一、*MIT* 模式工作说明

下图是电机 *MIT* 模式的控制框图

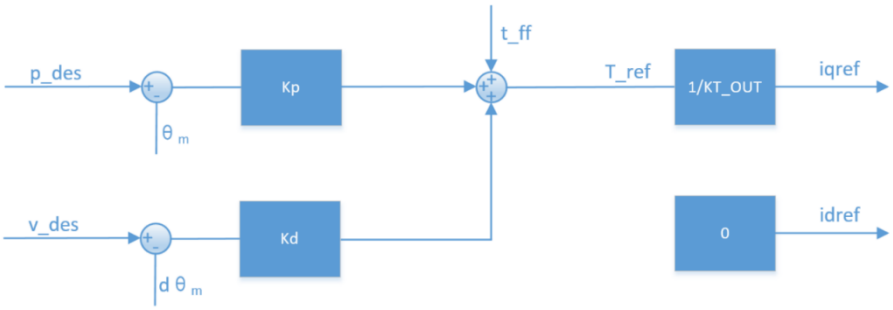

图 1 MIT 模式控制框图

MIT 模式可实现力矩、位置、速度三者混合控制，在上图中，位置环与速度环是并联形式，这里的位置环与速度环的输出值与前馈力矩 *t_ff* 相加得到参考力矩 *T_ref*：

$$T\_ref = kp*(p\_des - \theta_m) + kd*(v\_des - d\theta_m) + t\_ff$$

其中:

T_ref 为参考力矩，单位是 *N·m*。

kp 为位置增益。*kd* 为速度增益。

p_des 为电机输出轴的期望位置，单位为 *rad*。

θ_m 为电机输出轴的当前位置，单位为 *rad*。

v_des 为电机输出轴的期望速度，单位为 *rad/s*。

$d\theta_m$ 为电机输出轴的当前速度，单位为 *rad/s*。

参考力矩 *T_ref* 经过 *KT_OUT* 换算，得到参考电流 *iqref*，从而进入后续的电流 *PI* 控制器。

其中：

$$iqref = T\_ref / KT\_OUT$$

$$KT\_OUT = Kt*GR$$

$$Kt = 1.5*NPP*flux$$

iqref 为参考电流，单位为 *A*。

GR 为电机减速比。

Kt 为减速前的转矩常数，单位是 *N·m/A*。

NPP 是极对数。

flux 磁链，单位是 *Wb*，*可以通过读电机参数得出*。

二、*MIT* 模式用法说明

1、当 *kp=0*，*kd≠0* 时，给定 *v_des* 即可实现匀速转动。匀速转动过程中存在静差，另外 *kd* 不宜过大， *kd* 过大时会引起震荡。

2、当 *kp=0*，*kd=0*，给定 *t_ff* 即可实现给定扭矩输出。在该情况下，电机会持续输出一个恒定力矩。但是当电机空转或负载较小时，如果给定 *t_ff* 较大，电机会持续加速，直到最大速度，这时也仍然达不到目标力矩 *t_ff*。

3、当 *kp≠0*，*kd=0* 时，会引起震荡。即对位置进行控制时，*kd* 不能赋 0，否则会造成电机震荡，甚至失控。

4、当 *kp≠0*，*kd≠0* 时，有多种情况，这里简单介绍两种情况。

（1）当期望位置 *p_des* 为常量，期望速度 *v_des* 为 0 时，可实现定点控制，在这个过程中，实际位置 θ_m 趋近于 *p_des*，实际速度 $d\theta_m$ 趋近于 0。

（2）当 *p_des* 是随时间变化的连续可导函数时，同时 *v_des* 是 *p_des* 的导数，可实现位置跟踪和速度跟踪，即按照期望速度旋转期望角度。

以下是基于达妙 H7 开发板的一个简单例子：

```
void TIM2_IRQHandler(void)
{
  /* USER CODE BEGIN TIM2_IRQn 0 */
    time=time+0.001f;
    kp=1.0f;
    kd=1.0f;
    tor_set=0.0f;
    pos_set=sin(2*3.1415926f*1.0f*time);
    vel_set=2*3.1415926f*1.0f*cos(2*3.1415926f*1.0f*time);
    mit_ctrl(&hfdcan1, 1,pos_set, vel_set,kp, kd,tor_set);//MIT 模式发送力矩

  /* USER CODE END TIM2_IRQn 0 */
    HAL_TIM_IRQHandler(&htim2);
  /* USER CODE BEGIN TIM2_IRQn 1 */

  /* USER CODE END TIM2_IRQn 1 */
}
```

如图，在 1ms 定时器中断函数中发送控制指令给 *DM4310* 电机，此时电机是空载状态。这里设定的期望位置 *p_des* 是 1HZ、幅值为 1 的正弦信号，期望速度

v_des 是对应的导数，Kp 为 1，kd 为 1，前馈力矩 tor_set 为 0。结果如下：

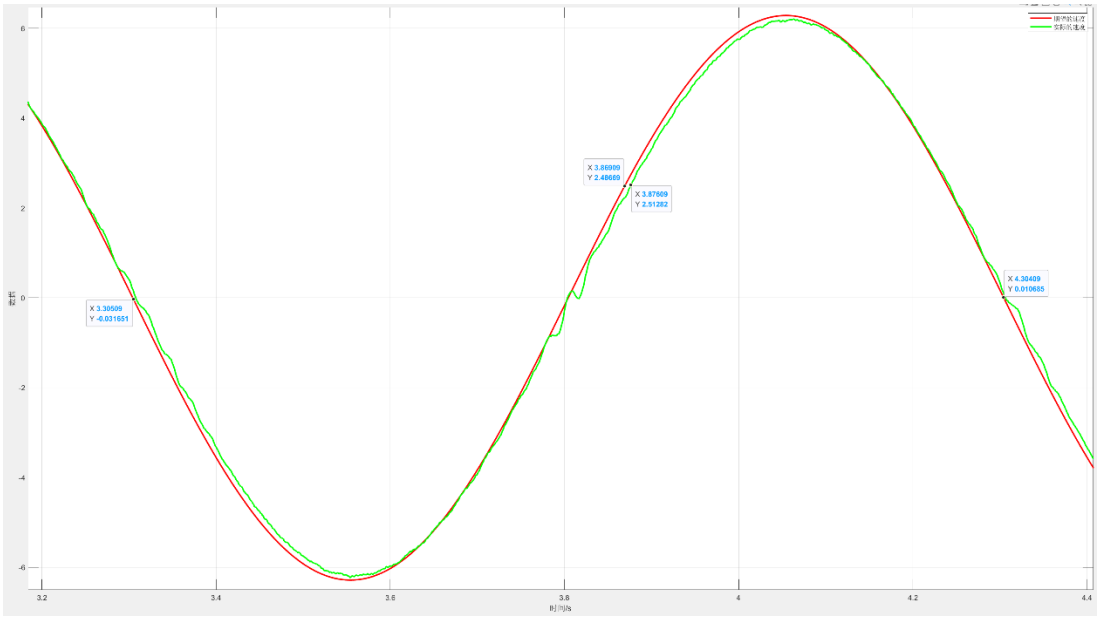

图 2 速度跟踪图

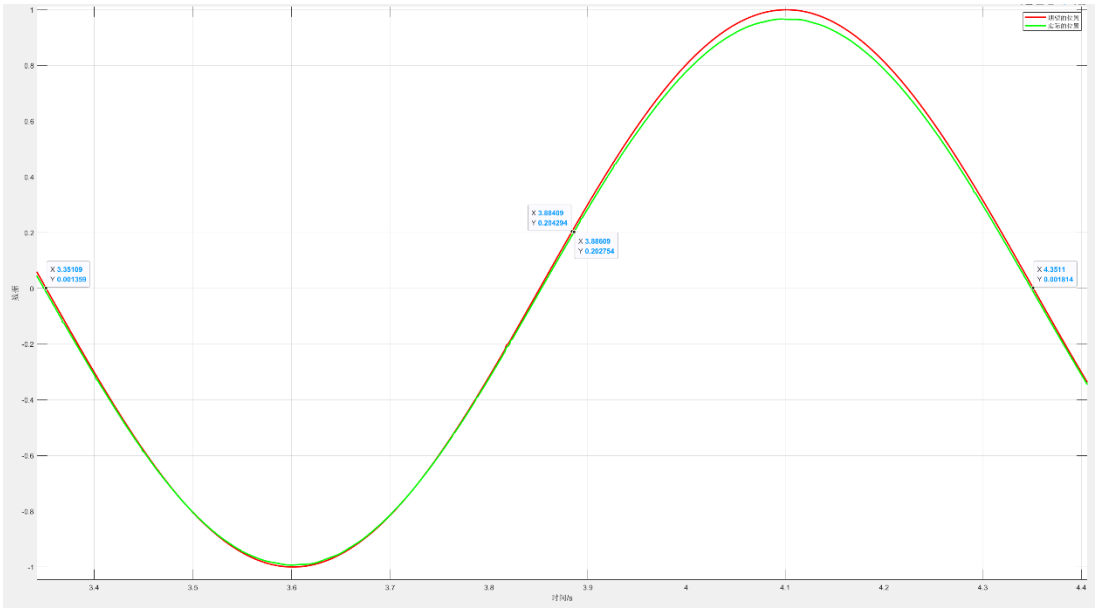

图 3 位置跟踪图

如图，在该条件下电机具有一定的跟踪效果。当电机带有负载时，则需要加入前馈力矩进行补偿。

三、示例代码

下面是 *MIT* 控制模式发送函数

```
/**
*********************************************************************
* @brief:       mit_ctrl: MIT 模式下的电机控制函数
* @param[in]:   hcan:           指向 CAN_HandleTypeDef 结构的指针，用于指定 CAN 总线
* @param[in]:   motor_id:    电机 ID，指定目标电机
* @param[in]:   pos:            位置给定值
* @param[in]:   vel:            速度给定值
* @param[in]:   kp:             位置比例系数
* @param[in]:   kd:             位置微分系数
* @param[in]:   torq:           转矩给定值
* @retval:      void
* @details:     通过 CAN 总线向电机发送 MIT 模式下的控制帧。
*********************************************************************
**/
void mit_ctrl(hcan_t* hcan, uint16_t motor_id, float pos, float vel,float kp, float kd, float torq)
{
    uint8_t data[8];
    uint16_t pos_tmp,vel_tmp,kp_tmp,kd_tmp,tor_tmp;
    uint16_t id = motor_id + MIT_MODE;// MIT_MODE=0x00

    //将浮点数据等比例转换成整数
    pos_tmp = float_to_uint(pos,  P_MIN,  P_MAX,  16);//（-12.5~12.5）
    vel_tmp = float_to_uint(vel,  V_MIN,  V_MAX,  12);//（-30.0~30.0）
    kp_tmp  = float_to_uint(kp,   KP_MIN, KP_MAX, 12);//（0.0~500.0）
    kd_tmp  = float_to_uint(kd,   KD_MIN, KD_MAX, 12);//（0.0~5.0）
    tor_tmp = float_to_uint(torq, T_MIN,  T_MAX,  12);//（-10.0~10.0）

    data[0] = (pos_tmp >> 8);
    data[1] = pos_tmp;
    data[2] = (vel_tmp >> 4);
    data[3] = ((vel_tmp&0xF)<<4)|(kp_tmp>>8);
    data[4] = kp_tmp;
    data[5] = (kd_tmp >> 4);
    data[6] = ((kd_tmp&0xF)<<4)|(tor_tmp>>8);
    data[7] = tor_tmp;

    //通过 can 总线  发送到电机驱动器
    canx_send_data(hcan, id, data, 8);
}
```

MIT 命令采用浮点数据等比例转换成整数发送到驱动器，驱动器再将接收到的整数等比例转换成浮点数据。这转换需要用到转换函数 *float_to_uint*，这转换函数需要首先确定两个等比例转换的最大最小值，这两个值可以在上位机参数设定页面查询，其中 *KP*、*KD* 的最大最小值默认分别为 0.0~500.0、0.0~5.0。*Pos*、*Vel*、*Torque* 分别预设为±12.5、±30、±10，这三个参数可以根据电机的实际参数进行调整。但发送控制命令时，一定要与设定值保持一致，否则会控制命令会发生等比例缩放。