Journal of Power and Energy Engineering, 2025, 13(1), 32-46
https://www.scirp.org/journal/jpee
ISSN Online: 2327-5901
ISSN Print: 2327-588X

LQR Discrete Time Control of a Buck Converter Using a Non-Minimal State Space Representation

Richard Tymerski

Department of Electrical & Computer Engineering, Portland State University, Portland, Oregon, USA
Email: tymerski@ee.pdx.edu

How to cite this paper: Tymerski, R. (2025) LQR Discrete Time Control of a Buck Converter Using a Non-Minimal State Space Representation. *Journal of Power and Energy Engineering*, **13**, 32-46.
https://doi.org/10.4236/jpee.2025.131003

Received: December 20, 2024
Accepted: January 23, 2025
Published: January 26, 2025

Abstract

This paper presents an advanced control strategy for DC-DC buck converters utilizing Non-Minimal State Space (NMSS) representation combined with Proportional-Integral-Plus (PIP) control, optimized through Linear Quadratic Regulator (LQR) design. The proposed approach leverages NMSS to eliminate the need for state observers, enhancing robustness against model mismatch and improving overall system performance. The PIP controller extends traditional PI control by incorporating additional dynamic feedback. Experimental results demonstrate that the NMSS-PIP-LQR controlled buck converter achieves excellent dynamic performance. The design procedure is fully documented, and microcontroller implementation issues are discussed.

Keywords

DC-DC Buck Converter, Non-Minimal State Space (NMSS), Proportional-Integral-Plus (PIP) Control, Linear Quadratic Regulator (LQR), F28069M Microcontroller

1. Introduction

Digital control techniques for DC-DC converters have seen significant advancements in recent years, offering improved performance and flexibility compared to traditional analog methods. These developments have been driven by the increasing demand for more efficient and reliable power conversion systems in various applications, including electric vehicles and renewable energy systems.

Several digital control methods have been applied to DC-DC converters, including [1]-[4].

1) Voltage-mode control [5]

2) Current-mode control [6] [7]

3) PID Control [8] [9]

4) Hysteretic control [10] [11]

5) Sliding mode control [12] [13]

6) Model predictive control [14]

7) Fuzzy logic control [15]

Among these digital control techniques, voltage-mode and current-mode control remain the most prevalent in industry due to their relative simplicity and well-established design procedures.

However, the approach [16] presented in this paper, utilizing Non-Minimal State Space (NMSS) representation combined with Proportional-Integral-Plus (PIP) control and optimized through Linear Quadratic Regulator (LQR) design, offers several advantages over traditional methods. This approach merits consideration for the following reasons [16].

1). Formulation eliminates the need for state observers, reducing sensitivity to model mismatch and improving overall system robustness.

2) PIP control extends traditional PI control, offering enhanced flexibility and performance, (particularly for higher-order systems or those with significant time delays).

3) The integration of LQR optimization allows for systematic tuning of controller parameters, balancing control effort and system performance.

By combining these advanced techniques, this paper aims to examine a control strategy for DC-DC buck converters that offers improved voltage regulation, transient response, and disturbance rejection compared to conventional methods while maintaining stability under various operating conditions.

The remainder of this paper is organized as follows: Section 2 presents the converter modelling approach. This starts with the state space averaged model and leads to the NMSS model representation. Section 3 undertakes the LQR optimal controller design. An overview of the implementation of the control law using a digital microcontroller is discussed in Section 4. More detailed practical information relating to our use of the TI F28069M microcontroller is presented in Section 5. The effectiveness of the design is shown in Section 6 where the load step response is examined. The conclusion follows in Section 7.

2. System Modelling

The schematic of a Buck dc-dc converter is shown in **Figure 1**. The circuit switches between two topologies depending on the position of the single pole double throw switch. The switch is operated cyclically with a period of T_s. The switch is in position 1 for a length of time of dT_s, where d, $0 < d < 1$, is the duty ratio. During the remainder of the period of length $(1-d)T_s$, the switch is in position 2. We will denote $d' \equiv 1-d$. We use a state space mathematical description of the two topologies.

During the first subinterval we have:

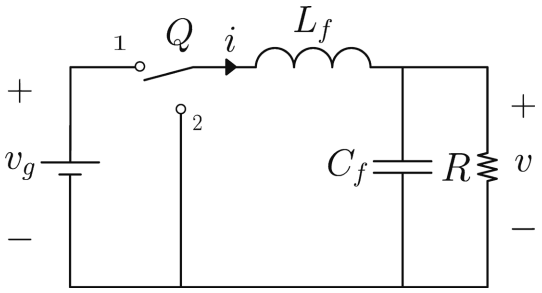

Figure 1. Buck DC-DC converter.

$$\frac{\mathrm{d}x}{\mathrm{d}t} = A_1 x + B_1 v_g \tag{1}$$

$$y = C_1 x + E_1 v_g \tag{2}$$

During the second subinterval we have:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = A_2 x + B_2 v_g \tag{3}$$

$$y = C_2 x + E_2 v_g \tag{4}$$

where x denotes a minimal length state vector, and v_g is the input voltage and y is the output. The state variables are typically the inductor current and capacitor voltage, such that $x = \begin{bmatrix} i & v \end{bmatrix}^{\mathrm{T}}$.

To represent the system as linear and time invariant the State Space Averaging (SSA) [17] model will be used. Variables d, x, v_g and y are assumed to undergo small variations $\hat{d}, \hat{x}, \hat{v}_g$ and $\hat{y}$ around their steady state operating point denoted by the capitalized symbols D, X, V_g and Y. The resulting small signal model is given by:

$$\frac{\mathrm{d}\hat{x}}{\mathrm{d}t} = A\hat{x} + B\hat{v}_g + B_d\hat{d} \tag{5}$$

$$\hat{y} = C\hat{x} + E\hat{v}_g + E_d\hat{d} \tag{6}$$

where

$$A = DA_1 + D'A_2 \tag{7}$$

$$B = DB_1 + D'B_2 \tag{8}$$

$$C = DC_1 + D'C_2 \tag{9}$$

$$E = DE_1 + D'E_2 \tag{10}$$

$$B_d = \left(A_1 - A_2\right)X + \left(B_1 - B_2\right)V_g \tag{11}$$

$$E_d = \left(C_1 - C_2\right)X + \left(E_1 - E_2\right)V_g \tag{12}$$

The steady state vector, X, is given from the SSA DC model:

$$X = -A^{-1}BV_g \tag{13}$$

The control input duty ratio to output voltage transfer function, $\frac{\hat{y}}{\hat{d}}$ is derived by using the state space quadruple $\left\{A, B_d, C, E_d\right\}$. For the Buck converter of

Figure 1 we find:

$$A=\begin{bmatrix} 0 & -\dfrac{1}{L_f} \\ \dfrac{1}{C_f} & -\dfrac{1}{RC_f} \end{bmatrix},\quad B_d=\begin{bmatrix} \dfrac{V_g}{Lf} \\ 0 \end{bmatrix},\quad C=\begin{bmatrix} 0 & 1 \end{bmatrix},\quad E_d=0 \tag{14}$$

To apply discrete time control, this continuous time system is discretized using the Matlab *c2d* function. The resulting discrete time state space model is then transformed into a discrete time transfer function, using the Matlab *tf* function. The resulting transfer function has the general form:

$$\frac{y}{u}=\frac{B\left(z^{-1}\right)}{A\left(z^{-1}\right)}=\frac{b_1z^{-1}+\cdots+b_mz^{-m}}{1+a_1z^{-1}+\cdots+a_nz^{-n}} \tag{15}$$

This represents the plant (*i.e.*, the converter) as seen in the general block diagram of Figure 2.

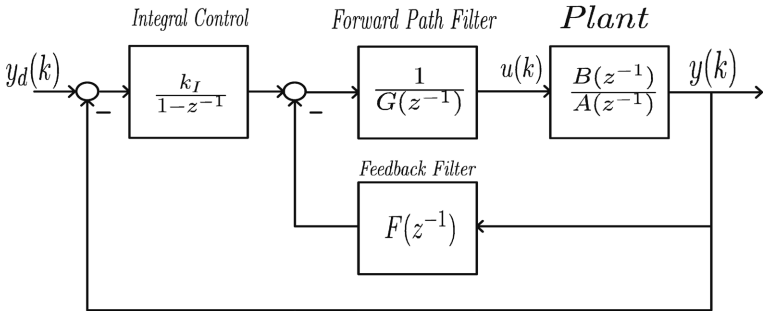

Figure 2. Proportional-Integral-Plus controller block diagram [16].

The remaining blocks represent the compensation, which has been termed Proportional Integral Plus (PIP) [16]. In order to reduce steady state errors, integral control has been introduced. The plant, together with integral control, (which adds an extra state variable, z), are now represented in a non-minimal state space form. The NMSS state vector is given by:

$$\begin{aligned} x(k)^{\mathrm{T}}=\big[&y(k)\quad y(k-1)\quad\cdots\quad y(k-n+1)\quad u(k-1)\quad\cdots \\ &u(k-2)\quad\cdots\quad u(k-m+1)\quad z(k)\big] \end{aligned} \tag{16}$$

The dimension of the state vector is $n+m$, *i.e.*, the sum of orders of the denominator and numerator. Recall that a minimal state space representation with an added integrator would have state dimension $n+1$, *i.e.*, the order of the denominator plus one.

The resulting NMSS state space representation of the plant augmented with an integrator is given by [16]:

$$x(k)=Fx(k-1)+gu(k-1)+dy_d(k) \tag{17}$$

$$y(k)=hx(k) \tag{18}$$

where

$$F=\begin{bmatrix} -a_1 & \cdots & -a_{n-1} & -a_n & b_2 & \cdots & b_{m-1} & b_m & 0 \\ 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & 0 & 0 \\ a_1 & \cdots & a_{n-1} & a_n & -b_2 & \cdots & -b_{m-1} & -b_m & 1 \end{bmatrix}$$

$$g=\begin{bmatrix} b_1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & -b_1 \end{bmatrix}^{\mathrm{T}}$$

$$d=\begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}^{\mathrm{T}}$$

$$h=\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

3. Optimal Controller Design

In this section, an optimal proportional-integral-plus (PIP) controller will be designed. The optimal PIP control presented relies on the aforementioned (NMSS) representation. As will be seen, the design process is relatively simple and very effective. The control law takes the form [16] of full state feedback:

$$u(k)=-k^{\mathrm{T}}x(k) \tag{19}$$

where

$$k^{\mathrm{T}}=\begin{bmatrix} f_0 & f_1 & \cdots & f_{n-1} & g_1 & \cdots & g_{m-1} & -k_I \end{bmatrix} \tag{20}$$

and $x(k)$ is given by (16).

The goal of optimal design is to minimize a quadratic cost function subject to the constraints of the system state equations. The cost function is given by [16]:

$$J=\sum_{k=0}^{\infty} x(k)^{\mathrm{T}} Qx(k)+r\left(u(k)\right)^2$$

where $x(k)$ is the non-minimum state vector, $u(k)$ is the control input, Q is the positive definite weighting matrix, and r is the positive weighting scalar. We define the matrix Q as the following diagonal matrix:

$$\begin{aligned} Q&=\operatorname{diag}\begin{pmatrix} q_1 & q_2 & \cdots & q_n & q_{n+1} & \cdots & q_{n+m-1} & q_{n+m} \end{pmatrix} \\ &=\operatorname{diag}\begin{pmatrix} q_y & q_y & \cdots & q_y & q_u & \cdots & q_u & q_e \end{pmatrix} \end{aligned}$$

In addition, the positive weighting scalar r will be set equal to q_u ([16], p. 111), which is defined next. By defining the following equations only three tuning parameters will be left to configure:

$$q_y=\frac{W_y}{n};\quad q_u=\frac{W_u}{m};\quad q_e=W_e$$

For many practical cases the three tuning variables (W_y, W_u, W_e) can be simply set to unity. It is necessary to use an Algebraic Riccati Equation to extract the corresponding optimal controller gains. The method proposed by [16] calls

for solving the following equations recursively:

$$k^{\mathrm{T}}(i)=\left(r+g^{\mathrm{T}}P(i+1)g\right)^{-1}g^{\mathrm{T}}P(i+1)F$$

where

$$P(i)=Q+F^{\mathrm{T}}P(i+1)F-F^{\mathrm{T}}P(i+1)gk^{\mathrm{T}}(i)$$

and

$$P(N)=0,\quad k(N)=0$$

Implementing the recursive equations in MATLAB can be done as shown in Figure 3.

```
% the following script recursively solves for the
% optimal gain values

% Inputs: F and g, the NMSS matrices of PIP system
%         n and m, the orders of the den and num polys
% Outputs: fpip, gpip, ki
%          fpip - coeff's of F(z^-1)
%          gpip - coeff's of G(z^-1)
%          ki - integral gain
%=======================================================

% Set up the weights
Wy = 1;                         % default weights
Wu = 1;
We = 1;

qy = Wy/n;
qu = Wu/m;
qe = We;

qy_n = qy*ones(1, n);
qu_m = qu*ones(1, m-1);
Q = diag([qy_n, qu_m, qe]);     % LQR weighting matrix Q
r = qu;                         % LQR weight r

%=======================================================
% Recursively solve for optimal gains k

k=zeros(n+m, 1);               % initialize k and P
P=zeros(n+m);

for ff = 1:200                  % maximum number of ...
                                % iterations is 200

    k0 = k;                     % save previous k value
    k = ((r+g'*P*g)\g'*P*F)';   % update k
    P = Q+F'*P*F-F'*P*g*k';     % update P

    if ff > 5                   % allow 5 iterations ...
                                % before checking ...
                                % if accuracy is met

        if norm(k-k0, Inf)<1e-4 % exit loop when ...
                                % accuracy is met
            break
        end

    end

end

fpip = k(1:n)';                 % coeff's of F(z^-1)
gpip = [1;  k(n+1:end-1)]';     % coeff's of G(z^-1)
ki = -k(end);                   % ki, integral gain
```

Figure 3. MATLAB script to solve for optimal controller gains.

For the Buck converter, with parameters, $C_f = 100$ μF, $L_f = 300$ μH, $R = 10$ Ω, as used in the constructed prototype discussed in the next section, the optimal controller gains lead to the following results:

$$F\left(z^{-1}\right) = 22 - 17.3z^{-1}$$

$$G\left(z^{-1}\right) = 1 + 0.263z^{-1}$$

$$k_I = 0.736$$

A straightforward block diagram reduction of the system shown in Figure 2 leads to that shown in Figure 4. We see that the equivalent compensator transfer function $G_c\left(z^{-1}\right)$ is given by:

$$G_c\left(z^{-1}\right) = \frac{k_I A\left(z^{-1}\right)}{\left[G\left(z^{-1}\right)A\left(z^{-1}\right) + F\left(z^{-1}\right)B\left(z^{-1}\right)\right]\left(1 - z^{-1}\right)} \tag{21}$$

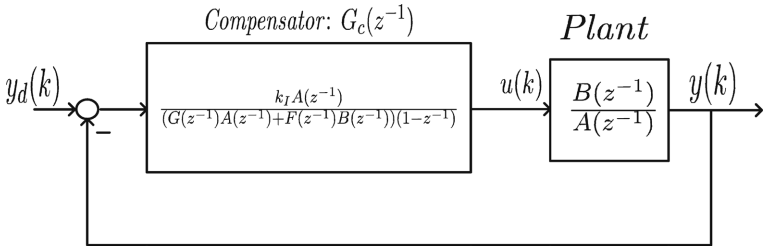

Figure 4. Equivalent system block diagram explicitly showing the compensator and plant combination.

The product of the compensator transfer function with that of the plant, gives the loop gain of the system. The frequency response of the loop gain is shown in Figure 5, which verifies the good stability margins.

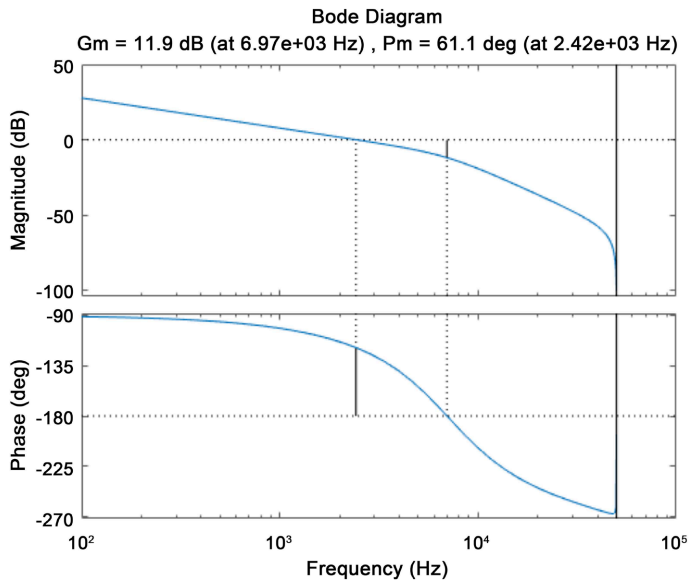

Figure 5. Bode plot of loop gain, showing a 61˚ phase margin.

4. Controller Hardware Overview

A digital microcontroller, specifically the C2000 development kit for the Texas Instruments F28069M microcontroller, was used to implement the PIP controller in hardware. An overall block diagram of the system is shown in Figure 6. The important controller modules for the implementation are the pulse width modulator (PWM) and analog to digital converter (ADC). Not explicitly shown is the general purpose input/output (GPIO) which assigns IO pins to the internal modules. The converter requires a PWM signal, which is directly available from the microcontroller.

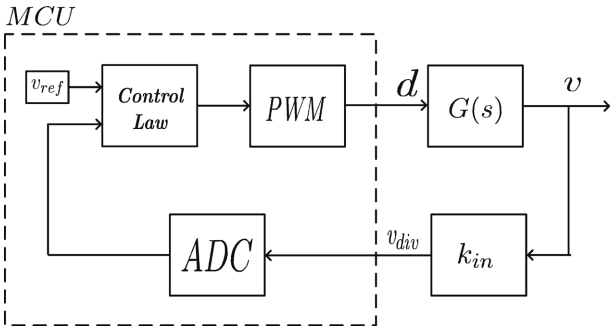

Figure 6. Block diagram of the closed loop system, showing the interconnection of the microcontroller unit (MCU) and the plant (*i.e.* the Buck converter and output voltage resistive divider).

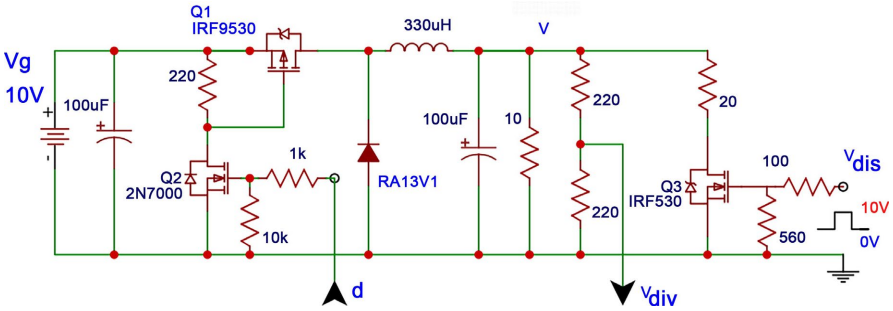

Figure 7. Buck converter schematic.

The PWM outputs a high voltage of 3.3 V or a low voltage of 0 V with a variable duty ratio. The ADC uses a resistor capacitor circuit to sample and hold a voltage ([18], pp. 520-525). The output voltage of the plant is scaled down to the ADC's input range of 0 to 3.3 V with a resistive voltage divider, as seen in the Buck converter (the plant) schematic in Figure 7. The ADC then converts that value to a digital number that can be read from a register in memory. The controller program then scales that value appropriately to get a precise and accurate reading of the output voltage. If a GPIO pin is in output mode, the voltage can be set to either 3.3 V or 0 V. In input mode, the voltage on the GPIO pin can be routed to an internal module, such as the ADC. The GPIO module can be configured to specify the pins to which the ADC and PWM have access.

The PWM works by using an internal counter, a period register and a counter compare register ([18], p. 244). The counter increments at each pulse of the system clock. When the counter is zero, the PWM asserts a digital output pin, *i.e.*, sets the output pin to logical true of 3.3 V. When the counter equals the value in the PWM's counter compare register, the PWM deasserts the same pin, *i.e.*, sets the output pin to a logical false of 0 V. When the counter equals the value in the PWM's period register, the counter resets to zero. The PWM also has a synchronization bit that, when written to, will reset the counter to zero. This is the most basic operation of the PWM module. There are more sophisticated counting options to synchronize with external or internal signals and ways to change the behavior of what is done at different points in the count. The basic mode of operation is sufficient for the PIP controller.

The PIP program controls the output by setting the counter compare register to a value that will produce the desired duty cycle. The new counter compare value is written to a shadow register which serves as a temporary holding place for the active register ([18], p. 244). Then, a software PWM resync is performed to load the new counter compare value from the shadow register and reset the counter to zero. There may be risks that this will result in inconsistent PWM periods. However, the processing should be precisely scheduled to ensure that this operation is done at a 100 kHz frequency, which is the switching frequency of the Buck converter. This requirement for the constant switching frequency was confirmed by measurement with the oscilloscope that showed that the PWM period was always 10 μs.

As previously mentioned, the ADC uses a resistor capacitor circuit to sample and hold a voltage it receives on a designated input pin. The ADC hardware then uses a set of comparators to convert that value to a digital number that can be read from a register in memory.

There are different modes of operation to control the timing of the ADC operation. For the purpose of this PIP program the ADC operates in non-continuous mode. This means a start of conversion (SOC) signal must be sent to the ADC to initiate a read of the voltage on the input pin. An adequate sample and hold time must be set such that the ADC charges its capacitor sufficiently ([18], p. 521). Once the timer indicates the sample and hold period is over, the ADC produces a digital read of the voltage. This value is in terms of "ticks." The F28069M controller has a 12-bit ADC register, therefore 0 ticks correspond to 0 V and 4095 ($=2^{12}-1$) ticks corresponds to 3.3 V. After the ADC finishes the read, it will update its register with a new value and produce an end of conversion (EOC) signal. This EOC signal is useful for minimizing delay between receiving a new digital input and processing it.

5. Program Architecture

The PIP program, seen as the control law in **Figure 6**, collects input from the ADC and produces output with the PWM. A common software element, the Interrupt

Service Routine (ISR), is introduced to ensure that the control law is executed periodically at a frequency of 100 kHz.

The architecture of the PIP program relies upon the ISR. This is for two reasons. The first is that the ISR can trigger the ADC's EOC signal. This will help ensure minimum delay between sampling the input and executing the difference equation, and ultimately producing a PWM output. The second is that the SOC is routed to a timer such that the ISR runs at the sampling frequency. This will ensure that the coefficients in the PIP's difference equation will be valid for the sample period for which they were derived.

Therefore, the program can be split into two main components, the main function and the ISR function. These two main components are discussed in the next two subsections.

5.1. Main Function

The purpose of the main function is to initialize the PIP program. This entails configuring the hardware modules previously described and mapping the ISR to the appropriate signal. The PIP program also includes libraries supplied by the F28069M software development kit. These libraries provide functions to configure clocks, phase locked loops and the default General Purpose Input Output (GPIO) pin map. Calling these functions first is a necessity for our program.

The main function also handles controller configuration specific to the PIP program. The first task is to configure the Peripheral Interrupt Expansion (PIE) ([18], p. 166). The PIE module routes external events to the processor. When an event is seen by the processor, it will switch from executing instructions from the main function to executing a function as mapped. Therefore, a function pointer for the PIP program's ISR is written in the PIE's table for ADC interrupt 1 so that when an ADC interrupt 1 occurs, the ISR function is called. ADC interrupt 1 will be configured as part of the ADC. After this configuration, the interrupt is enabled.

The ADC for the PIP program must also be initialized, calibrated and configured. Initialization and calibration are performed by calls to functions in the development kit libraries. The PIP program configures the ADC to operate in non-continuous mode. The PIP program selects EPWM 1 as the source for the SOC. The SOC event selection register is set to 4, which on the F28069M corresponds to the EPWM 1 counter compare equal signal. EPWM 1 is used to control the ADC. However, an internal CPU timer could also be used. The sample and hold window is set to 6 clock cycles which is the lowest value allowed. The input channel is set to ADCINA4 which is GPIO pin 69 on the development kit. The ADC control's interrupt pulse generation register is set to 1. On the F28069M this configuration means that the PIE's ADC interrupt will be triggered by the EOC signal.

After configuring the ADC and PIE as specific to the PIP program, main configures the PWM modules. Using the *InitEPwmGpio* function provided by the library, the GPIO mux is configured to route PWM signals to the GPIO. This way, the PWM can be attached to external hardware and viewed on an oscilloscope.

Both PWM 1 and 2 are set to a period of 0x1c4 clock cycles which corresponds to 10 μs. This was validated on the oscilloscope. Both PWM modules are configured using the simple count-up mode described previously. This ends the configuration procedure.

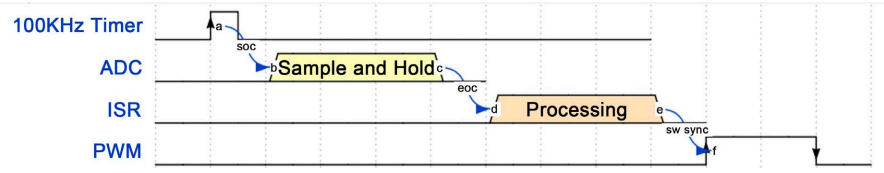

Figure 8. Timing sequence of ADC to PWM signal.

After enabling all interrupts, the main function will continuously perform a "NO-OP" inside a while loop. Given this configuration, Figure 8 illustrates the timing sequence between the components. First, the timer (PWM 1 in this program) triggers the ADC, which triggers the ISR, which then triggers the PWM 2. We observed an execution time of approximately 1 μs from the SOC to the start of PWM 2's period with the new duty cycle.

5.2. ISR Function

As previously mentioned, the PIP program contains an ISR that executes the difference equation. The ISR was mapped to launch each time an EOC signal is raised. The ISR will then read the value of the ADC register. It will scale this value twice. First to convert from an unsigned integer value in ticks to a floating-point value of Volts. After that, it will perform another scaling operation to counteract the voltage divider. This final scaled value is the value of $y(k)$ in the PIP controller from Figure 9. The topology of the ISR is presented in the following figure and described below.

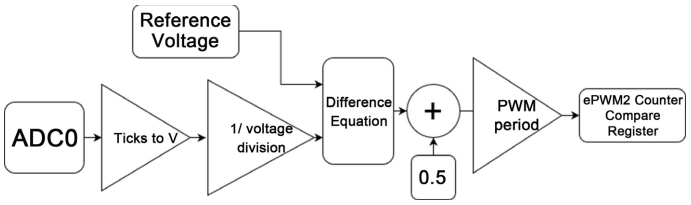

Figure 9. Layout of ISR function.

The ISR then calls a function dedicated to implementing the difference equation. The inputs are the $y(k)$ value read from the ADC and a $y_d(k)$ value. As $y_d(k)$ is a constant, it is hardcoded inside the PIP program to 5 V which is the desired output voltage.

The difference equation function accesses two global arrays of floating-point numbers. This way, it can store previous outputs and inputs. It also has access to the constant global floating-point numbers which are the coefficients derived from the controller design script in MATLAB. The difference equation is of the form:

$$output = u[1] + k \cdot (yd - y[0]) - f[0] \cdot (y[0] - y[1]) - f[1] \cdot (y[1] - y[2]) - g[1] \cdot (u[1] - u[2])$$

where u is the buffer of previous outputs and y is the buffer of previous inputs. The f, g and k coefficients correspond to the same values from Equation (20). For the input and output buffers, the index indicates the previous cycle, e.g. $y[0]$ corresponds to the current input and $u[1]$ corresponds to the output from 1 sample period ago. The difference equation function performs advances these buffers by updating them with the latest input and the new output it has generated and returns to the ISR.

After receiving the new output of the difference equation, the ISR will add 0.5 to it. This corresponds to the quiescent state of the duty ratio. To produce a duty cycle with the same ratio, the difference equation output (a value between 0 and 1) must be multiplied by the period of PWM 2. That value is then written to the counter compare shadow register of PWM 2. The ISR then sends a software synchronization pulse to the PWM module to load the value from the shadow register and begin a new period. This completes the ISR function.

6. Hardware Prototype

To complete the implementation and form the hardware prototype, the microcontroller is attached to the Buck converter as pictured in **Figure 6**. The divided output voltage of the plant, v_{div}, as shown in **Figure 7**, is connected to the ADCINA4 pin, which is the input for the ADC used by the controller implementation. The controller's EPWM 2 output pin is connected to point d in Figure 7. The signal d drives the power mosfet of the Buck converter. The converter takes a nominal 10 V input and steps it down to 5 V. The quiescent duty ratio is thus 50%.

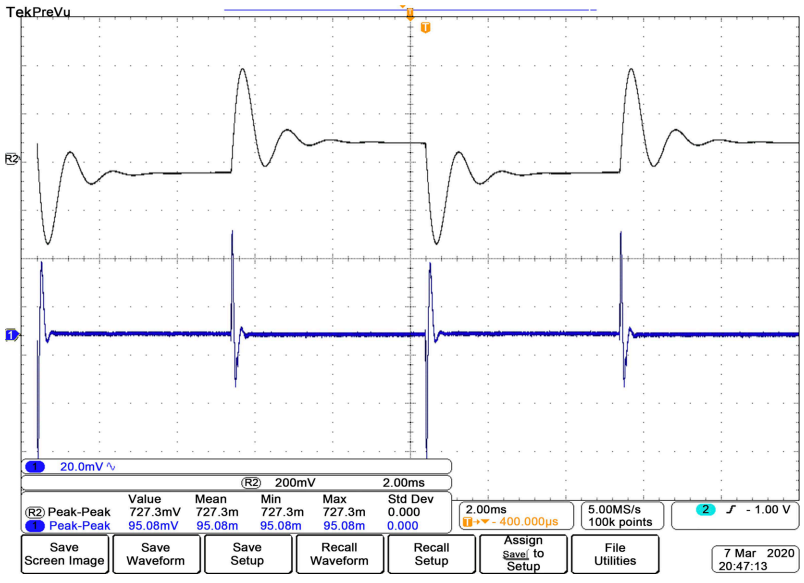

Figure 10. Output voltage response to a 50 percent step change in the output load current. Top waveform: open-loop response, vertical scale: 200 mV/div., bottom waveform: closed-loop response, vertical scale: 20 mV/div.

To test the efficacy of the controller, a step load disturbance signal is instigated by periodically switching an extra load across the output. This is done by turning mosfet $Q3$ on and off which adds and removes an additional shunt 20 Ω resistor. The results of this test are shown in Figure 10. The top trace shows the open loop response and the bottom trace is the closed loop response of the output voltage v. The open loop response shows a large transient of 727 mV peak to peak and a non-zero steady state difference of 100 mV between switched states. The closed loop implementation greatly reduces the transient to 95 mV peak to peak and eliminates steady state difference completely. This zero steady state error is achieved by having integral control in the loop as implemented by the PIP controller. Also evident is that the transient settling time is greatly reduced in the closed loop configuration.

7. Conclusions

The use of a non-minimal state space representation permits the use of output and input signals as the states thus obviating the need for a state estimator. Absent an estimator there is no need for procedures such as loop transfer recovery (LTR) needed to optimize system response.

With NMSS, optimal control may more straightforwardly be implemented. In this paper we have shown its application in the design of a Buck converter controller. The resulting discrete control law can be quickly implemented. We illustrated this with a Texas Instruments F28069M microcontroller. As demonstrated by the implementation process, a microcontroller is suitable for quickly prototyping a discrete controller since it is elementary to store the discrete set of inputs and outputs as required by NMSS control. To aid in future design, some fundamental issues concerning microcontroller implementation have been discussed at length. The hardware prototype demonstrates the efficacy of this design approach through excellent disturbance rejection performance.

Acknowledgements

Samuel Jacobs, a former student in the Electrical and Computer Engineering Department at Portland State University is acknowledged for his assistance in microcontroller implementation and providing the oscillogram and a number of the figures appearing in this paper.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Verma, S., Singh, S.K. and Rao, A.G. (2013) Overview of Control Techniques for DC-DC Converters. *Research Journal of Engineering Sciences*, **2**, 18-21.

[2] Wan, K., Liao, J. and Ferdowsi, M. (2007) Control Methods in DC-DC Power Conversion—A Comparative Study. 2007 *IEEE Power Electronics Specialists Conference*,

Orlando, 17-21 June 2007, 921-926. https://doi.org/10.1109/pesc.2007.4342111

[3] Lešo, M., Žilková, J., Biroš, M. and Talian, P. (2018) Survey of Control Methods for DC-DC Converters. *Acta Electrotechnica et Informatica*, **18**, 41-46. https://doi.org/10.15546/aeei-2018-0024

[4] Liu, Y.-F., Meyer, E. and Liu, X.D. (2009) Recent Developments in Digital Control Strategies for DC/DC Switching Power Converters. *IEEE Transactions on Power Electronics*, **24**, 2567-2577. https://doi.org/10.1109/tpel.2009.2030809

[5] Oliva, A.R., Ang, S.S. and Bortolotto, G.E. (2006) Digital Control of a Voltage-Mode Synchronous Buck Converter. *IEEE Transactions on Power Electronics*, **21**, 157-163. https://doi.org/10.1109/tpel.2005.861193

[6] Chattopadhyay, S. and Das, S. (2006) A Digital Current-Mode Control Technique for DC-DC Converters. *IEEE Transactions on Power Electronics*, **21**, 1718-1726. https://doi.org/10.1109/tpel.2006.882929

[7] Peng, H. and Maksimovic, D. (2005) Digital Current-Mode Controller for DC-DC Converters. 20*th Annual IEEE Applied Power Electronics Conference and Exposition*, Vol. 2, 899-905. https://doi.org/10.1109/apec.2005.1453091

[8] Seshagiri, S., Block, E., Larrea, I. and Soares, L. (2016) Optimal PID Design for Voltage Mode Control of DC-DC Buck Converters. 2016 *Indian Control Conference* (*ICC*), [Location], [Date], 99-104. https://doi.org/10.1109/indiancc.2016.7441112

[9] Ounnas, D., Guiza, D., Soufi, Y., Dhaouadi, R. and Bouden, A. (2019) Design and Implementation of a Digital PID Controller for DC-DC Buck Converter. 2019 1st International Conference on Sustainable Renewable Energy Systems and Applications (ICSRESA), Hyderabad, 4-6 January 2016, 1-4. https://doi.org/10.1109/icsresa49121.2019.9182430

[10] Nabeshima, T., Sato, T., Yoshida, S., Chiba, S. and Onda, K. (2004) Analysis and Design Considerations of a Buck Converter with a Hysteretic PWM Controller. 2004 *IEEE* 35*th Annual Power Electronics Specialists Conference*, Vol. 2, 1711-1716. https://doi.org/10.1109/pesc.2004.1355684

[11] Keskar, N. and Rincon-Mora, G.A. (2004) Self-Stabilizing, Integrated, Hysteretic Boost DC-DC Converter. 30*th Annual Conference of IEEE Industrial Electronics Society*, Vol. 1, 586-591.

[12] Huerta-Moro, S., Martínez-Fuentes, O., Gonzalez-Diaz, V.R. and Tlelo-Cuautle, E. (2023) On the Sliding Mode Control Applied to a DC-DC Buck Converter. *Technologies*, **11**, Article No. 33. https://doi.org/10.3390/technologies11020033

[13] Ahmed, M., Kuisma, M., Tolsa, K. and Silventoinen, P. (2003) Implementing Sliding Mode Control for Buck Converter. *IEEE* 34*th Annual Conference on Power Electronics Specialist*, Vol. 2, 634-637. https://doi.org/10.1109/pesc.2003.1218129

[14] Leng, Z. and Liu, Q. (2017) A Simple Model Predictive Control for Buck Converter Operating in CCM. 2017 *IEEE International Symposium on Predictive Control of Electrical Drives and Power Electronics* (*PRECEDE*), Pilsen, 4-6 September 2017, 19-24. https://doi.org/10.1109/precede.2017.8071262

[15] Bhat, N.D., Kanse, D.B., Patil, S.D. and Pawar, S.D. (2020) DC/DC Buck Converter Using Fuzzy Logic Controller. 2020 5*th International Conference on Communication and Electronics Systems* (*ICCES*), Coimbatore, 10-12 June 2020, 182-187. https://doi.org/10.1109/icces48766.2020.9138084

[16] Taylor, C.J., Young, P.C. and Chotai, A. (2013) True Digital Control: Statistical Modeling and Non-Minimal State Space Design. Wiley. https://doi.org/10.1002/9781118535523

[17] Middlebrook, R.D. and Cuk, S. (1976) A General Unified Approach to Modelling Switching-Converter Power Stages. 1976 *IEEE Power Electronics Specialists Conference*, Cleveland, 8-10 June 1976, 18-34. https://doi.org/10.1109/pesc.1976.7072895

[18] (2011) TMS320x2806x Microcontrollers: Technical Reference Manual. Texas Instruments Incorporated. http://www.ti.com/lit/ug/spruh18h/spruh18h.pdf