Improving programming skills of Mechanical Engineering students by teaching in C# multi-objective optimizations methods

Adrian Florea[1,*], and *Ileana Ioana* Cofaru[1]

[1]'Lucian Blaga' University of Sibiu, Computer Science and Electrical Engineering Department, 550025, 4 Emil Cioran Street, Sibiu, Romania

Abstract. Designing an optimized suspension system that meet the main functions of comfort, safety and handling on poor quality roads is a goal for researchers. This paper represents a software development guide for designers of suspension systems with less programming skills that will enable them to implement their own optimization methods that improve traditional methods by using their domain knowledge.

1 Introduction

The teams dedicated for research, development and hardware-software deployment from industrial companies tend to become more heterogeneous, combining researchers with mechanical, electronic and informatics skills. Thus, possession of basic knowledge of programming by mechanical engineers is essential. Regarding the issue of suspension system optimization, in most approaches it is analysed just from mechanical viewpoint or mechanical accompanied by vision of transport and road vehicles engineers. In this case, the non IT specialists apply a series of software tools specific to computer science domain for simulation and optimization without regard or understand the particularity of optimization method, in whose kernel you could then intervene to customize them to bring a real scientific knowledge gain. In this paper we start the suspension optimization from the perspective of optimization methods, doing a research analysis from computers scientist viewpoint who wants to understand the engineering problem, to apply then a mathematical model and then learn to develop its own optimization tool, very parameterized, flexible and extensible.

Therefore, the methodology for modelling, simulation and optimization is tested on a mechanical problem, namely the passive suspension system of a Half-Car model (HCM) [1] with 4 Degrees of Freedom (4 DOF) because the suspension model design represents a challenging task for the vehicle designers due to multiple control parameters, complex and conflicting objectives, the system running under several existing constraints. In our previous work we analysed the quarter-car model (QCM) with two-degrees-of-freedom (2DOF) [2-3] in order to find optimal parameters of stability and to keep high standard of ride comfort under different exploiting conditions, minimizing the discomfort during

* Corresponding author: adrian.florea@ulbsibiu.ro

movement. Although the quarter car model is simple and widely used for dynamic performance analysis, it fails to capture the more realistic results of actual behaviour of the vehicle. Thus, in this work we analysed HCM because captures important characteristics of full car model. Although the passive suspension systems have some inherent constraints we consider them in our analysis due to their provided trade-off between the ride comfort level and vehicle stability at low cost. The passive systems consists in elastic (spring) and dissipative (damper) elements, their dynamic behaviour being described by the characteristics of the items referred. To have a good performance of handling and stability at different driving conditions (load, road profile, speed), the spring stiffness of suspension system should be very high, which determines the passenger to experience uncomfortable driving feeling. Conversely, if we expect that the suspension system to ensure a high level of ride quality protecting the vehicle and driver from harmful stresses, the spring stiffness should be decreased, which will determine the instability of vehicle when is drove.

This work can be seen also as a useful learning tool for students of Industrial (Mechanical) Engineering specializations that successfully combines in their training knowledge from different scientific fields such as: Computer Science (Artificial Intelligence, Programming Languages, and Distributed Computing), Mathematics (Differential Equations), Transportation Engineering (Vehicle Design), Mechanical Engineering (Vibrations) to solve a real problem, even critical in Romania, characterized by a developing infrastructure and poor quality roads.

The organization of the rest of this paper is as follows. In section 2 we review some related work solutions applied in suspensions optimization, whereas section 3 describes the dynamic suspension system in order to software implement. Section 4 presents the mathematical model, provides programming details on solving the system of differential equations by "*Runge-Kutta*" method, exhibits the application GUI and illustrates simulation results. Finally, section 5 suggests directions for future work and concludes the paper.

2 Related work

In [1] the authors investigate the vibration and ride features of a passive HCM model with 5 DOF vehicle suspension system excited by a sinusoidal surface. Their model parameters are kept fixed and made simulations in Simulink and Matlab for determining the amplitude of vibrations. The main differences between their work and this paper consist in the software vision of optimization methods that we performed (C# implementation of NSGA-II algorithm), the number of degrees of freedom and the random road profile that we tested.

In [2] the authors present the mathematical model of QCM suspension's system from OPEL cars and in [3] the authors apply multi-objective optimization techniques for designing the quarter car model on random and sinusoidal road profiles. Unlike these two previous works, in this paper we changed the model of suspension (analysing HCM) and also we have added two more degrees of freedom that expanded the differential equations system of order two.

Studying the vibrations induced by surface irregularities in road pavements in [4] the authors make some Matlab tests where they have shown that the road profile is a combination of a large number of longer and shorter holes of different amplitudes. Based on their equations we implemented the random road profiles in C#.

In [5] the authors present an optimization of a QCM 4-DOF vehicle's human with seat suspension system using weighted sum genetic algorithms (with fixed weights) to determine suspension parameters. Unlike their work we apply on HCM 4-DOF Pareto multi-objective optimization algorithms.

3 The Half-Car suspension system

The Half-Car model is a basic model used to simulate the vehicle's suspension system performance by investigating the dynamic response of cars when running on different surfaces. HCM includes an independent Front and Rear vertical suspension (see Figure 1). The HCM name is due to the fact that total vehicle mass is divided equally on two and might be seen as combination of two quarter car models. The spring has significant role in establishing the suspension function of the system, while the damper is used to dissipate kinetic energy of spring and supply a damping function. We consider that there is no effect of eccentricity of driver seat when located on the centre of gravity and the tires are always touching the ground.

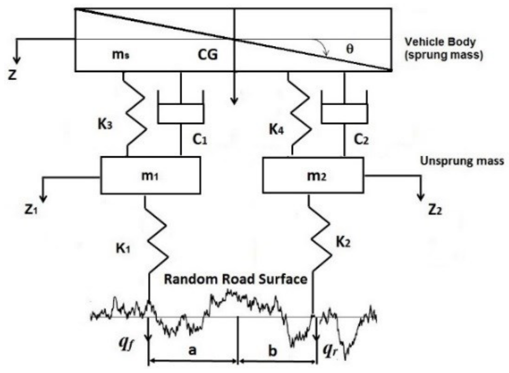


Fig. 1. The analytical model of half-car suspension system

We made the following notations:

- m_1, m_2, m_s: (in the following order) mass of the Front tire wheel, Rear tire wheel and the vehicle body (sprung mass).
- θ: pitching motion angle of car body.
- I_x: pitch moment of inertia.
- a: lateral distance between the body centroid and Front tire wheel.
- b: lateral distance between the body centroid and Rear tire wheel.
- c_1, c_2: Damping coefficient of the Front suspension and Rear suspension.
- k_1, k_2: Vertical spring stiffness of the Front tire, Rear tire.
- k_3, k_4: Vertical spring stiffness of the sprung body Front suspension and Rear suspension.
- Z_1, Z_2, Z_s: (in the following order) bounce motion of the Front tire wheel, Rear tire wheel and the vehicle body.

The four degree of freedom considered in this work are the three displacements: of car body, Front and Rear tire respectively, and the pitching angle of car body.

4 Modelling and software implementation of the suspension system

4.1 The mathematical model

In this section we described the mathematical model followed by C# software implementation of the half-car suspension model with four degrees-of-freedom (4-DOF) tested on a generic car. The purpose of modelling is to obtain a state space representation of the vehicle model. The mathematical model used for describing and studying of the suspension is provided by a system of four differential equations of order 2.

$$M \cdot \ddot{Z} + C \cdot \dot{Z} + K \cdot Z = F \cdot Q \tag{1}$$

In equation (1) M, C and K respectively, are mass matrix, damping matrix and stiffness matrix. The forces that are transmitted from the running unevenness surface are exhibited through Forces matrix (F).

Besides the aforementioned matrixes the system has as input the road excitation matrix Q, where $Q = (q_f \quad q_{fr})^T$ (the height of potholes on front and rear wheel) and as outputs the generalized displacement vector Z, where $Z = (Z_1 \ Z_2 \ Z_s \quad \theta)^T$.

$$M = \begin{pmatrix} m_1 & 0 & 0 & 0 \\ 0 & m_2 & 0 & 0 \\ 0 & 0 & m_s & 0 \\ 0 & 0 & 0 & I_x \end{pmatrix}; C = \begin{pmatrix} c_1 & 0 & -c_1 & -ac_1 \\ 0 & c_2 & -c_2 & bc_2 \\ -c_1 & -c_2 & c_1 + c_2 & ac_1 - bc_2 \\ -ac_1 & bc_2 & ac_1 - bc_2 & a^2c_1 + b^2c_2 \end{pmatrix};$$

$$K = \begin{pmatrix} k_1 + k_3 & 0 & -k_3 & -ak_3 \\ 0 & k_2 + k_4 & -k_4 & bk_4 \\ -k_3 & -k_4 & k_3 + k_4 & ak_3 - bk_4 \\ -ak_3 & bk_4 & ak_3 - bk_4 & a^2k_3 + b^2k_4 \end{pmatrix}; F = \begin{pmatrix} k_1 & 0 \\ 0 & k_2 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \tag{2}$$

The governing equations are:

$$\begin{cases} m_1 \cdot \ddot{Z}_1 + c_1 \cdot (\dot{Z}_1 - \dot{Z} - a \cdot \dot{\theta}) + k_1 \cdot Z_1 + k_3 \cdot (Z_1 - Z - a \cdot \theta) = k_1 \cdot q_f \\ m_2 \cdot \ddot{Z}_2 + c_2 \cdot \left(\dot{Z}_2 - \dot{Z} + b \cdot \dot{\theta}\right) + k_2 \cdot Z_2 + k_4 \cdot (Z_2 - Z + b \cdot \theta) = k_2 \cdot q_r \\ m_s \cdot \ddot{Z} + c_1 \cdot \left(\dot{Z} - \dot{Z}_1 + a \cdot \dot{\theta}\right) + c_2 \cdot \left(\dot{Z} - \dot{Z}_2 - b \cdot \dot{\theta}\right) + \ k_3 \cdot (Z - Z_1 + a \cdot \theta) \ + \\ \qquad + k_4 \cdot (Z - Z_2 - b \cdot \theta) = 0 \\ I_x \cdot \ddot{\theta} + c_1 \cdot \left(\dot{Z} - \dot{Z}_1 + a \cdot \dot{\theta}\right) \cdot a - c_2 \cdot \left(\dot{Z} - \dot{Z}_2 - b \cdot \dot{\theta}\right) \cdot b + \ + k_3 \cdot (Z - Z_1 + a \cdot \theta) \cdot a - \\ \qquad - \ k_4 \cdot (Z - Z_2 - b \cdot \theta) \cdot b = 0 \end{cases} \tag{3}$$

First two equations from system (2) describe the bounce motion of Front tire wheel and Rear tire wheel, respectively. The third equation illustrates the bounce motion of body vehicle while the last one shows the pitch motion of car. The system (2) will be transformed into 8 differential equations of first order by introducing new unknown functions. The next step aims solving the 8 kinematic equations of motion using C# DotNumerics.ODE [6] package (see section 4.2) by "*Runge-Kutta*" numerical time integration method.

4.2 The software design

The developed software application was implemented in Microsoft Visual Studio 2015 (C#), .NET Framework 4.5, using DotNumerics a numerical library for .NET [6, 7]. The class diagram of the project's solution is presented in Figure 2.

The class that contains the evaluation of the vehicle input variables and uses the "*Runge Kutta*" method [6] to solve the differential equations system in order to get the output variables of the system is the *SuspensionModelEvaluation* class. The principal method here is the *Evaluate* method, which receives as input parameters a *SuspensionDesignSolution* object and a *RandomRoadGenerator* object. First we briefly described these two classes and then we explain how *Evaluate* method works. Finally, it sets the displacement of the Front tire wheel, the displacement of Rear tire wheel and the body mass acceleration.

- ➢ *SuspensionDesignSolution* class contains the properties that characterize a solution:
- The design variables that we wanted to optimize (vertical stiffness of Front suspension (k_3), vertical stiffness of Rear suspension (k_4), vertical spring stiffness of the Front tire (k_1) and Rear tire (k_2), vertical damping coefficient of the Front suspension (c_1), vertical coefficient damping of Rear suspension (c_2)) as an array of type *DesignVariable*.
- The output variables that we want to optimize, such as *DisplacementSprungMassFrontTire* (Z_1), *DisplacementSprungMassRearTire* (Z_2), *bodyMassAcceleration* and their root mean square (RMS) version: *RmsDisplacementSprungMassFrontTire*, *RmsDisplacementSprungMassRearTire* and *RmsBodyAcceleration*.
- The *front number* (from all Pareto fronts) in which a solution is places, after evaluation of the results within NSGA-II algorithm.
- The *crowding distance* that is specific to the solution, also needed in NSGA-II.

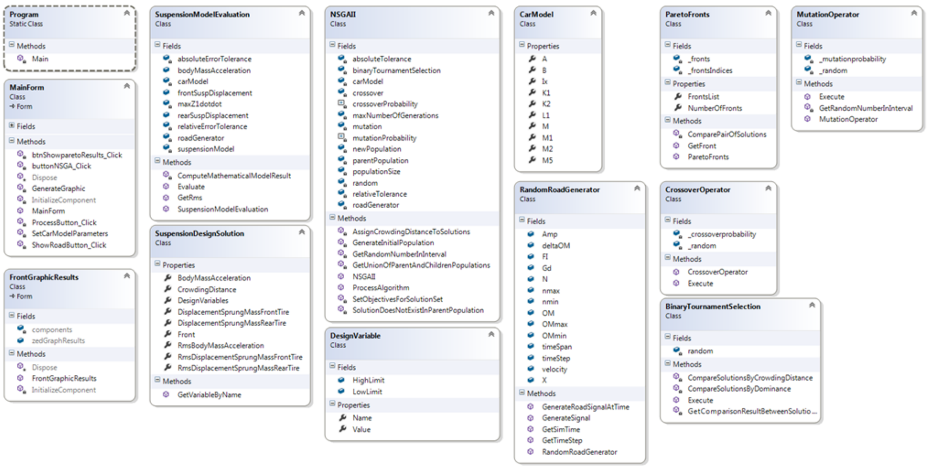


Fig. 2. The application classes

- ➢ *RandomRoadGenerator* is a class used for generating a random road signal on a specific length and with a certain velocity. The constructor gets a few parameters: the length of the road, the signal rate (step), the degree of roughness, the minimal and the maximal wavelength and the velocity.
- The method *GenerateSignal* calculates the road signal and returns an array of double values that represent the heights of the road on the specified length.
- Method *GenerateRoadSignalAtTime* is very important because it returns the road signal at a specific time. It is very useful when calculating the displacements and accelerations of vehicle model elements at a certain moment.
- Using *ZedGraph* functions package we realized 2D graphical representation of generated artificial road profiles. The *GenerateGraphic* method from principal *MainForm* class has the following parameters: a *ZedGraphControl* object, the title of the graphic, the X axis string, the Y axis string, the signal and the signal step.
- ➢ In the *Evaluate* method of the *SuspensionModelEvaluation* class we instantiate an *OdeExplicitRungeKutta45* object from DotNumerics library [6], establish the initial values of time (t0) and signal (y0), set the sampling interval of time array (t) and then with the method *Solve*(y0,t) we get as result a matrix that has the number of rows equal to the number of sampling values. In each row, there is an array of values that correspond to the outputs of the system that was solved with "*Runge Kutta*" method. The displacements and acceleration that we have interest in are saved in arrays and after

iterating through all the samples, we calculate the RMS values of the outputs, which are saved in the *RmsDisplacementSprungMassFrontTire*, *RmsDisplacementSprungMassRearTire* and *RmsBodyAcceleration* properties of the solution of type *SuspensionDesignSolution*.

- The method *ComputeMathematicalModelResult* is used in the solving of differential equations system and it contains the real differential equations. It is sent as a parameter to OdeExplicitRungeKutta45 method.

4.2.1 Multi-objective optimization. Implementation of NSGA-II algorithm

Modified non-dominated sorting genetic algorithm (NSGA-II) [8] has been used for multi-objective optimization of a 4-DOF vehicle model which excited with random road profile. The conflicting objective functions considered were vertical acceleration of sprung mass, relative displacement between body mass and Front tire wheel and relative displacement between body mass and Rear tire wheel. NSGA-II is one of the most used algorithms in experiments due to its capacity of generating good solutions regardless of the problem [3]. Because the pseudo-code of NSGA-II algorithm is presented in [3, 8] here we are limiting to briefly describe how the main algorithm works:

- We have a parent population P_0 of chromosomes (design solutions) that is randomly initialized. The algorithm applies genetic operators (Binary tournament selection, crossover, and mutation operators) in order to create a child population Q_0 of size N. The old and new populations are combined together, then a sorting is done using non-dominance as criteria. Each solution is assigned a fitness equal to its *rank* (non-domination level where 1 is the best level). Thus, minimization of fitness is assumed. A combined population $R_t = P_t \cup Q_t$ is formed, where t is the generation number. The population R_t will be of size $2N$. Then, the population R_t is sorted according to non-domination. The new parent population P_{t+1} is formed by adding solutions from the first fronts till the size exceeds N. The ranks and *crowding distance* (density of individuals surrounding a particular one) are used to guide the selection. An individual is better, if it has a smaller rank, or in case of equality, the one with the bigger crowding distance.
- In order to sort a population of size N according to the level of non-domination, each solution must be compared with every other solution in the population to find if it is dominated. This process is continued to find the members of the first non-dominated class for all population members. At this stage, all individuals in the first non-dominated front are found. In order to find the individuals in the next front, the solutions of the first front are temporarily discounted and the above procedure is repeated.
- First, for each solution we calculate two entities: (i) n_i, the number of solutions which dominate the i solution and (ii) S_i, a set of solutions which the i solution dominates. We identify all those points which have $n_i = 0$ and put them in a list. We call F_1 the current front.
- Then, for each solution in the current front we visit each member (j) in its set and reduce its n_j value by one. In doing so, if for any member the count becomes zero, we put it in a separate list H. When all members of the current front have been checked, we declare the members in the list F_1 as members of the first front. We then continue this process using the newly identified front H as our current front.
- To get an estimate of the density of solutions surrounding a particular point in the population we take the average distance of the two points on either side of this point along each of the objectives. This quantity distance serves as an estimate of the size of the largest cuboid enclosing the point without including any other point in the population (the crowding distance).

The implementation of the algorithm was done in *NSGAII* class. In the constructor we pass some parameters which are: the maximum number of generations of genetic algorithm, the population size, the road signal, the car parameters (as object of *CarModel* class) and the relative and absolute tolerance used in the evaluation of a solution. The most important method of NSGAII class is the *ProcessAlgorithm* method. It contains the main loop of the algorithm. The *GenerateInitialPopulation* method randomly initialized the first parent population. Then, each possible solution from the current parent population is evaluated to get the objective functions in method *SetObjectivesForSolutionSet*, which gets as input parameter the list of *SuspensionDesignSolution*. After this evaluation comes a *while* loop in which at each iteration from the current population is generated a new population, till the maximum number of iterations is reached. In this loop, the following actions are made:

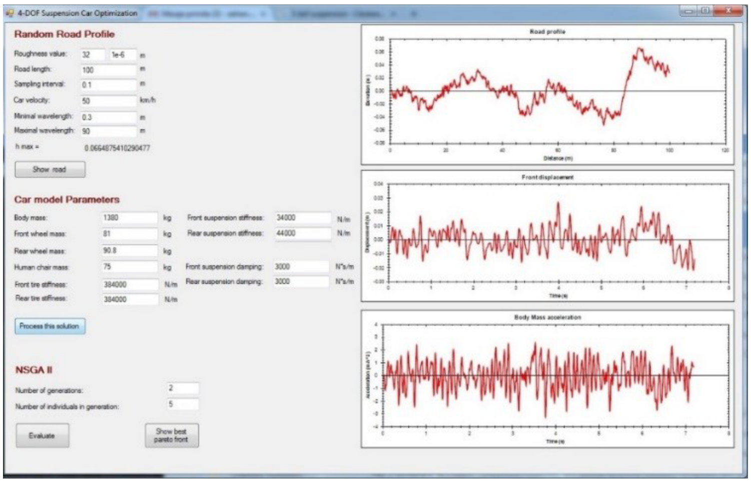
- A new offspring population is generated from the parent population. They have the same size. Pairs of two parents are selected by the binary tournament selection mechanism (from *BinaryTournamentSelection* class, in which in the *Execute* method is done the random selection of two solutions from population, then these solutions are compared by their values of objective functions and crowding distance and the best solution is returned). Then, recombination with one crossover point is done in *Execute* method of *CrossoverOperator* class. After this, mutation is applied to both offspring resulted from crossover. In method *Execute* from *MutationOperator* class is done the mutation by randomly changing the value of one design variable.
- Offspring population is evaluated from the perspective of the 3 objective functions.
- A new population is generated by combining together the parent and children populations. This one has twice the size of the parent / children population.
- The new population is split in Pareto fronts. Fronts are obtained mainly by getting for each solution the number of solutions which dominate it and the set of solutions which the solution dominates. The implementation is made in the constructor of the *ParetoFronts* class.
- Each front is taken into consideration iteratively, from the first to the last one. Solutions from the fronts are copied in a new population one by one, front by front, till the new population has the size of the initial parent population. Each solution of the fronts that is added to the population is assigned a crowding distance in this moment.
- Best solutions from the resulting population (the ones from the first front) are saved in a list of solutions, which will be returned by the *Process* method.

4.3 The user guide and some simulation results

To run the optimization algorithm, a few steps are needed before to initialize the random road profile and the car model parameters:

- In the interface, the parameters for road profile can be selected and a graphic of the road is generated when pressing the “Show road” button.
- Then, the car model parameters must be set. The body mass, the Front wheel mass, the Rear wheel mass, the Front tire stiffness and the Rear tire stiffness can be edited from the interface. Others are hardcoded: pitch moment of inertia, distance between the body centroid and Front tire and distance between the body centroid and Rear tire.

It is possible to evaluate a single solution pushing the “Process this solution” button (see Figure 3) by editing also the specific design variables: vertical spring stiffness of the sprung body Front suspension and Rear suspension, damping coefficient of the Front suspension and Rear suspension.



4-DOF Suspension Car Optimization

Random Road Profile

Roughness value: 32 1e-6 m
Road length: 100 m
Sampling interval: 0.1 m
Car velocity: 50 km/h
Minimal wavelength: 0.3 m
Maximal wavelength: 90 m
h max = 0.0664875410290477

Car model Parameters

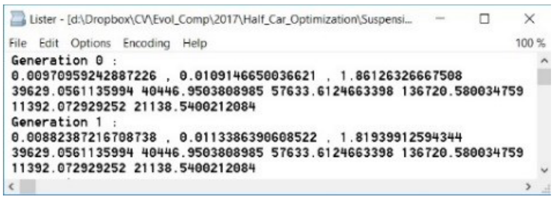
Body mass: 1380 kg
Front wheel mass: 81 kg
Rear wheel mass: 90.8 kg
Human chair mass: 75 kg
Front tire stiffness: 384000 N/m
Rear tire stiffness: 384000 N/m
Front suspension stiffness: 34000 N/m
Rear suspension stiffness: 44000 N/m
Front suspension damping: 3000 N*s/m
Rear suspension damping: 3000 N*s/m

NSGA II

Number of generations: 2
Number of individuals in generation: 5

Fig. 3. Application's GUI: Random road profile parameters and simulation results

These values along with the car model parameters and random road signal are passed to the *Evaluate* method of *SuspensionModelEvaluation* class, and the graphics of the evolution of objective functions in time can be seen in the right part of the interface. Then, to apply NSGAII algorithm, the number of generations and the population size must be set and "*Evaluate*" button pressed. The results are written in a text file (see Figure 4). In this text file the best solutions from each generation are showed in this way: in the first row the values of the three objective functions and on the second row are written the values of the design variables that generated those results.



Lister - [d:\Dropbox\CV\Evol_Comp\2017\Half_Car_Optimization\Suspensi...
File Edit Options Encoding Help
100 %

```
Generation 0 :
0.00970959242887226 , 0.0109146650036621 , 1.86126326667508
39629.0561135994 40446.9503808985 57633.6124663398 136720.580034759
11392.072929252 21138.5400212084
Generation 1 :
0.00882387216708738 , 0.0113386390608522 , 1.81939912594344
39629.0561135994 40446.9503808985 57633.6124663398 136720.580034759
11392.072929252 21138.5400212084
```

Fig. 4. NSGA-II simulation results

5 Conclusions and further work

This paper accomplished two-fold objectives: first, describe in details some C# functions useful for non-IT specialists to develop (custom) optimization methods of suspension systems and second, is concerned to improve 4-DOF HCM suspension's system by optimizing the stiffness and damping coefficients with the help of multi-objective methods (NSGA-II) in order to increase the ride comfort. Our preliminary results have shown that increasing generation number of genetic algorithm lead to obtaining better solutions.

References

1. M. S. Rahman, K. Muhammad, G. Kibria, Procedia Engineering **90,** 96-102 (2014)
2. L. Roman, A. Florea, I.I. Cofaru, Annals of the Oradea University Fascicle of Management and Technological Engineering **23** (3) 94-99 (2014)
3. A. Florea, I.I. Cofaru, L. Roman, N. Cofaru, Journal of Digital Information Management, **14**(6), 351-367 (2016)

4. M. Agostinacchio, D. Ciampa, S. Olita, *European Transport Research Review*, (Springer Berlin Heidelberg,2013)
5. S. Badran, A. Salah, Research Bulletin of Jordan, **2**, 42-51 (2012)
6. Santiago-Castillo Jose Antonio (2015) *DotNumerics*, http://www.dotnumerics.com/NumericalLibraries/DifferentialEquations/Default.aspx, (accessed 17 March 2017).
7. W.D. Passos, *Numerical Methods, Algorithms and Tools in C#* (CRC Press, 2009).
8. K. Deb, P. Amrit, A. Sameer, T. Meyariva, IEEE Transactions on Evolutionary Computation, **6**(2), 182 – 197 (2002)