Available online at www.sciencedirect.com
SCIENCE DIRECT
ELSEVIER
Mechanical Systems and Signal Processing 20 (2006) 1819–1831
Mechanical Systems and Signal Processing
www.elsevier.com/locate/jnlabr/ymssp

Identification of response surface models using genetic programming

T.L. Lew[a,*], A.B. Spencer[a], F. Scarpa[a], K. Worden[a], A. Rutherford[b], F. Hemez[b]

[a]*Dynamics Research Group, Department of Mechanical Engineering, University of Sheffield, Mappin Street, Sheffield S1 3JD, UK*
[b]*Weapon Response Group, Los Alamos National Laboratory, Los Alamos, New Mexico, USA*

Received 12 November 2004; received in revised form 22 November 2005; accepted 5 December 2005
Available online 19 January 2006

Abstract

There is a move in modern research in Structural Dynamics towards analysing the inherent uncertainty in a given problem. This may be quantifying or fusing uncertainty models, or can be propagation of uncertainty through a system or calculation. If the system of interest is represented by, e.g. a large Finite Element (FE) model 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 objective 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 both univariate and multivariate problems with both computational and experimental data.

Keywords: Surrogate/replacement model; Response surface models; Symbolic regression; Genetic programming

0. Introduction

The design and optimisation of structures and systems has become increasingly sophisticated and challenging in recent years by having to face more inherent uncertainties. Alongside Finite Element Methods (FEM), there are many other packages that help in reducing the modelling workload. However, in terms of understanding the effects of uncertainty the computational effort for running many analyses remains non-trivial and this raises serious problems for uncertainty propagation methods which rely on sampling from the population of parameters like Monte-Carlo methods. In addition to this, the query-and-response procedure is often by trial and error and the researcher may never fully understand the input–output relationship and therefore never be able to identify the best setting for input values.

*Corresponding author. Tel.: +44 114 222 7721; fax: +44 114 222 7890.
E-mail address: t.l.lew@Sheffield.ac.uk (T.L. Lew).

0888-3270/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ymssp.2005.12.003

The basic approach to this problem is to develop an approximation of the analyses that is more efficient to execute, and to obtain more 'insight' into the relationship between the input and output. This *approximate/surrogate* model of the original is also sometimes called a *metamodel*.

The usual methods for creation of a metamodel involve choosing a function or basis of functions in order to represent the data in the most appropriate way. Whether this is linear regression, quadratic or even a Fourier series, all require some form of background knowledge of the experimental/simulation data that one is attempting to fit. Once the proper function is chosen, one proceeds to a series of analyses which determine the 'best' values for the free parameters of the model.

A traditional metamodelling approach often involves using a multivariate response surface model. A general response surface method would be first to screen the data to reduce the set of design variables that are most influential to the output. Then with the chosen function, one would start to perform a general fitting, followed by some refining procedures. If the researcher decided that such a function does not give a promising fit, he/she may then consider the option of using different functions.

The research in traditional metamodelling reached its mature state some time ago, but there are still some known problems, such as difficulties with the polynomial basis functions, problems in trying to fit a highly non-linear model, explosion of coefficients, etc.

In short, the creation of the surrogate model requires both the discovery of the correct function that fits the data and the appropriate numeric coefficients of the function. The problem can be seen as finding a proper function, in symbolic form, which fits a given finite sample of data, or *symbolic regression* [1].

An alternative approach to the problem is to adopt an evolutionary algorithm, such as genetic programming (GP). Researchers and software developers apply computers to a wide variety of design problems. Rather than spending time and effort to choose the proper function for every new problem, a preferable choice would be to use an algorithm proven through extensive trials to be reusable and robust [2]. Genetic programming is a subclass of evolutionary computation techniques but unlike the other evolutionary algorithms that evolve numerical values; it evolves functions as a solution for a given problem.

In this work, the genetic programming was developed using the Java programming language [3] and applied to fit both computational and experimental data. The computational data were obtained according to the theoretical formulae defining the structural modulus and Poisson's ratio of the hexagonal honeycomb [4]. The experimental data was from Los Alamos National Laboratory, where a 9 DOF mass-spring system was studied for damage detection purposes [5]. (These examples are rather arbitrary, they were chosen simply as data sets of appropriate complexity to illustrate the effectiveness of the Genetic Program for fitting response surfaces via symbolic regression.) The results also showed that Genetic Programming could identify the most influential design variables with respect to the output.

1. Theory of genetic programming

Genetic Programming (GP) shares its dominant concept with the more well-known Genetic Algorithm (GA) in that it allows a population of solutions to a given problem to evolve under a model of Darwinian natural selection and iteratively improve in search of the optimum. However, GP increases the functionality (and complexity) of the algorithm by also allowing the structure of the solution to undergo adaptation. The structure of an individual in a GP is typically a hierarchical computer program or mathematical function of dynamically varying size and shape. It is the latter structure which is appropriate here. Each individual is typically encoded as a *tree* structure. Mathematically, a tree is simply a directed graph initiated from a single root vertex or node. Each node has connections to a number of 'child' branch nodes, but the characteristic of a tree is that connections back beyond the immediate 'parent' are forbidden. By using such a tree structure, the current approach excludes the possibility of recursion in the function. Two types of node are possible in the individual corresponding to the two types of terms which can occur in a symbolic expression. The first type of node is a *terminal* node, this is analogous to an object which does not depend on any arguments like a constant or one of the input variables of the expression. Such a node terminates a branch of the tree. The second type of node is a function node which is analogous to a function or operator which requires arguments, the arguments are connected to the function node via branches and these can be from other function nodes or from terminals. The number of branches into a function node is clearly the number of arguments taken by the function and