A User-Steered Energy Generation and Consumption Multi-Model Simulation for Pricing and Policy Development

Harrison B. Smith, *Student Member, IEEE,* Amy Pielow, Adithya Jayakumar, Matteo Muratori, *Student Member, IEEE,* B. J. Yurkovich, Ramteen Sioshansi, *Senior Member, IEEE,* Ashok Krishnamurthy, *Member, IEEE,* Giorgio Rizzoni, *Fellow, IEEE,* and Matthew C. Roberts

***Abstract*—Understanding energy use is critical. While simulation is valuable, such models are simplified abstractions of actual energy systems. We present an energy system multi-model implemented with the newly developed LAPIS computational steering API. We present an adaptable framework for the integration and development of multi-model simulations. This framework has key advantages including allowing independent development of component simulations, limiting coordination overhead between developers, and allowing modularity and flexibility in the overall multi-model simulation. We use case studies to demonstrate the capabilities of the multi-model energy system simulation and LAPIS.**

***Index Terms*—Multi-model simulation, computational steering, energy system modeling, energy policy**

I. Introduction

ENERGY-related issues are increasingly important. Reliance on fossil fuels has significant environmental, geopolitical, energy supply, and macroeconomic effects. A number of technologies, including renewable energy sources, plug-in electric vehicles (PEVs), and efficient and smart appliances, are proposed to mitigate these issues. These are not panacean solutions, however, and their use can have unintended consequences. Moreover, most modern energy systems are not centrally planned. Rather, energy technologies are adopted and used by individuals, based on cost and other considerations. Thus, governments, policymakers, and others often rely on indirect policy measures to guide energy system development.

Understanding interactions between new and existing energy technologies, and policy impacts therein, is key to driving sustainable energy use and economic growth. This endeavor is easier said than done. With more complex technologies and greater resource constraints, understanding the complete energy generation, distribution, and consumption picture is daunting. Fully understanding the intricacies of how renewable energy sources, an aging energy infrastructure, increasing global energy demand, and PEVs interact is complex, involving multiple domains of expertise. This multifaceted problem lends itself to designing and implementing a large-scale, interactive simulation that allows users to gain insights into these topics to help inform their decision making.

By building a multidisciplinary team with expertise in engineering, computer science, and economics, we are developing such a large-scale simulation. This paper presents the energy system models and the computational solution used to integrate the multi-model simulation. Illustrative results, demonstrating the models' value in energy system simulation and policy analysis, are also presented. The tools and techniques being developed are applicable to a wide array of domains and computational tasks. Thus, we also summarize the performance of our computational system in steering large-scale multi-model simulations.

A. Integrated Computational System for Energy Pricing and Policy

A computational model, called the Integrated Computational System for Energy Pricing and Policy (ICS-EPP), is currently being developed at The Ohio State University. The purpose of the ICS-EPP (see Fig. 1) is to assist the formulation of energy policy, pricing, and investment decisions. The ICS-EPP includes interacting sub-models of: (i) individuals' behavior; (ii) sector- and time-resolved electricity demand; (iii) the electric power system with distributed and stochastic supplies; (iv) vehicle energy consumption; (v) long-term technology investments; and (vi) different technology options.

The ICS-EPP is intended to help users explore the space of possible policy options. The model simulates the effects of policy on energy use and technology adoption. This allows users to design a new energy system that meets their desired

This material is based upon work supported by the National Science Foundation under Grant No. 1029337.

H. Smith and A. Jayakumar are with The Ohio State University, Department of Electrical and Computer Engineering, 2015 Neil Ave., Columbus, OH 43210, Tel: +1-614-292-2572, Fax: +1-614-292-7596 (e-mail: smith.3738@osu.edu and jayakumar.5@osu.edu).

A. Pielow and R. Sioshansi are with The Ohio State University, Department of Integrated Systems Engineering, 1971 Neil Ave., Columbus, OH 43210, Tel: +1-614-292-9461, Fax: +1-614-292-7852 (e-mail: pielow.1@osu.edu and sioshansi.1@osu.edu).

M. Muratori, B. Yurkovich, and G. Rizzoni are with The Ohio State University, Center for Automotive Research, 930 Kinnear Road, Columbus, OH 43212, Tel: +1-614-292-5990, Fax: +1-614-688-4111 (e-mail: muratori.2@osu.edu, yurkovich.7@osu.edu, and rizzoni.1@osu.edu).

A. Krishnamurthy is with the Renaissance Computing Institute of the University of North Carolina at Chapel Hill, 100 Europa Dr., Suite 540, Chapel Hill, NC 27517, Tel: +1-919-445-9640, Fax: +1-919-445-9669 (e-mail: ashok@renci.org).

M. Roberts is with The Ohio State University, Department of Agricultural, Environmental and Development Economics, 2120 Fyffe Rd., Room 103, Columbus, OH 43210, Tel: +1-614-292-7911, Fax: +1-614-292-0078 (e-mail: roberts.628@osu.edu).

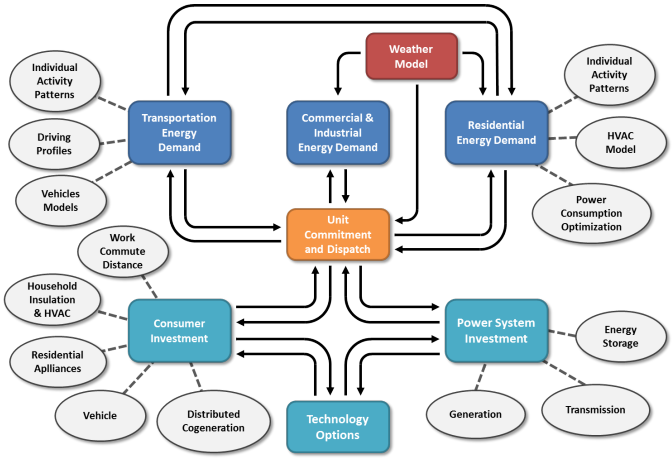

Fig. 1. The ICS-EPP multi-model simulation. Development of residential, commercial and industrial, and transportation energy demand and unit commitment and dispatch models is in progress. The remaining models have yet to be implemented and will be added at future stages of the project.

objectives (*e.g.*, reduced CO_2 emissions, greater renewable use, or minimized energy costs), while accounting for energy system dynamics and individuals' self-interested behavior. This requires access to a comprehensive model that allows the exploration of 'what-if?' scenarios over large regional areas and long time scales. Integration of the models comprising the ICS-EPP is accomplished using a computational steering platform developed at the Ohio Supercomputer Center (OSC), called the Language and Platform Independent Steering (LAPIS) system.

ICS-EPP has important differences from existing energy system models. The ICS-EPP is explicitly designed to model disparate decision making by interacting autonomous agents. This contrasts with the 'command-and-control' structure of existing models, which assume that all agents make socially optimal (*e.g.*, cost-minimizing) decisions, neglecting incentive or externality issues. The ICS-EPP is also agnostic to the overall objective, since the user steers the simulation based on his or her desired outcome. Many existing national energy models assume cost minimization. Finally, the use of LAPIS makes the ICS-EPP modular, allowing users to customize the simulation by plugging in purpose-built models. For example, if a user has a detailed power flow model, incorporation into the broader simulation is simple. This is because the LAPIS API allows codes using different languages and platforms to interact in a steered simulation. Although some existing models have a modular design, users are typically limited to the single language in which the underlying model is developed.

B. Computational Steering

Computational steering is defined as the interactive control over an executing simulation by a human or other agent in real-time [1]. Traditional scientific computation relies on a heavily structured, iterative process, or 'open loop' simulation. This methodology has numerous disadvantages with regard to efficiency, usefulness, and simulation methodology [1]–[3]. Computational steering addresses these inefficiencies by 'closing the loop,' allowing researchers to visualize and interact with simulations as they are running, thereby becoming an active simulation participant rather than just interpreting results (see Fig. 2, which illustrates how the researcher interacts with a traditional and steered simulations).

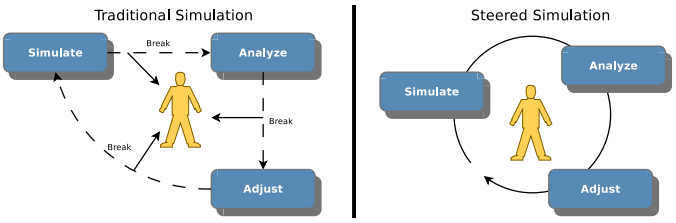

Fig. 2. Computational steering systems let the researcher act as a part of the simulation itself, allowing hard-to-capture domain knowledge to be applied directly to the problem at hand.

Steering also allows a number of simulation models that are otherwise unavailable, inefficient, or difficult to implement. Of particular note are model exploration, performance optimization, algorithm exploration, and multi-model systems. Model exploration systems allow the user to interaction with a simulation, algorithm, or model in real-time to gain increased depth of understanding, insight, and intuition [4]. One early example is the 3D turbulence model of Lake Erie developed at the OSC [5]. Later model exploration systems include the SCIRun programming environment [2]. Performance optimization steering systems manipulate the running simulation in ways that only affect its performance, not the simulation results. An example is the balancing of computational workload between processors in a distributed computation [6], [7]. Algorithm exploration allows users to experiment with computational methodologies within simulation code. Examples are limited in the literature, however LAPIS provides a mechanism for coarse-grain algorithmic steering. Finally, computational steering can be leveraged to create robust and dynamic multi-model simulations. Multi-model simulations are used successfully in many domains and a wide range of architectures [8]–[11], most notably in the field of environmental science. Nevertheless, few free or commercially available tools for their creation exist, and multi-model simulations are largely one-off development projects.

Both the required hardware and software tools exist to make a fully featured computational steering environment successful. Given steering's proven value, it is disappointing that it is not widely used. This lack of adoption is due to four shortcomings of existing steering packages [12]: development difficulty, portability issues, maintainability of code, and third-party hardware and software support. Existing steering tools are complicated to use, often requiring re-implementation of existing simulations [2], [13]. Such overhead is unacceptable to most researchers. Additionally, once a simulation is built or instrumented with a steering tool, it becomes linked to that steering technology. Many steering tools have considerable dependencies, which tie to the simulations that use them, creating maintainability and portability difficulties [13]–[15]. A lack of third-party support, in the realms of free tools and commercial options, leaves the steering developer to solve

complex technical issues unrelated to his or her primary field of research.

LAPIS is specifically designed to address these and other problems with steering. While developing the ICS-EPP simulation, we make use of some key methodologies discussed in detail below. These methodologies, in parallel with computational steering, results in a highly successful approach to simulation development and execution.

II. ICS-EPP Sub-Models

The models constituting the ICS-EPP multi-model simulation are developed independently, using a different set of languages, computational tools, and techniques.

A. Residential Energy Demand

Implemented in MATLAB, this model simulates the electricity consumption of a residential sector using a bottom-up approach [16]–[19]. Such energy demand is variable and depends on physical factors (*e.g.*, weather, temperature, and dwelling characteristics) and the household members' behavior. The total electric power demand of each dwelling is modeled as the sum of energy used by: (i) cold appliances (*e.g.*, refrigerators and freezers); (ii) heating, ventilation, and air conditioning (HVAC); (iii) the household members' activities; (iv) lighting; and (v) ubiquitous electric consumption (*i.e.*, lights that are always on and appliance stand-by power) [19]–[21]. The first three components are modeled using engineering physically-based models, while individuals' behaviors are modeled using a heterogeneous Markov chain. The model is calibrated and validated against metered electric load data provided by American Electric Power and behavioral data collected by the U.S. Bureau of Labor Statistics.

The model uses LAPIS to get weather data for HVAC simulation. Users can vary model parameters to explore the effects of different technologies. For instance, varying activity-related energy use represents different appliance efficiencies. These changes are communicated to the model during runtime using LAPIS. The technology investment model (under development) endogenously optimizes technology decisions, which are communicated to the residential energy demand model using LAPIS. The residential demand model outputs 10-minute resolved electric energy demand.

B. Commercial and Industrial Energy Demand

Implemented in MATLAB, this model is a two-part simulation that simulates electricity demand in the short-run on an hour-to-hour basis using an autoregressive regression with calendar (*i.e.*, hour of day, day of week, *etc.*) and temperature variables [22]. This captures diurnal and seasonal demand patterns. As the ICS-EPP moves forward in time, these forecasts are updated by a second long-run model of interannual demand growth that captures macroeconomic variables, such as electricity and natural gas prices, population, and gross state product. Regression-based methods are popular for capturing diurnal and interannual electricity demand patterns [23], [24]; however, integrating these predictions with long-term growth factors is novel. The regression models are fit using geographically-diverse data sets and comparisons of forecasts to out-of-sample actual consumption data show accurate predictive power [22]. Users can modify the number of number of commercial and industrial customers, the retail price structure, and macroeconomic growth rates, which are communicated to the model through LAPIS. The population input to the long-run model matches that of the residential energy demand model, as do the short-run weather variables. The output is a 10-minute resolved vector of aggregate commercial and industrial electricity demand.

C. Transportation Energy Demand

Implemented in MATLAB/Simulink, this model simulates energy use when household members leave the home (as determined by the residential model) [25]. This includes all leisure and work-related travel. This model does not consider commercial and industrial transport, as any associated electricity use is captured in the commercial and industrial demand model. The model uses a three-stage process. First, each trip's total travel time is determined based on the duration the individual is away. This is then translated into a velocity profile using a Markov-chain model. A backward vehicle dynamic simulator is finally used to compute energy use. This model architecture can simulate multiple vehicle types, including conventional vehicles and PEVs. The model is calibrated to empirical driving data and validated by comparing aggregate transportation energy consumption to national averages [25].

LAPIS is used to couple the transportation model with the residential and unit commitment models. The transportation model uses LAPIS to read simulated activity patterns, which determine when vehicle trips occur. Vehicle technology adoption will eventually be modeled endogenously, making further use of LAPIS.

D. Unit Commitment and Dispatch

Electricity generation is simulated using an industry-standard unit commitment and dispatch model [26]. The model is implemented as a mixed-integer program in Java using the CPLEX 12.3 optimization API. Inputs include energy demand data, which are given by the residential, commercial and industrial, and transportation models. ICS-EPP currently has the user specify the generation mix, which is input via LAPIS. A technology investment model (under development) will eventually model these decisions endogenously, based on a cost-minimization objective and pertinent constraints (*e.g.*, renewable portfolio standards or CO_2 restrictions). The model outputs, which are published to the visualization model using LAPIS, include the electric output of each generator, and associated costs and emissions.

E. Visualization

The visualization model is responsible for reporting and controlling the overall simulation. The model provides a user interface and control mechanism for the user to start, stop, monitor, and modify simulations during runtime. It is designed to be agnostic to the underlying simulation.

One issue that traditionally plagues the use of steering is the lack of an intuitive interface. Thus, the use of these systems is often limited to computer scientists with domain-specific knowledge of steering. The visualization model aims to allow novice and non-technical users to gain insights from application-specific simulations without needing advanced training in communication systems and parallel computing. In our case, business analysts and policymakers can control and monitor how different system inputs (*e.g.*, generation mix and technology uptake) affect simulation outputs (*e.g.*, electricity supply and energy use).

The visualization model has three software components (see Fig. 3), the front-end, back-end processing, and storage components.

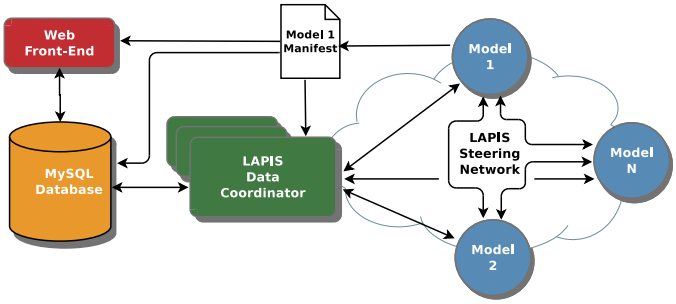

Fig. 3. LAPIS visualization system architecture.

1) Front-end Component: This component is entirely web-based, providing steering system users a complete abstraction from the details of the inner-workings of the simulation. The front-end, written in multiple web framework scripting languages, presents both novice and expert users the ability to dynamically change and monitor simulation inputs and outputs in real-time using a dynamic AJAX/HTML browser-based user interface.

2) Back-end Component: This component uses the LAPIS API to communicate with the other steered models. The back-end is written in Java and implements the native Java LAPIS API to appear on the LAPIS steering network as another model with its own defined inputs and outputs. By taking advantage of the user-defined inputs and outputs of the back-end component, other model authors can implement certain controllable inputs to (and outputs of) their models, providing on-line controllability of their specific model simulations from the web-based front-end.

The back-end component also ensures synchronization between the models. The need for this is simple—different models operate at different rates. For example, the commercial and industrial energy demand model takes less than one hundredth of the time taken by the residential model. Without a synchronization mechanism, the faster models generate more data than is needed by the other models to progress. This raises the need for complicated data and memory management. The back-end component simplifies this by requiring all models connected to the LAPIS network to publish a timestep variable, indicating the timestep of data being simulated by the particular model. These published variables are used to ensure that all of the models are time-synchronized as the simulation progresses, since a model only proceeds to the next timestep once all timestep dependencies are met (see Section III-B2 for further details).

3) Storage Component: Implemented with a relational database system (MySQL), this component is a go-between for the back- and front-end components and stores all simulation data and settings specified by the manifests for each individual simulation (see Section III-B1 for further details of the model manifest system). In addition to handling all simulation data, the storage component also manages user account information and user interactions on the system.

4) Demonstration Implementation: There currently exists a multi-layer demonstration implementation of the visualization model that includes a complete set of web layers that communicate, store, and display simulation states using LAPIS. A MySQL database stores the manifests, as well as the simulation data that can be displayed on the front end. On top of the database, there exists a Representational State Transfer (REST) web service that is implemented in PHP. As with many RESTful webservices, JSON is utilized to pass information back and forth via HTTP. The front-end is implemented using standard HTML/CSS and javascript (AJAX, jQuery) that utilizes the RESTful webservice. MATLAB and Java APIs also exist to facilitate the upload of initial and manifest data.

Fig. 4 shows the web-based interface of the ICS-EPP visualization model at runtime. The top of the page (1) provides a drop-down menu, allowing the user to select from a set of steered computations available on the back end. Based on this selection, the middle of the page (2) provides a list of the constituent sub-models. Selecting a sub-model presents a list of variables (3) that are published on the LAPIS network that can be viewed or changed. The sub-model and published variable lists are generated dynamically, based on the model manifest files. Finally, the interface dynamically generates figures displaying published variable information (4) and allows the user to change the values of variables that are specified as writable (5).

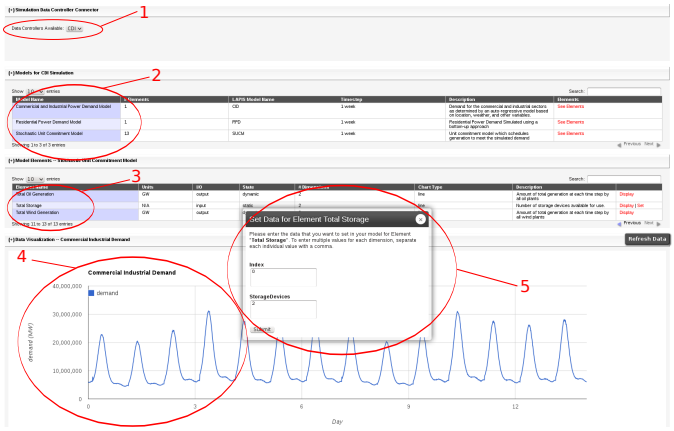

Fig. 4. Demonstration implementation of the ICS-EPP visualization model.

III. Integration of Independent Models: Forming the Multi-Model Simulation

The ICS-EPP is a multi-model simulation built out of independently developed models, each covering a specific domain of interest, and a front-end interface. These components are interconnected via LAPIS and rely upon each other for input parameters and data. We developed early versions of this system, during the course of which we established some key design strategies. Whereas model and front-end integration was initially time consuming and difficult, these strategies make the process nearly seamless and provide automation benefits.

A. The LAPIS System

The primary design goal of LAPIS is to provide a steering system that is sufficiently easy to use that non-specialist researchers could learn the basics of its use in an afternoon. At the same time, LAPIS is designed to be as platform-, operating system-, and language-independent as possible. This ensures that simulations using LAPIS are not compromised in terms of portability, maintainability, or structure. LAPIS also provides a simple mechanism for implementation of multi-model simulations. Much like MPI and OpenMP do for parallel computing, LAPIS provides an exampl standard for steering systems. Today, LAPIS provides a cross-platform middleware, a modular communication mechanism that currently supports TCP-IP based communication, and APIs for both Java and MATLAB. Future plans include additional communication modules supporting File I/O, Infiniband, and SSH tunnels. Additional plans involve the creation of APIs for C/C++, Python, and other languages.

The LAPIS system models complex steered applications as a collection of peer-to-peer connected nodes, each using a three-layer software stack (see Fig. 5). The peer-to-peer network is built and maintained automatically without the need for end-user intervention. The API layer helps ensure that a variety of languages can be used with LAPIS. The COM layer ensures that the communication mechanism used to connect components of the steered application can be easily changed without requiring any user implementation changes. Finally, the Daemon layer, written in Java, ensures that reimplementation of the core functionality of the LAPIS system is never required. The three-layer stack additionally ensures that LAPIS minimally impacts simulation performance. The stack also ensures that using LAPIS does not negatively affect maintainability or portability of the simulation. The stack provides a set of standard interfaces used to expand the functionality of the LAPIS system.

Use of the API itself is quite simple as usage is based on a small number of easy to understand methods. The function of the LAPIS system is based on a published-data model. Within this model, any given node on the steering network can publish internal state through an API call. Once published, a node's internal state can be accessed via '`get`' and '`set`' API calls by any other node on the network. Critical to the success and functionality of this model is the handling of requests for published data. Specifically, when a remote node requests either a '`get`' or '`set`,' LAPIS handles the request automatically without interruption of the user code in any way. As such, '`get`' and '`set`' commands can be issued and responded to without *a priori* knowledge on the part of the developer of either code body. For a simulation to be made steerable, code is added to initialize the LAPIS system and publish the internal states. No other modifications need be made to the simulation code. These modifications alone are sufficient to give a front-end interface, also using LAPIS, the ability to connect to, read, and modify the internal state of the simulation as it runs.

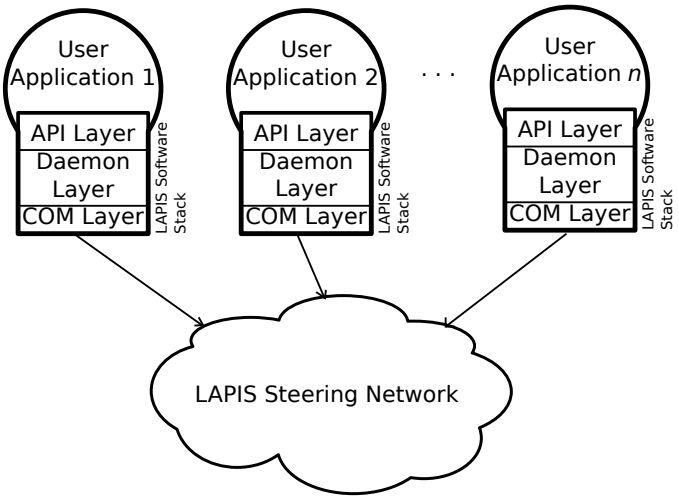

Fig. 5. The structure of the LAPIS system. LAPIS is as general purpose a steering system as possible. As such, the system makes no distinctions between nodes on the steering network.

B. Development and Design Strategies

Our experience in integrating the ICS-EPP reveals two useful model development strategies. Early on in the first-generation multi-model, individual models were connected via LAPIS manually. One developer was given responsibility for implementation of a model manager, inspecting the code of each model to extract the model name and the names of published values. These names were then used in the implementation of the manager. It was quickly determined that such a system would not only be a potential runtime bottleneck, but also a development bottleneck as a greater number of more complex models become involved in the simulation. With this in mind, the second-generation system was built with each model directly interfacing with other dependent models. Each model is responsible for monitoring the state of other dependent models. Dependent data are accessed only after the generating model passes a certain timestep within the simulation, and this time-synchronization is facilitated by the back-end controller of the visualization model. While this successfully eliminates the runtime bottleneck, full system integration was still complicated and time consuming.

After the second generation, the idea of model manifests was developed and implemented for each model. The addition of manifests to the situation-aware models of the second-generation system resulted in our third-generation model and

the approaches discussed below. Using this system, model integration is greatly simplified and modularized. Interfacing models to each other and to other components (such as front-end interfaces) is done procedurally, and computational and network loads can be distributed more evenly.

1) The Model Manifest System: Each model developer is required to provide what is termed a 'Model Manifest.' This manifest is a plain-text file that describes the full steering interface of the model. These manifests are written in JSON format, ensuring that they are human- and machine-readable. The steering interface of the model includes a number of key factors. These include the minimal timestep that is externally viewable from the model (which may differ from the internal timestep), the name of the model, and the published states within the model.

In our usage, the published states come in two categories: input parameters and simulation output. Input parameters are any scalar values and flags that affect the internal operation of the simulation (*e.g.*, pause flags, flags that control execution of certain model subsections, and coefficients used for convergence and scaling). The name, type, size, and default values for these parameters, and a human-readable explanation of they are used for within the simulation, are given in the manifest. Simulation output values are characterized by their continued growth as simulation time progresses. The manifest specifies the names, type, unit, and rate of growth of these output values, and a human-readable description of what the values represent.

These manifests allow other developers to make use of the simulations' outputs within their own simulations. Using LAPIS and the variable names and description within the manifests, each model developer can leverage the other models in the system without significant coordination effort. LAPIS provides the '`get`' and '`set`' commands which provide access to the published data by name. No further coordination is required so long as each published internal state matches the model manifest. Additionally, our front-end interface makes use of all the model manifests in order to dynamically generate the user interface. By reading each manifest, each model can be presented to the end user with all of its inputs and outputs available and clearly labeled.

2) Situation-Aware Models: The model manifests, paired with a scheme we call 'situation-aware models' and outline below, allows construction of large scale multi-models with almost no integration effort.

a) Simulation Timestep: Every model uses an outermost loop that iterates over timesteps. Models use LAPIS to publish their timestep variables and output data are indexed by timestep.

b) Dynamically Growing Output: Models use LAPIS to publish one-timestep sized instances of output data. As the simulation runs, all output data for the entire simulation run are kept available.

c) Time Dependencies: Developers are responsible for knowing their models' data dependencies and the associated time shift. Each developer is responsible for maintaining a small section of code that monitors model dependencies, and only having the model progress when such dependencies are met. Time-synchronization is managed by the back-end component of the visualization model.

d) Steering Inputs: Every model publishes all inputs that could conceivably be used to steer the simulation.

By having all developers follow these four principals, model integration simply involves reading manifests and making timestep comparisons. Models pause when data dependencies are not met, and resume when model data become available. Different developers need only exchange manifests, thus avoiding any developers having to analyze someone else's code. Finally, the manifests combined with timestep publishing allows a front-end interface to be dynamically generated in a highly logical way.

IV. Results of Strategies Used

LAPIS provides numerous benefits to multi-model developers and users. The additional code needed to implement the system is very limited and easy to write. The tools and functionalities provided by LAPIS are user-friendly, intuitive, compact, and can be written as a wrapper around the model itself. The coordination needed between models is minimal and developers need not know *a priori* about a future integration via LAPIS. Any change within a sub-model does not require changes in dependent models nor in the communication structure. Updated manifests reflect any differences in published variables (the only ones seen by other models) between updates, thus other models can easily be updated to match these changes, if needed.

A. Benefits of Situation-Aware, Manifest Integration

The manifests themselves are beneficial in several ways. JSON-format files are easily human-readable. Furthermore, the manifest includes all the relevant information about the published variables—*e.g.*, the meaning of the data, units, time interval in the simulation, frequency of publishing, size of the variable, plotting details.

B. Benefits Derived from LAPIS System for Multi-Model

Written as a wrapper around the model, LAPIS maintains history of all the internal states of the simulation at every run and saves them to an external database. This allows executions to be stopped, restarted, and replayed. A stopped execution can be examined and redirected if the simulation is moving in an undesirable direction. The entire system can also be loaded from a single checkpoint saved in the SQL database, then modified and rerun to see the effects of one or several variables in isolation.

In isolating the core LAPIS architecture away from the multi-model simulation, developers derive the benefits of a language- and platform-independent tool. The interface the user interacts with is standard and universal, and the code is written in the same language as the developer's model. Each model can be written in the most appropriate environment and models can operate together regardless of platform. Thus, if some models require specific tools or technologies to function, this does not limit the development options of the other researchers.

V. LAPIS Demonstration

To demonstrate the use of LAPIS in steering multi-model simulations and as a general-purpose computational steering tool, we present some illustrative results of its capabilities.

A. ICS-EPP Demonstration

We first demonstrate the use of the ICS-EPP multi-model in exploring the ramifications of energy policy decisions, through a case study using the residential energy demand, commercial and industrial energy demand, and unit commitment and dispatch models. The study compares generation costs, patterns, and CO_2 emissions in three cases: a base scenario, using only the available conventional and wind generation (≈1.9 GW) in Texas in 2005 (Madaeni and Sioshansi [27] detail the case study); a high wind-penetration scenario, in which 10 GW of added wind capacity; and a third scenario with the base scenario generation mix and a $30/ton CO_2 tax.

Table I summarizes modeled annual generation costs, fuel mix, and CO_2 emissions in the three cases modeled. As expected, both added wind and a carbon tax reduce CO_2 emissions, with the latter giving greater reductions. Table I reports fuel costs, which amount to about $16.30/MWh, $15.49/MWh, and $17.86/MWh in the three cases, respectively. The case with the carbon tax imposes an additional cost of $1.6 billion, which is collected by the government and could be redistributed or otherwise used.

TABLE I
Summary of Annual Generation Cost, Breakdown, and CO_2 Emissions

	Base	High-Wind	Carbon Tax
Generation Cost [$ billion]	2.15	2.04	2.35
Generation Breakdown [%]			
Coal	21	10	19
Natural Gas	40	32	41
Wind	5	25	5
CO_2 Emissions [million short tons]	65.9	45.6	52.1

B. LAPIS Demonstration

We also perform two scalability tests to demonstrate the capabilities of LAPIS as a general steering and multi-model simulation tool. The first test consists of passing arrays of varying lengths between nodes connected on a two-node LAPIS network. The first node publishes an array of double-precision floating point numbers while the second reads the values using the LAPIS '`get`' command. The second test is done by passing arrays of 100 double-precision floating point numbers between nodes of different-sized LAPIS networks. In these tests, the LAPIS network consists of a single 'main' node, which provides address information for the other N nodes. These N nodes each publish a single array. The test is conducted by having node n read the values published by node $n-1$ (node 1 reads the values published by node N) using the LAPIS '`get`' command. The '`get`' commands are issued repeatedly and simultaneously by the N nodes. All of the nodes used in these tests are implemented in Java and the `nanoTime` function from the Java system class is used to measure the time taken for the '`get`' commands to finish. The tests are conducted using the OSC Oakley system. Fig. 6 summarizes the results of the testing, showing the average times taken for the '`get`' command to finish under the different scenarios. The figure shows that the time taken for a '`get`' increases quadratically with array size. Conversely, the time taken for a `get`' is insensitive to LAPIS network size.

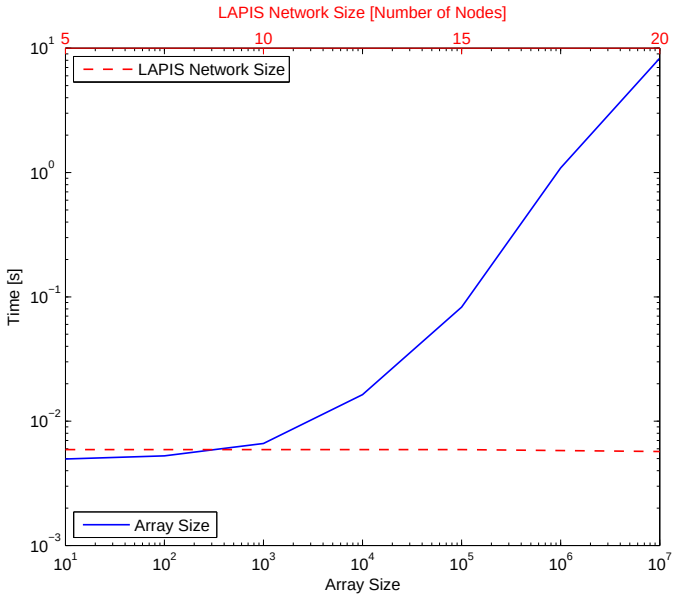

Fig. 6. Performance of the LAPIS system as a function of array and steering network size.

VI. Conclusion

Our group is using LAPIS to create a complex multi-model energy system simulation. This simulation is not being created monolithically, but rather from an assemblage of smaller, easier to develop and test models. These models are being developed by an interdisciplinary team of researchers working independently and targeting previously established interfaces and standards.

This model of simulation development carries some key advantages. First, LAPIS allows parallel development of model components without need for tight coordination between developers. Second, the manifest system and situation-aware models allow for easy integration of component models into a larger complex system. These benefits speed up development and give a more useful simulation. These techniques are widely applicable and can be used in any scenario where a variety of models interact.

We demonstrate the use of our multi-model simulation in examining the impacts of energy policy alternatives. The ICS-EPP will be further developed to allow simulation of more complex policy and technology alternatives. We also demonstrate the ability of LAPIS to efficiently steer simulations, involving different data array sizes and numbers of simulation nodes.

REFERENCES

[1] J. D. Mulder, J. J. van Wijk, and R. van Liere, "A Survey of Computational Steering Environments," *Future Generation Computer Systems*, vol. 15, no. 1, pp. 119–129, 1999.

[2] S. G. Parker and C. R. Johnson, "SCIRun: A Scientific Programming Environment for Computational Steering," in *Proceedings of the IEEE/ACM SC95 Conference Supercomputing*. San Diego, CA: Institute of Electrical and Electronics Engineers, 3-8 December 1995.

[3] S. Bullock, J. Cartlidge, and M. Thompson, "Prospects for Computational Steering of Evolutionary Computation," in *Proceedings of the Eighth International Conference on Artificial Life*, R. Standish, M. A. Bedau, and H. A. Abbass, Eds. The MIT Press, 2002, pp. 8–13.

[4] R. S. Kalawsky and S. P. Nee, "Important issues concerning interactive user interfaces in grid based computational steering systems," in *Proceedings of the UK e-Science All Hands Meeting 2004*, S. J. Cox, Ed. Nottingham, United Kingdom: Engineering and Physical Sciences Research Council, 31 August-3 September 2004, pp. 886–893.

[5] R. Marshall, J. Kempf, S. Dyer, and C.-C. Yen, "Visualization methods and simulation steering for a 3D turbulence model of Lake Erie," in *Proceedings of the 1990 symposium on Interactive 3D graphics*. New York, NY: Association for Computing Machinery, 1990, pp. 89–97.

[6] W. Gu, G. Eisenhauer, E. Kraemer, K. Schwan, J. Stasko, J. Vetter, and N. Mallavarupu, "Falcon: On-line Monitoring and Steering of Large-scale Parallel Programs," in *Proceedings of the Fifth Symposium on the Frontiers of Massively Parallel Computation*, McLean, VA, 6-9 February 1995, pp. 422–429.

[7] D. A. Reed, C. L. Elford, T. M. Madhyastha, E. Smirni, and S. E. Lamm, "The Next Frontier: Interactive and Closed Loop Performance Steering," in *Proceedings of the 1996 International Conference on Parallel Processing Workshop on Challenges for Parallel Processing*, Ithaca, NY, 12 August 1996, pp. 20–31.

[8] T. N. Palmer, A. Alessandri, U. Andersen, P. Cantelaube, M. Davey, P. Délécluse, M. Déqué, E. Díez, F. J. Doblas-Reyes, H. Feddersen, R. Graham, S. Gualdi, J.-F. Guérémy, R. Hagedorn, M. Hoshen, N. Keenlyside, M. Latif, A. Lazar, E. Maisonnave, V. Marletto, A. P. Morse, B. Orfila, P. Rogel, J.-M. Terres, and M. C. Thomson, "Development of a European Multi-Model Ensemble System for Seasonal to Inter-Annual Prediction (DEMETER)," *Bulletin of the American Meteorological Society*, vol. 85, no. 6, pp. 853–872, June 2004.

[9] K. P. Georgakakos, D.-J. Seo, H. Gupta, J. Schaake, and M. B. Butts, "Towards the characterization of streamflow simulation uncertainty through multimodel ensembles," *Journal of Hydrology*, vol. 298, no. 1-4, pp. 222–241, October 2004.

[10] Y. Osana, T. Fukushima, M. Yoshimi, Y. Iwaoka, A. Funahashi, N. Hiroi, Y. Shibata, H. Kitano, and H. Amano, "An FPGA-Based, Multi-model Simulation Method for Biochemical Systems," in *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*. Denver, CO: Institute of Electrical and Electronics Engineers, 3-8 April 2005.

[11] G. A. Meehl, C. Covey, T. Delworth, M. Latif, B. McAvaney, J. F. B. Mitchell, R. J. Stouffer, and K. E. Taylor, "The WCRP CMIP3 Multimodel Dataset: A New Era in Climate Change Research," *Bulletin of the American Meteorological Society*, vol. 88, pp. 1383–1394, Sepp 2007.

[12] S. M. Pickles, R. Haines, R. L. Pinning, and A. R. Porter, "Practical Tools for computational Steering," in *Proceedings of the UK e-Science All Hands Meeting 2004*, S. J. Cox, Ed. Nottingham, United Kingdom: Engineering and Physical Sciences Research Council, 31 August-3 September 2004, pp. 579–586.

[13] S. Jha, S. Pickles, and A. Porter, "A Computational Steering API for Scientific Grid Applications: Design, Implementation and Lessons," in *Proceedings of the Workshop on Grid Application Programming Interfaces*, Brussels, Belgium, 20 September 2004.

[14] G. Eisenhauer, K. Schwan, W. Gu, and N. Mallavarupu, "Falcon – Toward Interactive Parallel Programs: The On-line Steering of a Molecular Dynamics Application," in *Proceedings of the Third IEEE International Symposium on High Performance Distributed Computing*, San Francisco, CA, 2-5 August 1994, pp. 26–33.

[15] A. Modi, L. N. Long, and P. E. Plassmann, "Real-time visualization of wake-vortex simulations using computational steering and Beowulf clusters," in *VECPAR'02 Proceedings of the 5th international conference on High performance computing for computational science*, J. M. L. M. Palma, A. A. Sousa, J. Dongarra, and V. Hernndez, Eds. Heidelberg, Germany: Springer-Verlag Berlin, 2003, pp. 464–478.

[16] A. Capasso, W. Grattier, R. Lamedica, and A. Prudenzi, "A bottom-up approach to residential load modeling," *IEEE Transactions on Power Systems*, vol. 9, pp. 957–964, May 1994.

[17] I. Richardson, M. Thomson, and D. Infield, "A high-resolution domestic building occupancy model for energy demand simulations," *Energy and Buildings*, vol. 40, pp. 1560–1566, 2008.

[18] J. Widén and E. Wäckelgård, "A high-resolution stochastic model of domestic activity patterns and electricity demand," *Applied Energy*, vol. 87, pp. 1880–1892, June 2010.

[19] M. Muratori, M. C. Roberts, R. Sioshansi, V. Marano, and G. Rizzoni, "A highly resolved modeling technique to simulate residential power demand," *Applied Energy*, vol. 107, pp. 465–473, July 2013.

[20] M. Muratori, V. Marano, R. Sioshansi, and M. C. Roberts, "Residential Power Demand Prediction and Modelling," in *The 24th International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems*, Novi Sad, Serbia, 4-7 July 2011.

[21] M. Muratori, V. Marano, R. Sioshansi, and G. Rizzoni, "Energy consumption of residential HVAC systems: a simple physically-based model," in *2012 IEEE Power and Energy Society General Meeting*. San Diego, CA, USA: Institute of Electrical and Electronics Engineers, 22-26 July 2012.

[22] A. Pielow, R. Sioshansi, and M. C. Roberts, "Modeling Short-run Electricity Demand with Long-term Growth Rates and Consumer Price Elasticity in Commercial and Industrial Sectors," *Energy*, vol. 46, pp. 533–540, October 2012.

[23] R. Ramanathan, R. F. Engle, C. W. J. Granger, F. Vahid-Araghi, and C. Brace, "Short-run forecasts of electricity loads and peaks," *International Journal of Forecasting*, vol. 13, pp. 161–174, June 1997.

[24] V. Bianco, O. Manca, and S. Nardini, "Electricity consumption forecasting in Italy using linear regression models," *Energy*, vol. 34, pp. 1413–1421, September 2009.

[25] M. Muratori, M. J. Moran, E. Serra, and G. Rizzoni, "Highly-Resolved Modeling of Personal Transportation Energy Consumption in the United States," *Energy*, vol. 58, pp. 168–177, September 2013.

[26] G. S. Sheble and G. N. Fahd, "Unit commitment literature synopsis," *IEEE Transactions on Power Systems*, vol. 9, pp. 128–135, February 1994.

[27] S. H. Madaeni and R. Sioshansi, "The Impacts of Stochastic Programming and Demand Response on Wind Integration," *Energy Systems*, vol. 4, pp. 109–124, June 2013.

Harrison B. Smith (S'02) is high performance and scientific computing consultant. His research focuses on scientific computing and traditional and alternative high performance computing methods.

He holds a B.S., M.S., and Ph.D. in electrical and computer engineering from The Ohio State University.

Amy Pielow is a Ph.D. student in the Integrated Systems Engineering Department at The Ohio State University. Her research focuses on energy demand and stochastic power system planning models.

She holds a B.A. in econonomics and a B.S. in statistics from the University of Minnesota Duluth.

Adithya Jayakumar is a Ph.D. student in the Electrical and Computer Engineering Department at The Ohio State University. His research focuses on signal and image processing, agent-based modelling and simulation, and high-performance computing.

He holds a B.E. in electronics and communication engineering from Anna University.

Matteo Muratori (S'12) is a Ph.D. candidate in the Mechanical and Aerospace Engineering Department at The Ohio State University. His research focuses on advanced energy storage systems, modeling multi-scale integrated energy systems, and modeling and optimization of residential and transportation energy consumption.

He holds B.S. and M.S. degrees in energy engineering from Politecnico di Milano.

B. J. Yurkovich is on the research staff at the Center for Automotive Research at The Ohio State University. He specializes in the design and implementation of applied software and battery pack systems.

He holds a B.S. in computer science engineering and an M.S. in mechanical and aerospace engineering from The Ohio State University.

Ramteen Sioshansi (M'11–SM'12) is an assistant professor in the Integrated Systems Engineering Department at The Ohio State University. His research focuses on renewable and sustainable energy system analysis and the design of restructured competitive electricity markets.

He holds a B.A. in economics and applied mathematics and an M.S. and Ph.D. in industrial engineering and operations research from the University of California, Berkeley, and an M.Sc. in econometrics and mathematical economics from The London School of Economics and Political Science.

Ashok Krishnamurthy (S'82–M'83) is deputy director of the Renaissance Computing Institute and Adjunct Professor of Computer Science at the University of North Carolina at Chapel Hill. His research interests are in computational modeling, machine learning and signal processing.

He holds a Ph.D. from the University of Florida and an undergraduate degree in Electrical and Computer Engineering from the Indian Institute of Technology.

Giorgio Rizzoni (F'04) is the Ford Motor Company Chair in ElectroMechanical Systems and a professor in the Mechanical and Electrical and Computer Engineering Departments at The Ohio State University. His research interests are in future ground vehicle propulsion systems, including advanced engines, electric and hybrid-electric drivetrains, advanced batteries and fuel cell systems.

He received his B.S., M.S., and Ph.D. electrical and computer engineering from the University of Michigan.

Matthew C. Roberts is an associate professor in the Agricultural, Environmental, and Development Economics Department at The Ohio State University. His research interests are in price and revenue risk management in the commodity grain markets. He also tracks energy and agricultural commodity markets.

He holds a B.A. in economics from William Jewell College and a Ph.D. in economics from North Carolina State University.