Genetic Algorithms for Multi-Objective Optimization in Dynamic Systems

Ceyhun Eksin
Boğaziçi University
Department of Industrial Engineering
Boğaziçi University, Bebek 34342, İstanbul, Turkey
ceyhun.eksin@boun.edu.tr

Abstract:
This study uses parametric search to meet multiple goals in the behavior of dynamic systems. Parameters are searched using genetic algorithm. Main aim of this study is to discuss how multi-objective parameter search gives essential information about the system. A nonlinear electric circuit is one of the two dynamic models in this paper used for parameter optimization. The electric circuit model shows oscillatory behavior. A fitness function which evaluates period and amplitude and compares it with the desired oscillatory pattern is proposed. Genetic algorithm with the proposed fitness function gives satisfactory results. It is shown that time horizon for a simulation based optimization can be crucial. The second model is a generic System Dynamics model, the stock management problem with second order supply line. The policy parameters are weight of stock adjustment and supply line adjustment. A fitness function that evaluates the settling time, overshoot, and steady state error is proposed. The search results provide some insight on both the fitness function and the system. The obtained results are satisfactory and they show that the response time of the system can be decreased by small overshoot. The paper is a step towards simulation based parameter search becoming an essential support toolbox for model building and policy design in System Dynamics.

1) Introduction

Policy design or improvement of the system behavior is the utmost role of a System Dynamics (SD) model. Despite the growing interest in SD, the field has not succeeded to provide strong and reliable support toolbox to aid the modeler in policy design process. The need for the integration of new optimization tools to SD in model identification, behavior analysis, sensitivity analysis and policy design, has been pointed by several authors (Richardson 1999; Coyle 2000; Yucel and Barlas 2007).

One of the most widely used approaches in policy design is the so-called traditional approach. Traditional approach to policy design is usually an informal process where the model builder or an expert who has an understanding of the system's behavior applies parametric or structural changes to the system on a trial-and-error basis (Coyle 1977). The traditional approach is bounded by the model builder's or expert's intuitive ability to come up with acceptable and good enough policy alternatives. Though the traditional approach could not be abandoned, several studies have been proposed by SD researchers, which emphasize optimization methods to aid the model builder in generating acceptable policies efficiently. Some of these methods applied modal control theory methods (Mohapatra and Sharma 1985; Özveren and Sterman 1989) and others

made use of optimal control theory (Burns and Malone 1974; Keloharju 1982; Coyle 1985). Optimal control theory methods often use heuristics to optimize policy parameters according to a certain objective function. In this sense, they can also be regarded as simulation based optimization methods. Other applications of similar methods to SD models, which make use of simulation based algorithmic search, have followed the early applications. Wolstenholme and Al-Alusi (1987) make use of DYSMOD package to optimize the strategy of an army in a defense model. Dangerfield and Roberts (1999) use hill-climbing search algorithm to fit the model to AIDS data to obtain the distribution of incubation period. Graham and Ariza (2003) present a real world consulting application of the policy design by parameter optimization using simulate annealing. One such algorithm, which has prospect in obtaining desired results efficiently and effectively, is Genetic Algorithm (GA). There are few studies in the SD literature that have also applied GA in policy design. (Grossman 2002; McSharry 2004; Yücel and Barlas 2007; Duggan 2007)

One of the main reasons why such optimization methods are not well established in the literature is the difficulty to define a good objective function. Generally, the simulation based optimization applications to SD have been done via maximizing or minimizing the end value of a single parameter or a set of parameters in the system. (Wolstenholme and Al-Alusi 1987; Miller 1998; Grossmann 2002; McSharry 2004; Duggan 2007) Our study uses parameter search to obtain multiple goals in the behavior of dynamic systems. Parameters are searched using GA. This study shows that parameter optimization can cast doubt on the objective function which could be used as fruitful source of information to either obtain a better objective function or to gain insight about the system. One of the main aims of this study is to show that multi-objective parameter search according to a performance index can yield essential information about the system.

In the next section the GA applied will be explained. Section 3 will explain the application of the parameter search to nonlinear electric circuit where the definition of the objective function and results will be given. After these sections, the search will be applied to the generic Stock Management Model with second-order supply line. Parameter search will be applied with different versions of performance index to show that the parameter search can provide valuable knowledge about the system during the policy design process. In this final section before conclusion, there will be further analysis on the Stock Management Model.

2) Genetic Algorithm (GA)

The GA, as the name implies, mimic the biological evolution process to find the 'fittest' set of parameters. This application of biological process, named as GA was first introduced by John Holland in 1975 (Holland, 1975). GA searches the solution space in multiple and random directions. GA is observed to be an effective algorithm for highly nonlinear solution spaces since it is not typically trapped in local optima. Below is a pseudo-code for a general GA algorithm.

```
choose initial population
 evaluate each individual's fitness
 Scale every individual according to their fitness
 repeat
        select individuals based on their scaled fitness values
        mate pairs at random
```

```
                apply crossover operator with certain probability
                apply mutation operator with certain probability
          evaluate each individual's fitness
until terminating condition (e.g. until at least one individual has
          the desired fitness or enough generations have passed)
```

As the pseudo-code implies each of these steps could be done with various different operators, functions, or values. GAs have many options which provide high flexibility in coming up with an appropriate algorithm for the search space. The reader may refer to (Goldberg 1989) for a detailed description on GAs. For our study, we utilized the *"Genetic Algorithm and Direct Search Toolbox"* of MATLAB®.

The specifications of the GA

The initial population is chosen randomly. The population number is set to 50. Each individual has p genes where p is the number of policy parameters. The fitness values of individuals are scaled proportionally. The individuals for candidate parents are chosen by roulette wheel. In roulette wheel selection, each of the individuals is assigned a share at the roulette wheel proportional with its fitness value. Then the individuals are selected randomly with the individuals with higher share having a greater chance to carry their genetic information to the next generation. For the minimization of the fitness function, the lower the fitness value of the individual, the greater is its share on the wheel. The best individual is always carried to the next generation without any alteration. After the selection, the 70% of the next generation are created using single point crossover and the rest of the next generation is created using mutation. Each gene of a parent has a 0.05 probability of going through mutation. The single point crossover chooses a random integer n between 1 and the number of policy parameters. It takes the values of 1 to n from the first parent and n+1 to the number of policy parameters from the second parent to form the child. The mutation operator selects each gene of an individual and replaces it by a uniform random number with probability 0.05. The terminating condition is a fixed number of generations. It is chosen as 200. The fitness function for each model is different and will be explained with the corresponding models.

3) Application to Nonlinear Electric Circuit

The Model

The equations of the model are as shown below where I is the current and V is the voltage of the RLC circuit connected series.

$$\dot{I} = \frac{V}{L} - \frac{-I + I^3}{L}$$

$$\dot{V} = \frac{-I}{C}$$

Where

$$R = -I + I^3$$

This system is nonlinear as the resistance 'R' of the circuit is modeled nonlinear, and changes with the cube of the current.

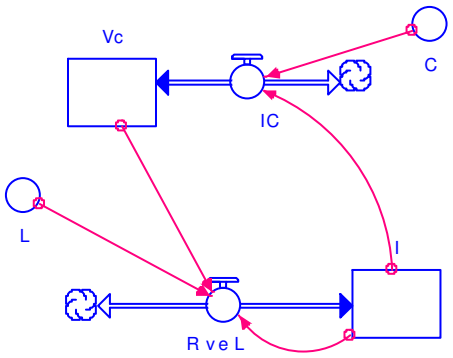

Figure 1: Stock-Flow diagram for the electric circuit model

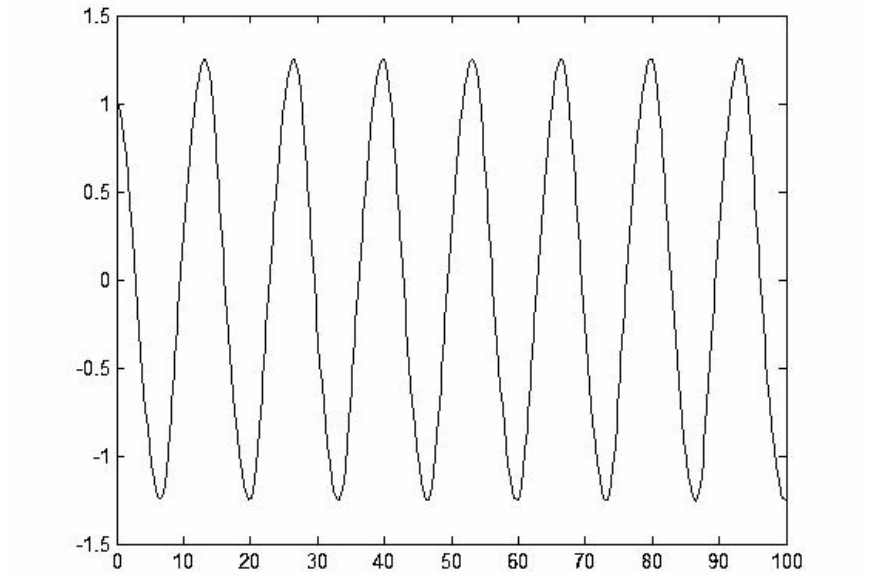

Figure 2: Behavior of the voltage with C = 2, L = 2. Initially I=0, V=1.

The search parameters are capacity (C) and inductance (L) values. The system shows constant oscillatory behavior.

Fitness Function for the Electric Circuit Model

The desired behavior is chosen as a cosine wave with certain period and amplitude for the voltage. The fitness function calculates the average period, and the amplitude. To obtain average period, first the times where oscillation reaches maximum values throughout the simulation time are marked (t_1, t_2, …, t_n). Then the spotted times are subtracted consecutively starting from t_n to obtain periods. The initial period where the system starts from one at time zero to t_1 is not calculated to omit transitory behavior.

$$period_{n-1} = t_n - t_{n-1}$$
$$period_{n-2} = t_{n-1} - t_{n-2}$$
$$\vdots$$
$$period_1 = t_2 - t_1$$

Finally the average period is calculated from these n-1 periods. To calculate the amplitude, max and min values of the voltage is subtracted. The calculated period and amplitude values are compared with the target period and amplitude of the cosine. Penalty for the period and amplitude difference is equal which means both goals are given equal importance. The desired cosine function is $\cos(\pi/10)$ which has a period of 20 and oscillates between of -1 and 1. Note that this fitness function could easily be applied to any model which has potential to show constant oscillation where the desired behavior is oscillatory.

Results

The parameter boundaries are given to be between 0.001 and 5 for both capacity and inductance. Initially the simulation time is set to be 100. The best L and C values and their fitness values obtained after 200 generations by the algorithm for three runs of the algorithm show that the algorithm generally finds near optimum solutions. Since the ideal optimum value can be zero. Additionally, the behaviors of the best sets of parameter values show that the fitness function is well defined.

Table 1: Best policy parameters and their fitness values, time horizon 100

Best three individuals	1	2	3
Fitness Value	0.0114	0.0044	0.2449
L	2.187381755	2.182040847	2.580582881
C	3.860666912	3.875191806	3.423580426

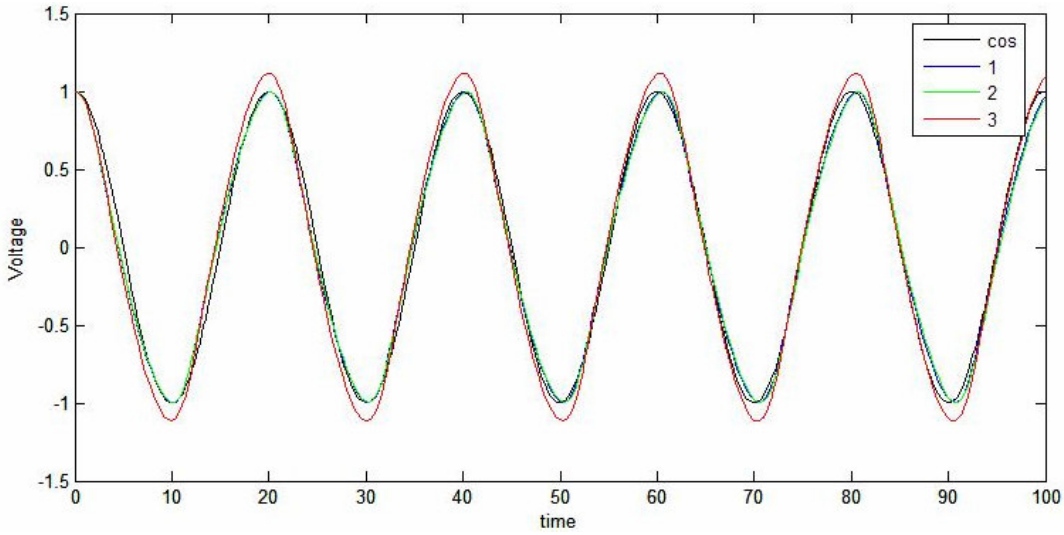

Figure 2: Runs of three best parameter sets, time horizon 100

In these three runs the simulation time was set to be 100 (see Figure 2). When the simulation time is set to 110, the fitness values of the best individuals increase (see Figure 3 and Table 2). This increase is due to the fact that the period is 20 and when t_1, t_2, …, t_n are marked t_n is generally equal to 100 where the time the real maximum value is

reached could be greater than 100. Hence, the average period calculated is affected by the run time.

Table 2: The same best policy parameters in table 1 and their fitness values with 110 as the simulation time

Best three individuals	1	2	3
Fitness Value	0.1555	0.2505	0.3559
L	2.187381755	2.182040847	2.580582881
C	3.860666912	3.875191806	3.423580426

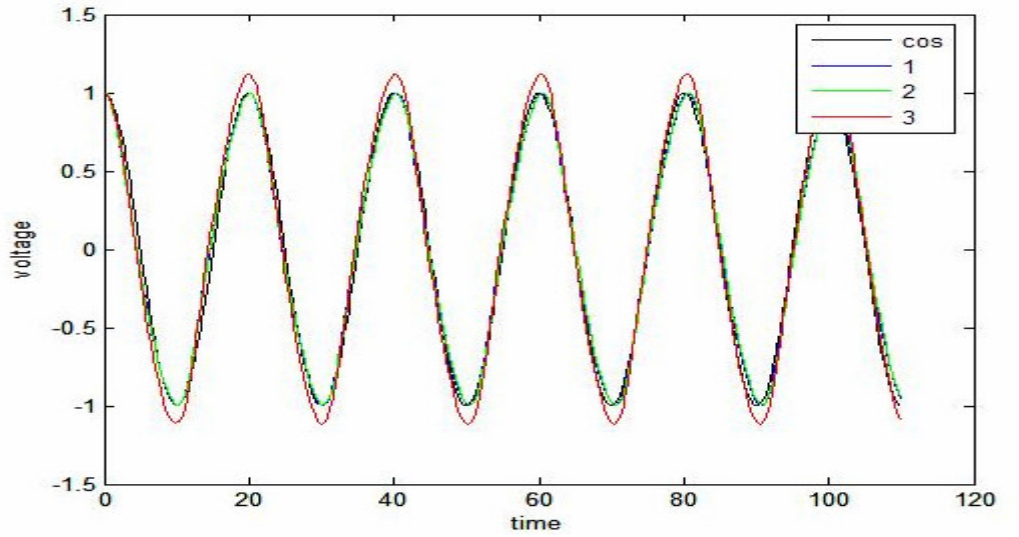

Figure 3: Runs of three best parameter sets, time horizon 110

When the algorithm is again run with simulation time set to 110, slightly better behaviors are observed (see Figure 4 and Table 3). This shows that simulation time is also a parameter that needs to be carefully chosen to avoid undesirable results.

Table 3: New best policy parameters and their fitness values, time horizon 110

Best three individuals	1	2	3
Fitness Value	0.0601	0.02866	0.1502
L	2.249054747	2.186929453	2.098249255
C	3.734942323	3.794276917	3.88143746

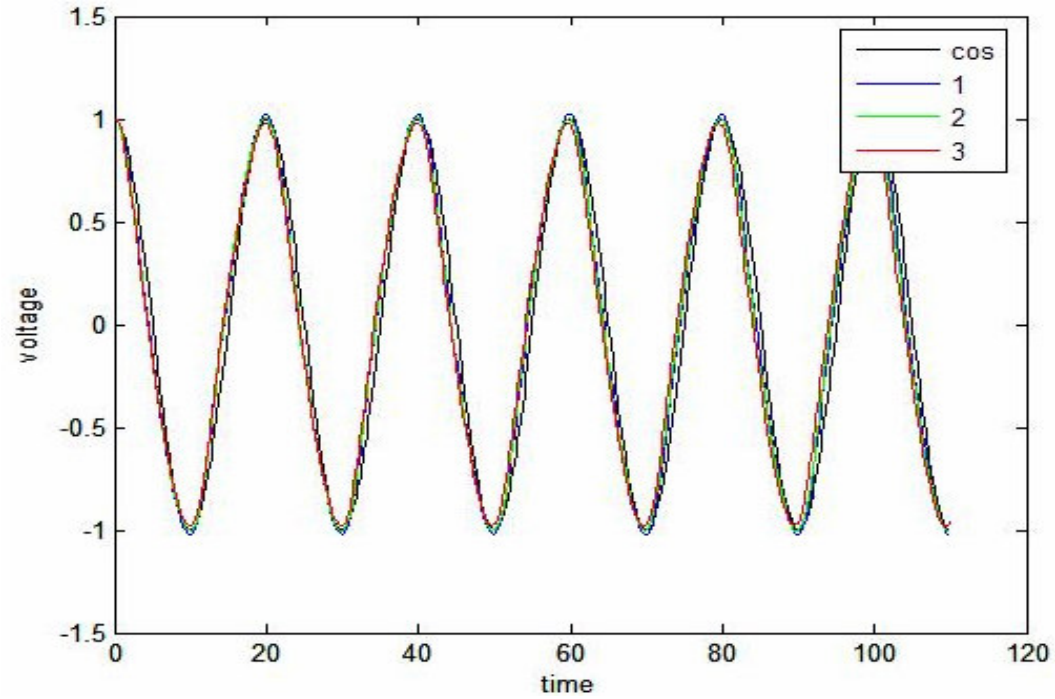

Figure 4: Runs of three new best parameter sets, time horizon 110

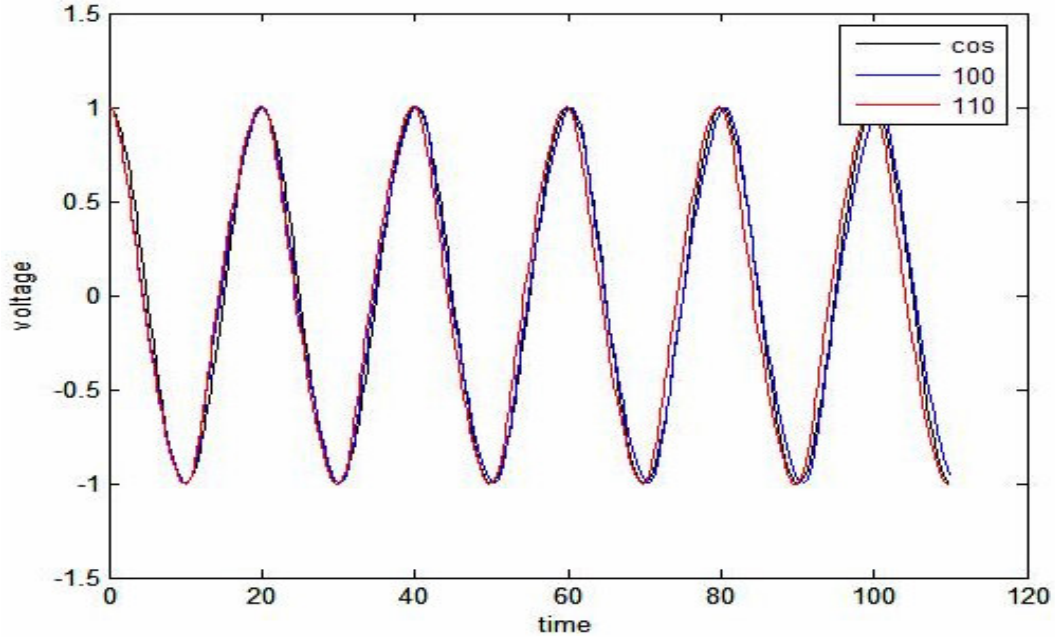

Figure 5: Comparison of the best individuals, individual 1 from table 1, and individual 2 from table 3

Although the time horizon affects the performance of the algorithm, the best set of parameters obtained by the algorithm run with 100unit time horizon, and the best set of parameters obtained by the algorithm run with 110unit time horizon do not have significant difference (see Figure 5). The best individual from the 100unit time horizon runs has a slightly larger period than the cosine function. In general, the algorithm finds good parameter values quickly.

4) Application to Stock Management Model with Second Order Supply Line

The Model

The model is a generic System Dynamics model. It is known that ignoring Supply Line may cause the Stock to produce undesired behaviors in Stock Management Models (Sterman 2000). When Weight of Stock (W_S) and Weight of Supply Line (W_{SL}) are both equal to one, the Stock cannot oscillate (Yaşarcan 2003). The general aim in the model is for the Stock to reach the desired level fast and stably. The response time of the system decreases when the Stock Adjustment Time (T_{SA}) is decreased. But this effect is not linear and after a point decreasing T_{SA} does not cause the response time to quicken. Furthermore, very small T_{SA} may cause stability problems (Yaşarcan 2003). The general conception in the literature for Stock Management is to decrease T_{SA} till the system responds fast enough and set W_S and W_{SL} to one (Yaşarcan 2003).

In the model used, Loss Flow is constant and is exactly known by the decision maker. The specifications of the model are given in Appendix. The ordering decision includes the Supply Line Adjustment (SLA) and Stock Adjustment (SA) structures.

$$ControlFlow = LossFlow + \frac{W_S xSA + W_{SL} xSLA}{T_{SA}}$$

For our analysis, initial values of all stocks are four and the desired stock is set to five. Acquisition delay (T_{AD}) and T_{SA} are both equal to 4. The system is at equilibrium when the desired stock is four. We perturb the system at time zero with a sudden change at desired level.

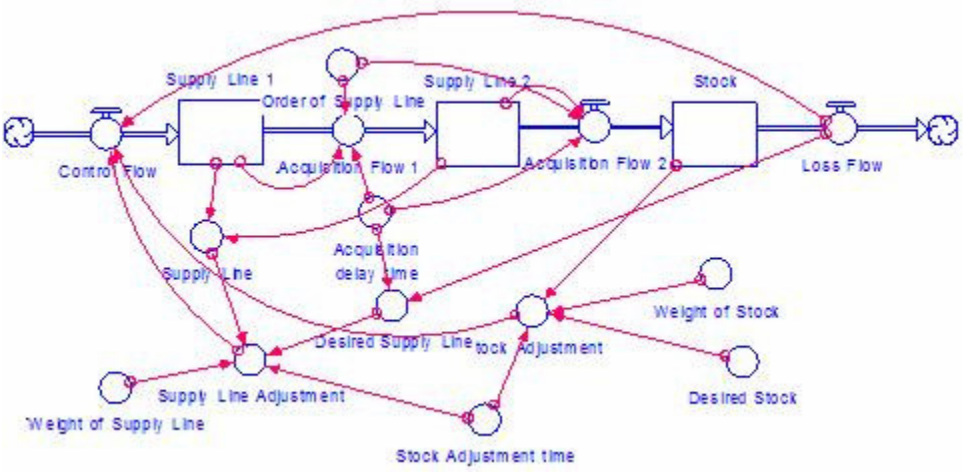

Figure 6: Stock-Flow diagram for Stock Management Model

Fitness Function for the Stock Management Model

For this model, we have chosen W_S and W_{SL} as the search parameters for the algorithm. Generally, the desired behavior in any stock management model is for the stock to reach desired level quickly, without oscillation. In the fitness function, these

were defined as settling time (t_s), overshoot, and steady state error (e_{ss}). Settling time is defined as the time the stock level goes into a band and never goes out. The band is an area defined between two points which are obtained by adding and subtracting a given percentage (p) of the settled level (S^s) from the S^s. Hence, the middle point is the S^s. The t_s is divided by the time horizon to normalize its effect on fitness function to a number between zero and one. If the system fails to stabilize by the time horizon, the time horizon is taken as the t_s. Overshoot is defined by subtracting S^s from the maximum level the Stock reaches. Steady state error is simply the difference between the last value obtained at the end of time horizon and desired level (S^*). The fitness value returned by the function is defined as weighted sum of t_s, overshoot, and e_{ss}. These weights define the importance of each variable for the policy maker.

Results

The time horizon for the search is set to 100. The parameters, W_S and W_{SL}, are set to have boundaries between zero and one since setting a limit that is higher than one for both of them would also mean decreasing T_{SA}. The percentage p of the band is set to 1%, meaning that the band is between 1.01* S^s and 0.99* S^s. Initially, the weights of t_s (Pts), overshoot (Po), and e_{ss} (Pess) were all set to one. The algorithm is again run three times. The results of the algorithm do not give the value one for both W_S and W_{SL}. In all cases, while W_S is very close to one, W_{SL} is always smaller than W_S. This outcome shows that improvement in settling time is greater than the increase in overshoot considering Pts, Po, and Pess.

Table 4: Summary of best policy parameters, their fitness values, fitness values of (W_S, W_{SL}) = (1, 1)

	Best three individuals	1			2			3			General policy		
(Pts, Po, Pess)= (1,1,1)	Fitness Value	0.1444			0.1452			0.1443			0.17427		
	(SA, SLA)	0.9963 0.7237			0.982964 0.68291			0.99586 0.7103			1 1		
		t_s = 13.55	O= 0.0089	e_{ss} = 0	t_s = 13.325	O= 0.012	e_{ss} = 0	t_s = 13.4	O= 0.01	e_{ss} = 0	t_s = 17.427	O= 0	e_{ss} = 0
(Pts, Po, Pess)= (10,1,1)													
	Fitness Value	0.15901			0.160242			0.158869			0.17427		
	(SA, SLA)	0.9946 0.8552			0.976355625 0.83538			0.9882 0.851			1 1		
		t_s = 15.36	O= 0.000538	e_{ss} = 0	t_s = 15.497	O = $5.274*10^{-4}$	e_{ss} = 0	t_s = 15.408	O= 0.000479	e_{ss} = 0	t_s = 17.427	O= 0	e_{ss} = 0
(Pts, Po, Pess)= (100,1,1)													
	Fitness Value	0.1628			0.1659			0.1653			0.17427		
	(SA, SLA)	0.9999 0.911522			0.9891 0.92157			0.975562 0.894278221			1 1		
		t_s = 16.082	O= 0.00002	e_{ss} = 0	t_s = 16.5846	O = $3.53x10^{-7}$	e_{ss} = 0	t_s = 16.4837	O = $4.75x10^{-7}$	e_{ss} = 0	t_s = 17.427	O= 0	e_{ss} = 0

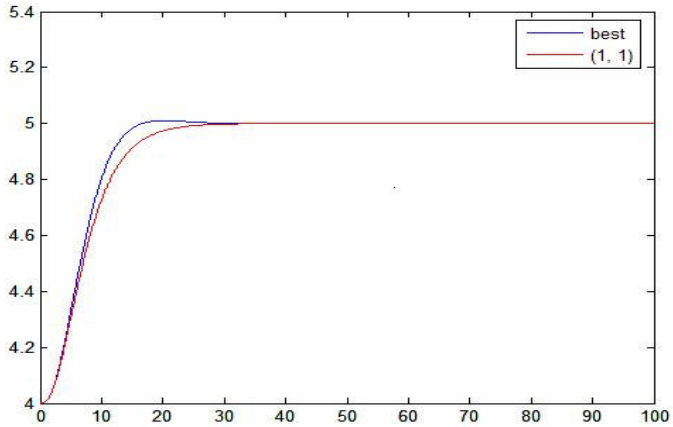

Figure 7: Comparison of best set of parameters (run 3) and (W_S, W_{SL}) = (1, 1) with (Pts, Po, Pess)= (1,1,1).

Secondly, we consider a fitness function where the decision maker penalizes the overshoot ten times more than t_s, and e_{ss}, meaning (Pts, Po, Pess)=(10, 1, 1). The results show that although the ratio between W_S, and W_{SL} decreased, it still does not equal to one. It can be speculated that as Pts is increased the optimal ratio of W_S, and W_{SL} approaches to one. However, when Pts is increased too much the simulation errors become important since a slight overshoot affects the fitness function when Pts is high. If the aim is to find the set of parameters which completely eliminate overshoot, the band for the settling time should also be made smaller so that even small overshoot values cause the system to trespass the boundaries of the band more than once. In this case, the overshoot also affects the settling time. Even then the Pts should also be considerably increased which can cause numeric errors. Table 4 also gives the results of the algorithm run with fitness weights (Pts, Po, Pess)=(100, 1, 1). The results coincide with the analysis made for the previous case.

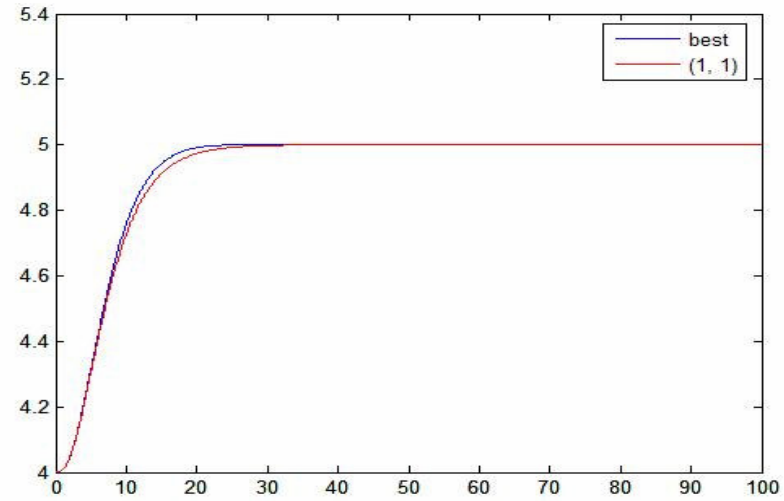

Figure 8: Comparison of best set of parameters (run 3) and (W_S, W_{SL}) = (1, 1) with (Pts, Po, Pess)= (10,1,1)

Further Analysis of the Stock Management Model

The parameter search indicates that it is possible to obtain improvements in settling time with allowing very small overshoot. In this section, the trade off between overshoot and settling time is analyzed. The effect of T_{SA} on the system is included in the analysis.

Table 5: Overshoot and settling time with different T_{SA} values.

W_S	W_{SL}	T_{SA}	ts	Overshoot
1	1	4	17.43	0
1	1	4/3	11.3	0
1	1	1/2	10.1	0
1	1	1/10	9.94	0

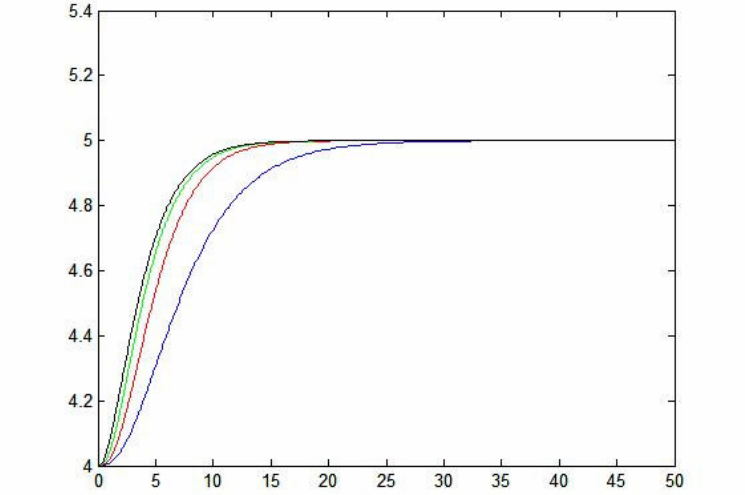

Figure 9: Behavior of Stock under the parameter values given on Table 5

Table 6: Overshoot and settling time with different T_{SA} values.

W_S	W_{SL}	T_{SA}	ts	Overshoot
0.75	0.5	4	16.1	0.0043
0.75	0.5	2	10.09	0.0365
0.75	0.5	1	11.42	0.0523
0.75	0.5	2/3	6.66	0.0479
0.75	0.5	1/4	5.88	0.0256
0.75	0.5	1/10	5.75	0.0164

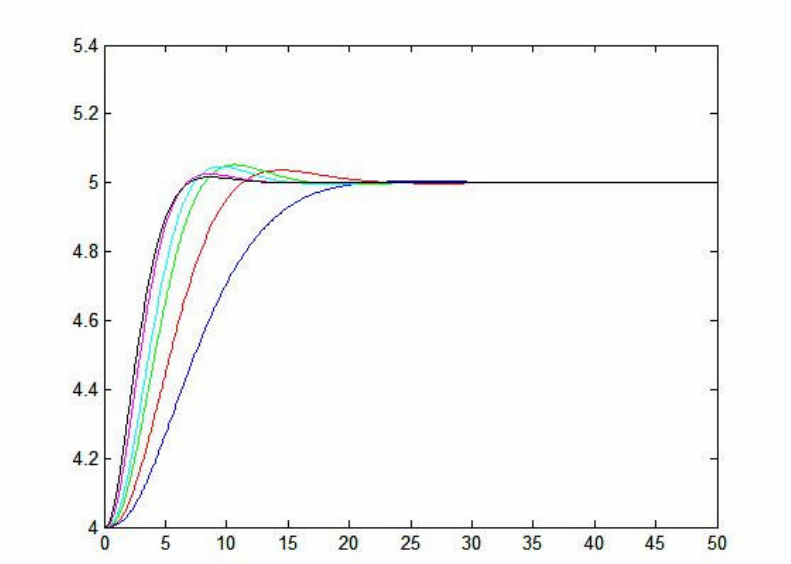

Figure 10: Behavior of Stock under the parameter values given on Table 6.

The analysis shows that the increase in W_S/W_{SL} ratio causes quicker responses but higher overshoot. However, when the overshoot or the ratio is too high, the stock passes the boundary twice rather than once. In our case, when the overshoot is greater than 0.05, crosses the band more than once. Hence, the settling time is longer (see Table 7).

Table 7: Overshoot and settling time with different T_{SA} values.

W_S	W_{SL}	T_{SA}	ts	Overshoot
1	0.5	4	11.4	0.0492
1	0.5	2	14.93	0.1139
1	0.5	1	11.6	0.139
1	0.5	2/3	10.27	0.1317
1	0.5	1/4	8.16	0.0855
1	0.5	1/10	7.14	0.0587

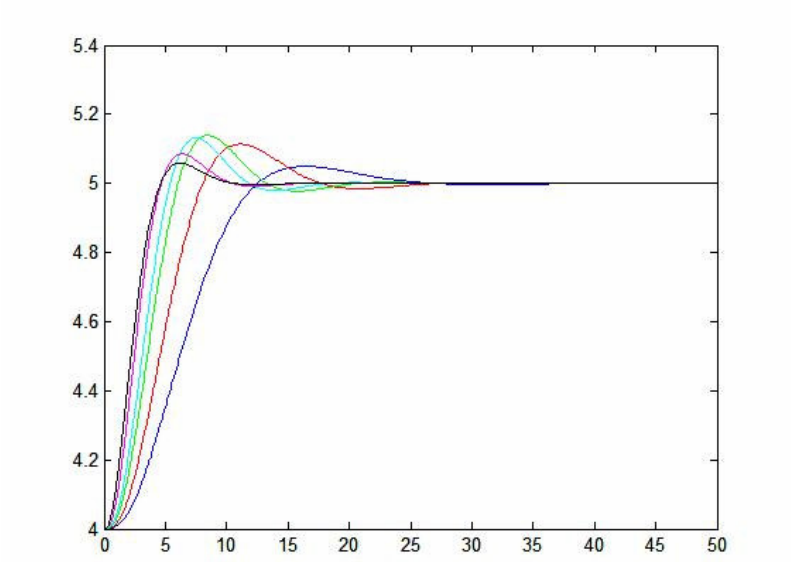

Figure 11: Behavior of Stock under the parameter values given on Table 7

The results indicate that, decreasing T_{SA} or increasing the weights W_S, and W_{SL} over one cause the system to respond faster (see Figures 9, 10, and 11). In cases with overshoot, decrease of T_{SA} while keeping the weights same causes overshoot to increase until a certain value is reached. From that point on, the decrease of T_{SA} causes overshoot to decrease together with settling time (see Figures 10, and 11).

Table 8: Change in overshoot and t_s with altering W_S/W_{SL}

W_S	W_{SL}	ts	Overshoot	Change in t_s w.r.t Ws, WsL=(1,1)
1	1	17.43	0%	
1	0.8	14.46	0.29%	-17.04%
1	0.6666	12.95	1.65%	-25.70%
1	0.5	11.4	4.92%	-34.60%

Table 8 displays that, without decreasing T_{SA} substantial increase in settling time can be achieved with allowing very small overshoot by adjusting W_S, and W_{SL}.

5) Conclusion

Policy parameter search is applied to an electric circuit model and a generic Stock Management model by using Genetic Algorithms. The search is simulation based. The paper points to the critical role of time horizon in simulation based search. The dynamic models are assumed to have multiple objectives. This paper emphasizes the weights of different goals on the objective function and shows that results may vary with respect to these weights. The algorithm simulates the models and evaluates the fitness function 10000 times in less than five minutes and obtains satisfactory 'optimal' results.

The second part of the paper concentrates on the policy parameters of the generic Stock Management model. This part is a direct result of information obtained from parameter search and comparison of this information with existing suggested policies. The results show that there can be better policies in terms of decreasing settling time for a

small cost of overshoot. Another result shows that decreasing stock adjustment time may cause overshoot to increase in certain cases.

A natural extension of this paper is to test the genetic algorithm on several other types of dynamic models and suitable, fitness functions. It may be necessary to carry out further analysis on these models, similar to the Stock Management model. The ultimate aim is to build confidence in the search algorithm, and use it as a support tool to gain insight about the system during model building, sensitivity and policy analysis.

References

Burns, J., and D. Malone. 1974. Optimization Techniques Applied to the Forrester Model of the World. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-4 (2): 164-171.

Coyle, R.G. 1977. *Management System Dynamics.* New York: Wiley.

Coyle, R.G. 1985. The use of optimization methods for policy design in a system dynamics model. *System Dynamics Review*. 1(1): 81-91.

Coyle, G. 2000. Qualitative and quantitative modeling in system dynamics: some research questions. *System Dynamics Review* 16(3): 225-244.

Dangerfield, B. and C. Roberts. 1999. Optimization as a statistical estimation tool: An example in estimating the AIDS treatment-free incubation period distribution. *System Dynamics Review.* 15(3): 273-291.

Duggan, Jim. 2007. Using system dynamics and multi objective optimization to support policy analysis for complex systems. In *Understanding Complex Systems,* 59–81. Springer Berlin/Heidelberg.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, Addison-Wesley.

Graham, A. K. and C.A. Ariza. 2003. Dynamic hard and strategic questions: using optimization to answer a marketing resource allocation question. *System Dynamics Review.* 19(1): 27-46.

Grossmann, B. 2002. Policy Optimization in Dynamic Models with Genetic Algorithms. International System Dynamics Conference, Palermo.

Holland, J.H. 1975. *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Keloharju, R. 1982. *Relativity Dynamics.* Series. A-40. Helsinki: School of Economics.

Keloharju, R. and E. F. Wolstenholme 1988. The basic concepts of system dynamics optimization. *Systems Practice.* 1(1): 65-86.

Miller, J. H. 1998. Active Nonlinear Tests (ANT) of Complex Simulation Models. *Management Science.* 44(6): 820-830.

McSharry, P. E. 2004. Optimisation of system dynamics models using genetic algorithms. *World Bank Report,* May 11.

Mohapatra, P.K.J., and S. K. Sharma. 1985. Synthetic design of policy decisions in system dynamic models: a modal control theoretical approach. *System Dynamics Review*. 1(1): 63-80.

Özveren, C.M., and J.D. Sterman. 1989. Control theory heuristics for improving the behavior of economic models. *System Dynamics Review*. 5(2): 130-147.

Richardson, G. P. 1999. Reflections for the future of system dynamics. *Journal of the Operational Research Society.* 50(4): 440-449.

Sterman JD. 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Irwin McGraw-Hill: Boston, MA.

Yaşarcan, H. 2003. Feedbacks, Delays And Non-linearities in Decision Structures. Industrial Engineering Dept. Istanbul, Bogazici University. **PhD**.

Yücel, G., and Y. Barlas. 2007. Pattern Based System Design/Optimization. International System Dynamics Conference, Boston.

Wolstenholme, E. F. and A.S. Al-Alusi. 1987. System dynamics and heuristic optimisation in defence analysis. *System Dynamics Review*. 3(2): 102-115.

APPENDIX
Model Equations from STELLA®

Stock(t) = Stock(t - dt) + (Acquisition_Flow_2 - Loss_Flow) * dt
INIT Stock = 4
Acquisition_Flow_2 = Supply_Line_2*Order_of_Supply_Line/(Acquisition_delay_time)
Loss_Flow = 2

Supply_Line_1(t) = Supply_Line_1(t - dt) + (Control_Flow - Acquisition_Flow_1) * dt
INIT Supply_Line_1 = 4
Control_Flow = Loss_Flow+1*Stock_Adjustment+1*Supply_Line_Adjustment
Acquisition_Flow_1 = Order_of_Supply_Line*Supply_Line_1/(Acquisition_delay_time)

Supply_Line_2(t) = Supply_Line_2(t - dt) + (Acquisition_Flow_1 - Acquisition_Flow_2) * dt
INIT Supply_Line_2 = 4
Acquisition_Flow_1 = Order_of_Supply_Line*Supply_Line_1/(Acquisition_delay_time)
Acquisition_Flow_2 = Supply_Line_2*Order_of_Supply_Line/(Acquisition_delay_time)

Acquisition_delay_time = 4
Desired_Stock = 5
Desired_Supply_Line = Loss_Flow*Acquisition_delay_time
Order_of_Supply_Line = 2
Stock_Adjustment = (Desired_Stock-Stock)/Stock_Adjustment_time
Stock_Adjustment_time = 4
Supply_Line = Supply_Line_1+Supply_Line_2
Supply_Line_Adjustment=Weight_of_Supply_Line*(Desired_Supply_Line-
Supply_Line)/Stock_Adjustment_time
Weight_of_Supply_Line = 1