CMST 10(2) 169-181 (2004) DOI:10.12921/cmst.2004.10.02.169-181

COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY **10(2)**, 169-181 (2004)

SURFACE RESPONSE OPTIMISATION OF AUXETIC HOMOGENISED CELLULAR PLATES USING GENETIC PROGRAMMING

T. L. LEW*, A. B. SPENCER,
F. SCARPA, AND K. WORDEN

Dynamics Research Group, Department of Mechanical Engineering
University of Sheffield, Mappin Street, SI 3JD Sheffield, UK

(Rec. 11 December 2004)

Abstract: This paper describes an approach based on Genetic Programming to perform the meta-modelling of cellular structure properties with in-plane auxetic behaviour. Common procedures to design microstructure topologies with complex shape is to use analytical and/or Finite Element (FE) models and quantify the variability of their homogenised mechanical properties *versus* internal cell parameters. For the FE case, the large number of computations involved can rule out many approaches due to the expense of carrying out many runs. One way of circumnavigating this problem is to replace the true system by an approximate *surrogate/replacement* model, which is fast-running compared to the original. In traditional approaches using *response surfaces* a simple least-squares multinomial model is often adopted. The object of this paper is to extend the class of possible models considerably by carrying out a general *symbolic regression* using a Genetic Programming approach. The approach is demonstrated on the optimisation of the unit cell of centresymmetric auxetic cellular solids composing a simply supported plate for maximum central deflection under transverse uniform pressure.

1. INTRODUCTION

Engineering computation has played a larger role compare to a few decades ago. With the advances of computer, various engineering computational programs have been developed, tailored for different aspects of engineering. Finite Element Analysis (FEA) alongside with others has become the essential tools to master.

Though the computers have made the life of engineers easier, the new challenges of engineering design nowadays are constantly pushed to solve more complicated problems. This will lead to the issue when the designs are very dependent to the computational power. While the design optimisations are taking too long to perform, engineers and scientists choose for some solutions that create a replacement/surrogate model of the simulation or experimental data.

*Corresponding author: e-mail: t.lew@shef.ac.uk, phone: (+44) 0114 222 77 21, fax: (+44) 0114 222 7890.

This replacement model will be used for the optimisation process, instead of re-generating the data from the simulation/experiments upon each optimisation step. The creation of the replacement/surrogate model is also known as metamodelling.

Traditional metamodelling techniques usually involve using a multivariate response surface model. The research in this field reached its mature state some time ago, but there are some known problems in using these approaches. For example, difficulties with the polynomial basis function, problems in trying to fit a highly nonlinear model using response surface functions, explosion of coefficients and low extrapolation-abilities. It also requires some prior knowledge of the data before choosing the right functions to fit [1]. Overview of the state of art metamodelling techniques can be found in [2].

Recent findings in metamodelling has developed into techniques that are inspired by the natural evolution, which created the most successful and remarkable designs known by mankind. Most engineer and scientists prefer to use an algorithm that is extensively proven for its reusability and robustness, rather spending time, efforts and money customising the data with the chosen functions. This technique, with today's phrase, it is called evolutionary algorithms. Evolutionary algorithms include a few subclasses, such as evolutionary computation, genetic algorithms, genetic programming *etc.* Evolutionary algorithms were proven to have successful results in all types of automated design system [3].

In this work, an example of simple analytical optimisation is presented with the use of genetic programming as a metamodelling technique. The genetic programming code was developed using Java programming language; it is used to fit the structural properties of the auxetic honeycomb, which generated from [4]. The optimisation routine was taken from [5], and is performed in Matlab using Sequential Quadratic Programming (SQP) method. A simply supported plate made by a cellular structure composed by centresymmetric auxetic grid provides the test case considered. Auxetic (*i.e.*, negative Poisson's ratio) materials and structures have recently attracted much attention in the research community for their unusual characteristics and the fact that auxetic behaviour is often accompanied by interesting performance of other multiphysics characteristics. Extended references and descriptions on auxetic systems can be found in [6-9]. Centresymmetric honeycomb structures have been between the first examples of system exhibiting in-plane negative Poisson's ratios. The conventional hexagonal unit cell of a regular honeycomb can be made re-entrant, with an internal negative angle, and exhibits lateral expansion when pulled along one direction [4, 10]. Re-entrant cell honeycombs are highly anisotropic, but features higher Voigt bounds for transverse shear modulus compared their conventional counterpart [11], directional band-gaps behaviour in flexural wave propagation cases [11], and strong dielectric anisotropicity that can be used together with the mechanical one to design electromagnetic compatibility characteristics in microwave absorbers [12, 13].

2. GENETIC PROGRAMMING THEORY

As mentioned above. Genetic programming is a subclass of evolutionary computation techniques, however unlike other evolutionary techniques that evolve numerical values, it evolves functions as a solution for a given problem.

Genetic Programming (GP) shares the same concepts as the well-known Genetic Algorithm (GA) but increases the complexity by allowing the structure of the solution to undergo adaptation. The structure is typically a hierarchical computer program or mathematical function of dynamically varying size and shape.

GP starts with an initial population of randomly generated individuals, which consists of function and terminal nodes appropriate to the problem domain. The appropriateness of the functions are less strict than the traditional metamodel, as it is just to decide if the function used would be arithmetical, logical etc., one can even use some user-defined functions that are suitable for the problem domain. Therefore, depending on the problem domain, the individuals may be real, complex, vector, symbolic, multiple valued etc.

Each individual is built from repeatedly combining all possible functions and terminals:

$$F=\{f_1,f_2,\ldots,f_M\};\ T=\{a_1,a_2,\ldots,a_N\}, \tag{1}$$

where F and T represent function sets and terminal sets respectively; M and N are the number of functions and terminals included in the GP. Each function f_i takes in a specific number of arguments - the number also known as the function's arity, while the terminals have null arity

$$z(f_1,f_2,\ldots,f_M)=\mathbf{Z},\qquad z(a_1,a_2,\ldots,a_N)=0\ . \tag{2}$$

There is however one rule to obey: each function must be applicable to any values returned by other functions and any values carried by the terminal nodes.

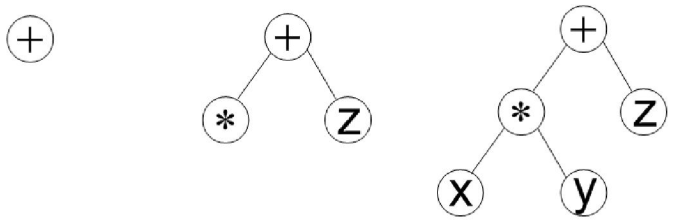


Fig. 1. Creation of individual of initial population

The creation of the initial random population is a blind search in the problem domain, whereby the birth of each individual is achieved by randomly generating a root node, and its subsequent branches. The root node must be chosen from the function set, as illustrated in Fig. 1, a root node '+' has arity 2, with each argument being represented by a connection to a subsequent node. It is then randomly combined with other nodes from either the terminal or function set. If a terminal node was chosen, it will stop branching out; else the growth will continue. The individual created from Fig. 1 represents an expression of $(x \cdot y) + z$.

There are a few growth strategies, namely full method, grow method and ramped half-and-half [14] The full method of generating the initial population involves creating individuals that grow up 'til the maximum allowable level. The grow method involves creating individuals that are variably shaped, whereby the depth can be of any level less than or equal to the maximum allowable level. The ramped half-and-half is a mixture of both. The grow method is adopted in this work.

To prevent the individuals from growing infinitely, a limit was enforced on the depth of the tree structure and on the total number of nodes a tree can have, after which only terminal nodes are chosen. Some other bloat *(i. e.* rapid increase of individual tree size) control methods can refer to [15, 16].

The driving force of GP, as for GA, is by computing the fitness of the individuals before 'mating' them to produce subsequent generations. Each individual in the population is measured in terms of its performance merit, known as the fitness measure. There are several ways of assigning a fitness measure based on different problem domains. [14] The fitness measure enables the best individuals to be chosen to inherit across the generations, and eliminates the unfit ones. The fitness of an individual in this work is assigned by the inverse of its percentage Mean Square Error (MSE):

$$\frac{1}{\text{Fitness}} = \text{MSE} = \frac{100}{N\sigma_y^2}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2 , \tag{3}$$

where N is the number of data points, σ_y^2 is the variance of the desired output, y_i and $\hat{y}_i$ are the desired and estimated output respectively. Generally, a fitness of 10, in another words an MSE percentage of 0.1, is considered as an excellent fitness.

At each generation, genetic operators are applied to modify the individuals to create offspring. The genetic operators are crossover and mutation. The GP has to go through the population to select the 'good/fit' parents, based on their fitness measure, in order to perform the genetic operations. There are several selection methods, like the GA, this work adopted fitness proportionate selections, others include Stochastic sampling with replacement, Stochastic sampling with partial replacement, Stochastic universal sampling [17] *etc.*

The fitness proportionate scheme works as a spin of a roulette wheel, the accumulative sum of the fitness over all individuals in the current population is determined. The individuals are then mapped one-to-one into adjacent intervals in the range [0, sum total]. The size of each individual interval corresponds to its fitness value. A random number is generated in the range of [0, sum total] and the individual whose segment spans the random number is then selected. The process is repeated until the desired number of individuals has been selected [17].

The crossover operation in GP brings variation into the population by producing offspring that inherit parts of each parent. As for the mutation, unlike the GA, the GP mutation operation is to randomly substituting a branch of expression, and often gives more constructive modification [18] than the GA mutation.

There are different crossover and mutation strategies, such as two-parent-one-child, two-parent-two-children crossover, size fair crossover [19], branch mutation, node mutation etc. The genetic operators adopted for this work are two-parent-two-children crossover and a mixture of branch and node mutation.

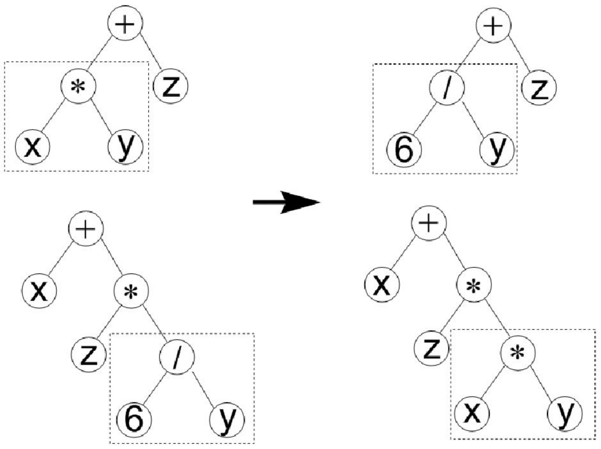


Fig. 2. Two-parent-two-children crossover method

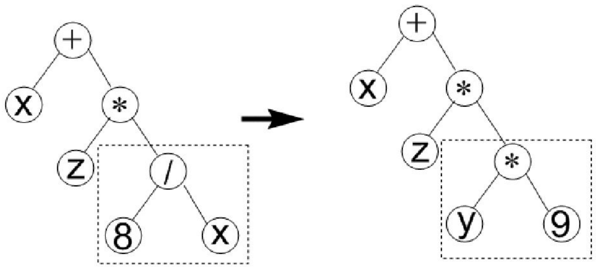


Fig. 3. Branch mutation method

As shown in Fig. 2, a two-parent-two-children crossover has taken place. A random node will be chosen from each of the parent, and the process is just to swap the two chosen node and its subsequent branch from one to the other. In Fig. 3, a branch mutation has taken place, whereby a random node is chosen from one parent, the node and its subsequent branches will be replaced. A node mutation is far simpler; it is just to substitute the randomly chosen node. However, one must take great care of the compatibility of arity between the original and replacement nodes. The mixture of mutation strategies used in this work is namely branch replacement, branch insertion, branch chopping and node mutation. For methods used by other GPs, one can refer to [1, 20-23].

3. SURFACE FITTING USING GENETIC PROGRAMMING

The structural moduli of hexagonal honeycomb were simulated from analytical model in [4].

$$E_1 = E_s\left(\frac{t}{l}\right)^3 \frac{\left(\frac{h}{l}+\sin\theta\right)}{\cos^3\theta}\frac{1}{\left[1+\left(2.4+1.5\upsilon_s+\tan^2\theta+\frac{2(h/l)}{\cos^2\theta}\right)\left(\frac{t}{l}\right)^2\right]}, \tag{4}$$

$$v_{12} = \frac{\sin\theta\left(\frac{h}{l}+\sin\theta\right)}{\cos^2\theta}\frac{1+\left(1.4+1.5v_s\right)\left(\frac{t}{l}\right)^2}{1+\left(2.4+1.5v_s+\tan^2\theta+\frac{2(h/l)}{\cos^2\theta}\right)\left(\frac{t}{l}\right)^2}, \tag{5}$$

$$E_2 = E_s\left(\frac{t}{l}\right)\frac{\alpha+\sin\theta}{\cos^3\theta}\cdot\frac{1}{\left[1+\left(2.4+1.5v_s+\tan^2\theta+\frac{2\alpha}{\cos^2\theta}\right)\left(\frac{t}{l}\right)^2\right]}, \tag{6}$$

$$v_{21} = \frac{\sin\theta\left(\alpha+\sin\theta\right)}{\cos^2\theta}\cdot\frac{1+\left(1.4+1.5v_s\right)\left(t/l\right)^2}{1+\left(2.4+1.5v_s+\tan^2\theta+\frac{2\alpha}{\cos^2\theta}\right)\left(\frac{t}{l}\right)^2}, \tag{7}$$

$$G_{12} = \frac{E_s\left(t/l\right)^2\left(\alpha+\sin\theta\right)}{\alpha^2\cos\theta}\cdot\frac{1}{F},$$

$$F = \left[1+2\alpha+\left(\frac{t}{l}\right)^2\left\{\begin{matrix}\frac{\left(2.4+1.5v_s\right)\left(2+\alpha+\sin\theta\right)}{\alpha} \\ +\frac{\alpha+\sin\theta}{\alpha^2}\left[\left(\alpha+\sin\theta\right)\tan^2\theta+\sin\theta\right]\end{matrix}\right\}\right], \tag{8}$$

where

$$\alpha = \frac{h}{l} = [1,\ 2,\ldots,9],\quad \theta = [-30°,-25°,\ldots,60°],\quad E_s = 1.85\times10^{10}\,Pa,\quad v_s = 0.33,\quad \frac{t}{l} = 0.01,$$

and subscript 1 and 2 are directions, as shown in Fig. 4.

From the above equations, the functions can be simplified in the form of:

$$(E_1, \nu_{12}, E_2, \nu_{21}, G_{12}) = f(\alpha, \theta).$$

Therefore the terminal set and function set can be defined as:

$$T = \{\theta, \alpha, \mathbf{R}\},$$

$$F = \{+, -, \times, \div, \sin, \cos, \text{power}, \log\},$$

with the associated arity numbers being {2, 2, 2, 2, 1, 1,1,1} respectively.

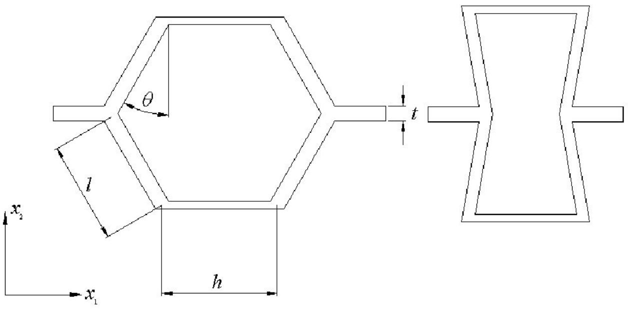


Fig. 4. Classical hexagonal honeycomb configuration with variation of θ

The GP code was first tested with the E_1 data and ν_{12} data with 10000 generations and 500 individuals configured for E_1 and 10000 generations and 100 individuals for ν_{12}. The GP was initially structured to have only branch mutation, two-parent-two-children crossover, maximum allowable nodes and maximum allowable levels was set as 50 and 10 respectively. As it

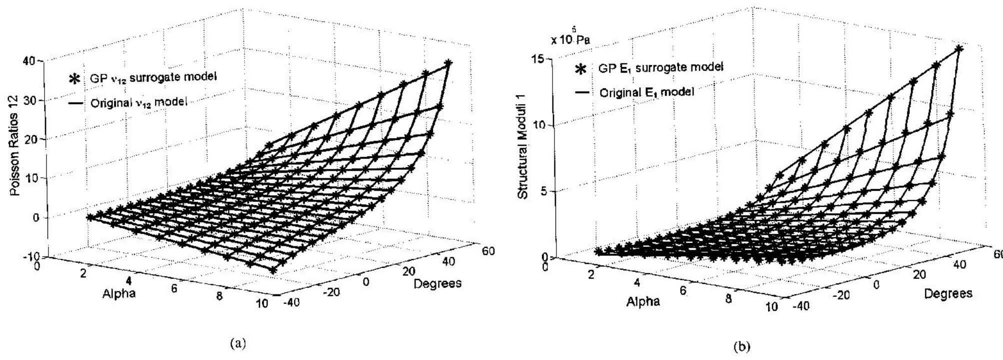


Fig. 5. (a) Comparison between GP-fitted E_1 with theoretical equation, (b) Comparison between GP-fitted ν_{12} with theoretical equation

was at the beginning of the GP development, the test run was performed as trial and error, in another words, the crossover rate and the mutation rate was set purely based on the judgement of the researcher. In this case, the crossover and mutation rates were set as 0.6, 0.5 for E_1 and 0.5, 0.5 for v_{12}.

The test of the GP code was proven to be successful. Figure 5 shows the fitness of GP model compares to the analytical equations: E_1 and v_{12} both gives a fitness of 6.95 and 17.18 respectively. Figure 6 illustrates the tree representation of the analytical solution of E_1 and v_{12} in comparison with the GP fitted model; it shows that within a predefined range of interest, the GP models are much more compact compare to the analytical ones.

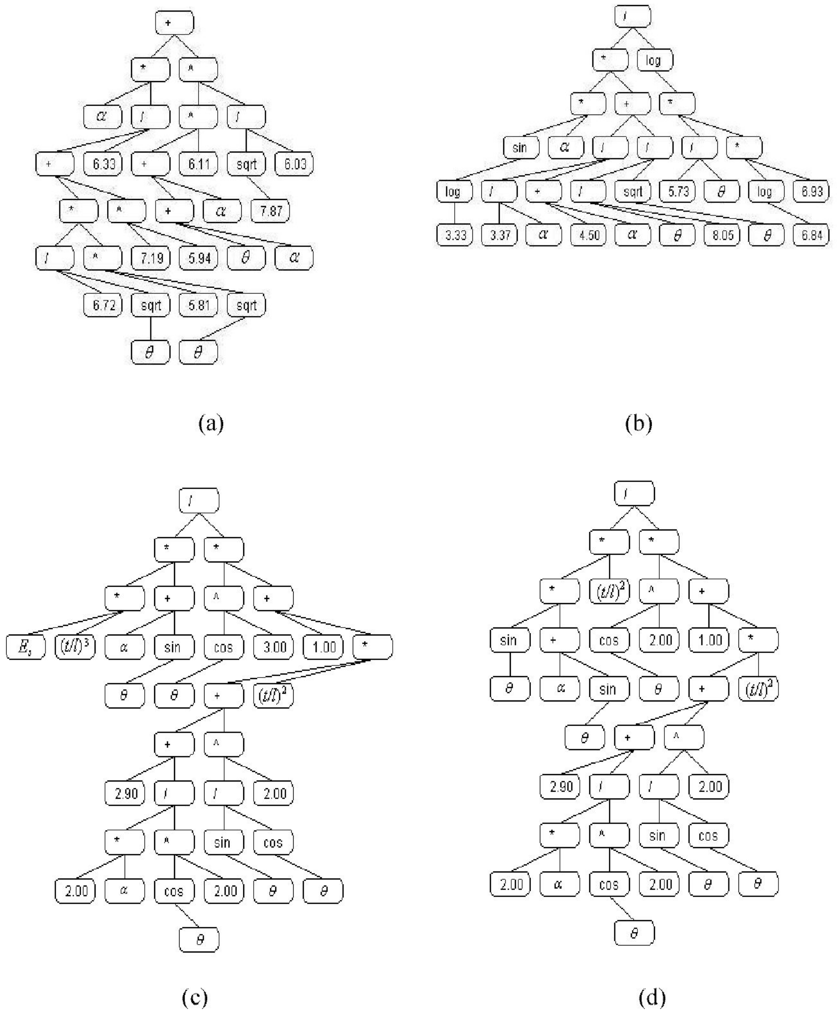
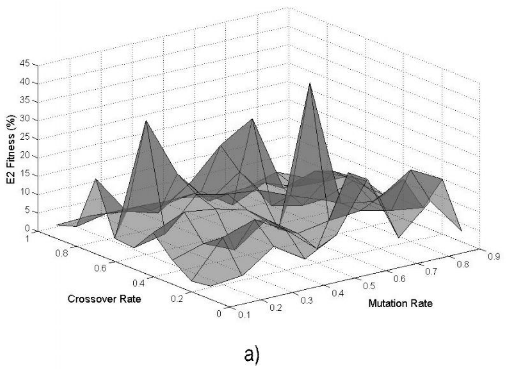
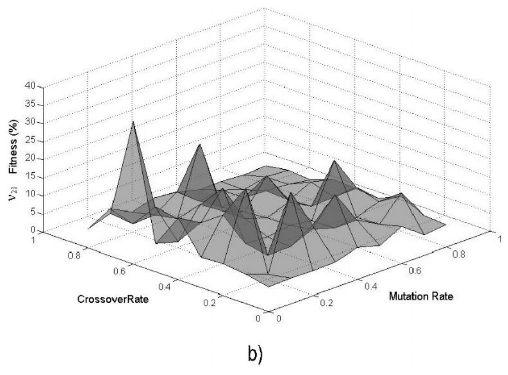


Fig. 6. (a) GP-fitted result of E_1 in tree representation, (b) GP-fitted result of v_{12} in tree representation, (c) Original model of E_1 in tree representation, (d) Original model of v_{12} in tree representation

As the GP development has become more mature, subsequent data (E_2, v_{21}, and G_{12}) were trained only at 2000 generation with 300 generation, while the crossover and mutation rates were determined by sweeping through all the combinations between 0.1 to 0.9, with each combination being iterated 10 times. The mutation method for E_2, v_{21}, and G_{12} were also modified to contain more varieties, *i.e.* for each mutation rate, it consists of 50% branch replacement, 30% of branch insertion, 10% of branch chopping, 30% of function node mutations, 10% of terminal node mutations.

Figure 7 shows the best fitness of all combinations for E_2, v_{21}, and G_{12}. The GP fitted models for each of these datasets were determined by the highest fitness GP model, as listed in Table 1.





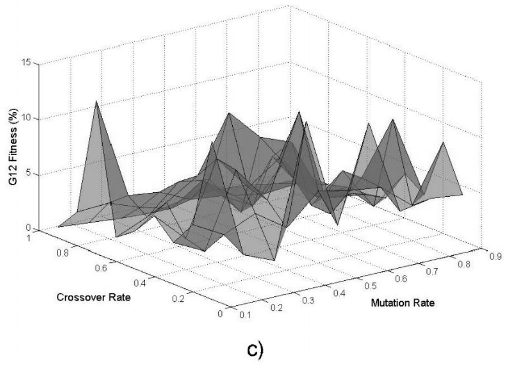


Fig. 7. Best Illness from the every combination of crossover and mutation, (a) E_2, (b) v_{21} (c) G_{12}

Table 1. List of best configuration for E_2, v_{21} and G_{12}

	Crossover Rate	Mutation Rate	Fitness (%)
E_2	0.4	0.6	42.72
v_{21}	0.7	0.1	35.02
G_{12}	0.1	0.1	14.20

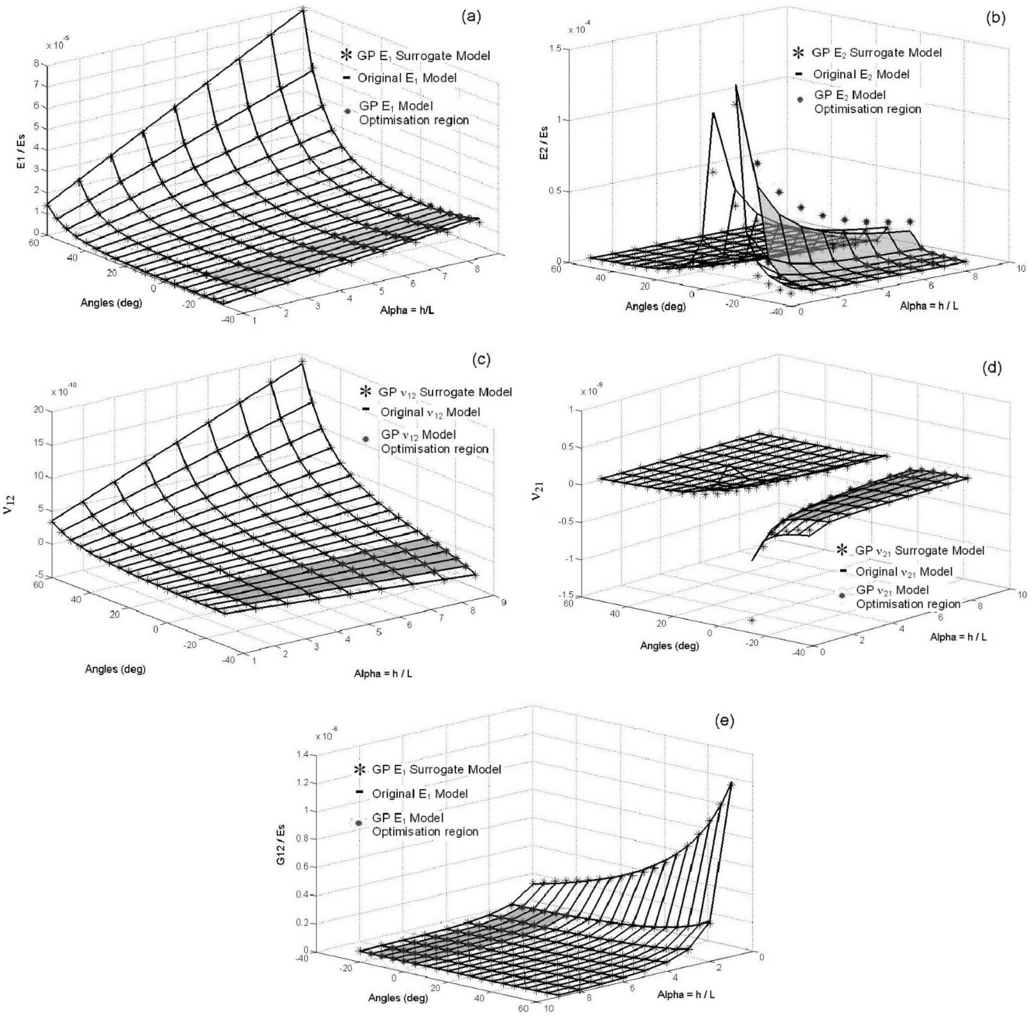


Fig. 8. (a) Optimisation region on E_1/E_s. (b) Optimisation region on v_{12}/E_s.(c) Comparison between GP-fitted E_2 with theoretical equation, (d) Comparison between GP-fitted v_{21} with theoretical equation, (e) Comparison between GP-fitted G_{12} with theoretical equation

The fitnesses of these GP model compares to the empirical equations are illustrated in Fig. 8. The shaded areas exhibit auxetic properties, and are used for the optimisation studies, which will be discussed later.

4. OPTIMISATION CASE STUDY

An example of a special orthotropic plate optimisation is illustrated as an application of the GP metamodel. The analytical calculation of the optimisation routine was done by Sequential Quadratic Programming (SQP) [24].

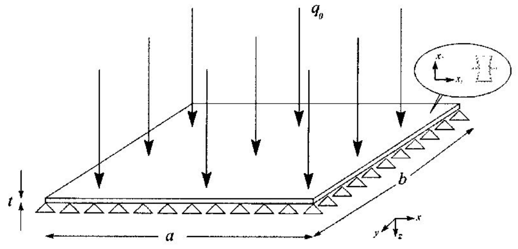


Fig. 9. Displacement of auxetic honeycomb plate

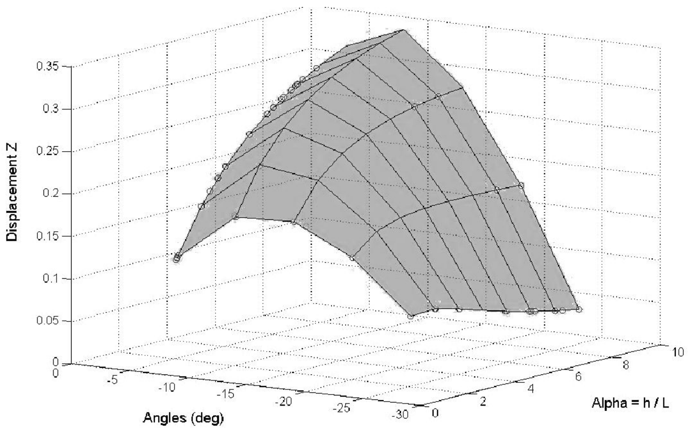


Fig. 10. Illustration of the plate displacement objective function with respect to the microstructure design parameters, 'o' indicates successful optimization solution

Figure 9 featured the plane stress optimisation example: A square plate simply-supported on all sides was made using periodical auxetic hexagonal honeycomb microstructure as described in the last section. A uniformly distributed load q_0 = 100 N was applied on the top surface. The displacement function of the plate ω is described as [5, 25]:

$$\omega = \frac{a^4}{\pi^4} \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} \frac{q_{mn}}{D_{mn}} \sin\left(\frac{\pi m x}{a}\right) \sin\left(\frac{\pi n y}{b}\right), \tag{9}$$

where

$$q_{mn} = \begin{cases} \dfrac{16 q_0}{\pi^2 m n} & \text{if } m, n = \text{even numbers} \\ \\ 0 & \text{if } m, n = \text{odd numbers}, \end{cases}$$

$$a = b = 10 \text{ m},$$
$$-0.5 \text{ m} \leq \omega \leq 0.3 \text{ m},$$
$$D_{mn} = D_{11}m^4 + 2\left(D_{21} + 2D_{33}\right)\left(mnR\right)^2 + D_{22}\left(nR\right)^4,$$
$$R = \frac{a}{b}, \qquad D_{ij} = \int_{-t/2}^{t/2} C_{ij} z^2 dz \quad i, j = 1,2,3 .$$

t is the measure ofthe plate thickness, which is significantly small in the plane stress case. The bounds of the microstructure geometry were set as: $-25° \leq \theta \leq -5°$, $2 \leq \alpha \leq 9$. The optimisation procedures were performed using the built-in command in the MATLAB65 optimisation toolbox (fmincon) [24]. The procedure serves the purpose of finding the minimum plate displacement ofthe constrained nonlinear multivariable displacement function ω and the corresponding microstructure configuration based on SQP. All values within $-25° \leq \theta \leq 60°$, $2 \leq \alpha \leq 9$ were used as initial guesses. Only feasible solutions from the optimisation were illustrated in Fig. 10.

5. CONCLUSION

The use of FE technique offers an easier and more efficient analysis for engineering designs. However, it is time consuming and very costly when implemented in engineering design optimisation.

The reduced-order FE model using an GP approach was developed here to overcome the difficulties by approximating the input-output relationships of the FE computation. The FE surrogate model in this work predicted the structural properties of solids composed of a hexagonal honeycomb cell with various geometric properties, calculated from empirical models.

The metamodel demonstrated a good generalisation in representing the FE model. It is worthwhile noticing that each metamodel developed is case dependent. In other words, it only represents one feature, *i.e.,* $(t/l) = 0.01$, of the honeycomb microstructure. A more powerful metamodel would require a larger number of simulation runs and more variety of microstructures.

The approach presented in this paper demonstrated its ability to provide a surrogate model based on analytical functions of the in-plane mechanical properties. Future work will be dedicated on applying the technique to design microstructures with more complicated topologies, where essentially FE analysis would be able to provide source models to determine the mechanical properties ofthe homogenised material.

References

[1] T. L. Lew, A. B. Spencer, F. Scarpa, K. Worden, A. Rutherford, and F. Hemez, *Identification of surface response function using genetic programming,* Proceedings of ISMA 3287-3299 (2004).

[2] R. R. Barton, *Metamodelling: A state of art review,* Proceedings of the 1994 Winter Simulation Conference, 1994, pp. 237-244.

[3] P. Bently, *Evolutionary design by computers - Chapter 1: An introduction of evolutionaiy design by computers,* 1999, Morgan Kaufmann Publisher Inc., San Francisco, USA.

[4] L. J. Gibson, and M. F. Ashby, *Cellular solids structure and properties,* 2nd Edition, Cambridge University Press, Cambridge, UK, pp. 93-172 (1997).

[5] J. M. Whitney, *Structural Analysis of Laminated Anistropic Plates,* Technomic Publishing Co. Ltd., USA, pp. 87-125 (1987).

[6] K. E Evans and A. Alderson, *Auxetic Materials: Functionals materials and Structures from lateral thinking!.* Advanced Materials **12**(9), 617-628 (2000).

[7] P. J. Stott, R. Mitchell, K. Alderson, and A. Alderson, *Auxetic materials - an introduction,* http://www.azom.com/Details.asp?ArticleID=l 67, January 2002.

[8] P. J. Stott, R. Mitchell, K. Alderson, and A. Alderson, *Auxetic materials - applications,* http://www.azom.com/Details.asp?ArticleID=l 68, January 2002.

[9] W. Yang, Z. M. Li, W. Shi, B. U. Xie, and M. Yang Bo, *Review on Auxetic Materials.* Journal of Materials Science **39**, 3269-3279 (2004).

[10] I. G. Masters and K. E. Evans, *Models for the elastic deformation of honeycombs.* Composite Structures, **35**(4), 403-422 (1996).

[11] F. Scarpa, and P. J. Tomlin, *On the transverse shear modulus of negative Poisson 's ratio honey-comb structures.* Fatigue and Fracture in Engineering Materials and Structures, 23, 717-720 (2000).

[12] M. Ruzzene, F. Scarpaand, and F. Soranna, *Wave beaming effects in two-dimensional cellular structures.* Smart Materials and Structures, **12**, 1-10 (2003).

[13] Scarpa, F., Burriesci, B. Smith, F. C. and B. Chambers, *Mechanical and electromagnetic behav-iour of auxetic honeycomb structures.* The Aeronautical Journal 107(1069), 175-183 (2003).

[14] J. R. Koza, *Genetic Programming On the Programming of Computers by Means of Natural Selection,* 7th. Printing, 1992, Massachusetts, USA: MIT Press, pp. 73-164.

[15] L. Panait and S. Luke, *Alternative Bloat Control Methods,* GECCO 2004, LNCS 3103, pp. 630-641.

[16] T. Soule and R. B. Heckendorn, *An Analysis of the Causes of Code Growth in Genetic Program-ming,* Genetic Programming and Evolvable Machines 3, 283-309 (2002).

[17] P. Fleming, *MSc. Lecture Notes on Optimisation and Search,* University of Sheffield, Automatic Control and Systems Engineering Department (2001).

[18] S. Luke and L. Spector, *A Comparison of Crossover and Mutation in Genetic Programming,* Pro-ceedings of the Second Annual Conference on Genetic Programming (GP-97), Morgan Kaufmann (1997).

[19] W. Banzhaf, and W. B. Langdon, *Some Consideration on the Reason for Bloat,* Genetic Program-ming and Evolvable Machines 3, 81-91 (2002).

[20] A. H. Watson and I. C. Parmee, *System Identification Using Genetic Programming,* Proceedings of 2nd International Conference on Adaptive Computing in Engineering Design and Control (PEDC), pp. 248-255 (1996).

[21] P. J. Angeline and D. B. Fogel, *An Evolutionaiy Program for the Identification of Dynamical Sys-tems,* Proceedings of the International Society of Optical Engineering (SPIE), vol. 3077, 409-417 (1997).

[22] L. Zhang, L. B. Jack, and A. K Nandi, *Fault Detection Using Genetic Programming,* University of Liverpool, Department of Electrical Engineering and Electronics (2004).

[23] A. P. Fraser, *Genetic Programming in C++ (A manual in Progress for gpc++, a public domain genetic programming system),* University of Salford, Cybernetics Research Institute (1994).

[24] The Mathworks Inc., *Optimisation Toolbox for Use with MATLAB: User Guide version 2,* The Mathworks Inc., MA, 1999, pp. 4.32-4.44.

[25] W. Marsden and D. J. Irving, *NAFEMS: How To - Analvse Composite,* NAFEMS Ltd., Glasgow (2002).