APPENDIX B

Simulating and Synthesizing Substructures Using Neural Network and Genetic Algorithms

Youhua Liu*, Rakesh K. Kapania* and Hugh F. VanLandingham†
*: Department of Aerospace and Ocean Engineering
†: Bradley Department of Electrical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061, USA

Summary

The feasibility of simulating and synthesizing substructures by computational neural network models is illustrated by investigating a statically indeterminate beam, using both a 1-D and a 2-D plane stress modelling. The beam can be decomposed into two cantilevers with free-end loads. By training neural networks to simulate the cantilever responses to different loads, the original beam problem can be solved as a match-up between two subsystems under compatible interface conditions. The genetic algorithms are succesfully used to solve the match-up problem. Simulated results are found in good agreement with the analytical or FEM solutions.

Introduction

Many aerospace structures are composed of substructures that are geometrically simpler and whose responses to loads are relatively easier to calculate. While a typical load-response relationship of the substructure can be established by utlizing neural network (NN) models trained from finite element analysis data, how to incorprate the NN models to get the characteristics of the whole system is a problem to be solved. We investigate the feasibility of synthesizing computational neural network models of aerospace substructures by studying some simple problems. For statically determinate structures in linear deformation, superposition can be applied and synthesizing of the NN models for the substructures is straightforward.

The statically indeterminate structures, on the other hand, must be split into statically determinate substructures first. When neural networks representing the substructures are trained, compatibility conditions needs to be enforced to obtain the unknown interface loads. But how to solve this match-up problem is not obvious. In this paper a global search technique, i.e. the genetic algorithms (GA), is used to treat the match-up problem as an optimization process of minimizing the differences between the interface quantities.

Since the major part of solving the statically determinate structures, i.e. specifying and trainning NNs, is also addressed in the statically indeterminate problems, only the latter will be addressed here.

A 1-D Beam and Its Analytical Solutions

The statically indeterminate beam problem shown in Fig. 1(a) has the following exact analytical solutions, if the beam is uniform with a bending rigidity of EI. The interface moment and force at point B:

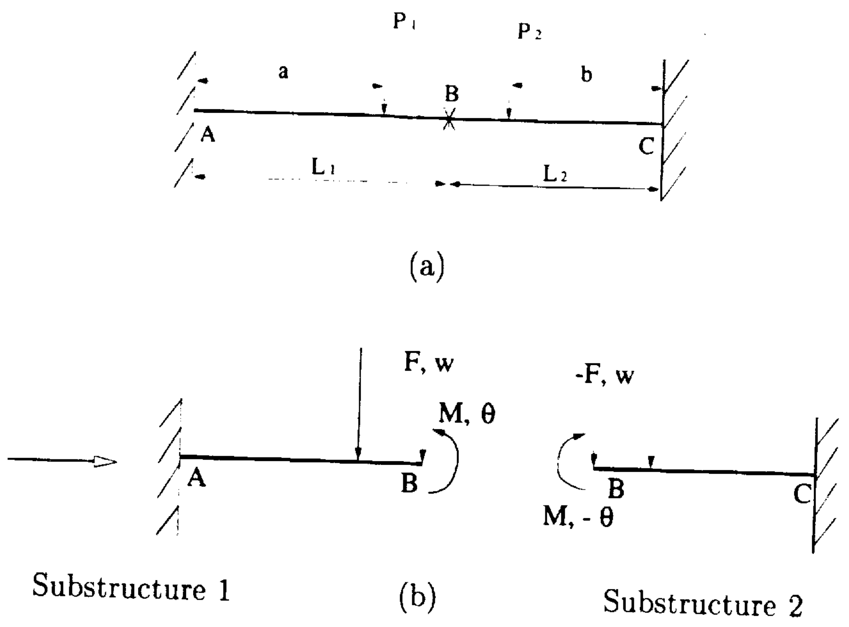


Fig. 1 A Statically Indeterminate Beam and its Decomposition

$$\begin{cases} M = \frac{(P_1 a^3 - P_2 b^3)(L_1 - L_2) - P_1 a^2 (L_1^2 - L_1 L_2 - 2L_2^2) + P_2 b^2 (2L_1^2 + L_1 L_2 - L_2^2)}{(L_1 + L_2)^3} \\ F = \frac{3(L_1 + L_2)(P_1 a^2 - P_2 b^2) - 2(P_1 a^3 - P_2 b^3)}{(L_1 + L_2)^3} \end{cases} \qquad (1)$$

The displacements at point B:

$$\begin{cases} w_B = \frac{P_1 a^3 L_2^2 (3L_1 + L_2) + P_2 b^3 L_1^2 (L_1 + 3L_2) - 3P_1 a^2 L_1 L_2^2 (L_1 + L_2) - 3P_2 b^2 L_1^2 L_2 (L_1 + L_2)}{6EI(L_1 + L_2)^3} \\ \theta_B = \frac{2(P_2 b^3 - P_1 a^3) L_1 L_2 + P_1 a^2 L_2 (2L_1^2 + L_1 L_2 - L_2^2) - P_2 b^2 L_1 (-L_1^2 + L_1 L_2 + 2L_2^2)}{6EI(L_1 + L_2)^3} \end{cases} \qquad (2)$$

Simulation of the 1-D Beam by NN and GA

Artificial Neural Networks (ANN), or simply Neural Networks (NN) are computational systems inspired by the biological brain in their structure, data processing and restoring method, and learning ability. The many desirable qualities of NN, such as a universal approximator, have enabled it being widely applied in engineering. For details of NN one can consult [1].

To simulate the structure shown in Fig. 1(a) by Neural Networks, the statically indeterminate beam can be split into two substructures as shown in Fig. 1(b). These substructures (cantilevers here) are statically determinate provided the interface forces at point B (bending moment M and shear force F)are known.

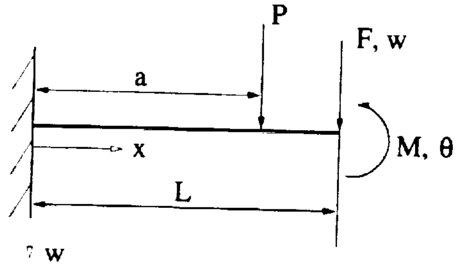


Fig. 2 Cantilever Beam to be Simulated by a NN

The two substructures in Fig. 1(b) can be represented by one single neural network simulating the cantilever beam with free-end moment and force, see Fig. 2. The network has three input variables and two output variables:

$$\{M, F, a\} \Longrightarrow NN \Longrightarrow \{w, \theta\} \tag{3}$$

The neural network will be trained by using training data generated from the analytical solutions

$$\begin{cases} \overline{w_B} = -\frac{1}{2}\overline{M} + \frac{1}{3}\overline{F} + \frac{1}{2}\overline{a}^2 - \frac{1}{6}\overline{a}^3 \\ \overline{\theta_B} = \overline{M} - \frac{1}{2}\overline{F} - \frac{1}{2}\overline{a}^2 \end{cases} \tag{4}$$

where all the quantities have been nondimesionalized:

$$\overline{w_B} = \frac{EIw_B}{PL^3}, \overline{\theta_B} = \frac{EI\theta_B}{PL^2}, \overline{M} = \frac{M}{PL}, \overline{F} = \frac{F}{P}, \overline{a} = \frac{a}{L}.$$

The neural network used was a MLP (multi-layer perceptron, also called feed-forward NN) with 1 hidden layer and 10 hidden layer neuros. There are 180 of training data sets, a combination of 6×6×5 variations in $\overline{M}$, $\overline{F}$ and $\overline{a}$ respectively.

Training is a critical part of a NN simulation, taking significant time and energy. But it seems that the most important thing would be to choose a proper training algorithm. A good algorithm can train a network in minutes, compared to hours of computation with a poor turnout when an inappropriate algorithm is used.

For the present problem, it is found that the MATLAB program for training feed-forward NN with Levenberg-Marquardt, *trainlm*, gives the best results. Other programs, such as *trainbpa* (training feed-forward network with back-propagation plus adaptive learning), could not give good enough performances when the number of input variables is 3 or more.

When the NN representing the substructures has been trained, the interface moment M and force F at point B can be obtained by enforcing the following compatibility and equilibrium conditions

$$M_1 = M_2; \theta_{B1} = -\theta_{B2}; F_1 = -F_2; w_{B1} = w_{B2}.$$

where the sub-index 1 and 2 indicate substructure 1 and 2 respectively.

First the following approach was tried:

$$\{M, F\} \longrightarrow \{M, F, a\} \Longrightarrow NN1 \Longrightarrow \{w, \theta\}$$

$$\longrightarrow \{w, \theta, a\} \Longrightarrow NN2 \Longrightarrow \{M, F\}$$

where NN1 is the neural network of the same scheme as in Eqn. (3), while NN2, having inputs of $\{w, \theta, a\}$ and outputs of $\{M, F\}$, is the reverse network to NN1. This approach did not work since the pair of values $\{M, F\}$ would not converge. It was seen that even those iterations which started with an initial $\{M, F\}$ same as the exact soloutions of Eqn. (4) and its reverse form

$$\begin{cases} \overline{M} = 6\overline{w_B} + 4\overline{\theta_B} - \overline{a}^2 + \overline{a}^3 \\ \overline{F} = 12\overline{w_B} + 6\overline{\theta_B} - 3\overline{a}^2 + 2\overline{a}^3 \end{cases} \tag{5}$$

did not converge mainly because Eqn. (5) magnifies the error between the iterations enormously.

To solve the match-up problem in another way the following optimization process can be formulated

$$Min_{M,F}E(M, F) \tag{6}$$

where

$$E(M, F) = \{\frac{1}{2}[(w_{B2} - w_{B1})^2 + (\theta_{B2} - \theta_{B1})^2]\}^{\frac{1}{2}}. \tag{7}$$

This optimization problem can be solved by using genetic algorithms (GA), the derivative-free stochastic methods based on the concepts of natural selection and evolutionary processes. The GAs were first proposed by Holland in 1975 ([2]), and have been widely used as a general-purpose optimization tool.

Major components of GA are composed of *encoding scheme, fitness evaluation, fitness sharing, elitism, selection, crossover* and *mutation*. Details of GA can be found in [2]. Specifications of the GA used in the present study are described as follows:

(a) The values of $\overline{M}$ and $\overline{F}$ are transformed into a 22-bit binary string with one 11-bit part related to $\overline{M}$ and the other 11-bit part dedictated to $\overline{F}$. One bit of the string is called a chromosome or a gene. In each of the 11-bit string the left-most bit represents the sign, with 0 representing +, and 1 representing -. The remaining 10-bit string represents the magnitude times 10^{-4}. So the smallest non-zero values recognizable in this GA is ± 0.0001.

(b) The whole population consists of n_p members, each of which is represented by a 22-bit binary string as defined in (a). For the present study, n_p=10.

(c) The fitness represents the degree of merit of a member in the population. A member with a higher fitness has a larger chance of being chosen as a parent for the next generation. It can be defined as: $f_i = \frac{1}{E(M_i,F_i)}$, where E is defined in Eqn. (7).

(d) The new generation keeps a few members of the old generation which have the best fitness values. For the present study the number of this part of members is: $n_e \cong 0.3n_p$.

(e) The other members of the new generation are produced by one or two parents from the old generation. The probability of a member in the old generation to be chosen as a parent is fitness-related: $p_i = \frac{f_i}{\sum_{k=1}^{n_p} f_k}$.

(f) The new member can be produced by crossover of two chosen parents at a probability of p_{cross}. The crossover is a process where portions of the parents' chromosome strings are exchanged. In the present study one-point and two-point crossovers are used, in which the chromosome string are divided into two and three portions respectively. The division of portions is based on a random process.

(g) If there is no crossover, the new member would inherit all the chromosomes of a chosen parent. But all its chromosomes and the results from (f) should undergo a mutation (i.e. 0 becomes 1 or 1 becomes 0) in a given probability of p_{muta}.

(h) The criterion of terminating GA is when $Min_i E(M_i, F_i) \leq \epsilon$. Otherwise, when the number of generations is too large, or $n_{gen} \geq \Lambda$, the GA search has to be stopped.

The values of probabilities p_{cross} and p_{muta} have profound effects on the performance of the GA. In the first case, they are specified as: $p_{cross} = 0.5; p_{muta} = .1; \epsilon = 0.0015; \Lambda = 2000$. This formulation works quite well but has a probability of 10~30% of not meeting the terminating criterion, therefore some of the results would not be good enough. In these cases, another formulation with p_{cross} and p_{muta} as functions of n_{gen} were used and improvements were seen.

A 2-D Beam and Its Solution

The above method of simulating and synthesizing substructures were used to solve a 2-D problem. As shown in Fig.3, a 2-D beam clamped at both ends has a parabolically distributed load P, and the stresses and deformation at plane A-A are to be evaluated.

The shear and normal stress at A-A were assumed to have the following distribution

$$\tau_{xy} = a_2(1 - \overline{y}^2), \sigma_x = b_0 + b_1\overline{y} + b_2\overline{y}.^2 \qquad (8)$$

where $\overline{y} = \frac{2y}{h}$.

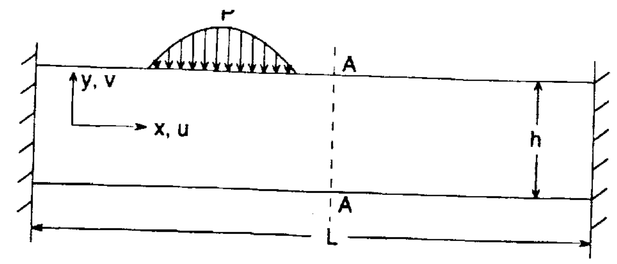


Fig. 3 A 2-D Beam under Parabolic Load

Imagine that the beam is split at A-A and become two cantilevers with a shear and normal load. Neural networks can be trained by taking FEM solutions of the node displacements as the targets and combinations of a_2,b_0,b_1 and b_2 as the inputs:

$$\{a_2, b_0\} \Longrightarrow NN1 \Longrightarrow \{u_i, v_i \text{ at A-A, left cantilever}\},$$
$$\{b_1, b_2\} \Longrightarrow NN2 \Longrightarrow \{u_i, v_i \text{ at A-A, left cantilever}\},$$
$$\{a_2, b_0\} \Longrightarrow NN3 \Longrightarrow \{u_i, v_i \text{ at A-A, right cantilever}\},$$
$$\{b_1, b_2\} \Longrightarrow NN4 \Longrightarrow \{u_i, v_i \text{ at A-A, right cantilever}\}.$$

Then the interface stresses and deformation can be obtained by solving

$$Min_{a_2,b_0,b_1,b_2} E(a_2, b_0, b_1, b_2) \tag{9}$$

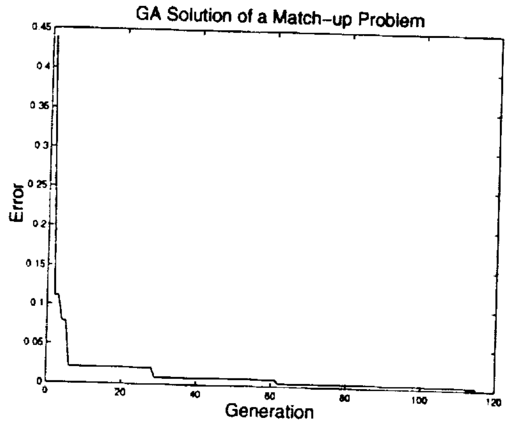
where

$$E(a_2, b_0, b_1, b_2) \sim \sum_i [(u_i^{LC} - u_i^{RC})^2 + (v_i^{LC} - v_i^{RC})^2]^{\frac{1}{2}} \tag{10}$$

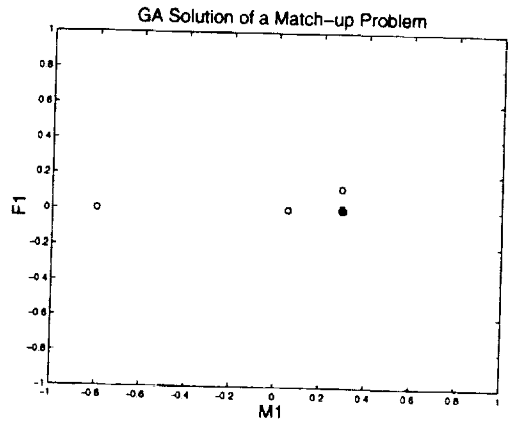
in which LC and RC indicates the *left cantilever* and *right cantilever* respectively. The GA used was similar to that for the 1-D problem. A major difference is that since now we have four unkowns, each of them was allocated only a 6-bit string compared with a 11-bit string in the 1-D problem. As will be shown in the next section, satisfactory results can still be obtained.

Some Results

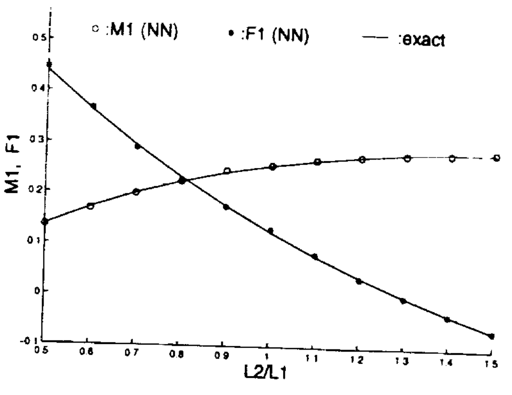
For the 1-D problem the following cases were used: $\frac{a}{L_1} = 0.8$; $\frac{b}{L_1} = 0.8$; $\frac{P_2}{P_1} = 0.8$; $\frac{L_2}{L_1} = 0.5 \sim 1.5$. Results are shown in Fig. 4, in which part (a) is the search history of the GA for the first point ($L_2/L_1 = 1.5$), (b) is the distribution of the population on the search plane when GA is terminated, (c) is the comparison of simulaions and the exact solutions of $\overline{M}_1$ and $\overline{F}_1$, and (d) is the comparison of $\overline{w}_{B_1}$ and $\overline{\theta}_{B_1}$. Here the subscrirpt 1 indicates that the nondimensionalization is based on L_1.



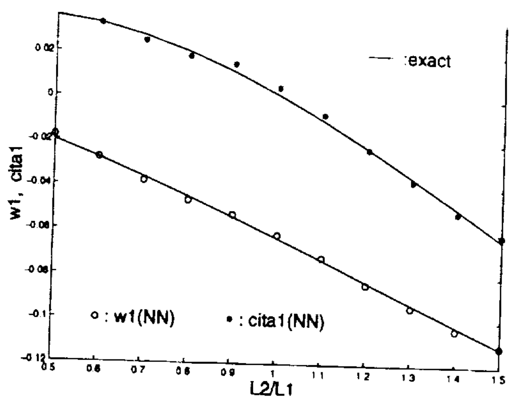
(a)



(b)



(c)



(d)

Fig. 4 Calculation of forces and displacements at point B

For the 2-D problem the simulated stress distributions and deformation at interface A-A compared with FEM results are shown in Figs. 5 and 6. We used a 8-node rectangular element ([3]) to do the finite element analysis, and the mesh is shown in Fig. 6.

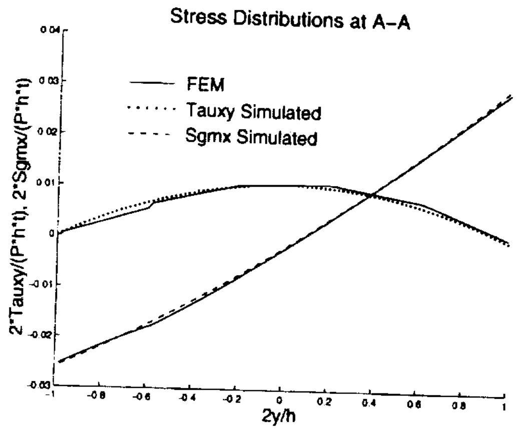
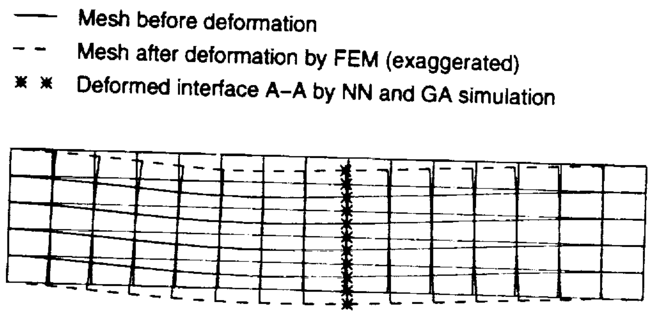


Fig. 5 Stress Distributions at A-A



Mesh before deformation
Mesh after deformation by FEM (exaggerated)
Deformed interface A-A by NN and GA simulation

Fig. 6 Mesh and Deformations

6. Conclusion

The feasibility of simulating substructures by computational neural network models is studied. Synthesizing the substructures together is proved to be possible by formulating the match-up problem into an optimization process which can be solved using genetic algorithms.

It was observed that the synthesizing problem cannot be solved by simple iterations using the direct and reverse NNs representing the substructures.

Although the study was carried out on simple beam structures, it could be readily applied to more complicated cases. For the use of NN method, as a universal approximator the NN find no difference on whether the system to be simulated is linear or highly non-linear. Of course if the input or output variables are large in number it would be harder to find a algorithm to train the NN properly. If there are more interface parameters between the substructures, it is not known whether the GA can still work or other optimization techniques should be used. These are problems to be further studied.

References

1. M. T. Hagen, H. B. Demuth and M. Beale (1996), Neural Network Design, PWS Publishing Co.
2. J.-S. R. Jang, C.-T. Sun and E. Mizutani (1997), Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence, Prentice Hall, Inc.
3. T. Y. Yang (1985), Finite Element Structural Analysis, Prentice Hall, Inc.