Computational intelligence methods for helicopter loads estimation

Valdes, Julio J.; Cheung, Catherine; Wang, Weichao

Publisher's version / Version de l'éditeur:
https://doi.org/10.1109/IJCNN.2011.6033451
Neural Networks (IJCNN), The 2011 International Joint Conference on, pp. 1864-1871, 2011-08-03

NRC Publications Record / Notice d'Archives des publications de CNRC:
https://nrc-publications.canada.ca/eng/view/object/?id=e1722e05-3ddc-4f06-853d-4dbb8e30dde1
https://publications-cnrc.canada.ca/fra/voir/objet/?id=e1722e05-3ddc-4f06-853d-4dbb8e30dde1

Computational Intelligence Methods for Helicopter Loads Estimation

Julio J. Valdés *Senior Member IEEE,* Catherine Cheung and Weichao Wang

***Abstract*— Accurately determining component loads on a helicopter is an important goal in the helicopter structural integrity field. While measuring dynamic component loads directly is possible, these measurement methods are not reliable and are difficult to maintain. This paper explores the potential of using computational intelligence methods to estimate some of these helicopter dynamic loads. Thirty standard time-dependent flight state and control system parameters were used to construct a set of 180 input variables to estimate the main rotor blade normal bending during forward level flight at full speed. Unsupervised nonlinear mapping was used to study the structure of the multidimensional time series from the predictor and target variables. Based on these criteria, black and white box modeling techniques (including ensemble models) for main rotor blade normal bending prediction were applied. They include neural networks, local linear regression and model trees, in combination with genetic algorithms based on residual variance (gamma test) for predictor variables selection. The results from this initial work demonstrate that accurate models for predicting component loads can be obtained using the entire set of predictor variables, as well as with smaller subsets found by computational intelligence based approaches.**

I. Introduction

THE operational loads experienced by rotary-wing aircraft are complex due to the dynamic rotating components operating at high frequencies. As a result of the large number of load cycles produced by the rotating components and the wide load spectrum experienced from a rotary-wing aircraft's broad range of manoeuvres, the fatigue lives of many components can be affected by even small changes in loads. While measuring dynamic component loads directly is possible, these measurement methods are not reliable and are difficult to maintain. Therefore, an accurate and robust process to estimate these loads indirectly would be more practical and efficient.

Much research has been carried out using machine learning methods to model operational loads experienced by fixed-wing aircraft structure [19], [6]. In the case of rotary-wing aircraft, the loading spectrum experienced by the airframe structure is significantly more complex since the dynamic rotating components operate at frequencies several orders of magnitude higher than for fixed-wing aircraft. There have been a number of attempts at estimating these loads on the helicopter indirectly with varying degrees of success [14]. While the use of artificial neural networks for this problem is not uncommon [7], [12], an in-depth study of the applicability of neural networks or other machine learning methods for helicopter loads estimation is lacking.

This paper describes the preliminary study exploring the potential of using computational intelligence methods for helicopter loads estimation. The specific problem was to estimate one of the loads in the main rotor system of the Australian Army Black Hawk helicopter (shown in Fig. 1) using only flight state and control system (FSCS) parameters as input variables. The objectives of this work were as follows: 1) to determine if the problem could be solved by computational intelligence techniques and generate accurate models for the target parameter; 2) to extract information from the data that could enable a better understanding of the physical process of the input/output relationship; and 3) to identify and eliminate irrelevant and/or noisy information from the input variables and create much simpler but well-behaved models to use as input.

Fig. 1. Australian Army Black Hawk helicopter.

II. The Helicopter Loads Estimation Problem

Operational requirements are significantly expanding the role of military helicopter fleets in many countries. This expansion has resulted in helicopters flying missions that are beyond the design usage spectrum, which was originally used to life fatigue critical components. Due to this change

This work was supported in part by Defence Research and Development Canada (13ph13). Access to the data was granted by the Defence Science and Technology Organisation.

Julio J. Valdés is with the National Research Council Canada, Institute for Information Technology, 1200 Montreal Rd. Bldg M50, Ottawa, ON K1A 0R6, Canada (phone: 1-613-993-0887; fax: 1-613-993-0215; email: julio.valdes@nrc-cnrc.gc.ca).

Catherine Cheung is with the National Research Council Canada, Institute for Aerospace Research, 1200 Montreal Rd. Bldg M14, Ottawa, ON K1A 0R6, Canada (phone: 1-613-998-1541; fax: 1-613-952-7136; email:catherine.cheung@nrc-cnrc.gc.ca).

Weichao Wang is with the National Research Council Canada, Institute for Aerospace Research, 1200 Montreal Rd. Bldg M03, Ottawa, ON K1A 0R6, Canada (phone: 1-613-949-5991; fax: 1-613-952-7136; email:winston.wang@nrc-cnrc.gc.ca).

in usage, there is a need to monitor individual aircraft usage to compare with the original design usage spectrum in order to more accurately determine the life of critical components. One of the key elements to tracking individual aircraft usage and calculating component retirement times is accurate determination of the component loads.

The rotor system components and attachments are some of the most fatigue-critical structural components on a helicopter. Direct measurement of the dynamic loads in these areas has traditionally been accomplished through slip rings or telemetry systems, however, these techniques are difficult to implement and are often unreliable. While advances in sensor technologies in the past decade have produced compact, lightweight, and economical devices, high equipment costs and large data storage requirements still make direct load monitoring impractical.

A. Flight conditions and parameters of interest

One of the main goals of this research was to determine if the dynamic loads on the helicopter could be predicted solely from the FSCS parameters, as these parameters are already recorded by the flight data recorder found on most helicopters. Thirty FSCS parameters that were used as input variables are listed in Table I.

While many of the helicopter dynamic loads are of interest, this study selected only one of these loads as the target parameter: the main rotor blade normal bending (MRNBX). Similarly while there were over 50 flight conditions performed during the flight loads survey, the results from only one manoeuvre are presented: forward level flight at full speed (V_H). Even though this flight condition is a simple steady state manoeuvre, the purpose of this preliminary study was to explore the applicability of computational intelligence methods for this problem and so this flight condition was deemed an appropriate starting point.

TABLE I
FLIGHT STATE AND CONTROL SYSTEM PARAMETERS.

Parameter	Parameter
Air speed (boom)	Directional pedal position
Vertical acceleration, load factor at CG	Collective stick position
Angle of attack (boom)	Stabilator position
Sideslip angle (boom)	% of max main rotor speed
Pitch altitude	Retreating tip speed
Pitch rate	Main rotor speed
Pitch acceleration	Tail rotor speed
Roll attitude	No.1 Engine torque
Roll rate	No.2 Engine torque
Roll acceleration	No.1 Eng power lever (temp)
Heading	No.2 Eng power lever (temp)
Yaw rate	Barometric altitude (boom)
Yaw acceleration	Temperature (Kelvin)
Longitudinal stick/cyclic position	Altitude (height density)
Lateral stick/cyclic position	Barometric rate of climb (boom)

B. The Data

The data used for this work were obtained from a S-70A-9 Australian Army Black Hawk (UH-60/HH-60 variant) flight loads survey conducted in 2000 in a joint flight loads measurement program between the United States Air Force and the Australian Defence Force [9]. During these flight trials, 65 hours of useable flight test data were collected for a number of different steady state and transient flight conditions at several different altitudes and aircraft configurations. The strain data from the Black Hawk flight load survey were captured by 321 strain gauges, with 249 gauges on the airframe and 72 gauges on dynamic components. The airframe gauges were mounted on areas prone to cracking and structural distress, primarily in the upper cabin, tail cone, tail pylon, horizontal stabilator, external stores support system, and main rotor pylon. Accelerometers were installed to measure accelerations at several locations on the aircraft and other sensors captured flight state and control system parameters. The parameters were recorded at one of three sampling frequencies: 52 Hz, 416 Hz, and 832 Hz. Full details of the instrumentation and flight loads survey are provided in [9].

III. Computational Intelligence Methods

A. Time Series Analysis in High Dimensional Spaces

In time series analysis coming from dynamic systems, phase space methods play an important role. However, when the embedding dimension of the series is large and/or the number of significant time lags exceeds 3 standard graphical representations become unfeasible. The same situation arises when the number of series that describe the state of a system is large, as in the present case. However, nonlinear transformation of these spaces into lower dimensional ones becomes an alternative, especially if it is possible to assess the accuracy with which the transformed space preserves the internal structure of the information contained in the original space.

The construction of a smaller feature space can be performed via a nonlinear transformation that maps the original set of N-dimensional objects under study $\mathcal{O}$ into another space $\hat{\mathcal{O}}$ of smaller dimension $\hat{d} < N$. This approach has been used for data representation and Visual Data Mining (knowledge and data exploration) [22]. There are essentially three kinds of spaces generally sought [23]: *i)* spaces preserving the structure of the objects as determined by the original set of attributes or other properties (unsupervised approach), *ii*) spaces preserving the distribution of an existing class or partition defined over the set of objects (supervised approach), and *iii*) hybrid spaces. Data structure is one of the most important elements to consider and it can be approached by looking at *similarity relationships* [2], [1] between the objects, as given by the set of original attributes [22]. From this point of view, transformations φ can be constructed that minimize error measures of information loss [20], based on similarities or distances.

If $\delta(\vec{x}, \vec{y})$ is a dissimilarity measure between any two objects $\vec{x}\vec{y} \in \mathcal{O}$, and $\zeta(\hat{\vec{x}}, \hat{\vec{y}})$ is another dissimilarity measure defined on objects $\hat{\vec{x}}, \hat{\vec{y}} \in \hat{\mathcal{O}}$ $(\hat{\vec{x}} = \varphi(\vec{x}), \hat{\vec{y}} = \varphi(\vec{y}))$, a

frequently used error measure associated to the mapping φ is the Sammon error [20]:

$$\mathcal{S}_e = \frac{1}{\sum_{\vec{x}\neq\vec{y}} \delta(\vec{x},\vec{y})} \frac{\sum_{\vec{x}\neq\vec{y}} \left(\delta(\vec{x},\vec{y}) - \zeta(\hat{\vec{x}},\hat{\vec{y}})\right)^2}{\delta(\vec{x},\vec{y})} \quad (1)$$

It is possible to minimize Eq. 1 with a wide variety of methods, ranging from classical optimization to computational intelligence-based techniques. Here the Fletcher-Reeves algorithm (FR) is used, which is a well known technique used in deterministic optimization [15]. It assumes that the function to optimize can be approximated as a quadratic form in the neighborhood of a N dimensional point $\mathbf{P}$ and exploits the information contained in the partial derivatives of the objective function. This kind of optimization is prone to local extrema entrapment, therefore it is recommended to try different random initial parameter vectors.

The accuracy of the mapping depends on the final error obtained in the optimization process. Explicit mappings can however be obtained from these solutions using neural networks, genetic programming, and other techniques. In general φ is a nonlinear function and in order to compare results from transformations obtained with different algorithms or different initializations, a canonical representation is preferred. It can be obtained by performing a principal component transformation $\mathcal{P}$ after φ, so that the overall transformation is given by the composition

$$\hat{\varphi} = (\varphi \centerdot \mathcal{P}) \quad (2)$$

referred to as the canonical mapping. Since $\mathcal{P}$ does not change the dimension of the new space, an advantage of the canonical mapping is that it simplifies the comparison of different solutions. It also contributes to the interpretability of the new variables, as they have a monotonic distribution of the variance. The images of the mapped objects can be used for the construction of a 3D model using virtual reality for visual data mining and data exploration, as it is done in this paper.

B. Gamma Test (Residual Variance) Analysis

The Gamma test is an algorithm developed by [10], [21], [4] as a tool to aid in the construction of data-driven models of smooth systems. It is a technique aimed at estimating the level of noise (its variance) present in a dataset. Noise is understood as any source of variation in the output (target) variable that cannot be explained by a smooth transformation (model) relating the output (predicted or dependent variable) with the input (predictor) variables.

The fundamental information provided by this estimate is whether it is hopeful or hopeless to find (fit) a smooth model to the data. Here a 'smooth' model is understood as one in which the first and second partial derivatives are bounded by finite constants for every point of observation. The gamma estimate indicates whether it is possible to explain the dependent variable by a smooth deterministic model involving the observed input and output variables. Model search is a costly, time consuming data mining operation. Therefore, knowing beforehand that the information provided by the input variables is not enough to build a smooth model is very helpful. It may give an indication that more explanatory variables should be incorporated to the data or that the underlying model may be very complex. If for a given dataset, the gamma estimates are small, it means that a smooth deterministic dependency can be expected. It also gives a threshold in order to avoid overfitting and it can give an indication of how many observations are minimally required in order to build a model which performs with that mean squared error. Overall it gives a measure of the quality of the data.

Let $\mathcal{S}$ be a system described in terms of a set of variables and with $y \in \mathbb{R}$ being a variable of interest, potentially related to a set of m variables $\overleftarrow{x} \in \mathbb{R}^m$ expressed as

$$y = f(\overleftarrow{x}) + r \quad (3)$$

where f is a smooth unknown function representing the system, $\overleftarrow{x}$ is a set of predictor variables and r is a random variable representing noise or unexplained variation.

Despite f being an unknown function, under some assumptions it is possible to estimate the variance of the residual term (r) using available data obtained from $\mathcal{S}$. This will give an indication about the possibility of developing models for y based on the information contained in $\overleftarrow{x}$. Among the most important assumptions are:

- The function f is continuous within the input space.
- The noise is independent of the input vector $\overleftarrow{x}$.
- The function f has bounded first and second partial derivatives.

Let $\overleftarrow{x}_{N\lfloor i,k\rfloor}$ denote the k-th nearest neighbor of $\overleftarrow{x}_i$ in the input set $\{\overleftarrow{x}_1, \cdots, \overleftarrow{x}_M\}$. If p is the number of nearest neighbors considered, for every $k \in [1,p]$ a sequence of estimates of $\mathbf{E}\left(\frac{1}{2}(y'-y)^2\right)$ based on sample means is computed as

$$\gamma_M(k) = \frac{1}{2M}\sum_{i=1}^{M} |y_{N\lfloor i,k\rfloor} - y_i|^2 \quad (4)$$

In each case, an 'error' indication is given by the mean squared distances between the k nearest neighbors, given by

$$\delta_M(k) = \frac{1}{M}\sum_{i=1}^{M} |\overleftarrow{x}_{N\lfloor i,k\rfloor} - \overleftarrow{x}_i|^2 \quad (5)$$

where $\mathbf{E}$ denotes the mathematical expectation and $|.|$ Euclidean distance.

The relationship between $\gamma_M(k)$ and $\delta_M(k)$ is assumed linear as $\delta_M(k) \to 0$ and then $\Gamma = Var(r)$ is obtained by linear regression

$$\gamma_M(k) = \Gamma + G\ \delta_M(k) \quad (6)$$

A derived parameter of particular importance is the vRatio (V_r), defined as a normalized γ value. Since it is measured

in units of the variance of the output variable, it allows comparisons across different datasets:

$$V_r = \frac{\Gamma}{Var(y)} \tag{7}$$

This vRatio will be the fundamental parameter used in the analysis of the present data.

C. Modeling Techniques

1) Neural Networks: Neural networks (NN) are universal function approximators that can be applied to a wide range of problems such as classification and model building. It is already a mature field within computational intelligence and there are many different NN paradigms. Multilayer feed-forward networks are the most popular and a large number of training algorithms have been proposed. In this paper, networks with sigmoid functions $1/(1+e^{-x})$ in the hidden layers and linear output are used, which is a usual choice for function approximation. The neural network weights are found by optimizing backpropagation errors using the Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS) [5], [13]. BFGS is a quasi-Newton, second-derivatives method which is more efficient than steepest descent, conjugate gradient and other training approaches. The search direction is determined using an approximation to the inverse of the Hessian as opposed to gradient information only and line search is used to determine the step size rather than using a fixed one. The BFGS algorithm uses a procedure for updating the approximated Hessian after each step of the optimization process and only a small number of the recent updates are required for computing its inverse. Moreover, as the updating process progresses, the approximations of the Hessian become increasingly accurate. From the point of view of training speed, BFGS requires more function evaluations/iteration, but since it requires only a small fraction of of the number of iterations than classical backpropagation, the training time is significantly shorter.

2) Local Linear Models: Local linear regression models (LLR) belong to the class of classical K-nn techniques [3] from which non-parametric classifiers and function interpolators have been derived. To make a prediction for a given query point in the input space, LLR first finds the k nearest neighbours of the query point from the given data set (user defined) and then builds a multivariate linear model for the target variable using these k data points. Then the model is applied to the query point thus producing a predicted output. Local linear regression using the k nearest neighbours (in the training data) of the query point can be accomplished quickly. Thus local linear regression is a very fast and capable predictive tool.

It is most effective in regions of the input space with a high density of data points. If data points are few and far between in the vicinity of the query point then LLR will not be very effective if the underlying function we are trying to model is highly nonlinear. However, as the number of data points increases, the global function produced by LLR will converge rapidly to the unknown function generating the data, provided this is itself a smooth function. LLR can produce very accurate predictions in regions of high data density in the input space, but it is liable to produce unreliable results for nonlinear functions in regions of low data density. Sometimes LLR does not generalize well but is a very good interpolative tool for large amounts of data.

There are three user-defined parameters in LLR: the number of near neighbours, whether or not to include a constant term in the linear model, and a threshold value for filtering the local eigenvectors. The choice of the number of near neighbours k in LLR is quite critical. If the noise level of the output (i.e. the asymptotic Gamma statistic) is low then some small multiple of the number of inputs should suffice. If the noise level on the output is high then k needs to be larger to obtain better noise cancelation. The final user definable parameter is equivalent to a local principal components threshold filter on the eigenvectors of the local linear model. The idea is to predict along the tangent plane of the local flow and eigenvectors corresponding to relatively small eigenvalues probably represent noise and lie outside the tangent plane. The threshold decides which eigenvectors we should ignore. Setting it low or zero will essentially include all eigenvectors in the local model, the default is around 10^{-6}. Raising this threshold will filter out more eigenvectors.

3) Model Trees: A decision tree (DT) consists of leaf or answer nodes that indicate a class defined on the predicted (dependent or target) variable and non-leaf or decision nodes that contain a predictor attribute (variable) name and branches to other decision trees, one for each value of the predictor attribute. The top-down induction of decision trees is a popular approach in which the process starts from a root node and proceeds to generate sub-trees until leaf nodes are created. Classical algorithms for building decision trees are ID3 and C4.5 [16], [18]. The success with decision trees on classification problems induced the development of extensions of this approach to regression problems with continuous variables [17]. In some schemes the leaf nodes represent ranges in the values of the dependent (numeric) variable, in other constant values (i.e. 0-th order regression models), 1-st order linear models, splines, etc. In these cases what is produced is a Model tree (MT). In the particular case of M5 models, the leaf nodes are multivariate linear regression models. Accordingly, a M5 model is actually a combination of piecewise linear models each of which is suitable for a particular sub-domain of the input space as determined by the set of predictor variables.

There is a difference with pure linear regression: instead of a single regression there are several and the necessary (sub)optimal splitting of the input space is performed automatically. MTs can learn efficiently and can tackle tasks with high dimensionality which can be up to hundreds of attributes. They are transparent and simple to understand: basically they describe the conditions under which a particular multivariate linear model describes well the observed predicted variable. The segmentation (partition) of the input space can be done in many ways. Easy and fast are greedy

algorithms that explore only a single predictor variable at a time in a top-down recursive partitioning procedure and a criterion used for splitting the current variable values for the creation of the tree branches. Typically the criterion function has the form:

$$\phi(M, A) = I(M) - \sum_{i=1}^{n} I(m_i)p_i \tag{8}$$

where I is an inhomogeneity (impurity) function, M is the parent node, A the current attribute (predictor variable) considered, m_i the i-th child node, n the number of child nodes and p_i is the relative frequency of the instances within the i-th child node. At each step of the algorithm the idea is to choose the attribute value that minimizes Eq. 8. The impurity function is usually chosen to be a measure of the variability of observations. In the M5 algorithm, the standard deviation (sd) is used as the impurity [25] function. The splitting process terminates either when the number of observations into the node is less than a fixed value (generally equal to or less than 4), or when the standard deviation of the instances that reach the leaf is less than a minimum threshold (generally 5%) of the standard deviation of the original instance set. After having obtained the model tree, it is interpreted through 'if/then' rules induced by the nodes. A variety of heuristics and procedures have been proposed for tree pruning and producing a Rule Set from a MT. Here the approach described in [24], [8] is used.

IV. Experimental Settings

For this work, the experimental methodology consists of the application of computational intelligence and machine learning techniques in two phases: *I)* data exploration: characterization of the internal structure of the data and assessment of the information content of the predictor variables and its relation to the predicted (dependent) variables; and *II)* modeling: build models relating the dependent and the predictor variables.

For phase *(I)* an unsupervised study of the time dependencies structure of multidimensional time series describing the predictors (FSCS) and the target variable (MRNBX) (SectionII-A) was performed using nonlinear mapping techniques (Section III-A). The Gamma test was used for estimating the embedded dimension of the target series and was used in combination with a genetic algorithm for finding subsets of the predictor variables with modeling potential (i.e. input variables for function approximation techniques). These results were used as a base for model search at a subsequent stage.

In phase *(II)* three different modeling techniques were used: Neural Networks (NN), Local Linear Regression (LLR) and Model Trees (MT) (Section III-C). The first two are usually regarded as *black box* techniques, whereas the last is of the *white box* kind.

A. Data preprocessing

In order to overcome the effect of the different units of measurement used for the description of the input variables, which creates semantic incompatibilities, a normalization procedure for all variables was required. Among the different normalization approaches the one applied here transforms the mean of each variable to zero and its standard deviation to 0.5. With zero mean and constant variance, all variables have an equal chance to contribute to an output prediction and also have an equal weight on similarity and distance measures, which play a crucial role in the nonlinear space transformations and the gamma test techniques.

In order to explore the structure of the time dependencies within the FSCS and the MRNBX variables, phase space methods [11] were used. In the case of the MRNBX series (denoted as T), 20 time lags were considered (which cover its embedding dimension).

$$\{T(t-20), T(t-19), \cdots, T(t-1), T(t)\} \tag{9}$$

For the FSCS variables, a more complex setting was constructed in order to capture the nature of the lagged interactions between the whole set of predictors. If $P_k(t)$, denotes the k-th FSCS time series ($k \in [1, 30]$), tuples describing the state of the systems in terms of the predictors and the target can be formed as

$$\{[P_1(t-\tau), \cdots, P_1(t)], [P_2(t-\tau), \cdots, P_2(t)], \cdots, [P_{30}(t-\tau), \cdots, P_{30}(t)]\}, \quad T(t) \tag{10}$$

where τ is a maximum embedding lag for the MRNBX variables and the curly brackets separate the predictor from the target components of the tuple. In this study $\tau = 6$ in order to cover a time span of approximately two times the embedding of MRNBX (they are sampled with a 1:8 frequency ratio). The tuples in Eq.9 determine a 20-D phase space, whereas those of the predictor part of Eq.10, a 180-D space. The starting point of the analysis was the nonlinear canonical mapping to a suitable low-D space of the 180-D FSCS and 20-D MRNBX data. It was done by solving Eq. 2 with the Fletcher-Reeves method, a target dimension of 3 and using Euclidean distance as dissimilarity measure for both the original and the transformed spaces ($\delta(\vec{x}, \vec{y})$ and $\zeta(\hat{\vec{x}}, \hat{\vec{y}})$ in Eq. 1). Three different initial random configurations were used and the one with the smallest mapping error was kept.

From the point of view of the relationship between the predictors and the target in Eq.10, there are $2^{180} - 1$ combinations. They were explored using genetic algorithms (GA) aimed at minimizing V_r (Eq. 7). The problem representation chosen was based on binary chromosomes coding the characteristic functions of each predictor variable, so that the position of the 1 bits indicate whether the corresponding variable from the set of 180 predictors is chosen for assembling a multivariate vector used as input variables for the V_r computation. The number of predictor variables in the initial GA population is controlled by a parameter representing the probability of generating 1-bits in the initial chromosomes (a broad range was considered). Table II shows the set of GA parameters, defining a collection of 450 GA experiments. Since this was a preliminary study, exploration was favored over exploitation (100 generations only). In particular, the

TABLE II

EXPERIMENTAL SETTINGS FOR THE GENETIC ALGORITHM

Parameter	Value
number of objects	701
number of predictor variables	180
number of target variables	1
number of nearest neighbors	90
number of generations	100
number of individuals	100
elitism	yes
crossover probability	$0.3, 0.5, 0.6, 0.8, 0.9$
mutation probability	0.010000
quit if function reduced to this amount	$1e-7$
probability of generating '1's in the chromosomes of the initial population	$0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$
number of random seeds	10

number of nearest neighbors used for the computation of V_r was determined from a previous study of their relationship.

B. Modeling Techniques

During phase *(II)* the original set of normalized tuples (Eq. 10) was divided into training and testing sets with sizes 701 and 216 respectively (76%). The neural network models were trained according to the description and algorithm given in Section III-C.1 using a 180-50-10-1 or 80-50-10-1 architecture (input layer, first hidden, second hidden, output layer respectively). In the case of the Local Linear Models the number of nearest neighbors was set to 10 (not to be confused with those used for computing V_r), a constant term was included and the local principal components threshold was 10^{-6}. For M5 Model trees, the minimal number of observation within a node was set to 4, the standard deviation of the instances that reach the leaf to 5% of the standard deviation of the original instance set and the trees were pruned using the heuristics proposed in [8], [24].

V. RESULTS

A snapshot of the representation of the 20-D phase space of MRNBX via its nonlinear transformation to a virtual reality 3-D space is shown in Fig. 2(Top) (it is impossible to reproduce 3-D structures on hard media. Therefore only 2-D snapshots from a fixed perspective can be presented). It can be seen that the initial and final states of the system are very different (distance is inversely related to similarity) and that the system's trajectory in the mapped phase space passes through the same set of states at different times. They correspond to the two areas associated with the extreme states, joined in the middle region by a third group of intermediate or pivotal states, more similar (closer) to each other. It is interesting to note that a similar situation can be identified in the mapped phase space corresponding to the FSCS predictors (Fig. 2(Bottom)), in spite of the fact that it comes from an original 180 dimensional space, therefore much harder to map. In this case the mapping error is only two times larger (0.0694) than the one for the MRNBX phase space (0.0332), but the original dimensionalities have a whole order of difference. Therefore, the representation is

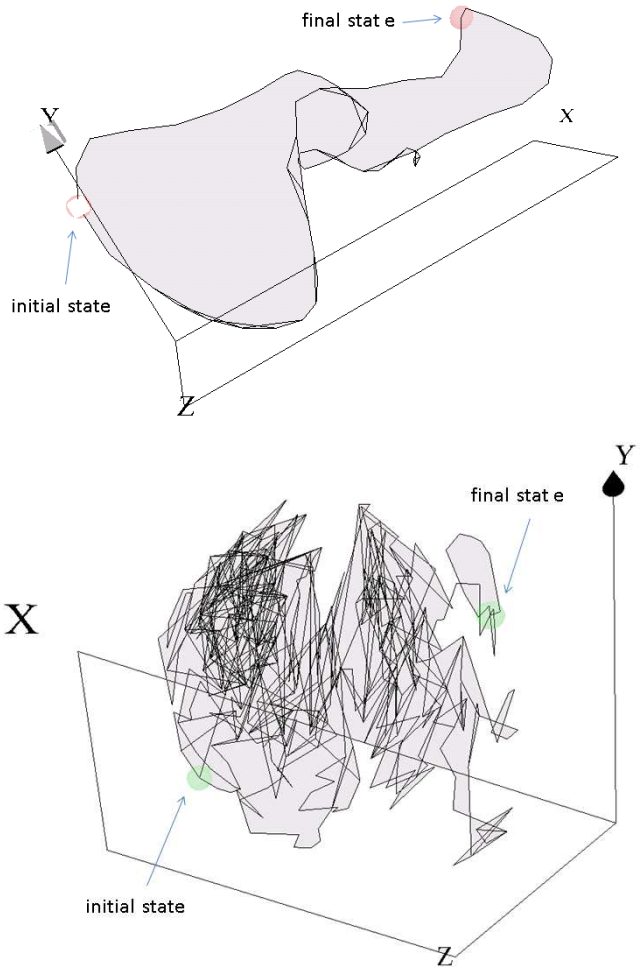

Fig. 2. Nonlinear mapping of the target and predictor phase spaces. In both cases a 3-D polyline links consecutive points along time for the original 180-dimensional time series (from the starting to end state). Top: Space corresponding to the target MRNBX ($\mathbb{R}^{20} \rightarrow \mathbb{R}^3$ mapping. Error = 0.0332). Bottom: Space corresponding to the FSCS predictors ($\mathbb{R}^{180} \rightarrow \mathbb{R}^3$ mapping. Error = 0.0694).

a reasonable approximation of the high-dimensional process. The FSCS phase space also shows two very different initial and final states (recall that both spaces are synchronized in time, so they can be compared), with two extreme regions bridged by bottleneck states. The nature of the time transitions is more irregular in this case, due to the combined effects of the space distortions introduced by the nonlinear mapping to a very small dimensional space (3) and the presence of redundant and/or noisy variables unrelated to the target. It is important to consider that the unsupervised nature of the mapping of the FSCS phase space makes more appealing its structural similarity with the MRNBX phase space. It suggests the presence of variables within FSCS with predictive potential, mixed with irrelevant and/or noisy ones.

The search of suitable subsets of the FSCS predictors made by the GA experiments led to $45,000$ candidate subsets of predictors in the final populations. The distribution of the V_r values is shown in Fig. 3. Its high skewness towards the lower

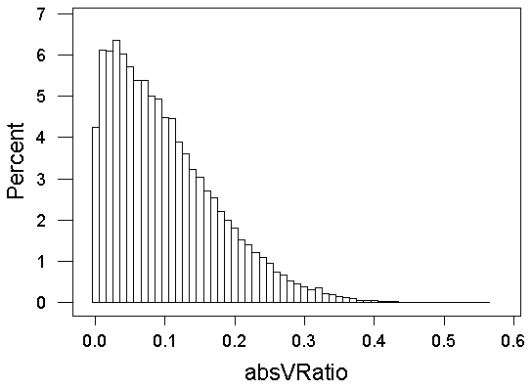

Fig. 3. Distribution of V_r values in the 450 GA experiments.

end indicates that despite the very different search conditions implied by the broad range of GA controlling parameters used, the evolutionary process converged to subsets of predictors with low residual variance, hence, good for modeling.

TABLE III
TEST SET RMSE ERRORS AND CORRELATIONS. LLR: LOCAL LINEAR REGRESSION. BFGS NEURAL NETWORK, M5: MODEL TREE

All predictor variables (180)				
	LLR	BFGS	M5	Ensemble Model
rmse	0.30536	0.24020	0.16504	0.16837
corr	0.78580	0.89595	0.94352	0.94023
Example from the Genetic Algorithm (80 predictor variables)				
	LLR	BFGS	M5	Ensemble Model
rmse	0.31017	0.31898	0.27316	0.21127
corr	0.77716	0.84365	0.85585	0.90861

Table III shows the results obtained with the individual and the ensemble models when using both all predictor and a subset among those found by the V_r-based GA search. The first case allows all 180 predictors to be used by the models, that is, the 30 FSCS parameters as well as their time history going back 5 time steps for a total of 180 predictor variables (Eq. 10). The second case uses the subset determined by one of the GA experiments that after 100 iterations reduced the number of predictor variables to 80. This case was selected from the left side of the mode in Fig. 3 but is not necessarily the best output of from all the experiments.

In the first case, with all 180 predictor variables available to the models, the LLR, BFGS-NN and M5-Model tree techniques produced models that predict the target parameter quite well, with relatively low RMSE and high correlation. The best individual model was generated by the model tree, however, it should be noted that it is a greedy technique that has the ability to discard input variables deemed not significant during the learning process, whereas the black box modeling techniques are forced to use all input variables, particularly LLR. In all cases the ensemble models were obtained by a simple average of the individual models. They generated a very close prediction for the target parameter, illustrated in Fig. 4(Top). It can be seen that the main as well as the secondary peaks of the main rotor blade normal bending have been reproduced very well. The phase of the predicted signal matches the original observations very well, which is an important feature for helicopter load monitoring.

In the second set of results using only the subset of 80 predictor variables, the models that were generated using the different techniques again produced close estimates of MRNBX with slightly higher RMSE values and slightly lower correlation coefficients than the first case of results. While the number of input variables was reduced to 44% of the original number, the performance of the models in predicting the target sensor suffered very little. The advantage of a smaller predictor subset is that irrelevant information is removed allowing for faster and more efficient processing. In this case, the ensemble model that was constructed from the three models demonstrated an optimal performance, shown in Fig. 4(Bottom). From this figure, we see again that the predicted and target signals line up very well with the peaks estimated quite closely. There is some under-prediction of the magnitudes of the peaks, more so than the first case, but overall the results are very promising.

While the flight condition examined in this work was a simple steady state manoeuvre, the results of the models provide some insight into the predictor/target relationship between the FSCS parameters and the main rotor blade normal bending. From the results of the GA experiment leading to the subset of 80 predictor variables, there was only one FSCS parameter that was completely omitted from the subset, the roll acceleration, the reason for which could be related to the flight condition chosen for this study (i.e. forward level flight). The remaining 29 FSCS parameters were included in the subset with varying numbers of time history points. While the GA produced a smaller subset of predictors, all but one of the FSCS were considered important in estimating the target sensor.

The highlights of the approach used in this work, not so often found in traditional methods, include: *i)* visual data mining of the high-dimensional phase spaces of the multivariate time series portraying the structural similarity of the target and predictor variables from the point of view of their time evolution and their states, *ii)* use of residual variance analysis to determine the likelihood of building a smooth deterministic model for the target sensor given the available predictors and to determine the relevant time dependencies appropriate for predicting the target sensor, and *iii)* the use of both black and white box machine learning techniques for modeling. The inclusion of these aspects makes for a more robust, statistically sound approach to the problem of helicopter loads estimation.

VI. CONCLUSIONS

In this work a study of the applicability of computational intelligence methods to estimate helicopter dynamic loads was carried out. While the work thus far is preliminary, the results are able to provide a "proof-of-concept" that such an approach is feasible since accurate and highly correlated models for the target sensor were built. The ability to generate satisfactory estimates show that the problem is solvable

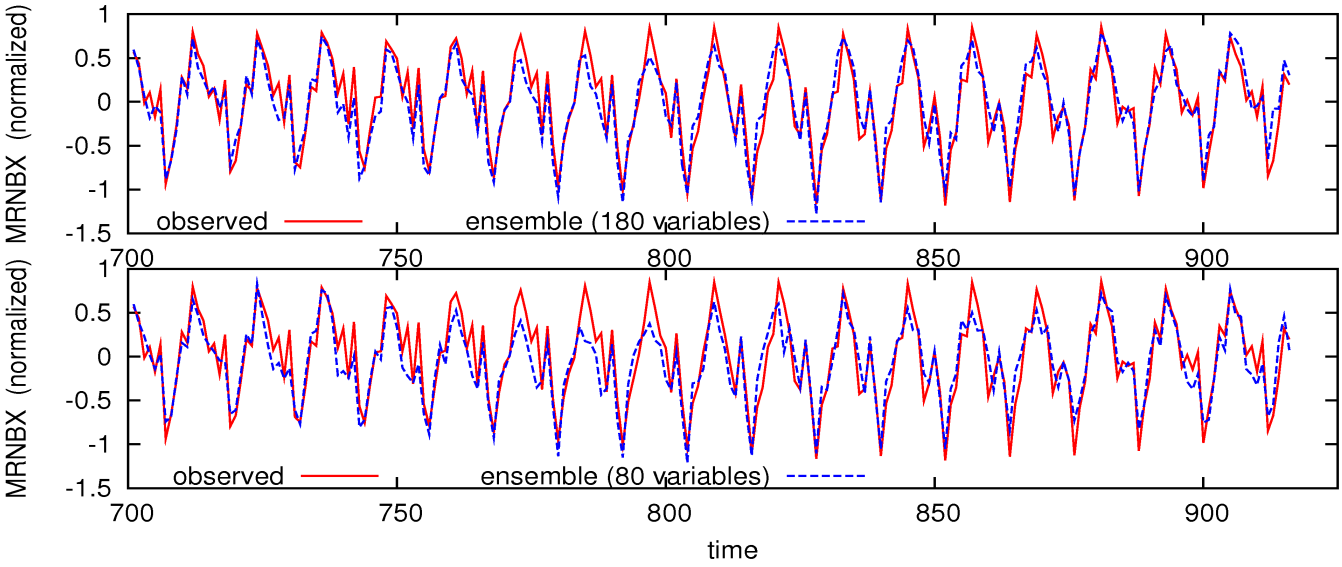

Fig. 4. Main rotor blade normal bending prediction based on FSCS. Top: Ensemble model using 180 predictors (6 time lags)(Eq.10). Bottom: Ensemble model using one of the subsets found by the genetic algorithm (80 predictors).

by computational intelligence methods. The information obtained enables a better understanding of the physical process behind the input/output parameter relationship. From the set of 30 sensors and their time histories used as input variables, a large amount of irrelevant and/or noisy information was discovered and identified, but from these 30 inputs it was possible to create models that were much simpler and also well-behaved. For the flight condition presented in this paper, forward level flight at maximum speed (V_H), the models were able to predict the main rotor blade normal bending based on input from 30 flight state and control system parameters with low error and high correlation. Having verified that this approach is indeed robust and useful for this problem, the next task will be to extend the scope and complexity of the output parameters and flight conditions and work towards the goal of accurately estimating the dynamic loads on the helicopter indirectly for its entire usage spectrum.

REFERENCES

[1] I. Borg. *Multidimensional similarity structure analysis*. Springer-Verlag, New York, NY, 1987.

[2] L. Chandon and S. Pinson. *Analyse typologique. Théorie et applications*. Masson, 1981.

[3] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[4] D. Evans and A.J. Jones. A proof of the gamma test. *Proc. Roy. Soc. Lond. A*, 458:1–41, 2002.

[5] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 1987.

[6] J. Gómez-Escalonilla, J. García, M.M. Andrés, and J.I. Armijo. Strain predictions using artificial neural networks for a full-scale fatigue monitoring system. In *Proc. 6th DSTO Int. Conf. on Health & Usage Monitoring*, Melbourne, Australia, March 2009.

[7] D.J. Haas, J. Milano, and L. Flitter. Prediction of helicopter component loads using neural networks. *J. American Helicopter Society*, 40(1):72–82, 1995.

[8] G. Holmes, M. Hall, and E. Frank. Generating rule sets from model trees. In *12th Australian Joint Conf. on Artificial Intelligence*, pages 1–12, 1999.

[9] Georgia Tech Research Institute. Joint USAF-ADF S-70A-9 flight test program, summary report. Technical Report A-6186, Georgia Tech Research Institute, 2001.

[10] A.J. Jones, D. Evans, S. Margetts, and P. Durrant. The gamma test. *Heuristic and Optimization for Knowledge Discovery*, 2002.

[11] H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis (2nd ed)*. Cambridge University Press, UK, 2003.

[12] A. Liu, C. Cheung, and M. Martinez. Use of artificial neural networks for helicopter load monitoring. In *Proc. 7th DSTO Int. Conf. on Health & Usage Monitoring*, Melbourne, Australia, March 2011.

[13] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35:773–782, 1980.

[14] F. Polanco. Estimation of structural component loads in helicopters: a review of current methodologies. Technical Report DSTO-TN-0239, Defence Science and Technology Organisation, 1999.

[15] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 2nd edition, 1992.

[16] J.R. Quinlan. Induction of decision trees. *Machine learning*, 1:181–186, 1986.

[17] J.R. Quinlan. Learning with continuous classes. In *5th Australian Joint Conf. on Artificial Intelligence*, pages 343–348, Singapore, 1992.

[18] J.R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, San Francisco, CA, 1993.

[19] S. Reed and D. Cole. Development of a parametric aircraft fatigue monitoring system using artificial neural networks. In *Proc. 22nd Symp. Int. Committee on Aeronautical Fatigue*, Lucerne, Switzerland, March 2003.

[20] J.W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Trans. Comput.*, 18(5):401–409, 1969.

[21] A. Stefánsson, N. Končar, and A.J. Jones. A note on the gamma test. *Neural Computing and Applications*, 5:131–133, 1997.

[22] J. Valdés. Virtual reality representation of information systems and decision rules: an exploratory technique for understanding data knowledge structure. In *Lecture Notes in Artificial Intelligence, LNAI*, volume 2639, pages 615–618. Springer-Verlag, 2003.

[23] J. Valdés and A. Barton. Virtual reality spaces for visual data mining with multiobjective evolutionary optimization: Implicit and explicit function representations mixing unsupervised and supervised properties. In *Proc. IEEE Congr. Evolutionary Computation*, pages 592–598, 2006.

[24] Y. Wang and I.H. Witten. Induction of model trees for predicting continuous classes. In *Proc. European Conf. on Machine Learning*, pages 128–137, Prague, 1997.

[25] I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, CA, 2005.