Comparison of Optimization Algorithms for Boost Converter Controller Design

Kevin C. Fronczak

***Abstract*—Switched DC-DC Boost Converter controllers can take on many topologies and the design depends on the performance of the power conversion stage itself. To aid in the design process, the use of Genetic Algorithms and Particle Swarm Optimization algorithms is explored here. Each algorithm is implemented and the performance compared in order to provide a thorough analysis of what optimization algorithms explore the controller design solution space more efficiently. The algorithms are operated on a Voltage-Mode-Controlled Boost Converter operating in both CCM and DCM. The Genetic Algorithm utilizes a technique known as the 'Queen-Bee' method while the two Particle Swarm Optimization algorithms compared utilizes a constriction factor and a chaotic inertial weight. The Genetic Algorithm was able to produce a solution within 0.2% of the ideal solution while the best PSO implementation only came within 74% with the same operating conditions.**

I. Introduction

SWITCHED DC-DC converters have many frequency response characteristics that must be optimized in order to produce the best product and eliminate stability concerns. The design of the controller is critical as it determines the steady-state accuracy, response time, and stability margins (both gain and phase). Unfortunately, amplifier design is not trivial as improving one performance metric will, typically, decrease another. For example, in order to increase the phase margin of a system, one must typically lower the gain in order to compensate. While this would improve stability, it would also decrease steady-state accuracy.

Due to this complexity, switched converter controller design is well suited to solving via optimization algorithms. Electronics design via optimization algorithms is a well-established field of research and shows the possibility of providing optimal solutions for high-complexity design [17], [18], [23], [25], [30]–[35].

There are two main areas in which an optimization algorithm could be useful within the structure of a Switched DC-DC Boost Converter such as the power conversion stage [18], [25] and the controller stage [33]. Here, the controller stage is of more interest due to its direct impact on the frequency response of the system. The design of the controller is one of the most important tasks as it is used to eliminate steady-state error at the output of the converter and is used to guarantee that the converter is stable under all operating conditions.

The optimization algorithms compared here are that of Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO). For the Genetic Algorithm implementation, the 'Queen-Bee' method is used wherein the algorithm is modeled off of the interaction of bees within a hive (abbreviated as QBGA) [22], [32], [35]. For the Particle Swarm Optimization implementation, three algorithms are compared: Constriction (PSOC), Chaotic Descending Inertia Weight (CDIW), and Chaotic Random Intertia Weight (CRIW) [20], [26], [27], [29].

Both QBGA and all three PSO algorithms attempted to optimize the transfer function of a controller by iterating over the parameters of the controller transfer function for the Boost Converter in both CCM and DCM. Various fitness function implementations were compared including a Normalized Linear Function (NLF), Normalized Parabolic Function (NPF), and both NLF and NPF with a penalty function used to aid in convergence [15], [16]. Preliminary work was performed previously in [4].

II. Boost Converter Control

The block diagram for a typical boost converter is shown in Fig. 1 while a circuit implementation utilizing Voltage-Mode-Control (VMC) is shown in Fig. 2. There are two modes of operation for a boost converter, known as Continuous-Conduction Mode (CCM) and Discontinuous Conduction Mode (DCM). Each mode is characterized by observing the current through the inductor over one switching cycle. If the current goes to zero before one switching cycle ends, the converter is said to be operating in DCM. On the other-hand, if the current does not drop to zero withing one switching cycle it is said to be operating in CCM. Distinguishing the two is important as each mode has implications on the small-

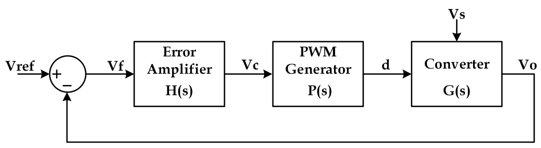

Fig. 1: Boost Converter Block Diagram

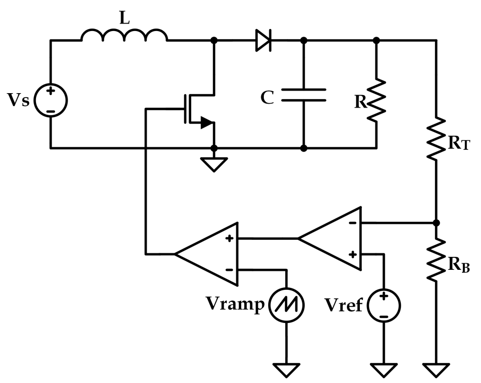

Fig. 2: Voltage-Mode-Controlled Boost Converter Circuit Architecture

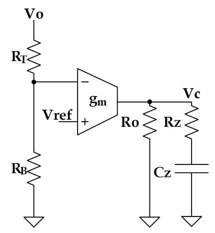

Fig. 3: Lag Compensator Circuit

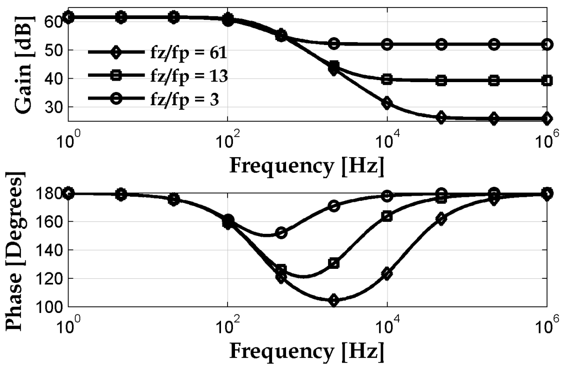

Fig. 4: Bode Plot for Lag Compensator Circuit with $R_O = 6\,k\Omega, C_Z = 100\,nF, g_m = -1\,S, R_T = 2\,M\Omega, R_B = 500\,k\Omega$ and R_Z swept with values of $100\,\Omega$, $500\,\Omega$, and $3\,k\Omega$.

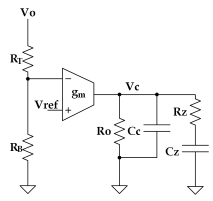

Fig. 5: Lag Compensator Circuit Plus Pole

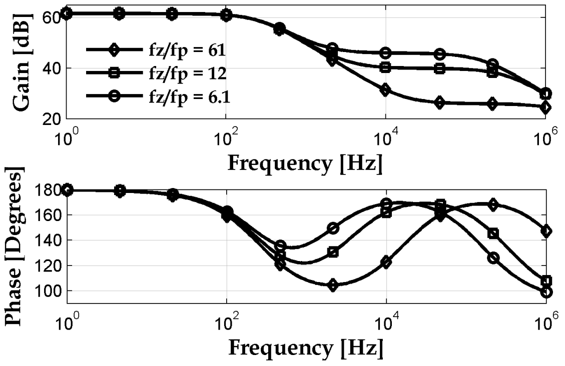

Fig. 6: Bode Plot for Lag Compensator Circuit Plus Pole with $R_O = 6\,k\Omega$, $C_Z = 100\,nF$, $C_C = 1\,nF$, $g_m = -1\,S$, $R_T = 2\,M\Omega$, $R_B = 500\,k\Omega$ and R_Z swept with values of $100\,\Omega$, $500\,\Omega$, and $1\,k\Omega$.

signal performance of the circuit as a whole: the VMC-CCM Converter exhibits LC resonance while the VMC-DCM Converter does not [1]–[6]. This makes controller designs much easier to implement for VMC-DCM Converters rather than VMC-CCM Converters.

There are many different error amplifier architectures that can be used for the controller [5], [7], [8], but only three are explained here: the Lag, Lag-Plus-Pole, and PID. Each controller explored is implemented with an OTA due to their more ideal characteristics at an integrated circuit level [8]–[12].

Note that for all subsequent controller transfer function equations in this section,

$$K = g_m R_O \frac{R_B}{R_T + R_B} \tag{1}$$

A. Lag Controller

The Lag Compensator is given in Figure 3 and has a transfer function given by (2). The feature of this type of compensator is that it provides a phase “lag” at a given peak frequency. Note that for (2) to be true, $R_O \gg R_Z$.

$$\frac{V_c}{V_o} = K\left[\frac{sR_ZC_Z + 1}{sR_OC_Z + 1}\right] \tag{2}$$

$$f_p = \frac{1}{2\pi R_O C_Z} \tag{3}$$

$$f_z = \frac{1}{2\pi R_Z C_Z} \tag{4}$$

The benefit of using a lag compensator, also known as a proportional-integral (PI) controller, is that it can be used to boost the gain of a system without affecting the behavior near the $0\,dB$ crossing point. As such, it’s ideal for converters that already have a good phase margin, but just need the DC gain boost in order to eliminate steady-state error (such as the VMC-DCM Converter).

B. Lag Controller Plus Pole

By adding a capacitor, C_C, in parallel with the series RC string, a high-frequency pole can be added to the system, as shown in Figure 6. In terms of equations, this circuit can be shown to have a transfer function equivalent to (5) where $R_O \gg R_Z$ and $C_Z \gg C_C$. Note that the only difference between this compensator and the one shown previously in Figure 4 is the addition of a high frequency pole.

$$\frac{V_c}{V_o} = K\frac{sR_ZC_Z + 1}{s^2R_OR_ZC_CC_Z + sR_OC_Z + 1} \tag{5}$$

C. PID Controller

The PID (Proportional-Integral-Derivative) controller, as shown in Figure 7, is useful as it provides both phase lag and lead in order to maximize phase and gain compensation. This makes it ideal for VMC-CCM Boost Converters as it provides a large DC-gain as well as the ability to boost the phase such that the phase-margin is of an acceptable value. The transfer function for this controller can be found below in (6). Again, it is assumed that $R_O \gg R_Z$.

$$\frac{V_c}{V_o} = \frac{K}{sR_O(C_Z + C_C)}\frac{(sR_TC_1 + 1)(sR_ZC_Z + 1)}{(s\frac{R_ZC_ZC_C}{C_Z+C_C} + 1)(s(R_T\|R_B)C_1 + 1)} \tag{6}$$

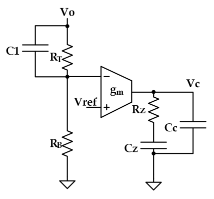

Fig. 7: Lag-Lead Compensator Circuit

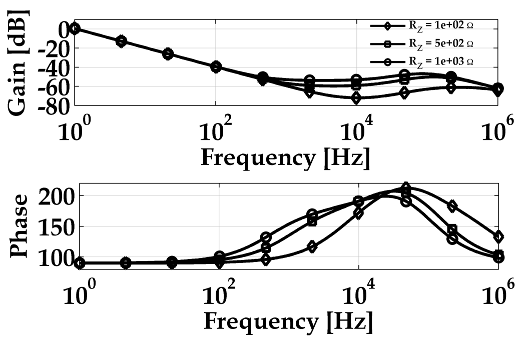

Fig. 8: Bode Plot for Lag-Lead Compensator Circuit with $C_1 = 0.8\,pF$, $C_C = 2\,nF$, $g_m = -10\,\mu S$, $R_B = 500\,k\Omega$, $R_T = 2\,M\Omega$, $C_Z = 300\,nF$, and R_Z swept with values of $100\,\Omega$, $500\,\Omega$, and $1\,k\Omega$.

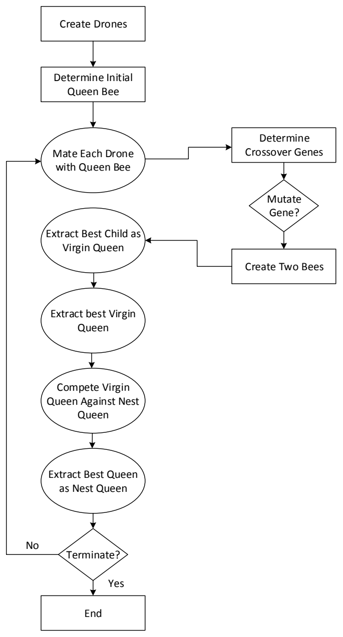
Create Drones
Determine Initial Queen Bee
Mate Each Drone with Queen Bee
Determine Crossover Genes
Mutate Gene?
Create Two Bees
Extract Best Child as Virgin Queen
Extract best Virgin Queen
Compete Virgin Queen Against Nest Queen
Extract Best Queen as Nest Queen
No
Terminate?
Yes
End

Fig. 9: Block Diagram for a Queen-Bee Genetic Algorithm for Optimization of Switched-Converter Controller Transfer Function.

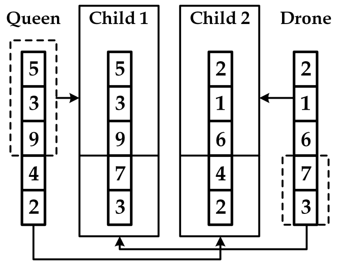

Fig. 10: Gene Crossover Diagram for QBGA.

III. Optimization Algorithms

The idea behind using optimization algorithms to solve the controller design problem is to remove some of the burden off of a circuit designer. These algorithms are, typically, much better at exploring the design solution space than a designer would be and, as such, can convergence on an acceptable solution more quickly. Even if the design is not an *optimal* design, it provides a circuit designer with a good starting point and still serves to minimize the required time spent on a circuit block.

A. Queen-Bee GA

There are many different GA implementations, but the one analyzed here is known as the 'Queen-Bee' method [22], [32]. Fig. 9 shows the flow of a Queen-Bee-type GA (abbreviated as QBGA).

The QBGA is an algorithm based around the interactions of bees in a hive. Essentially, there exists a single queen with which all other bees, known as drones, mate. Occasionally a female bee is produced that ousts the current queen and becomes the new queen.

To implement this in software, drones are created with completely randomized genes and the algorithm proceeds as follows:

1) Initialization: Drones compete and best is selected as nest queen.
2) Drones mate with queen, produce two offspring (gene crossover example in Fig. 10).
3) Offspring, known as virgin queens, compete and best survives.
4) Every virgin queen competes and best competes with current queen; whichever survives becomes new nest queen.
5) Randomize drones, repeat.

During gene crossover, however, it is not sufficient to *only* crossover the genes as there must be some probability of gene mutation in order to guarantee variability in the gene pool to avoid converging on local optima. To do so, most algorithms implement a constant that determines the gene mutation probability [22], [23], [32].

In an attempt to speed up convergence, a variable mutation probability was used in this implementation based on the age of the nest queen as an alternative to simply choosing a constant. As the nest queen moves from generation to generation, it is likely that the QBGA has either converged on the best solution or it could be stuck on a local optima. By setting the mutation probability as a variable based on how many generations the nest queen has existed for, more and more variability will be introduced into the gene pool. The goal here is to attempt to push the algorithm out of any potential locally optimal points. If the program has, in fact, found a

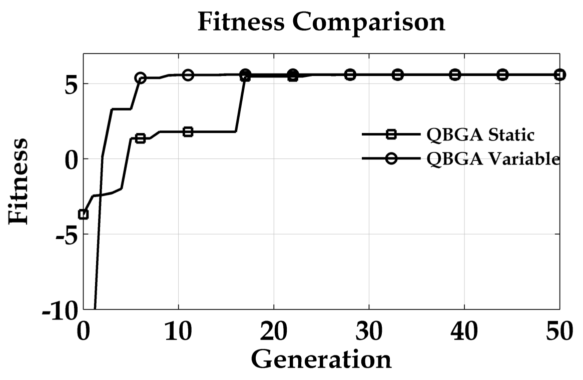

Fig. 11: Comparison of QBGA Convergence with a Varied and Constant Rate of Mutation for a nest size of 30 and over 20 iterations.

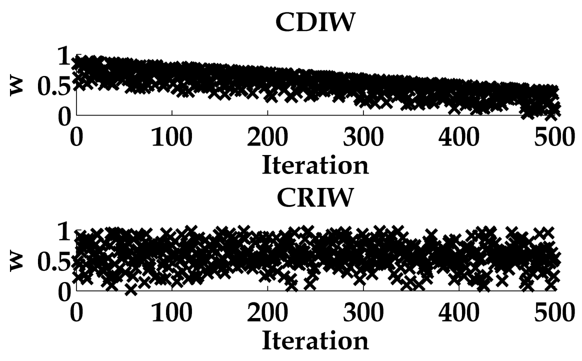

Fig. 12: Comparison of w for CDIW and CRIW over 500 iterations with $w_1 = 0.9$ and $w_2 = 0.4$.

global optimum, then an increased mutation rate should have no effect on the solution. The pseudo-code for such a mutation algorithm is described below:

Algorithm 1: Variable QBGA Mutation Based on Queen Age

```
for gene in crossover do
    if rand() $\geq$ C*queenAge then
        gene = mutate(gene);
    end
    crossover[i] = gene;
end
```

By setting C to some fractional value, it will eventually converge to have a 100% mutation rate when $\frac{1}{C} = queenAge$. However, this value *must not* be set too high as otherwise the GA will mutate too quickly and have difficulty converging on a solution. Setting C to be between 0.05 and 0.1 yielded the best results.

A comparison of convergence rates for the QBGA with and without the variable mutation rate is shown in Figure 11. Here it takes eleven generations for the varied mutation rate implementation to converge while it takes *twenty-five* generations for the constant rate of mutation implementation to converge to the same fitness value. This indicates that the variable rate of mutation does, indeed, aid in convergence speed as expected.

B. Particle Swarm Optimization

Another type of optimization algorithm that is useful for high-complexity circuit design is PSO. Using PSO as a circuit design optimization tool has been explored previously in [33], [34].

This method is different from that of GAs in that there are a set number of particles that have an associated position and velocity in N-dimensional space. Each particle's position is evaluated and each keeps track of its own best position. All of the best positions for each particle are then evaluated and the best global position is stored. Each particle has knowledge of its current position, current velocity, best local position, and best swarm global position. As the swarm begins to move, the velocity of the particles change based on those parameters in order to converge on a solution.

1) Chaotic Inertial Weight: The idea of adding some random behavior to the inertial weight of a particle's velocity has been explored extensively [27]–[29]. The typical equation for particle velocity with inertial weight is shown below in (7) where β_p is the best local particle position, X_p is the current particle position, and G is the global best position. The value for w is typically set to be less than one and $C_1 + C_2 \geq 4$. The inertial weight factor is typically decreased with each iteration [21], and is used to force a particle to stop excessive exploration of the solution space.

$$V_p = w \cdot V_p + C_1[U(0,1) \cdot (\beta_p - X_p)] + C_2[U(0,1) \cdot (G - X_p)] \tag{7}$$

By adding a randomness to the value for w, it's possible to speed up the convergence of the PSO algorithm. This can be done with either Chaotic Descending Inertial Weight (CDIW) or Chaotic Random Inertial Weight (CRIW) [27]. In CDIW, w is calculated by first randomly choosing a value within the interval of $(0, 1)$, called z. This value is then logistically mapped by the equation in (8). Next, w is calculated with this logistically mapped variable as per (9). Here, i_{max} is the maximum number of PSO iterations and i is the current iteration. The variables w_1 and w_2 are set as the desired starting and ending inertial weights (typically 0.9 and 0.4, as mentioned earlier).

$$z = 4 \cdot z \cdot (1 - z) \tag{8}$$

$$w = \frac{(w_1 - w_2)(i_{max} - i)}{i_{max}} + z \cdot w_2 \tag{9}$$

An alternative Chaotic Inertial Weight implementation is called Chaotic Random Inertial Weight (CRIW). This differs from CDIW in that the value for w is random every iteration as opposed to decreasing, as Fig. 12 shows. The equation for determining w in CRIW is shown in (10).

$$w = \frac{1}{2}[U(0,1) + z] \tag{10}$$

2) Constriction: Maurice Clerc *et al.* introduced a new way to modify a particle's velocity in order to speed convergence to a solution in [19], [20] and was later explored more in depth in [26]. This is done by use of constriction factor, χ, and weighted velocity constants, C_1 and C_2. This constriction factor serves to limit the swarm to a smaller search area as it converges on an optimal solution. The velocity coefficients of C_1 and C_2 are used to provide more emphasis on a particle's local best solution or the global best solution, respectively. The velocity update equation can be seen in (11) where β_p is the best local particle position, X_p is the current particle position, and G is the global best position. Equation (12) shows the equation to determine the new particle position. Equation (14) shows the equation used to calculate the constriction factor, χ.

$$V_p = \chi(V_p + C_1[U(0,1)\cdot(\beta_p - X_p)] + C_2[U(0,1)\cdot(G - X_p)]) \tag{11}$$

$$X_p = X_p + V_p \tag{12}$$

$$\phi = C_1 + C_2 \tag{13}$$

$$\chi = \frac{2}{2 - \phi - \sqrt{\phi^2 - 4\phi}} \tag{14}$$

C. Fitness Function

The biggest design challenge with optimization problems is that of the fitness function [13]–[15], [24]. Care must be taken to add in as many variables as possible in order to eliminate the possibility of unwanted states (for example, a 90° Phase Margin but $10\,dB$ Gain). However, when more variables are added they will typically need to be weighted in order to emphasize the more important parameters of the circuit.

A way to aid in the convergence of GAs is to add a penalty to the fitness function [15], [16]. The idea is to decrease the value of the fitness function based on the region that the variable is in. Thus, a penalty function could produce a value for anything below a given acceptable range to cause the fitness value to decrease more rapidly the further it is away from the ideal value.

A generalized fitness function equation with penalties is shown below in (15) where α is a weighted constant for variable θ and $P(\theta)$ is a function that determines the penalty associated with the solution presented by θ.

$$F = \sum_{i=1}^{N} \alpha_i \theta_i - P(\theta_i) \tag{15}$$

For the algorithm implementations explored here, the following variables were selected for the fitness function:

- Phase Margin, ϕ_M
- Gain Margin, G_M
- DC Gain, A_0

Since each of these variables have varying sizes, it is wise to normalize them to some ideal value such that the sum of all the ideal, normalized values is equal to the number of parameters. This makes it significantly easier to determine the necessary values for α that help to speed convergence.

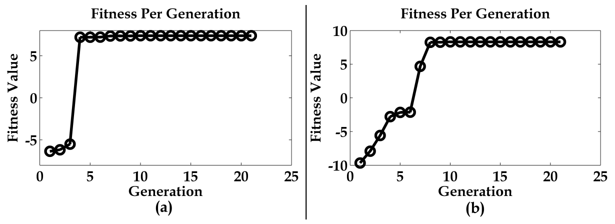

Fig. 13: Comparison of Convergence Speeds for a) QBGA with NPF and b) QBGA with NLF. Drones = 30, Mutation Rate = $0.1 \cdot queenAge$, $\alpha_{\phi_M} = 3$, $\alpha_{G_M} = 0.6$, $\alpha_{A_0} = 2$.

TABLE I: Comparison of Optimization Algorithms for Size of 10 and Max Iteration of 100.

Algorithm	Error From Expected
QBGA (Static)	0.268%
QBGA (Variable)	0.179%
PSO-CDIW	133.286%
PSO-CRIW	133.036%
PSO-Constrict	73.750%

In addition to normalizing all of the variables, each value was placed into a function that transforms the linear data into parabolic data such that it has a maximum at the ideal point. For example, if the ideal ϕ_M is chosen as 70°, $N_{\phi_M} = \frac{\phi_M}{70}$ and thus $f(\theta(\phi_M)) = -(N_{\phi_M} - 1)^2 + 1$. The constant scalar is there so that the maximum value occurs at a value of 1 instead of 0.

Experimental results show that the normalized parabolic fitness function, NPF, exhibits faster global optimum convergence than using only the normalized data alone as shown in Figure 13. Note that the fitness value in the linear case is, in fact, larger than the NPF case. This, however, is expected as values *greater* than the the normalized ideal parameter will subtract from the fitness value in the NPF while it adds in the linear case. As such, comparing the fitness values directly does not result in a good comparison of fitness function strength. The only valid comparison is to compare the number of generations it takes to converge. Equation (16) shows the full NPF used for this implementation.

$$F = \sum_{i=1}^{N} -\alpha_i (N(\theta_i) - 1)^2 + \alpha_i - P(\theta_i) \tag{16}$$

IV. Algorithm Comparisons

The algorithms compared are QBGA with static mutation, QBGA with variable mutation, PSO-CDIW, PSO-CRIW, and PSOC (all with NPF). Table I shows a comparison of the error from the optimal fitness value after 100 iterations. Fig. 14 shows this convergence graphically (omitting the static mutation QBGA since this was compared earlier).

It's clear that the QBGA implementation is much better than any of the PSO algorithms tested for boost converter controller optimization. Both inertial weight algorithms had difficulty escaping from their local optima. The PSOC had a similar problem but was able to converge on a better solution than either inertial weight algorithms. The better performance of the PSOC is expected based on previous results [26].

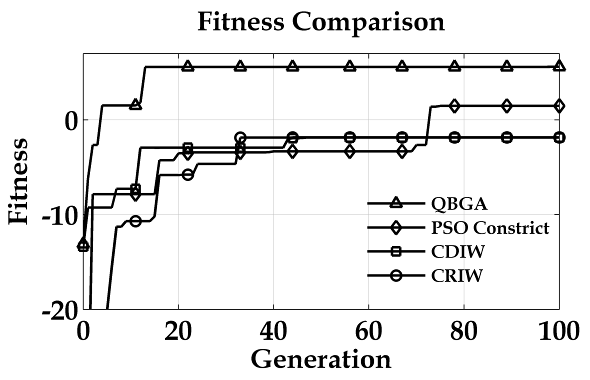

Fig. 14: Comparison of Convergence Speeds for QBGA and PSO with size of 10 and max iteration of 100.

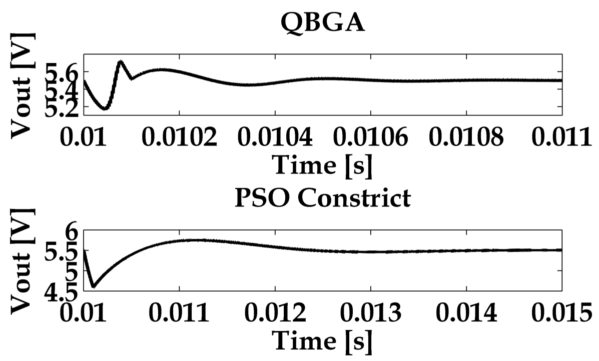

Fig. 15: Simulation Comparison of Optimization Algorithm Controller Solution for DCM Boost Converter.

A. Simulation with Generated Controller

In order to properly compare the solutions generated by the algorithms, the controller solutions were placed into MATLAB's Simulink toolbox using Powerlib.

In order to test the solution of each algorithm, the final transfer function was placed into the circuit and then the circuit was simulated over a time of $20\,ms$. At $10\,ms$ the load was stepped to 25% its original value. For the DCM Converter, the extracted transfer function from the QBGA algorithm is shown in (17) while the transfer function from the Constricted PSO algorithm is shown in (18). The QBGA algorithm produces an expected solution equivalent to a Lag Controller, whose general transfer function was shown previously in (2). The PSO algorithm converged on a solution similar to the PID transfer function in (6) which, for a DCM Converter, is valid but unnecessary.

$$T_{QBGA}(s) = \frac{0.0034s + 159.8}{0.0026s + 1} \tag{17}$$

$$T_{PSO}(s) = \frac{147s^2 + 288.9s + 3.016}{0.196s^3 + 0.0066s^2 + 0.000021s + 0.00002} \tag{18}$$

Fig. 15 shows a comparison of the controller solutions after simulation. As can be seen, the QBGA solution is faster (note the time-scale difference) and is stable, albeit slightly underdamped. The odd peak in the step response for the QBGA solution is due to the converter exiting the small-signal region and briefly incurring large-signal (non-linear) behavior. The PSO solution is also stable, but slower than the QBGA solution. While it is a perfectly adequate controller solution, the QBGA clearly performs better.

Next, the algorithms were run on a Boost Converter operating in CCM. The generated controller solutions are shown below for the QBGA and PSO algorithms in (19) and (20), respectively. Both algorithms converge on a PID-like transfer function which is absolutely neccessary for a Voltage-Mode-Controlled CCM Boost Converter.

$$T_{QBGA}(s) = \frac{348.6s + 348.6}{0.00017s^3 + s^2 + 3.5 \times 10^{-7}s + 1} \tag{19}$$

$$T_{PSO}(s) = \frac{4.2 \times 10^4 s + 2.02 \times 10^4}{0.06s^3 + 48.53s^2 + 0.27s} \tag{20}$$

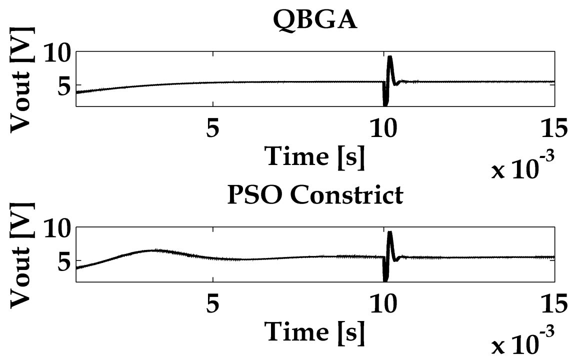

Fig. 16: Simulation Comparison of Optimization Algorithm Controller Solution for DCM Boost Converter.

The comparison of the simulations for each solution are shown in Fig. 16. The step-response is nearly identical for both, indicating that the small-signal model for the whole system is also quite similar. However, observing the start-up of the system shows that the large-signal response is more variable. The QBGA gracefully rises to a steady-state value while the Constricted PSO has slight oscillation before settling to a steady-state value. That said, the QBGA solution is only marginally better than the PSO here. This is because the CCM Boost Converter provides fewer acceptable solutions than the DCM case which makes it much more difficult for an optimization algorithm to converge on an acceptable solution (both Fitness values were around 1, as opposed to the ideal value of 5.6).

V. Conclusion

It has been shown that for optimizing the controllers in VMC Boost Converter, a Queen-Bee GA achieves better results in less time than that of three different PSO algorithms. All algorithms performed much better when optimizing Boost Converters in DCM as opposed to CCM. This is expected due to the similarity in solution space size for both modes, yet a smaller acceptable solution region in the CCM case. The large phase shift due to the LC resonance in the CCM converter causes the optimization algorithms explored here to get stuck in regions of acceptable, but non-ideal, solutions and they have difficulty escaping.

The run-time of each algorithm was found to be rather intensive, taking nearly ten minutes to complete 200 iterations at a swarm/nest size of 20. Due to the large solution space, it's imperative to allow the algorithm to run for as long as possible in order to guarantee convergence to an optimal solution. However, it seems to be far more beneficial to run the algorithms for a short amount of time in order to converge *near* an optimal solution and use the values generated as a starting point in the design of the controller. This allows a circuit designer to start from a point of known operation rather than requiring tedious calculations to be performed and iterated on to find an acceptable solution.

In terms of Power Electronics Circuit optimization, it's clear that Queen-Bee Genetic Algorithms hold significant promise as a design-tool that can be used as an aid when designing various circuit blocks.

References

[1] Vatché Vorpérian, "Simplified Analysis of PWM Converters Using Model of PWM Switch Part I and Part II," in *IEEE Trans. Aerosp. Electron. Syst.*, vol. 26, no. 3, pp. 490-505, May 1990.

[2] R.D. Middlebrook, and Slobodan Ćuk, "A General Unified Approach to Modeling Switching-Converter Power Stages," in *Proceedings of the IEEE Power Electron. Special. Conf.*, pp. 18-34, 1976.

[3] R.D. Middlebrook, "Small-Signal Modeling of Pulse-Width Modulated Switched-Mode Power Converters," in *Proceedings of the IEEE*, vol. 76, no. 4, pp. 343-354, April 1988.

[4] Kevin C. Fronczak, "Stability Analysis of Switched DC-DC Boost Converters for Integrated Circuits," Master's Thesis, Department of Electrical and Microelectronic Engineering, Rochester Inst. of Tech., Rochester, NY, August 2013.

[5] Christophe P. Basso, *Switched-Mode Power Supplies*, 1st ed. New York, NY: McGraw-Hill, 2008.

[6] Alberto Reatti, and M.K. Kazimierczuk, "Current-Controlled Current Source Model for a PWM DC-DC Boost Converter Operated in Discontinuous Conduction Mode," in *IEEE Internat. Symp. Circuits Syst.*, Geneva, Switzerland, May 28-31, 2000, pp. III/239-III/242.

[7] Robert W. Erickson, and Dragan Maksimović, *Fundamentals of Power Electronics*, 2nd ed. Secaucus, NJ: Academic, 2000.

[8] Jeongjin Roh, "High-Performance Error Amplifier for Fast Transient DC-DC Converters," in *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 52, no. 9, pp. 591-595, September 2005.

[9] Cheung Fai Lee and Philip K.T. Mok, "A Monolithic Current-Mode CMOS DC-DC Converter With On-Chip Current-Sensing Technique," in *IEEE J. Solid-State Circuits*, vol. 39, no. 1, pp. 3-14, January 2004.

[10] Hong Yao, "Modeling and Design of a Current Mode Control Boost Converter," M.S. Thesis, Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, 2012.

[11] Hong-Wei Huang, Hsin-Hsin Ho, Chia-Jung Chang, Ke-Horng Chen, and Sy-Yen Kuo, "On-Chip Compensated Error Amplifier for Fast Transient DC-DC Converters," in *IEEE Internat. Conf. Electro/inform. Tech.*, East Lansing, MI, pp. 103-108. May 7-10, 2006.

[12] Chan-Soo Lee, Young-Jin Oh, Kee-Yeol Na, Yeong-Seuk Kim, Nam-Soo Kim "Integrated BiCMOS Control Circuits for High-Performance DC-DC Boost Converter," in *IEEE Trans. Power Electron.*, vol. 28, no. 5, pp. 2596-2603, May 2013.

[13] Günter Rudolph, "Convergence Analysis of Canonical Genetic Algorithms," in *IEEE Trans. Neural Networks*, vol. 5, no. 1, pp. 96-101, January 1994.

[14] Jeffrey A. Joines, Christopher R. Houck, "On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA's," in *Proceedings of the First IEEE Conf. on Evol. Comput.* vol. 2, pp. 579-584, June 1994.

[15] Mitsuo Gen, Runwei Cheng, "A Survey of Penalty Techniques in Genetic Algorithms," in *Proceedings of IEEE Internat. Conf. Evol. Comput.*, pp. 804-809, May 1996.

[16] Vassilios Petridis, Spyros Kazarlis, Anastasios Bakirtzis, "Varying Fitness Functions in Genetic Algorithm Constrained Optimization: The Cutting Stock and Unit Commitment Problems," in *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 28, no. 5, pp. 629-640, October 1998.

[17] Adrian Thompson, Paul Layzell, Ricardo Salem Zebulum, "Explorations in Design Space: Unconventional Electronics Design Through Artificial Evolution," in *IEEE Trans. Evol. Comput.*, vol. 3, no. 3, pp. 167-196, September 1999.

[18] Jun Zhang, Henry Shu-Hung Chung, Wai-Lun Lo, S.Y. Hui, Angus Kwok-Ming Wu, "Implementation of a Decoupled Optimization Technique for Design of Switching Regulators Using Genetic Algorithms," in *IEEE Trans. Power Electron.*, vol. 16, no. 6, pp. 752-763, November 2001.

[19] Maurice Clerc, "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization," in *Proceedings of the 1999 Congress Evol. Comput.*, vol. 3, pp. 1951-1957, July 1999.

[20] Maurice Clerc, James Kennedy, "The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space," in *IEEE Trans. Evol. Comput.*, vol. 6, no. 1 pp. 58-73, February 2002.

[21] R.C. Eberhart, Y. Shi, "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization," in *Proceedings of the 2000 Congress Evol. Comput.*, vol. 1, pp. 84-88, July 2000.

[22] Sung Hoon Jung, "Queen-bee Evolution for Genetic Algorithms," in *Electron Letters* vol. 39, no. 6, pp. 575-576, March 2003.

[23] Kazuhiro Takemura, Tetsushi Koide, Hans Jürgen Mattausch, Toshio Tsuji, "Analog-Circuit-Component Optimization with Genetic Algorithm," in *IEEE Internat. Midwest Symp. Circuits Syst.*, vol. 1, pp. 489-492, July 2004.

[24] Shinn-Ying Ho, Li-Sun Shu, Jian-Hung Chen, "Intelligent Evolutionary Algorithms for Large Parameter Optimization Problems," in *IEEE Trans. Evol. Comput.*, vol. 8, no. 6, pp. 522-541, December 2004.

[25] Jun Zhang, Henry S.H. Chung, W.L. Lo, "Pseudocoevolutionary Genetic Algorithms for Power Electronics Circuits Optimization," in *IEEE Trans. Syst., Man, Cybern.*, vol. 36, no. 4, pp. 590-598, July 2006.

[26] Daniel Bratton, James Kennedy, "Defining a Standard for Particle Swarm Optimization," in *Proceedings of the IEEE Swarm Intell. Symp.*, pp. 120-127, April 2007.

[27] Yong Feng, Gui-Fa Teng, Ai-Xin Wang, Yong-Mei Yao, "Chaotic Inertia Weight in Particle Swarm Optimization," in *Second Internat. Conf. Innovat. Comput., Inform. Control*, pp. 475-478, September 2007.

[28] ZHang Tao, Cai Jin-ding, "A New Chaotic PSO with Dynamic Inertia Weight for Economic Dispatch Problem," in *Internat. Conf. Sustain. Power Gen. Supply*, pp. 1-6, April 2009.

[29] J. C. Bansal, P. K. Singh Mukesh Saraswat, Abhishek Verma, Shimpi Singh Jadon, Ajith Abraham, "Inertia Weight Strategies in Particle Swarm Optimization," in *Third World Congress on Nature and Bio. Inspired Comput.*, pp. 633-640, October 2011.

[30] Zhiguo Bao, Takahiro Watanabe, "A New Approach for Circuit Design Optimization Using Genetic Algorithm," in *Internat. SoC Design Conf.*, vol. 1, pp. I383-I386, November 2008.

[31] Zhiguo Bao, Takahiro Watanabe, "A Novel Genetic Algorithm with Cell Crossover for Circuit Design Optimization," in *IEEE Internat. Symp. Circuits Syst.*, pp. 2982-2985, May 2009.

[32] Kinattingal Sundareswaren, V.T. Sreedevi, "Boost Converter Controller Design Using Queen-Bee-Assisted GA," in *IEEE Trans. Indust. Electron.*, vol. 56, no. 3, pp. 778-783, March 2009.

[33] Ahmed F. Zobaa, Agostino Lecci, "Power Circuit Design Based on PSO Optimization," in *Internat. Power Electron. and Motion Control Conf.*, pp. 121-125, September 2010.

[34] M. Kotti, B. Benhala, M. Fakhfakh, A. Ahaitouf, B. Benlahbib, M. Loulou, A. Mecheqrane, "Comparison between PSO and ACO Techniques for Analog Circuit Performance Optimization," in *Internat. Conf. Microelect.*, pp. 1-6, December 2011.

[35] Xuejan Song, Yanli Cui, Aiting Li, "Optimization Algorithm of Evolutionary Design of Circuits Based on Genetic Algorithm," in *Fifth Internat. Symp. Comput. Intell. Design*, vol. 1, pp. 336-339, October 2012.