BRUSHLESS DC MOTOR

SIMULINK SIMULATOR

Usage Manual

Devendra Rai,
Department of Electronics and Communication Engineering,
National Institute of Technology Karnataka,
Surathkal 575 025
INDIA
devenrai@rediffmail.com

ABOUT THE MANUAL

The manual describes the working of brushless dc motor simulator. The simulator was developed as a final year project at National Institute of Technology, Karnataka. The work is based around the trapezoidal back emf , ,star wound stator type brushless dc motor, although the logic used here can be applied to any type of flux distribution.

The Simulator has been written exclusively in Matlab R13. The details of all the packages used are given in the report.

Much of the work involved the development of the core BLDC 'mybldc' block, and a logical equivalent of the inverter. The idea has been to avoid the use of SIMULINK Simpower blockset that are slow to simulate and require large memory and phenomenal processor power.

The approach used is to reap the benefits of both Matlab and Simulink, thereby giving a very flexible but reliable model.

Since, the product was developed as a student project, I would be glad to receive any feedbacks on its performance.

Devendra Rai,
devenrai@rediffmail.com

Department of Electronics and Communication Engineering,
National Institute of Technology Karnataka,
Surathkal – 575 025.
INDIA

DISCLAIMER

SYSTEM REQUIREMENTS AND OTHER IMPORTANT INFORMATION

MATLAB Version 6.5.0.180913a (R13)
Operating System: Microsoft Windows XP Version 5.1 (Build 2600)
Java VM Version: Java 1.3.1_01 with Sun Microsystems Inc. Java HotSpot(TM)

MATLAB	Version 6.5	(R13)
Simulink	Version 5.0	(R13)
MATLAB Compiler	Version 3.0	(R13)
RAM (min recommended)	256 MB	

The Processor should be powerful. The modeling was done on a Pentium 4 (1.7 Ghz) processor.

A good graphics card to view the results

THE RESULTS OF SIMULATION ARE HIGHLY DEPENDENT UPON THE CHOICE OF THE SOLVER. WE HAVE USED ODE113(ADAMS). SOLVERS LIKE ODE45 ARE FAST BUT GIVE POOR RESULTS.

The main Simulink file name is ‘mybldc2_mdl2.mdl’. The other .m and .mdl files that are supplied are required by the main .mdl file.

CONTENTS

1. MODELING THE BRUSHLESS DC MOTOR

The modeling of brushless dc motor involves solving many simultaneous differential equations, each depending upon the inputs to the motor and the simulation constants. Simulation constants are values like the phase inductance that do not change during simulation. Therefore these parameters can be treated as constants during a simulation. However, the model provides for dialogue boxes that can be used to vary the values of these constants.

The equations for the brushless dc motor are listed as under. Star wound rotor is assumed. For more information the reader can refer to [1], [2] mentioned in the appendix.

The core block for the brushless dc motor has been written as a state space model. During the course of the project, many approaches were tried and it was realized that state space modeling enabled accurate and easy description of the brushless dc motor.

DETAILS OF STATE SPACE MODELING[1][2]

The coupled circuit equations of the stator windings in terms of motor electrical constants are

$$\begin{bmatrix} V_{as}-v_n \\ V_{bs}-v_n \\ V_{cs}-v_n \end{bmatrix} = \begin{bmatrix} R_s & 0 & 0 \\ 0 & R_s & 0 \\ 0 & 0 & R_s \end{bmatrix}\begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} + \mathbf{p}\begin{bmatrix} L_{aa} & L_{ab} & L_{ac} \\ L_{ba} & L_{bb} & L_{bc} \\ L_{ca} & L_{cb} & L_{cc} \end{bmatrix}\begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} + \begin{bmatrix} E_a \\ E_b \\ E_c \end{bmatrix}$$

--(1)

where:

R_s : Stator resistance per phase

I_a, I_b, I_c : Stator phase currents

p : $\frac{d}{dt}$ is the time derivative operator

E_a, E_b, E_c represent the back emfs in the respective phases in (1)

v_n : is the neutral point node voltage given by[2]:

$$v_n = \frac{1}{3}[V_{as} + V_{bs} + V_{cs}] - \sum BEMFs \qquad \text{--(2)}$$

$\sum BEMFs$ means summing up the individual phase emfs on an instant to instant basis.
The induced emfs are all assumed to be trapezoidal, whose peak value is given by:

Ep=(BLv)N = N(Blrω) = N$\Phi\omega = \lambda\ \omega$. --(3)

Where:
B is the flux density of the field in webers
L is the rotor length
N is the number of turns per phase
ω is the electrical angular speed in rad/sec
Φ represents flux linkage=BLr
λ represents the total flux linkage given as the product number of conductors and flux linkage/conductor.

If there is no change in rotor reluctance with angle because of nonsalient rotor and assuming three symmetric phases, inductances and mutual inductances are assumed to be symmetric for all phases, i.e. (1) becomes:

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = R_s * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} + \mathbf{p} \begin{bmatrix} L & M & M \\ M & L & M \\ M & M & L \end{bmatrix} \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} + \begin{bmatrix} E_a \\ E_b \\ E_c \end{bmatrix} \qquad \text{-(4)}$$

Simplifying (3) further we get equation (4)

$$\begin{bmatrix} Vas \\ Vbs \\ Vcs \end{bmatrix} = R_s * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} + \mathbf{p} \begin{bmatrix} L-M & 0 & 0 \\ 0 & L-M & 0 \\ 0 & 0 & L-M \end{bmatrix} \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} + \begin{bmatrix} E_a \\ E_b \\ E_c \end{bmatrix} \qquad \text{-(5)}$$

The generated electromagnetic torque is given by

$$T_e = [E_a I_a + E_b I_b + E_c I_c]/\omega \text{ (in N.m)} \qquad \text{- (6)}$$

The induced emfs can be written as

$$\begin{aligned} E_a &= \text{fa}(\theta)\lambda\ \omega. \\ E_b &= \text{fb}(\theta)\lambda\ \omega. \\ E_c &= \text{fc}(\theta)\lambda\ \omega. \end{aligned} \qquad \text{-(7)}$$

Where:

fa(θ), fb(θ) and fc(θ) are functions having same shapes as back emfs with maximum magnitude of ± 1.These are written as 'LA', 'LB','LC' in the file 'mybldc2.m'. (names are derived as ***L***ookup table for ***A*** etc.)

These values from (6) can be substituted in (5) to obtain the value of torque.

Also,

$$J(d\omega/dt) + B\omega = T_e - T_l \qquad \text{-(8)}$$

Where:
T_l : load torque
J: moment of inertia
B: friction coefficient.

Electrical rotor speed and position are related by

$$d\theta/dt = (P/2)^* \ \omega \qquad \text{-(9)}$$

Where P is the number of poles in the motor.

combining all the equations ,the system space form becomes

$$X' = Ax + Bu \qquad \text{--(10)}$$

Where $x = \begin{bmatrix} I_a & I_b & I_c & \omega & \theta \end{bmatrix}^T$ -(11)

Thus the state space matrix becomes:

$$A = \begin{bmatrix} -R_S/L_1 & 0 & 0 & (\lambda p^* f_a(\theta))/L_1 & 0 \\ 0 & -R_s/L_1 & 0 & (\lambda p^* f_b(\theta))/L_1 & 0 \\ 0 & 0 & -R_s/L_1 & (\lambda p^* f_s(\theta))/L_1 & 0 \\ (\lambda p^* f_a(\theta))/J & (\lambda p^* f_b(\theta))/J & (\lambda p^* f_c(\theta))/J & -B/J & 0 \\ 0 & 0 & 0 & P/2 & 0 \end{bmatrix} \qquad \text{-(12)}$$

$$B = \begin{bmatrix} 1/L_1 & 0 & 0 & 0 \\ 0 & 1/L_1 & 0 & 0 \\ 0 & 0 & 1/L_1 & 0 \\ 0 & 0 & 0 & -1/J \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{-(13)}$$

and

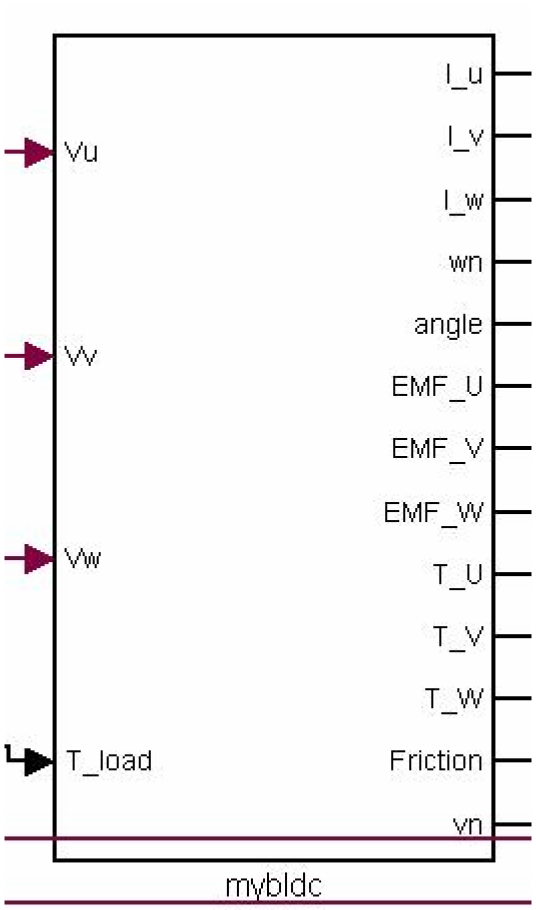
$$U = \begin{bmatrix} V_a & V_b & V_c & T_l \end{bmatrix}^T \qquad -(14)$$

Where:
L_1: L-M;
L: self inductance of the winding per phase
M: The mutual inductance per phase
V_a, V_b, V_c are the per phase impressed voltages on the motor windings.

All the equations form the entire state space model for the BLDC.
Consistent system of units must be used.

2.THE CORE

(describes the '*mybldc*' and its sub blocks)



The core of the state space model calculates the five variables given by equation (11). These variables are further processed to give a total of 25 outputs that can be monitored.

The Block takes the voltage values that are impressed upon the windings along with the load torque information.

The values to be input are derived from the general Simulink sources and not the Simpower systems blockset. The outputs are calculated by solving differential equations so only numerical values are required.

We had tried to use Simpower system blockset, but that approach requires very powerful processors and very large memory. The time required to compute values was in the order of hours, so there is definitely a trade off here, but we have tried our best not to compromise on the accuracy.

The Core is driven by an inverter that has been custom designed for this kind of blockset usage, and <u>we do not recommend that it be used as an independent unit.</u>

The inputs are:
V_a, V_b, V_c or V_U, V_V, V_W which are the impressed voltage values (numerical values only). **The input signals can be derived from general Simulink sources, but this block should not be connected to the Simpower system blockset(s).**

THE OUTPUTS ARE:

(1*): I_u, I_v, I_w* are the individual phase currents
(2): *wn* is the rotor electrical speed in rad/s
(3): *angle* is the rotor electrical position as compared to the initial position
(4): *EMF_U,EMF_V,EMF_W* are the back emf values generated in the three phases.
(5*): T_U,T_V,T_W* are the individual phase torques.
(6):*Friction*: the values of the friction faced by the rotor. Contains both static and coulomb friction.
(7): *vn* is the neutral point node voltage. (STAR WOUND STATOR IS ASSUMED).

THE INPUTS EXPECTED ARE:

(1*): Vu,Vv,Vw* are the phase voltage impressed upon the motor. It is important to note that the voltages are just numbers that are part of the differential equations that are solved by MATLAB. **Hence, these should not be derived from the Simpower blockset.** The signals to these ports can be derived from the general sources in the SIMULINK toolbox.
(2):*Tl* is the external load torque that is applied to the system. Thus the total retarding torque that is experienced by the system is a combination of friction and the load torque.

THE SIMULATION CONSTANTS:

Double click on the 'mybldc' block to go inside. There you will find two sub blocks: 'my state-space' and 'S function'.

'my state-space' solves the state space equations related to the motor as have been just described. The initial conditions can be changed under he field 'Initial conditions:'. The initial conditions are $I_u, I_v, I_w, \omega, \theta$. If required appropriate initial values can be filled in. The fields are separated by commas.
The fields 'A'. 'B','C','D' are as described in the previous chapter. The field under A is a time dependent field, i.e, the value in this field is changed and re-written by SIMULINK as the simulation progresses. Hence, every time a fresh simulation is run, double clicking 'RESET CONTROLLER' icon will write the initial values in this field. More on this has been described in the 'RESET CONTROLLER' section of this chapter.

Double clicking on the 'S-function' will present a table that contains the simulation constants like number of turns per phase, rotor length etc. Appropriate values should be filled in. A consistent system of units should be used.

RESET CONTROLLER: This icon on the main pane is used to reset the matrix 'A' while starting the fresh simulation. The values that are written are

calculated from the default motor parameters. If motor parameters are required to be changed, then the corresponding m-file has to be updated. Double clicking the ‘RESET CONTROLLER; icon call the file:
mybldc2([],[],[],'reset',find_system(get_param(gcs,'Handle'),'LookUnderMasks','all','Tag','PPC'));
The callback is written in BLOCK PROPERTIES\callbacks\openfcn*. It calls the m file ‘mybldc2.m’. On a double click, SIMULINK generates the flag ‘reset’ which call the function named ‘*LocalResetController(name);*’.In the function definition area, the line:

[line number 462]
*up_A=[-(R/(L-M)) 0 0 -N*BM*Rl*Rr*LA/(L-M) 0; 0 -(R/(L-M)) 0 -N*BM*Rl*Rr*LB/(L-M) 0; 0 0 -(R/(L-M)) -N*BM*Rl*Rr*LC/(L-M) 0; N*BM*Rl*Rr*LA/J N*BM*Rl*Rr*LB/J N*BM*Rl*Rr*LC/J -DF/J 0; 0 0 0 P/2 0];*

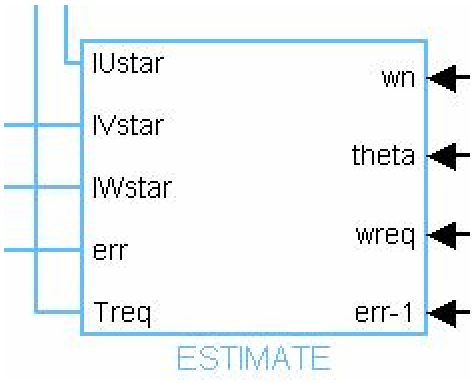
calculates the initial values and writes them into the ‘A’ field of ‘my state space’ block.

Changing the values of the motor parameters in ‘S function’ automatically updates this filed.

The S function block calculates the additional parameters like the normalized angular value and the back emfs.

3.THE ESTIMATION BLOCK

(describes '*ESTIMATE*',''*subsystem*','*err_gen*' *blocks*)



The estimation block contains the central controller. The block again is an M file S function '*calc_core.m*'. The S function calculates the reference phase currents from the speed and torque requirement. The working of the block is described as under:

1. It calculates the error in speed (difference in the actual and the command speed)
2. Next, it used a simple PID controller to estimate the torque that is required:

Treq= sys(4)(KP + KI*0.5*4e-6 + KD/4e-6) + u(4)*(KI*0.5*4e-6 - KD/4e-6);*
4e-6 is the sampling time period. This can be changed by the user. An additional change should me made to the value of ts in line number 125 in file '*calc_core.m*'.

Where:
KP is the proportionality constant;
KI is the integral constant;
KD is the differential constant.
These values and other parameters can be changed on double clicking the 'ESTIMATE' block.

The saturation is applied to the total torque requirement as a matter of safety.

Care should be taken that consistent values are fed into this and the 'mybldc\S-function' block. The blocks are developed as independent of each other so some parameters have to be fed twice in the blocks. (It allows you to make your own systems)

3. Next, the total current command is evaluated:
   *Ireq= sys(5)/(N*BM*Rl*Rr);*

   Where sys(5) contains the torque required that is calculated in the previous step.
4. A small transformation is carried out to calculate the direct and the quadrature components. The value of 'delta' can be adjusted in the field '*phase difference between current and torque'* of the interface table that pops up when the 'ESTIMATE' block is double clicked.

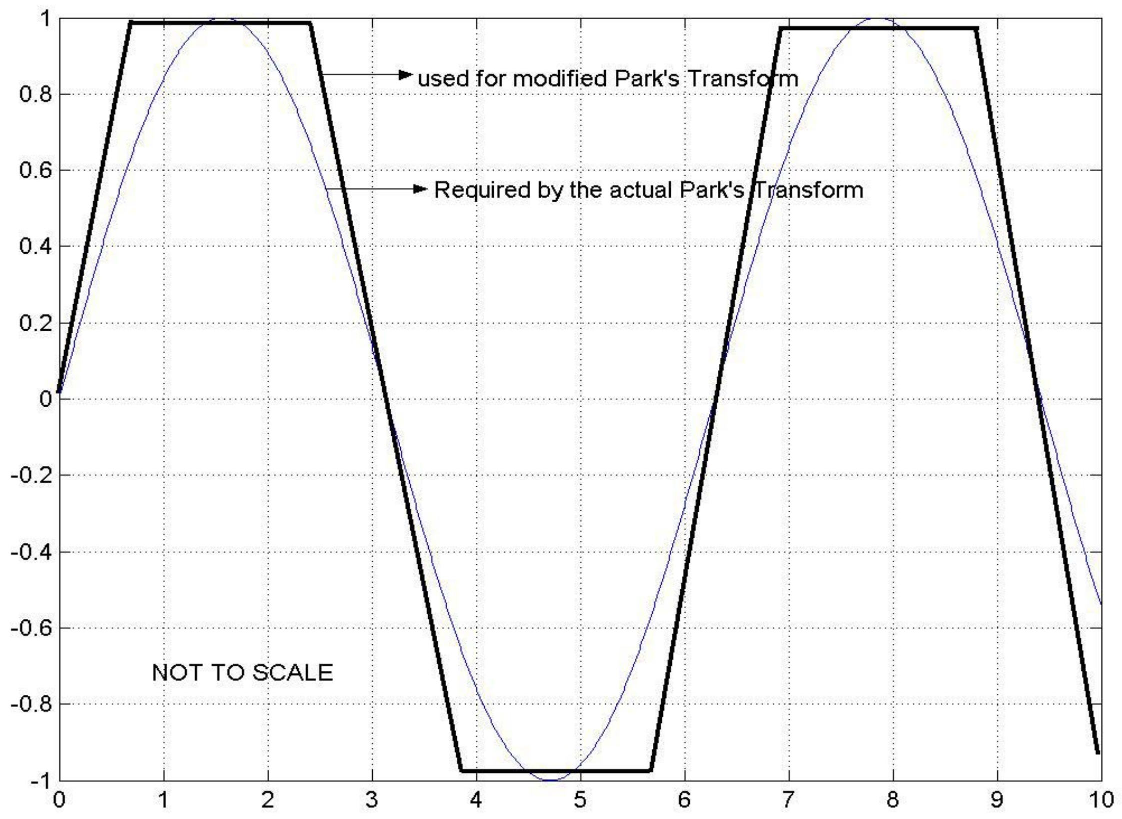
5. At this stage a modified Park's transform is carried out:

```
matrix=[COSTHETA        SINTHETA        %the transformation matrix
        COSTHETARS      SINTHETARS
        COSTHETALS      SINTHETALS];

Istar=matrix*I_QD;
```

The values COSTHETA, SINTHETA, COSTHETARS (i.e. costheta right shifted by 120 degrees) SINTHETARS (sintheta right shifted by 120 degrees) ,COSTHETALS (costheta left shifted by 120 degrees) and SINTHETALS (sintheta left shifted by 120 degrees).

The values of these are derived by replacing cos(_), sin(_) as required in the Park's Transform with the actual and appropriately shifted flux or back emf distribution as shown below in the diagram:



Above shown in a direct replacement of sin(_). Similarly, other parameters can be calculated and replaced.

Istar is actually a column matrix that contains the fields ss1,ss2,ss3.

6. The values ss1,ss2,ss3 are checked for bounds determined from the value of the base current allowed and are clipped if they exceed twice the rated current. Then from these values *Iustar, Ivstar* and *Iwstar* are derived. These are the required phase currents (current commands) for a particular value of speed and/or torque command.

THE INPUTS EXPECTED ARE:

1. *wn:* the actual value of the rotor speed (electrical rad/s);
2. *theta*: rotor shaft position in electrical radians;
3. *wreq*: the required rotor speed (electrical rad/s);
4. *err-1*: this is fed back from the output. This is the error between the command and the actual speed at the previous instant.

THE OUTPUTS ARE:

1. *IUstar, IVstar,IWstar*: The command phase currents
2. *err:* the error between the command speed and the actual speed at the current time step.
3. *Treq:* The value of torque that is to be generated by the motor so as to meet the command speed and/or load torque conditions.

The other blocks like the 'subsystem' and the 'error gen' are required to insert appropriate delays for the PID controller.

4.Zero Crossing, IC, 120 deg trigger BLOCKS:

‘Zero Crossing’ block evaluates the zero crossings of :

1. UV: zero crossing detection of BEMF U – BEMF V
2. VW: zero crossing detection of BEMF V – BEMF W
3. WU: zero crossing detection of BEMF W – BEMF U

These signals can be used to synchronize the controller or the inverter.

‘IC’ block is used to hold the controller to its initial state till the time motor picks up sufficient speed and the back emf voltages are significant. It is to be noted here that sensorless control design has been implemented, which depends heavily on the values of the back emfs. Initially, since the motor is at the standstill, the back emfs have not built up. Thus the motor is ‘blindly’ ramped up till sufficient voltages are built up. It is important that during this period the controller does not lose its initial state. More about the ramping is describe in the ‘120 deg trigger’ block.

‘120 deg trigger’ block ramps up the motor for some time determined by the value in the field ‘threshold time’ in the block ‘CHANGER’. After Threshold time the block ‘CHANGER’ just disconnects the ‘120 deg trigger’ block and puts the motor in the closed loop.

The sequence followed by the ‘120 deg trigger’ block is Q5Q6-Q1Q6-Q1Q2-Q3Q2-Q3Q4-Q5Q4. Q_i refers to firing the gate of the ith transistor in the inverter. The above pattern is followed in the 120 degree mode of excitation, hence the name ‘120 deg trigger’.

The appropriate switching and waveforms are shown on the next page.

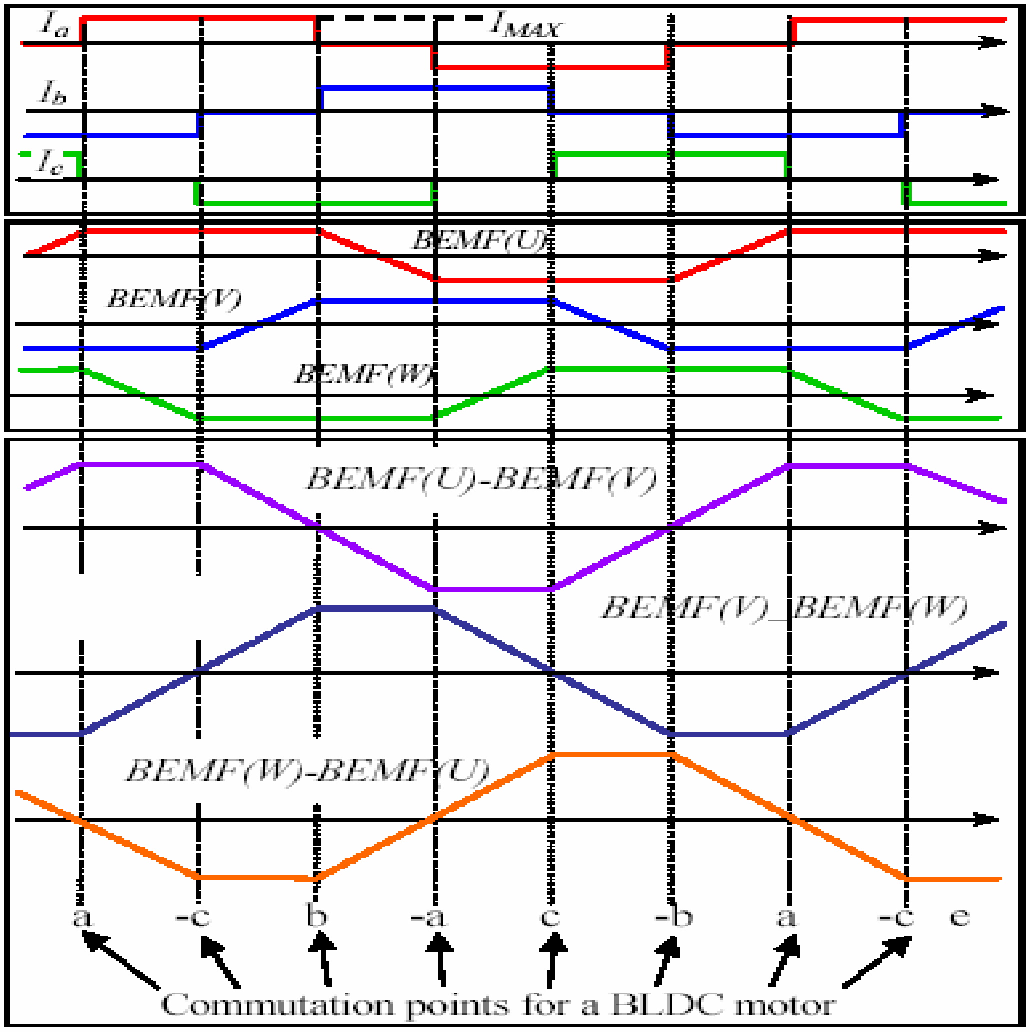
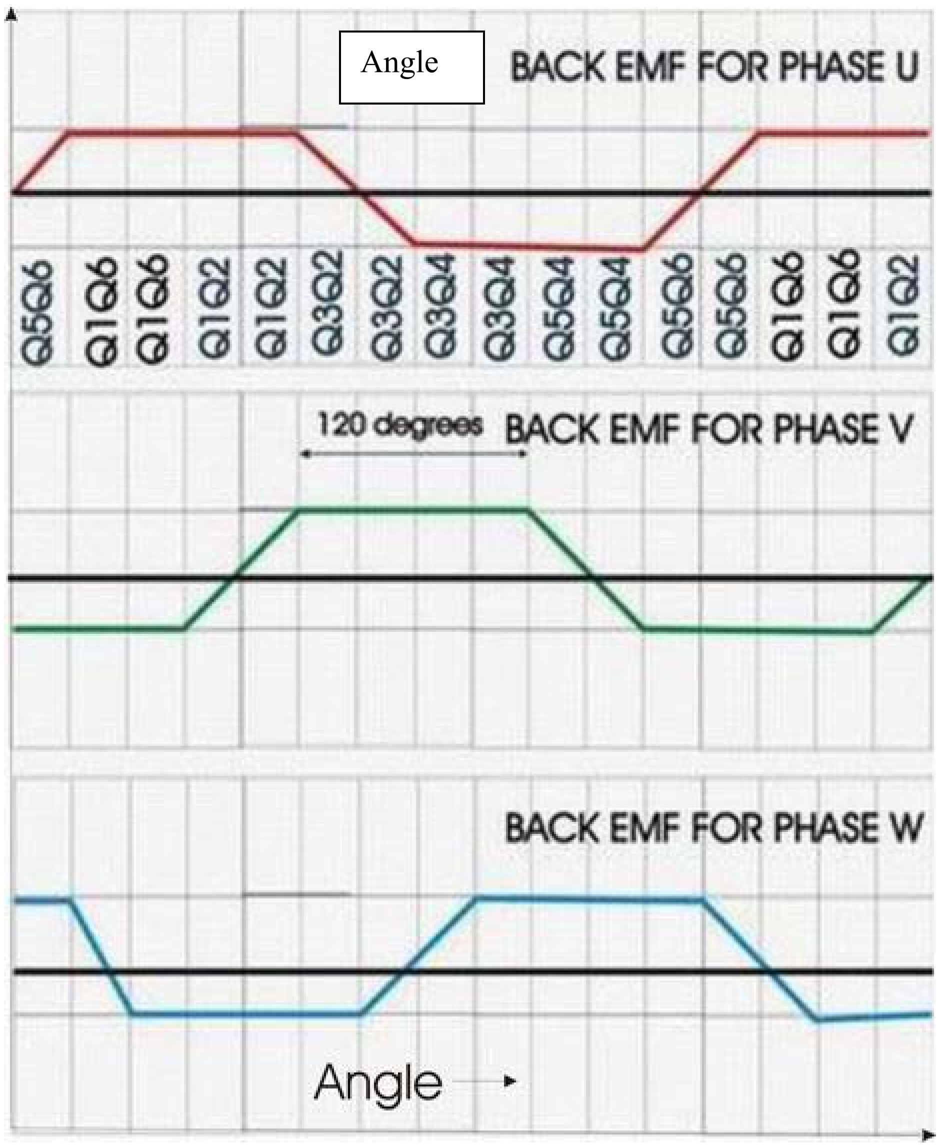


Figure reproduced from: ***Simplified Sensorless Control for BLDC Motor, Using DSP Technology***: Juan W. Dixon, Matías Rodríguez and Rodrigo Huerta.

Depending upon the Back EMF waveform: BEMF(U),BEMF(V),BEMF(W), the appropriate gates of the inverter are fired. Each gate in turned on for 60 degree interval. Thus there are 6 different firing sets to make what is called a ‘6 pulse generator’.

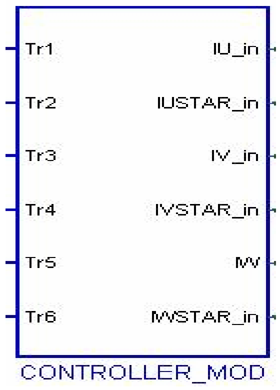
The switching sequence depending upon the back emfs of the phase are shown on the next page. Here Qi refers to the gate of the ith transistor being activated in the inverter.



The figure showing the back emfs and the switching sequence. Qi represents the ith transistor in the H bridge that is to be fired. This is equivalent to activating the GATEi of the inverter 'ALL PHASE' block.

5.THE CONTROLLER_MOD BLOCK:

NOTE: THE TECHNIQUE USED HERE TO CONTROL THE BLDC MAY NOT BE OPTIMIUM, AND THEREFORE THE USER CAN USE HIS/HER OWN LOGIC. THIS SECTION IS WRITTEN FOR MERE GUIDANCE,



After the references have been generated by the 'ESTIMATE' block, the 'CONTROLLER_MOD' block takes over and triggers the appropriate gates of the inverter to establish and maintain the desired current.

For reference the structure of the inverter is as shown:

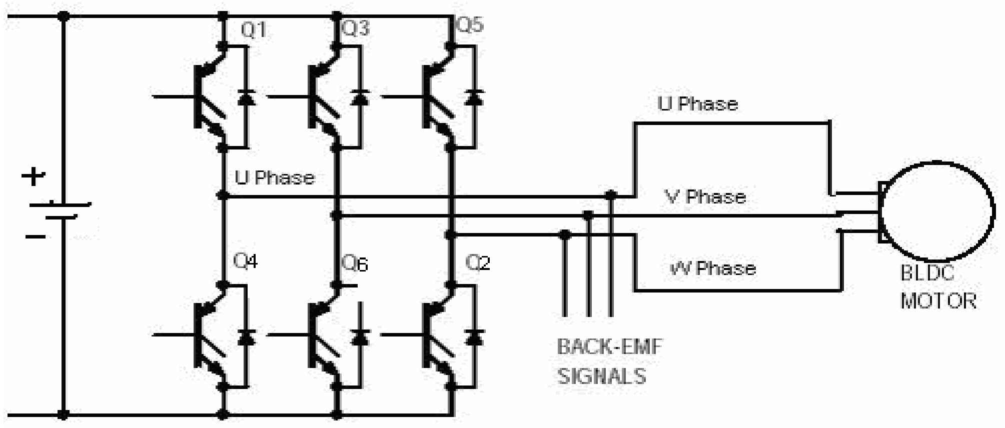


Figure adapted from: **Literature Number:AN1858, (Motorola Inc.,)**
Sensorless Brushless dc Motor Using the MC68HC908MR32 Embedded Motion Control Development System

The currents are taken as positive when they are going into the phase.

The 'CONTROLLER_MOD' block has three sub blocks. Each of these blocks is a finite state machine. Each of the machine has the task of maintaining the required current either in U,V or W phase windings.

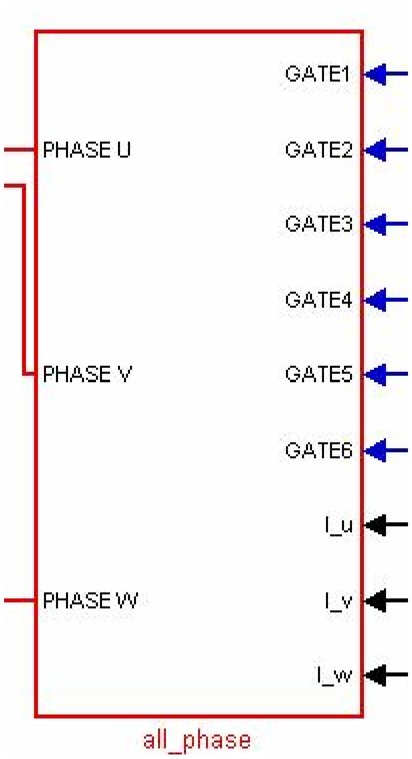
Together they exercise a parallel control over the inverter gates.

The logic used by any of the machines is very simple: for example, for the machine that controls the U phase, it checks to see whether the actual current in phase U ('IU') is greater than the required current ('IUSTAR'), If it is then it will activate transistor Q4. Otherwise, it will activate transistor Q1.

THE INPUTS EXPECTED ARE:
1. *IU_in* is the actual value of current in the U phase winding
2. *IUSTAR_in* is the command current for phase U
3. *IV* is the actual current in the phase V winding
4. *IVSTAR_in* is the command current for phase V
5 *IW_in* is the actual value of the current in the W winding
6. *IWSTAR_in* is the command current for phase W.

THE OUTPUTS EXPECTED ARE:
1. Tr1, Tr2, …Tr6: are the logic signals that are fed to the inverter gated (inputs to the 'all phase' block.

6.THE INVERTER ‘ALL PHASE’ BLOCK:



This is actually a logical version of the power electronics 6 transistor inverter. As the core module ‘mybldc.mdl’ was written as a set of differential equations, the ‘voltages’ that are impressed upon the module are signals derived from the general ‘sources’ in SIMULINK. Hence, the inverter ‘ALL PHASE’ is a completely logical unit and should not be interfaced with the Power electronics block set components.

Also, an equivalent inverter made out of the power electronics block set is very slow and demands a phenomenal computing power. This implementation, though not as accurate (when it comes to calculating transients) but is fast and reliable. At least, for this kind of implementation, this inverter implementation would suffice.

The logic used is straight forward:
(*refer to the diagram in the previous section*)

If the gate of transistor Q4 is fired, the phase U is connected to the –ve terminal of the voltage source. If however Q1 is activated, the phase U is oonnected to the +ve terminal of the voltage source. If however both the transistors are off, then depending upon the sign (ie direction) of the current ‘IU’ the phase U may be connected to either side of the voltage source. This is to simulate freewheeling. If both the switches are off and the current is +ve then the phase U is connected to the –ve side of the voltage source and vice versa.

Similar logic is followed for all the other legs.

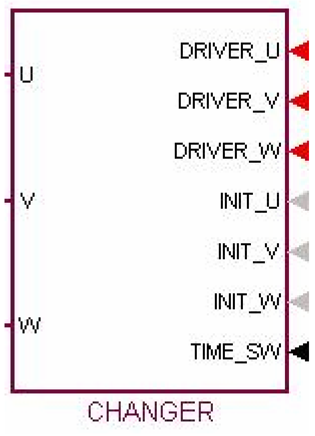
THE INPUTS EXPECTED ARE:
1. *GATE1, GATE2,….GATE6* are the logical signals (zero/non zero) that activate the ‘transistors’.
2*. Iu,Iv,Iw* are the values of the phase currents that are required to decide the ‘freewheeling’ logic.

THE OUTPUTS ARE:

1. PHASEU, PHASEV,PHASEW are the signals ‘voltages’ of the respective phases.

A split voltage supply is assumed by default. However, this can be easily changed by little manipulation inside the ‘ALL PHASES’ block.

7. THE CHANGER BLOCK



As already mentioned, the BLDC motor in a sensor less scheme cannot be started by the controller. This is because the sensor less algorithm depends on the back emfs and initially when the rotor is at standstill, the back emfs have not yet built up. Thus, for some small amount of time, the motor is 'blindly' excited (without feedback, or in open loop) till sufficient back emf is built up. The motor is excited 'ramped up' in an open loop through the trigger block '120 deg trigger', for an amount of time equal to 'threshold value'. The 'threshold value can be changed by double clicking the 'CHANGER' block and entering the appropriate threshold value.

For the initial ramp up time, the values at the U,V and W output ports are derived from the INIT_U, INIT_V,INIT_W input ports and then from DRIVER_U, DRIVER_V,DRIVER_W input ports. The latter three input ports are fed by the controller (closed loop operation).

THE INPUTS EXPECTED ARE:

1. *DRIVER_U,DRIVER_V,DRIVER_W* are the inputs derived from the controller. (For closed loop operation).
2. *INIT_U, INIT_V,INIT_W* are the inputs derived from the '120 deg trigger' block. This is for the ramp up.
3. *TIME_SW:* connected to a digital clock. Used to decide the instant of switchover from an open loop operation to the closed loop operation.

THE OUTPUTS EXPECTED ARE:

1. *U* is the U phase 'voltage' that is meant to be supplied to the 'mybldc' block.
2. *V* is the V phase 'voltage' that is meant to be supplied to the 'mybldc' block
3. *W* is the W phase 'voltage' that is meant to be supplied to the 'mybldc' block

8. THE SIMULINK OUTPUT FILE

By default, when the simulation is completed, the simulation results are stored in a file called ‘ result.mat’ and when double clicked, the variable ‘res’ is created in the workspace. The variable ‘res’ contains the simulation results distributed in 25 fields. (this means that 25 different quantities can be monitored and each field contained in a separate row.) Below is the list of all the row-wise entries (The serial numbers are also row numbers):

1. The first row contain the the information about the time instants when the results were calculated by SIMULINK. (Time field)
2. **I_U**: instantaneous values of the currents in phase U of BLDC
3. **I_V**: instantaneous values of the currents in phase V of BLDC
4. **I_W**: instantaneous values of the currents in phase W of BLDC
5. **wn**: instantaneous value of the rotor speed in electrical rad/s
6. **angle**: instantaneous value of the rotor displacement in electrical rad
7. **EMF_U**: the instantaneous value of the back emf for phase U
8. **EMF_V**: the instantaneous value of the back emf for phase V
9. **EMF_W**: the instantaneous value of the back emf for phase W
10. **T_U**: instantaneous value of the torque generated in phase U
11. **T_V**: instantaneous value of the torque generated in phase V
12. **T_W**: instantaneous value of the torque generated in phase W
13. **Friction**: instantaneous value of the friction faced
14. **V_U**: instantaneous value of the voltage applied to phase U of BLDC
15. **V_V**: instantaneous value of the voltage applied to phase V of BLDC
16. **V_W**: instantaneous value of the voltage applied to phase W of BLDC
17. **I_USTAR**: instantaneous value of the phase U command current generated by the controller to meet the speed and/or load torque
18. **I_VSTAR**: instantaneous value of the phase V command current generated by the controller to meet the speed and/or load torque
19. **I_WSTAR**: instantaneous value of the phase W command current generated by the controller to meet the speed and/or load torque
20. **Zero_UV**: The zero crossing detection instants for BEMF (U-V).
21. **Zero_VW**: The zero crossing detection instants for BEMF (V-W).
22. **Zero_WU**: The zero crossing detection instants for BEMF (W-U).
23. **Treq**: The instantaneous value of the torque that is required to meet the speed and/or load command(generated by the controller).
24. **err_out**: The difference in the actual speed and the command speed at the previous time step. By default, the time step has been taken to be 4e-6 s
25. **vn**: The neutral point node voltage.(Star wound stator assumed).

While plotting the independent variable should be time (row 1). For example to plot the speed the following command can be given at the prompt:
Plot(res(1,:) , res(5,:));

9. REFERENCES

[1]. R Krishnan, *Electric motor drives- modeling, analysis and control*, Prentice Hall of India Private Limited, 2002.
[2]. Takashi Kenjo and Shigenobu Nagamori, *Brushless Motors- Advanced Theory and Modern Applications*, Sogo Electronics Press,2003.
[3]. Juan W. Dixon, Matías Rodríguez and Rodrigo Huerta., *Simplified Sensorless Control for BLDC Motor, Using DSP Technology*

10. ACKNOWLEDGEMENTS

The project was by no means an easy task. Also, I did not have any background in the working of electrical machines. Obviously, I required an army of people to help me out with this project.

I am deeply indebted to my guide Mr Jora M Gonda, gonda@nitk.ac.in Asst Prof, Department of Electrical and Electronics Engineering, for guiding me and encouraging me through out the project.

Special thanks go to Dr Takashi Kenjo, Professor, Polytechnic University of Japan, and Dr Tatsua Kikuchi, Professor, Tokyo Institute, Polytechnic University of Japan, who, inspite of being busy, sent a starter book and helped get over beginner's blues.

I am grateful to Dr P S Bhat, head of department, department of E&C, for providing me encouragement and help to finish the project.

I am grateful to all the people at MATLAB CENTRAL who have helped me clear my ideas and finish the project.