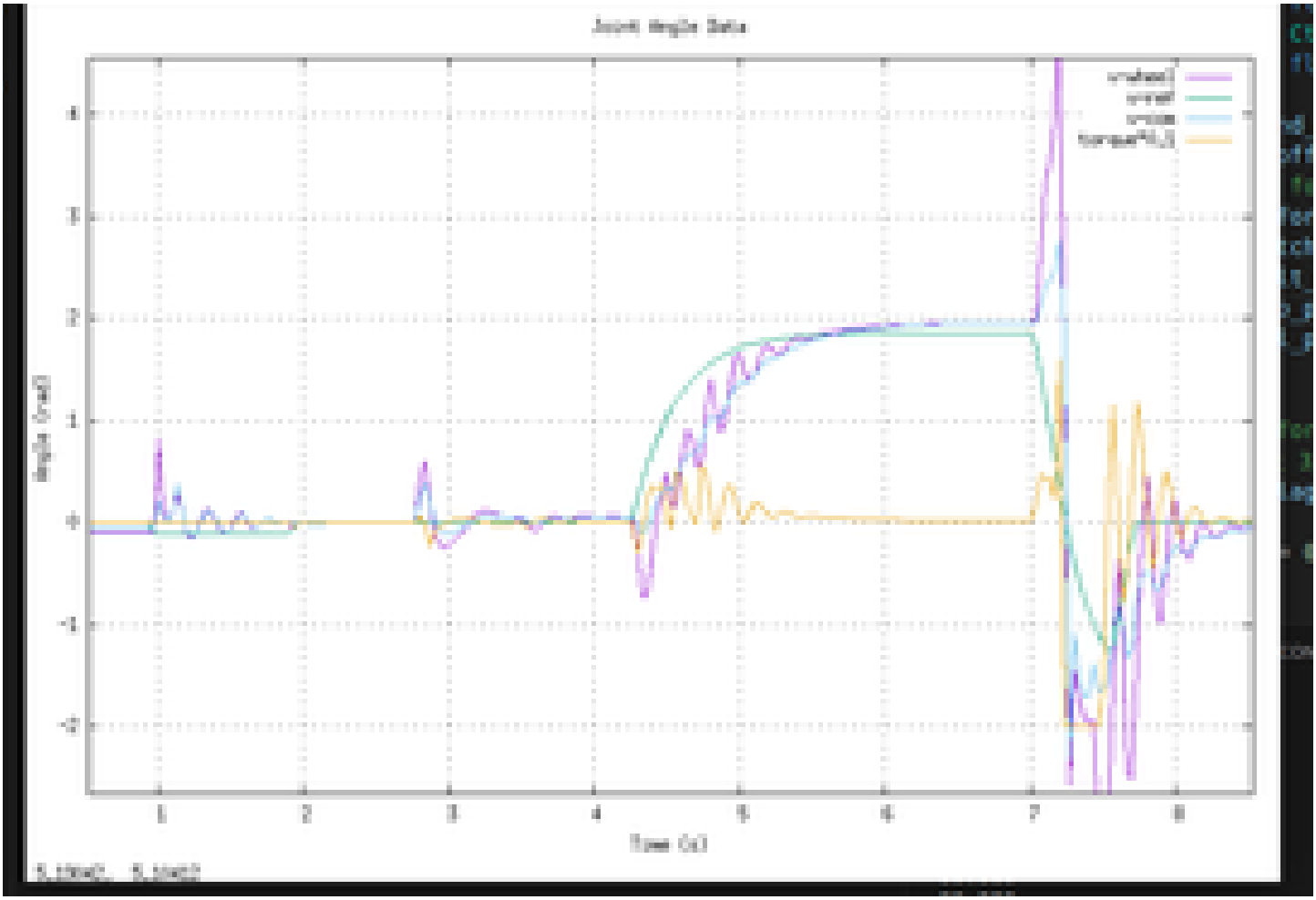
当前在刑天A1上提高加速度的方法有以下两种：

1. 增大ACCL_LIMIT，即系统输入的速度指令的加速度大小。
2. 调整LQR，一般而言，就是增大速度项在cost function中的权重，或者减少其他项的权重（最有效的是直接调整tilt）.

对于方法1，问题是提升性能有限，哪怕ACCL_LIMIT调到无穷大，即输入速度指令为方波，最终系统能否稳住，仍取决于LQR控制器的设计，即问题2上。

对于方法2，**容易顾此失彼**，比如刚刚在一定的加速度下调整好控制参数，如果目标加速度要继续提高，则整个控制器可能都要跟着变，且效果不明确，还可能会有机器人抖震等现象。

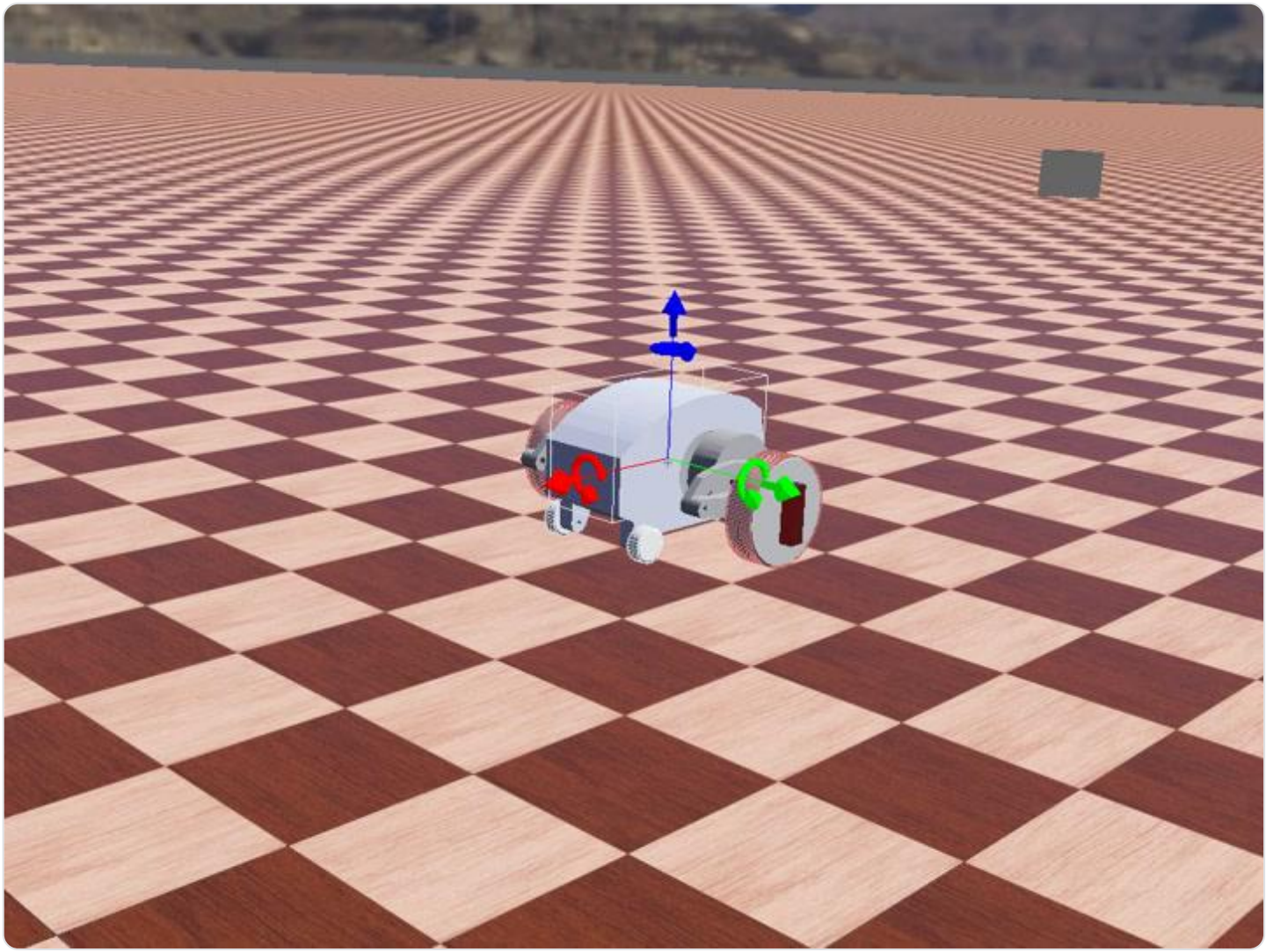
下面是一个案例，通过直接增大机器人前进速度的权重 (vel_psc和d_vel_psc从1.0变为2.0)，实现机器人的加速。加速度约为1.28m/s^2，且出现明显振铃，若继续增大上述参数可能会出现机器人原地站不稳。



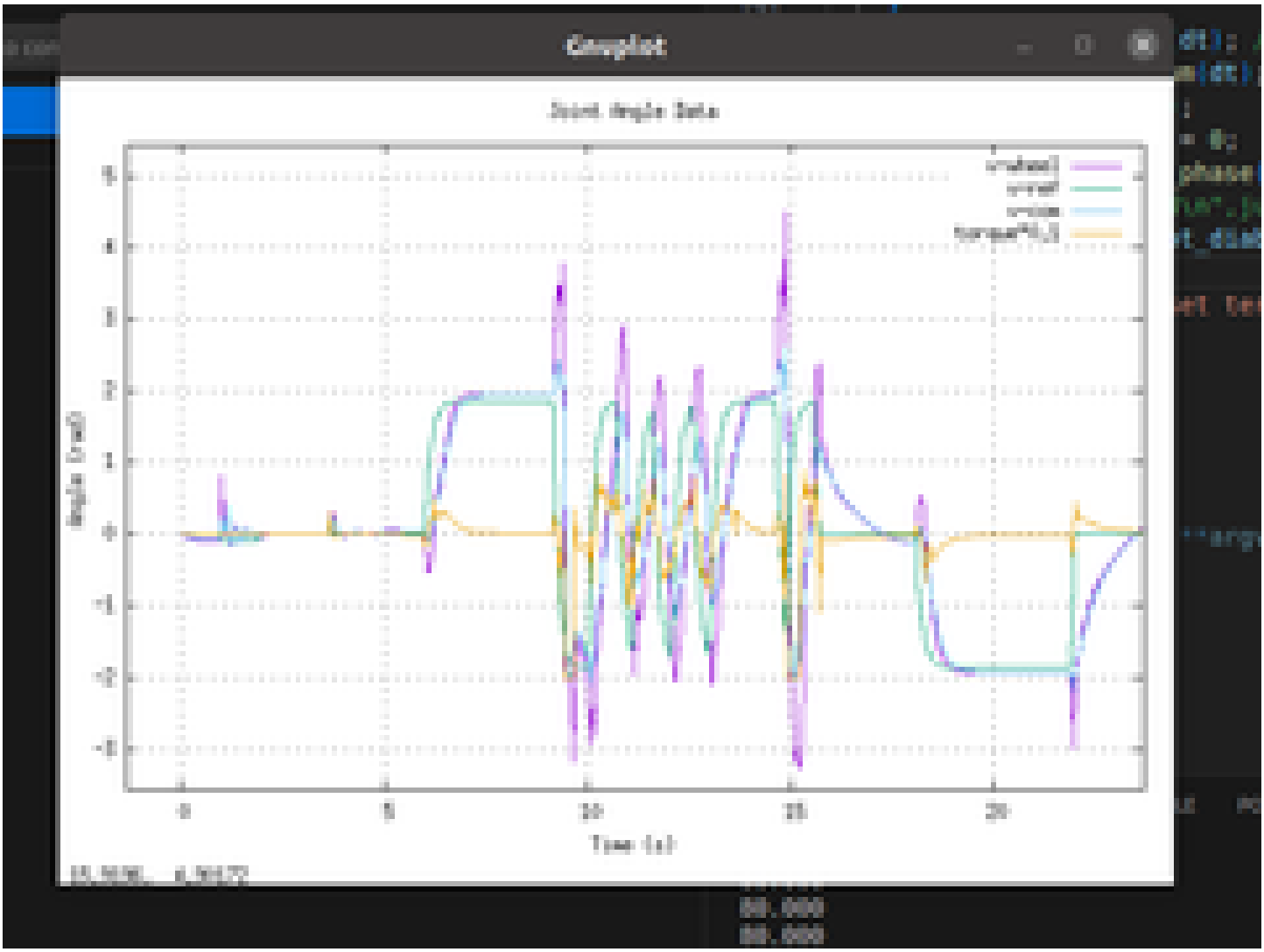
通过输入规划增加速度跟踪能力（目前仿真有效）

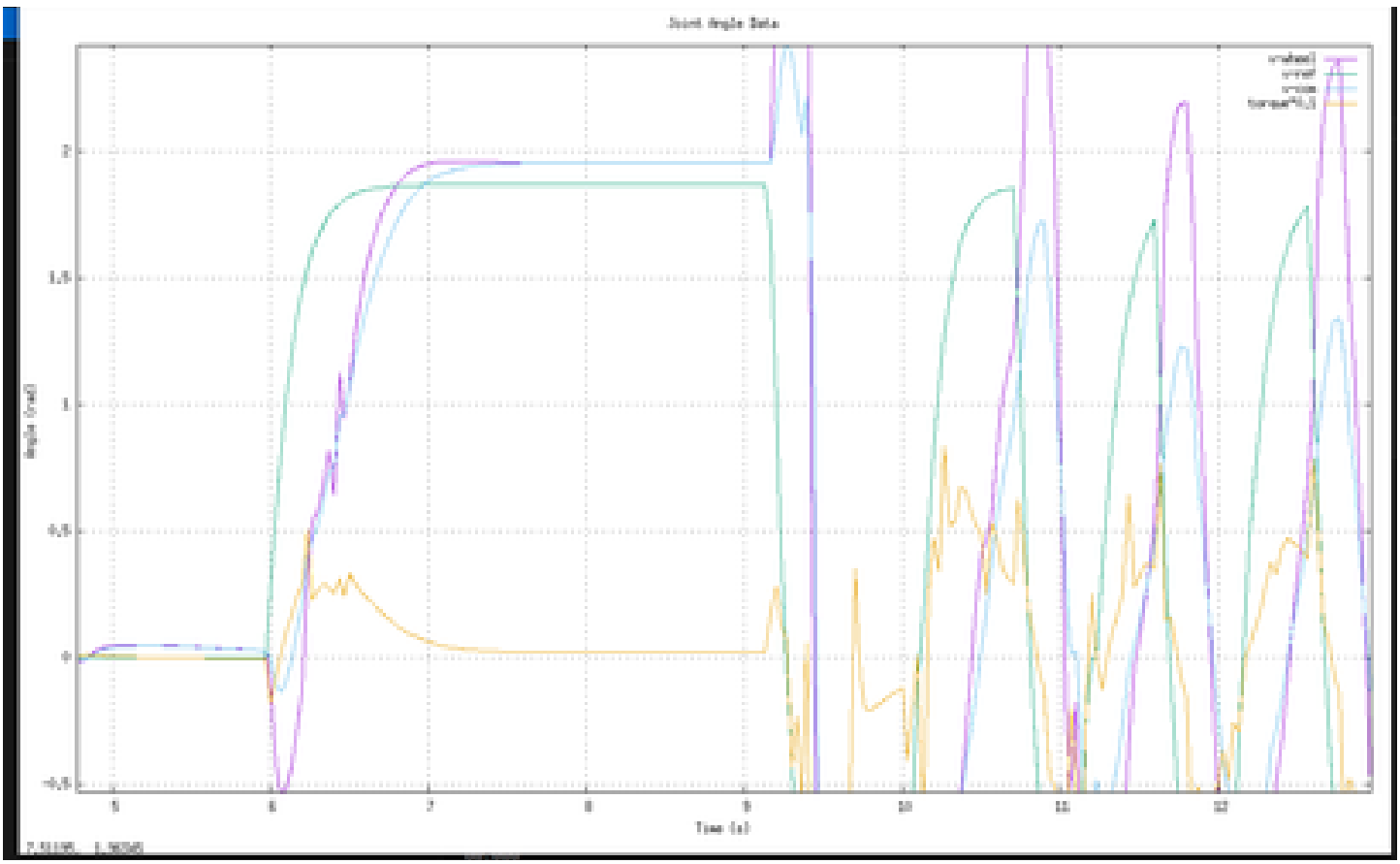
Planner也是一种控制器 —— by 高飞

大致思路是，在不改变系统指令输入（ACCL_LIMIT）、不改变LQR参数的情况下，通过规划**虚拟目标速度**，来提升轮足机器人的速度响应能力。下面提供一个参考案例，效果如下：



视频中所有的伸腿缩腿，均为机器人主动加减速，非机器人失控。曲线图如下，加速度可达1.9m/s^2左右,无明显振铃（刑天参数）。如果上调ACCL_LIMIT,最大加速度实测大概在2.5m/s^2。


Gnuplot
Joint Angle Data
Angle (rad)
Time (s)


Joint Angle Data
Angle (rad)
Time (s)

实现的源码如下：

```
    //以下为规划部分
    float set_vel_ff = 0.f;
    //计算机器人质心速度和目标速度的差值
    float act_d_forward_err = (set_forward_val) * robot.param_ptr->core.WHEEL_RADIUS - robot.posture_ptr->com_vel_forward[0];
    //引入sigmoid算子用来对虚拟目标速度进行平滑
    float sigmoid_err = (1 / (1 + std::exp(-0.01f * (act_d_forward_err) / 0.092f)) - 0.5f);
    set_vel_ff = SIGN_ABS(set_forward_val) * sigmoid_err * 400.f;
    bound(set_vel_ff,55.f);
    //以下开始LQR计算，注意此时重新使用轮速作为LQR控制量
    forward_err   =   pos_forward_err * robot.param_ptr->core.WHEEL_RADIUS;
    d_forward_err = (set_forward_val + set_vel_ff) * robot.param_ptr->core.WHEEL_RADIUS - robot.posture_ptr->vel_forward;
```

大致做了这么几件事情：

1. 在原有的速度指令上叠加前馈的虚拟速度set_vel_ff
2. 规划set_vel_ff，大致效果是希望接近目标速度时，速度前馈项平滑地降为0，在远离目标速度时，能迅速增大前馈项。经过考虑选择了sigmoid函数：

$$v_{ff} = K_1(\frac{1}{1+e^{-K_2*err_v}} - 0.5)$$

这样，在保证原有LQR稳定性的情况下，只要调整增益K1和K2，即可调整前馈速度输入的曲线即可。

有几个注意的点：

1. 比起直接做线性规划，sigmoid的非线性规划更有效地压制了振铃
2. 实际调试的过程中，如果加速度过大，有可能会导致腿长过长，导致腿部控制器切换，造成控制不稳定，目前的做法是腿长先不要调到最高。
3. sigmoid中的K1和K2调的不好也会出现1中线性规划中的振铃，一般是由于K2太大导致的。
4. 注意要用机身质心的速度，纯轮速因为腿部前后摆的关系，波动大，也容易导致振铃。

通过增加前馈，增强LQR本体的跟踪能力（粗测效果不理想，有待严格做验证）

出处：最优控制理论（七）LQR伺服跟踪控制器设计

一般LQR的cost function:

$$J = \int (x^T Q x + u^T R u) dt$$

此处的建模，使用的是状态变量x作为代价函数J的估计。 为了获得更好的跟踪效果，调整cost function为以下形式：

$$J_2 = \int (e^T Q e + u^T R u) dt$$

其中

$$e = y_r - y(t), y(t) = Cx$$

求J2的极小值后，控制序列u的表达式为：

$$u(t) = -R^{-1} B^T (Px - g)$$

$$g \approx (PBR^{-1}B^T - A^T)^{-1} C^T Q y_r$$

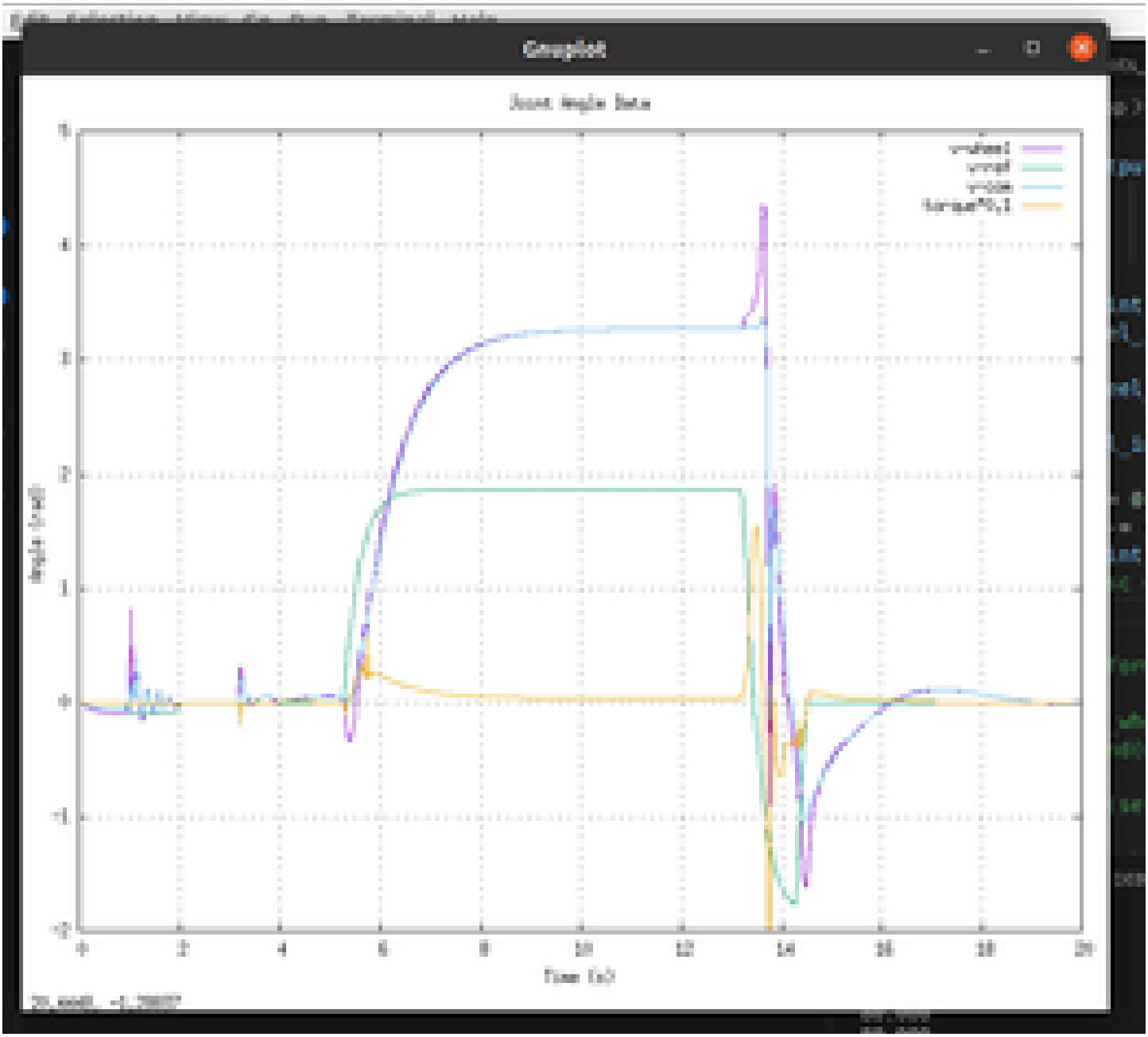
上述两式中， P是如下ricatti方程的解：

$$A^T P + PA - PBR^{-1}B^T + P + C^T Q C = 0$$

相比于普通的LQR，针对输出y的跟踪LQR里面，不仅要考虑状态矩阵A,B，还要考虑输出矩阵C。输出矩阵C理论上可以根据我们希望跟踪的输出信号（状态变量）来定义。

同时，观察u(t)，可以发现除了LQR的反馈增益 $-R^{-1}B^T P$,还多了一个前馈项， $-R^{-1}B^T g$ ，从表达式上看，这是一个与系统的参考输入yr相关的项，是一个与所有反馈状态都无关的前馈项。

我这里简单地试了下，直接在我们原始的LQR控制器里，加了一个 uf = k1 * yr，yr是目标速度输入，最终结果是，加速度确实上去了，**但稳态误差也更大了**，目前不知道怎么做处理，也许应该按Ricatti方程求一次k1的理论值，又或者要加积分项来弥补前馈力矩带来的稳态跃迁。总之暂时搁浅，最终效果如下：


Gnuplot
Joint Angle Data
Angle (rad)
Time (s)