AIAA JOURNAL
Vol. 30, No. 5, May 1992

Identification of Time-Varying Structural Dynamic Systems: An Artificial Intelligence Approach

B. J. Glass* and S. Hanagud†
Georgia Institute of Technology, Atlanta, Georgia 30332

An application of the artificial intelligence-derived methodologies of heuristic search and object-oriented programming to the problem of identifying the form of the model and the associated parameters of a time-varying structural dynamic system is presented in this paper. Possible model variations due to changes in boundary conditions or configurations of a structure are organized into a taxonomy of models, and a variant of best-first search is used to identify the model whose simulated response best matches that of the current physical structure. Simulated model responses are verified experimentally. An output-error approach is used in a discontinuous model space, and an equation-error approach is used in the parameter space. The advantages of the AI methods used, compared with conventional programming techniques for implementing knowledge structuring and inheritance, are discussed. Convergence conditions and example problems have been discussed. In the example problem, both the time-varying model and its new parameters have been identified when changes occur.

Nomenclature

A = amplitude
a,b = weights
b = beam base width
b_i = best model at abstraction level i
C = damping matrix
C = total number of models searched
C() = a cardinal number
c_{ij} = damping parameters
c_ϕ = mode shape correlation index
E = Young's modulus
E = error matrix
EF = average parameter error
F = family of models: a grouping
$f(x, t)$ = forcing functions
h = beam height
H^m = measured (observed) response of physical system
Δh = difference in J value between neighboring models at the same level of abstraction
$\Delta \bar{h}$ = largest value of Δh at a given level of abstraction
Δh^* = smallest value of Δh at a given level of abstraction
I = moment of inertia in given direction
I = identity matrix
J = heuristic error function
K = stiffness matrix
k = model index in a search tree
k_{ij} = stiffness parameters
L = beam length
L = level of search in tree
l = number of parameters common to all models
L_i = differential operator
LA = level of abstraction
M = mass matrix
m = model index in a search tree
m_{ij} = mass parameters
N = number of models
n = branching factor
n = branching factor matrix
N_m = number of modes used in model evaluation function
p_{ij} = generic parameter matrix element
R = set of most likely models
R = set of all models
S = common parameter vector
Sp = number of springs
t = time
$u(x, t)$ = vector of dependent variables
v = distinguishing-parameter vector
x_s = boundary locations
z = state variable vector
α = set of variables, in an object or frame (i.e., slots with values)
β = set of attached procedures, in an object or frame
β_j = boundary condition
γ = phase angle
δ_b = bound on the magnitude of the parameter set error
$\Delta\theta$ = parameter set error
η_k = initial condition
θ = parameter vector
Λ = diagonal eigenvector matrix
μ = model
ξ = mass identity
ρ = material density
σ = memory used to store a model space
ϕ = mode shape or eigenvector
ω = natural frequency
ω_d = analytical frequency
$()^{-1}$ = inverse matrix
$()^m$ = measured quantity
$()^T$ = transpose matrix
$()^*$ = optimal value
$()_0$ = initial value
$()_n$ = next-best value in a given set with unequal members' values

Introduction

MANY procedures for achieving an optimal control of structural dynamics systems require an accurate analytical model of the physical system. Analytical models, whether obtained as sets of partial differential equations or as approximate sets of ordinary differential equations from a finite element analysis, are usually verified by comparing the model's predicted response to observations during operation or laboratory experiments. If the model response lies outside

Received July 27, 1987; revision received June 25, 1989; accepted for publication July 24, 1989.

*Graduate Research Assistant, School of Aerospace Engineering; currently Aerospace Scientist, Control Systems, Information Sciences Division, M. S. 269-4, NASA Ames Research Center, Moffett Field, CA 94035. Member AIAA.

†Professor, School of Aerospace Engineering. Member AIAA.

the acceptable bounds established by measured data, then some degree of system identification (or an inverse process) can be used to obtain an accurate model from a knowledge of the measurements. In general, system identification procedures may or may not use an a priori or an approximately known model in addition to experimental observations. This paper is concerned with procedures that use an a priori model. In these cases, structural dynamics system identification is usually comprised of two steps: first, the basic mathematical form of the model is determined from physical assumptions. Historically, work in structural dynamic system identification has assumed that structures are time-invariant with respect to the form of the model, so that a form of the model is assumed to be known initially. Once the model structure is established, its parameters are then subject to identification.

The a priori model usually takes the form of a large discretized model with known values for some or all of the mass, stiffness, or damping matrix elements. The data used for identification may be in the form of time histories, frequency-domain response functions, or modal parameters obtained from measured data. The procedures for parameter identification differ in the manner in which performance or objective criteria are formulated in terms of the measured response. There are many different approaches to the parameter identification problem depending on the noise level, available data, real-time performance requirements, and the model characteristics. A simple approach is the output-error method,[1,2] which seeks to minimize the difference between some given model output and the observed system output. If there is sufficient confidence in the a priori model, a minimum deviation approach[3,4] seeks a set of system parameters that deviate least from the a priori model while not violating the required constraints and specified bounds of measured response. In the equation-error approach,[5,6] the parameters are estimated by minimizing the error in the model equations when actual data are substituted for dependent variables.[7] In this context, structural dynamic system identification is a procedure for finding realistic model parameters that justify the reasonable nature of the assumptions.

However, there is a different type of identification, termed model identification. This concerns a class of structures that consists of time-varying structures. In these structures, the physical nature of the structure changes with time. This paper addresses such problems of system identification of time-varying structural dynamic systems. Changes in model are necessary because of large-scale, discontinuous changes caused by component failures, configuration changes, or payloads. These changes sometimes affect the correct choice of model equations, boundary conditions, and number of degrees of freedom, and cause drastic shifts in parameter values. The need for changes in the model form requires that the system identification process include identification of time-varying forms of models as well as the identification of the model parameters.

Approach

In this paper, the problem of model identification has been studied by using computational methods originally developed by the artificial intelligence (AI) community. A heuristic search is used to identify the model that best matches the current system response, with an output-error approach for the model evaluation criteria. A heuristic search is guided by generalized relations or evaluation criteria, which are domain-dependent. Rather than considering an infinite number of possible models, a finite search space is constructed of a set of models that includes, it is hoped, those most commonly encountered. The models in the search space are represented by using an object-oriented approach, which is shown to efficiently structure the model space and reduce memory requirements due to inheritance of model data. By associating a precomputed discrete degrees-of-freedom model with each possible model in the search space, the concept of a single a priori model of the conventional identification approach has been extended to a space of possible models, as shown in Fig. 1. Procedures for retrieving the approximate a priori model by using AI-based model identification techniques are followed by an equation-error method[5] for parameter identification. Modal characteristics of the system are assumed to be the input to this parameter-identifying method.

The chief contributions of this paper are the development of identification techniques for structural dynamic systems that have time-varying forms of models, and the use of novel AI-based computational methods for solving these structural dynamic system model and parameter identification problems. Time-varying structures are of concern because of their occurrence in large space structures (LSS), robot manipulators, and rotorcraft systems with failure and sudden model changes. For these systems, discontinuous changes alter the identity of the correct model, making the notion of optimal control, based on the previous model, meaningless. Issues also considered in this paper are those of the organization and selection of a model space, and the efficiency-enhancing aspects of model inheritance.

In the first section of this paper, the problem of system identification, including both model and parameter identification, is formulated. In the second section, reasons for using levels of abstraction in organizing a model space, and the benefits of inheritance in that model space, are discussed. Next, the benefits to be derived from the use of the AI techniques of heuristic search, discrimination nets and frames (i.e., object-oriented programming) are outlined. The developed procedure is compared with a conventional programming approach and convergence criteria discussed. Finally, the procedures for combined model and parameter identification are implemented for a selected class of time-varying structures. These results are then compared with system identification procedures that consider only parameter identification.

Problem Setting

Standard Structural Dynamics Problem

A structural dynamics problem for a given model μ is usually defined in terms of differential operators L_i, boundary conditions β_j, and initial conditions η_k

$$\mu: \begin{Bmatrix} L_i[\theta_r, u(x, t), f(x, t)] \\ \beta_j[\theta_r, u(x_s, t)] \\ \eta_k[u(x, t_0)] \end{Bmatrix} \tag{1}$$

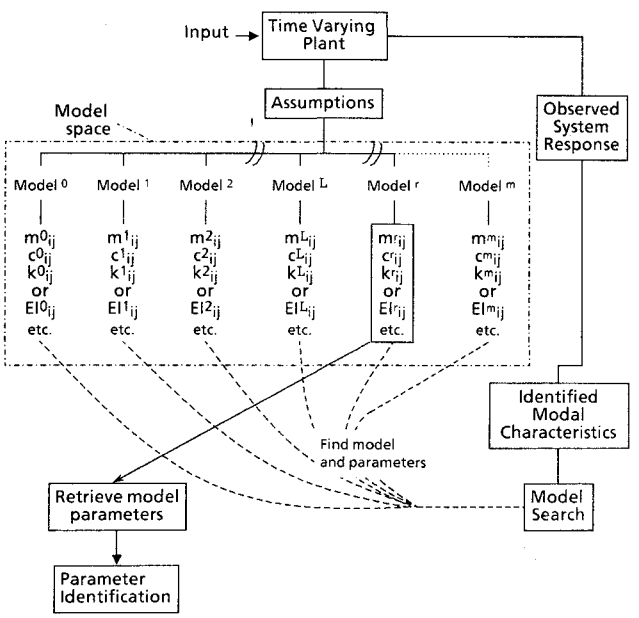

Fig. 1 Model identification process.

$$i = 1, \ldots, N; \quad j = 1, \ldots, P; \quad k = 1, 2 \text{ in general}$$

where θ_r are the model parameters, $\boldsymbol{u}(\boldsymbol{x}, t)$ are the dependent variables $\boldsymbol{x}_s$ represent boundary locations, and $\boldsymbol{f}(\boldsymbol{x}, t)$ are forcing functions. It is usually assumed that the form of μ is known in structural dynamics problems. A given model can also be expressed as a discrete degree-of-freedom (DOF) model, in which case θ_r would include the mass and stiffness matrix elements M and K. For typical structural dynamics problems, μ, θ_r, and $\boldsymbol{f}$ are known and it is desired to find the response $\boldsymbol{u}(\boldsymbol{x}, t)$.

Structural Dynamics Parameter Identification

Although system identification encompasses both model and parameter identification, usually structural dynamics system identification is formulated as a parameter identification problem. This is because the form of the model μ is known, including its component L_i, β_j, and η_k. Typically, sets of dependent variables $\boldsymbol{u}(\boldsymbol{x}, t)$ are known for sets of $\boldsymbol{f}$ and η_k, but the parameters θ_r are unknown. In these cases, the response is assumed to be a function only of the unknown parameters, i.e., $\boldsymbol{u}(\theta_r, \boldsymbol{x}, t)$ as all else is defined

$$\boldsymbol{u} = \boldsymbol{u}(\theta_r, \mu, \boldsymbol{f}, \eta_k, L_i, \beta_j, \boldsymbol{x}, t) = \boldsymbol{u}(\theta_r, x, t) \tag{2}$$

From experimental measurements or observations, the measured response $\boldsymbol{u}^m(x_i, t)$ is obtained at specified locations x_i and times. A simple example of the problem of parameter identification would then consist of minimizing an error function with respect to the unknown parameters θ_r. For instance, in an output-error method of parameter identification

$$J = \int_{\text{time}} \int_{\text{space}} \left| \boldsymbol{u}(\theta_r, x_i, t) - \boldsymbol{u}^m(x_i, t) \right| \mathrm{d}y \, \mathrm{d}t \tag{3}$$

is minimized for a continuous distributed-parameter system, or

$$J = \int_{\text{time}} \left[\boldsymbol{u}(\theta_r, x_i, t) - \boldsymbol{u}^m(x_i, t) \right]^2 \mathrm{d}t \tag{4}$$

for a system with discrete DOF. By setting

$$\frac{\partial J}{\partial \theta_r} = 0 \tag{5}$$

and solving for θ_r, parameters are identified. If solved iteratively, a starting set of θ_r is used, which together with the model μ constitutes an assumed a priori model. This initial guess for θ_r may lack some parameter values, but its specified values are assumed to be in a neighborhood of the actual values. In the foregoing procedure, J is assumed to be differentiable in the parameter space.

Structural Dynamic Model and Parameter Identification Problems

Model identification becomes necessary when the form of the model μ is not known, but belongs to a space of models of the form

$$\mu_l: \begin{Bmatrix} L_{il}[\theta_{rl}, \boldsymbol{u}_l(\boldsymbol{x}, t), \boldsymbol{f}(\boldsymbol{x}, t)] \\ \mu_1: \beta_{jl}[\theta_{rl}, \boldsymbol{u}_l(\boldsymbol{x}_s, t)] \\ \eta_{kl}[\boldsymbol{u}_l(\boldsymbol{x}, t_0)] \end{Bmatrix} \tag{6}$$

$$l = 1, \ldots, N_l; \quad i = 1, \ldots, N_i$$

In these equations μ_l is the space of all possible models. In this case, the simulated responses of the possible models, $\boldsymbol{u}(\boldsymbol{x}, t, \mu, \theta_r)$, must be available so that they may be compared with the measurements $\boldsymbol{u}^m(x, t)$ from the current physical structure. An appropriate performance function is then minimized to identify μ.

Rather than search the infinitely large group of models $\boldsymbol{R}$, computational and model-generating limits require that the search space be restricted to R, the set of the n likeliest models. Unlikely structural configurations are thereby excluded. This restriction to a finite number of boundary condition (BC) combinations implies the possibility of approximate, rather than exact, model identification when a given model $\mu \epsilon R$; i.e., the observed system might not correspond exactly to any model included in the search space. In that case, the closest matching model in R is returned.

In this implementation of model identification for structural dynamics systems, it was decided to express the response in terms of the modal characteristics, available from the time domain response through modal identification and analysis. Noise is assumed to be eliminated by the use of cross-correlation techniques.

$$\boldsymbol{u}_l = \boldsymbol{u}_l(x, t, \mu_l, \theta_{rl}) \rightarrow \omega_l, \phi_l \tag{7}$$

$$\boldsymbol{u}_l^m = \boldsymbol{u}_l^m(x, t) \rightarrow \omega_l^m, \phi_l^m$$

By selecting ω and ϕ to represent response, it is assumed that the values of ω_{il} and ϕ_{il} are distinguishable for models in R. Analogous to the problem of parameter identification, we now seek to find the model μ_l that minimizes an objective function. Then a heuristic model evaluation function is defined in terms of modal characteristics

$$J = \sum_{i=1}^{N_l} \left\{ a_i \left[\omega_i^m - \omega_i(\mu_l, \theta_{rl}) \right]^2 + b_i \left[1 - c_{\phi_i}(\mu_l, \theta_{rl}) \right] \right\} \tag{8}$$

where $N_j \leq N_i$, a_i and b_i are weights, and c is a mode shape correlation index in [0, 1]. It should be noted that now J is a function of both the model form and parameters

$$J = J(\mu_l, \theta_{rl}) \tag{9}$$

Even though the form of a time-varying structural dynamics model may be uncertain, it is assumed that reasonable estimates of parameter values are available from off-line analytical estimates of the mass and stiffness values of the initial model, or else from an a priori knowledge of material densities, cross-sectional areas, etc. For instance, a beam with discontinuously varying tip masses has relatively constant parameter values along the beam's length, but dramatic shifts at the tip. Reasonable estimates of mass and stiffness along the length are then available for finite element beam models. The assumption of the knowledge of an a priori set of parameters θ_{rl} reduces the evaluation function to

$$J = J(\mu_l) \tag{10}$$

One might then expect to minimize J in a similar fashion to parameter identification; however, μ is a discontinuous, discrete variable in R, so $\partial J / \partial \mu$ cannot be defined.

To minimize J in a discrete model space, the method of heuristic, best-first search is used to selectively examine and compare model responses until the model is found that minimizes J, or else reduces it below a required threshold level. The procedure then is to search to minimize the heuristic

$$H = \min \left[J^*, J(\mu_l) \right], \quad \mu_l \epsilon R' \tag{11}$$

where J^* is the minimum J found by the search prior to evaluating $J(\mu_l)$.

Having then obtained a model form μ, we vary the parameters θ_r by applying parameter identification as in the previous section, but using an equation-error approach that has been developed recently.[5,7] The identified model and parameters then define the new a priori model, whose parameters can be used in successive iterations of model and parameter identification if necessary. These cycles of model and parameter identification are iterated repeatedly for a given response until a new, acceptable structural dynamics model is obtained.

Example of the Inadequacy of Only Using Parameter Identification

Current system identification methods[1-7] using parameter identification alone are inadequate for identifying discontinuously time-varying models, as their structural identity quickly changes. Parameter identification methods assume a time-invariant model structure. For example, a parameter-identifying procedure for an original cantilever beam might receive current dynamic response data corresponding to a new model, such as a cantilever beam with a nonstructural mass at the nonclamped end. The parameter-identifying program could accommodate the lowered natural frequencies only by altering the stiffness or mass values of all the elements of its cantilever model.

AI-Based Identification Procedure

Model Organization

Obtaining Levels of Abstraction

The distinguishing characteristics of models lend themselves to model taxonomization by representing simple and detailed models at differing levels of abstraction (LA) such as used by Sacerdoti's ABSTRIPS[8] to distinguish levels of detail in planning. Solutions vary for the differing sets of boundary conditions pertaining to each structural dynamic model, but their resulting mode shapes and frequencies will be similar for structures with similar geometric boundary conditions. These permutations of an underlying model, such as a cantilever beam with different nonstructural masses at the tip, typically result in clusters, or family groupings, of several related model variations. These distinct models have several boundary conditions and/or parameters in common. However, similarity of response generally decreases proportional to the decline in the commonality of boundary conditions (BCs) and parameters. We choose to represent a family of models as a hierarchy consisting of a prototypical model with sets of differences at each level of abstraction that distinguish each model in a family. If the hierarchy of classes so obtained is "well-ordered," the differences in J [as defined in (8)] between the models at the lower levels of the hierarchy will be smaller than those at higher levels of abstraction.

A model hierarchy can be explained by considering one-degree-of-freedom spring-mass systems shown in Fig. 2, with the first system having two possible masses, $m_4 > m_1$. The governing equations for these systems is

$$\ddot{x} + \omega^2 x = 0; \qquad \omega^2 = \frac{k_i}{m_i} \tag{12}$$

with a solution of the form

$$x(t) = A\cos(\omega t + \gamma) \tag{13}$$

where γ is the phase angle, A is the amplitude, and γ and A are functions of the initial displacement x_0 and initial velocity v_0. Common parameters for these systems are

$$s = \left[k,\ x^m,\ x_0^m,\ v_0^m\right] \tag{14}$$

with distinguishing parameters of Sp (number of springs) and ξ (mass value)

$$v_1 = Sp, \qquad v_2 = \xi \tag{15}$$

which results in a search tree, shown in Fig. 3.

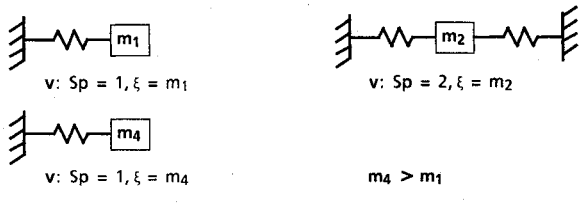

Fig. 2 Simple undamped spring-mass systems.

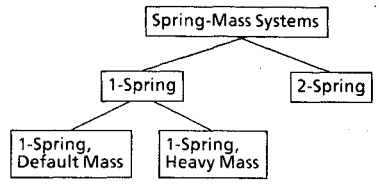

Fig. 3 Search tree for spring-mass example.

Considering the composition of a structural dynamics model, we may define a set of state variables z that describe model characteristics and a set of functions that relate them to previous state values

$$\mu \therefore \left\{ \mathbf{z}^{\mathbf{T}} = [s_1, s_1, \ldots, s_k;\ v_1, v_2, \ldots, v_l];\ f_1(z), f_2(z), \ldots, f_m(z) \right|$$

$$\left. v_{j_{(t=N+1)}} = f_j\left[z_{(t=N)}^{(p)}, z_{(N)}^{(p-1)}, \ldots, z_{(N)}^{(1)}\right] \right\} \tag{16}$$

where s_i represent the k parameters common to all models and v_j are parameters whose values distinguish models in R. These parameters are not necessarily mass and stiffness matrix values; rather, in the case of v_j, they may be any set of attributes that are sufficient to distinguish it from all others in R. For example, a structural dynamics model would include both its governing set of differential equations and its moment of inertia equation(s) in f, whereas examples of parameters might be a material constant for s_3 and a boundary condition for v_2. For a simply supported beam example, the corresponding model could include

$$\mu \therefore \Bigg(z^T = \{s_1{:}E,\ s_2{:}I,\ s_3{:}L,\ s_4{:}b,\ s_5{:}h,\ s_6{:}\rho;$$

$$v_1{:}\mathrm{BC}(l) \therefore [w(l),\ w'(l) = 0],\ v_2{:}\mathrm{BC}(0) \therefore [w(0),\ w'(0) = 0]\}$$

$$f_1(z){:}\left[\frac{\partial^2}{\partial x^2}\left(EI\frac{\partial^2 w}{\partial x^2}\right) + m\frac{\partial^2 w}{\partial t^2} = f(x, t)\right]$$

$$f_2(z){:}I = \frac{bh^3}{12}, \quad f_3(z){:}m = \rho bhL \tag{17}$$

$$f_4(z){:}\omega_n = (n\Pi)^2\left(\frac{EI}{mL^4}\right)^{1/2}, \quad f_5(z){:}\Phi_n = \left(\frac{2}{mL}\right)^{1/2}\sin\frac{n\Pi x}{L}\Bigg)$$

By defining a model holistically, including both parameter values, BC's and equations, as in Eq. (16), it is possible to use an approach similar to "objects" or "frames" as defined by Minsky.[9]

To organize these model objects, we define the level of abstraction (LA) of a particular subset of R as inversely proportional to the number of nonnull v_j common to its members. In other words, highly abstract, prototypical models have fewer details specified than the fully detailed models at lower levels of abstraction.

$$\mathrm{LA} \propto [tr(vI)]^{-1} \tag{18}$$

For example, the LA of an unadorned cantilever is higher, since it is a less detailed model, than the values for cantilevers with attached springs and/or masses since more v_j are specified for the added-mass and added-spring models. Since the total set of parameters $\bar{v}$ is sufficient to distinguish all models from each other, $\mathbf{C}(\bar{v})$ equals the maximum number of levels of abstraction for R where $\mathbf{C}()$ designates a cardinal number. Hence, the level of abstraction for a model μ^j may be defined as $\mathbf{C}(\bar{v})$ less the number of v_j defined for that model, or

$$\mathrm{LA} = \mathbf{C}(\bar{v}) - \mathbf{C}(v^j) \tag{19}$$

Modeling of the Knowledge Base: Hierarchical Objects with Inheritance

By grouping related models into families and ranking models by their LA, it is possible to organize the model space R into a directed graph, resembling a hierarchically branching, tree like structure, as shown in Fig. 4. These abstract data structures, often called search trees, have been commonly used and improved in artificial intelligence research and applications for nearly 30 years.[10-13] Beginning with only the common parameters s_j at the top node (most general prototype), distinguishing characteristics v_j are added as the depth in the search tree increases. Note that models with one or more null v_j values are used as prototypes and are introduced above the lowest level of abstraction. Sometimes the actual physical system may more closely resemble an uncomplicated high-level model than more complex lower-level models, so all models once introduced are propagated downward through the search tree. This is accomplished by using zero or null values for one or more v_j. For instance, a cantilever beam with a tip mass is in the family of simple cantilever beams, but the generic simple cantilever model may still be the closest in response to the current structure at a low level of abstraction—despite the greater model detail and additional models considered at the lower LA. A search tree thus construed, with generic models included in the set of their children, will have all N models in R represented at its terminal or bottom nodes. By letting $L \equiv$ level of search in the tree and $n \equiv$ the branching factor, it can easily be shown that, for a regular search tree of the type shown in Fig. 4,

$$n^L = N \tag{20}$$

Although the order in which the v_j are considered is not specified, the order should be one that causes increasingly similar model evaluations among children nodes as one progresses down a given branch of the search tree. This requirement that a search tree be "well-ordered" is necessary to insure progressive selection of the subset containing the optimal model. Otherwise, lower-level models might be closer in response to other prototypes than to their own prototypes, which might require search to backtrack in order to find the best model.

Data structures of the type just shown are also known as discrimination nets, first implemented in Feigenbaum's EPAM,[14] with each nonterminal node representing a yes-no test. In 1975, Minsky[9] enhanced this idea by organizing frames (a knowledge representation paradigm) into inheritance hierarchies, grouped at higher levels of abstraction by their similarities as has been done in this model space. More recently, both Schank[15,16] and Kolodner[17,18] have used networks of frame like memory organization packets (MOPs) to model human long-term memory, again grouping by similarities and indexing by differences. The MOP approaches differ from the model space here in the type of knowledge represented, and that in this paper the search tree is indexed by only one v_j at each level. In contrast, both of the MOP-based systems have considered multiple v_j at every level of abstraction,[19] resulting in the creation of multiple paths to the same node. Although necessary for natural language processing (in which a given input concept may be expressed in multiple forms), this redundancy is not needed for this model identification problem

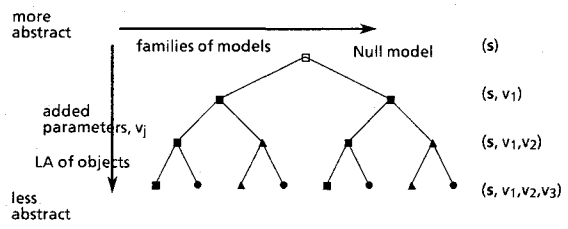

Fig. 4 Directed graph of R with $N = 8$ (a regular search tree).

since the form of the input is known. Unlike EPAM, both Schank's and Kolodner's approaches and this approach all discriminate on the basis of the specified values of the v_j, not on the absence of a particular value.[19]

Benefits of Search Trees and Inheritance

Our desire for efficient search also leads to the use of concepts of knowledge organization, i.e., a networked or hierarchically structured model space R, rather than an unstructured or single-array arrangement. Also, by allowing low-level models to refer to a value stored at a parent node, rather than duplicate the same value in both models, the concept of inheritance may allow memory needs to be reduced.

If C is defined as the total number of models searched, for an unstructured or single-array model space containing N models, all models must be checked to insure that the optimal match is found, or

$$C = N \tag{21}$$

This lack of structure might be found if models were stored in Fortran files or in a single unnormalized relational database table. If, instead, the models are structured in a search tree, as in Fig. 4, it is possible to derive new values of the total number of model evaluations

$$C = n + \sum_{i=2}^{L} (n-1), \quad \text{and} \quad C \propto O(nL) \tag{22}$$

for regular (n = constant) search trees and for irregular trees

$$C = n_{01} + \sum_{i=2}^{L} (n_{i-1,b_{i-1}} - 1) \quad \text{and} \quad C \propto O(\bar{n}L) \tag{23}$$

where $\bar{n}$ is the norm of the branching matrix $\boldsymbol{n}$, n_{01} is the branching factor at the root, and b_i designates the best model at abstraction level i in the tree. Given that $N = n^L$ from (20), we see that C is reduced from an exponential to a polynomial quantity if a structured knowledge base R is used.

By providing upward and downward links between models at varying levels, the use of a structured model space may also allow more compact storage of R in memory by means of inheritance. Downward links are provided in a discrimination net by the distinguishing characteristics v_j, whereas subclass-superclass relations (so-called "ISA" links, e.g., "a cantilever with a tip mass ISA cantilever") provide upward links. For a general R, the memory needed to contain all N of the models in the model space is, from (16), proportional to

$$\varsigma = N \cdot \|z\|, \qquad \propto N(k+L) = NL + Nk \tag{24}$$

where z is the parameter vector defined in (16), k is the number of parameters common to all models, and L is the number of distinguishing characteristics. Since the discrimination net here uses only one index at each level, L also corresponds to the number of levels of abstraction. Without inheritance, all models have the same values for the k-fixed parameters s_j so that these values are replicated in all N models. By inheriting values, it is possible to reduce ς by defining these common values only once, at the top of the search tree, and referring to these values from lower-level models by traversing ISA-links upward. This reduces storage considerably

$$\varsigma' \propto NL + k \tag{25}$$

or, substituting (20), we obtain

$$\varsigma' \propto n^L L + k \tag{26}$$

If we consider also the values of v_j that, once defined, are common to models at lower levels, further reduction of memory requirements is possible by allowing these values to be inherited by related models at subsequent levels of abstraction. The resulting storage is then proportional to

$$\varsigma'' \propto n^L + k = N + k \tag{27}$$

for regular model identification search trees, since no more than one new distinguishing v_j is specified for any of the N models.

Substituting (24) and (20) into (27), we can get a relation indicative of the advantage of using inheritance in model identification

$$\varsigma'' = \frac{\varsigma}{L} - k\left(\frac{N}{L} - 1\right) \tag{28}$$

Even for the worst-case search tree, the use of inheritance clearly reduces memory use when compared with either model trees without inheritance or a single-array search group.

Knowledge Representation Paradigm Selected

Given the desirability of a structured model space (levels of abstraction) with inheritance of data values, the next consideration is the programming approach that best implements these required attributes of model identification efficiently. It should be noted that levels of abstraction and the inheritance of data are not synonymous: model space organized into a network or search tree may nevertheless replicate all parameters and relations for every model, and it is also possible for models in a single-array model space to refer to a set of global parameters, such as material constants. However, the advantages of both levels of abstraction and inheritance, as given in the previous section, are such that the use of both is the preferred approach.

Implementation Approaches: Object-Oriented Programming

In a procedure-oriented language such as Fortran or Pascal, we would need to build the search tree relationships between models into the program or system structure, calling a subroutine to calculate the response of each model. Many models would correspond to many subroutines and so would engender an increasingly complex control algorithm to maintain the levels of abstraction in the model space. The common parameters s could be read and made global, as in a Fortran common block. However, progressive inheritance of the distinguishing parameters v_i, as they are specified at various levels of abstraction, would require that all old model parameters be retained in the common block of the program until completion of the search, and that the lower-LA subroutines be encoded explicitly with the knowledge regarding which old parameters to use. Alternatively, data could be reread, when needed, from an indexed file rather than putting the data in a common block.

Furthermore, this conventional approach suffers from the separation of programs and data. Separate programming schemes are needed to access both the model response subroutines and the model parameter data at progressive levels of search. The savings due to inheritance of model parameters from higher-LA models are offset by the increased programming complexity within the search program or its model response subroutines. Also, in a sequential programming language like Fortran, only one model can be considered at a time, as its subroutine is called by the main program. Although parallel hardware could be used to simultaneously evaluate multiple models, this is an expensive solution and one that grants similar benefits to competing approaches.

In contrast to a conventional procedural approach, in object-oriented programming systems (such as Smalltalk-80,[20] Flavors,[21] or LOOPS[22]) procedures and data are localized to specific models, which are treated as abstract entities called objects. An object includes both data values, such as model parameters (θ), and the procedures necessary to calculate or manipulate its data values, such as structural dynamic response and moment-of-inertia formulae (μ). Rather than explicitly call a procedure on data to return a result, in an object-oriented approach a set of messages is sent to a given object, which uses its built-in procedures to calculate the results, which in turn are passed back to the user (which itself may be another object). In our case, the search program would query a model for its dynamic response.

Although it may be possible to simulate these functions in Fortran or a similar language, the information structures that comprise frames or objects are most easily handled as lists of predicate-value pairs; i.e., data labels with their associated variable values. The language Lisp is therefore preferable for implementing these systems, given its wealth of existing, easily used list-processing functions. Lisp's ability to treat programs as data is also very useful in implementing objects.

By removing the separation between programs and their data, an object-oriented approach simulates high-level interactions between intelligent entities. An analogy could be made to a department head in an engineering firm who desires the dynamic analysis of a structural component. The department head will send a message to one of his staff requesting the results of an analysis of the structure. The engineer will set up the problem, make assumptions, locate needed data, and find the desired results. Like an object in an object-oriented system, the engineer knows both how to conduct the analysis and where to get the necessary data. Given this capability, the department head does not need to tell the engineer how to conduct the analysis or where to find the data when he requests the results. Just as the engineer belongs to the class of structural dynamicists, so objects belong to classes that describe their characteristics, common parameter values and procedures. These classes are themselves objects and may be described, in turn, by superclasses. This hierarchy of objects allows levels of abstraction in our model space to be naturally and easily constructed in terms of objects, classes, and superclasses, each of which can represent a model at some abstraction level.

Object-oriented systems also provide built-in inheritance between superclasses and subclasses by automatic querying of an object's superclass for a value if the value cannot be found by a given object. This inheritance feature of object-oriented systems greatly simplifies the implementation of search, since it is then unnecessary to implement the details of inheritance in the searcher.

Unlike the software written in sequential languages that can only consider one model at a time (in a Fortran block IF-THEN statement, for example), object-oriented systems allow multiple models to be queried as many messages are sent in parallel. This feature is likely to be particularly beneficial for complex identification problems with large model spaces, as it permits the searcher to simultaneously consider models at different levels and branches of the tree. Parallel querying of models in an object-oriented system also enables parallel searches, using different heuristic J's, to be conducted. If we use the department analogy, this would be equivalent to a department head asking different staff members for the same dynamic analysis to be conducted in several different ways, such as by closed-form distributed-parameter system solution and by the use of the NASTRAN and STRUDL finite element analysis programs, all the results of which could then be compared.

Object-oriented systems are useful for us since they also implement frame systems.[9] A frame system organizes knowledge in a multileveled fashion thought to be similar to natural intelligence. Frames have been used to represent knowledge for vision systems,[9] natural language understanding systems,[15-18] and other reasoning systems and have been used extensively in recent expert systems.

Comparative Examples

The simple spring-mass example used previously may clarify some of the advantages of an object-oriented approach over a conventional procedure-oriented approach. This example, however, will not demonstrate all of the specific benefits of object-oriented programming. Consider a very simple model space, composed of the undamped one-degree-of-freedom systems shown in Fig. 2.

As shown in Fig. 5a, in a conventional procedural approach the model response subroutines called to generate the evaluation criteria J for each model must access both the common parameters and the parameters specific to their model. The logical links between models at succeeding levels of abstraction are internal to the main program.

In the object-oriented approach shown in Fig. 5b, the equations specific to each model are located with their model parameters. The levels of abstraction are defined in the knowledge representation, and not in the main program. Finally, the subclass-superclass object relations that link the models permit the third model (1-spring, default mass) to inherit the mass value from the first model, as well as enable the inheritance of s by all the models. Note that queries to low-level objects pass automatically through the intermediate classes.

A characteristic of models at lower levels of abstraction is that they are similar, but distinct, from a model at a higher level. The class structure of a frame system corresponds to this by using inherited parameters at lower levels with only the distinguishing parameters v specified in the lower models. The class structure of an object-oriented system, together with its built-in inheritance characteristics, is then superior to a conventional procedural approach in meeting our need for levels of abstraction and inheritance.

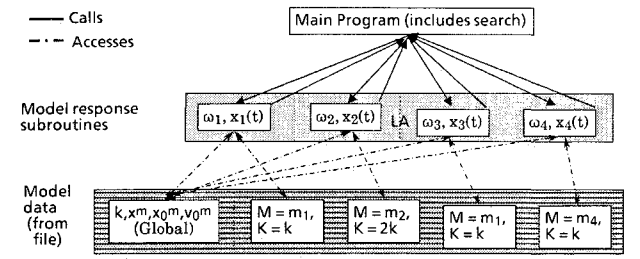

Fig. 5a Spring-mass example model space, implemented with a conventional procedural (e.g., Fortran) approach.

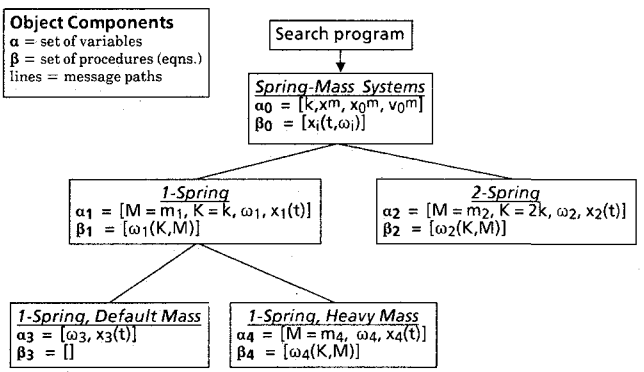

Fig. 5b Spring-mass example model space, implemented as an object-oriented system.

Search: The Best-First Approach

Model Search

To minimize processing time, a directed, rather than blind, search method should be used. Directed search methods include both heuristic and nonheuristic types. Examples of nonheuristic search are depth-first search, which completely explores every branch of the search tree to its bottom before trying another, and breadth-first search, which checks every node at a given level in the search tree before checking any at the next level.[23] If a set of evaluation criteria exists for the nodes, it may be used for the direction of heuristic search methods. One type of heuristic search, called best-first search, sorts the list of evaluated nodes after each addition to the list, choosing the best node for the direction of the next search step. A common best-first algorithm called A^* has been proven by Hart et al.[11] to return with the optimum node in the search space, provided that the heuristic evaluation function provides an optimistic estimate of the distance remaining to the optimal or goal state. For this problem, the search method selected is best-first, with a model evaluation heuristic defined as a weighted combination of squared frequency error and Allemang and Brown's[24] modal correlation coefficient.

$$e_m = \frac{(\Phi^T \cdot \Phi^m)^2}{(\Phi^T \cdot \Phi)(\Phi^{mT} \cdot \Phi^m)}, \qquad e_m \in [0,1]$$

$$e_c = \left[\frac{(\omega^m - \omega)^2}{\omega^{m2}}\right] \tag{29}$$

$$J = a\, e_c + b(1 - e_m)$$

where a and b are weights. As discussed, optimal search avoids calculating the performance index J for every model, at the cost of organizing the model space with levels of abstraction.

An identified model is assumed to remain valid until J violates a preset bound. When this occurs, the model is considered suspect, so best-first search[23,25] is then begun by forming a queue, consisting initially of just the root object of the search tree (which here is associated with no specific model). The children (subclasses) of this object, which correspond to the first level of models, are then substituted for it, and the queue is then sorted by the least estimated error, resulting in the model with the least error being placed at the front of the queue. The process repeats recursively until the first model in the queue either satisfies the heuristic threshold constraints (i.e., a match), or resides at the bottom level of the search tree (in which case the first model is also returned, being the closest to a match).

Procedures within each object, triggered by requests for variable values from the searcher and from higher-level procedures, calculate or retrieve the model's simulated modal response for comparison with the actual response. When triggered by the search process, these small attached programs calculate estimates of mass values and locations, given the current output from the physical system. If exact solutions are available, similar relatively small procedures may also calculate the vibration frequencies and mode shapes used by the searcher. These procedures embody the functions defined in (16). For complex structures, stored vibration frequencies and modes, obtained off-line from a discrete degree-of-freedom analysis, are instead retrieved for each model.

The search implementation used in this work differs from conventional best-first search[11,23,25] in the structuring of the search tree to include each node in the set of its children. A prototype model may be a better match to the current physical system than any of its variants, so it should remain in the search queue in case the end of the search tree is reached without finding an exact match. A side effect of this tree structuring is that search is progressively forced to commit to branches of the model space as depth increases, i.e., does not allow consideration of branches other than those emanating from the parent node, since $\min J^{(\mathrm{LA}=i)} \leq \min J^{(\mathrm{LA}=i+1)}$. This elimination of backtracking improves efficiency, is in keeping with the taxonomic organization of the models, and its optimality depends on the assumption of a well-ordered R.

Search Convergence Conditions

Let Δh_{ab} denote the magnitude of the difference in the heuristic model evaluation function between models a and b, or

$$\Delta h_{ab} = |J_a - J_b| \tag{30}$$

At a given level of abstraction k in a search tree,

$$\overline{\Delta h}(k) = \text{largest } \Delta h \text{ at level } k$$

$$\Delta h^*(k) = \text{smallest } \Delta h \text{ at level } k \tag{31}$$

are defined as the maximum and minimum values of Δh at level k, or the largest and smallest model separations in terms of J.

It should be recalled that in a best-first search procedure, the list of open nodes is sorted after every expansion. In order to avoid backtracking in the search tree, the correct model must be chosen for expansion at each level to insure that the search converges to the optimal match. If we assume that similar structural models have similar values of J, the closest model at a given level is that which minimizes J at that level. Define the J value for the closest model at level k as

$$J^*(k, m_1) = J(\mu^*, \theta) \tag{32}$$

where m is an index and the parameters θ are assumed known, and let the next-best model in terms of J be

$$J_n(k, m_2) = J(\mu_n, \theta) \tag{33}$$

where

$$J^*(k, m_1) + \Delta h^*(k) \leq J_n(k, m_2) \leq J^*(k, m_1) + \overline{\Delta h}(k) \tag{34}$$

At a lower level of abstraction in the search tree, these models have sets of variations, or families F, in which each parent model is included for a model space

$$J^*(k, m_1) \epsilon F_b(k+1, m_{1i}), \quad i = 1, 2, \ldots, n_b$$

$$J_n(k, m_2) \epsilon F_n(k+1, m_{2j}), \quad j = 1, 2, \ldots, n_n \tag{35}$$

In the foregoing notation, F = family, n_b = number of children in F_b, n_n = number of children in F_n. Within each of these families of models, as shown in Fig. 6, some models will minimize J(if we assume distinct model responses)

$$J_b^*(k+1, m_3) \epsilon F_b(k+1, m_{1i}), \quad i = 1, 2, \ldots, n_b$$

$$J_n^*(k+1, m_4) \epsilon F_n(k+1, m_{2j}), \quad j = 1, 2, \ldots, n_n \tag{36}$$

Note that $J_n^*(k+1)$ is the best member of $F_n(k+1)$. It is not necessarily the same as its parent $J_n(k)$. Since the correct model chosen at level k is also a member of F_b(because it is the parent of F_b), it then provides an upper bound at level $k+1$

$$J_b^*(k+1, m_3) \leq J^*(k, m_1) \tag{37}$$

Likewise, from (34), $J_n(k)$ provides a bound on its family at level $k+1$

$$J_n(k, m_2) - \overline{\Delta h}(k+1) \leq J_n^*(k+1, m_4) \leq J_n(k, m_2) \tag{38}$$

In order for the model corresponding to $J^*(k, m_1)$ to have been the wrong direction of search, it is necessary that the best member of F_n[i.e., $J_n^*(k+1)$] be better than the best member of F_b [i.e., $J_b^*(k+1)$] or

$$J_n^*(k+1, m_4) < J_b^*(k+1, m_3) \tag{39}$$

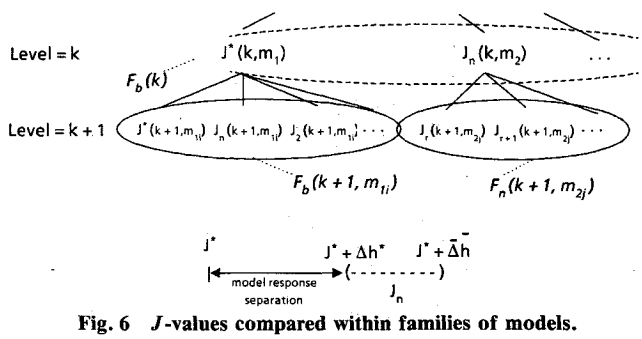

Fig. 6 J-values compared within families of models.

In order to relate J_b^* to J_n^*, suppose that J_n in (34) and (38) is taken at the minimum possible value, and J_b^* in (37) is taken at the maximum possible value. This results in the closest possible interval between the two values, such that the distinction between them is minimized

$$J^*(k, m_1) + \Delta h^*(k) = J_n(k, m_2)$$

$$J_n(k, m_2) - \overline{\Delta h}(k+1) = J_n^*(k+1, m_4) \tag{40}$$

$$\text{and} \quad J_b^*(k+1, m_3) = J^*(k, m_1)$$

(40) then equals

$$J_b^*(k+1, m_3) + \Delta h^*(k) - \overline{\Delta h}(k+1) = J_n^*(k+1, m_4) \tag{41}$$

If (39) is true, then (41) would become a requirement that for the wrong direction to have been taken in search at level k, then the search tree would have to have a model separation at level $k+1$ larger than the smallest separation at level k

$$\Delta h^*(k) - \overline{\Delta h}(k+1) < 0 \tag{42}$$

$$\text{or} \quad \Delta h^*(k) < \overline{\Delta h}(k+1)$$

Therefore, if a "well-ordered" search tree is constructed such that differences in model response in terms of a chosen J clearly diminish at progressively lower levels of abstraction, then

$$\Delta h^*(k) > \overline{\Delta h}(k+1) \tag{43}$$

contradicts (42) and is therefore a sufficient condition to insure that the best model is always found without backtracking, if we assume it falls within R. It is to be noted that if (43) holds, then (37) will describe a search with a monotonically decreasing J.

It is also noteworthy that in a conventional search tree, in which nodes do not include lower-level versions of themselves in their set of children, the worst case becomes instead

$$J_b^*(k+1, m_3) = J^*(k, m_1) + \overline{\Delta h}(k+1) \tag{44}$$

If we proceed as just shown the result requires that

$$2\,\overline{\Delta h}(k+1) < \Delta h^*(k) \tag{45}$$

which effectively halves the allowable model discrimination at level $k+1$ for optimality to be insured without backtracking. Otherwise, the search must either backtrack or degenerate into a suboptimal method known as "hill-climbing."

Effect of Parameter Set Error

The foregoing analysis assumes knowledge of the true parameter set θ. If θ is in error, the relative values of the model evaluations may be affected. It has been shown[26] that for parameter set errors bounded by

$$|\Delta\theta| < \delta_b \tag{46}$$

and if δ_b is sufficiently small, Eq. (43) and the proof of convergence will be preserved provided

$$\frac{\partial J}{\partial \theta} \Delta\theta > 0 \tag{47}$$

is true for the true identified model.

In a given well-ordered search tree, Δh bounds the allowable evaluation error in $J(\mu, \theta + \Delta\theta)$ since by the definition of best-first search the "true" model at each level of abstraction must be the closest in response to that measured from the

apparent model for proper convergence to occur. At each level of the search k,

$$\frac{\Delta h}{2} > |J(\mu, \theta + \Delta\theta) - J(\mu, \theta)| \tag{48}$$

Specific values of δ_b for a well-ordered search tree will depend on the chosen form of J and the Δh that separates the models. The need for a well-ordered search tree and the development of approximate search functions and bounds on the initial estimates of parameters are illustrated in the example given in the appendix.

Model Evaluation Function: An Example

In this paper, the model evaluation function J is

$$J = \sum_{j=1}^{N_M} \left\{ [\omega_j^m - \omega_j(k, m, \theta_i)]^2 + [1 - c_{\phi_j}(k, m, \theta_{ij}, \phi_j^m)] \right\} \tag{49}$$

where c_{ϕ_j} is e_m from (29). The value of c_{ϕ_j} ranges from zero for orthogonal modes to one for linearly dependent modes, for all ϕ_j.

Perturbing the modal correlation coefficient in (29) and assuming that denominator perturbations may be ignored for small $\Delta\phi$, we obtain

$$c_{\phi + \Delta\phi} = \frac{\left([\phi + \Delta\phi]^T \cdot \phi^m\right)^2}{\left(\phi^T \cdot \phi\right)\left(\phi^{mT} \cdot \phi^m\right)} \tag{50}$$

Neglecting the higher-order perturbations yields

$$c_{\phi + \Delta\phi} = c_\phi \left[1 + \frac{2(\Delta\phi^T \cdot \phi^m)}{(\phi^T \cdot \phi^m)} \right] \tag{51}$$

Given (46), it is possible to neglect the higher-order differences ($\Delta\theta^2$, $\Delta\theta^3$, etc.), so

$$\left(\frac{\partial J}{\partial \theta}\right)_i = \lim_{\Delta\theta_i \to 0} \frac{J(k, m, \bar{\theta} + \Delta\theta_i) - J(k, m, \bar{\theta})}{\Delta\theta_i} \tag{52}$$

Alternatively, this can be expressed as

$$\frac{\partial J}{\partial \theta} \Delta\theta = J(k, m, \bar{\theta} + \Delta\theta) - J(k, m, \bar{\theta}) \tag{53}$$

For a given mode,

$$\frac{\partial J_j}{\partial \theta}\Delta\theta = \left\{ \left[\omega_j^m - (\overline{\omega_j} + \Delta\omega_j)\right]^2 - \left[\omega_j^m - \overline{\omega_j}\right]^2 \right\} + \left[(1 - c_{\phi + \Delta\phi}) - (1 - c_\phi)\right] \tag{54}$$

Simplifying for the true model since $(\omega^m - \bar{\omega}) = 0 = (1 - c_{\bar{\phi}})$, we get

$$\left[\frac{\partial J_i}{\partial \theta}\Delta\theta\right]_{\overline{km}} = \Delta\omega^2 + \left[1 - \frac{2(\Delta\phi^T \cdot \phi^m)}{(\phi^T \cdot \phi^m)}\right] \tag{55}$$

Since $c_{\bar{\phi}} = 1$, from (51)

$$-1 \leq \frac{2(\Delta\phi^T \cdot \phi^m)}{(\phi^T \cdot \phi^m)} \leq 0 \tag{56}$$

Hence, satisfying (47),

$$\left[\frac{\partial J_i}{\partial \theta}\Delta\theta\right]_{\overline{km}} \geq 1 + \Delta\omega^2 > 0 \tag{57}$$

so the correct model will still be identified for the parameter set error that satisfies (46).

Parameter Search

The identified model may include initial parameter estimates, such as mass value and location, in addition to the precomputed finite element parameters or other analytical values associated with the model that are retrieved upon model identification. The superposition of parameter estimates with retrieved parameters can then be given as input to a recursive parameter identification program, such as MCKID,[5] to quantitatively identify the model parameters. In this approach, a priori values of θ_r—in this case, mass, stiffness and damping matrices—are substituted into the governing set of linear differential equations pertaining to the given model μ. These equations are uncoupled using the eigensolution of the a priori model. If we assume that incorrect θ_r values of mass, stiffness, and damping are present in the a priori model, the governing equations will define an error matrix

$$E_{ij} = \sum_{m=1}^{N} \left(M_{im} U_{mj} + C_{im} V_{mj} + K_{im} W_{mj} \right) \tag{58}$$

where

$$U = \Lambda^2 \phi, \qquad V = \Lambda\phi, \qquad W = \phi \tag{59}$$

with Λ and ϕ defined as the diagonal eigenvalue matrix and eigenvector matrix, respectively. Identification then proceeds by minimizing the Euclidean norm of E by varying the elements of the system matrices, subject to the constraint of matrix symmetry. This approach has been shown to find accurate, symmetric mass, stiffness, and damping matrices, whether the damping is proportional or nonproportional. Parameter identification, following model identification, completes one cycle of the system identification process for time-varying distributed-parameter systems. Only one cycle of identification has been used for the time-varying systems considered in this paper, and this has been found to be satisfactory; however, complex systems may require several identification iterations.

Application of AI-Based Model Identification Procedure

A taxonomy of structural models was constructed as a lattice of Loops[22] objects, representing simple beam models with added masses at various locations, as in the tree shown in Fig. 7. The structural dynamics theory embodied in each model was that of simple beams without significant shear or rotary inertia effects, i.e.,

$$\frac{\partial^2}{\partial \alpha^2}\left[EI\frac{\partial^2 w}{\partial x^2}\right] = \omega^2 m w \tag{60}$$

Geometric BC's w', w

Model space:

$R \equiv$ {cantilever, supported, cantilever with tip mass, supported with added mass, clamped-supported}

Heuristic search was conducted in this simple model space considering the simulated responses, which were generated from either the closed-form solutions or finite-element analyses for each model. Two levels of abstraction were considered:

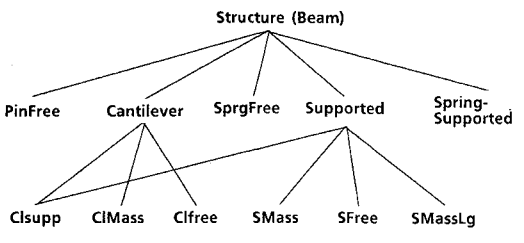

Fig. 7 A model space containing simple beam models.

Input Response	Identified Model
(2.6996 ((0.0 0.0) (.2381 .0888) (.4286 .2598) (.619 .4853) (.8095 .7385) (1.0 1.0))) (.3248 ((0.0 0.0) (.2381 .07843) (.4286 .2365) (.619 .4566) (.8095 .718) (1.0 1.0))) (11.841 ((0.0 0.0) (.2381 .4036) (.4286 .8633) (.619 1.0) (.8095 .6655) (1.0 0.0))) (7.579 ((0.0 0.0) (.2381 .6977) (.4286 1.0) (.619 .9548) (.8095 .5778) (1.0 0.0)))	(#$Cantilever.1 .00006311804 1.0 (.0002888 .00002849 -.00001633 6.142E-6 -2.054E-6) (.00002849 .0002815 .00003115 -.0000151 6.459E-6) (-.00001633 .00003115 .0002823 .00002513 -.00001856) (6.142E-6 -.0000151 .00002513 .0002974 .00005039) (-2.054E-6 6.459E-6 -.00001856 .00005039 .00008936)) ** (#$ClMass.1 .3936654 1.0 (.0002888 .00002849 -.00001633 6.142E-6 -2.054E-6) (.00002849 .0002815 .00003115 -.0000151 6.459E-6) (-.00001633 .00003115 .0002823 .00002513 -.00001856) (6.142E-6 -.0000151 .00002513 .0002974 .00005039) (-2.054E-6 6.459E-6 -.00001856 .00005039 .3937548)) (#$Clsupp.1 .0001168586 .619 (.0002888 .00002849 -.00001633 6.142E-6 -2.054E-6) (.00002849 .0002815 .00003115 -.0000151 6.459E-6) (-.00001633 .00003115 .0002823 .00002513 -.00001856) (6.142E-6 -.0000151 .00002513 .0002974 .00005039) (-2.054E-6 6.459E-6 -.00001856 .00005039 .00008936)) (#$Supported.1 .0001570419 1.0 (.0003536 3.91E-6 -8.688E-6 3.993E-6) (3.91E-6 .0002907 .00002833 -.00001432) (-8.688E-6 .00002833 .0002832 .0000249) (3.993E-6 -.00001432 .0000249 .0002975))
Format: (ω (ϕ)) Shown: 5-dof beam models, first frequencies and mode shapes	Format: (Name Added-Mass Mass-Location (M-matrix)) ** denotes a best, rather than exact match

Physical Parameters: Aluminum, Length = 47.25in, Width = .75in, Height = .1875in

Fig. 8 A typical test run in the beam model space, using first-order response input.

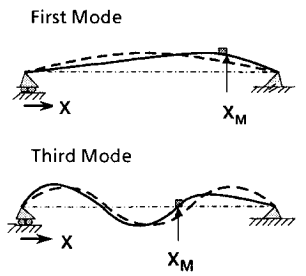

Fig. 9 The mass location heuristic for supported beams was based on node or peak shift interpolations.

first, prototype models representing general configurations, then more detailed models including added masses. These corresponded to the second- and third-level nodes in the graph in Fig. 7. The root node (structure) was associated with the response of the previously identified model, representing the previous state of the physical structure. If the response associated with the root model satisfied the model evaluation threshold at a given response input, search was terminated and the current model remained in effect. If the threshold was violated, the identity of the previous model was expunged from the root and search continued with the first-level set of models. The simulated responses were experimentally verified for the cantilever and cantilever-with-tip-mass models.

The response input data, which were also experimentally verified with measurements of actual structures, were generated by corrupting analytically obtained frequencies and mode shapes with 10% noise. This was done in order to resemble the input produced by a response feature extractor, such as that suggested by Spriet and Vansteenkiste.[27] Examples of response input and the corresponding identifications are given in Fig. 8.

Superposition of flexibility influence coefficients and reference to tables of empirically obtained data were used to relate frequency shift to the addition or movement of lumped point masses on a continuous beam. For instance, tip mass values for cantilevers were estimated with the relation

$$\frac{M}{m} = \alpha\left[\left(\frac{\omega_d}{\omega}\right)^2 - 1\right] \tag{61}$$

obtained both from curve-fitting the results of finite-element analyses and from experiment, where ω_d was the default natural frequency without added mass, ω the frequency with mass

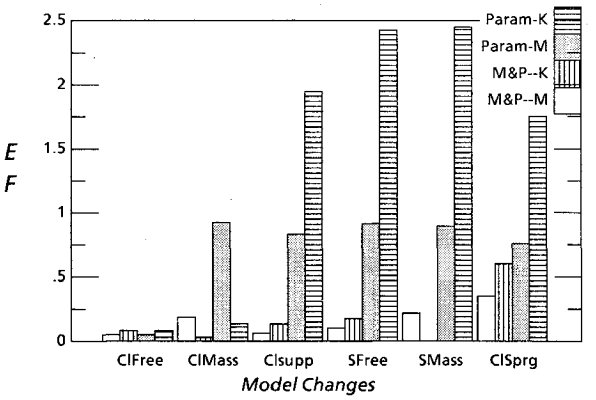

Fig. 10 Model identification and parameter identification vs parameter identification only (initial model was a cantilever beam, model C1Free).

added, M the tip mass, m the mass of the beam, and α a constant dependent on the ratio of the closed-form frequency coefficients of ω_d and ω. Mass locations on supported beams were estimated from node shifts, as shown in Fig. 9, which were found by comparing the observed mode shape to the analytical mode of the same order associated with the same model without added mass.

Once a model was returned by the search procedure, a stored finite element model associated with that model or class of model was retrieved. Mass value and location estimates, if any, were used to correct the parameters of this finite element model. A parameter-identifying program that used the equation-error approach, MCKID,[5] was then given both the identified model and the system response as its input.

Results

A comparison of the models produced by using both model identification and parameter identification with those produced using parameter identification alone was made. In the former process, models and parameters were identified in the manner developed in this paper and discussed in the previous section, implemented on a Xerox 1108. These models, and the current system response, were given as input to MCKID. In the latter process, the a priori model given to MCKID was held constant, while the input system response data varied. The initial model postulated was an unmodified cantilever, and identical response input was given to both procedures. As shown in Fig. 10, the average parameter error, defined as

$$EF = \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{(p_{ij} - p_{ij}^*)^2}{p_{ij}^2}, \quad p_{ij} \in [M, K] \tag{62}$$

for both mass and stiffness matrices was markedly lower when model identification was employed, even for a structure (clamped-spring-support, *ClSpring*) not included in the model space R. This performance comparison was slightly influenced by the tendency of the numerical parameter identification algorithm to spread mass changes over several elements, but comparable results shown here for models without added masses (supported, clamped-supported) indicate that the effect is a minor one. The provision of several, rather than one, degrees of freedom of response to the standalone parameter identification program was not found to significantly improve error vis à vis the use of model identification, despite the provision of only one degree of freedom to the model identification program.

Conclusions

An AI-based technique using heuristic search has been developed to identify time-varying structural dynamic system models. Both model and parameters can be identified. The

developed technique has been used to identify a selected system. The test runs indicate the superiority of this model- and parameter-identifying approach over conventional parameter identification techniques used without model identification.

It has also been demonstrated that the knowledge representation methods of abstraction levels in a model space, and object- or frame-oriented programming with data inheritance, provide an efficient procedure for structural dynamics model identification.

It was found that even in the presence of significant noise, the search returned the best fitting model available, in the relatively small search space considered. In the example given here, the separation of the models' responses was greater than the error level, so error did not affect the model identification process. If the model responses were closely spaced, the error in the measurement of ω's and ϕ's would have to be eliminated by using cross-correlation or other estimation techniques. The occurrence of input from a model not in the model space reduced the accuracy of initial parameter estimates, but did not abort the search, demonstrating a degree of fault tolerance inherent in the AI approach used. Integration of this model form characterization procedure with a parameter identification program resulted in the quantitative identification of model parameters, as shown in Fig. 10, with much better accuracy than parameter identification used alone. The process of convergence of the model identification procedure depends on the establishment of a model space, a well-ordered search tree, an appropriate search function and knowledge of parameters within appropriate established bounds.

Further work will include the optimization of the search heuristics for use in large model spaces, examination of the necessary levels of abstraction, and the integration of model identification with adaptive controls.

Appendix: Effects of Parameter Errors on Model Identification

As discussed in this paper, a well-ordered search tree and corresponding bounds on the initial estimates of the parameters are necessary to assure convergence of the described model identification procedure. This is illustrated by considering a simple example similar to the system of Fig. 2. It is assumed that $m_1 = 1$, $m_4 = 2$, or $k_1 = 100$. In order to slightly complicate the system, a spring-mass system with $m_4 = 2$ and $k_1 = k_2 = 100$ are added to the right branch. It should be noted that the natural frequency of the system is same as the system shown with $m_1 = 1$ and $k = 100$ on the left branch. The number of springs and quantitative values of mass separate the two systems. Another system with $\omega^2 = 150$ has been added to the right branch to provide uniform separation of J_m values at the level of abstraction $k = 2$. A block diagram of the system is given in Fig. A1. Note that here, k without subscripts refers to modex index rather than stiffness parameters, as is given in the nomenclature.

Because a knowledge of frequencies alone is not sufficient to discriminate between some of the systems, the model evaluation criteria must include characteristics other than $J = \Delta\omega^2$. In this example, model evaluations are performed by selecting smaller values of

$$\Delta\omega^2 = (J_m - J^m) \qquad \text{(A1)}$$

where J_m are the values of J established for each model in the search tree. Similarly, J^m are the measured values of J.

Because a knowledge of frequencies alone cannot separate some of the systems, the quantity J must include characteristics other than ω^2. For example, for a two-level function:

$$J = (\Sigma k_i)p_k + \Delta\omega^2 q_k$$

$$p_k(k = 1) = 1; \qquad p_k(k = 2) = 0$$

$$q_k(k = 1) = 0; \qquad q_k(k = 2) = 1 \qquad \text{(A2)}$$

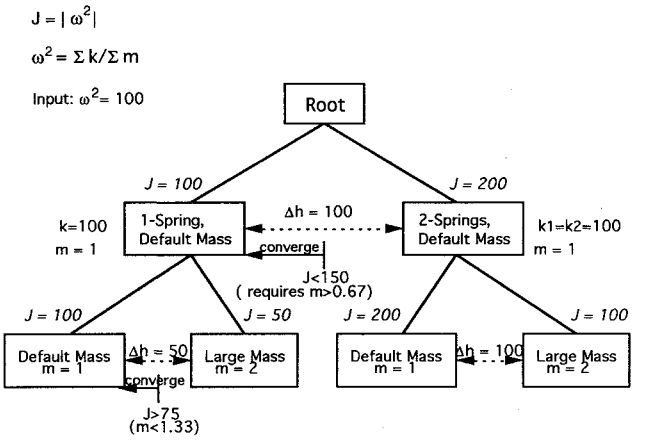

Fig. A1 Search diagram for spring-mass example.

The next step is to establish bounds on the initial estimates of the parameters (or the initial knowledge about the parameters) for the required convergence. This is accomplished by using Eqs. (46) and (47) i.e., for $k = 1$ and right branch spring $k_1 = \Sigma k_i$

$$\frac{\Delta h^*}{2}(k = 1) = 50 \geq |100 - \delta_a| \qquad \text{(A3)}$$

where δ_a is the allowable error on the initial estimate (or the knowledge) of the spring constant $\bar{k}_i$. From this inequality it is found that δ_a cannot exceed 50. Similarly, an error δ_b for the cases with Σk_i on the right branch can be shown to be 150 and 250, from the inequality

$$50 \geq |200 - \delta_b| \qquad \text{(A4)}$$

Then, in the right-branch models, the value of the spring constant k_1 can be known to be in the range of $50 < k_1 < 150$ and, similarly, limits on k_2 are $150 < k_2 < 250$. For the second level ($k = 2$), we have to establish the bounds on m depending on the given stiffness errors of k. Alternatively, we can establish bounds on ω^2. It is easily seen that the bounds on m must be such that the errors on ω^2 should be within a range of ± 25 of true values.

For example, if we start with a true value corresponding to $m_1 = 1$, $k_1 = 100$, $\omega^2 = 100$, the value of measured J^m for the level $k = 1$ will be 100. If the parameters were known to be near the error limit for the level $k = 1$ ($J_{\text{left}} = 149$ for the one spring case or left branch and $J_{\text{right}} = 151$ for the right branch), the identification process will then select the left branch:

$$|J_{\text{left}} - J^m| = 49 < |J_{\text{right}} - J^m| \qquad \text{(A5)}$$

picks the smaller of 49 and 51. At the second level ($k = 2$), the correct model will be selected as long as ω_1^2 (parameter) is known between $75 < \omega_1^2 < 124$ because

$$|J_{m=1} - J^m|_{\max} = 24 < 26 = |J_{m=2} - J^m| \qquad \text{(A6)}$$

is the smaller value of the two values 24 and 26.

It can be concluded that in some physical systems, such as certain structural dynamic systems, it is possible to obtain well-structured trees and convergence to the correct model when the errors are as large as, e.g., 24% and 49% in certain parameters.

Acknowledgments

This research was supported in part by U.S. Army Research Contract DAAG 29-82-K0094. The authors are grateful to Janet Kolodner for her helpful suggestions and criticism, and to Wayne Book for the use of his robotic test facilities. The

authors thank Xerox Corporation for providing hardware and software facilities.

References

[1]Bard, Y., *Nonlinear Parameter Estimation,* Academic Press, New York, 1974.

[2]Lewis, T. O., and Odell, P. L., *Estimation in Linear Models,* Prentice-Hall, Englewood Cliffs, NJ, 1971.

[3]Baruch, M., "Optimal Correction of Mass and Stiffness Matrices Using Measured Modes," *AIAA Journal*, Vol. 20, Nov. 1982, pp. 1623–1626.

[4]Berman, A., "System Identification of Structural Dynamic Models—Theoretical and Practical Bounds" *Proceedings of the AIAA 25th Structures, Structural Dynamics, and Materials Conference*; also AIAA Paper 84-929, Palm Springs, CA, 1984.

[5]Hanagud, S., Meyappa, M., Cheng, Y. P., and Craig, J. I., "Identification of Structural Dynamic Systems with Nonproportional Damping," *AIAA Journal,* Vol. 24, Nov. 1986, pp. 1880–1882.

[6]Leuridan, J. M., Brown, D. L., and Allemang, R. J., "Direct System Parameter Identification of Mechanical Structures with Application to Model Analysis," *Proceedings of the AIAA 23rd Structures, Structural Dynamics, and Materials Conference*; also AIAA Paper 82-767, 1982.

[7]Cheng, Y. P., "System Identification Methods for Non-Proportionally Damped Structures," Ph.D. Thesis, School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA, Jan. 1987.

[8]Sacerdoti, E., "Planning in a Hierarchy of Abstraction Spaces," *Artificial Intelligence,* Vol. 5, 1974, pp. 115–135.

[9]Minsky, M., "A Framework for Representing Knowledge," *The Psychology of Computer Vision*, edited by P. Winston, McGraw-Hill, New York, 1975, pp. 211–277.

[10]Gelertner, H., "Realization of a Geometry Proving Machine," International Conference on Information Processing, UNESCO, Paris, 1959; also *Computer and Thought,* edited by E. Feigenbaum and J. Feldman, McGraw-Hill, New York, 1963.

[11]Hart, P. E., Nilsson, N. J., and Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems, Science and Cybernetics,* Vol. SSC-4, 1968, pp. 100–107.

[12]Pearl, J., "Knowledge Versus Search: A Quantitative Analysis Using A*," *Artificial Intelligence,* Vol. 20, 1983, pp. 1–13.

[13]Pearl, J., *Heuristics*, Addison-Wesley, Reading, MA, 1984.

[14]Feigenbaum, E., "The Simulation of Verbal Learning Behavior," *Computers and Thought,* edited by E. Feigenbaum and J. Feldman, McGraw-Hill, New York, 1963, pp. 297-309.

[15]Schank, R., "Language and Memory," *Journal of Cognitive Science,* Vol. 4, 1980, pp. 243–284.

[16] Schank, R., *Dynamic Memory,* Cambridge University Press, New York, 1982.

[17]Kolodner, J. L., "Organization and Retrieval in a Conceptual Memory for Events," *Proceedings of International Joint Conference on Artificial Intelligence,* Vancouver, Canada, 1981, pp. 227–233.

[18]Kolodner, J. L., *Retrieval and Organizational Strategies in Concetual Memory: A Computer Model*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1984.

[19]Barsalou, L. W., and Bower, G. H., "Discrimination Nets as Psychological Models," *Journal of Cognitive Science,* Vol. 8, 1984, pp. 1–26.

[20]Goldberg, A., and Robson, D., *Smalltalk-80: The Language and Its Implementation,* Addison-Wesley, Reading, MA, 1983.

[21]Burke, G. S., et al., *NIL Reference Manual,* Version 0.286, Lab. for Computer Science, Massachusetts Inst. of Technology, Cambridge, MA, 1984.

[22]Bobrow, D. G., and Stefik, M., *The Loops Manual*, Knowledge Systems Area, Xerox Palo Alto Res. Ctr., Palo Alto, CA, Dec. 1983.

[23]Rich, E., *Artificial Intelligence,* McGraw-Hill, New York, 1983.

[24]Allemang, R. J., and Brown, D. L., "A Correlation Coefficient for Modal Vector Analysis," 1st International Modal Analysis Conference, 1982.

[25]Nilsson, N. J., *Principles of Artificial Intelligence,* Tioga Publishing, Palo Alto, CA, 1980.

[26]Hanagud, S., Glass, B. J., and Calise, A. J., "An AI-Based Model-Adaptive Approach to Flexible Structure Control," *Proceedings of the AIAA 28th Guidance, Navigation, and Control Conference*; also AIAA Paper 87-2457, Monterey, CA, Aug. 1987.

[27]Spriet, J. A., and Vansteenkiste, G. C., "Structure Characterization for Ill-Defined Systems," *Simulation and Model-Based Methodologies: An Integrative View*, edited by T. I. Oren et. al., Springer-Verlag, Berlin, 1984.