Networked Assembly of Affine Physical System Models

E. Motato
College of Engineering,
Pontificia Universidad Javeriana,
Cali, Colombia
e-mail: emotato@puj.edu.co

C. Radcliffe
Department of Mechanical Engineering,
Michigan State University,
East Lansing, MI 48824-1229
e-mail: radcliff@msu.edu

Engineering design is evolving into a global strategy that distributes engineering effort to team members around the world. Because modern engineering design uses analytical models, model information must be distributed globally through computer networks. This strategy would be improved if component suppliers were able to efficiently provide dynamic models of supplied components. Furthermore, to use these component models, they must be efficiently assembled to obtain a dynamic model of a product using them. Four characteristics are needed to enable this distribution and assembly process. These characteristics are a unique standard model format, an exchange of model information through a single-query network transmission, external component models protecting proprietary internal design details, and, finally, a recursive assembly process. The modular model assembly method (MMAM) (Radcliffe et al., 2009, "Networked Assembly of Mechatronic Linear Physical System Models," ASME J. Dyn. Syst., Meas., Control, ***131****, p. 021003) is a model assembly algorithm that satisfies these requirements. The MMAM algorithm assembles linear physical system models with dynamic stiffness matrices. In an affine system, deviations in the inputs and outputs exhibit a proportional relationship, but the outputs of the system are nonzero at zero input (Buck and Willcox, 1971, Calculus of Several Variables, Houghton Mifflin, Boston). One motivation for developing a process to assemble affine systems is the wide use of such models resulting from local linearization of general differentiable nonlinear physical system models about a nonzero, but constant, operating point. This paper provides the first general approach to the "operating point problem," where the operating points of each individual component are solved as a function of the desired operating point of the model of an assembly of those components. The solution of this problem allows the assembly of linearized system models at any requested system operating point. This paper extends the MMAM to nonlinear affine system models. The MMAM uses internet agents to provide external models of components when requested by either users or other model agents. Assembly agents use the models provided by component agents to build an analytical model using models provided by component agents and assembly constraints within the assembly model agent. MMAM models are supplied in a standard form that allows an assembly agent to put together efficiently a model of the assembly that is also in the standard form. The process is recursive and facilitates hierarchical use of agents to efficiently build assemblies of assemblies to any level of complexity.* [DOI: 10.1115/1.4002471]

1 Introduction

Engineering analysts are in the process of creating a global engineering strategy that is able to integrate product design, product development, marketing analysis, and manufacturing process [1]. Today's engineers can communicate with thousands of suppliers through supply chain management systems over the internet [2]. The trend is to use the internet as the media to achieve global engineering design.

Internet based performance evaluation of physical components is executed in three stages (Fig. 1). The first stage is modeling. Modeling is the process of generating a mathematical function that describes the input-output behavior of physical components. The second stage is component model distribution and assembly. Models of components in a standard format are distributed from remote computer servers through the internet and are assembled in a local computer server to obtain a model in the same standard format. This second stage is executed efficiently using the modular model assembly method (MMAM) procedure described in this work. The third and final stage is dynamic systems analysis. The dynamic performance of complex physical systems is analyzed through computer simulations using assembled dynamic models. The MMAM is neither a modeling methodology nor a system analysis tool. In contrast, the MMAM is a model distribution and assembly procedure suitable for use in a distributed internet engineering environment.

The second stage, distribution and assembly of dynamic models over the internet, requires four characteristics. They are models with unique standard format, information exchange through a single-query network transmission, dynamic models not revealing proprietary internal design details, and, finally, a recursive model assembly process. These attributes make global engineering design practical [3].

A unique standard component model format is the key to handling model exchange through model reuse [4]. A unique component model representation facilitates model query standardization, prevents model reformulation, and decreases model exchange time computation. The finite element method (FEM) [5] is an example of a modeling methodology that uses a unique standard model format. FEM uses the same mathematical format to represent both the elements and the system assembled from those elements. A modeling method that uses a unique standard format is called *modular*.

Single-query exchange of model information reduces network traffic during the assembly of dynamic models through the internet. A single-query model data exchange process retrieves the full component model using a single request and answer on the internet. Multiple iterative data exchange transmissions must be

Contributed by the Dynamic Systems Division of ASME for publication in the JOURNAL OF DYNAMIC SYSTEMS, MEASUREMENT, AND CONTROL. Manuscript received December 28, 2007; final manuscript received March 28, 2010; published online October 28, 2010. Assoc. Editor: Jeffrey L. Stein.

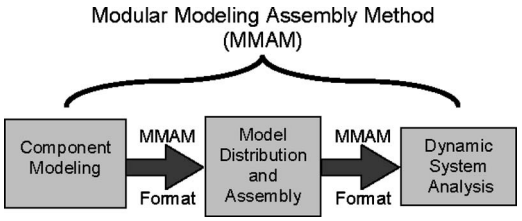

Fig. 1 Global engineering design procedures

avoided because repeated queries increase network traffic dramatically. A differential algebraic equation (DAE) [6] is an example of a single-query transmission model format. Global engineering strategy demands assembly processes that reduce network load through exchange of model information with a single query. A model assembly method using a single transmission format is called *single-query*.

An input-output model predicts external behavior using only input and output variables to protect internal proprietary design details. Because design is a dominant cost of new product development, internal product design details must be protected from competitors. These design details might include the components used in the assembly, the order of connection of such components, the physical parameters of the components, and the performance of each component. Protection of proprietary information is critical to the commercial acceptance of any model exchange system. Gu and Asada [2] used input and output variables to allow the co-simulation of a collection of dynamic subsimulators without disclosing proprietary information. A model assembly method that uses models that predict external behavior using input and output variables is called *external*.

A recursive model assembly process uses standard format component models to produce an assembly model in the same format. Once recursion is established at a single system model assembly level, the model assembly method is easily extended to higher-level, more complex system models. The DAE [6] is an example of an assembly methodology that recursively obtains DAE system models from either DAE elements or DAE subsystems. A model assembly process that uses standard format component models to produce an assembly model in the same format is called *recursive*.

All four characteristics are required simultaneously for a successful global engineering model assembly method. Co-simulation [2] is external but is not single-query because network iteration is required to simultaneously execute dynamic subsimulators. The DAE approach [6] is recursive and single-query but is not external because DAE models provide information about assembly components, component connectivity, and internal parameters. FEM [5] is modular and single-query but is not recursive. Assembly of FEM models generally requires a model global reformulation to guarantee geometric nodal compatibility. Bond graphs [7] and the behavioral approach [8] are modular but are not external because they provide information about assembly components, component connectivity, and internal parameters.

The MMAM [3] is a modular, single-query, external, and recursive model assembly and distribution procedure that protects proprietary design details. The MMAM is specifically designed to perform on a global networked design environment. It is important to clarify that MMAM is not a dynamic system modeling technique but a dynamic model distribution and assembly procedure. The past MMAM algorithm [3] assembled linear physical system models from standard format dynamic stiffness component models. This paper will extend previous linear MMAM procedures to the assembly of affine dynamic models. This affine approach is particularly important because it can also be used to assemble local linear approximations of general differentiable nonlinear physical system models performing around a constant operating condition.

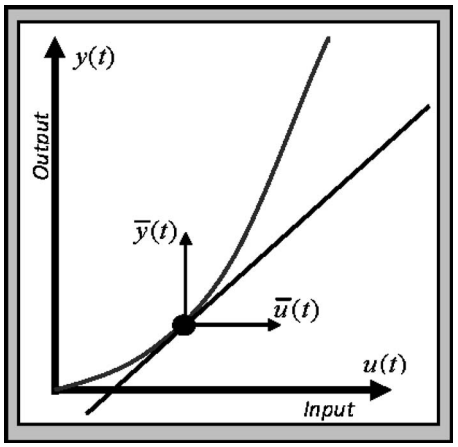

Fig. 2 Linearization yielding an affine system

2 Affine Physical System Models

Affine system models are an important class of nonlinear models. In an affine system, deviations in inputs cause proportional deviations in outputs, but system outputs are nonzero at zero input. Affine systems are nonlinear because they do not obey the two linear system properties: superposition and homogeneity [9]. Affine systems can be static or dynamic. A static affine system is characterized by an input-output relationship that is independent of the system's input and output time derivatives. Conversely, dynamic affine models include system input and output derivatives.

An affine system is an intermediate result of any linearization process. A linearization (Fig. 2) generates a linear model with respect to the operating point deviation variables $\bar{y}$ and $\bar{u}$ but an affine model generates a linear model with respect to the physical variables y and u. The MMAM for affine systems is important because, in general, the assembly of linear approximations of differentiable nonlinear systems performed at a specified operating condition can be executed as the assembly of the affine approximation for each of its components.

Assume a general differentiable r port nonlinear model

$$\mathbf{f}(\mathbf{y}(t),\dot{\mathbf{y}}(t),\ddot{\mathbf{y}}(t),\ldots,\mathbf{u}(t),\dot{\mathbf{u}}(t),\ddot{\mathbf{u}}(t),\ldots)=\mathbf{0} \tag{1}$$

where $\mathbf{f}(\bullet)$ is a $(r\times 1)$ vector of differentiable functions and $\mathbf{y}(t)$, $\mathbf{y}_o(t)$, $\mathbf{u}(t)$, and $\mathbf{u}_o(t)$ are $(r\times 1)$ time dependent vectors. An affine approximation of Eq. (1) results from a Taylor expansion about a specified operating condition $oc=(\mathbf{y}_o(t),\mathbf{u}_o(t))$ truncated to first order terms

$$\begin{aligned}\mathbf{f}(\mathbf{y}(t),\mathbf{u}(t),\dot{\mathbf{y}}(t),\dot{\mathbf{u}}(t),\ldots) \cong \mathbf{f}(\mathbf{y}_o(t),\mathbf{u}_o(t),\ldots) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{oc}(\mathbf{y}(t) \\ -\mathbf{y}_o(t)) + \left.\frac{\partial \mathbf{f}}{\partial \dot{\mathbf{y}}}\right|_{oc}(\dot{\mathbf{y}}(t)-\dot{\mathbf{y}}_o(t)) + \cdots \\ + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{oc}(\mathbf{u}(t)-\mathbf{u}_o(t)) + \left.\frac{\partial \mathbf{f}}{\partial \dot{\mathbf{u}}}\right|_{oc}(\dot{\mathbf{u}}(t) \\ -\dot{\mathbf{u}}_o(t)) + \cdots = \mathbf{0}\end{aligned} \tag{2}$$

This truncated Taylor expansion takes the general affine form

$$\begin{aligned}\mathbf{f}(\mathbf{y}(t),\mathbf{u}(t),\ldots) \cong \mathbf{f}(\mathbf{y}_o(t),\mathbf{u}_o(t),\ldots) + \mathbf{c} + \mathbf{N}_o\mathbf{y}(t) + \mathbf{N}_1\dot{\mathbf{y}}(t) + \cdots \\ + \mathbf{M}_o\mathbf{u}(t) + \mathbf{M}_1\dot{\mathbf{u}}(t) + \cdots = \mathbf{0}\end{aligned} \tag{3}$$

where the $(r\times r)$ matrices

$$\mathbf{N}_i = \left[\frac{\partial \mathbf{f}(\mathbf{y}(t),\mathbf{u}(t),\ldots)}{\partial (d^i\mathbf{y}/dt^i)}\right]\Bigg|_{oc} \tag{4}$$

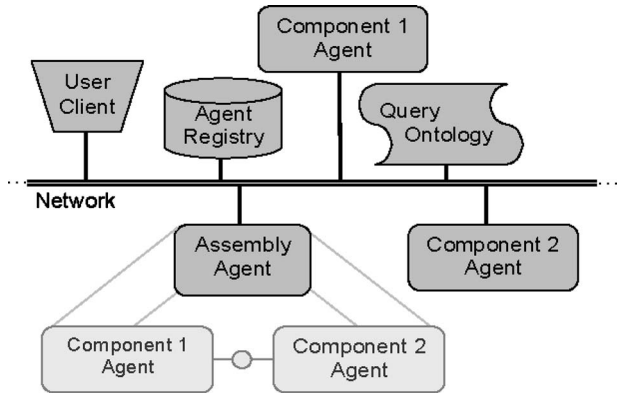

Fig. 3 Physical view of a networked modeling system [3]

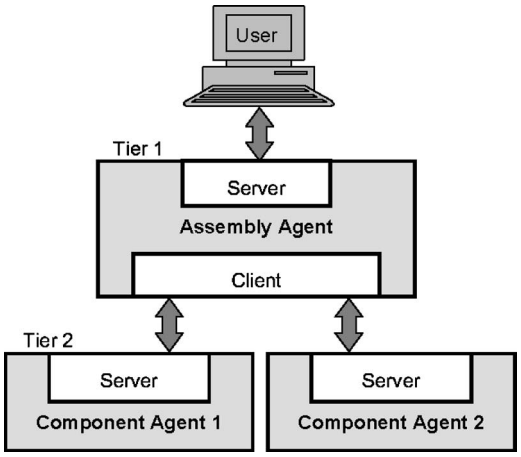

Fig. 4 Information flow between agents [3]

$$\mathbf{M}_j = \left[\frac{\partial \mathbf{f}(\mathbf{y}(t), \mathbf{u}(t), \ldots)}{\partial (d^j \mathbf{u}/dt^j)} \right] \Bigg|_{oc} \tag{5}$$

and $\mathbf{c}$ is a $(r \times 1)$ vector of constants.

A linear representation of Eq. (1) using Eq. (2) is found by applying two conditions. First, the term $\mathbf{f}(\mathbf{y}_o(t), \mathbf{u}_o(t), \ldots)$ must be removed by identifying input and output functions $\mathbf{u}_o(t)$ and $\mathbf{y}_o(t)$ that are a solution to the operating condition problem

$$\mathbf{f}(\mathbf{y}_o(t), \mathbf{u}_o(t), \ldots) = \mathbf{0} \tag{6}$$

Second, a change of variables is required to define a new set of variables that represent deviations from the operating condition found by solving Eq. (6). Define the new output vector

$$\bar{\mathbf{y}}(t) = [\mathbf{y}(t) - \mathbf{y}_o(t)] \tag{7}$$

and the new input vector

$$\bar{\mathbf{u}}(t) = [\mathbf{u}(t) - \mathbf{u}_o(t)] \tag{8}$$

Note that even if the operating condition problem is solved, the problem remains affine in the original physical variables because the operating point contributions to the Taylor expansion sum to a constant vector

$$\mathbf{c} = -\mathbf{N}_o \mathbf{y}_o(t) - \mathbf{N}_1 \dot{\mathbf{y}}_o(t) - \cdots - \mathbf{M}_o \mathbf{u}_o(t) - \mathbf{M}_1 \dot{\mathbf{u}}_o(t) - \cdots \tag{9}$$

and, in general, $\mathbf{c} \neq \mathbf{0}$.

The operating condition output response $\mathbf{y}_o(t)$ is traditionally defined first and substituted into Eq. (6) to find the required operating condition input solution $\mathbf{u}_o(t)$. For a linearizable system about a nonzero operating point, such a solution always exists. If it does not, the system is not linearizable at the requested response condition $\mathbf{y}_o(t)$. Substituting the deviation variables $\bar{\mathbf{y}}(t)$ and $\bar{\mathbf{u}}(t)$ into Eq. (2) yields the linearized model

$$\mathbf{N}_o \bar{\mathbf{y}}(t) + \mathbf{N}_1 \dot{\bar{\mathbf{y}}}(t) + \cdots + \mathbf{M}_o \bar{\mathbf{u}}(t) + \mathbf{M}_1 \dot{\bar{\mathbf{u}}}(t) + \cdots = \mathbf{0} \tag{10}$$

An appropriate output operating condition $\mathbf{y}_o(t)$ cannot be freely selected. It depends on the class of system analyzed. As an example, an output operating condition in the form $(\mathbf{y}_o(t) = \mathbf{c}_1, \quad \mathbf{u}_o(t) = \mathbf{c}_2)$, where $\mathbf{c}_1$ and $\mathbf{c}_2$ are constants, is not an appropriate selection for a system whose matrix $\mathbf{N}_o$ is zero because under this operating condition, Eq. (10) cannot be satisfied.

3 Internet Distribution of Affine Models

The performance of a networked model distribution system (Fig. 3) relies on the operation of autonomous and flexible computational systems on the internet called agents [10]. Four classes of agent are used: component agents, assembly agents, agent registry, and query ontology. Initially, a *user* connected to the internet consults the *agent registry* for the locations of *assembly agents* and *component agents* on the network. The user uses standardized queries obtained from the *query ontology*, which publishes a list of valid system queries and their formats. An assembly agent assembles models by using queries to lower-level component agents. Assembly and component agents respond to queries with models in the format specified by the ontology.

The networked distribution of models is hierarchical. Each agent has the capability of performing either as a client, as a server, or both. The user performs only as a client requesting models from lower-level agents. Assembly agents perform either as clients requesting models from lower-level agents or as servers providing models to higher-level agents. Finally, component agents perform only as servers providing model information to higher-level agents. In a two-level network example (Fig. 4), the user requests model information from a tier 1 assembly agent's server. The tier 1 assembly agent's client requests model information from tier 2 component agent servers. Starting at the lowest tier in the system, model information is provided by servers to higher-level assembly agents that assemble them as necessary. These assembled models are then provided recursively to agent servers that respond to queries from above.

The networked distribution of affine models requires a two-part query-response format. In the first part, agent clients request from lower-level server agents the models of systems valid around specific output operating conditions. In the second, server agents provide the models and the input operating conditions required to operate the system at the desired outputs. The process of computing component operating conditions from assembly operating condition is treated in a future section.

In the two-tier network example's information flow (Fig. 5), the user client makes a request to the tier 1's server agent for a model of an assembly model valid around the user's desired assembly operating output condition $\mathbf{y}_{a,o}$. The assembly agent determines that two component models are required. They are the component 1 operating around the operating condition output $\mathbf{y}_{c1,o}$ and the component 2 operating around the operating condition output $\mathbf{y}_{c2,o}$. The tier 1 assembly client requests a model from each of the two component agents. Two responses are generated. In the first response, the component 1 server returns the component 1 model and the required input $\mathbf{u}_{c1,o}$ to operate it around $\mathbf{y}_{c1,o}$. In the second response, the component 2 server returns the component 2 model and the required input $\mathbf{u}_{c2,o}$ to operate it around $\mathbf{y}_{c2,o}$.

The assembly agent uses the MMAM, the component model information, and its knowledge of the assembly's topology to execute the assembly of both a model and that assembly model's input $\mathbf{u}_{a,o}$ at the operating condition. The assembly agent then returns to the user client both the assembly model and the required assembly model inputs $\mathbf{u}_{a,o}$ at the desired outputs $\mathbf{y}_{a,o}$.

If the assembly model is constructed from local linearizations from one or more of its components, there exists a resultant error

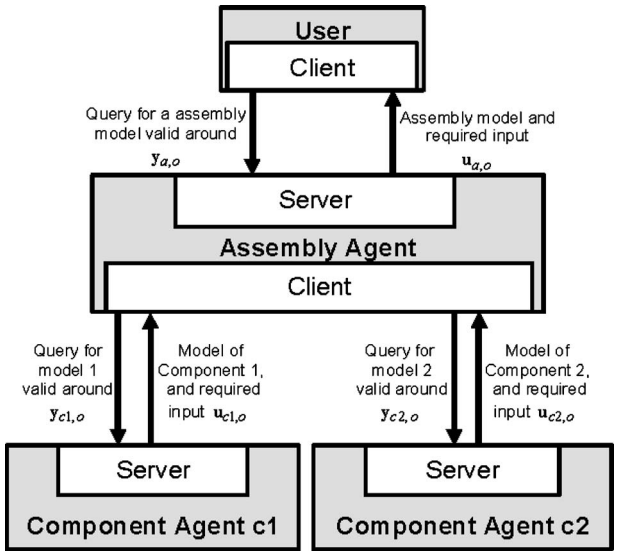

Fig. 5 Information flow for a two-level model assembly

associated with those component linearizations in the assembled model. Assembly model error analysis from component model linearization error is outside the distribution and assembly topics of this paper and is the subject of future work.

4 Standard MMAM Affine Model Formats

A standard dynamic model format used in the MMAM and the standard operating condition transformations from deviation variables into physical variables facilitate standard efficient processes to produce assembly models and their deviation-physical model variable transformations. In the proposed networked environment, component models are distributed as linear models defined in deviation variables but assembled as affine models defined in physical variables. Initially, the definition of physical systems and port-based models is presented, then the standard MMAM port-based format is shown, and, finally, the required variable transform equations are defined.

A physical system is an entity separated from the environment that interchanges energy through a boundary [7]. Physical systems are composed of interacting components that perform in a synchronized way to generate an energy flow. This energy flow is transferred through physical connections consisting of output-input pairs called ports. The product of the input and output variables of a port defines the energy flow through the port. For the procedures described in this paper, positive energy flow through a port is defined as the work done on that system. The total energy flow into a physical system is the sum of all the energy flows through each of its ports.

Physical systems can be modeled using a port-based approach. Port-based models always have an equal number of inputs and outputs because an input-output pair defines each port [7]. Port-based models are considered external models if those models are defined only as functions of the model's external port variables. A valid external port-based model has independent external energy ports. This characteristic requires the number of model equations to be equal to the number of system port outputs.

The standard MMAM affine port-based model format can be obtained by substituting the change of variables $\bar{\mathbf{y}}(t)=\mathbf{y}(t)-\mathbf{y}_o(t)$ and $\bar{\mathbf{u}}(t)=\mathbf{u}(t)-\mathbf{u}_o(t)$ into the affine model (2) under operating conditions $(\mathbf{u}_o(t),\mathbf{y}_o(t))$ satisfying the operating condition problem (6). This two-part transformation yields a linear differential equation in the deviation variables $\bar{\mathbf{y}}(t)$ and $\bar{\mathbf{u}}(t)$

$$\mathbf{N}_0\bar{\mathbf{y}}(t)+\cdots+\mathbf{N}_n(d^n\bar{\mathbf{y}}(t)/dt^n)=\mathbf{M}_0\bar{\mathbf{u}}(t)+\cdots+\mathbf{M}_m(d^m\bar{\mathbf{u}}(t)/dt^m) \tag{11}$$

Equation (11) describes an affine approximation of a differentiable r port-based nonlinear system. Some nonlinear systems with discontinuous nonlinear representations (i.e., coulomb friction) are not linearizable and Eq. (11) does not exist at those discontinuities. In a port-based model, the matrices $\mathbf{N}_i$ and $\mathbf{M}_j$ are $(r\times r)$ square matrices defined in Eqs. (4) and (5). Applying the Laplace transform into Eq. (11) yields

$$\mathbf{N}(s)\bar{\mathbf{Y}}(s)=\mathbf{M}(s)\bar{\mathbf{U}}(s) \tag{12}$$

where $\bar{\mathbf{Y}}(s)$ and $\bar{\mathbf{U}}(s)$ are, respectively, the Laplace transform of the deviation variables $\bar{\mathbf{y}}(t)$ and $\bar{\mathbf{u}}(t)$ and the two $(r\times r)$ polynomial matrices

$$\mathbf{N}(s)=[\mathbf{N}_0+\mathbf{N}_1 s+\cdots+\mathbf{N}_n s^n] \tag{13}$$

$$\mathbf{M}(s)=[\mathbf{M}_0+\mathbf{M}_1 s+\cdots+\mathbf{M}_m s^m] \tag{14}$$

Two commonly used external model representations can be derived from Eq. (12). They are the dynamic stiffness matrix [11] and the transfer function representation [12]. The dynamic stiffness representation

$$\mathbf{K}(s)\bar{\mathbf{Y}}(s)=\bar{\mathbf{U}}(s) \tag{15}$$

uses the $(r\times r)$ matrix

$$\mathbf{K}(s)=[\mathbf{M}(s)]^{-1}\mathbf{N}(s) \tag{16}$$

This first external representation is the format used in structural analysis [5] and finite element analysis. The dynamic stiffness matrix $\mathbf{K}(s)$ exists only if $\mathbf{M}(s)$ is nonsingular. If $\mathbf{M}(s)$ is nonsingular, the effects of the port inputs in the equations are independent. Port-based models formulated correctly always include the independent port energy inputs required to make $\mathbf{M}(s)$ nonsingular. Representation (15) exists even if $\mathbf{N}(s)$ is singular, allowing models where the effects of the port outputs in the equations are dependent. Although Eq. (15) is convenient for assembling models, it is not appropriate for simulations [3]. A different format is needed for simulation.

The transfer function representation

$$\bar{\mathbf{Y}}(s)=\mathbf{G}(s)\bar{\mathbf{U}}(s) \tag{17}$$

uses the matrix

$$\mathbf{G}(s)=[\mathbf{N}(s)]^{-1}\mathbf{M}(s) \tag{18}$$

This second external representation is the format used in system dynamics analysis such as simulation. The transfer function $\mathbf{G}(s)$ exists only if $\mathbf{N}(s)$ is nonsingular. If $\mathbf{N}(s)$ is nonsingular, the effects of the port outputs in the equations are independent. The transfer function model (17) exists even if $\mathbf{M}(s)$ is singular, allowing models where the effects of the port inputs in the equations are dependent. The transfer function model in Eq. (17) is appropriate for simulation but not for MMAM assembly [3].

5 Assembly Through Physical Model Constraints

The MMAM assembly of physical system models is characterized by two concepts. The first concept states that when physical systems are connected, their physical responses (outputs) at assembly points are equal. The second concept states that all physical systems' connections must satisfy conservation of energy. Physical system assembly requires both properties: equal system output response and energy conservation.

The connection of physical subsystem models is executed by joining ports that exchange energy. The external input-output variable pair that constitutes each port [7] is defined by the MMAM using the two concepts that characterize assembly of physical systems [3]. The MMAM defines the *output port variable* as the port

Table 1 Standard output and input variables for various energy domains

Energy domain	Output (y)	Input (u)
Electrical	Potential	Displacement (charge)
Mechanical translation	Displacement	Force
Mechanical rotation	Angle	Torque
Hydraulic	Pressure	Volume
Acoustic	Sound pressure	Volume
Heat transfer	Temperature	Heat

variable physically constrained to be equal to other outputs when ports are connected. The MMAM defines the *input port variable* as the variable required to compute units of energy when it is multiplied by the respective port output. Furthermore, connecting physical model ports requires all connected ports to be in the same energy domain. Energy standardization of the input-output variables for published models is a strict requirement for the MMAM that allows standard, efficient, recursive assembly of models in that standard format. The MMAM is recursive because the assembly models are in the same format as the component models used to produce them. Assembly models are immediately ready to be reused as subassembly models for higher-level assemblies.

The MMAM uses a strict standard to select input and output model variables. In general, from the modeling perspective, the choice of inputs and outputs depends on the model user's needs for both model creation and model analysis. The MMAM only specifies a standard format for model assembly and distribution (Fig. 1) and not for model creation and analysis. As will be apparent below, this standard format for distribution and assembly is easily derived from common model creation and analysis formats and causal structure. From an energy-based model assembly perspective, the choice of inputs and outputs is based on the two physical system's assembly properties [3]. MMAM *standard output variables* are selected as the variables physically constrained to be equal when two or more physical systems are assembled. MMAM *standard input variables* are selected as complementary variables that obey a summation property associated with energy conservation. This input-output standard is required for a standard MMAM energy-based model assembly algorithm. The rewards for this standard format include a standard, efficient, recursive assembly of distributed models, a single model representation, and a solution of the operating condition problem for assemblies of many affine components.

The MMAM standard input and output pairs for different energy domains are shown in Table 1. When ports are assembled, the MMAM outputs measured with respect to the assembly reference are equal at any defined port connection. The ports assembled in the electric domain require equal potentials. The ports assembled in the mechanical domain require either equal angular or linear displacements. The ports assembled in the hydraulic or acoustic domain require equal pressure. The ports assembled in the heat transfer domain require equal temperatures. In general, confusion can emerge if the MMAM standard is evaluated with example connections including only two ports. The MMAM input-output standard is most clear when examples of connections including three or more ports are analyzed. This feature is exploited in the examples provided below.

The MMAM input-output standardization might be thought to reduce the modeling generality, however, this is not strictly true. Using the Laplace transform to provide a frequency domain representation of any system's model allows the easy reordering of external input-output variables. The MMAM is motivated by the need for global model distribution and the input-output standardization allows for a single representation of an assembly that can be assembled to any other standard model. The input-output standardization allows the physical assembly constraints required to attach models to be applied without the need for matrix inverses.

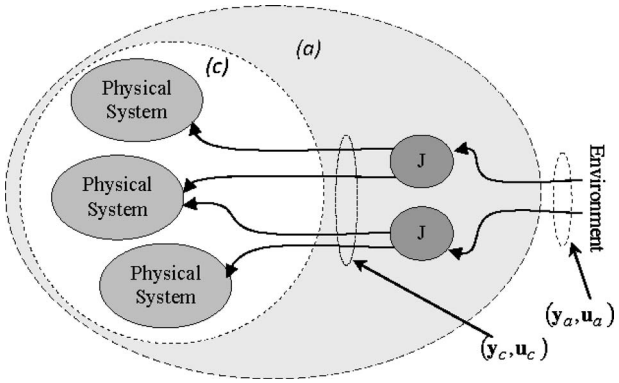

Fig. 6 Unconstrained and assembly control volumes

The MMAM's unique input-output model format facilitates assembly recursion and prevents the need for model reformulation. A standard, positive energy into a model port, sign convention independent of the element modeled is also required for assembly based simply on the connection topology and not on the internal function of the component modeled. Any element: a resistor or a battery, a source or a sink, or a suspended weight or a spring, has the same energy flow sign convention and input-output definitions allowing a standard model assembly procedure. Input-output standardization facilitates a successful global engineering model assembly method.

Joins are used at port connections to enforce physical assembly constraints on output variables. A join is not a model of a physical connection subsystem and does not store nor dissipate energy. Joins are mathematical mechanisms that provide the proper physical connection constraints for connecting the ports of physical components within a single energy domain [13]. A *join* is graphically represented here as a circle enclosing the letter J (Fig. 6). Lines represent connected ports with the direction of positive energy flow indicated by the arrowheads.

Two kinds of control volumes (Fig. 6) are defined in any assembly. The component control volume (c) interchanges energy with the assembly control volume (a) through the component port variables $\mathbf{y}_c(t)$ and $\mathbf{u}_c(t)$. The assembly control volume interchanges energy with the environment through the assembly port variables $\mathbf{y}_a(t)$ and $\mathbf{u}_a(t)$. If a set of components with r ports uses f joins with p constraints, the assembly has $l=r+f-p$ ports. In a practical assembly, $f \leq p$ and $l \leq r$. The dimension of the component and assembly vectors is, respectively, $(r \times 1)$ and $(l \times 1)$.

The two concepts that characterize a physical system assembly can be represented using two equations. The first equation constrains connected outputs at each join to be equal. For a join connecting d outputs

$$y_1(t) = y_2(t) = \ \ldots\ = y_i(t) = \ \ldots\ = y_d(t) = y_a(t) \tag{19}$$

where $\forall(i=1,\ldots,d)$, $y_i(t)$ is the ith connected output of the join, and $y_a(t)$ is the assembly output of the join. For a set of joins that join l assembly variables and a total of r component ports, Eq. (19) can be written in matrix form as

$$\mathbf{y}_c(t) = \mathbf{S}\mathbf{y}_a(t) \tag{20}$$

where $\mathbf{S}$ is the $(r \times l)$ constraint matrix. The second concept requires the conservation of energy at all joins. This requirement is met by the constraint equation

$$\mathbf{u}_c^T(t)\mathbf{y}_c(t) = \mathbf{u}_a^T(t)\mathbf{y}_a(t) \tag{21}$$

Equation (21) is a representation of the first law of thermodynamics for systems that do not store energy. Since joins do not store energy, the energy entering a join is equal to the energy leaving the join. The total energy supplied to the component control volume equals the total energy supplied to the assembly con-

trol volume. A second energy conservation constraint equation that relates component and assembly inputs can be derived from Eqs. (20) and (21)

$$\mathbf{S}^T\mathbf{u}_c(t) = \mathbf{u}_a(t) \tag{22}$$

The constraint equations used in the MMAM process are obtained by applying the Laplace transform to Eqs. (20) and (22)

$$\mathbf{Y}_c(s) = \mathbf{S}\mathbf{Y}_a(s) \tag{23a}$$

$$\mathbf{S}^T\mathbf{U}_c(s) = \mathbf{U}_a(s) \tag{23b}$$

Affine component models are defined at an operating condition at component outputs $\mathbf{y}_{c,o}$ and inputs $\mathbf{u}_{c,o}$. The resultant assembly affine model is defined at a corresponding operating condition at assembly outputs $\mathbf{y}_{a,o}$ and inputs $\mathbf{u}_{a,o}$. Since component operating point vectors $\mathbf{y}_{c,o}$ and $\mathbf{u}_{c,o}$ are conditions for the component variables $\mathbf{y}_c(t)$ and $\mathbf{u}_c(t)$, and the assembly operating point vectors $\mathbf{y}_{a,o}$ and $\mathbf{u}_{a,o}$ are conditions for the assembly variables $\mathbf{y}_a(t)$ and $\mathbf{u}_a(t)$, Eqs. (23*a*) and (23*b*) are also satisfied by the component and assembly operating point variables

$$\mathbf{Y}_{c,o}(s) = \mathbf{S}\mathbf{Y}_{a,o}(s) \tag{24a}$$

$$\mathbf{S}^T\mathbf{U}_{c,o}(s) = \mathbf{U}_{a,o}(s) \tag{24b}$$

where $\mathbf{Y}_{c,o}(s)$ and $\mathbf{U}_{c,o}(s)$ are $(r\times 1)$ vectors of Laplace transformed operating conditions for all components and $\mathbf{Y}_{a,o}(s)$ and $\mathbf{U}_{a,o}(s)$ are $(l\times 1)$ vectors of Laplace transformed assembly operating conditions. Equation (24*a*) computes the operating condition $\mathbf{Y}_{c,o}(s)$ of components given the operating condition $\mathbf{Y}_{a,o}(s)$ of their assembly. Equation (24*b*) computes the operating condition $\mathbf{U}_{a,o}(s)$ of the assembly given operating condition $\mathbf{U}_{c,o}(s)$ of its components. Both computations are required in the query-response model distribution format (Fig. 5) and use the procedure below to automate computational model assembly.

The assembly agent knows the assembly constraints $\mathbf{S}$ and uses Eq. (24*a*) to compute the operating point outputs $\mathbf{Y}_{c,o}(s)$ for all the assembly's components given the assembly operating condition outputs $\mathbf{Y}_{a,o}(s)$ specified by the user's query to the assembly agent. These component outputs are sent to component agents as part of a query to each component agent for a model. At every level, each component agent responds to a model request with that component's linearized model and that component's operating point input $\mathbf{U}_{c,o}(s)$. The assembly agent then determines the assembly's operating point input $\mathbf{U}_{a,o}(s)$ using the matrix $\mathbf{S}$ and Eq. (24*b*).

Some assembly operating conditions could be unknown for the client. Nonspecified assembly operating condition outputs are considered internal output variables constituting internal ports. Even though these internal operating outputs are initially unknown, component agents can determine them from component models by using the fact that its corresponding internal operating port inputs are zero.

6 Assembly of Affine Physical Models

The MMAM internet agents use a standard process to assemble dynamic stiffness matrix-based models (15). These models have port pairs standardized through the two concepts that characterize physical system assembly: output constraints and energy conservation. The assembly process for affine models includes two steps. These steps are (1) the generation of an unconstrained, unassembled collection of standard dynamic stiffness component models and (2) the application of assembly constraints to the collected component model equations to form an assembly model in the standard dynamic stiffness format.

The unconstrained collection of standard dynamic component models is formulated as an unconstrained matrix of dynamic component models (15) in diagonal form. The collection of k component models with a total number of r collected ports yields

$$\mathbf{K}_c(s) = \begin{bmatrix} \mathbf{K}_1(s) & 0 & \ddots \\ 0 & \ddots & 0 \\ \ddots & 0 & \mathbf{K}_k(s) \end{bmatrix} \tag{25}$$

where $\mathbf{K}_i(s)$ is the $(r_i\times r_i)$ dynamic matrix of the ith component and $\mathbf{K}_c(s)$ is the $(r\times r)$ collected dynamic matrix from the k components, where $r=\Sigma_{i=1}^{k} r_i$. The unconstrained matrix of the collected component models relates the component deviation output vector $\bar{\mathbf{Y}}_c(s)$ to the deviation component input vector $\bar{\mathbf{U}}_c(s)$ in the form

$$\mathbf{K}_c(s)\bar{\mathbf{Y}}_c(s) = \bar{\mathbf{U}}_c(s) \tag{26}$$

The required equations to transform deviation variables into physical variables are also available.

$$\bar{\mathbf{Y}}_c(s) = \mathbf{Y}_c(s) - \mathbf{Y}_{c,o}(s) \tag{27a}$$

$$\bar{\mathbf{U}}_c(s) = \mathbf{U}_c(s) - \mathbf{U}_{c,o}(s) \tag{27b}$$

where $\mathbf{Y}_{c,o}(s)$ and $\mathbf{U}_{c,o}(s)$ are the transformed operating condition outputs and inputs in the Laplace domain. The values of each of the component's dynamic stiffness matrix and operating condition outputs and inputs are obtained through the query-response process (Fig. 5).

Application of assembly constraints by the assembly agent proceeds using the constraint matrix $\mathbf{S}$ that relates component variables and assembly variables. This matrix defines constraints on port output connection (Eq. (24*a*)) and energy conservation (Eq. (24*b*)). The process is presented both below and in the next section by an example for an affine assembly. The process is a direct extension to the MMAM of linear systems [3]. Initially, Eqs. (27*a*) and (27*b*) are substituted into Eq. (26)

$$\mathbf{K}_c(s)[\mathbf{Y}_c(s) - \mathbf{Y}_{c,o}(s)] = [\mathbf{U}_c(s) - \mathbf{U}_{c,o}(s)] \tag{28}$$

Multiplying both sides of Eq. (28) by $\mathbf{S}^T$ and then substituting Eqs. (23*a*), (23*b*), (24*a*), and (24*b*) into Eq. (28) and simplifying yields

$$\mathbf{S}^T\mathbf{K}_c(s)[\mathbf{S}\mathbf{Y}_a(s) - \mathbf{S}\mathbf{Y}_{a,o}(s)] = [\mathbf{U}_a(s) - \mathbf{U}_{a,o}(s)] \tag{29}$$

where the assembly input operating condition $\mathbf{U}_{a,o}(s)$ is computed by the MMAM internet agent from the component operating condition $\mathbf{U}_{c,o}(s)$ provided to it by each component. Factoring matrix $\mathbf{S}$ in the right side and rewriting Eq. (29)

$$\mathbf{K}_a(s)[\mathbf{Y}_a(s) - \mathbf{Y}_{a,o}(s)] = [\mathbf{U}_a(s) - \mathbf{U}_{a,o}(s)] \tag{30}$$

where the $(l\times l)$ assembly dynamic stiffness matrix is

$$\mathbf{K}_a(s) = \mathbf{S}^T\mathbf{K}_c(s)\mathbf{S} \tag{31}$$

Defining a new set of $(l\times 1)$ deviation variable vectors

$$\bar{\mathbf{Y}}_a(s) = [\mathbf{Y}_a(s) - \mathbf{Y}_{a,o}(s)] \tag{32a}$$

$$\bar{\mathbf{U}}_a(s) = [\mathbf{U}_a(s) - \mathbf{U}_{a,o}(s)] \tag{32b}$$

and, finally, substituting Eqs. (32*a*) and (32*b*) into Eq. (30) yields the assembly model in deviation variables

$$[\mathbf{K}_a(s)]\bar{\mathbf{Y}}_a(s) = \bar{\mathbf{U}}_a(s) \tag{33}$$

The MMAM described above is recursive. A group of component models, each model in a specific standard format, is assembled using constraints to form an assembly model in the same standard format. The assembly model (33) is in the MMAM standard model format (26). The assembly model (33) can now be assembled as a component to other similar standard component models to form a higher-level assembly using the same MMAM algorithm.

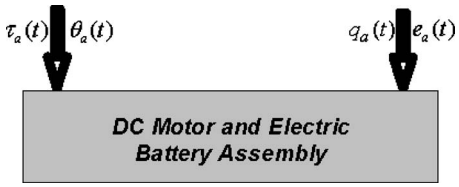

Fig. 7 External view of a dc motor and an electric battery assembly requested by a user from an assembly agent

7 Affine Model Assembly Example

The internet distributed model assembly process for the assembly of a MMAM model of an electric motor and electric battery assembly (Fig. 7) is illustrated here. The user of such a model would request it from an assembly agent. The assembled model provided is external, includes two external ports and hides the internal port connection topology. The assembly agent has the internal topology (Fig. 8) of the motor-battery assembly as well as the addresses of component agents that will provide external models of the components. Internal proprietary design details (shaded in Fig. 8) including the model structure would not be available in the final assembly model provided to the user.

Assume that both a dc permanent magnet electric motor model and an affine battery model are available to the motor-battery assembly agent from other agents on the internet. Both models use numerical parameters because they are models of specific physical components. Both models use MMAM energy variables with a positive energy flow sign convention for all model ports. Queries by the motor-battery assembly agent to the two motor and battery model agents reveal that a linear electric motor with two ports and an affine electric battery model with one port are available. The electric motor's first port is electrical with port variables input electrical charge $q_m(t)$ and output applied potential $e_m(t)$. The electric motor's second port is mechanical rotation with port variables input rotational torque $\tau_m(t)$ and output angular displacement $\theta_m(t)$. The battery model's port variables are the input electrical charge $q_b(t)$ and the battery output potential $e_b(t)$. Both component models are affine in their physical coordinates. The assembly of the affine component models will yield an affine result about the operating condition requested by the user.

A request for information to the assembly agent would inform the user that the model available was affine and defined about an operating point specified at operating point values for the first derivative of motor angular displacement $\omega_{a,o}$ and drive potential $e_{a,o}$. The assembly agent would typically provide further general information such as the 48 V nominal battery voltage of the assembly and the 125 A maximum continuous current allowed. For a model operating point, the user decides to use an assembly potential $e_{a,o}=48$ V, the nominal battery voltage, along with an assembly speed of 3400 rpm yielding $\omega_{a,o}=2\pi\times3400/60=356$ rad/s. These two output conditions are specified to the assembly agent with

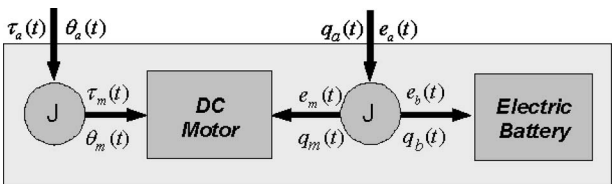

Fig. 8 Internal details of a dc motor and an electric battery assembly as assembled by the assembly agent

$$[\Theta_{a,o}(s) \quad E_{a,o}(s)]^T = [356/s^2 \quad 48/s]^T \tag{34}$$

where $\Theta_{a,o}(s)$ is the angular displacement operating condition for the assembly and $E_{a,o}(s)$ is the drive potential condition for the assembly both defined in the frequency domain.

Using the topology of the motor-battery assembly shaded in Fig. 7, the assembly agent uses the assembly constraints

$$\Theta_m(s) = \Theta_a(s)$$

$$E_m(s) = E_b(s) = E_a(s) \tag{35}$$

to assemble the constraint equations

$$\begin{bmatrix} E_m(s) \\ \Theta_m(s) \\ E_b(s) \end{bmatrix} = \mathbf{S}\begin{bmatrix} E_a(s) \\ \Theta_a(s) \end{bmatrix} \tag{36}$$

where

$$\mathbf{S} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

The assembly agent uses the component assembly constraints (36) in Eq. (24*a*) to compute the operating condition output values for all components

$$\begin{bmatrix} E_{m,o}(s) \\ \Theta_{m,o}(s) \\ E_{b,o}(s) \end{bmatrix} = \mathbf{S}\begin{bmatrix} E_{a,o}(s) \\ \Theta_{a,o}(s) \end{bmatrix} = \begin{bmatrix} E_{a,o}(s) \\ \Theta_{m,o}(s) \\ E_{a,o}(s) \end{bmatrix} = \begin{bmatrix} e_{m,o}/s \\ \omega_{m,o}/s^2 \\ e_{b,o}/s \end{bmatrix} = \begin{bmatrix} 48.0/s \\ 356/s^2 \\ 48.0/s \end{bmatrix} \tag{37}$$

These components' operating point outputs are then used by the assembly agent in a model query (Fig. 5) to request from each of the component agents its respective component model.

The MMAM affine dc electric motor model is available to the assembly agent from a component agent for a MARS ME708 dc motor [14]. A request to the motor model agent for general information reveals that this motor is a 48 V motor with a maximum current rating of 125 A and weighs 30 lbs (13.6 kg). The assembly agent queries the component agent for a motor model at the motor output operating conditions

$$\begin{bmatrix} E_{m,o}(s) \\ \Theta_{m,o}(s) \end{bmatrix} = \begin{bmatrix} 48.0/s \\ 356/s^2 \end{bmatrix} \tag{38}$$

Using Eq. (38), the motor agent responds to the query with both a linear operating point model and the operating condition inputs. At the specified operating condition outputs, the motor model dynamic stiffness is

$$\mathbf{K}_m(s) = \begin{bmatrix} \left(\dfrac{35.752}{s}\right) & (-4.539) \\ (-4.539) & (0.589s^2 + 0.578s) \end{bmatrix} \tag{39}$$

As shown above, an internet model distribution for a real product would use numerical values, not internal parameters, to protect internal proprietary design details. The dynamic stiffness model (39) is a specification of motor dynamic performance that does not reveal the proprietary internal geometrical, material, or manufacturing design details used to produce the motor. When supplied with operating condition output values, the agent also returns the corresponding motor operating condition inputs

$$\begin{bmatrix} Q_{m,o}(s) \\ \mathrm{T}_{m,o}(s) \end{bmatrix} = \begin{bmatrix} 100/s^2 \\ -11.7/s \end{bmatrix} \tag{40}$$

The motor output deviation $\bar{\mathbf{Y}}_{\text{motor}}(s)=[\bar{E}_m(s) \;\; \bar{\Theta}_m(s)]^T$ and input deviation $\bar{\mathbf{U}}_{\text{motor}}(s)=[\bar{Q}_m(s) \;\; \bar{\mathrm{T}}_m(s)]^T$, where $\bar{E}_m(s)$ is the Laplace transform of the output potential deviation of the motor $\bar{e}_m(t)$ in volts, $\bar{Q}_m(s)$ is the Laplace transform of the input electri-

cal displacement deviation $\bar{q}_m(t)$ in coulombs, $\bar{\mathrm{T}}_m(s)$ is the Laplace transform of the input rotational torque deviation $\bar{\tau}_m(t)$ in N m, and $\bar{\Theta}_m(s)$ is the Laplace transform of the output angular displacement deviation $\bar{\theta}_m(t)$ in radians.

The resulting input operating condition for the motor are the following.

(1) Electrical displacement ramp $q_{m,o}=(100t)h(t)$ yielding an operating current $i_{m,o}=(100)h(t)$ A.
(2) Mechanical torque $\tau_{m,o}=(-11.7)h(t)$ N m with a negative value indicating net energy flows out, where $h(t)$ is the Heaviside step function. The motor model agent returns the model (Eq. (39)) and the operating condition input (Eq. (40)) to the assembly agent.

The MMAM battery model available is for a 48 V sealed lead-acid battery pack with manufacturer specifications of a current rating of 870 cold amps (CA), and a storage capacity of 55 A h. When the battery model is requested at a constant operating condition of $E_{b,o}(s)=48.0/s$ V, the battery agent's affine model of the battery returned is

$$\left[\frac{25}{s}\right]\bar{E}_b(s)=\bar{Q}_b(s) \tag{41}$$

The battery's dynamic stiffness model (41) is a specification of battery dynamic performance, where the battery's electrical potential output deviation $\bar{E}_b(s)=E_b(s)-(48.0/s)$ and electrical displacement input deviation $\bar{Q}_b(s)=Q_b(s)-Q_{b,o}(s)$. The battery's dynamic performance model (41) does not reveal the proprietary internal geometrical, material, or manufacturing design details used to produce the battery. When supplied with operating condition output values, the battery agent also returns the corresponding motor operating condition inputs. At the requested electrical potential output operating condition $E_{b,o}(s)=48.0/s$, the battery model agent returns the required battery electrical displacement input

$$Q_{b,o}(s)=\frac{-140}{s^2} \tag{42}$$

indicating an energy output from the battery port with a battery port electrical displacement input ramp $q_{b,o}=(-100t)h(t)$ (coulombs) yielding a constant input operating current $i_{b,o}=(-100)h(t)$ A out of the battery.

The assembly agent uses the models (39) and (41) and the operating point inputs (40) and (42) supplied by component agents to assemble both an assembly model and an assembly external input vector. The assembly agent first forms an unconstrained collection of the motor and battery component models

$$\begin{bmatrix}(35.752/s) & (-4.539) & 0\\ (-4.539) & (0.589s^2+0.578s) & 0\\ 0 & 0 & (25/s)\end{bmatrix}\begin{bmatrix}\bar{E}_m(s)\\ \bar{\Theta}_m(s)\\ \bar{E}_b(s)\end{bmatrix}=\begin{bmatrix}\bar{Q}_m(s)\\ \bar{\mathrm{T}}_m(s)\\ \bar{Q}_b(s)\end{bmatrix} \tag{43}$$

where the component deviation variables are defined above. The battery-motor assembly agent then computes the assembly model using the definition of the assembly dynamic matrix (31), the assembly constraint (36), and unconstrained dynamic stiffness model (43).

$$\begin{bmatrix}\left(\frac{60.752}{s}\right) & (-4.539)\\ (-4.539) & (0.589s^2+0.578s)\end{bmatrix}\begin{bmatrix}\bar{E}_a(s)\\ \bar{\Theta}_a(s)\end{bmatrix}=\begin{bmatrix}\bar{Q}_a(s)\\ \bar{\mathrm{T}}_a(s)\end{bmatrix} \tag{44}$$

The battery-motor dynamic stiffness model (44) is a specification of battery dynamic performance using specific numerical parameter values. Because the user is only supplied with an external battery-motor assembly model (Fig. 8) and not with the topology of the assembly (Fig. 8), the battery-motor dynamic performance model (44) does not reveal the proprietary internal geometrical, material, or manufacturing design details used to produce the battery. The battery-motor assembly external model inputs are computed using Eq. (24*b*) and the constraint matrix in Eq. (36) on the component internal inputs (40) and (42)

$$\begin{bmatrix}Q_{a,o}(s)\\ \mathrm{T}_{a,o}(s)\end{bmatrix}=\begin{bmatrix}\left(\frac{100-140}{s^2}\right)\\ \left(\frac{-11.7}{s}\right)\end{bmatrix}=\begin{bmatrix}\left(\frac{-40}{s^2}\right)\\ \left(\frac{-11.7}{s}\right)\end{bmatrix}\quad \text{for } t>0 \tag{45}$$

The assembly agent now returns to the user both the assembly model (44) and the model's required input at the specified operating condition (45). Correct model and operating condition inputs are computed above for the specified operating condition outputs. Clearly, the model operating conditions computed require external current $sQ_{a,o}(s)=-40$ A to be drawn from the assembly. One possible issue is that the model includes an open external electrical port requiring a nonzero external electrical displacement input at the user specified operating condition outputs.

A procedure similar to the condensation procedure in Ref. [3] allows the computation of model operating condition outputs requiring zero electrical displacement $Q_{a,o}(s)=0$ through the removal of the electrical port on the assembly. This procedure can be performed either exactly upon request to the assembly or approximately by the user using the affine model and the required operating condition inputs returned by the assembly agent. If a user desires a model for zero external electrical displacement input $Q_{a,o}(s)=0$, a new value for the operating condition $E_{a,o}(s)$ is required.

Substituting the operating conditions $Q_{a,o}(s)=-40/s^2, E_{a,o}(s)=48/s$ into the corresponding model supplied by the assembly agent Eq. (44) yields

$$\begin{bmatrix}\left(\frac{60.752}{s}\right) & (-4.539)\\ (-4.539) & (0.589s^2+0.578s)\end{bmatrix}\begin{bmatrix}E_a(s)-(48/s)\\ \bar{\Theta}_a(s)\end{bmatrix}=\begin{bmatrix}Q_a(s)+(40/s^2)\\ \bar{\mathrm{T}}_a(s)\end{bmatrix} \tag{46}$$

At the operating point $\bar{\Theta}_a(s)=0$, $\bar{\mathrm{T}}_a(s)=0$, the user computes the value of $E_a(s)$ for $Q_a(s)=0$. The upper equation of Eq. (46) can be solved for the physical assembly potential

$$E_a(s)=\left(\frac{s}{60.752}\right)\left[Q_{a,o}(s)+\frac{40}{s^2}\right]+\frac{48}{s}=\frac{48.6584}{s} \tag{47}$$

The potential $E_a(s)$ from Eq. (47) defines a new operating point, where $Q_{a,o}(s)=0$ in the affine model (46). A query to the assembly with the assembly operating point output specified as

$$\begin{bmatrix}E_{a,o}(s)\\ \Theta_{a,o}(s)\end{bmatrix}=\begin{bmatrix}48.6584/s\\ 356/s^2\end{bmatrix} \tag{48}$$

will return the same affine model (44) and the new required operating point inputs

$$\begin{bmatrix}Q_{a,o}(s)\\ \mathrm{T}_{a,o}(s)\end{bmatrix}=\begin{bmatrix}0/s^2\\ -14.69/s\end{bmatrix}\quad \text{for } t>0 \tag{49}$$

The solution is exact because the assembly and both component models are affine. Their linear deviation models are exact globally with any change of coordinates. An assembly model derived from local linearizations would not generate an exact result but successive iterative solutions should converge locally. The general prob-

lem of local linearization and condensation for nonlinear models is a subject of current work for the authors.

The example above demonstrates that the MMAM is a modular, single-query, external, and recursive model assembly and distribution procedure that protects proprietary design details. It is modular because it uses a unique standard format for component and assembly models. It is single-query because it retrieves a model with a request to any model agent. The MMAM is external because the models received are expressed as functions of external port variables. The MMAM is recursive because the model representation for any assembly is identical to those of its components allowing recursive assembly of higher-level models.

The MMAM protects internal proprietary design information because an assembly model cannot be used to uniquely determine the models of its components. Specifically, there are an infinite number of motor and battery parameter combinations that would produce the motor-battery model result (44). Even in the simple example above, the assembly model parameter "60.752" is a function of separate internal battery and motor parameters. Given the external performance model (46), the specific values of the internal motor and battery parameters cannot be determined. Only the performance resulting from their combined performance is available. The proportional relationship between assembly potential deviation $\bar{E}_a(s)$ and assembly electrical displacement deviation $\bar{Q}_a(s)$ is a measurable external property of the assembly that can be achieved through an infinite number of combinations of motor and battery properties. Reverse engineering can produce another design with equivalent external performance; however, no testing method can deduce specific measures of the particular component properties that produced that external assembly performance. As assembly model complexity increases, the advantages and protection provided by MMAM increases due to the simple, recursive, matrix-based assembly and external model representation.

8 Conclusions

In this work, a method for producing a reusable, standardized, nonlinear affine model of an assembly through networked queries to agents providing models of the assembly's components has been presented. The affine assembly models produced are recursively reusable because they are in the same format as the models of their components and can be used as subassemblies of higher-level assemblies. Affine systems are important because the affine models of physical systems are common and the affine approximate linearization models of general differentiable nonlinear components around a constant operating condition yield a local affine approximate model of the nonlinear assembly response. In the proposed networked environment, standard component models are distributed as linearized models defined in deviation variables but are assembled as nonlinear affine models defined in physical variables.

The MMAM networked distribution and assembly process for affine models uses a single query-response system to retrieve an external model to protect proprietary information while reducing network traffic. Previous work [2] has used co-simulation to protect proprietary design details requiring many network communications per simulation for model evaluations. In contrast, the MMAM requires one query-response network communication per component model and allows later simulation without further network communication. In MMAM queries, clients request models from networked agents by providing the desired operating condition port output values for the model. In the response, each agent model server returns a standard format dynamic model in deviation variables and the component's port input values required at the desired operating condition. The operating condition outputs specified and required inputs returned provide the transformations between deviation variables and physical variables. The single query-response communication per model reduces network traffic for model retrieval and simulation from co-simulation's requirement for multiple communications per simulation time step.

The MMAM assembly process is executed by using two assembly constraint equations. The port output constraint states that MMAM outputs must be equal at each connection of the assembly. The port input constraint provides a complementary energy conservation constraint at each connection of the assembly. This model assembly process is recursive because it uses the same algorithm to assemble simple component models or models of assemblies of components. In addition, the MMAM protects proprietary information because it uses external input-output models at every level. The model of an assembly reveals only the external performance of the assembly, not the performance of its individual components nor the component connection topology. This represents a significant advancement in global engineering because it eliminates the need for legal agreements that protect proprietary information.

An important contribution of this paper is the application of the MMAM to solve the general operating point problem for assemblies of locally linearized component models. That operating condition problem solution is independent of the model type: linear, affine, or local linearization. The MMAM operating condition solution uses only the assembly topology information through the constraint matrix $\mathbf{S}$ and not the form of the model being assembled. The operating condition problem solution removes the need for operating condition analysis at any assembly level and only requires the operating point solutions at the level of the lowest simplest component of that assembly by agents providing models of those components. Once component model operating conditions are solved by their remote agents, a closed form solution to the operating condition problem is executed recursively through each subassembly level until the operating condition at the highest level of the assembly model is determined.

Future work will include three important modeling aspects of the MMAM. The first aspect is the recursive construction of linearization error estimates for an assembly from error estimates provided by that assembly's component agents. Second, a condensation method for affine assembled models to remove external ports and their input variables will be developed similar to the condensation process for linear MMAM [3]. Future publications will also discuss MMAM external representations and assembly methods for more general nonlinear system models.

References

[1] Tian, G. Y., Yin, G., and Taylor, D., 2002, "Internet-Based Manufacturing: A Review and a New Infrastructure for Distributed Intelligent Manufacturing," J. Intell. Manuf., **13**, pp. 323–338.

[2] Gu, B., and Asada, H., 2004, "Co-Simulation of Algebraic Coupled Dynamic Systems Without Disclosure of Proprietary Subsystem Models," ASME J. Dyn. Syst., Meas., Control, **126**, pp. 1–13.

[3] Radcliffe, C. J., Motato, E., and Reichenbach, D., 2009, "Networked Assembly of Mechatronic Linear Physical System Models," ASME J. Dyn. Syst., Meas., Control, **131**, p. 021003.

[4] Elmqvist, H., 1999, "Modelica: A Language for Physical System Modeling, Visualization and Interaction," IEEE Symposium on Computer-Aided Control System Design, CACSD '99, Hawaii, pp. 22–27.

[5] Zienkiewicz, O., 1989, *The Finite Element Method*, McGraw-Hill, London.

[6] Mattsson, E., 1993, "Index Reduction in Differential Algebraic Equations Using Dummy Derivatives," SIAM J. Sci. Comput. (USA), **14**(3), pp. 677–692.

[7] Karnopp, D., Margolis, D., and Rosenberg, R., 2000, *System Dynamics: Modeling and Simulation of Mechatronic Systems*, Wiley, New York.

[8] Willems, J. C., 2007, "The Behavioral Approach to Open and Interconnected Systems," IEEE Control Syst. Mag., **27**, pp. 46–99.

[9] Buck, R., and Willcox, A., 1971, *Calculus of Several Variables*, Houghton Mifflin, Boston.

[10] Giret, A., and Botti, V., 2004, "Holons and Agents," J. Intell. Manuf., **15**, pp. 645–659.

[11] Genta, G., 1999, *Vibration of Structures and Machines*, Springer-Verlag, New York.

[12] Skogestad, S., and Postlethwaite, I., 2000, *Multivariable Feedback Control*, Wiley, England.

[13] Byam, B., 1999, "Modular Modeling of Engineering Systems Using Fixed I-O Structure," Ph.D. thesis, MSU, East Lansing, MI.

[14] http://www.marselectricllc.com/me07081.html