International Conference on Control, Automation and Systems 2008
Oct. 14-17, 2008 in COEX, Seoul, Korea

Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) Optimizing Linear Quadratic Regulator Controller On DC-DC Converter

S. Eshtehardiha[1], M. Bayati Poudeh[1], S. A, Emami[2] and M. Moradiyan[1]

[1] Department of Electrical Engineering, Islamic Azad University, Najaf abad Branch, Isfahan , Iran
(Eshtehardiha@gmail.com, poudeh@gmail.com)
[2] Department of Electrical Engineering, Islamic Azad University, Khomeinishahr Branch, Isfahan , Iran
(emami_dna@yahoo.com)

Abstract: In this paper, two different control methods on DC-DC Converter are compared with together. These converters are used for the stabilization or the control of DC voltage of a battery. In addition to other applications, DC converters feed electric vehicles (trucks, electric vehicles, and subway locomotives), telephone sets and civil inverters. Lately, improvement the performance of the DC-DC converter is one of the goals of the engineers in the industries. In this way, several control methods are used to control Buck converters. In this paper, Linear Quadratic Regulator controller is used to optimize the DC-DC converter performance. Also other controller, by the way, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) are used with these control methods, to obtain the best coefficients in them. The results are shown the capability of the control methods in the improvement of the above-mentioned converter functioning.

Keywords: Genetic Algorithm , Particle Swarm Optimization , Linear Quadratic Regulator , DC-DC converter

1. INTRODUCTION

A buck converter or step-down switch mode power supply can also be called a switch mode regulator. Popularity of a switch mode regulator is due to its fairly high efficiency and compact size and a switch mode regulator is used in place of a linear voltage regulator at relatively high output, because linear voltage regulators are inefficient. The DC converter is a device which transforms AC to DC. This device is also known as an AC to DC converter. A Chopper can be considered as a DC equivalent of an AC transformer with a convertible constant convertible in a continuous form. Like a transformer, the converter can be employed for stepwise increase or reduction of DC source voltage. The converters are wildly used for the control of motor voltage in electric cars, ceiling elevators, mine excavation etc. Their specific features are the precise control of acceleration with high efficiency and fast dynamic response [1].

Since the power devices used in linear regulators have to dissipate a fairly large amount of power, they have to be adequately cooled, by mounting them on heatsinks and the heat is transferred from the heatsinks to the surrounding air either by natural convection or by forced-air cooling. Heatsinks and provision for cooling makes the regulator bulky and large. In applications where size and efficiency are critical, linear voltage regulators cannot be used. Converters are also employed in DC motors to return the energy to its source. In this way, it results in the saving of energy in the transportation systems in prolonged stoppage. Converters are also used in DC voltage regulators along with an inductor to produce a DC current source especially for the current source inverters [2]. A switch mode regulator overcomes the drawbacks of linear regulators. Switched power supplies are more efficient and they tend to have an efficiency of 80% or more. They can be packaged in a fraction of the size of linear regulators. Unlike linear regulators, switched power supplies can step up or step down the input voltage.

Some control methods have stated the issue of control through pole placement [1]. Another method is the use of state feedback in the control of DC-DC converters [2]. In modeling area of DC-DC converters, a variety of models are presented which comprises desirable responses by administration of control methods. Most of the articles have concentrated on controlling designs of PID and PI [3]. Of other control methods, the feedback loop is another among others, [4], and [5]. The use of LQR method for the improvement of Buck converter function is the subject presented in [6].

In this paper, Section 2 describes the model of the Buck converter. The LQR control method is detailed in Section 3, Genetic algorithm is shown in Section 4, another optimization method, Particle Swarm Optimization, is described in Section 5. The computer simulation results are presented and discussed in Section 6, and Finally, Section 7 concludes this paper.

2. BUCK CONVERTER

DC-DC converters were referred to as choppers earlier, when SCRs were used. Nowadays, IGBTs and MOSFETs are the devices used for dc-dc conversion and these circuits can be classified as switch mode power supply circuits. The abbreviation or acronym for switch mode power supply is SMPS [7]. In a Buck converter the average amount of output voltage V_{out} is less than the input voltage V_{in}. The regulator circuit uses the power switch depicted in Fig. 1.

The dc-dc converters can have two distinct modes of operation: Continuous conduction mode (CCM) and discontinuous conduction mode (DCM).The function of the circuit is divided into two parts. The first part starts

978-89-93215-01-4-98560/08/$15 ⓒICROS

when the switch is turned on at *t=0*. The input current which is rising passes through *L* filter inductor, *C* filter capacitor, and *R* charge resistance. The second part starts when the switch is turned off at $t=t_1$. Due to the presence of stored energy in the inductor, the inductor current continues passing from *L*, *C*, load and *D*.

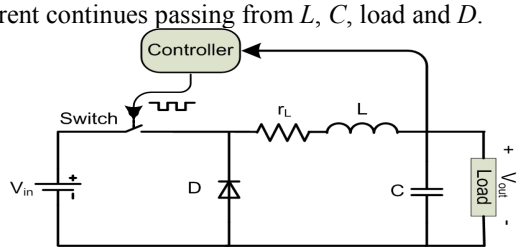

Fig. 1. Buck converter

The inductor current declines until the second switch switching in the next cycle. In this model, *Vo* is the system output voltage and V_{ref} is the converter voltage. The converter state equations in low-frequency state, is presented.

Switch ON:

$$V_{dc} - V_o = (R_s + R_L)i_L + L(di_L/dt) \ ,$$

$$i_L = C(dV_o/dt) + V_o/R \ ,$$

$$\dot{X} = A_1 X + B_1 V_{dc} \ , \qquad (1)$$

$$B_1 = \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} \ . \ A_1 = \begin{bmatrix} \frac{-(R_L + R_S)}{L} & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC} \end{bmatrix} \ ,$$

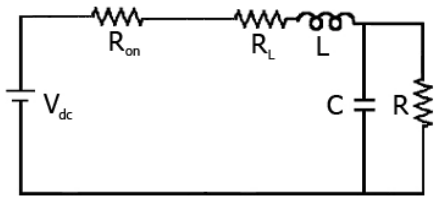

Fig. 2. Switch on for a time duration *DT*

Switch OFF:

$$(R_d + R_L)i_L + L(di_L/dt) + V_o = 0 \ ,$$

$$i_L = C(dV_o/dt) + V_o/R \ ,$$

$$\dot{X} = A_2 X + B_2 V_{dc} \ ,$$

$$A_2 = \begin{bmatrix} \frac{-(R_L + R_d)}{L} & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC} \end{bmatrix} \ , \ B2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \ .$$

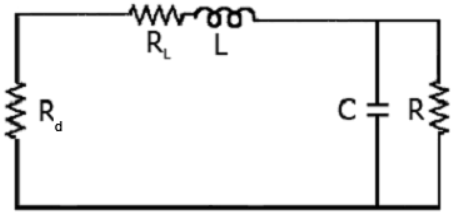

Fig. 3. Switch off for time duration *(1-DT)*

Now it is required to show the effect of on and off durations of switch in equations (1) and (2) to obtain the mean values of state equations.

$$\dot{X} = AX + BV_{dc} \qquad X_1 = i_L \ , \qquad X_2 = V_o \ , \qquad X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} \ ,$$

$$A = dA_1 + (1-d)A_2 \quad , \ B = dB_1 + (1-d)B_2 \qquad (3)$$

$$d = \frac{t_{on}}{T}$$

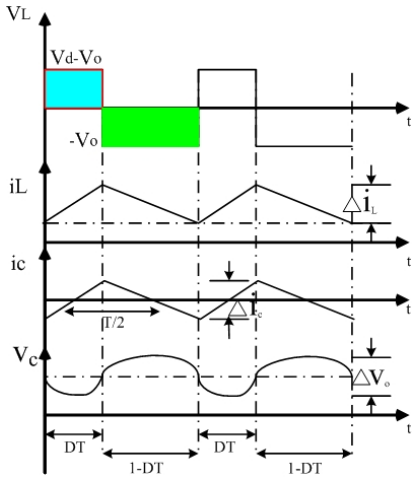

Fig. 4.key waveforms

3. LQR DESIGN

The LQR design problem has been extensively investigated for the past four decades [8-12]. Consequently, the theory of linear optimal control has been very well developed. Central to the optimal stabilizer design problem is the formulation of the cost function and the choice of the state and control weighting matrices. Although the need for the design of a cost function that reflects the physical characteristics and practicalities of the studied system is reasonably understood, the choice of the elements of the weighting matrices that would translate this understanding still remains a difficulty. This is mainly due to the interactive nature of the states and controls of linear multivariable dynamic systems. In the following we present a novel systematic approach to this problem which overcomes the shortcomings of existing approaches.

In this method, the feedback gain matrix is determined if *J* energy function is optimized. To achieve equilibrium among range control parameters, response speed, settling time, and proper overshoot rate**,** all of which guarantee the system stability, the LQR is employed.

3.1 LQR ALGORITHM

For a system in the form of $\dot{X} = AX + BU$ the LQR method determines the *K* matrix of the equation $U(t) = -kX(t)$ to minimize the

$J = \int_0^\infty (X^T QX + U^T RU)dt$ function.

R and Q matrices express the relation between error and energy expense rate. R and Q are also the definite positive matrices. From the above equation, we have [5]:

$$J = \int_0^\infty (X^T QX + X^T K^T RKX)dt$$
$$= \int_0^\infty (X^T (Q + K^T RK)X)dt \quad (4)$$

3.2 FLOW CHART OF LQR CONTROLLER

Statement of the Problem	
Given the plant as x(t) = Ax(t) + Bu(t) the performance index as $J = \int_0^\infty (X^T QX + X^T K^T RKX)dt$ $= \int_0^\infty (X^T (Q + K^T RK)X)dt$ and the its conditions $x(t_0) = x_0; x(\infty) = 0,$ Find the optimal control, index.	
Solution of the Problem	
Step1	Solve the matrix algebraic Riccati equation $A^T P + PA - PBR^{-1}B^T P + Q = 0$
Step 2	Solve the optimal state x* (t) from $X^*(t) = (A - BR^{-1}B^T \bar{P})X^*(t)$ with initial condition $x(t_0) = x_0$.
Step 3	Obtain the optimal control u* (t) from $u^*(t) = -R^{-1}B^T PX^*(t)$
Step 4	Obtain the optimal performance index from $J^* = \frac{1}{2}e^{2at_0}x^{*'}(t_0)\bar{P}x^*(t_0)$

4. GENETIC ALGORITHM

Genetic Algorithms (GA) is a technique that mimics natural evolution to solve problems in wide variety of domains. In 1960 the first serious investigation into Genetic Algorithms (GAs) was undertaken by John Holland. Genetic algorithms have become popular due to several factors. They have been successfully applied to self-adaptive control systems and to function optimization problems. In nature, individuals best suited to competition for scanty resources survive. Adapting to changing environment is essential for the survival of individuals of each species. While the various features that uniquely characterize an individual determine its survival capacity, the features in turn are determined by the individual's genetic content [13]. They are a powerful search technique, yet are computationally simple. The search method they use is robust since it is not limited like other search methods with regard to assumptions about the search space.

Genetic algorithms are different from normal search methods encountered in engineering optimization in the following ways:

1- GA work with a coding of the parameter set not the parameters themselves.

2- GA search from a population of points, not a single point.

3- GA use probabilistic transition rules, not deterministic transition rules.

The genetic algorithm is an algorithm which is based on natural evolution and the survival of the best chromosome. There are three basic differences between genetic algorithm and optimization classical methods. Firstly, the genetic algorithm works on the encoded strings of the problem parameters. Each string is the representative of one answer to the problem, and the real quantities of the parameters are obtained from the decoding of these strings. Secondly, the genetic algorithm is a search algorithm which works on a population of search spaces [5]. This quality causes the genetic algorithm to search different response spaces simultaneously reducing the possibility of being entrapped at local optimized points. Thirdly, the genetic algorithm does not need previous data from the problem response space such as convexity and derivable. It is only necessary to calculate a response function named fitness function. This function expresses the rate of response proximity to the goal function of the intended algorithm. Different methods are used for encoding the parameters in genetic algorithm [14].

A GA operates through a simple cycle of stages as follows:

1- Creation of a "population" of strings (using a random generator).

2- Evaluation of each string (using a fitness function).

3- Selection of best strings (using a suitable selection method).

4- Genetic manipulation to create the new population of strings.

A simple genetic algorithm is composed of three operators: *selection*, *crossover*, and *mutation*.

1- Selection is a process where an old string is carried through into a new population depending on the value of fitness function. Due to this move, strings with better fitness values get larger numbers of copies in the next generation. Selecting good strings for this operation can be implemented in many different ways.

2- A simple crossover follows the selection is made in three steps. First, the newly selected strings are paired together at random. Second, an integer position "n" along every pair of strings is selected uniformly at random. Finally, based on a probability of crossover (p_c),

the paired strings undergo crossing over at the integer position "n" along the string.

3- Mutation is simply an occasional random alteration of a string position based on probability of mutation (p_m). In a binary code, this involves changing a 1 to a 0 and vice versa. The mutation operator helps in avoiding the possibility of mistaking a local minimum for a global minimum.

Several practical means of deciding when to stop regeneration are used. These are:

1- Stop after a period of time (Maximum number of generations).

2- Stop when the average fitness is close to the minimum (or the maximum) of the fitness.

3- Stop when there is no improvement in the maximum (or minimum) value of the fitness.

4- Stop after finding a better solution than the one proposed by other means or criterion.

5- Stop after finding the desired maximum (or minimum).

The genetic algorithm flowchart is shown in Fig. 5.

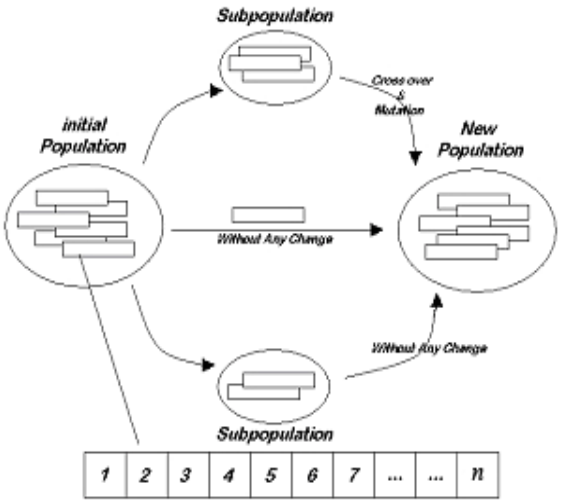

Fig.5. Genetic algorithm flowchart

5. PARTICLE SWARM OPTIMIZATION (PSO)

The particle swarm optimization is an evolutionary computation technique developed by Eberhart and Kennedy inspired by the social behavior of bird flocking and fish schooling [15]. PSO is a population based optimization tool. In PSOs, that are inspired by flocks of birds and shoals of fish, a number of simple entities (particles) are placed in the parameter space of some problem or function, and each evaluates the fitness at its current location. The advantages of PSO compared to other evolutionary computational techniques are [16]:

1. PSO is easy to implement.
2. There are few parameters to be adjusted in PSO.
3. All the particles tend to converge to the best solution quickly.

In PSO each particle adjusts its flight according to its own and its companion's flying experience. Each particle keeps track of its co-ordinates in the solution Space which are associated with the best solution (fitness) that has achieved for by that particle. This value is called personal best, 'J_{pbest}'. Another best value obtained so far by any particle in the neighborhood of that particle. This value is called global best, 'X_{gbest}'. The basic concept of PSO lies in accelerating each particle towards its 'J_{pbest}' and the 'X_{gbest}' locations, with a random weighted acceleration at each time step. Each particle tries to modify its position using the information such as the current positions, the current velocities, the distance between the current position and 'J_{pbest}', the distance between the current position and the 'X_{gbest}'. The mathematical equations for the searching process are [17]:

$$v_i^{k+1} = wv_i^1 + c_1\, rand_1(...)(pbest_i - s_i^k) + c_2\, rand_2(....)(gbest - s_i^k) \quad (5)$$

$$x_I^{K+1} = X_I^K + v_i^{k+1} \quad (6)$$

Where

v_i^k Velocity of particle i at iteration k

W weighting function

c_1 and c_2 weighting factor rand uniformly distributed random number between 0 and 1.

s_i^k Current position of the particle i at iteration k

$pbest_i$ pbest of particle i

gbest best value obtained by any particle so far

The first term in (14) is the former velocity of the particle(s), the second is the cognition modal, which expresses the thought of the particle itself, and the third represents the social model. The three parts together determines the space searching ability. The first part has the ability to search for local minimum. The second part causes the swarm to have a strong ability to search for global minimum and avoid local minimum. The third part reflects the information sharing among the particles. Under the influence of the three parts, the particle can reach the best position [18]. In the above procedure, the maximum velocity v_{max} determines the resolution of fitness regions are searched between the present position and target position. If max v too high, particle might fly past good solution. If max v is too small, the convergence could be slower. According to experience of PSO, v_{max} takes often 10% to 25% of the dynamic range of the velocity.

The PSO algorithm used in this thesis can be briefly discussed by the following steps.

1: Initialize a population of 'pop' particles with random positions within the lower and upper bound of the problem space. Similarly initialize randomly 'pop' velocities associated with the particles.

2: Evaluate the optimization fitness functions J for the initial population.

3: Find the minimum fitness value for fitness functions J in step 2 and call it J_{pbest} and let the particle associated with it be X_{pbest}.

4: Initially set J_{gbest} equal to J_{pbest}.