Optimal Pulse Patterns Modulator using Real Time Linux.

David A. Ulloa G.
Universidad Técnica Federico Santa María, Chile.
Guest student in Bergische Universität Gesamthochschule Wuppertal, Germany.
Catedral 1679 depto. 405 Santiago Centro, Santiago,Chile.
dulloa@mail.com

Abstract

Since that micro-controllers and DSP's started to be used in power electronics many ideas could be realized and the Optimal Pulse Pattern Modulator was developed together with the Trajectory Controller in order to correct its problem. Working in this kind of research one needs a stable and comfortable system which must be flexible to make almost any change. According to that a combination between a Hitachi SH2 core and a personal computer under Real Time Linux was implemented. This paper describes some aspects about the theory of this modulator, how that was implemented and shows results obtained with this system.

1 Introduction.

Many years ago Power Electronics gave us the possibility to control big volumes of energy using converters which are made principally of Transistors, MOSFET's, IGBT's or IGCT's as switches. One can find a large variety of different topologies for this kind of converters and this work was done by a IGBT three level inverter (Fig. 1).

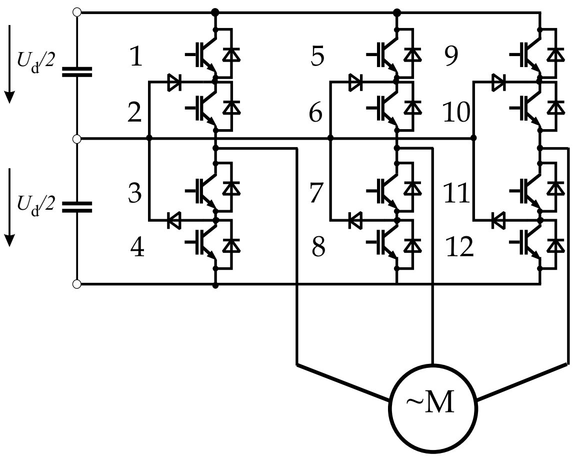

FIGURE 1: *Three Levels Inverter with load*

The three level inverter has this name due to the number of voltage levels per phase that it can produce in its output. These outputs can have a positive voltage (Ud/2), a zero voltage and a negative voltage (-Ud/2). Due to efficiency reasons, power converters work in the switching mode, generating harmonics as undesired consequence of this operating principle. Those harmonics are smaller if the switching frequency is enhanced, but on the other hand when one increases the switching frequency one gets more switching losses (Fig. 2). Between switching losses and low distortion exist a big tradeoff which is managed using Optimal Pulses Patterns modulation.

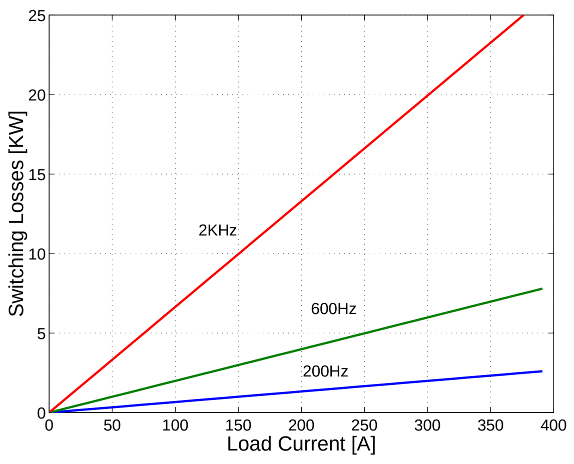

FIGURE 2: *Switching losses in a 2000[HP] three levels inverter. Notice that a normal apartment can be supplied with 4 [kW].*

2 Theoretic background.

2.1 Space vector modulation.

This kind of modulation is the most typical way to drive an inverter. All voltage combinations at the output are described as a fix vector in the alpha-beta space through the expression (1), and this for a three levels inverter, produces a vector map as in Fig. 3. A reference voltage vector U* is generated by the three closer fix vectors through an average in time of these three. This procedure is call Space Vector Modulation (SVM)[1] and it has a good behavior when the switching frequency is over 2[KHz], otherwise this modulation brings us undesired distortion.

$$U_{\alpha,\beta} = \frac{2}{3}(u_u + u_v e^{j\frac{2\pi}{3}} + u_w e^{j\frac{4\pi}{3}}) \qquad (1)$$

Alpha-Beta transformation

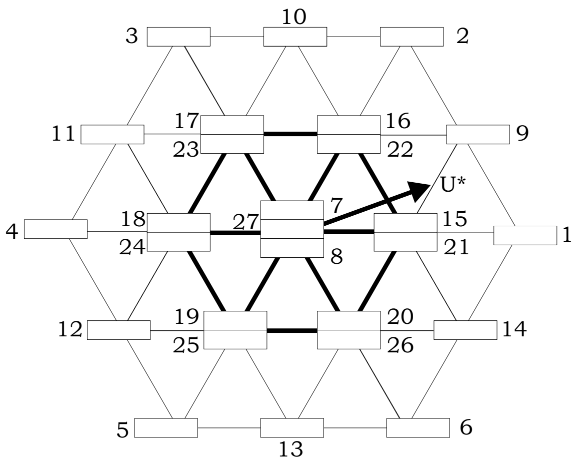

FIGURE 3: *This figure shows all possible output voltages combinations of a three levels inverter. The vector U* is a possible reference vector for the modulator.*

2.2 Optimal pulse patterns modulation.

The most effective way to reduce the harmonic and subharmonic voltage component is to take in account all switching angles in a fundamental period as variables for a closed optimization. All switching times are calculated off-line, and stored in memory, assuming steady state operation of the machine. This modulation gives a defined steady state current and stator flux trajectory in the alpha-beta space, these trajectories depend upon the specific pulse pattern that the modulator is sending to the inverter. The optimal pulse patterns modulation (OPPM) does not produce subharmonics, hence it is in synchronism with the fundamental voltage frequency. A comparison between the voltage harmonics of SVM (Fig. 4) and OPPM (Fig. 5) shows that the distortion in this modulation is lower than in SVM.

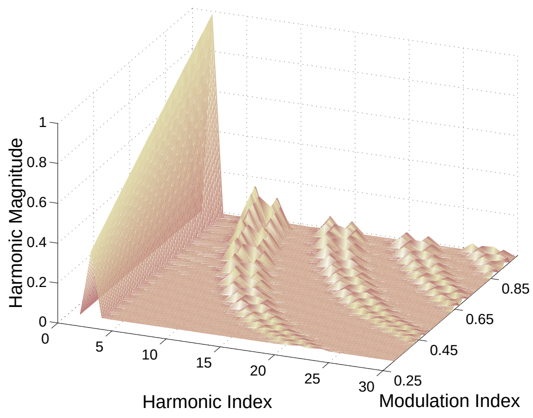

FIGURE 4: *Harmonic contains in SVM. Notice that the first harmonic of the switching frequency is closer to the fundamental harmonic and its magnitude is at its maximum when the modulation index is at its maximum.*

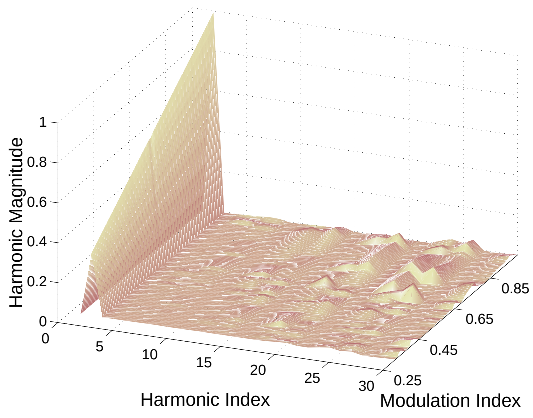

FIGURE 5: *Harmonic contains in OPPM. In this modulation the harmonic magnitudes are bigger for higher harmonics however this is not a problem because they are filtered by the machine.*

2.3 Flux trajectory controller.

When the machine speed controller changes its operation point, the optimal pulse patterns modulator must exchange one pattern for another. Due to all patterns being different, a change between them produces a temporal flux and current error. This error is called Dynamic Modulation Error (DME) and it must be corrected with a trajectory controller [2]. A flux trajectory controller is implemented and it is all the time comparing the machine flux trajectory with the steady state flux trajectory. When they are different, a DME error is produced and the controller corrects the trajectory, making changes at the calculated switching angles.

3 Implementation.

When one works in fast dynamic systems, such as AC machine controllers, it is extremely important to have a ***hard*** real time platform. A PC system is not enough for this requirement because the delay between an hardware interrupt signal and its software answer is not fixed.

The system was implemented using a SH2 core Hitachi micro-controller connected through a SRAM with a PC system. The micro-controller that keeps the ***hard*** real time does not have a variable delay between a hardware interrupt signal and its software answer. Owing to the micro-controller not being mathematically powerful enough, another processor is required for the *"number crunching"* and a PC under real time Linux was added for this function.

3.1 Communication.

The communication sequence between the micro-controller and the PC is showed in Fig. 6. The micro-controller gets an Interrupt from an internal timer. After each Interrupt it sets all switch timers for the inverter, retrieves all data from the AD (Analog to Digital) converters, stores all measurements in the SRAM and sends an Interrupt to the PC. The real time Linux system answers as fast as possible the hardware interrupt from the micro-controller, then the interrupt service routine gets the measurements from the SRAM, the PC makes all calculations, then sends the results to the SRAM and then returns an interrupt to micro-controller. Finally the micro-controller gets all calculated values from the SRAM and waits for the next Interrupt from the internal timer. This sequence has a frequency of 2[KHz].

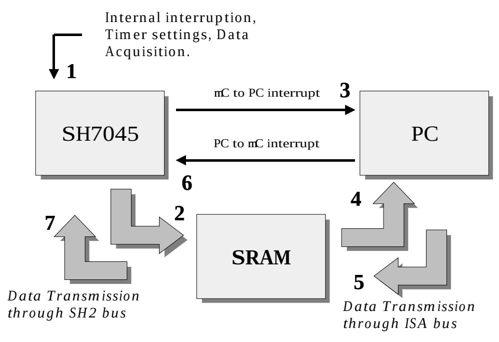
Internal interruption, Timer settings, Data Acquisition.
1
SH7045
mC to PC interrupt
3
PC
PC to mC interrupt
6
4
2
7
SRAM
5
Data Transmission through SH2 bus
Data Transmission through ISA bus

FIGURE 6: *Communication sequence.*

3.2 Interface board.

The SRAM is mounted in an ISA interface board. This board contains buffers and a CPLD and it is practically a discrete dual port RAM. The memory can be accessed by the PC ISA bus and by the micro-controller bus. The micro-controller bus has the memory access priority. All control signals are handled by the CPLD which at the same time controls the hardware interrupts in both directions. The PC IRQ channel is selected by software setting an internal register on the CPLD. Fig. 8 shows a block diagram for the interface board and Fig. 7 shows the complete system.

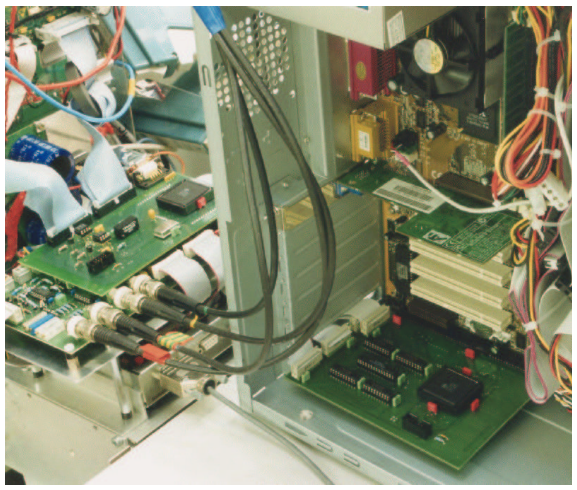

FIGURE 7: *Hardware system.*

3.3 Software.

The micro-controller is programed in C without an operating system. The SRAM is an eight bits data memory and its access by the micro-controller is defined as follow:

```
#define REG1 *((volatile unsigned int *)ADDRES)
```

Now the definition REG1 can be handled as a normal variable. The definition as an *unsigned int* even though the memory is only an eight bits memory, is in order to simplify the hardware implementation. One can notice that on the board diagram, specifically on the address lines for the SH2 bus, that they are from bit number 2 to bit number 9, that means a four bytes data access. The PC sees each byte in the SRAM as a I/O port. The kernel module contains all initializations, the optimal pulses information and all algorithms for the trajectory controller. This was made in order to simplify the code exportation from Linux to another platform such as a powerful DSP

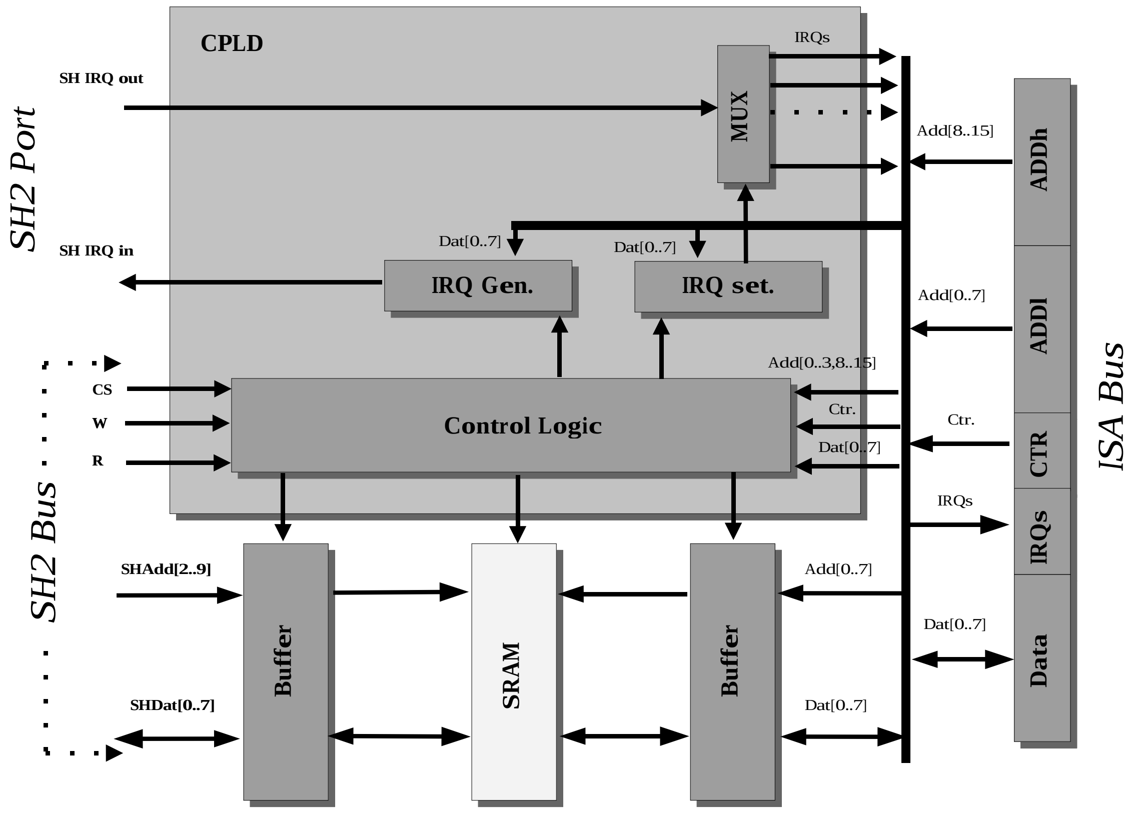

FIGURE 8: *Interface board.*

or micro-processor especially dedicated for this purpose. The project is divided in different header files as follow:

register.h(560L). This file contains all constant and variable definitions, vector structures, scalar and vectorial PI controllers structures, scalar first order system, vectorial first order systems and all I/O ports definitions for SRAM access.

tab4r.h(3064L). Contains all optimal pulse pattern tables and some initial conditions for the steady state stator flux.

vmath.h(278L). This file contains all scalar and vectorial mathematic functions, and includes the whole mathematics system manipulation. With this header file one can constructs in a simple manner complex vectorial models (Fig. 10) such as an asynchronous AC machine.

init.h(410L). Here are handled all initializations such as cleaning the SRAM, calculating the steady state stator flux for all patterns, vectorial machine models initialization and finally the speed control setup.

dme.h(1332L). All calculation related with the dynamic modulation error and the trajectory controller are in this file. It can handles the steady state stator flux and it can calculate the present stator flux of the machine. Also when a change of patters is produced it can calculate the future DME (prediction) in order to correct, when it is possible, the DME before it occurs. The trajectory controller takes the error and decides between a *dead beat algorithm* and a *step wise algorithm* for the correction.

modulador.h(2050L). Implemented in this file was a space vector modulator and the optimal pulse patterns modulator. The optimal pulse patterns modulator sends information to several functions in the *dme.h* file and receives corrections from the trajectory controller. The modulator reads the pulse tables and possible corrections, and sends finally all switching times for the IGBT's in the inverter.

adda.h(91L). It handles the information from the micro-controller AD converters and sends the data for the DA (Digital to Analog) converters.

control.h(195L). This file is still under development and contains algorithms for the machine torque control and the machine speed regulation. It includes also complex AC machine models and different close loop controllers from a simple PI controller to a vectorial flux corrector.

4 Experimental results.

These results were obtain using a IGBT three levels inverter with a DC link of 120 [V] supplying a 33[kW] asynchronous machine. The switching frequency in OPPM for this experiments was between 372[Hz] and 446[Hz]. Fig. 9 shows the alpha-beta stator current using the optimal pulse patterns modulator at 46[Hz] fundamental frequency and with a modulation index of 0.93. Notice that in this current shape the harmonics are in synchronism with the fundamental current. Fig. 13 shows the DME alpha component in a pattern transition. The first channel is the measured DME and the next is the predicted DME. An addition of these provides the DME one interrupt period before.

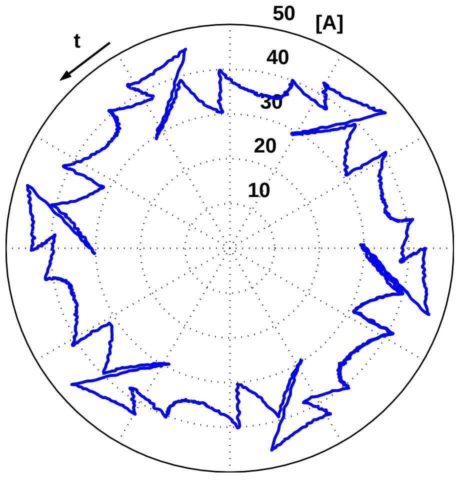

FIGURE 9: *Alpha-Beta Stator Current with OPPM.*

```
//vectors declarations
struct TVector U;
struct TVector R;
struct TVector Aux;

//vectorial PI controllers
struct TVPI Vector_PI1;
struct TVPI Vector_PI2;

//vectorial 1st order systems
struct TV_F_O_Sys Sys1;
struct TV_F_O_Sys Sys2;

⋮

//Initializations
VPiInit(&Vector_PI1,0.1,8,0,25,-25,25);
VPiInit(&Vector_PI2,0.01,0.3,0,2.7,-2.7,5);
VSysInit(&Sys1,1,tR,SFstate,255);
VSysInit(&Sys2,1,tR,RFstate,255);

⋮

int Main_Interrupt_Service_Routine(void){

⋮

//the whole system
Aux=VAction(&Sys1,VPIAction(
    &Vector_PI1,VSub(R,Aux)));
U=VAction(&Sys2,VPIAction(
          &Vector_PI2,VSub(Aux,U)));

⋮

return (0);
}
```

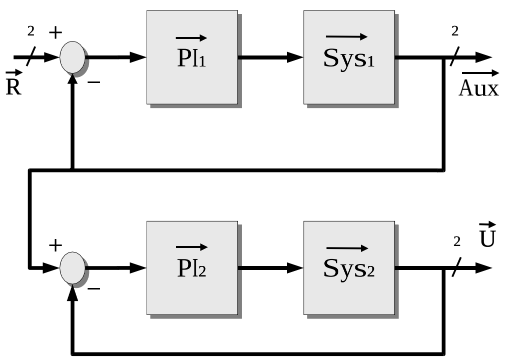

FIGURE 10: *System example. The code for this system is commented as "the whole system". One can build different systems as easy as block diagrams.*

The main objective of a trajectory controller is to force the machine stator flux and the machine stator current to follow steady state trajectories. Fig. 11 shows the moment when a pattern change is produced. The dotted trajectory corresponds to the steady state stator current and the difference is clear between it and the present stator current trajectory. One can sees that in Fig. 12, after the transition, the trajectory controller changes the pulses in order to correct the DME.

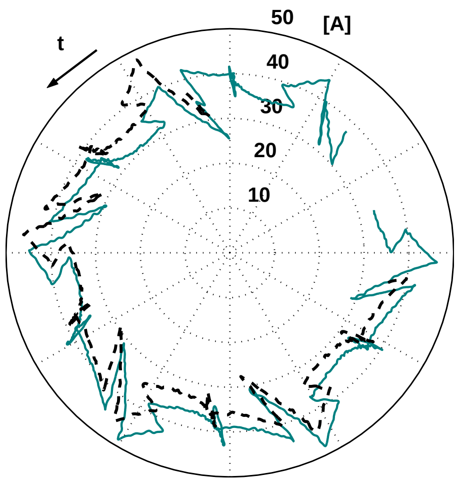

FIGURE 11: *Current transient without trajectory controller.*

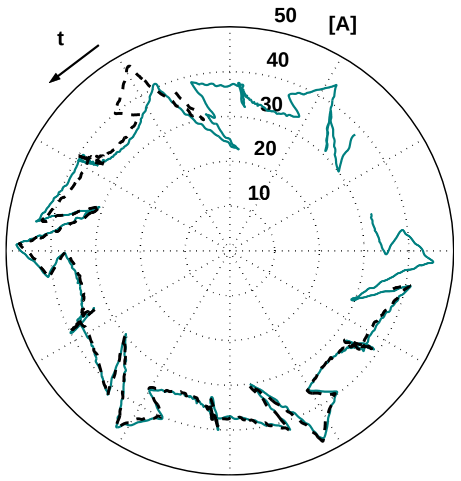

FIGURE 12: *Current transient with trajectory controller.*

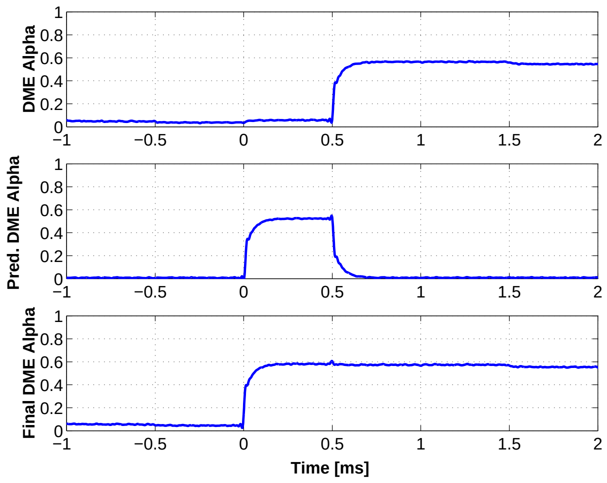

FIGURE 13: *DME Alpha component. Zero seconds is the begining of the next pattern. The first order response is owing to an output filter on the DA converter.*

5 Comments and conclusions.

The combination of a micro-controller and a PC system under real time Linux proved that it can keep the ***hard*** real time and makes it possible to have peripherals (like AD/DA converters, PWM module, digital ports, etc.) which can be easily handled by the micro-controller's software. The power of a PC under real time Linux give us the possibility to make almost any project in this area due to its fast interrupt answers and its stability.

It would be of great personal interest to see a powerful micro-processor or DSP running under real time Linux. One can build a complete system with AD/DA peripherals and ethernet for communication. There are many applications where it is necessary for more than one calculation source and if we connect these systems and a PC as a terminal though a LAN it would be an extremely powerful and comfortable work station for research.

Acknowledgements.

The author acknowledge the theoretic support of the Univ.-Prof. em. Dr.-Ing. Joachim Holtz and the Electrical Machines and Drives laboratory in the Wuppertal University.

References

[1] J. Holtz, *"Pulse width Modulation - A Survey"*, IEEE Transanctions on Industry Electronics, Vol. 39 pp. 410-420, Oct. 1992.

[2] J. Holtz and B. Beyer, *"Fast Current Trajectory Tracking Control Based on Synchronous Optimal Pulsewidth Modulation"*, IEEE Transaction on Industry Applications Vol. 31 pp. 1110-1120, Sep./Oct. 1995.