Real-time Model for Motor Control Coursework

Alecksey Anuchin, Dmitriy Savkin, Yulia Khanova, Daria Grishchuk
Electric Drive Department
of National Research University “Moscow Power Engineering Institute”
Moscow, Russia
Anuchin.Alecksey@gmail.com

Abstract — **This paper considers the model of different types of electric drives which is designed as a project for Code Composer Studio for TMS320F28xx microcontrollers and is used in Motor Control course for coursework. The review of different solutions for motor control education course is given and the demands to the model are stated. The description of the model is given and the sample of the control system with experimental results is shown. The link to the project for Code Composer Studio and its description is given.**

Keywords — motor control, real-time simulation, education, programming, software development, coursework

I. INTRODUCTION

Motor control training course for master degree students in Moscow Power Engineering Institute covers various types of control system for different types of electrical machines. During this course students do the motor control coursework. Recently the coursework was done using Simulink MATLAB but this tool is much different from real control system design process. The most of the modern control systems are implemented on microcontrollers and we also have a course of microcontroller real-time programming for master degree students. Our department has good collaboration with Texas Instruments and we have an educational center “Texas Instruments – MPEI” equipped with the modern microcontrollers of this company and motor control kits with small PMSMs.

So there are several possible ways to make coursework closer to practice:

- to use real-time MATLAB target support with power converter and motor [1];
- to use motor control kit with small motor [2, 3] and software development system like Code Composer Studio with ControlSUITE [4];
- to use microcontroller with power converter and standard motor and software development system like Code Composer Studio with ControlSUITE;
- to use microcontroller with model of the power converter and motor implemented on FPGA [5, 6];
- to use microcontroller with model of the drive and software development system like Code Composer Studio.

The main problem if we use the real power converter and motor is the possibility of unrecoverable fault in case of software error in code or algorithm. This can be solved if we will use motor control kit like DRV8301-69M-KIT. It is more reliable than standard power converters due to low voltage operation and small power. On other hand the properties and behavior of the small motors are very different from the motors of rated power from 1kW and higher. So the model with reference to coursework is the best solution for large audience of students. It can be implemented on microcontroller or FPGA. The solution with FPGA is closer to the reality [5, 6]. The software project for power converter or FPGA has no significant differences but in this case we need a separate FPGA device. If we want more simple solution then the model of the electric drive may operates on the same microcontroller.

This solution can be configured to any motor type and rated power and the development of the control system is very similar to the real development process. The only problem is that the student does not see the real motor and can’t use the real measurement devices for tests but in fact it is possible to move working student’s project to the real power converter after debug on the software model.

II. THE DESCRIPTION OF THE MODEL

The model of the electric drive must have the same kind of control inputs and same type of sensor outputs like the real system. That means that control of PWM generator must be done in ticks of PWM timer for duty cycles and dead-time generation. The control system core and the model can be situated in the routine executed by PWM timer interrupt. The model must produce data from sensors in their natural format like ADC code with noise.

Model is built in the project for Code Composer Studio integrated development environment. It is written in C and given to a student as a compiled binary file. Binary file instead of source code file is used to hide the motor parameters from the student. So he can only measure them making tests with the model.

The considered model has no sensor service routines like QEP timer compare interrupt for speed estimation. These routines are very specific and hardware dependent.

Today existing model includes three types of electrical machines: direct current motor, permanent magnets synchronous motor and induction motor. Every type of the machine contains ten parameters sets for machines from 1 to 30 kW. The connection of the PMSM to the power converter is shown in Fig. 1a (induction motor is connected in a similar way) and connection of the direct current motor is shown in Fig. 1b.

All inputs and outputs of the model are collected together in one structure called "*drive*". The inputs are:

- *n* — number of the motor type and parameter set (0x for induction motors, 1x for PMSMs and 2x for direct current motors) from 1 to 30;
- *cmpr1*, *cmpr2*, *cmpr3* — duty cycles for each arm of the converter in ticks of PWM timer;
- *tpr* — PWM timer period in ticks (timer counts in up/down mode);
- *dt* — dead-time duration in ticks of PWM timer;
- *load* — load torque at the motor shaft.

The model outputs are:

- *adcSpeed* — signal from tachogenerator in ADC scale;
- *qepCounter* — position counter for incremental encoder;
- *hallSensor* — position from Hall-sensor;
- *iA*, *iB* — signals from current sensors in ADC scale;
- *fault* — state of the build-in protections: overcurrent, high speed;
- *time* — time in seconds from the beginning of simulation.

The inverter represented as zero-order hold. Electrical potential for each phase is calculated for one PWM cycle using equation:

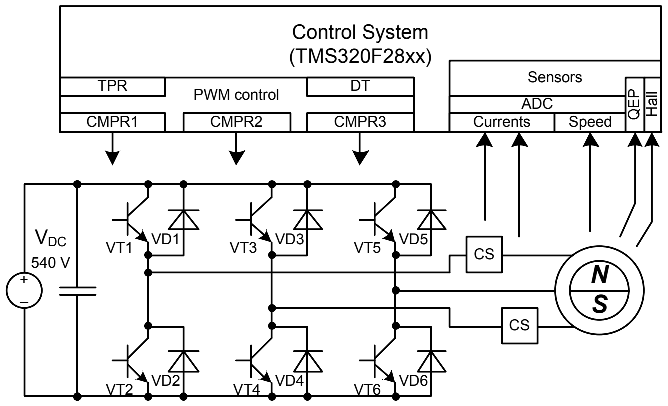
$$\varphi_x = V_{DC} \cdot \left(\left(cmprx - 0,5 \cdot \operatorname{sign}(i_x) \cdot dt \right) / tpr \right), \qquad (1)$$

where V_{DC} is the DC-link voltage, $cmprx$ — the duty cycle for current PWM cycle in ticks, i_x — current in the phase x, dt — dead-time duration in ticks, tpr — timer period in ticks. This approach gives good results only if the current in the phase is not equal to zero. Then the phase voltages are calculated using known potentials.

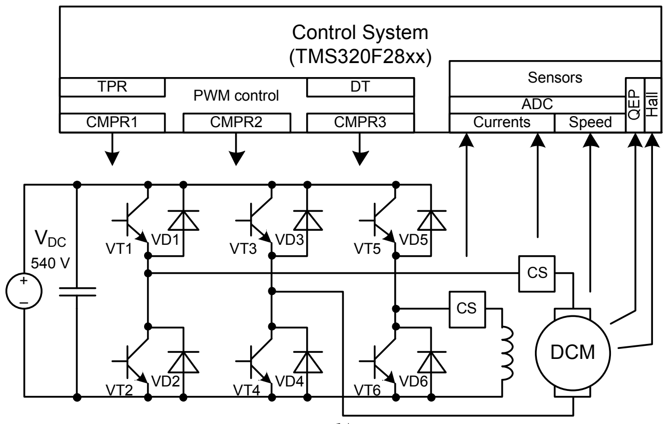
The motors are represented with two phase model which is calculated using Runge-Kutta second-order method in a single precision floating point. The direct current motor has saturation in the field winding; induction and permanent magnet machines are linear.

Currents and speed are measured with current sensors and tachogenerator and their values are available as a code from ADC. The ADC noise is added to the actual value using equation:

$$ADC_x = 0\text{xFFF0} \,\&\, \left(k_{sensor} \cdot x + 0\text{x7FE0} + 16 \cdot \operatorname{rnd}(3) \right), \quad (2)$$



a)



b)

Fig. 1. Power converters and their connection to the motors and control system

where x — the exact value of the measured signal from the model, 0x7FE0 — offset, k_{sensor} — ADC scale factor.

The rotor position can be sensed using incremental encoder or Hall-sensor. Incremental encoder position is stored in *qepCounter* variable and varies from zero to number of pulses per revolution. Hall-sensor state can be monitored via *hallSensor* variable.

The model has current and speed protections and indicates about these faults in *fault* variable. During the debug process student can refer *time* variable of the model time. The *time* variable is needed if the model and control system execution time is greater than PWM cycle. This may happens if student will set high PWM frequency or the fixed point microcontroller is used to run the simulation.

III. The Task for the Coursework

At the beginning of the course lecture gives every student a task for coursework. The type of the motor and the control structure depends on the topic of his future diploma. The possible control structures are:

- for direct current motors — nested loop control, nested loops with field weakening, hysteresis control;

- for permanent magnet synchronous machines — brushless DC motor, flux-vector control, sensorless flux-vector control;
- for induction machine — v/f control, flux-vector control, sensorless flux-vector control, direct torque control and etc.

The fulfillment of the coursework starts from motor parameters estimation. Students get acquainted with the model, calculate current sensor gain and make some test to determine resistances and inductances of the windings, permanent magnet flux linkage (for PMSM) and moment of inertia.

The next step is implementation of the motor and the control system in Simulink MATLAB. This step is necessary to be sure that the considered control structure works and to test its performance and accuracy.

On the next step students can choose whether to use floating point and SI units or fixed point and PU [7] for control structure implementation in microcontroller. They get extra scores for fixed point implementation. Then they write a program using C-language, test its behavior and compare the results with achieved in Simulink MATLAB.

IV. An Example of the Control System and Experimental Results

One of the tasks this year was the development of the control system for PMSM with hysteresis current controllers and speed loop with PI-controller.

The estimation of the motor parameters can be done in a few steps. First test is a voltage step response. Student applies voltage to the motor and observes the current curve (see Fig. 2). In steady state they can calculate winding resistance using known applied voltage and measured current. The form of the current curve can be used to identify time constant and then to calculate inductance of the winding.

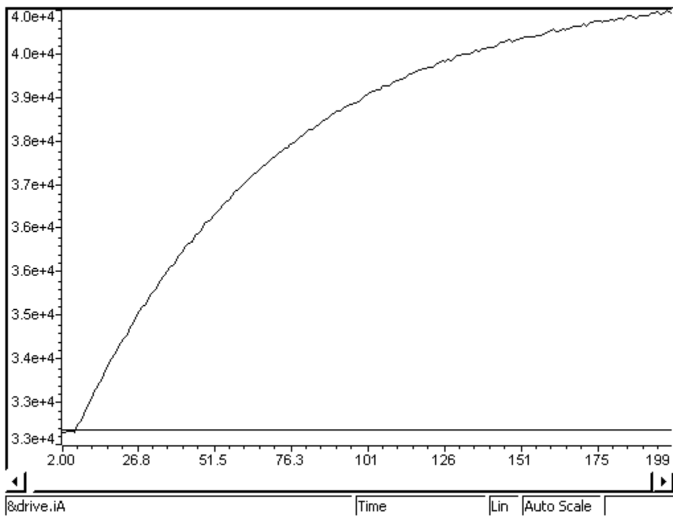


Fig. 2. Current curve for the step response for motor winding parameter estimation (time represented in PWM cycles, current in ADC scale)

On the next step the flux linkage of the permanent magnet can be obtained. Student applies some load torque to the shaft of the motor while voltage is still applied to it. This is done changing *load* variable. Then the load should be increased. The load change rate must be very slow so that during the stretching of the electromagnetic spring current should stay in a steady state. When the rotor turns to $\pi/2$ the applied torque and current can be used to obtain flux linkage of the permanent magnet Ψ_{PM}. Rotor position can be tracked by *qepCounter* variable.

After estimation of the motor parameters and designing a model of the drive and control system in Simulink MATLAB student should write a code of the control core in Code Composer Studio IDE and test his algorithms on a real-time model.

The control system structure is shown in Fig. 3. It contains PI controller in the speed loop. Hysteresis current controller uses the position feedback from the Hall-sensor.

Student should place the code of the control algorithm in timer interrupt routine **ControlInt**. A sample of this code is shown in Fig. 4. At first, ADC data must be converted to IQ format. After this conversion we will have three phase currents and speed in PU. On the next step there is PI speed controller which produces current reference for the hysteresis current controller. PI controller has an anti-windup circuit. At first, the value of controller output is calculated without saturation. Then the relation between saturated and desired output is calculated. This relation is used to correct the input error to avoid controller saturation and excessive growth of the integral component [8].

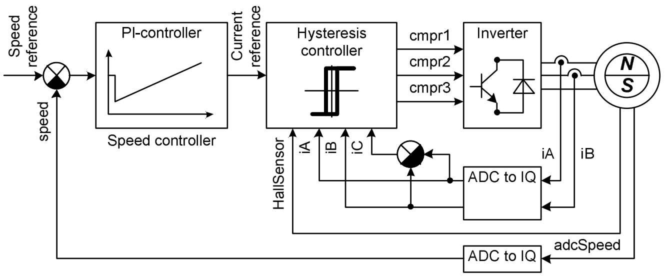
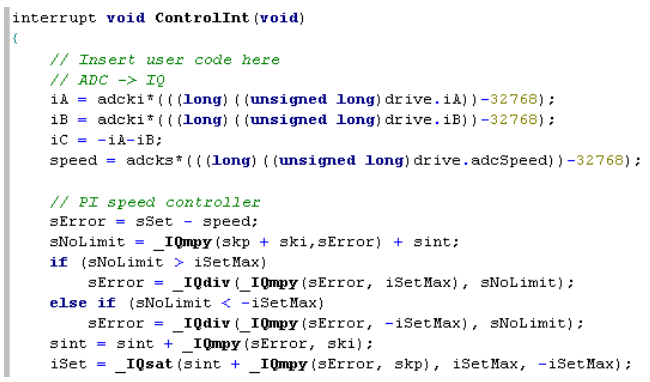


Fig. 3. Control system structure



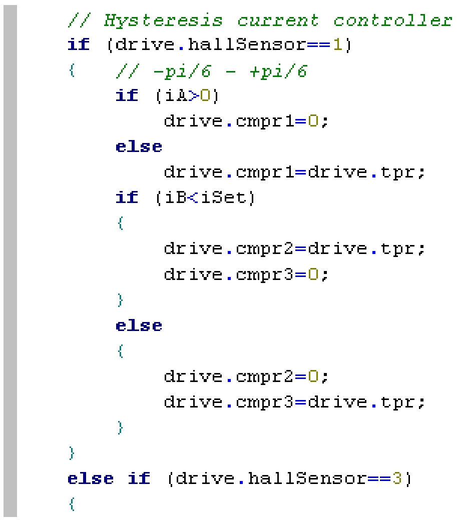
```
interrupt void ControlInt(void)
{
    // Insert user code here
    // ADC -> IQ
    iA = adcki*(((long)((unsigned long)drive.iA))-32768);
    iB = adcki*(((long)((unsigned long)drive.iB))-32768);
    iC = -iA-iB;
    speed = adcks*(((long)((unsigned long)drive.adcSpeed))-32768);

    // PI speed controller
    sError = sSet - speed;
    sNoLimit = _IQmpy(skp + ski,sError) + sint;
    if (sNoLimit > iSetMax)
        sError = _IQdiv(_IQmpy(sError, iSetMax), sNoLimit);
    else if (sNoLimit < -iSetMax)
        sError = _IQdiv(_IQmpy(sError, -iSetMax), sNoLimit);
    sint = sint + _IQmpy(sError, ski);
    iSet = _IQsat(sint + _IQmpy(sError, skp), iSetMax, -iSetMax);
```

Fig. 4. ADC data convertion and PI speed controller code

After the reference for the stator current is calculated hysteresis controller reads rotor position from *hallSensor* variable. There are six different position codes and there is a branch of code for each of them. The code for angle from $-\pi/6$ to $+\pi/6$ is presented in Fig. 5. For this sector the current of phase A is referenced to zero and the current for phases B and C would be similar with different signs. So controller reads the value only of the current in phase B and control inverter arms of phases B and C in complementary way. At the end the compare variables are loaded with zero or maximum value so the output voltage swings from zero to DC power supply.

The result of the motor start with maximum current reference is shown in Fig. 6. The plot shows the operation of the hysteresis current controller and the influence of EMF on it.



```
// Hysteresis current controller
if (drive.hallSensor==1)
{   // -pi/6 - +pi/6
    if (iA>0)
        drive.cmpr1=0;
    else
        drive.cmpr1=drive.tpr;
    if (iB<iSet)
    {
        drive.cmpr2=drive.tpr;
        drive.cmpr3=0;
    }
    else
    {
        drive.cmpr2=0;
        drive.cmpr3=drive.tpr;
    }
}
else if (drive.hallSensor==3)
{
```

Fig. 5. A fragment of the hysteresis current controller

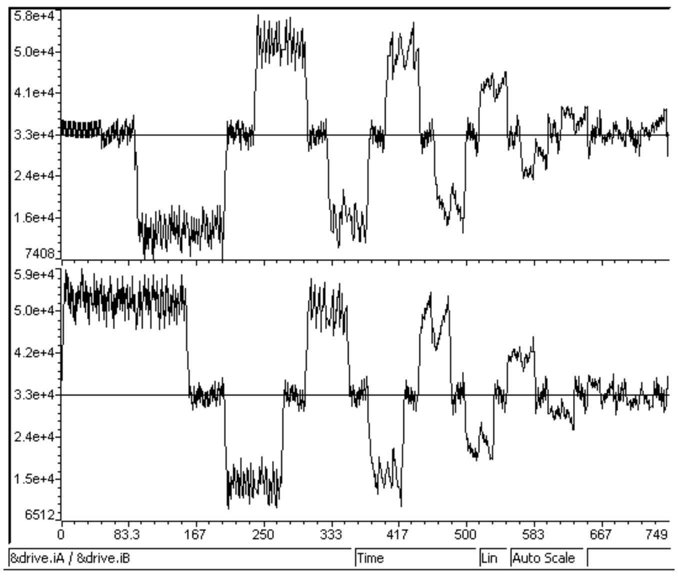


Fig. 6. Motor start with constant current reference

V. Conclusions

The considered real-time model was used for motor control coursework for four years. The model can run on a relatively cheap microcontroller boards (e.g. F28069 Piccolo controlSTICK for $39 from Texas Instruments) and gives the good approximation of the control system development process.

Most of the control systems that students considered in this coursework can be tested in instructional laboratory of Electric Drive Department of Moscow Power Engineering Institute. This laboratory equipped with power converters with configurable control systems and allows students to make test on real induction motors and direct current machines [9].

That coursework helps students to understand how different courses like control theory, power electronics, software design and other can be combined together in a real control system. Approximately 60% of the students use the newly acquired skills in their diploma works. This can be not only real-time control software design but controller schematic design, PCB routing, power converter and electrical machine design where the capabilities of the control system software must be taken into account.

Authors plan to upgrade the model with saturation effect for induction and synchronous machines, make a salient pole synchronous machine and DC-excited machine, implement models for switch-reluctance motor, synchronous reluctance motor and implement a model of rectifier, DC-link and a brake circuit.

References

[1] D. Hercog; M., Curkovic, K. Jezernik, DSP based rapid control prototyping systems for engineering education and research, Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006, pp.2292-2297

[2] R. Bayindir, E. Kabalci, O. Kaplan, Y.E. Oz, Microcontroller based electrical machines training set, Power Electronics and Motion Control Conference (EPE/PEMC), 2012 15th International, pp.DS3e.12-1-DS3e.12-4

[3] R.E. Araujo, H. Teixeira, J. Barbosa, V. Leite, A hardware tool for explained power electronics control of induction motors, European Conference on Power Electronics and Applications, 2005

[4] D. Montesinos, S. Galceran, A. Sudria, O. Gomis, A laboratory test bed for PM brushless motor control, European Conference on Power Electronics and Applications, 2005

[5] A.M. Zyuzev, K.E. Nesterov, M.V. Mudrov, “A hardware-software complex for real-time modeling of electric drives, Russian Electrical Engineering, September 2014, Volume 85, Issue 9, pp 591-596

[6] A.M. Zyuzev, K.E. Nesterov, M.V. Mudrov, “The software-hardware simulator of the electric drive”, 16th European Conference on Power Electronics and Applications (EPE'14-ECCE Europe), 2014

[7] IQMath Library, a Virtual Floating Point Engine. Texas instruments, Dallas, Jun. 2002.

[8] A.S. Anuchin, Structures of a digital PI controller for an electric drive, Russian Electrical Engineering, Allerton Press, 2014, vol.85, pp. 419-423

[9] A. Anuchin, Yu. Vagapov, Configurable Control Systems of Power Converters for Instructional Laboratories, ITA 2015, submitted