Dead-time Elimination when Operating in 3-Phase Space-Vector PWM Mode

Michael E. Aiello

May 19, 2019

Keywords: Space-Vector PWM, AC Motor Drives

Abstract

Dead-time control or time delay between the switching of adjacent MOSFET or IGBT devices in a bridge circuit is of mandatory concern in the operation of single or multi-level converters. Traditional methods for controlling the delay between top and bottom switches made the assumption that the current flow through the switches was always present. As a result, delay was aways added when turning on the adjacent switch while turning off the switch that currently supplied current in the forward direction. This so called *unenlightened* mode for controlling opposing switches can introduce non-linearity in control system designed into systems that require a high level of positioning accuracy with very low friction (e.g. Horizontal air bearing systems controlled with 3-phase linear motors).

Here I present some insight into how the *dead-band* effect caused by this delay effects operation amplifier operating in Space-Vector PWM mode. It should be noted that this information can also be applied to traditional carrier based PWM operation. However, it is more intuitive to demonstrate this principal using Space-Vector PWM (SVPWM). Specifically a particular form of this modulation I have termed *minus side referenced* SVPWM mode.

1 Introduction

The reader should refer to document [1] for a concise background into SVPWM implementation. As for the elimination (or reduction) of the *dead-band* effect, numerous papers have been presented that describe a technique for adding an offset to the PWM command reference based on the direction of current flow in in the switches. One paper that describes this technique is presented in document [2] and is expanded here.

Generally, the technique described in [2] and similar papers, presents the validation of this implementation based on simulation tools that represent the circuit

in abstract form (e.g., ideal switching devices and loads that do not accurately describe in detail motor characteristics like mutual inductance).

In this presentation, the simulation is performed using a special *SPICE* mode of the simulator. Unlike the simulation described in [1], SVPWM operation in implemented entirely as a *SPICE* time stepping algorythm. As such, no ODE modeling and no control of the motor as a motion element is presented. Here, the motor is characterized as *stalled*. The driving voltage is applied *open-loop* with magnitude and frequency selected to demonstrate the effects of the dead-time compensation control. The operation of the circuit converges to *steady-state* within 5 mSec.

2 Eliminating Dead-time in a Three Phase Bridge Circuit

A simplified diagram of a three phase bridge is shown in Figure 1 below. Here switch S1 through S6 can represent either Power MOSFET or IGBT technology. In both cases it is assumed that a free-wheeling diode is incorporated in each switch [1]. In the case of both MOSFET or IGBT, there is a condition of stored charge in the gate of the device that limits the ability to instantaneously change the forward on state between S1/S4, S3/S6 and S5/S2 pairs. It is this stored charge that predominant in determining the required delay between changes in *on state* in each pair. Finally, it should be noted that results presented in this paper based on the dead-time compensation described in [2] tends to be closer to that expected in an actual implementation due to the fact that the entire bridge is simulated using a *SPICE-like* algorithm.

A diagram of the circuit under simulation is shown in Figure 1. The labeling of the components in this circuit are referenced by various graphs and sections of code presented below. The parameters for this circuit are presented in Table 1

[1] In the case of Power MOSFET's the diode is inherent in the structure of the silicon.

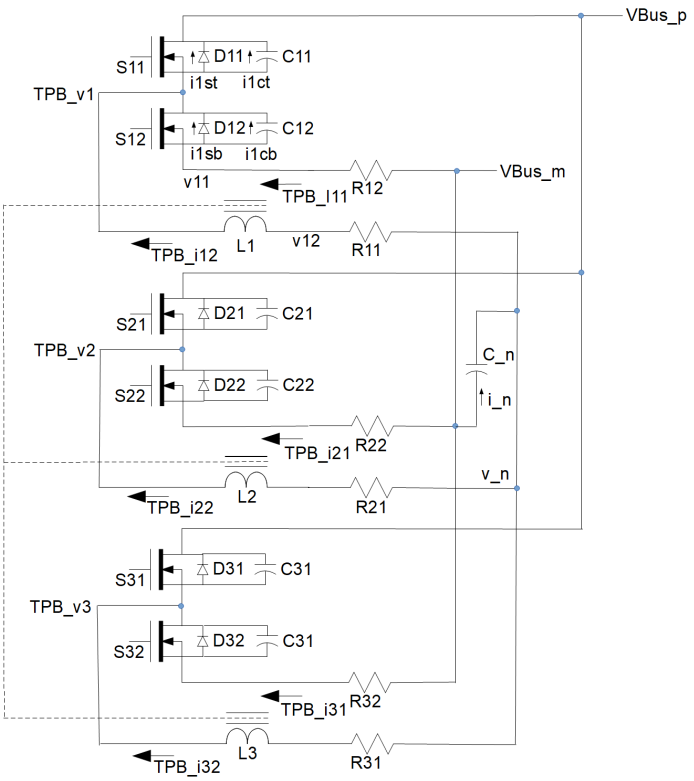

Figure 1: Diagram of a three phase PWM bridge circuit.

A vector diagram of SVPWM state transitions (for a single level converter) is shown in Figure 2 below. Here operation of the transition states is shown for Sector 1 for some arbitrary state transition times defined as T0, T1 and T2. The plots presented in Figures 6 and 7 are based on a time snap shot of SVPWM operation in Sector 4 of Figure 2 below.

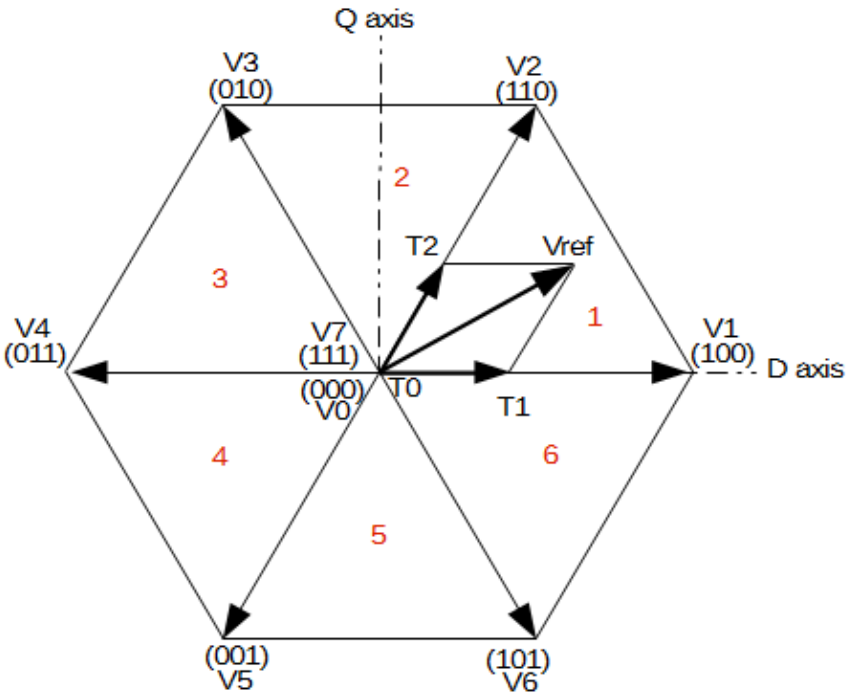

Figure 2: Switching vectors of SVPWM (single level) with the six associated sectors shown in red. For all tests presented in this paper, the PWM is minus side reference such that only zero vector **(000)** is used. (Drawing courtesy of [1]

For operation under the ideal condition where dead-time is set to zero (see Figure 16, at comment **"No dead time test"** we have near ideal sinusoidal values of phase currents **tpb_i12**, **tpb_i22** and **tpb_i32**. This is shown in Figure 3.

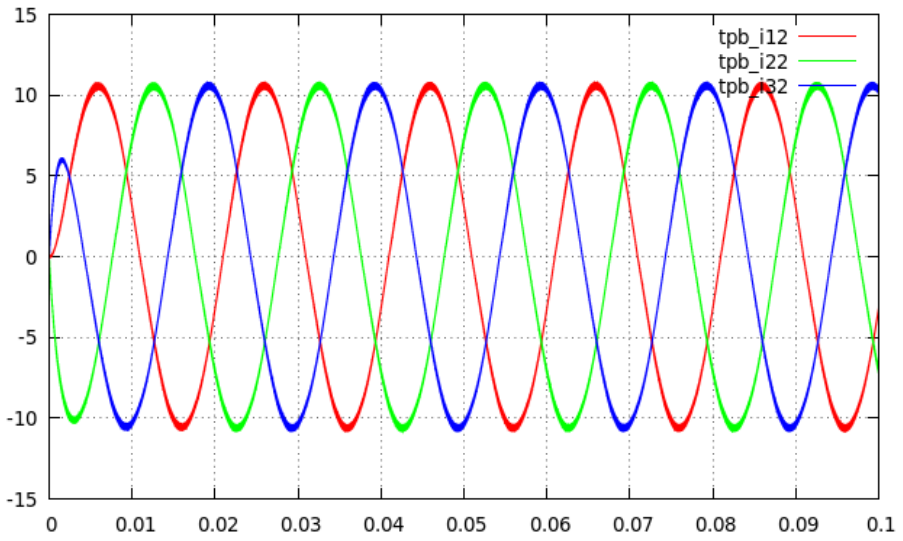

Figure 3: Operation of the circuit depicted in Figure 1 showing the phase currents. Circuit is operating in the ideal condition with no switching *dead-time*

The small size of the time step allows for an accurate representation of ripple current. This can be seen in Figure 4 for phase current **tpb_i12**.

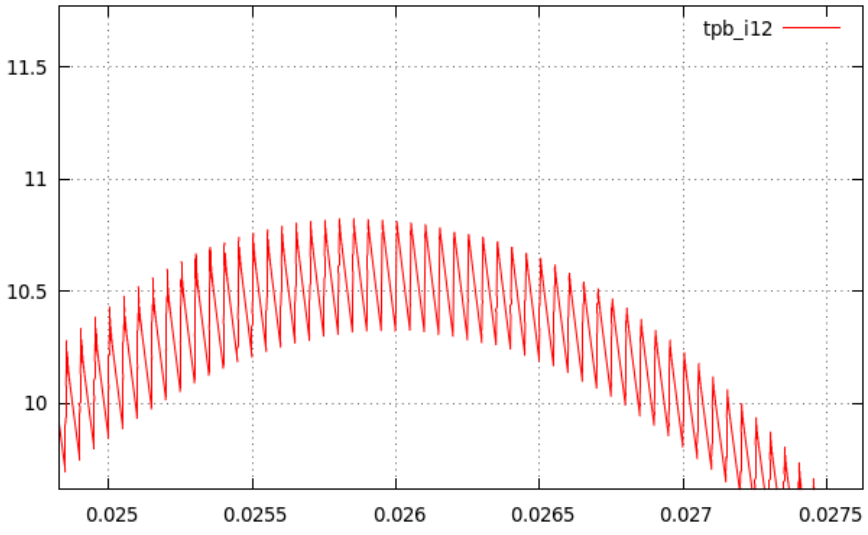

Figure 4: Close-up of phase current **tpb_i12** in Figure 3 showing the switching ripple current.

A snapshot in time is taken in Sector 4 of the SVPWM state. At this point, values of the phase voltages **tpb_v1**, **tpb_v2** and **tpb_v3** are shown in Figure 5. Note that in Sector 4, phase voltage **tpb_v1** maintains an effective value of zero volts during the entire switching cycle.

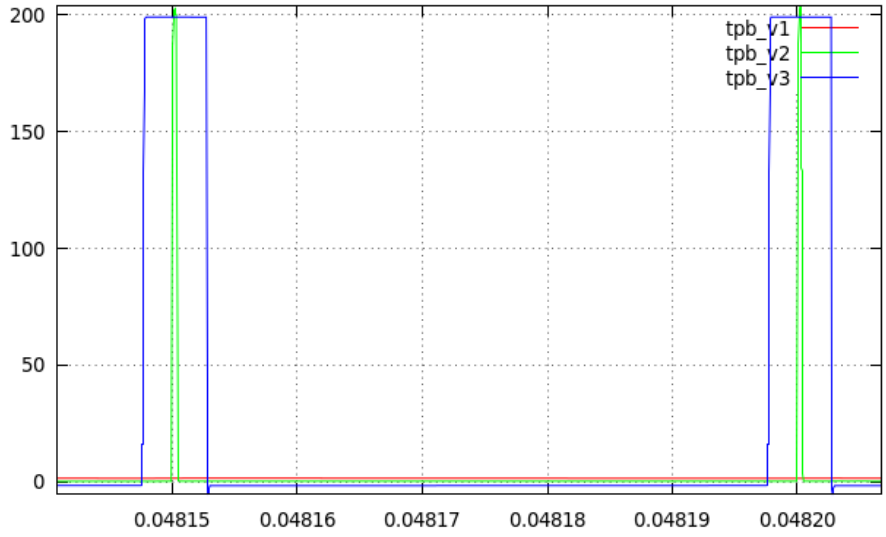

Figure 5: Phase voltages of the circuit depicted in Figure 1. The PWM is operating in Sector No. 4 (see Figure 2)

Details on how the currents are sampled are shown in Figures 6 and 7 below for current **tpb_i11** (the method for **tpb_i21** and **tpb_i31** are similar). The logic used to determine the sampling point is described in Figure 16, at comment **"Smart dead time test"**.

This sampling of the current in the minus side bus (using resistors R12, R22 and R32 shown in Figure 1) is necessary for controlling the dead-time compensation control as discussed below.

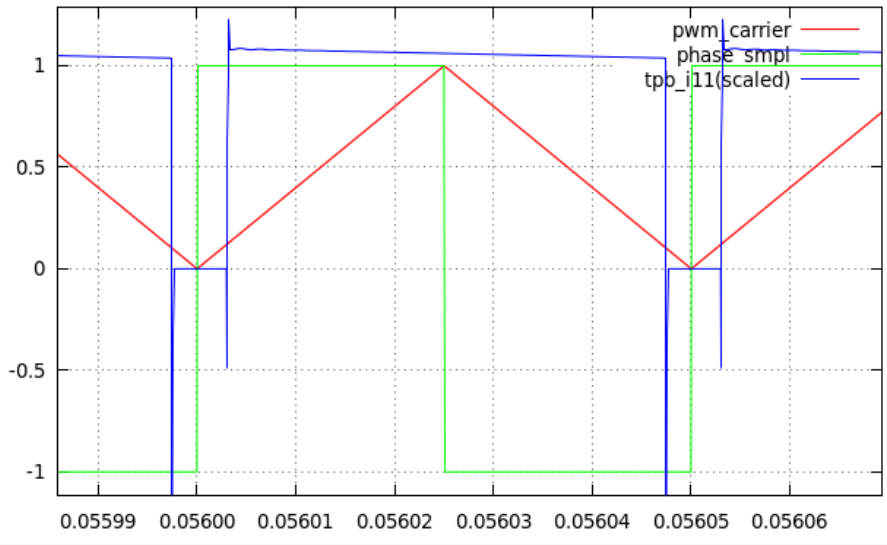

Figure 6: Minus side leg current **tpb_i11** flowing through sense resistor **R12** (see Figure 1 and Figure 16) during a point in the switching cycle when current is flowing *out of* VBus_m. This sample is taken on the falling edge of **phase_smpl**.

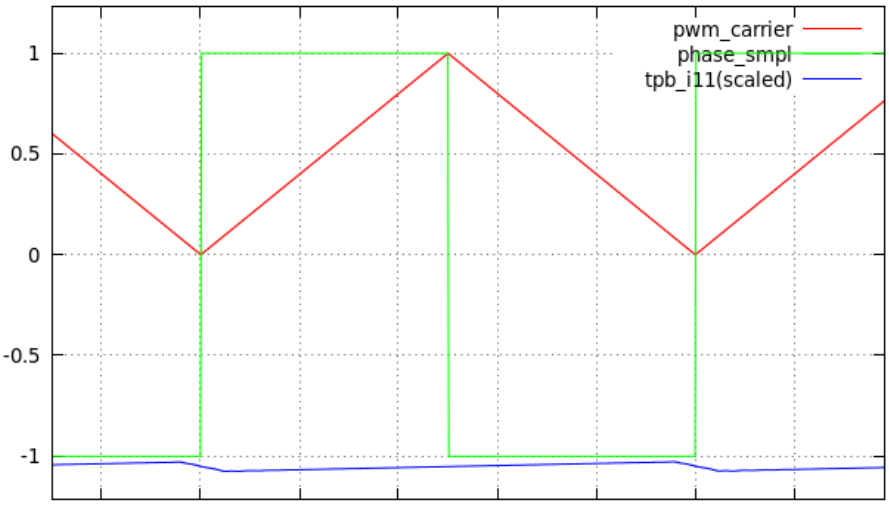

Figure 7: Minus side leg current **tpb_i11** flowing through sense resistor **R12** (see Figure 1 and Figure 16) during a point in the switching cycle when current is flowing *into* VBus_m. The sample is taken on the falling edge of **phase_smpl**.

The mechanism for sampling the currents in each leg of the bridge as described in Figures 6 and 7 is necessitated by the need to know the direction of current flow. This is used to apply the offset to the command reference to correct the dead-time error. This will be described in more detail below.

There is of course the need to know the magnitude as well as the direction of the current in each leg for the purpose of controlling current in a closed loop system which is described in detail in paper [3]. This is done by simply adding a *Zero Order Hold* (e.g. sampling ADC) to the circuit. The application of this to signal to the current **tpb_i11** is shown in Figure 8 below.

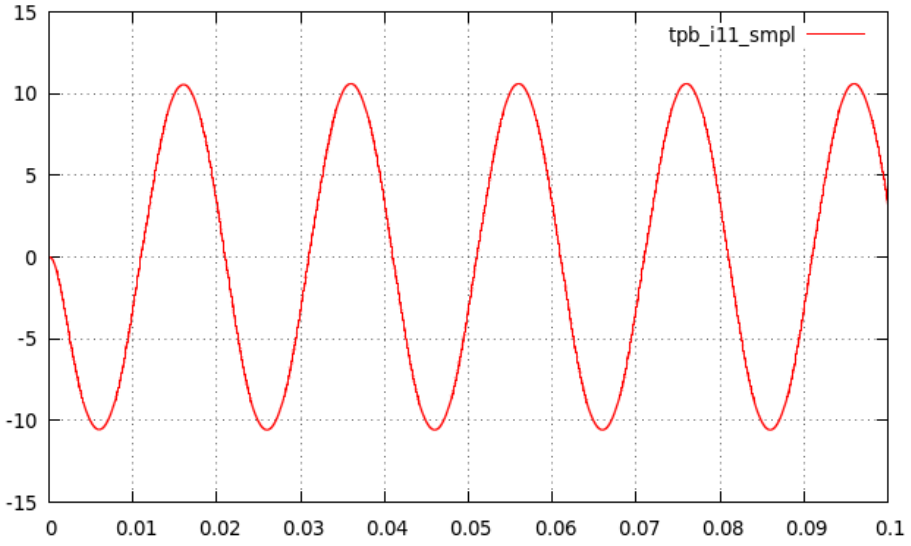

Figure 8: Minus side leg current **tpb_i11** sampled through sense resistor **R12** and held for current switch cycle (e.g. *Zero Order Hold* produces a current similar to **tpb_i12** in Figure 3) without switching ripple. This signal would be used as feedback of Phase 1 current to the control system. (Similarly, this would be done for Phase 2 and Phase 3 using resistors **R22** and **R32** respectively).

As described in detail in [3], the control of the bridge is based on Space Vector PWM with a bias to the minus side of the bus. The reference command for Phase 1 producing the currents and voltages illustrated up to this point is shown in Figure 9 below. The reference commands for Phase 2 and 3 are similar, shifted by 120 and 240 degrees respectively. (Note that for all tests described so far, the reference command is generated by a sinusoidal command generated by the code labeled **"Sweep test"** in Figure 14).

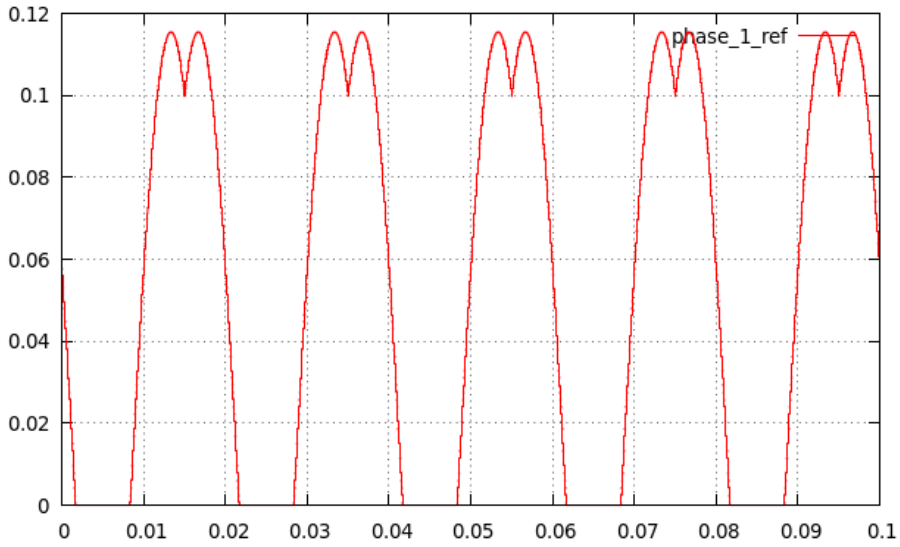

Figure 9: Command reference for current **tpb_i12** (see Figure 14 and Figure 16). The comparison of this signal with signal **pwm_carrier** produces the switchings control for **S11** and **S12**.

Up to this point, the operation of the bridge shown in Figure 1 and the accompanying waveforms above have been with operation of the circuit under the ideal condition of *zero dead time* control (no delay between the turning on and off of the switch pairs S11/S12, S21/S22 and S31/S32.

We next apply 3 uSec of dead-time to the switch pairs S11/S12, S21/S22 and S31/S32. This is demonstrated by the code labeled **"Legacy dead time test"** in Figure 16 and the code that controls the dead-time control as presented in Figure 17. The result is the distorted Phase 1 current **tpb_i12** shown in Figure 10 below [2].

Notice that along with the distortion of the waveform is a reduction in the amplitude (from approximately + 10 to + 5 amps peak-to-peak). Under these conditions, if the circuit in Figure 1 were operating in *closed-loop*, the control would tend to compensate for the distortion and reduction of peak current and would create a result that is closer to the ideal condition of pure sinusoidal.

Indeed, if the control was without delay, simply closing the current loop would be an adequate solution for the degradation caused by the insertion of dead-time in the switches. However, in reality a typical control system has delay (typically 50 uSec in standard servo control systems). Thus, it can be said that under

[2]The effect on **tpb_i22** and **tpb_i32** would be similar

these realistic conditions, the control is non-linear.

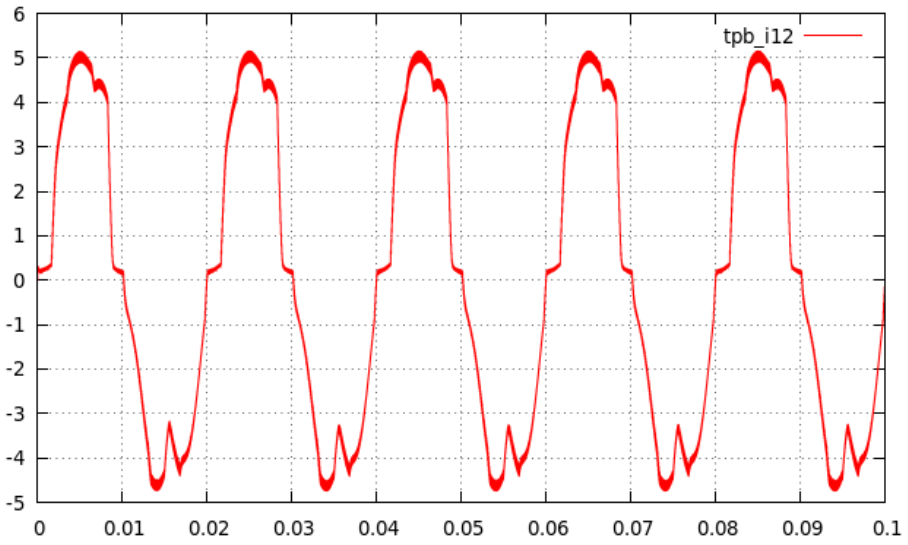

Figure 10: Operation of cirucit shown in Figure 1 with *uncompensated dead-time* control. Only phase **tpb_i12** is shown (**tpb_i22** and **tpb_i32** are similar).

Now, we apply the dead-time compensation control in a way that is similar to that described in reference [2]. The result is the waveform shown in Figure 11 below. Notice that unlike the the dead-time compensation demonstrated in [2], the result is not a recreation of an ideal sinusoidal signal. This is because the simulation done here models much more precisely, the characteristics of an actual three phase bridge and motor load. Interesting enough however is the peak-to-peak amplitude of the current is restored to approximately + 10 amps, similar to that shown for the ideal operating condition of Figure 3.

The code that describes this compensation is labeled **"Smart dead time test"** in Figure 16 along with the code that controls the dead-time control as presented in Figure 17.

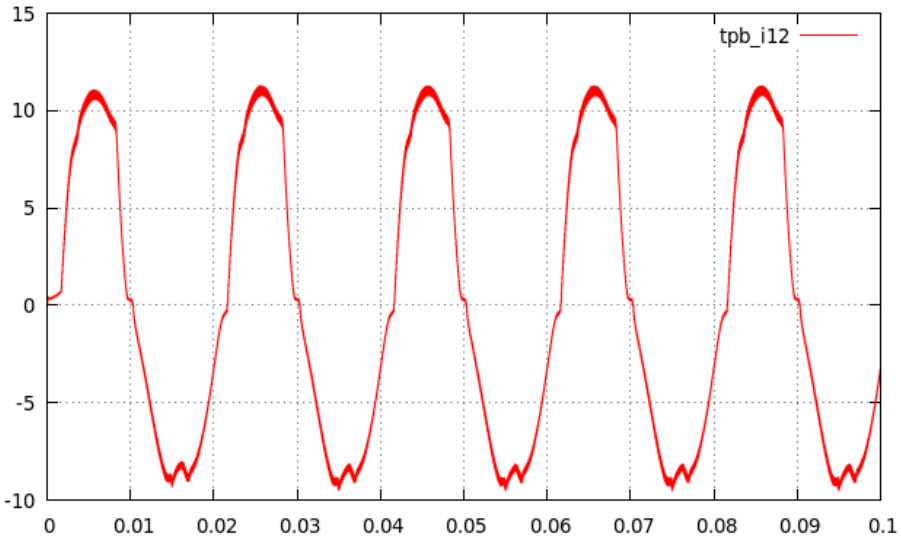

Figure 11: Operation of the circuit shown in Figure 1 with *compensated dead-time* control. Only phase **tpb_i12** is shown.

For the next test, the control is set to create a condition similar to what is termed as *PID "jitter"* found in typical digital closed loop servo systems. This is done by enabling operation with the code labeled **"Dither test"** in Figure 14. Figure 12 below shows the result of phase voltages generated using this test.

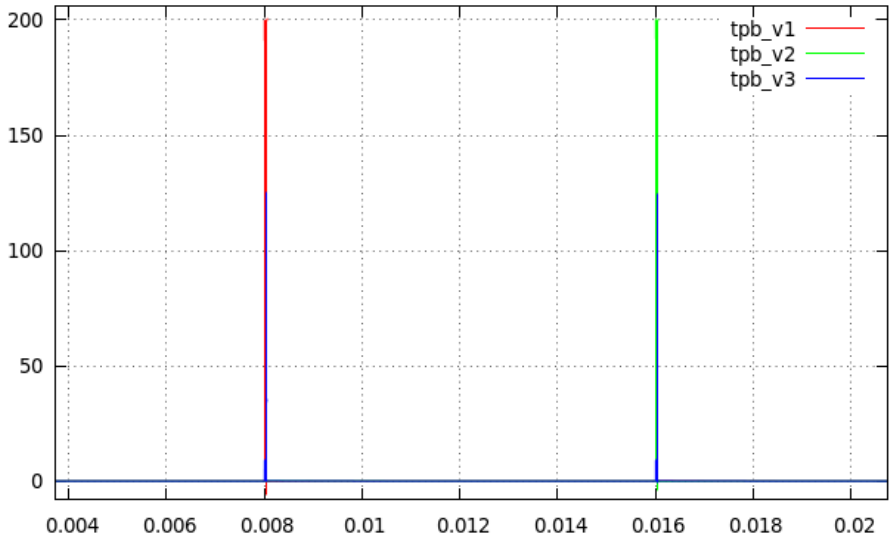

Figure 12: Plot of phase voltages **tpb_v1, tpb_v2** and **tpb_v3** with the "Dither test" enabled in Figure 14. This test allows only one 50 uSec pulse set every 80 mSec. Typical of the minus side referenced SVPWM, only two out of three phases operate at any given time. Here phase 1 and 3 are pulsed at time .008 and phase 2 and 3 pulse at time .016.

Of significance is the current produces from the resulting voltages of Figure 12. These current are shown in Figure 13 below

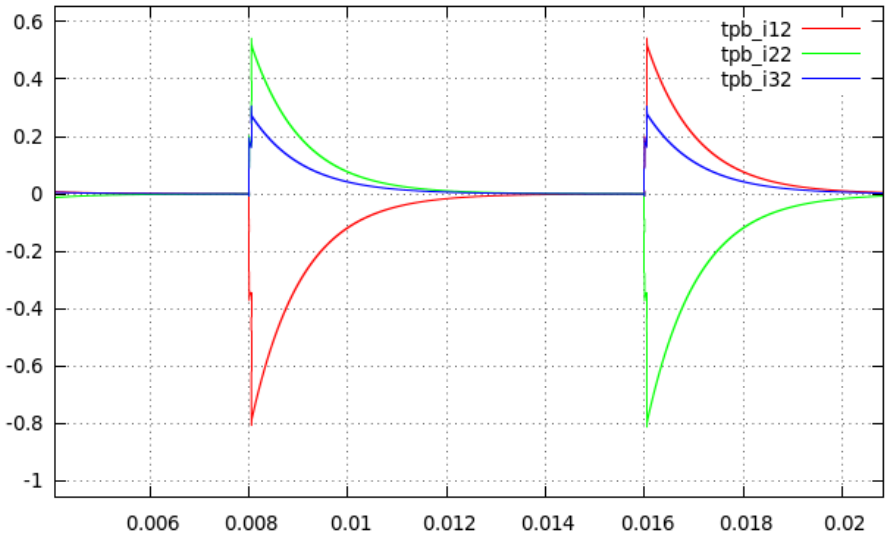

Figure 13: Plot of phase currents **tpb_i12, tpb_i22** and **tpb_i32** with the "Dither test" enabled in Figure 14. These plots coincide with the voltage pulses shown in Figure 12. The currents go to zero before the beginning of the next pulse set.

Notice that all three currents are allowed to go to zero before the next set of voltage pulses.

The importance here is that in typical PWM implementations (which includes classical *carrier* based as well as SVPWM, both zero vectors **(000)** and **(111)** (see Figure 2) are used in the switching topology. When vectors **(000)** and **(111)** are the sole switching states (indicating zero voltage to the motor), the *edge* between the states **(000)** and **(111)** are side-by-side (e.g. all three phases enter the dead-time interval concurrently). This results in all three phases entering the high impedance state at the same time.

In high voltages systems (typically off-line, 480 VAC operation) under this condition, components of the 60 Hz rectification can be coupled onto the motor leads and can show up as low frequency distortion in the current feedback to the servo control system. This cannot happen with *minus side referenced* SVPWM since referring back to Figure 12, there is always at least one phase connected to the minus DC bus at any given time under the condition of continuous or dis-continuous current flow to the motor.

A Simulation Details

A.1 Overview

This section provides some of the details of the simulation that was described in the previous section. As mentioned previously, a special *SPICE-like* mode was added to the simulator described in [3] which allows both ODE and first order time stepping within the same simulation run. For the simulation presented in this paper, the first order time stepping mode was used explicitly.

Figures 14, 15, 16 and 17 present some details in the control functions driving the simulator. Notice that all these functions are provided the parameter **t** (time) that runs concurrently with the time stepping algorithm. Except for the code in Figure 14, all functions run at the simulation step time. The code in Figure 14 is a special *CtrlObject* that is set to run every 50 uSec.

Figures 18, 19, 20 and 21 provide details of the time stepping algorithm used in the simulation.

Finally, Table 1 provides the parameters of the components (switching bridge and load) illustrated in Figure 1.

A.2 Simulation Objects

As previously mentioned, the entire circuit described in Figure 1 is controlled open-loop where the values for all circuit currents approach steady-state within a short time after the simulation is started. The *driving function* is shown in Figure 14 below.

```
void VDqCmd::CtrlFunction(double t)
{

#define DITHER_TEST //Define for "Dither test"
#ifdef DITHER_TEST
        //"Dither test"

    static int CmdPolarityIdx = 0;
        //Generate a 50 uSec "ditter" pulse every 8 mSec.
    if(CmdPolarityIdx ¡ 1)
    {
        vcmd_mag = 25.0;
        vcmd_ang = PI/2 + PI/3;
    }
    else if((CmdPolarityIdx ¿= 1) && (CmdPolarityIdx ¡ 160))
    {
        vcmd_mag = 0;
        vcmd_ang = 0;
    }
    else if((CmdPolarityIdx ¿= 160) && (CmdPolarityIdx ¡ 161))
    {
        vcmd_mag = 25.0;
        vcmd_ang = - PI/2 + PI/3;
    }
    else if((CmdPolarityIdx ¿= 161) && (CmdPolarityIdx ¡ 320))
    {
        vcmd_mag = 0;
        vcmd_ang = 0;
    }

    if(CmdPolarityIdx != 319)
    {
        CmdPolarityIdx++;
    }
    else
    {
        CmdPolarityIdx = 0;
    }

#else
        //"Sweep test"

        //For now, we provide constant commands for magnitude
        //and vary angle open loop.
    vcmd_mag = 20.0; //Volts
    Freq = 50.0; //Hz
    double Freq;
    Period = 1.0 / Freq;
    theta = 2.0 * PI * t / Period;
    vcmd_ang = PI/2 + theta;

        //generate the space vector references.
    SpaceVectorControl(&phase_1_ref, &phase_2_ref, &phase_3_ref);

#endif
}
```

Figure 14: Process that provides the sinusoidal command signal processed by the space vector control algorithm.

The carrier frequency for the SVPWM algorithm is provided by Figure 15 below.

```
void TriangleWave::SrcFunction(double t)
{

        //Generate the triangle wave carrier signal.
    t_mod += (t - t_prev);
    t_prev = t;

    if(t_mod ¿ HALF_PWM_CYCLE)
        t_mod -= HALF_PWM_CYCLE;
        PwmRampDir *= -1.0;
    }

        //Carrier signal (triangle wave) to Phase 1,2 and 3 PWM control.
    pwm_carrier = PWM_GAIN*(t_mod / HALF_PWM_CYCLE - .5) * PwmRampDir + PWM_OFFSET;

        //Current sampling signal to Phase 1,2 and 3 PWM control.
    phase_smpl = PwmRampDir;

}
```

Figure 15: Process that provides for the generation of the carrier signal used by all three phases 1, 2, and 3.

There are three modes of simulation described in the previous section *No dead-time*, *legacy* (without dead-time compensation) and *smart* (with dead-time compensation). All three of these are controlled using the function described in Figure 16 below.

```
void PwmA::SrcFunction(double t)
{
	//Both definitions commented out produces "No dead time test".
//#define DEADTIME_CTRL_LEGACY //Define for "Legacy dead time test"
#define DEADTIME_CTRL_SMART //Define for "Smart dead time test"
#if defined(DEADTIME_CTRL_LEGACY)
		//"Legacy dead time test"
	bool EnableSwitching = DeadtimeGenerator(t, phase_1_ref ¿= pwm_carrier);
	if(phase_1_ref ¿ pwm_carrier)
	{
		GateCtrl_11 = EnableSwitching;
		GateCtrl_12 = 0;
	}
	else
	{
		GateCtrl_11 = 0;
		GateCtrl_12 = EnableSwitching;
	}
#elif defined(DEADTIME_CTRL_SMART)
		//"Smart dead time test"
		//For now, we only take one sample at the center of the flyback current (where
		//current is flowing through D12 and sensed by R12in Figure ????).
		//Additionally we may take multiple samples before and after this point and average them out.
	if((phase_smpl_prev== 1) and (phase_smpl == -1))
	{
		tpb_i11_smpl = tpb_i11;
		if(tpb_i11_smpl ¿= 0
		{
			ref_offset = PWM_GAIN *
			DEADTIME_VALUE / (HALF_PWM_CYCLE * 2); //PWM_GAIN * 3uS/50uS
		}
		else
		{
			ref_offset = 0;
		}
	}
	phase_smpl_prev = phase_smpl;
	bool EnableSwitching = DeadtimeGenerator(t, (phase_1_ref + ref_offset) ¿= pwm_carrier);
	if((phase_1_ref + ref_offset) ¿ pwm_carrier)
	{
		GateCtrl_11 = EnableSwitching;
		GateCtrl_12 = 0;
	}
	else
	{
		GateCtrl_11 = 0;
		GateCtrl_12 = EnableSwitching;
	}
#else
		//"No dead time test"
	if(phase_1_ref ¿ pwm_carrier)
	{
		GateCtrl_11 = 1;
		GateCtrl_12 = 0;
	}
	else		{
		GateCtrl_11 = 0;
		GateCtrl_12 = 1;
	}
#endif
}
```

Figure 16: Process that provides the simulation tests for three control scenarios, no-dead time, *legacy* (or traditional) dead time, and *Smart* dead time.

For *legacy* and *smart* operation, the dead-time is controlled using the function described in Figure 17 below.

```
bool PwmA::DeadtimeGenerator(double t, bool state)
{
	static double t_previous = DEADTIME_VALUE;
	static bool state_previous = 0;
	//"state" is the difference between the adjusted phase reference
	//reference and pwm_carrier.
	if(state != state_previous)
	{
		t_previous = t + DEADTIME_VALUE;
	}
	state_previous = state;
	if(t ¿= t_previous)
	{
		t_previous = t;
		return TRUE;
	}
	else
	{
		return FALSE;
	}
}
```

Figure 17: Function that controls PWM A (phase 1) dead time interval. Functions for PWM B (phase 2) and C (phase 3) are similar.

A.3 Time Step Model

The details of the time-stepping algorithm are presented in Figures 18, 19, 20 and 21 below. Note the form is *first-order time step* which is used by typical SPICE simulation tools. Here, variable h is the time-step quantity.

$$\frac{tpb\_v1}{rS11} - \frac{VBus_p}{rS11} + \frac{tpb\_v1}{rD11} - \frac{VBus_p}{rD11} = i1st \quad (1)$$

$$\frac{v11}{rS12} - \frac{tpb\_v1}{rS12} + \frac{v11}{rD12} - \frac{tpb\_v1}{rD12} = i1sb \quad (2)$$

$$tpb\_i12 + tpb\_i11 = i13 \quad (3)$$

$$i1st + i1ct = i13 \quad (4)$$

$$i1sb + i1cb = tpb\_i11 \quad (5)$$

$$tpb\_v1\frac{C11}{h} - VBus_p\frac{C11}{h} - tpb\_v1_{prev}\frac{C11}{h} + VBus_{p\_prev}\frac{C11}{h} = i1ct \quad (6)$$

$$v11\frac{C12}{h} - tpb\_v1\frac{C12}{h} - v11_{prev}\frac{C12}{h} + tpb\_v1_{prev}\frac{C12}{h} = i1cb \quad (7)$$

$$\frac{VBus_m}{R12} - \frac{v11}{R12} = tpb\_i11 \quad (8)$$

$$\begin{aligned}(L1\frac{tpb\_i12}{h} - L1\frac{tpb\_i12_{prev}}{h}) - \\ (M12\frac{tpb\_i22}{h} - M12\frac{tpb\_i22_{prev}}{h}) - \\ (M31\frac{tpb\_i32}{h} - M31\frac{tpb\_i32_{prev}}{h}) = v12 - tpb\_v1\end{aligned} \quad (9)$$

$$\frac{v\_n}{R11} - \frac{v12}{R11} = tpb\_i12 \quad (10)$$

Figure 18: The subset of Spice equations associated with the control of switchs S11 and S12 for phase 1.

$$\frac{tpb\_v2}{rS21} - \frac{VBus_p}{rS21} + \frac{tpb\_v2}{rD21} - \frac{VBus_p}{rD21} = i2st \quad (11)$$

$$\frac{v21}{rS22} - \frac{tpb\_v2}{rS22} + \frac{v21}{rD22} - \frac{tpb\_v2}{rD22} = i2sb \quad (12)$$

$$tpb\_i22 + tpb\_i21 = i23 \quad (13)$$

$$i2st + i2ct = i23 \quad (14)$$

$$i2sb + i2cb = tpb\_i21 \quad (15)$$

$$tpb\_v2\frac{C21}{h} - VBus_p\frac{C21}{h} - tpb\_v2_{prev}\frac{C21}{h} + VBus_{p\_prev}\frac{C21}{h} = i2ct \quad (16)$$

$$v21\frac{C22}{h} - tpb\_v2\frac{C22}{h} - v21_{prev}\frac{C22}{h} + tpb\_v2_{prev}\frac{C22}{h} = i2cb \quad (17)$$

$$\frac{VBus_m}{R22} - \frac{v21}{R22} = tpb\_i21 \quad (18)$$

$$\begin{aligned}(L2\frac{tpb\_i22}{h} - L2\frac{tpb\_i22_{prev}}{h}) - \\ (M12\frac{tpb\_i12}{h} - M12\frac{tpb\_i12_{prev}}{h}) - \\ (M23\frac{tpb\_i32}{h} - M23\frac{tpb\_i32_{prev}}{h}) = v22 - tpb\_v2\end{aligned} \quad (19)$$

$$\frac{v\_n}{R21} - \frac{v22}{R21} = tpb\_i22 \quad (20)$$

Figure 19: The subset of Spice equations associated with the control of switchs S21 and S22 for phase 2.

$$\frac{tpb\_v3}{rS31} - \frac{VBus_p}{rS31} + \frac{tpb\_v3}{rD31} - \frac{VBus_p}{rD31} = i3st \quad (21)$$

$$\frac{v31}{rS32} - \frac{tpb\_v3}{rS32} + \frac{v31}{rD32} - \frac{tpb\_v3}{rD32} = i3sb \quad (22)$$

$$tpb\_i32 + tpb\_i31 = i33 \quad (23)$$

$$i3st + i3ct = i33 \quad (24)$$

$$i3sb + i3cb = tpb\_i31 \quad (25)$$

$$tpb\_v3\frac{C31}{h} - VBus_p\frac{C31}{h} - tpb\_v3_{prev}\frac{C31}{h} + VBus_{p\_prev}\frac{C31}{h} = i3ct \quad (26)$$

$$v31\frac{C32}{h} - tpb\_v3\frac{C32}{h} - v31_{prev}\frac{C32}{h} + tpb\_v3_{prev}\frac{C32}{h} = i3cb \quad (27)$$

$$\frac{VBus_m}{R32} - \frac{v31}{R32} = tpb\_i31 \quad (28)$$

$$(L3\frac{tpb\_i32}{h} - L3\frac{tpb\_i32_{prev}}{h}) -$$
$$(M31\frac{tpb\_i12}{h} - M31\frac{tpb\_i12_{prev}}{h}) -$$
$$(M23\frac{tpb\_i22}{h} - M23\frac{tpb\_i22_{prev}}{h}) = v32 - tpb\_v3 \quad (29)$$

$$\frac{v\_n}{R31} - \frac{v32}{R31} = tpb\_i32 \quad (30)$$

Figure 20: The subset of Spice equations associated with the control of switchs S31 and S32 for phase 3.

$$tpb\_i12 + tpb\_i22 + tpb\_i32 = i\_n \quad (31)$$

$$vBus_m\frac{C\_n}{h} - v\_n\frac{C\_n}{h} - vBus_{m\_prev}\frac{C\_n}{h} + v\_n_{prev}\frac{C\_n}{h} = i\_n \quad (32)$$

Figure 21: The subset of Spice equations that completes the circuit for phases 1, 2 and 3.

Finally, a listing for all simulation parameters used in this paper is provided in Table 1 below.

Parameter	Value
L1,L2,L3	1.0 mH
M12,M23,M31 (mutual ind.)	.2 mH
R11,R21,R31	1.0 Ω
R12,R22,R32	.1 Ω
C11,C12,C21,C22,C31,C32	.005 uF
C_n	.0005 uF
Vbus_p	200.0 VDC
Vbus_n	0 VDC
Sxx On Resistance	.1 Ω
Dxx On Resistance	.1 Ω
Sxx Dead-time (when enabled)	3.0 uS
SVPWM Switching Freq	20.0 KHz
Simulation time-step (**h** min.)	.05 uS

Table 1: Parameters for equations described in Figures 18, 19, 20 and 21, and circuit illistrated in Figure 1

References

[1] *Project #2 Space Vector PWM Inverter, February 20, 2005.*
Jin-Woo Jung
Mechatronic Systems Laboratory, Department of Electrical and Computer Engineering, Ohio State University
`http://meaconsultingdotorg.files.wordpress.com/2015/12/spacevector_pwm_inverter.pdf`

[2] *The Analysis and Compensation of Dead-Time Effects in PWM Inverters*
Seung-Gi Jeong
IEEE Transactions on Industrial Electronics, Vol. 38, No. 2, April 1991
`https://meaconsultingdotorg.files.wordpress.com/2019/04/the-analysis-and-compensation-of-dead-time-effects-in-pwm-inverters.pdf`

[3] *Implementation of an Advanced AC Brushless Motor Controller for Use in High Reliability Applications*
Michael E. Aiello, April 2 2016.
`https://meaconsultingdotorg.files.wordpress.com/2015/12/controller.pdf`