Applied Mathematics and Computation 329 (2018) 239–250

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A genetic algorithm for the characterization of hyperelastic materials

J.R. Fernández [a,*], J.A. López-Campos [b], A. Segade [b], J.A. Vilán [b]

[a] Departamento de Matemática Aplicada I, Universidade de Vigo Escola de Enxeñería de Telecomunicación, Campus As Lagoas Marcosende s/n, Vigo 36310, Spain

[b] Departamento de Ingeniería Mecánica, Máquinas y Motores Térmicos y Fluídos Escola de Enxeñería Industrial, Campus As Lagoas Marcosende s/n, Vigo 36310, Spain

ARTICLE INFO

Keywords:
Hyperelasticity
FE analysis
Non-linear problem
Genetic algorithms

ABSTRACT

This work deals with the characterization of a hyperelastic material and the subsequent validation in different stressed states. The well-known three-parameter Mooney–Rivlin model is chosen for the sake of simplicity. In order to obtain the mechanical properties of this material, a specimen is tested using tensile forces. Once the tests are performed, the material constants are determined using a genetic algorithm to fit the experimental curve. An accurate fitness function is defined and the procedure to obtain the generations is described. Finally, with the aim to model and validate the results in a more complex setting, physical tests are performed in combination with numerical studies and FEM simulations.

1. Introduction

Hyperelastic constitutive laws are used to model materials that respond elastically when subjected to very large strains. They account both for nonlinear material behaviour and large shape changes. The main applications of the theory are to model the rubbery behaviour of a polymeric material, and to model polymeric foams that can be subjected to large reversible shape changes (e.g. a sponge).

Even if this kind of materials is widely used in industry, the difficulties to obtain an accurate mathematical characterization limit the possibilities to generate feasible numerical models that predict their behaviour. As a consequence, the virtual Finite Element (FE) models using hyperelastic materials are in some cases unable to reach accurate results. Even if nowadays there exist a good number of models (see, e.g., [1,2,5,7,8,17–19,21]), theirs material constants usually are not simple to be obtained. The simple test performed in order to find the material data is a uniaxial tensile test [24]. Once the test is carried out, the obtained results are fitted by a three-parameter Mooney–Rivlin model [16,23] using a fully incompressible assumption for the material [1,15]. The first challenge in this work is to get an accurate model characterization for a complex material using only simple tests on a specimen. Once the characterization is obtained, it will be used in a FE code in order to solve a more complex problem involving different stress states.

In order to solve characterization problems, several different methods are available to find the value of parameters in hyperelastic models. Most of authors use least square fitting methods [13], very often based on iterative processes which

* Corresponding author.
E-mail address: jose.fernandez@uvigo.es (J.R. Fernández).

https://doi.org/10.1016/j.amc.2018.02.008
0096-3003/

require an initial data to initialize the method and also differentiation of the objective function. In our case, we use Genetic Algorithms (GA) to perform the searching of parameters in order to characterize the material. This technique does not require differentiation and, although there are other available and more sophisticated methods which do not require it [26], GA are methods oriented to global optimization and are less sensitive to initialization as other techniques. Moreover, using GA we do not need to provide an initial data for the parameters to run the algorithm, although a parametric domain must be defined. Finally, using this method we are able to include multi-objective optimization if needed (for example in order to use different tests for the characterization) in a very simple way.

Genetic Algorithms represent a highly efficient search procedure for the determination of global extrema of multivariable functions, imitating the patterns of genetic reproduction in living organisms. They were first introduced by Goldberg [10] and Holland [11] and differ from conventional search algorithms in the following aspects: (1) they do not work directly with the parameter set; rather, they manipulate strings of characters representing the parameters themselves, (2) they consider many points in the search space simultaneously, (3) they use random choice to guide their search, and (4) they require no derivative information. During the last twenty years, a large number of papers have been published dealing with problems solved with this type of algorithms (see, e.g., [6,9,12,14,20,22,25,27,29] and the numerous references cited therein). They work by successive modifications of a collection (often referred to as population) of parameter combinations in the search space, using a binary coded representation termed chromosome. The initial set of chromosomes evolves through a number of operators which will be detailed in the next sections, so that subsequent generations produce more and more chromosomes in the neighbourhood of the optimum, defined through a given figure of merit, or fitness, which should be previously stated.

In this work, in order to obtain the constitutive parameters of the Mooney–Rivlin model, a genetic algorithm is used to find theirs suitable values and the better fitness as possible with the material model. Once this is fitted and the hyperelastic behaviour is reproduced, new different tests are performed, based on the characterization done, and a more complex FE model is built. The new problem involves stresses in several directions (not only uniaxial tensile stresses) and the obtained characterization will be used in a FE code to solve a large deformation problem.

The paper is outlined as follows. In Section 2, the classical three-parameter Mooney–Rivlin model is presented. Then, in Section 3 the genetic algorithm implemented for the calculation of these parameters is described, providing details concerning the population initiation, the genetic codification, the selection operators and the fitness evaluation. The results obtained with this algorithm are shown in Section 4, including a basic example to show the accuracy with a known solution and a real case. Finally, the data obtained with the second example are used in Section 5 for the simulation of the deformation of a rubber plate under extensional forces.

2. The Mooney–Rivlin model

In the classical linear elasticity theory used to model metallic behaviour at small strains, the constitutive relation between stress and strain remains linear. This assumption could be right when talking about metals suffering from stresses under their yield stress but, when talking about rubbers, foams or, for example, biological tissues, the relationship between stress and strain takes a more complex form. In the case of elastomers, strong non-linearities are observed in their stress-strain curve. Hence, the classical elasticity theory is not fulfilled by such materials and a different way to model their behaviour must be used.

In this framework, the strain energy density function is defined in order to model the hyperelastic behaviour. This function is understood as the area below the stress-strain curve, plotted in Fig. 1 as the red line, and so the shaded zone represents it.

The aim of a hyperelastic model is to obtain a formula which reproduces the strain energy density function, and after this, the constitutive law can be obtained for each elastomer material by using the large deformation elasticity theory. The definition of the strain energy density function, also called elastic potential, is given as a function of the invariants of the right Cauchy tensor.

There exist a large number of hyperelastic models (see, for instance, [1]) which can be used with different number of material constants, depending on how it is interesting to take into account the invariants of the Cauchy tensor, and the quantity and quality of the data obtained from the material to be modelled.

In this work we deal with the Mooney–Rivlin models (see, e.g., [1]), because they are one of the most widely used hyperelastic models, especially to model rubbers. Among all available Mooney–Rivlin models, the three-parameter model is the chosen one because it is a simple model which takes into account the two first invariants of the Cauchy tensor and also their product. Although models with more constants could be used, the resulting data are probably not enough to obtain a good quality of the solution.

The hyperelastic equation used to fit the obtained data must be defined according to the FE software which will be used (ANSYS). In this way, the strain energy density function is defined exactly as detailed in ANSYS user's manual and so, once the parameters are obtained, they can be used immediately in the software. It has the following form:

$$W = C_{10}(\bar{I}_1 - 3) + C_{01}(\bar{I}_2 - 3) + C_{11}(\bar{I}_1 - 3)(\bar{I}_2 - 3) + \frac{1}{d}(J-1)^2,$$

where W denotes the strain energy density function, C_{10}, C_{01} and C_{11} are the constants that the algorithm must approximate and d is the compressibility parameter. In this case, since the material is assumed incompressible, this term is neglected. Finally, $\bar{I}_1$ and $\bar{I}_2$ are the first two invariants of the right Cauchy–Green deformation tensor defined by Bonet [4].

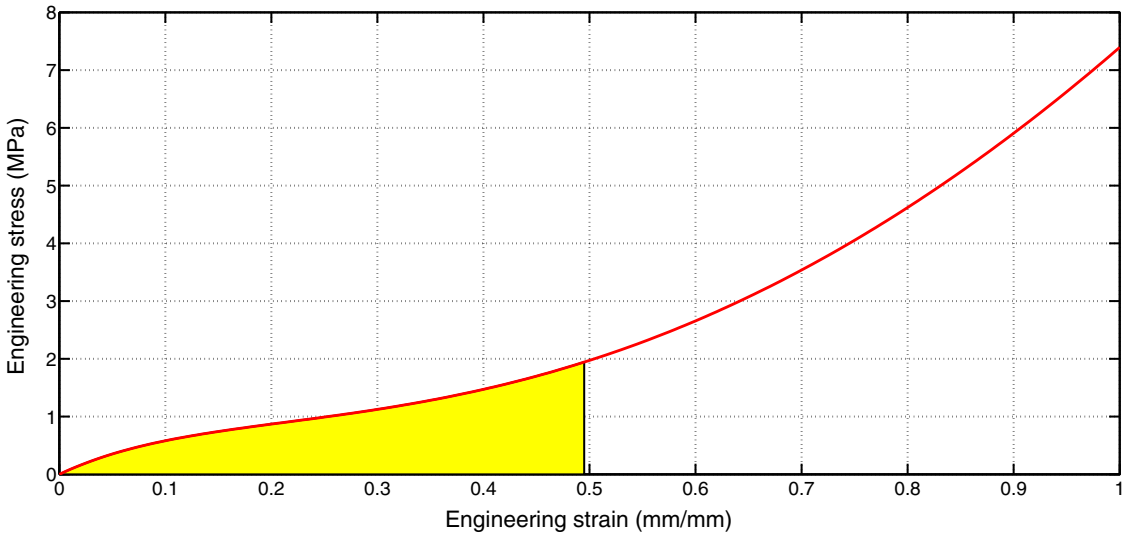

Fig. 1. Strain energy density function.

Hence, taking into account all the features previously commented and, as the starting point, the data obtained from a rubber specimen in physical tests, our objective is to find the set of Mooney–Rivlin parameters that better fits the real data. In order to obtain the suitable values for the parameters, a fitness process applying a genetic algorithm will be implemented using Matlab code.

3. The genetic algorithm

The search of the suitable parameters for the Mooney–Rivlin model, in order to reproduce the behaviour of a given specimen, is the most important feature in this work. In order to obtain the better fitness of the curve, a genetic algorithm will be used.

This algorithm is inspired in the Darwin's theory based on the evolution of the species. On one hand, according to this theory, the individuals better fitted to their environment are able to survive and theirs genes could be transferred to the future generation. On the other hand, the worst-fitted individuals would not survive and theirs genes finally will be eliminated during the evolution of their specie. Also, during the evolution process, some individuals could suffer the alteration of theirs genes (mutation), leading to, in this way, very different individuals. The mutation could be an advantage from the point of view of their survival or not: if their mutation is an advantage, theirs mutated genes will be transferred to the new generations but, on the contrary, if the mutation is an inconvenience, theirs genes will disappear.

As we commented previously, in order to solve the problem, a set of individuals is evaluated. These individuals are characterized by theirs values for the constants C_{10}, C_{01} and C_{11}. The values of these parameters always belong to one space called parametric domain, and the algorithm will look for the solution always inside this domain, which must be defined taking into account the character of the problem. Because of the different values of the Mooney–Rivlin parameters, each individual reaches different fitness level with respect to the given test data.

Therefore, the individuals whose values for Mooney–Rivlin constants imply better fitness are selected to transfer theirs genes to a subsequent generation. In this way, the genes of the individuals of the new generation are a combination of the genes of the selected individuals in previous generations. Consequently, better genes will be transferred to the new generations while the worst genes will be eliminated by the selection process. Moreover, during the evolution process, some mutations in the genes could occur and, if this mutation is an advantage, the individuals are selected to transfer theirs genes to the next generation in their offspring.

Finally, during a specific number of generations, the population will have evolved and the individuals of the last generation will be, in general, better fitted than the ones of the first generation. Due to this fact, theirs values for the Mooney–Rivlin constants are more suitable to model the behaviour of the rubber. After enough generations, the algorithm should converge and the values for the fitness shall not be increased. At this point, the values for the parameters are almost the same for each individual in the same generation and the fitness process can be considered as finished.

3.1. Population initialization

The first step to implement a genetic algorithm is to define the first generation. In general, this algorithm will create the generations by using crossover and mutation operations based on the previous generations. In the case of the first generation, since there are not predecessors to obtain the population, one initialization phase must be carried out.

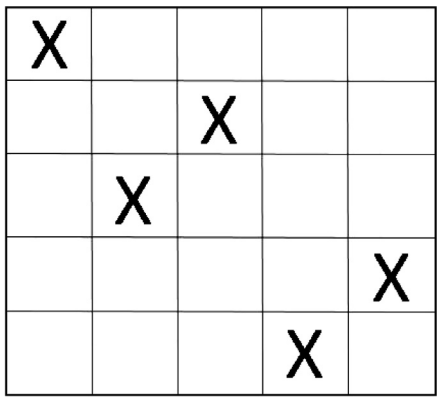

Fig. 2. Latin Square Sampling graphic model.

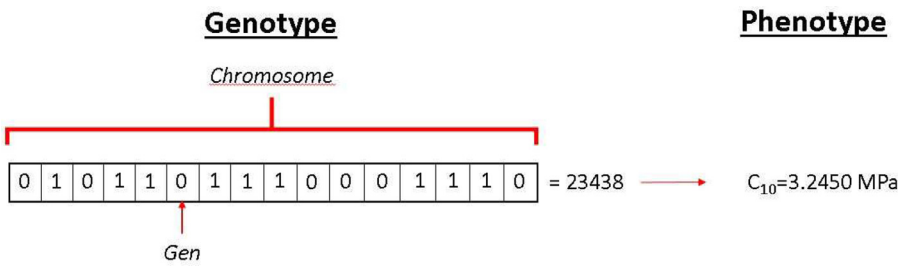

Fig. 3. Genetic codification of the problem.

The initialization phase is a very important step and, depending on it, the algorithm could reach or not convergence. As we commented previously, the algorithm works in a parametric domain, and each one of the coefficients could take different values between a maximum and a minimum value, which represent the boundaries of the parametric domain.

In order to define an optimal way to initialize the population, a Latin Hypercube Sampling (LHS) method is used (see, for instance, [28]). This method is applied to generate a semi-random sample in a parametric domain in such a way that two individuals will not share input parameters with the same values. Indeed, the sampling of the parametric domain is optimized to generate a population whose parameter values are spread on the given domain and the individuals fill in an optimal way the domain.

As an example, in Fig. 2 a representation of the operation of the LHS algorithm is shown. In this case, the figure shows the algorithm working in a two-dimensional parametric space and so, the sampling method is often called Latin Square but, in its generalization to a multidimensional parametric domain, the method gets its name, that is, Latin Hypercube.

3.2. Genetic codification

As when talking in biological terms, all individuals possess a different set of genes which codifies the properties of the individuals. In the studied case, the properties are the Mooney–Rivlin parameters and, as a consequence of the values of these parameters, the individual has more or less possibilities to survive the selection process.

The power of this method resides in the fact that all the evolution process is carried out with a codification of the problem, not with the original one. Thus, it is very important to employ a suitable codification of these problems, in the way that genes of the individuals are related to theirs corresponding Mooney–Rivlin parameters, leading to the fact that the crossover and mutation operators, which will be described later, work over the genes, not over the Mooney–Rivlin parameters.

According to the previous comments, each one of the individuals is characterized by three sets of genes, called *chromosomes*. These three chromosomes correspond to each one of the three Mooney–Rivlin parameters and so, between each chromosome and its corresponding Mooney–Rivlin parameter, a mapping function is defined. In biological terms, the set of genes could be called *genotype* and the corresponding value for the parameter called *phenotype*. Now, it is necessary to find a proper codification for each chromosome using several genes and defining their value.

In this case, each chromosome has 16 genes in a row and it is codified using binary code, with values zero or one. In Fig. 3, a representation of the codification process is shown. It can be seen how the structure of one chromosome is built and how, depending on the value of its genes, the value of the corresponding Mooney–Rivlin parameter is obtained as it is detailed below.

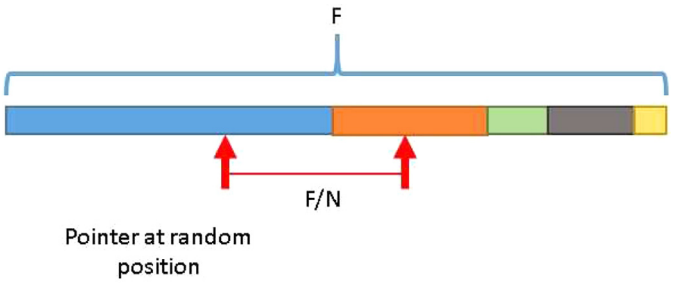

Fig. 4. SUS graphic example.

To obtain the value of the Mooney–Rivlin parameters and vice versa, a mapping has to be defined between both genotype and phenotype. Therefore, a relationship is established using the minimum and maximum values for the parameter and the structure of the chromosomes. In order to gain generality, ϕ_{max} represents the maximum value of the parameter (upper boundary of the parametric domain) and ϕ_{min} is the minimum value, whereas ϕ_i denotes the individual value of the parameter. Also, b is the decimal value of the chromosome binary string and N is the number of genes (bits) for each chromosome. Thus, the resulting formula is [12]:

$$\phi_i = \phi_{min} + \frac{b}{2^N - 1}(\phi_{max} - \phi_{min}).$$

So, with the previous equation, the mapping between genes strings and Mooney–Rivlin values can be carried out and the codification for the problem is complete. This fact will allow to use a codification of the problem and no the original problem in evolution operators. Thus, all the mutations and crossovers are defined over the codified problem.

3.3. Selection operator

With the aim to create new individuals, the individuals with better fitness in the current generation must be selected to transfer theirs genes to the next generations. This is a key process and it allows to select the suitable genes and to eliminate the worst ones. There are many different ways to carry out the selection process but, in this work, two methods are used at the same time. In both methods, the predecessors and the offspring in the same generation could be selected. Therefore, if one individual has better fitness than its offspring, then it is possible that such individual could be selected but not its offspring.

The first method used to select individuals in one generation is the so-called *tournament selection*. With this method, all the individuals in the current generation are ranked depending on their fitness and then, the first three individuals are selected to transfer theirs genes to their offspring.

Then, in order to add randomly to the algorithm, a Stochastic Universal Sampling (SUS) method (see, e.g., [3]) is used to select four more candidates. Thus, the individuals are selected from the population according to their fitness. Hence, all individuals could be selected but it is more probable to select best-fitted individuals.

In Fig. 4 an example of the SUS method is shown. The colour bars in the figure represent the fitness of different individuals. According to the total fitness of the generation, and depending on the number of individuals selected, the distance between pointers is able to be computed. The first pointer is randomly initialized at any point in such a way that the remaining pointer positions are known, and the selection could be carried out.

During the selection process, the same individual can not be selected twice and, in the case two pointers select the same individual, this individual is selected and then the pointer is re-initialized in order to select another non-selected individual.

3.4. Crossover operator

Once the selection is carried out, now with the aim to generate new and better individuals, the previously selected individuals must be recombined to generate an offspring with, in principle, better properties.

There are many possible options to generate new individuals using crossover. During this process, two chromosomes are recombined using one part from one of the chromosomes and the other part from the other one. The way to combine theirs genes will make the difference. In this work, three different crossover methods have been proved, being the uniform crossover the most suitable method attending to their convergence.

In Fig. 5, the three methods are shown: one point crossover and two point crossover are based on determining one or two crossover points and then, using these points as a reference, recombining the two genes string. However, in the uniform crossover, each gene is allowed to be from progenitor A or B, taking the decision using a 50% probability. So each gene has the same probability to be from progenitor A than from progenitor B.

As we noticed previously, in this case the uniform crossover has been finally chosen after comparison with the other two methods, due to the behaviour in the convergence of the proposed tests (its rate of convergence is slightly better). From the

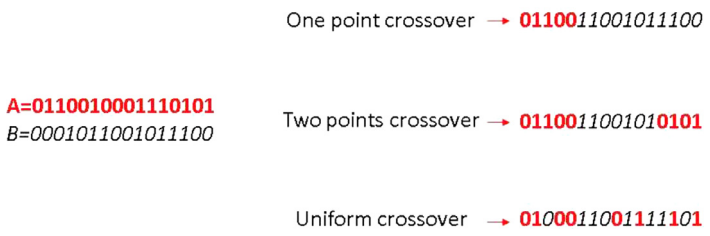

Fig. 5. Crossover methods.

two progenitors, other two descendants are created at the same time and the combinations are always complementary. With this method, new individuals are created from the previous ones inheriting theirs genes.

3.5. Mutation operator

The last operator used to simulate the evolution of the species is the mutation (see [14] for details). As in biology, the mutation is a phenomenon which implies the alteration of one or several genes in an individual. This is an uncommon phenomenon and the probability of its occurrence is very low (at the order of 1% or even less). However, this is an important operator because, thanks to it, some genes are able to take values that, without mutation, would be impossible and, in some cases, the mutations could be favourable.

With this genetic operator, the parametric domain can be scanned in a better way, reaching points that, only with selection and crossover, would not be obtained. In this algorithm, mutation is the last operator and any gene of each individual in the new generation could be mutated. So, the size of the population is slightly variable from one generation to the next one due to mutations. In this case, each gene is associated to a mutation with probability equals to 1 %, and randomly the algorithm decides if this gene mutates or not. If the answer is positive, the mutated individual is generated and the population is increased with both individuals.

3.6. Fitness evaluation

Finally, the algorithm must evaluate the fitness level of the new generations in order to select the following candidates whose genes will be recombinated. In this case, it is clear that the fitness function must be related to the error between the curve created with a set of Mooney–Rivlin parameters and the data obtained with the real test.

In this case, the fitness takes values from 100 to 0 according to the absolute error measured between the generated curve and the real data. So, if the absolute error is known, the fitness of the individuals is computed using the curve shown in Fig. 6. In particular, we use the following formula:

$$F(x) = \begin{cases} 100 - 16x & \text{if} \quad x \in [0, 5], \\ \dfrac{100}{x} & \text{if} \quad x > 5, \end{cases}$$

where x represents the absolute error and $F(x)$ denotes its corresponding fitness value. This fitness evaluation guarantees that the domain for the error is infinite and it is a very intuitive way to evaluate the fitness of an individual. So, when the error of the curve is near to zero, the fitness is near to 100% and vice versa.

4. Results for the genetic algorithm

Once the algorithm is implemented with all the previous considerations, it can be used to fit one experimental curve.

4.1. Simulation of a known solution

In order to test the behaviour of the algorithm, we select one set of constants for the parameters C_{10}, C_{01} and C_{11} with values 10, 20 and 5 *MPa*. This parameter set is the solution of the problem and we will try to find this solution employing our algorithm with the aim to check its performance. Therefore, using these parameters we generate a curve corresponding to a uniaxial tensile test until a 300 % strain level. Such curve will act as input data to run the genetic algorithm. Once it fits the curve, it finds a possible solution to the problem – a parameter set near to the proposed one.

The algorithm runs during 50 generations initializing with 40 individuals, the overall of the execution is completed in 83 s of CPU time (and so each generation is obtained in 1.66 s). Finally, after several runs, the same solution is found. The obtained Mooney–Rivlin parameters are 8.73, 22.5 and 5.078 MPa for C_{10}, C_{01} and C_{11}, respectively, and the mean square relative error is 0.15 %. We note that this solution is therefore close to the known solution established at the beginning of the section.

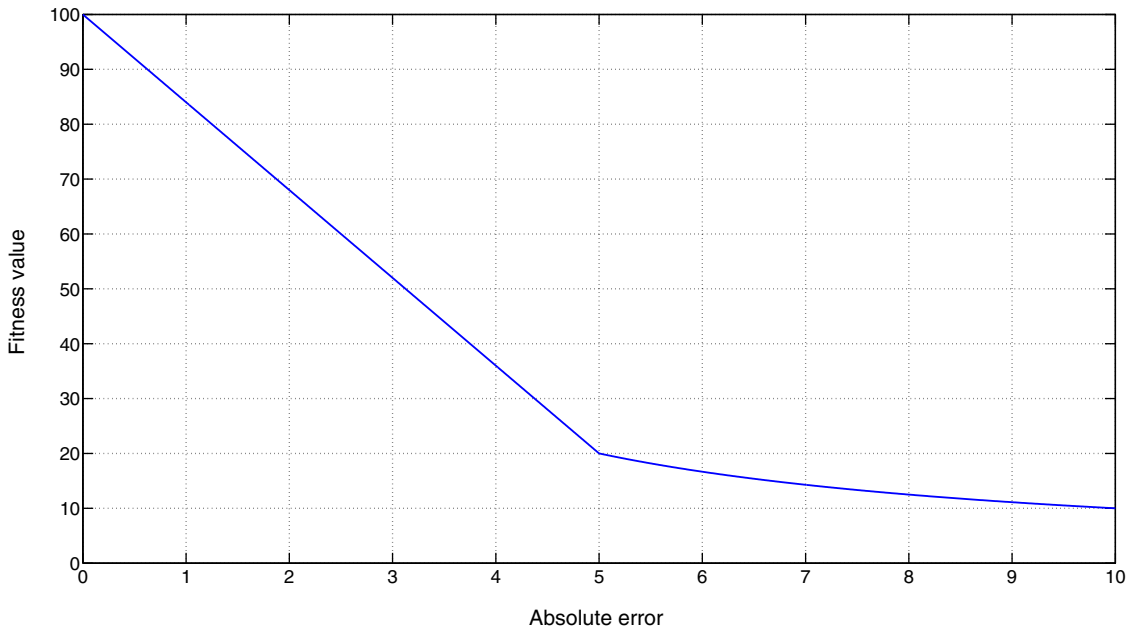

Fig. 6. Fitness value vs. absolute error.

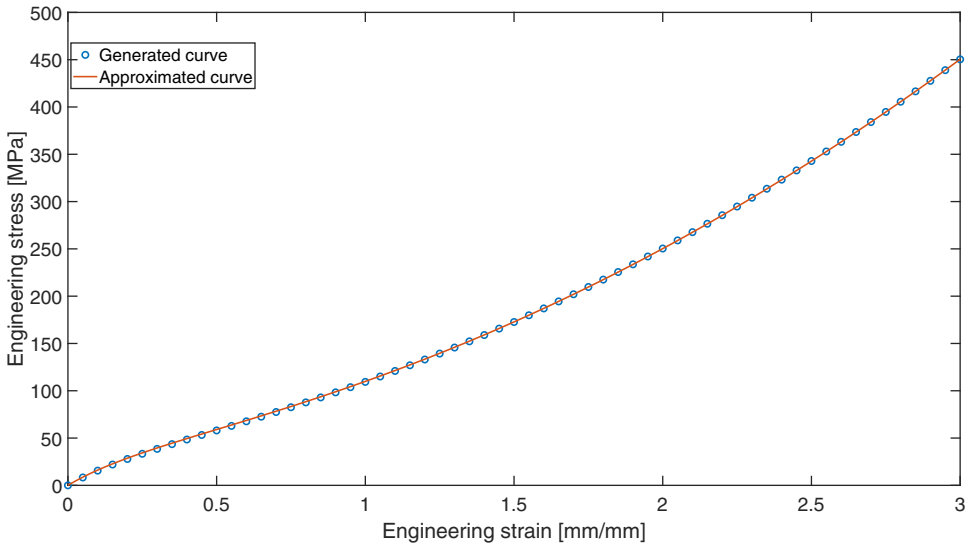

Fig. 7. Stress-strain curve (candidate and exact curve).

With the aim to show the obtained results, both generated and approximated curves are shown in Fig. 7, where it can be highlighted that the two curves almost coincide along all the strain domain. So, it can be concluded that, if enough amount of data is used to fit the curve, the results will converge and they approach correctly the input data.

4.2. Simulation of a real case

In this second case, the stress-strain curve involves some signal noise due to the instrumentation used in data acquisition, so the fitness is not able to reach 100%.

The algorithm runs for 100 generations and the population is initialized with 70 individuals in a specific parametric domain. Two runs are performed depending on the size of the parametric domain. One run is carried out with a bigger parametric domain and then, based on the results of the first run, the parametric domain is decreased.

Table 1
Genetic algorithm results.

Mooney–Rivlin parameter	C_{10} (MPa)	C_{01} (MPa)	C_{11} (MPa)	Fitness
Candidate 1	−4.431296	5.918471	1.868925	62.13495
Candidate 2	−5.940032	7.583124	2.499886	62.760516
Candidate 3	−6.806516	8.554818	2.812467	52.765446

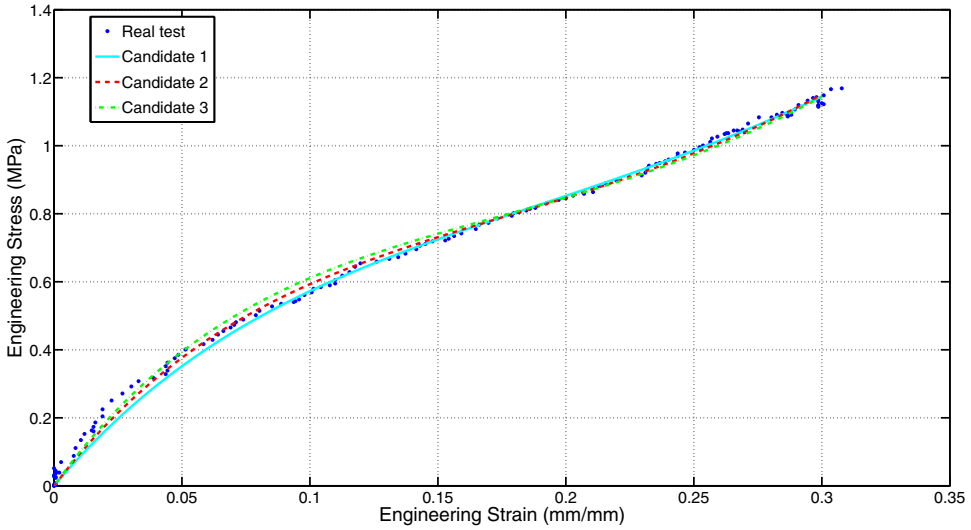

Fig. 8. Stress-strain curve (candidates and real test).

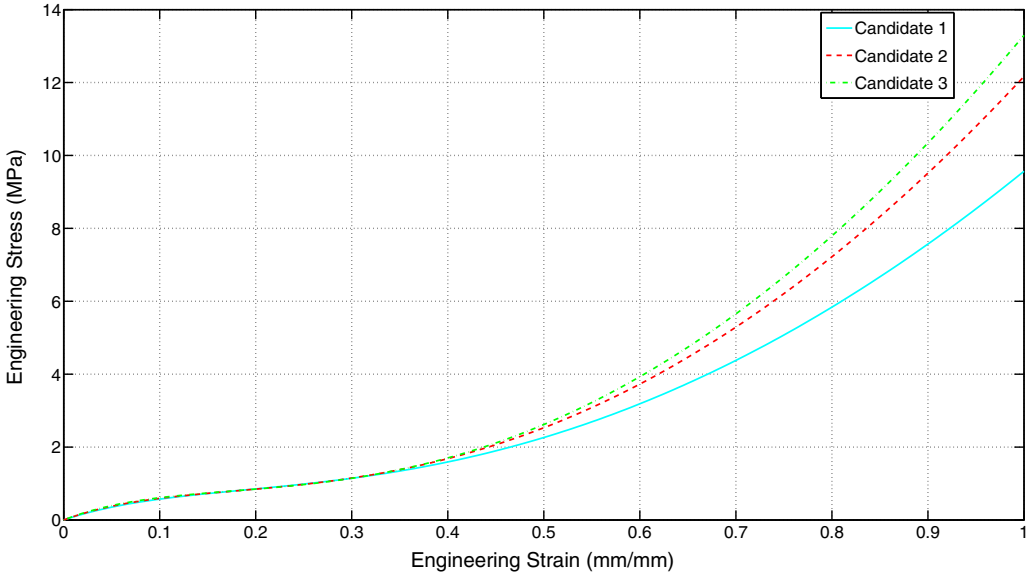

Fig. 9. Stress-strain curve extended until 100% strain.

During several runs, the algorithm converges in a recursive way to three solutions. These solutions characterize the same curve with three different sets of Mooney–Rivlin parameters but, in the domain of the strain where the rubber is tested (under 30% strain), the three curves are similar. The obtained results using the algorithm are shown in Table 1.

In order to show in a graphic way the behaviour of the three candidates, Fig. 8 is built, where the engineering stress-strain curves generated by each of the found solutions are plotted. Moreover, a cloud of points representing the real behaviour is shown. Such data are obtained from real tests performed using load and displacement sensors to extract the real response of the test specimen. In spite of signal noise due to sensors, using the proposed genetic algorithm we can solve

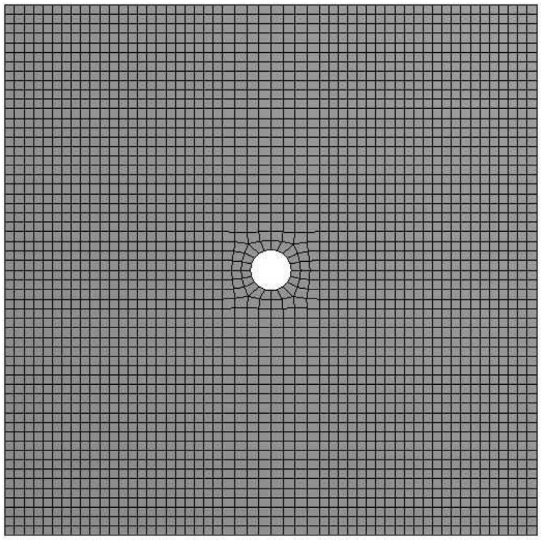

Fig. 10. Mesh of the model used for the FE analyses.

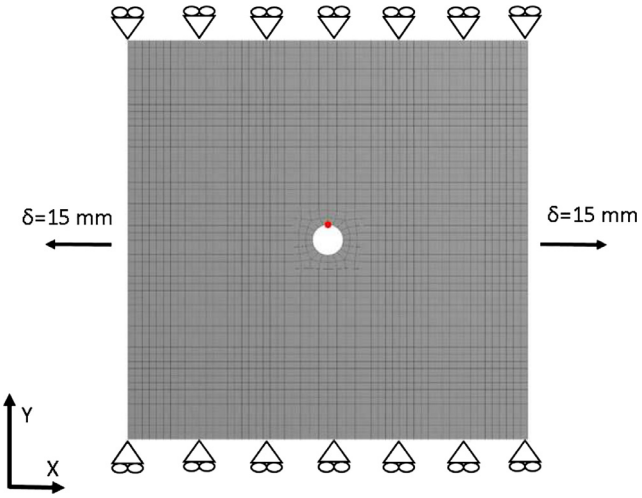

Fig. 11. Boundary conditions for the FE model.

the minimization error problem taking into account such possible dispersion and so, we can use data directly from output of sensors.

As shown in the figure, the three candidates imply a good fitness level taking as a reference the real test data. According to these results, we can conclude that the three candidates could be a good choice to reproduce the behaviour of the rubber.

It is important to highlight the fact that, in the tensile strain, because of the available resources, the strain only reaches 30%. Then, the algorithm takes only into account the available data and, although the three candidates have a similar behaviour in the domain used in the test, it is possible that, at higher strain levels, the behaviour differs clearly among them as shown in Fig. 9. So, if higher strains were reached in real tests, maybe some of these candidates would not be selected by the algorithm.

5. FE analyses

Finally, in order to prove the utility of the fitting, the obtained parameters will be used in a FE model with the commercial code ANSYS. The algorithm was originally thought to obtain the Mooney–Rivlin parameters according to the definition given by ANSYS for the elastic potential. Hence, the obtained values may be introduced directly in the software. The simulated problem consists in a rubber plate 2 mm thick with a hole at the center. The plate will be a square with 160 mm. The mesh is composed by 3020 quadrilateral linear elements and 3140 nodes, see Fig. 10.

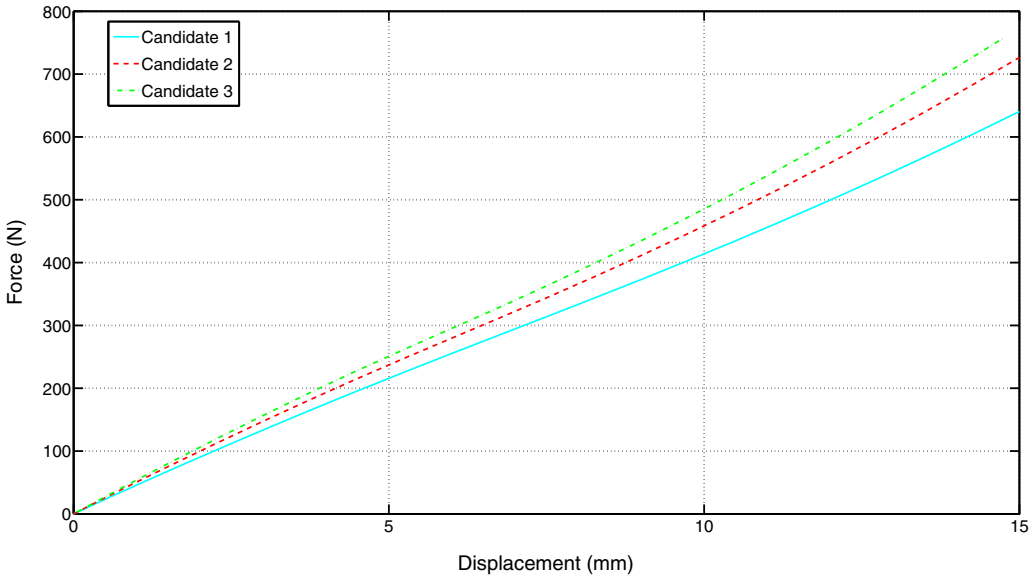

Fig. 12. Comparison of the reaction forces for the three candidates.

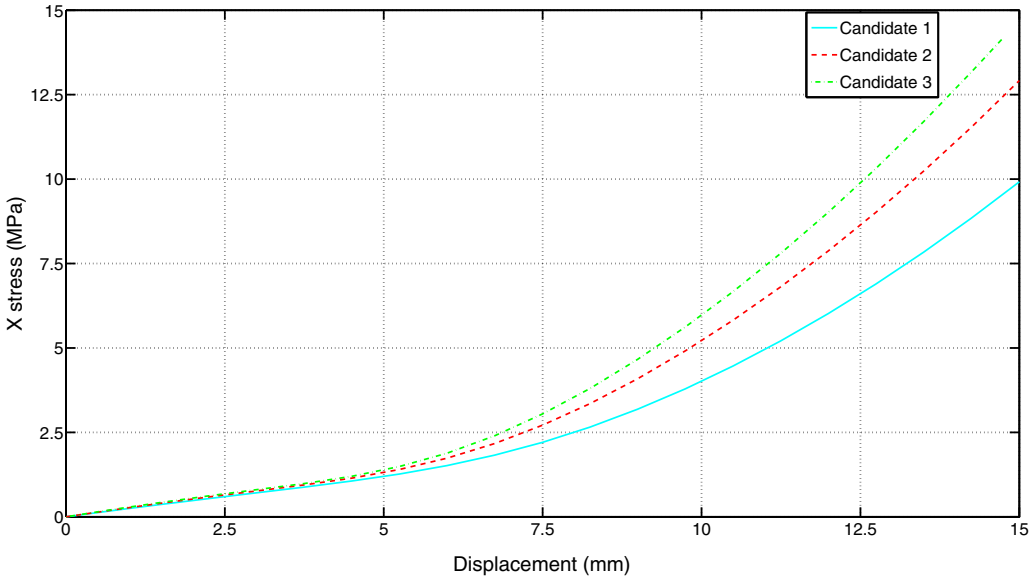

Fig. 13. *X* principal stress vs. displacement for the three candidates.

This geometry is submitted to a more complex load state in order to prove the effectiveness of the computed coefficients. All three candidates will be used and the results will be compared for the three cases. As boundary conditions, the horizontal boundaries are constrained in the *Y*-direction, meanwhile, in the vertical ones, a horizontal displacement of 15 mm is enforced (see Fig. 11).

The three candidates will be tested with exactly the same mesh and boundary conditions, in order to compare the corresponding responses generated for each one of the parameter sets obtained with the Mooney–Rivlin model. In all three analyses, the material is assumed as fully incompressible.

Then, the problem is solved now using ANSYS software and the obtained results are post-processed. In this case, there are some results that could be interesting to study and to compare for each of the candidates. The first result is the force needed to produce the 15 mm fixed displacement in each specimen. In Fig. 12, a comparative curve of this force in each case is shown. As can be seen, near to the zero displacement, the results are very similar. This is because of, as previously commented, the test data used to fit the curve reaches only 30% strain. The same feature shown in Fig. 9 is observed and, when the model works close to the zero displacement, in the domain where test data were taken, the three models

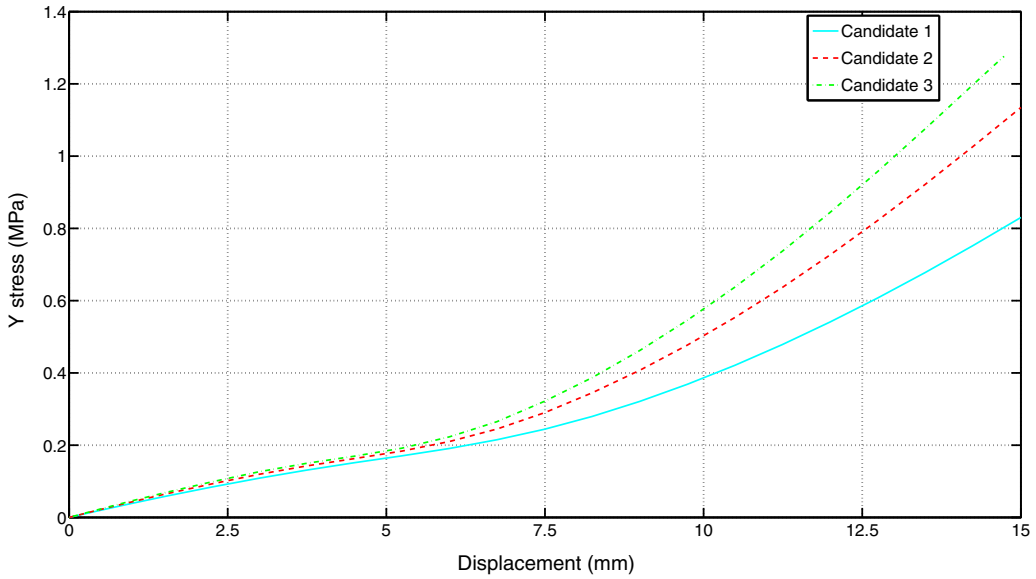

Fig. 14. *Y* principal stress vs. displacement for the three candidates.

produce similar results. On the contrary, if a large deformation takes place, where the test data do not give any information, the three models diverge. So, in order to use the algorithm to obtain data for FE simulations, it is important to test the rubber according to the strain domain where it will be simulated.

In addition to the reaction forces, also the stress is evaluated in each of these models. The stress is evaluated at the node highlighted in Fig. 11 in $X-$ and Y-directions. Thus, in Figs. 13 and 14 the stresses at such point are shown for each candidate found by the algorithm.

6. Conclusion

This study shows the possibility to use genetic algorithms to fit experimental data, in order to obtain the necessary parameters to model hyperelastic materials in a FE software. During the study, a functional genetic algorithm was designed to fit Mooney–Rivlin parameters according to real data. This kind of algorithms has the main advantage that no differentiation is needed in order to establish a searching direction as to Newton methods, and so the fitness function could be non-differentiable. In addition, the use of genetic algorithms allows us to include in a very simple way test data with noise and even to extend the algorithm to use data from several tests (uniaxial, biaxial, shear) if available. Indeed, it is possible to think about the possibility to perform multi-objective optimization and to characterize materials with more complex behaviour including effects as nonlinear viscoelasticity or creep.

Finally, once the curve fitting is solved several times by the algorithm, three possible candidates are computed. The difference between these models is negligible in the domain of the test data but, far from this domain, the three models begin to diverge. The three candidates are also tested in a FE model using a more complex load case. In view of the results obtained using this FE model, the same conclusion could be taken. As a final conclusion, it is important to point out that the genetic algorithm is able to fit a Mooney–Rivlin model and to generate useful parameters which could be included into a FE model, but the test data used to compute the material parameters must be obtained from the strain level where the real component is considered.

Acknowledgments

The work of J.R. Fernández was partially supported by the Ministerio de Economía y Competitividad under the research project MTM2015-66640-P (with FEDER Funds).

J.A. López-Campos, A. Segade and J.A. Vilán gratefully acknowledge the funding by Xunta de Galicia, Spain, under the program *Grupos de Referencia Competitiva* with Ref. GRC2015/016.

J.A. López-Campos also acknowledges the funding from the University of Vigo under the program *Axudas predoutorais* with Ref. 00VI131H641.02.

References

[1] A. Ali, M. Hosseini, B.B. Sahari, A review of constitutive models for rubber-like materials, Am. J. Eng. Appl. Sci. 3 (1) (2010) 232–239.
[2] M.M. Attard, Finite strain–isotropic hyperelasticity, Int. J. Solids Struct. 40 (2003) 4353–4378.

[3] J.E. Baker, Reducing bias and inefficiency in the selection algorithm, genetic algorithms and their applications, in: Proceedings of the Second International Conference on Genetic Algorithms, 1987, pp. 14–21.
[4] J. Bonet, R.D. Wood, Nonlinear Continuum Mechanic for Finite Element Analysis, Cambridge University Press, 2008. Chapter 2–3.
[5] G. Chagnon, G. Marckmann, E. Verron, A comparison of the hart-smith model with Arruda–Boyce and gent formulations for rubber elasticity, Rubber Chem. Technol. 77 (2004) 724–735.
[6] C.K. Chen, J.M. Lin, C.L. Chen, Error bounds estimate of weighted residuals method using genetic algorithm, Appl. Math. Comput. 81 (1997) 207–219.
[7] A.D. Drodzdov, Constitutive equations in finite elasticity of rubbers, Int. J. Solids Struct. 44 (2007) 272–297.
[8] A.N. Gent, Engineering with rubber, Oxford University Press, New York, 1992.
[9] M.K. Ghorabaee, M. Amiri, P. Azimi, Genetic algorithm for solving bi-objective redundancy allocation problem with k-out-of-n subsystems, Appl. Math. Model. 39 (2015) 6396–6409.
[10] D.E. Goldberg, Genetic algorithms in search, in: Optimization and Machine Learning, Addison Wesley, 1989.
[11] J.H. Holland, Adaptation in Natural and Artifficial Systems, MIT press, Cambridge, Massachussets, 1992.
[12] C.L. Karr, B. Weckm, D.L. Massart, P. Vankeerberghen, Least median squares curve fitting using a genetic algorithm, Eng. Appl. 8 (2) (1995) 177–189.
[13] C. Li, J. Lua, A hyper-viscoelastic constitutive model for polyurea, Mater. Lett. 63 (2009) 877–880.
[14] S. Marsili-Libelli, P. Alba, Adaptive mutation in genetic algorithms, Soft Comput. 4 (2000) 76–80.
[15] P.A.L.S. Martins, R.M. Natal, A.J.M. Ferreira, A comparative study of several material models for prediction of hyperelastic properties: application to silicone-rubber and soft tissues, Strain 42 (3) (2006) 135–7147.
[16] M. Mooney, A theory of large elastic deformation, J. Appl. Phys. 11 (1940) 582–17592.
[17] A.H. Muhr, Modelling the stress-strain behavior of rubber, Rubber Chem. Technol. 78 (2005) 391–425.
[18] K. Naghdi, Rubber as an Engineering Material, Hanser Publisher, Munich, 1993.
[19] B. Naser, M. Kaliske, M. Andre, Durability simulations of elastomeric structures, in: P.E. Austree, L. Kari (Eds.), Constitutive models for Rubber IV, AA Balkema Publishers, UK, pp. 45–50.
[20] F.J. Navarro-González, P. Compañ, R. Satorre, Y. Villacampa, Numerical determination for solving the symmetric eigenvector problem using genetic algorithm, Appl. Math. Model. 40 (2016) 4935–4947.
[21] R.W. Ogden, Non-Linear Elastic Deformation, Ellis-Horwood, Chichester, 1984.
[22] A. Pourrajabian, R. Ebrahimi, M. Mirzaei, M. Shams, Applying genetic algorithms for solving nonlinear algebraic equations, Appl. Math. Comput. 219 (2013) 11483–11494.
[23] R.S. Rivlin, Large elastic deformations of isotropic materials: I. fundamental concepts. II. some uniqueness theorem for pure homogeneous deformation, Philos. Trans. R. Soc. Ser. A 240 (1948) 459–17508.
[24] M. Sasso, G. Palmieri, G. Chiappini, D. Amodio, Characterization of hyperelastic rubber-like materials by biaxial and uniaxial stretching tests based on optical methods, Polym. Test. 28 (2008) 995–1004.
[25] P.H.T. Schimit, Evolutionary aspects of spatial prisoner's dilemma in a populatin modeled by continuous probabilistic cellular automata and genetic algorithm, Appl. Math. Comput. 290 (2016) 178–188.
[26] M.E.T. Silva, S. Brandao, M.P.L. Parente, T. Mascarenhas, R.M.N. Jorge, Establishing the biomechanical properties of the pelvic soft tissues through an inverse finite element analysis using magnetic resonance imaging, Proc. I. Mech. E. Part H: J. Eng. Med. 230 (2016) 298–309.
[27] K. Spranger, C. Capelli, G.M. Bosi, S. Schievano, Y. Ventikos, Comparison and calibration of a real-time virtual stenting algorithm using finite element analysis and genetic algorithm, Comput. Methods Appl. Mech. Eng. 293 (2015) 462–480.
[28] M. Stein, Large sample properties of simulations using latin hypercube sampling, Technometrics 29 (2) (1987) 143–151.
[29] I.G. Tsoulos, Solving constrained optimization problems using a novel genetic algorithm, Appl. Math. Comput. 208 (2009) 273–283.