ICSC 2019

Core Competence Enhanced by MBD

IDAJ CAE Solution Conference

IPM电机控制系统建模

IDAJ中国—电磁技术部　张志金

目录

JMAG控制系统

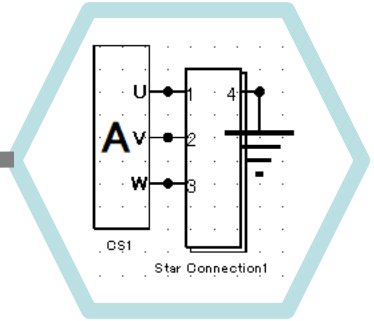
■旧版本控制系统

旧版本的电路

无控制

单纯正弦波

无法计算控制下的损失



旧版本的电路

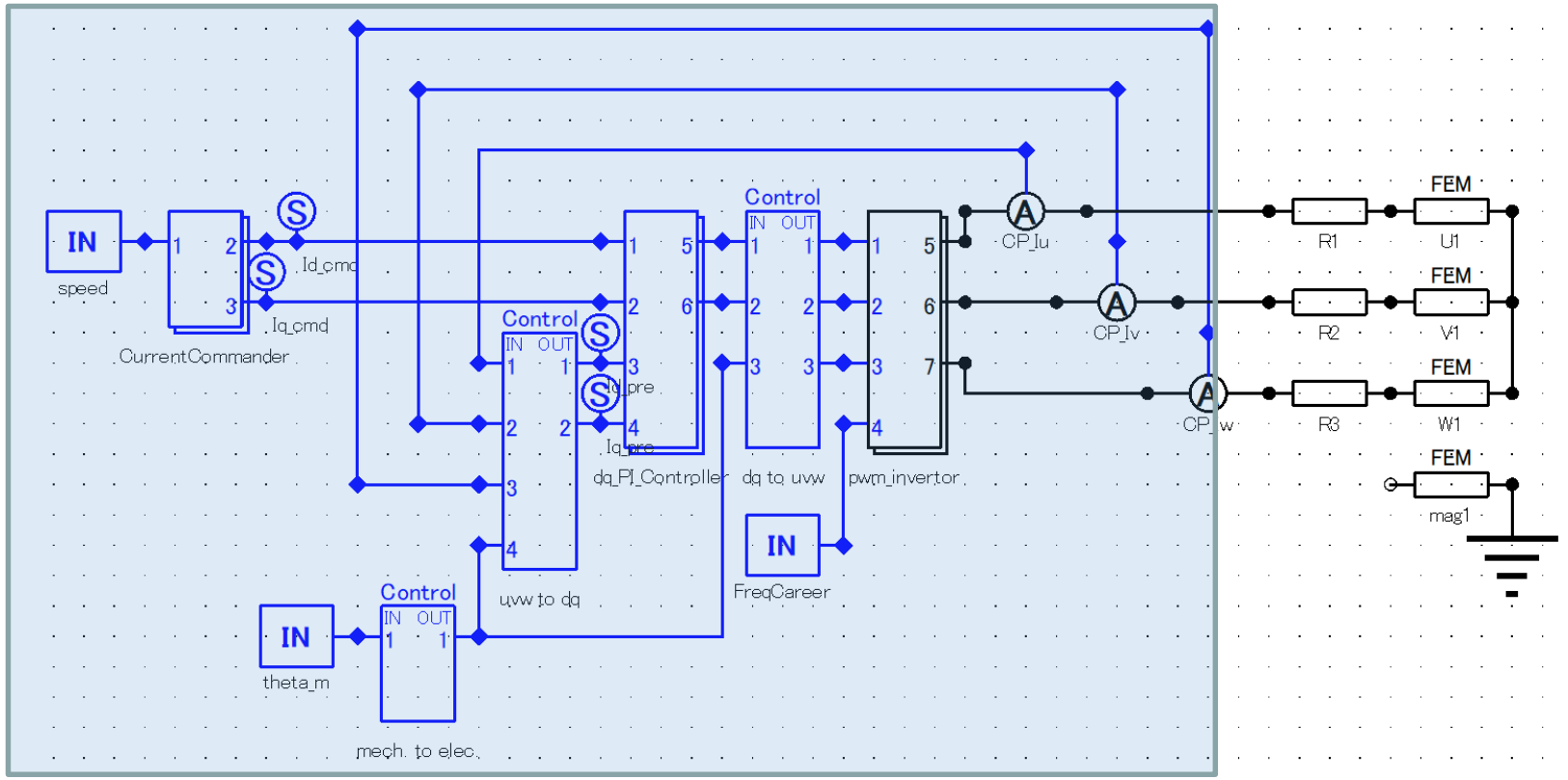
操作难度大

控制实现方式繁琐

对编程能力要求高

JMAG控制系统

■新版本控制系统，加入了类似于MATLAB/Simulink的电路控制功能。



框内蓝色线为控制电路

黑色线为功率电路

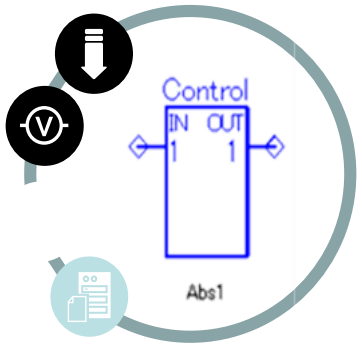
目录

- JMAG控制系统
- 元件介绍
- 模型介绍
- 控制原理
- 仿真结果
- 总结

JMAG控制元件

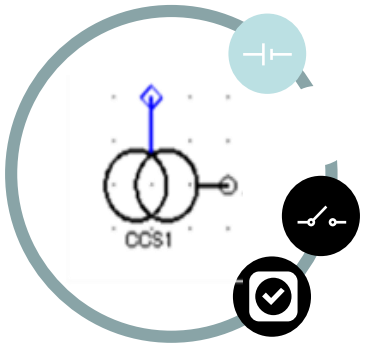
■控制元件分类

- 控制输入元件
- 控制输出元件
- 控制信号探针元件
- 用户自定义元件
- 函数元件
- JMAG-RT 元件
- PWM信号生成器元件



纯控制元件

输入输出皆为控制信号，图中为蓝色线



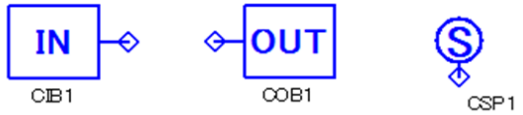
- 控制电压源元件
- 控制电流源元件
- 控制电压探针元件
- 控制电流探针元件
- 控制开关元件
- 逆变器元件

过渡元件

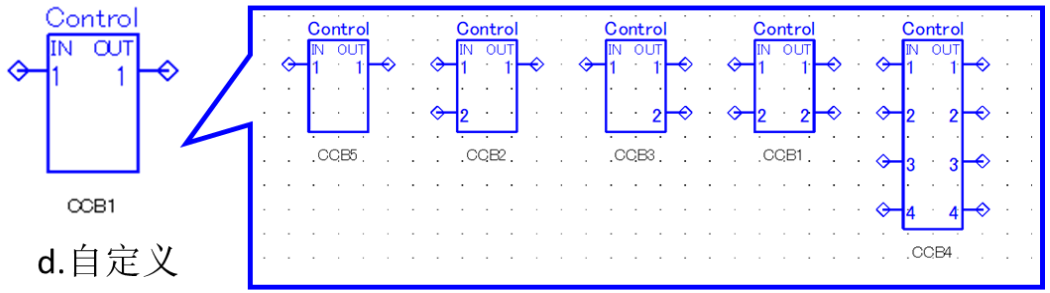
输入或输出的一方为控制信号，另一方为电路信号(电压、电流等)，在图中为蓝色和黑色

JMAG控制元件

■纯控制元件



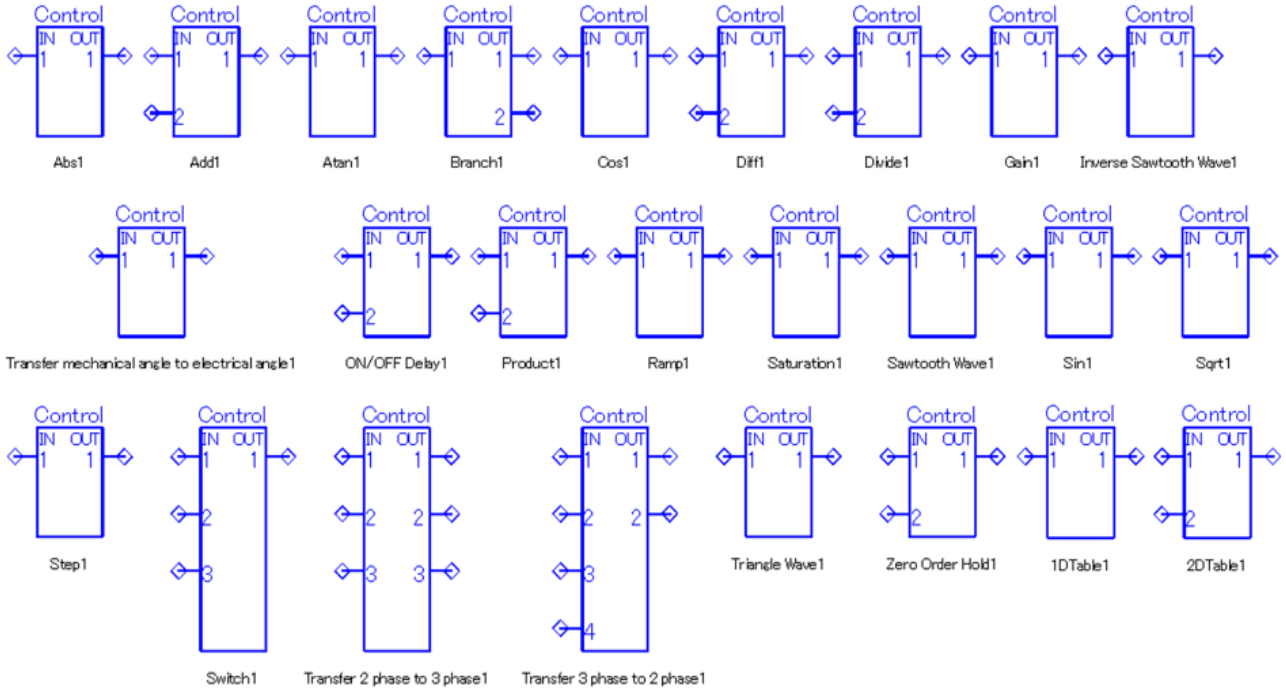
a.控制输入　b.控制输出　c.控制探针



d.自定义

Python　2.7.10

■函数元件



JMAG控制元件

■函数元件

常见运算

[加]（Add）
[减]（Diff）
[乘]（Product）
[除]（Divide）
[正弦]（Sin）
[余弦]（Cos）
[反正切]（Atan）
[增益]（Gain）
[平方根]（Sqrt）
[绝对值]（Abs）
[分支]（Branch）

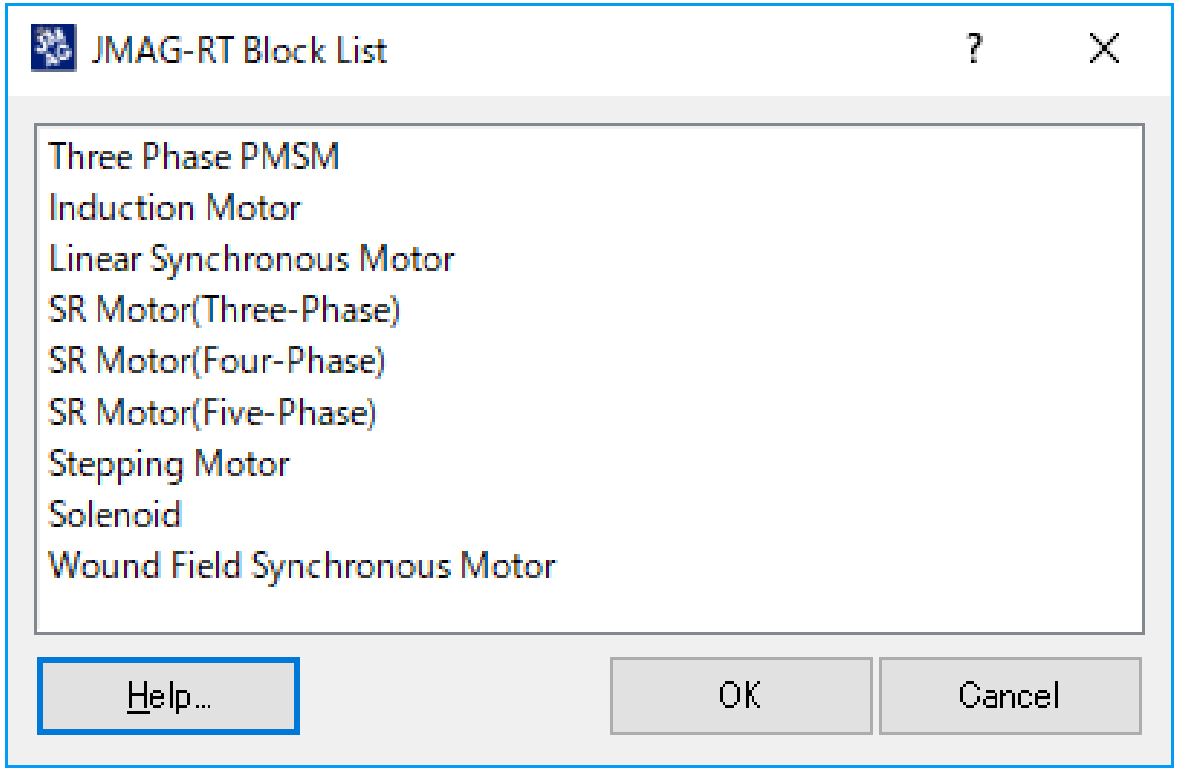
常见信号

[开关]（Switch）
[斜坡]（Ramp）
[饱和]（Saturation）
[Step]（Step）
[三角波]（Triangle Wave）
[锯齿波]（Sawtooth Wave）
[反锯齿波]（Inverse Sawtooth Wave）
[零阶保持器]（Zero Order Hold）
[ON/OFF延时]（ON/OFF Delay）
[1D表格]（1DTable）
[2D表格]（2DTable）

电机常见需求

[机械角电气角转换]（Transfer mechanical angle to electrical angle）
[2相3相转换]（Transfer 2 phase to 3 phase）
[3相2相转换]（Transfer 3 phase to 2 phase）

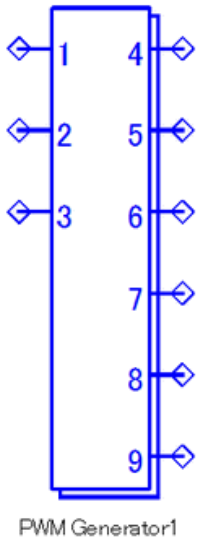
■JMAG-RT 元件：快速进入稳态，缩减仿真时间



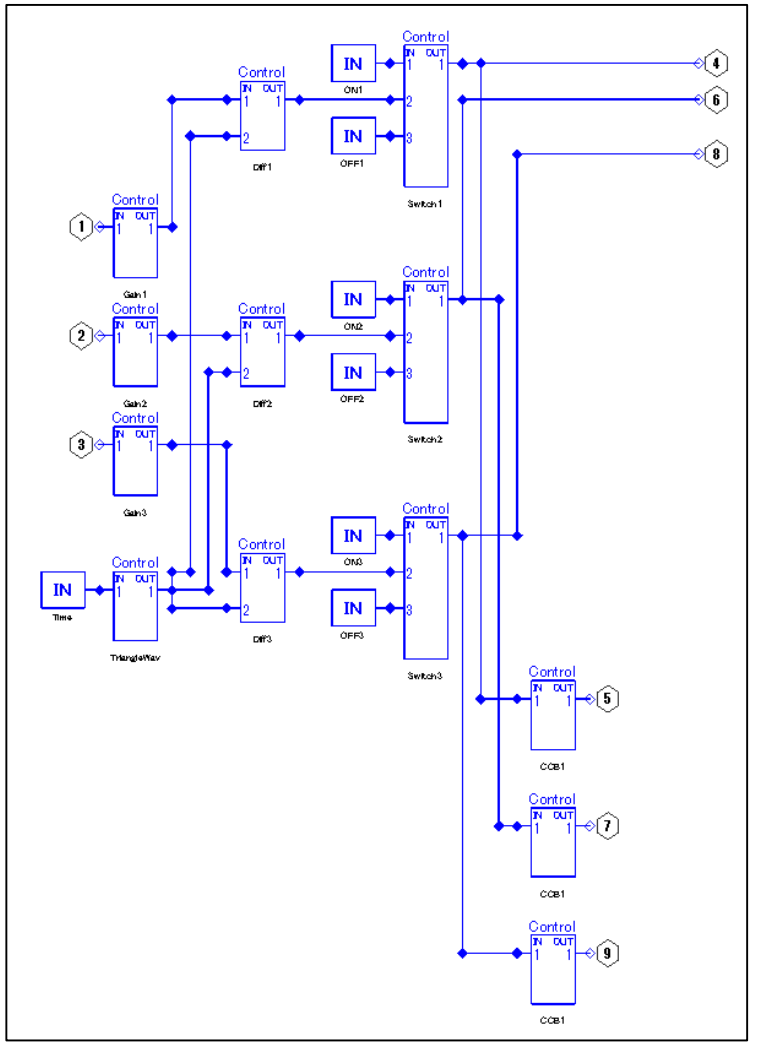
JMAG控制元件

■PWM信号生成器元件

U、V、W三相电压信号

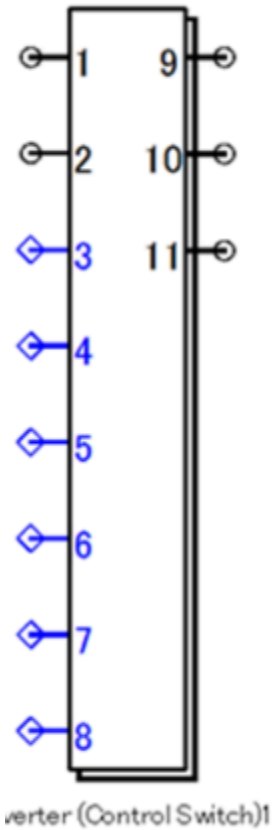


三相桥上下桥臂开关信号



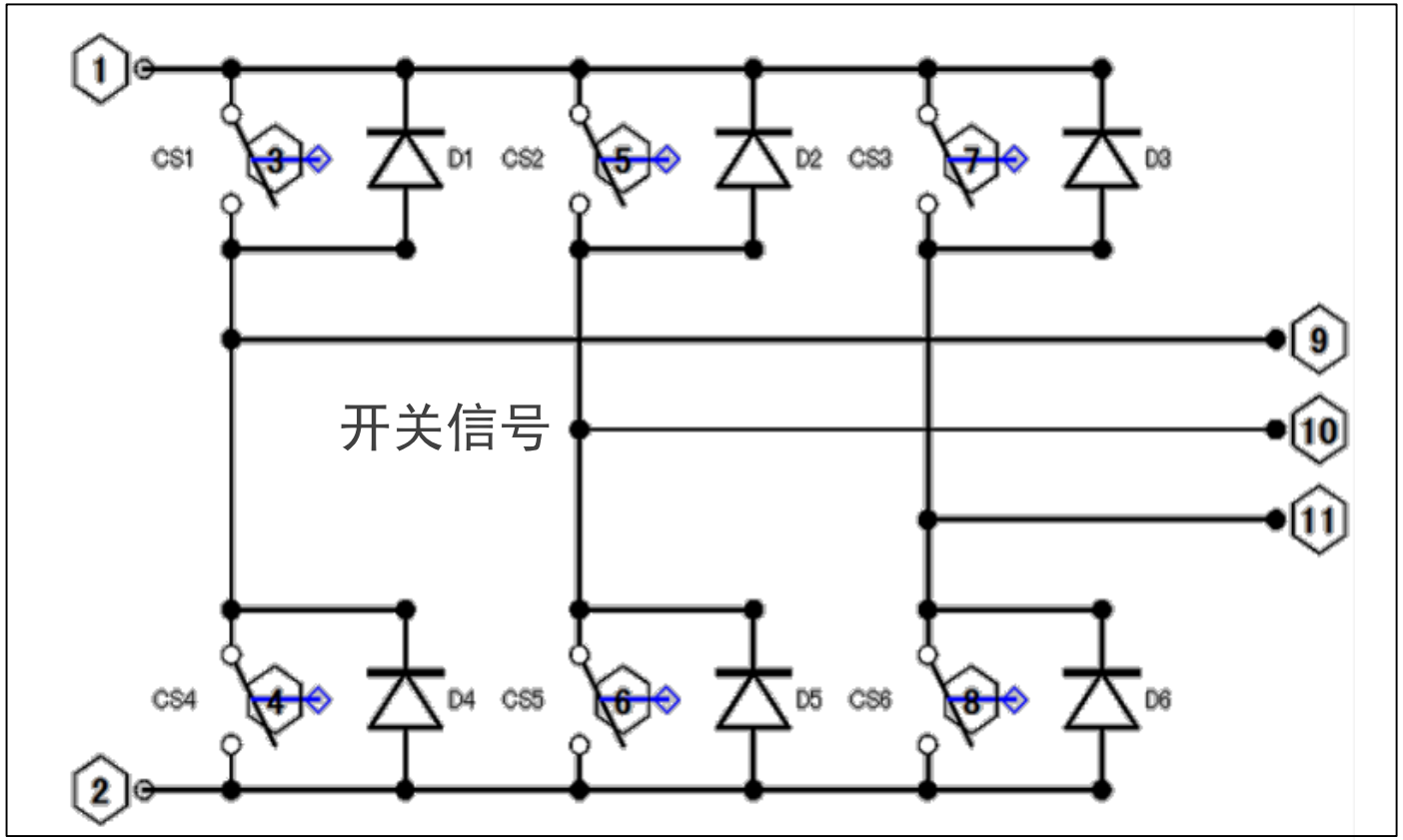
JMAG控制元件

■逆变器元件



verter (Control Switch)1

电源信号



开关信号

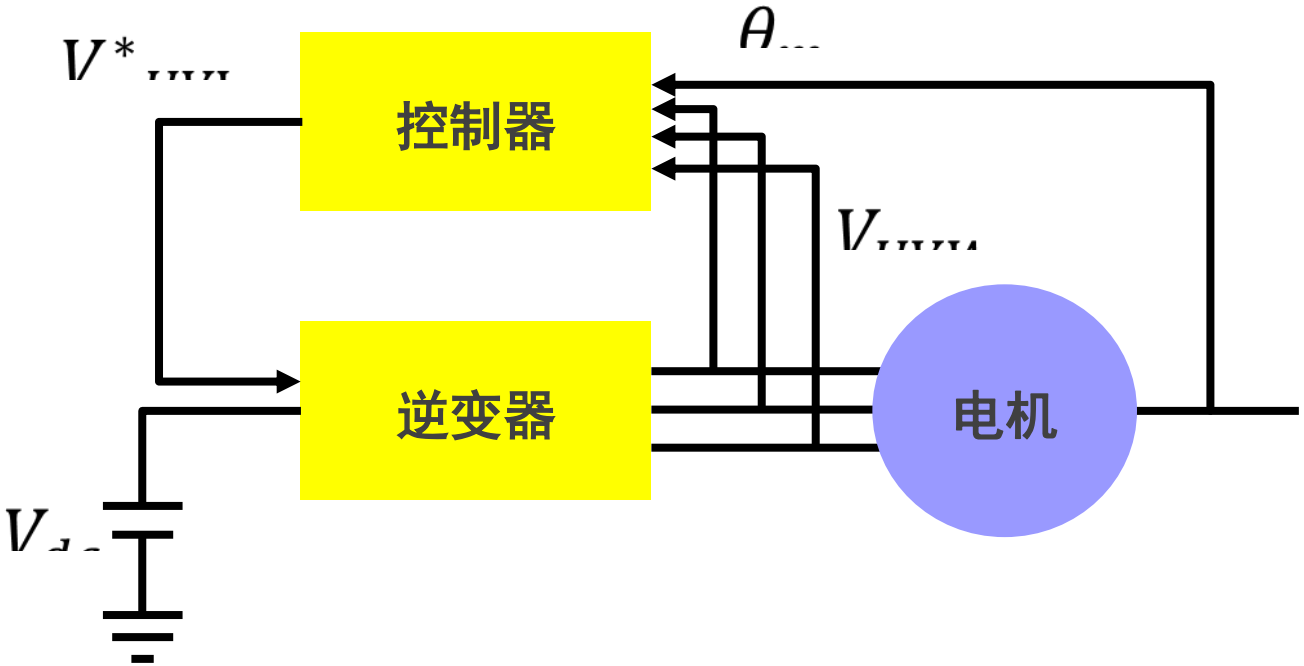
三相电压信号

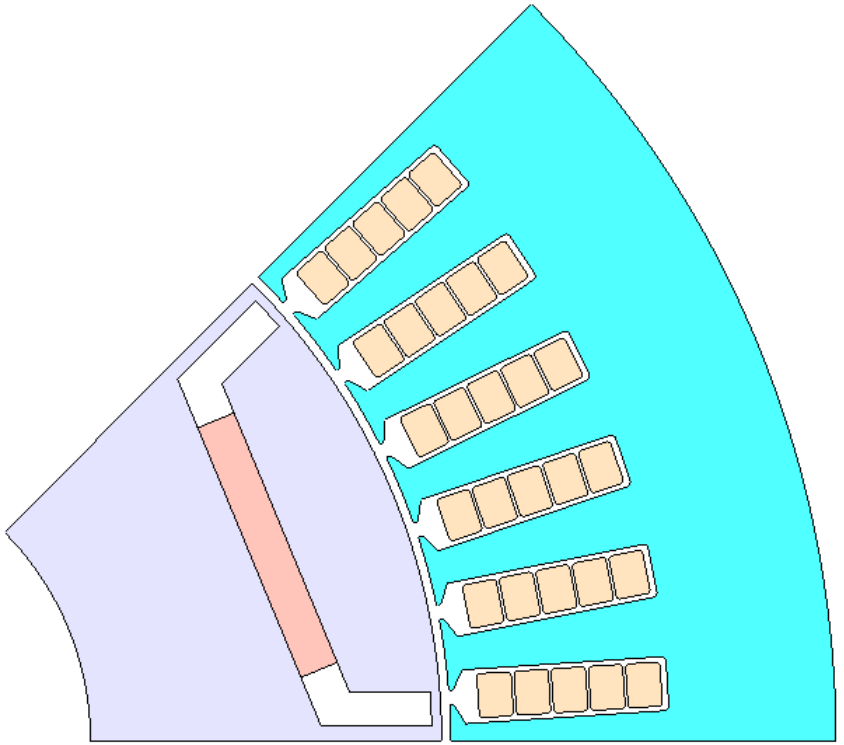
目录

- JMAG控制系统
- 元件介绍
- 模型介绍
- 控制原理
- 仿真结果
- 总结

模型介绍

- 48槽8极IPM电机
- 考虑电流控制的谐波损耗分析





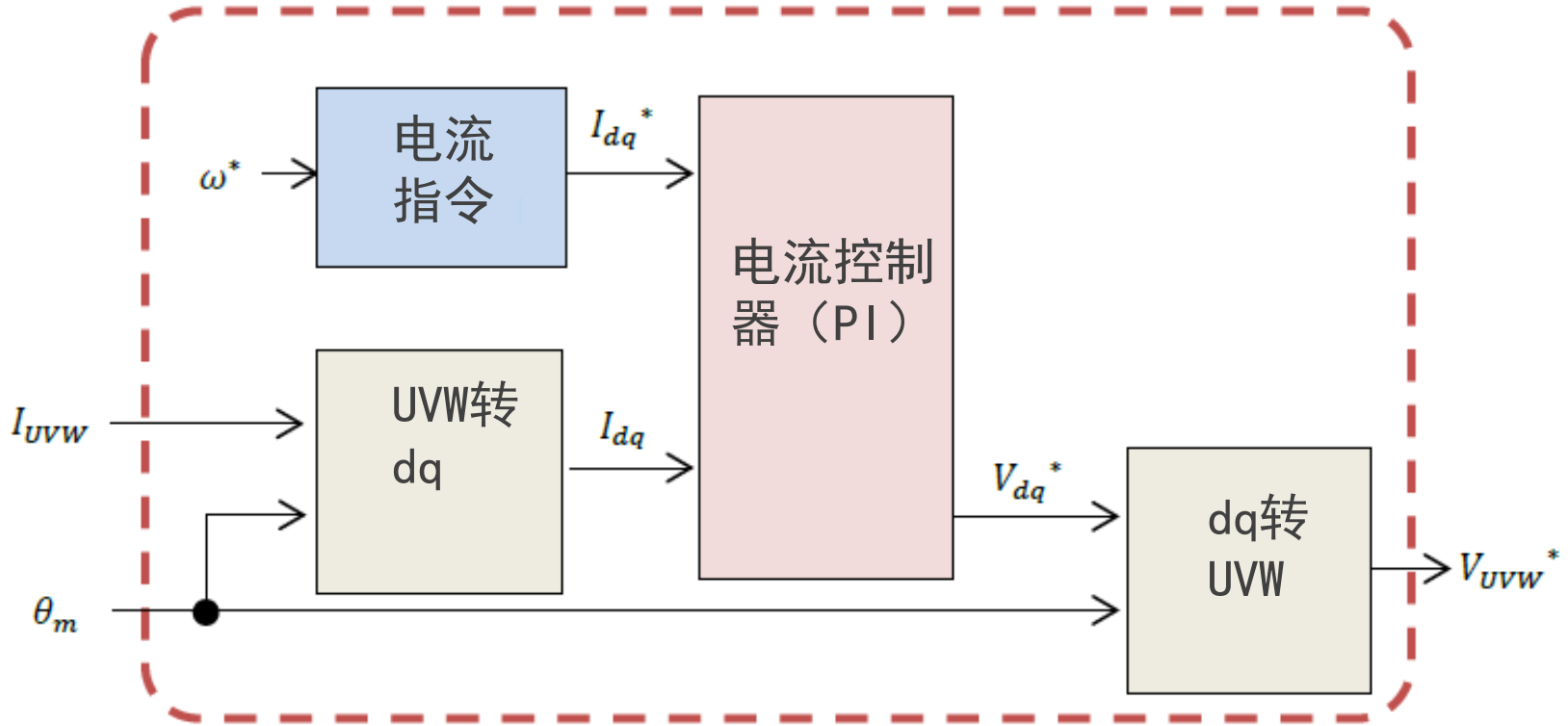
目录

- JMAG控制系统
- 元件介绍
- 模型介绍
- 控制原理
- 仿真结果
- 总结

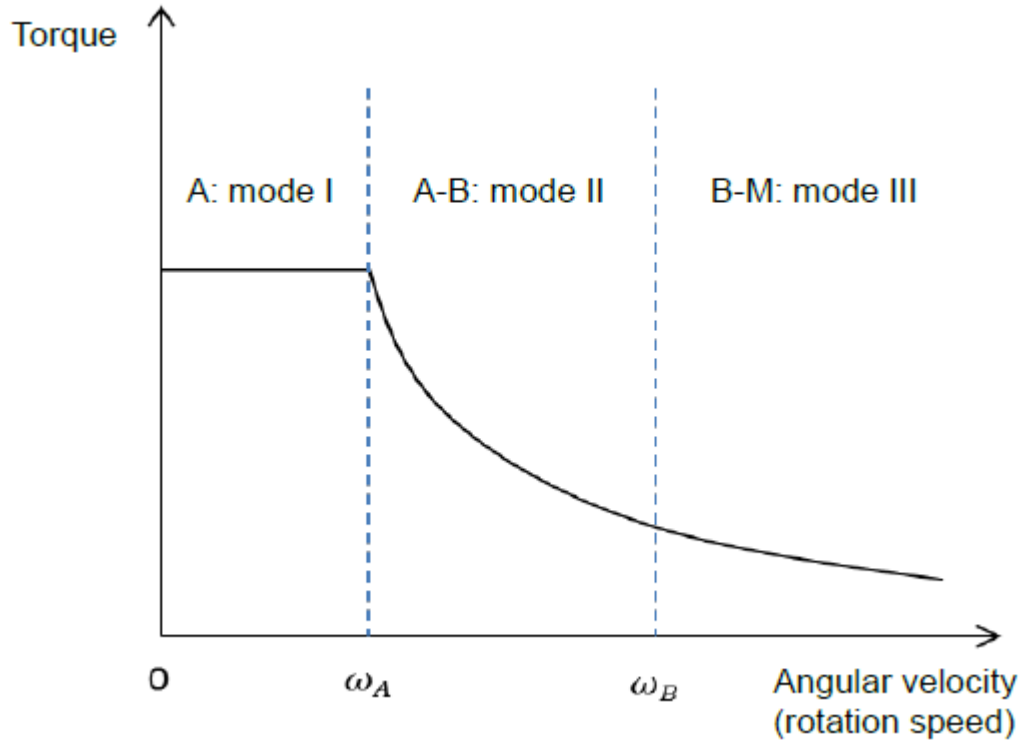
控制方法

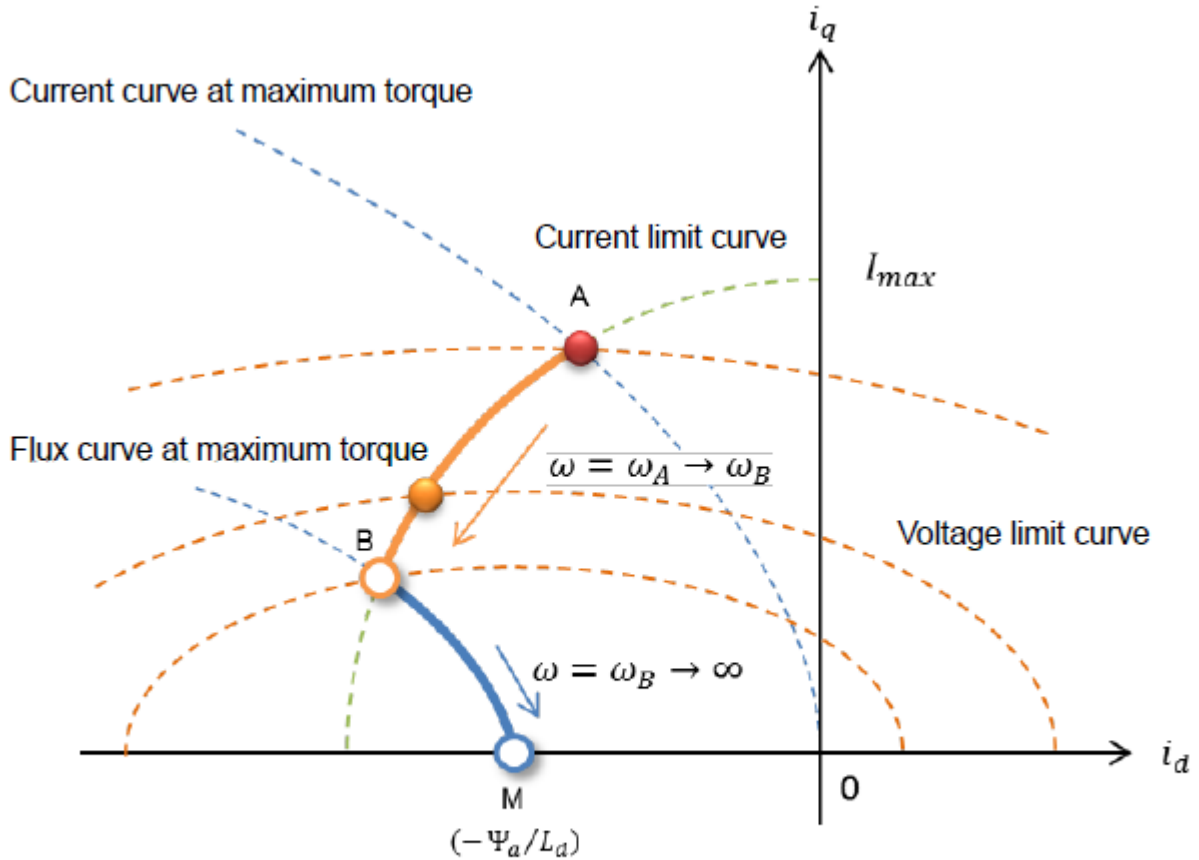
■三相电流和转子旋转角度作为输入。将输入由UVW三相转为dq两相，并使用转换后的dq电流值及指令dq电流值作为输入进行PI控制。将dq轴电压转换为UVW三相电压指令值，并输出到逆变器。

ω^*:指令角速度
I_{dq}^*:指令dq轴电流
θ_m:转子旋转角度
I_{UVW}:相电流
I_{dq}:dq轴电流
V_{dq}^*：指令dq轴电压
V_{UVW}^*：指令三相电压



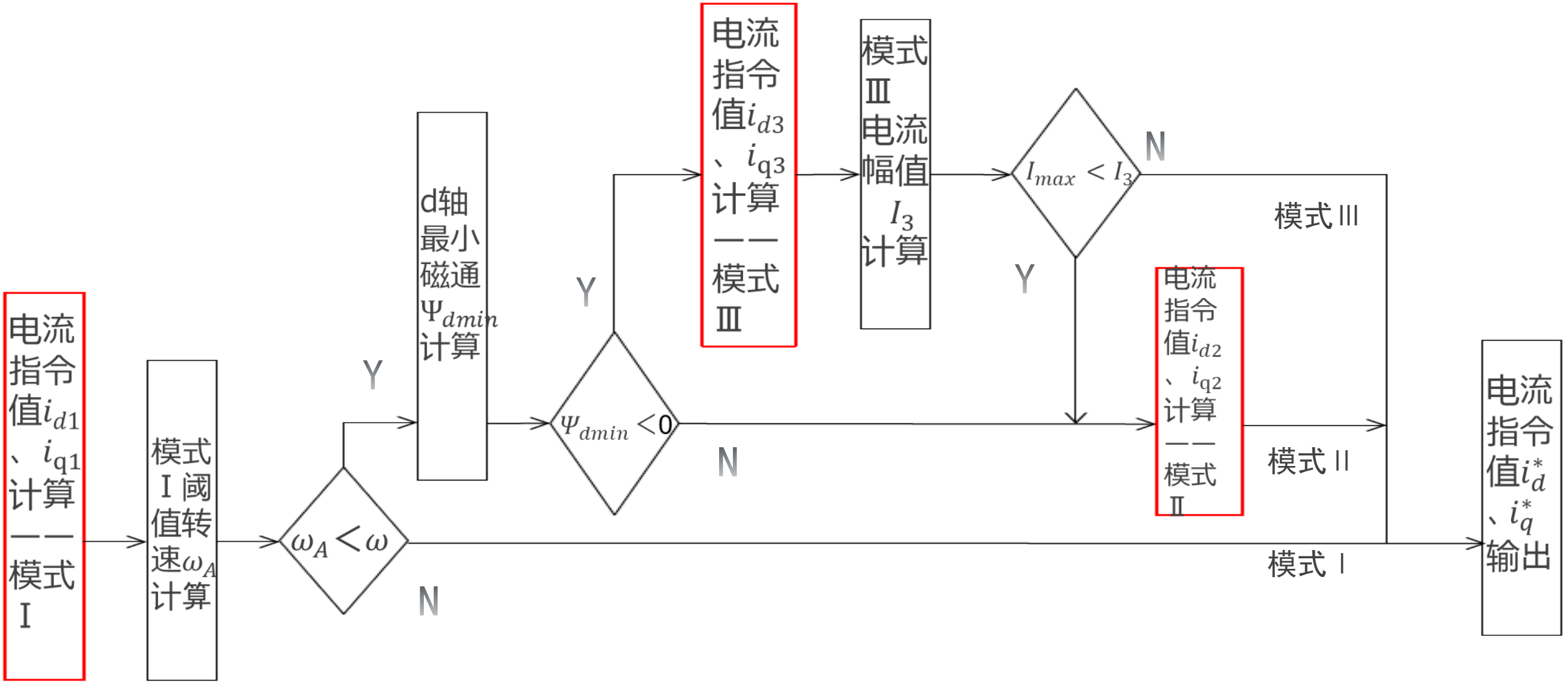
最大输出控制电流指令值计算流程



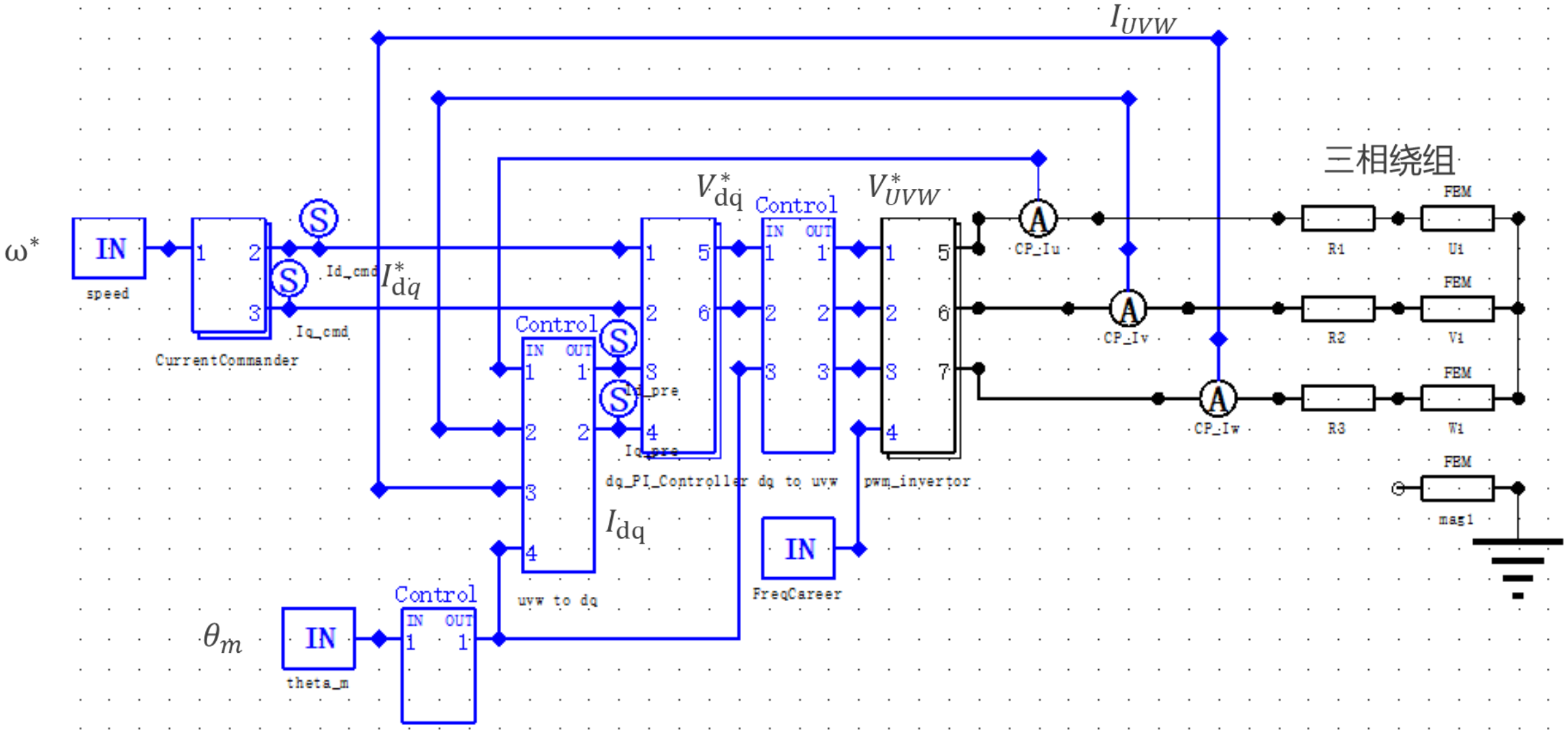


ω_A: Mode I threshold frequency (rad/sec),
ω_B: mode II threshold frequency (rad/sec)

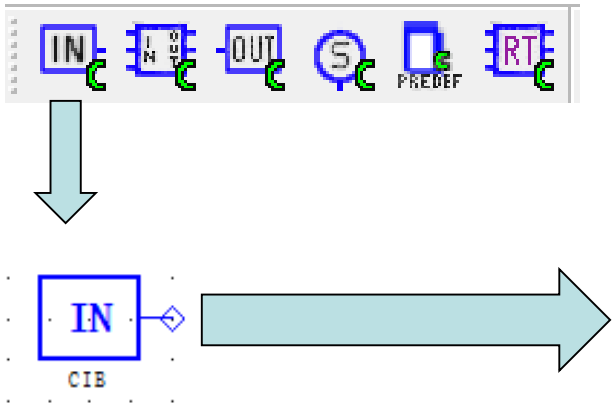
最大输出控制电流指令值计算流程

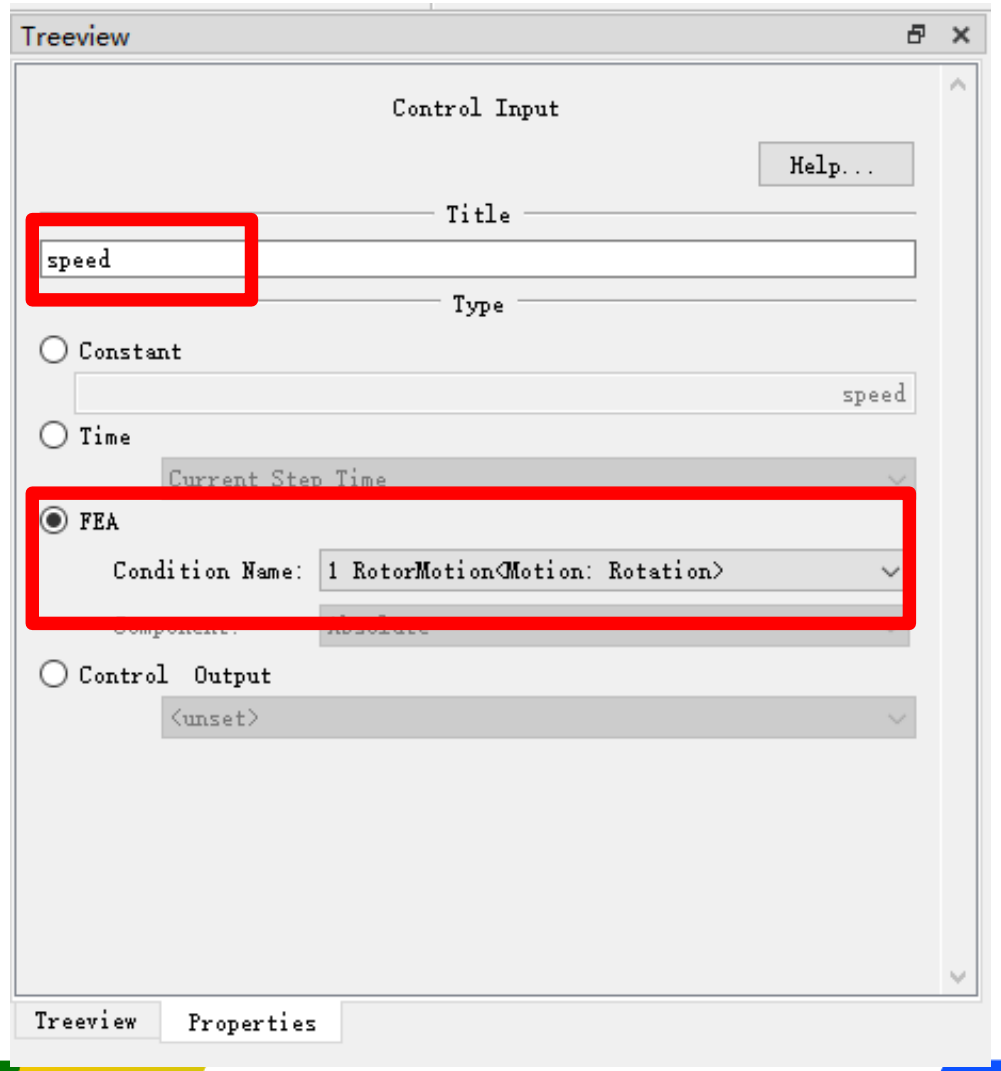


控制方法



转速输入

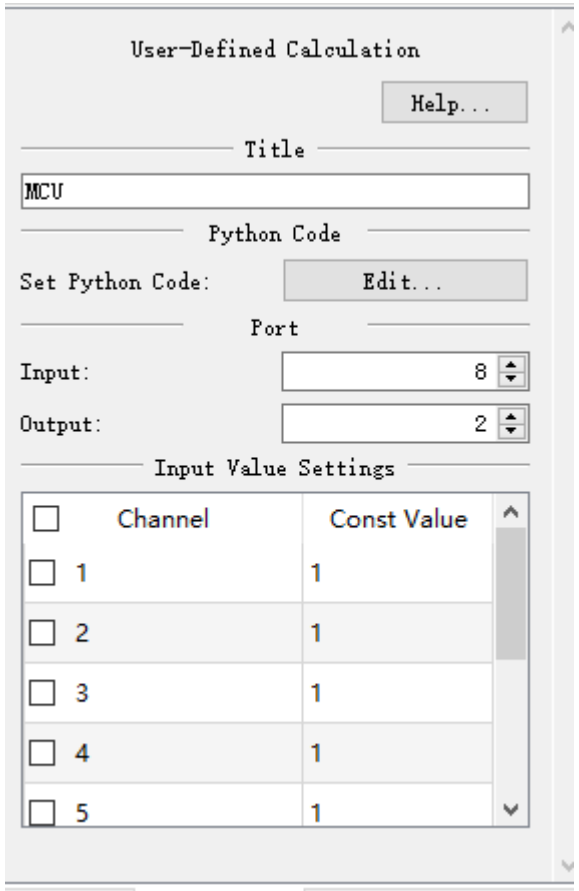


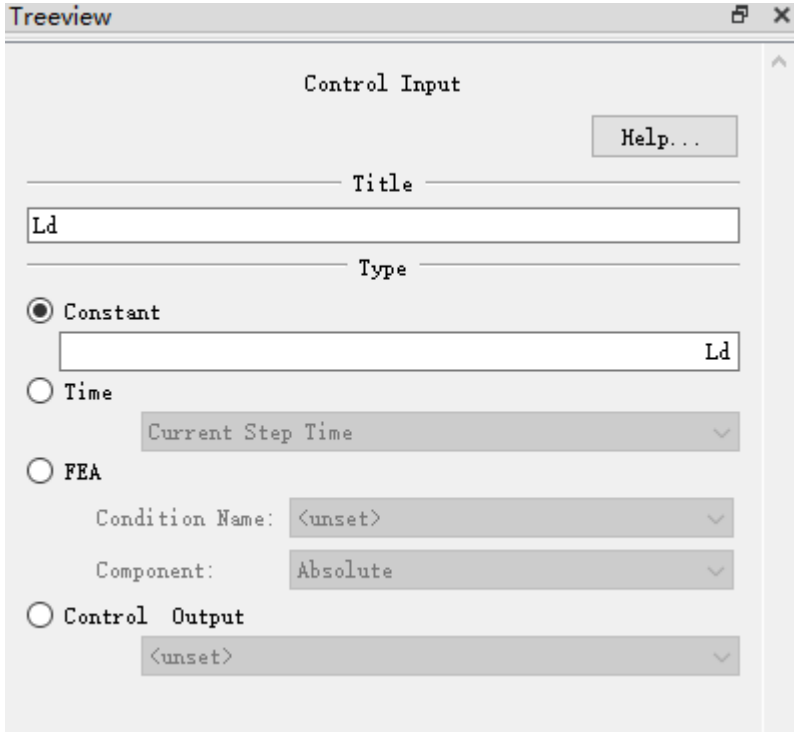


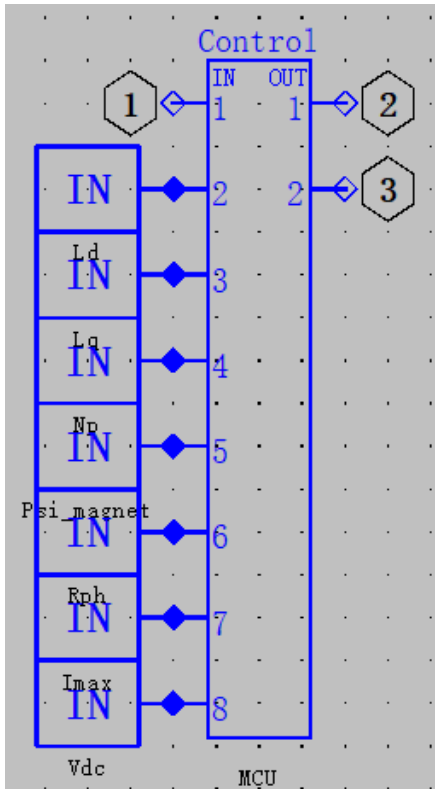
- Constant:恒定转速
- Time:输入时间步
- FEA：输入有限元结果
- Control Output:控制元件输出

自定义计算

- 选择用户自定义计算  ，定义Input为2，Output为8，命名为MCU。
- 添加输入模块，进行重命名，并将值设置为参数。

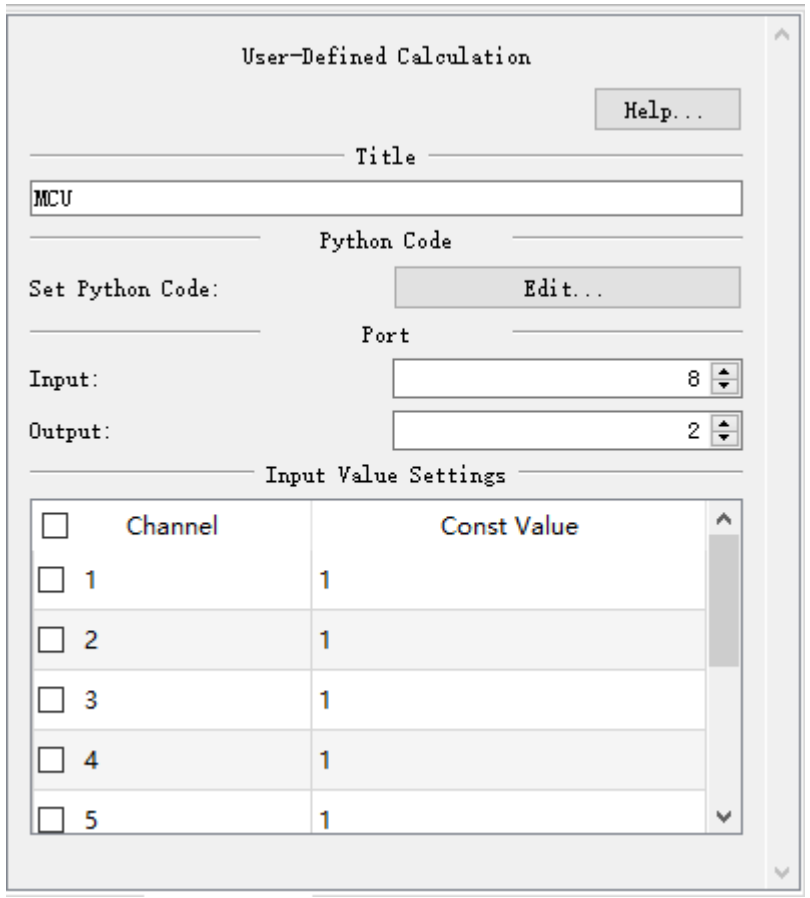






自定义计算

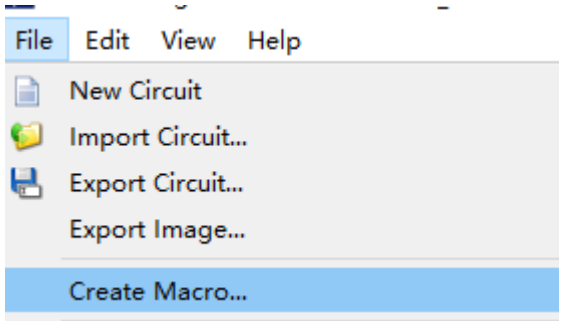
■双击自定义模块，点击Edit，使用Python语言进行模块定义。

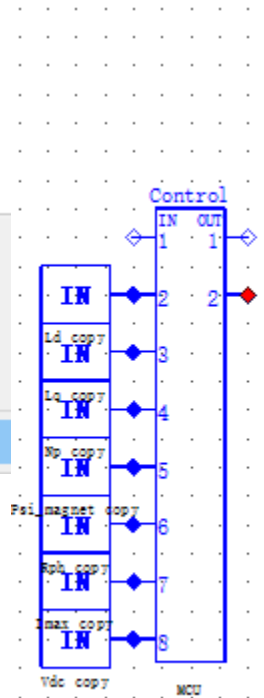


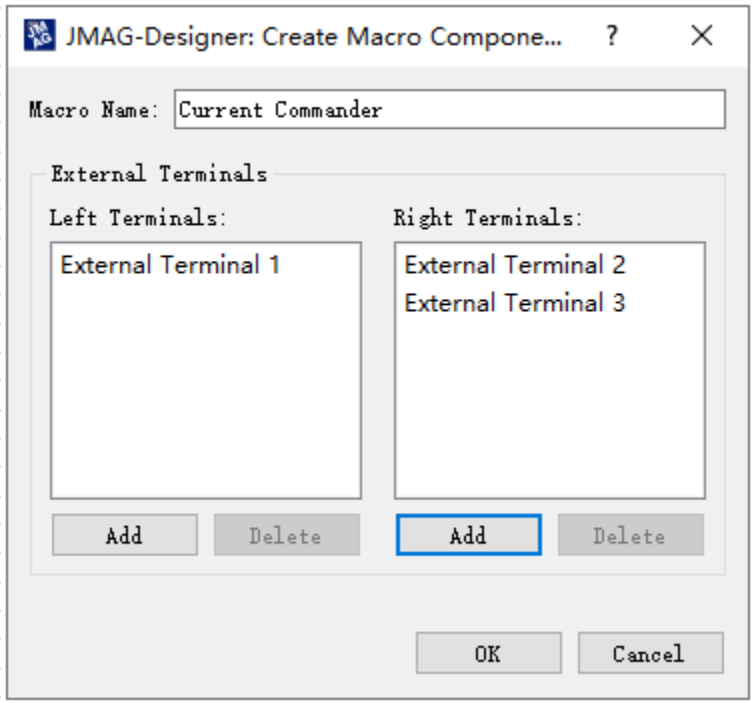

```python
import math
# -*- coding: utf-8 -*-
#Main routine
def jmag_control_block_MCU(vin):
    Vel_rpm = vin[0]
    Ld = vin[1]
    Lq = vin[2]
    Np = vin[3]
    Psi = vin[4]
    Rph = vin[5]
    Imax = vin[6]
    Vdc = vin[7]

    Vom = Vdc - Rph*Imax
    omega_e = (Vel_rpm/60.0) * (2.0*math.pi) * (Np/2.0)

    #-- Controller I : Maximum Torque Control
    K1 = 0.0
    dL = Lq - Ld
    if( abs(dL) > 0.0):
        K1 = Psi/dL
    Id_cmd = K1/4.0 - math.sqrt( K1*K1/16.0 + Imax*Imax/2.0)
    Iq_cmd = math.sqrt( Imax*Imax - Id_cmd*Id_cmd )

    #Calculate temporary variables
    K2 = Psi + Ld*Id_cmd
    K3 = Lq*Iq_cmd
    Omega_base = 0.0
    if( abs(K2) > 0.0):
        Omega_base = Vom / math.sqrt( K2*K2 + K3*K3 )
```

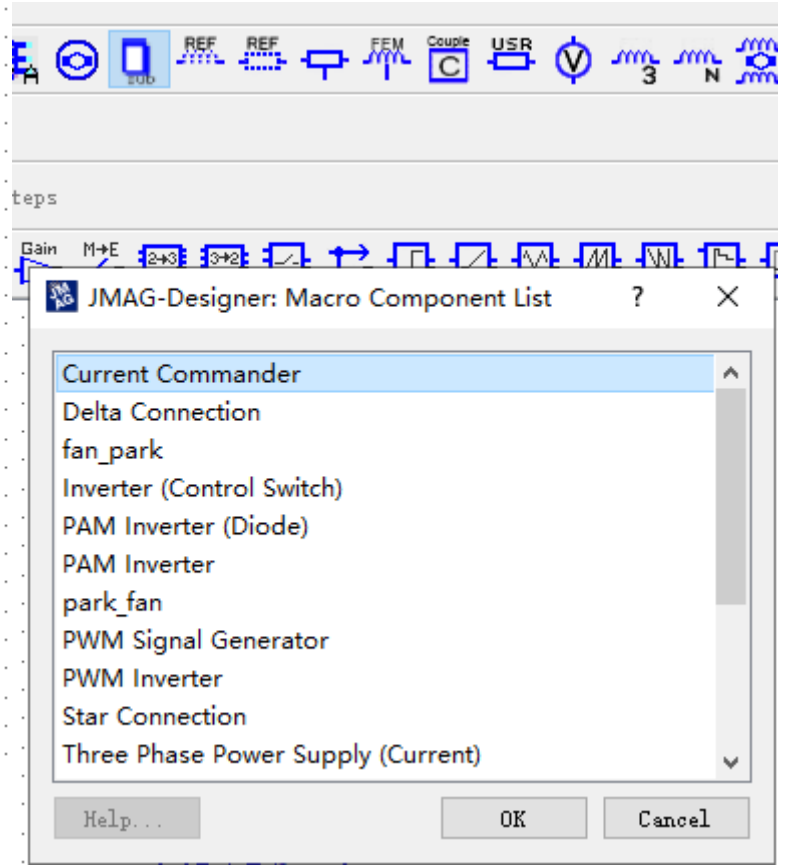
自定义计算

■定义完成之后，封装为一个模块。选择File，点击Create Macro。在出现的对话框中，首先选中电路中左端点，之后点击Left Terminals下方Add，增加左端点。同样方式增加右端点，进行重命名为Current Commander，之后点击OK。在Macro Component List中出现封装好的模块。



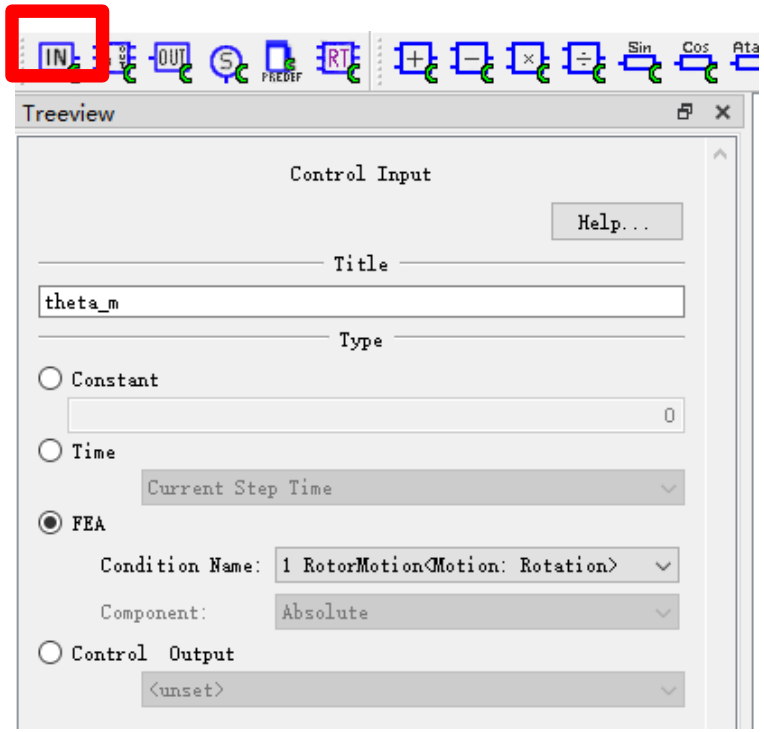


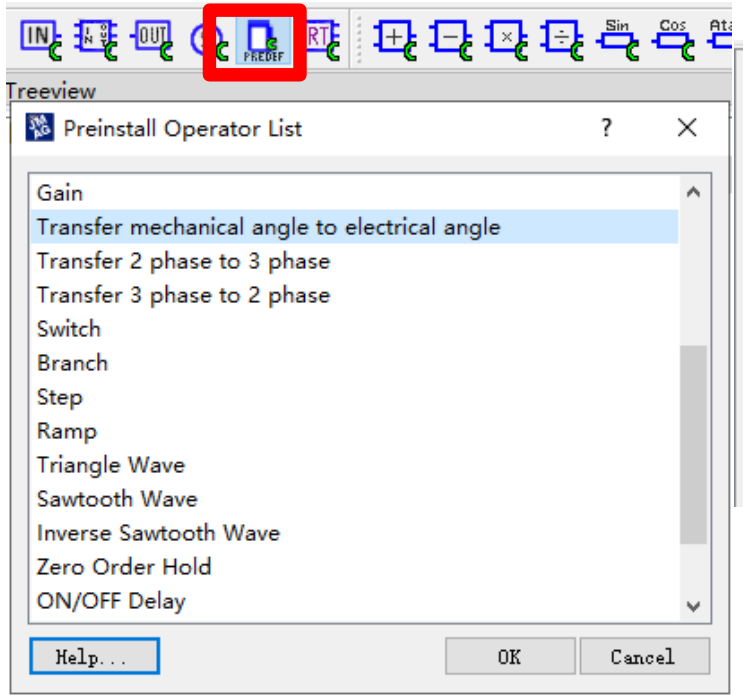


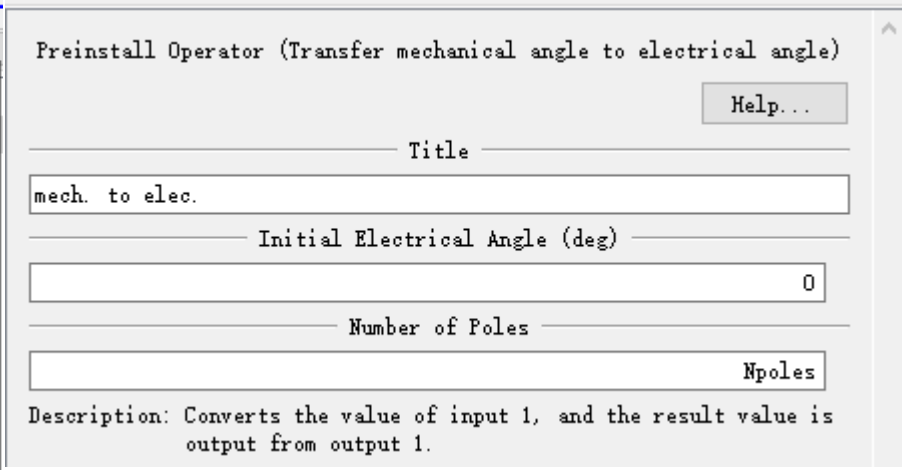


转子角度设置

■选择输入模块，并重命名为theta_m，选择输入至为FEA，来自于转动条件。预定义模块中选择机械角度转换电角度。极数设置为参数Npoles。

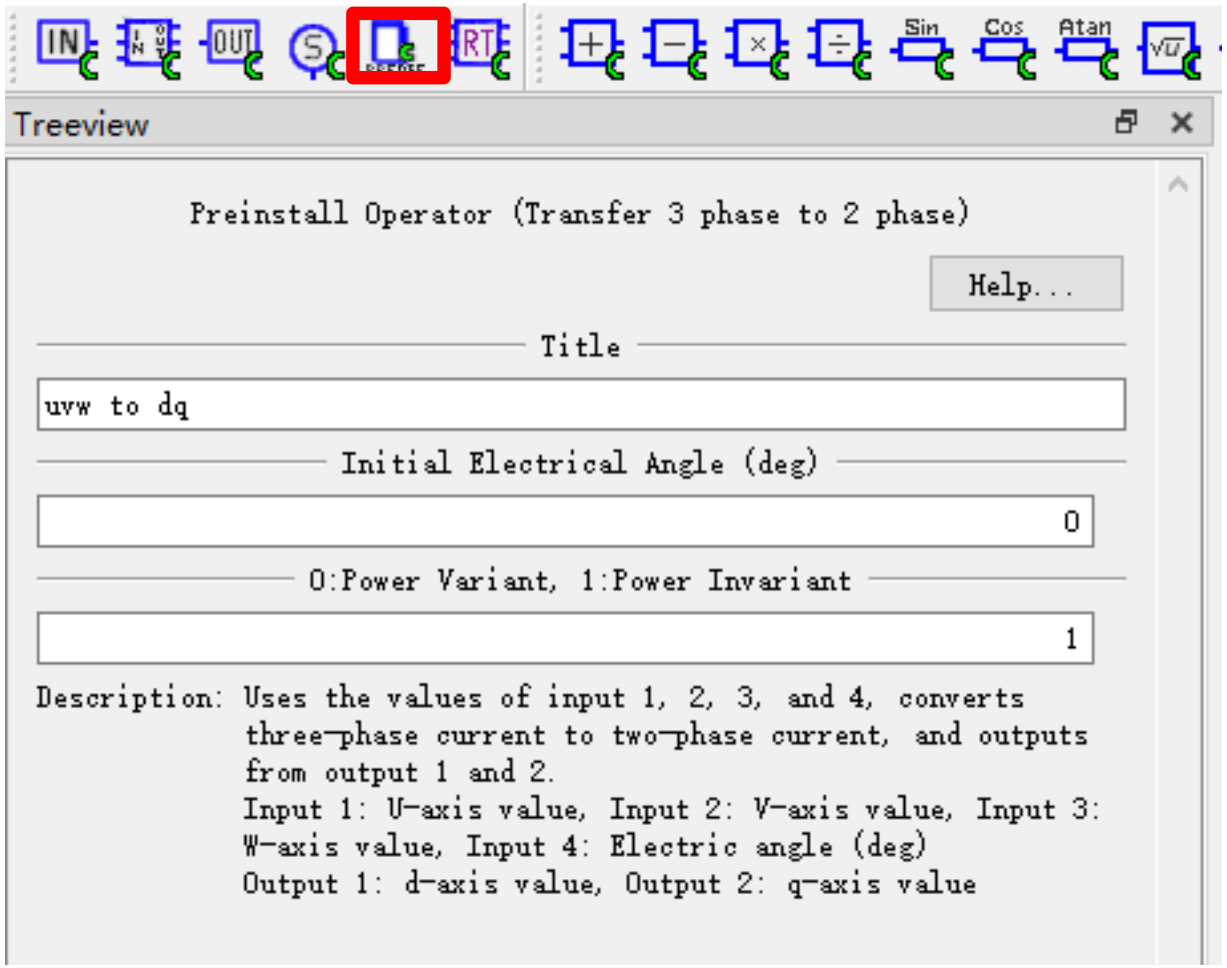






三相转dq设置

■预定义模块中选择三相转两相模块，设置为1恒功率。



$$\begin{Bmatrix} Y_d \\ Y_q \end{Bmatrix} = K \times \begin{bmatrix} \cos\Theta & \cos\left(\Theta - \frac{2}{3}\pi\right) & \cos\left(\Theta + \frac{2}{3}\pi\right) \\ -\sin\Theta & -\sin\left(\Theta - \frac{2}{3}\pi\right) & -\sin\left(\Theta + \frac{2}{3}\pi\right) \end{bmatrix} \times \begin{Bmatrix} X_u \\ X_v \\ X_w \end{Bmatrix}$$

Y_d, Y_q: d-axis and q-axis physical quantities after conversion

K: Real number (Power variant: $K = 2/3$, Power invariant: $K = \sqrt{2/3}$)

Θ, Θ_{rad}: Electric angle (rad)

$\Theta_{rad} = \theta_{deg} \times \pi / 180 \quad \theta_{deg} = \theta_e + \theta_{e0}$

θ_{deg}: Electric angle (deg)

θ_e: Input electric angle (deg)

θ_{e0}: Initial electric angle (deg)

X_u, X_v, X_w: U-phase, V-phase, and W-phase physical quantities prior to conversion

PI控制

i_d^*:指令d轴电流

i_d ：d轴电流

i_q^*:指令q轴电流

i_q ：q轴电流

V_d^*:指令d轴电流

V_q^*:指令q轴电流

L_d ：d轴电感

L_q ：q轴电感

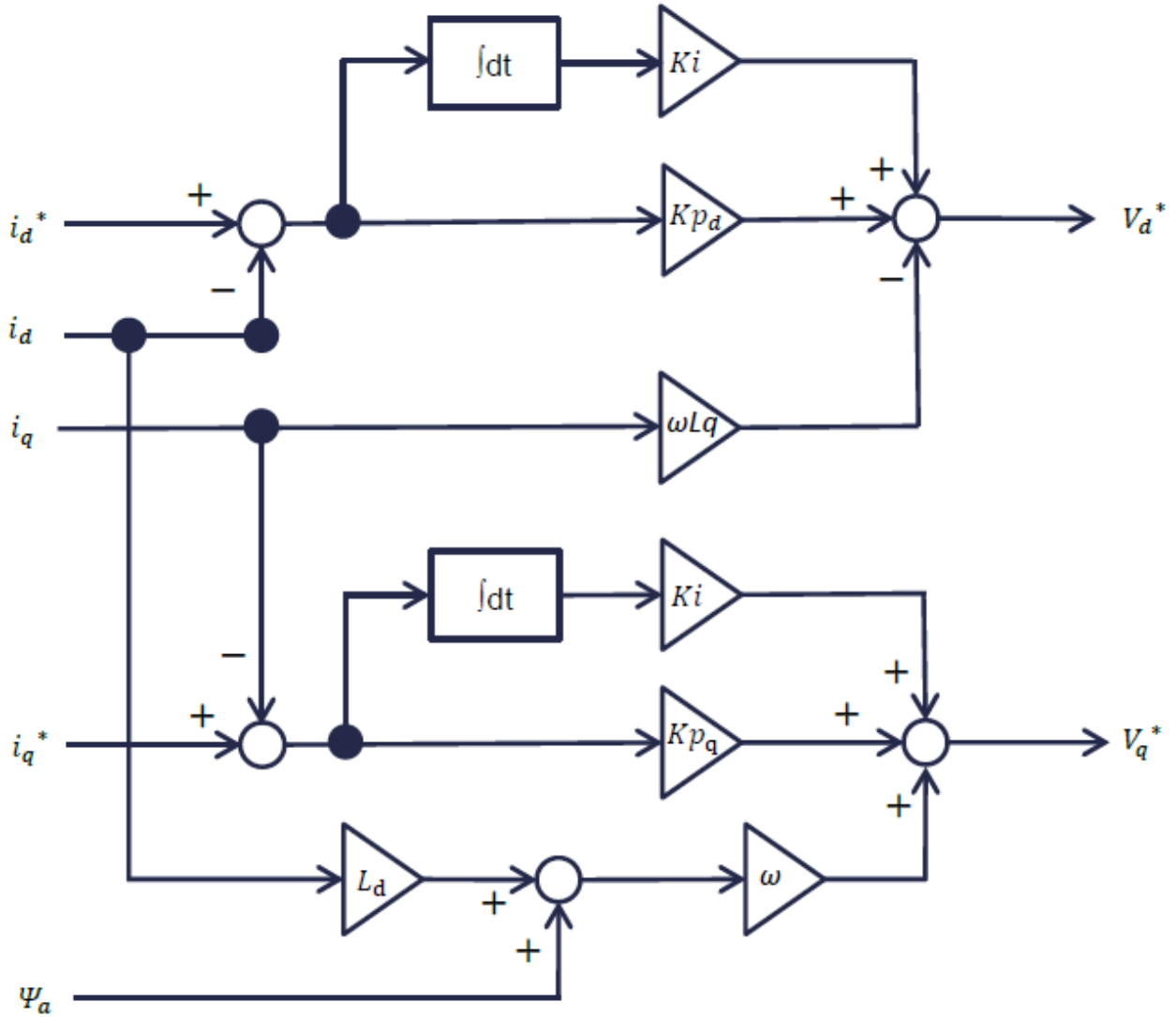
ω ：转动角速度

Ki:积分比例常数

K_{P_d}:d轴电流比例常数

K_{P_q}: q轴电流比例常数

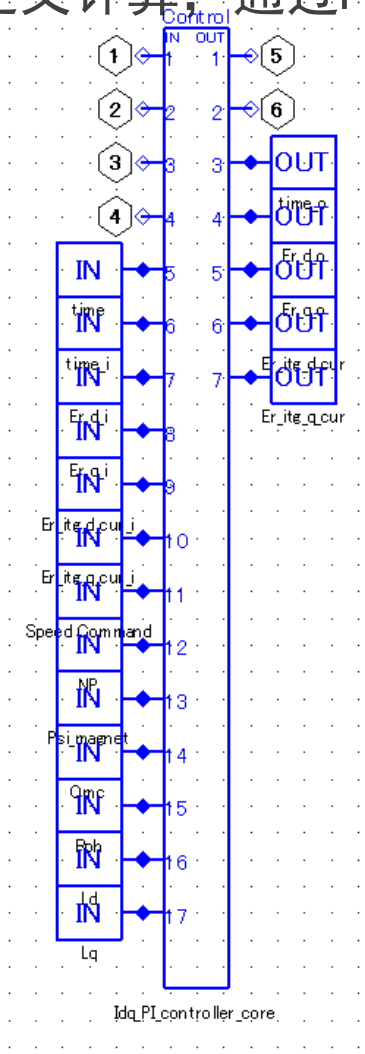
ψ_a:dq坐标系空载磁通



PI控制

■PI控制器同样使用用户自定义计算，通过Python语言进行功能设置。

- Id Iq指令值
- Id Iq反馈值
- 额定转速
- 极数
- 无负载磁铁磁通
- 相电阻
- 电流控制周期
- Ld Lq
- 反馈积分项

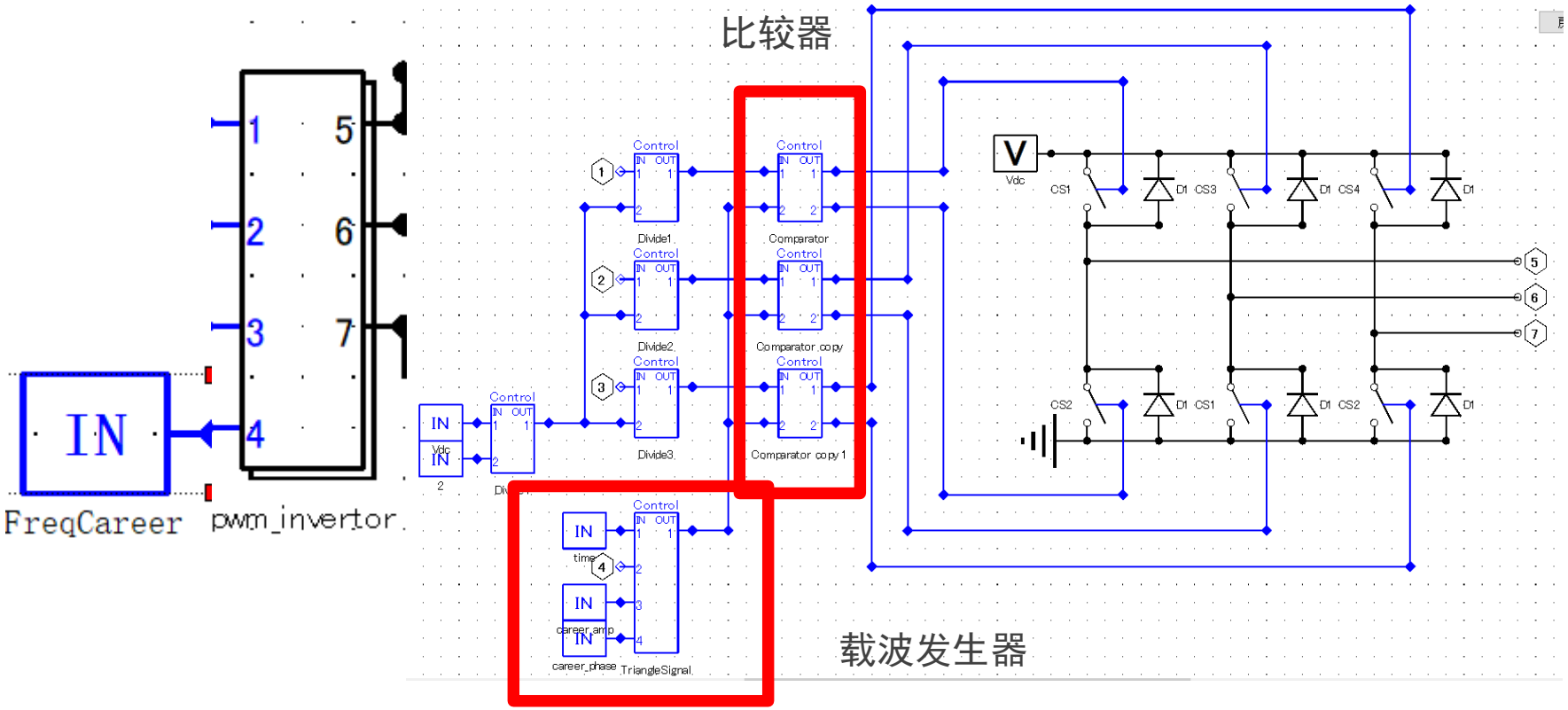



```
# ################ Calculate voltage command
#Calculate temporary variables
omegae = Vel_rpm/60.0 * 2.0*math.pi * (Npoles/2.0) #Angular velocity
dt = time - time_pre #Time interval(sec)

#Calculate Gains
Kp_d = Omc*Ld          P增益
Kp_q = Omc*Lq          I增益
Ki   = Omc*Rph*10

#Calculate Voltage Command                    Δid
Er_d = Id_cmd - Id_pre #Error of Id
Er_q = Iq_cmd - Iq_pre #Error of Iq           Δiq
Er_itg_d_cur = Er_itg_d + (Er_d + Er_d_pre)/2.0 * dt #integrate the error of Id   ∫id
Er_itg_q_cur = Er_itg_q + (Er_q + Er_q_pre)/2.0 * dt #integrate the error of Iq   ∫iq
Vind_d = -omegae*Lq*Iq_pre #Induced voltage by Iq
Vind_q =  omegae*(Ld*Id_pre + Psi) #Induced voltage by Id

Vd_cmd = Kp_d*Er_d + Ki*Er_itg_d_cur + Vind_d   Vd
Vq_cmd = Kp_q*Er_q + Ki*Er_itg_q_cur + Vind_q   Vq

vout = (Vd_cmd, Vq_cmd, time, Er_d, Er_q, Er_itg_d_cur, Er_itg_q_cur)
return vout
```

Vd、Vq指令值输出

PWM控制

■PWM中载波频率设置为参数Cfreq。



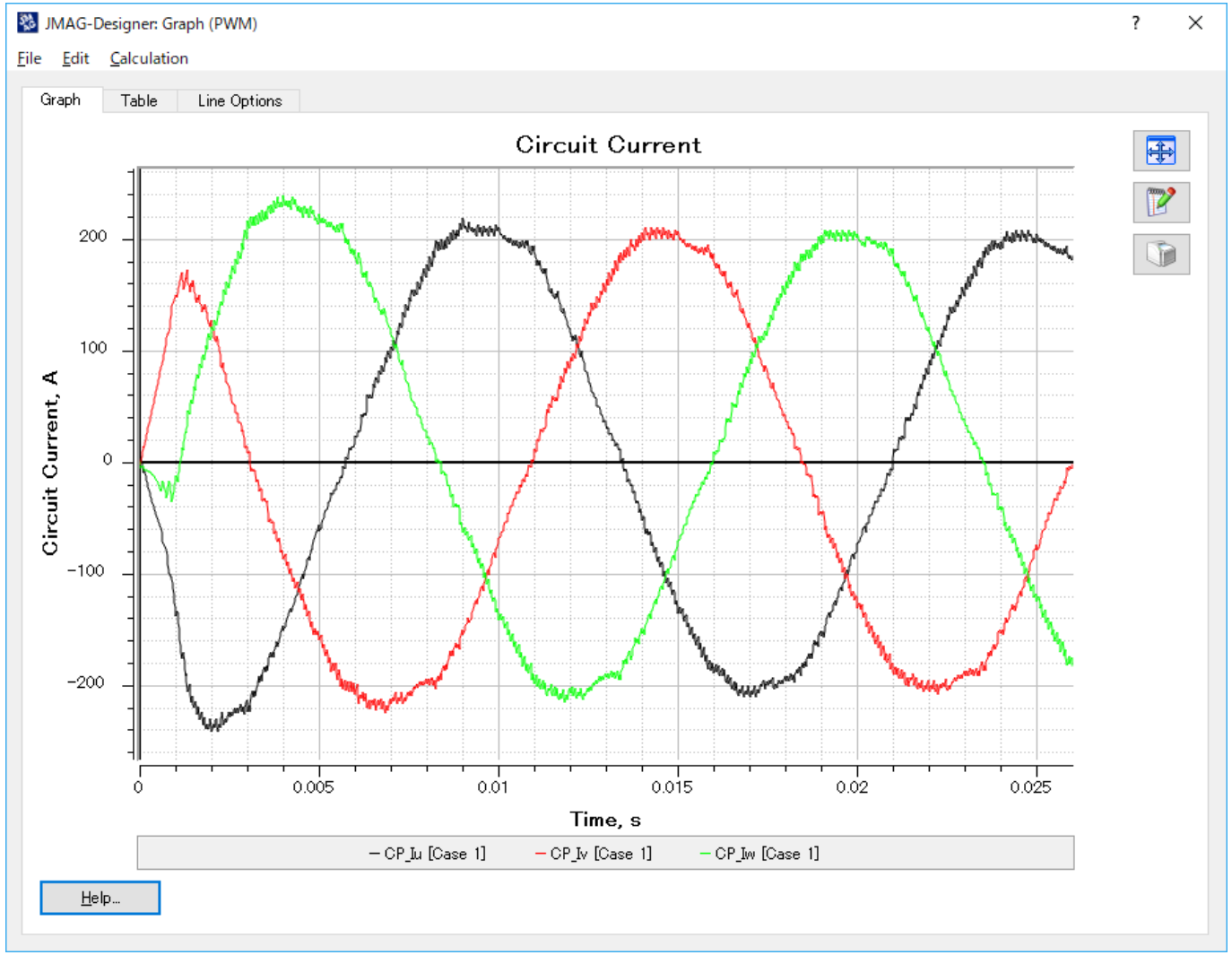
载波发生器

```
import math
# -*- coding: utf-8 -*-
#Main routine
def triangle_signal(vin):
    time = vin[0]
    freq = vin[1]
    amp = vin[2]
    t0 = vin[3]

    T = 1.0/freq
    tl = (time + t0) % T

    if tl <= T/4:
        d = 4.0 * amp * freq * tl
    elif tl <= T*3.0/4:
        d = - 4.0 * amp *freq * tl + 2.0 * amp
    else:
        d = 4.0 * amp * freq * tl - 4.0 * amp

    vout = (d,)
    return vout
```

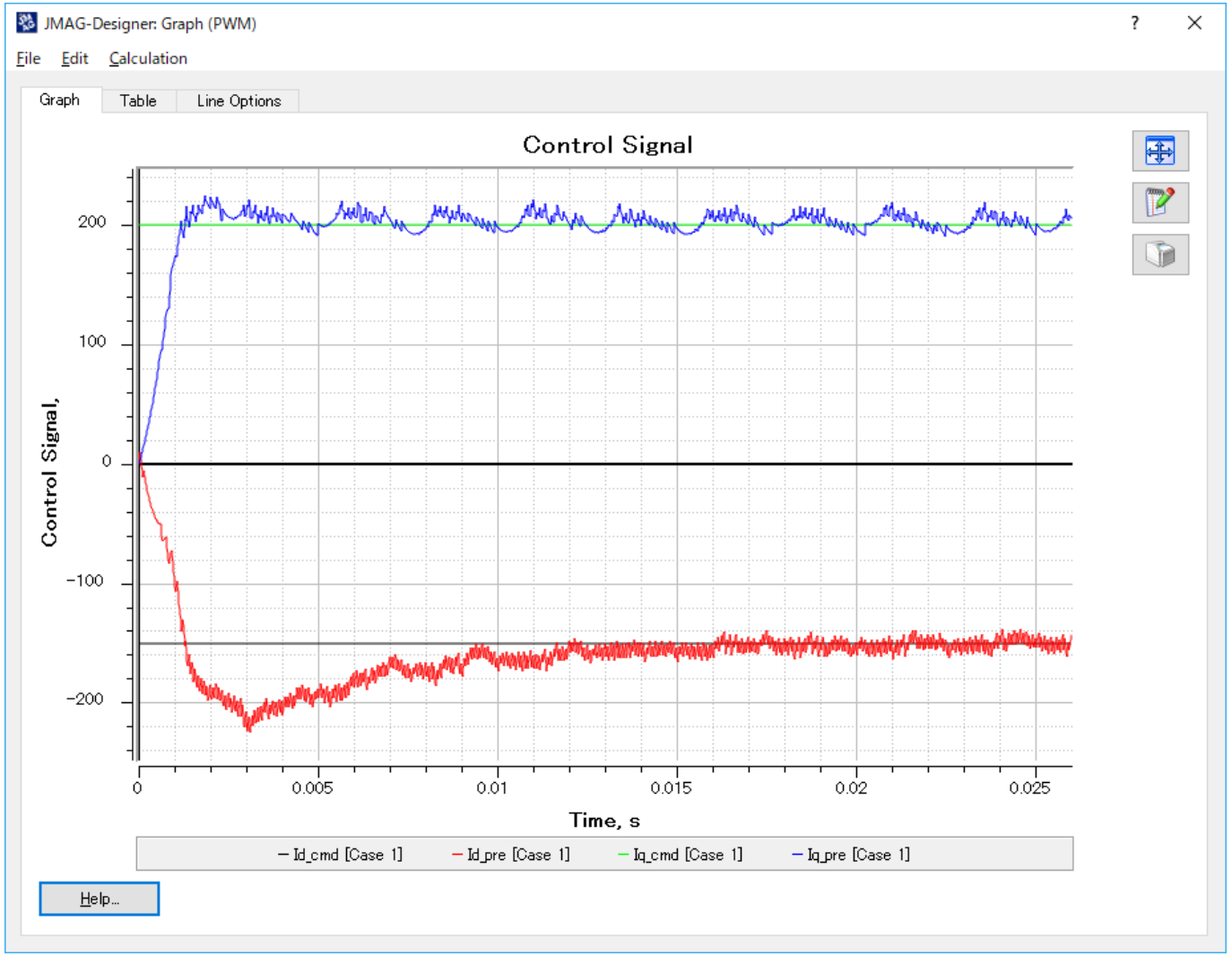
目录

- JMAG控制系统
- 元件介绍
- 模型介绍
- 控制原理
- 仿真结果
- 总结

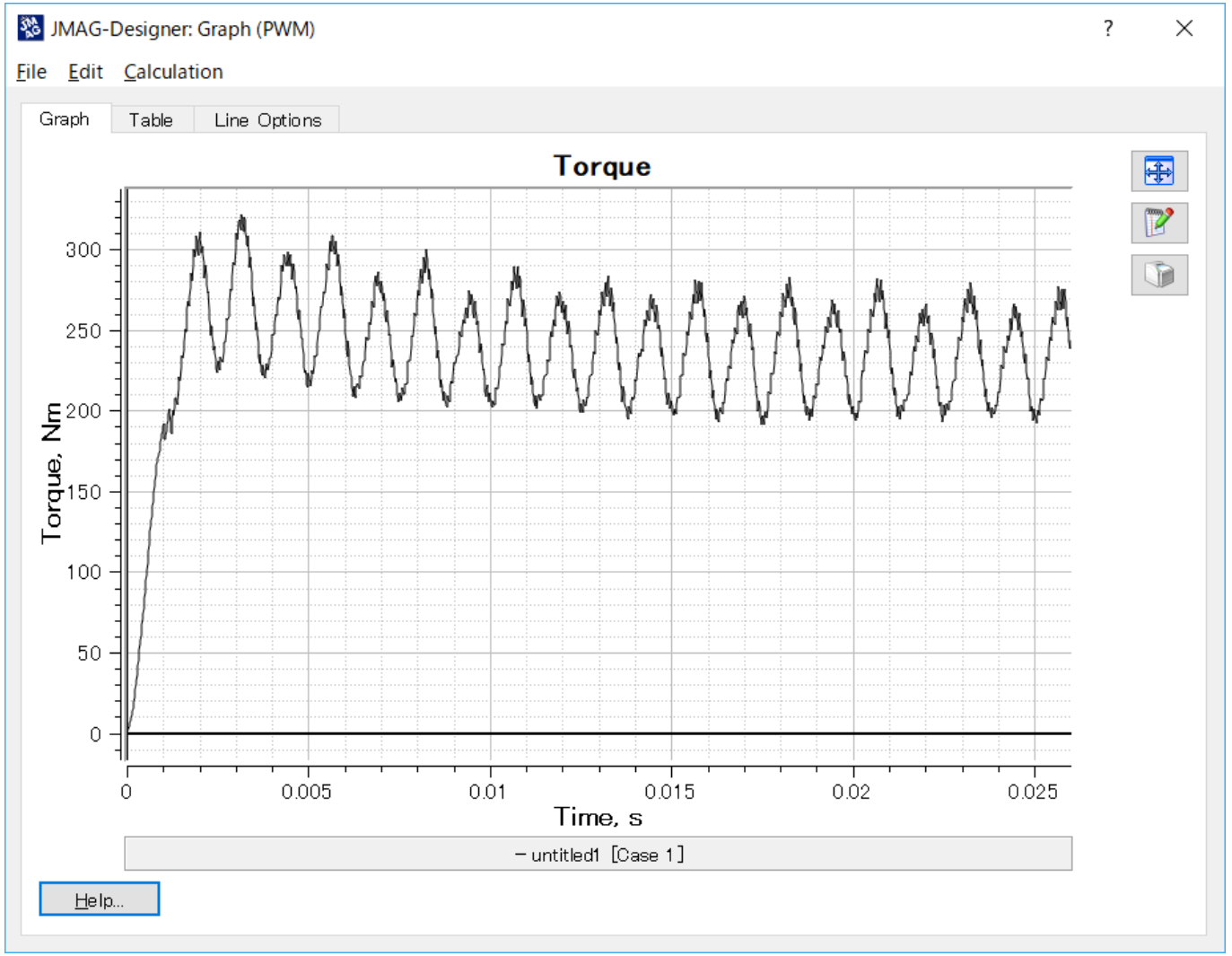
仿真结果



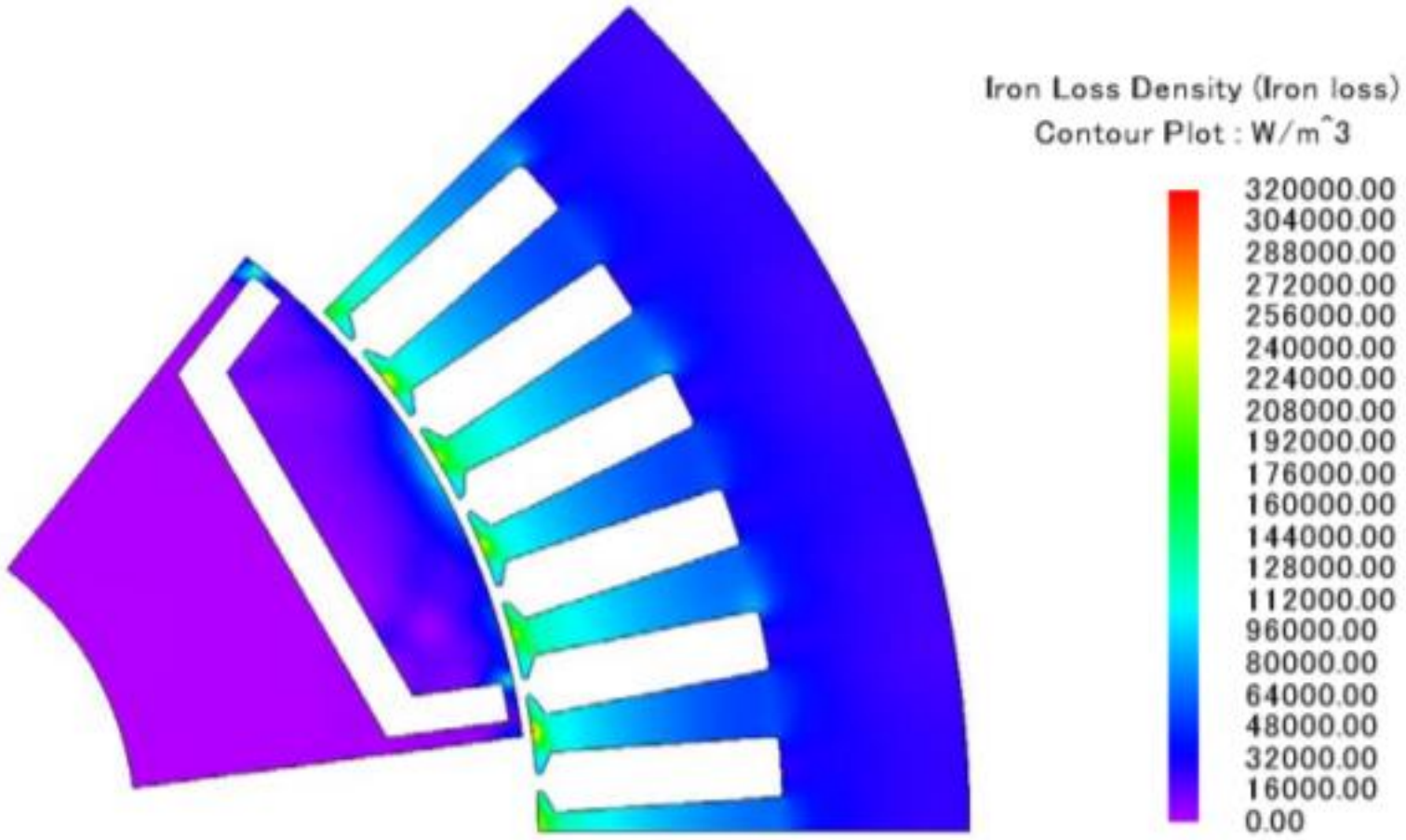
仿真结果



仿真结果



仿真结果



目录

- JMAG控制系统
- 元件介绍
- 模型介绍
- 控制原理
- 仿真结果
- 总结

总结

- JMAG控制电路设置与其它常用控制软件类似，上手快
- JMAG-RT文件在控制系统中调用会加快计算速度
- JMAG控制电路自定义模块使用Python语言进行设置，需要有一定语言基础
- JMAG控制电路的增加，有利于精确得出电机效率Map，可以考虑到谐波损耗

艾迪捷信息科技(上海)有限公司

扫一扫，关注艾迪捷
第一时间获得更多产品介绍/成功案例/市场活动

联系我们

——软件试用/报价/技术培训/专题活动

- 网站: https://www.idaj.cn/
- 邮箱: idaj.marketing@idaj.cn
- 电话: 021-50588290；010-65881497