Towards an efficient multidisciplinary system-level framework for designing and modeling complex engineered microsystems

J. V. Clark[1,2], Yi Zeng[1], Pankaj Jha[1]

[1]School of Electrical and Computer Engineering
[2]School of Mechanical Engineering
Purdue University, USA, E-mail: jvclark@purdue.edu

ABSTRACT

We present advances in designing and modeling complex microsystems at the network/system-level. For design, we develop a graphical user interface that allows users to quickly configure complex systems in 3D using a computer mouse or pen. And we couple it with a powerful Matlab-based netlist language for design flexibility. For modeling, we apply recent advances in analytical system dynamics and differential-algebraic equations into a framework that facilitates the systematic modeling of multidisciplinary systems with holonomic and non-holonomic constraints. For a test case, we efficiently simulate a microsystem comprising gears, hinges, slider, electronic components, comb drives, and electromechanical flexures. In comparison, it is difficult to model, configure, and simulate such a microsystem using conventional tools.

Keywords: GUI, netlist, DAE, PSugar, Sugar, MEMS, design, modeling, simulation, constraints, multidisciplinary.

1. INTRODUCTION

Distributed-element tools using finite element analysis (FEA) are often used for characterizing new geometries and interactions [1]. However, a significant number of microsystem designers reuse parameterized sets of well-characterized elements. This area of design and modeling led to the development of network/system-level tools for microsystems. Examples include Sugar [2], Nodas [3], Architect [4], and Synple [5]. What such tools have in common are: (1) the ability to abstract the configuration of a system in the form of a netlist, where each line of text describes an element's connectivity to other elements, its orientation, and it modeling parameters; and (2) a significant reduction in the computation time (when compared to FEA) by using computationally-efficient models based on reduced order modeling, matrix structural analysis, and or modified nodal analysis [2]-[5]. Such benefits facilitate the exploration of parameterized design spaces of systems with a multitude of components.

Recently, there has been interest in accommodating the user's need to efficiently design, model, and simulate complex engineered microsystems. Users demand friendlier graphical user interfaces, more flexible netlist capabilities, and less complicated modeling frameworks [2]. In effort toward fulfilling these requirements, in this paper we expand upon our previous effort called Sugar to develop an interactive GUI-netlist and a systematic modeling framework. The present effort is called PSugar. Its architecture is illustrated in Figure 1.

In Section 2, we present our powerful netlist features. In Section 3, we present our novel 3D interactive GUI technology. In Section 4, we present our systematic modeling framework. In Section 5, we discuss differential-algebraic equation solvers. And in Section 6, we provide a simulation example based on a microsystem developed by Sandia National Lab.

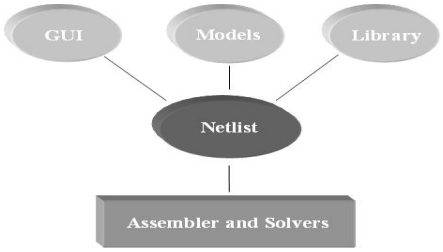

Figure 1: Architecture of PSugar. The ellipses symbolize parameters and constitutive relationships of a prescribed system; the rectangle symbolizes solution algorithms, which assemble and solve the mathematical representation of the system.

2. NETLIST

In this section we present PSugar's netlist capabilities by comparing it to Sugar's netlist capabilities.

What we improve on is the limited capabilities of Sugar's netlist language. For instance, Sugar's netlist syntax has the form

$$model_1 \;\; layer_1 \;\; [nodes_1] \;\; [parameters_1]$$
$$model_N \;\; layer_N \;\; [nodes_N] \;\; [parameters_N]$$

where *model* is the name of the element model, e.g., mechanical flexure, resistor, comb drive, anchor, etc.; *layer* is the material that the element is composed of; *nodes* is a list of element nodes which are the places that the element can be connected to the nodes of other elements; and *parameters* is the list of modeling parameters, e.g. orientation, Young's modulus, residual stress, temperature, etc. Sugar's netlist language allows for subnets and importing parameter values from the Matlab workspace into the netlist. However, some problems with this format are as follows. The format requires the user to continuously

switch between two incompatible syntaxes – one for Sugar, and another for Matlab and Sugar functions. The netlist format allows simple arithmetic expressions such as defining constants, allowing nodes to be represented by simple array variables, and nested loops. However, it does not allow sophisticated computations such as those dealing with matrices or those needing access to external functions from within the netlist. This limits design flexibility.

PSugar's netlist format overcomes these issues by being 100% Matlab compatible. Many students and professionals are familiar with Matlab. PSugar's netlist is an m-file with arbitrary format, where the only requirement is that the output must be a Matlab cell structure of the form

$$[\ model_1\ \{nodes_1\}\ \{parameters_1\};$$
$$model_N\ \{nodes_N\}\ \{parameters_N\}];$$

where *model* is the name of the element model; *nodes* is a list of node names; and *parameters* is the list of modeling parameters such as,

{'L' 100e-6; 'force' comb(V,W,H, L,N); 'C' cross(A,B) };

where the designer has unfettered access to all Matlab functions, workspace quantities, and third party plug-ins.

3. GUI

In this section, we present a few of our efficient, interactive GUI features.

Besides using a netlist, another way to configure a system in PSugar is by using its graphic configuration window (GCW). With a computer mouse or pen, drawplanes and elements can be quickly and easily positioned, repositioned, and default values can be edited in the GCW. The GCW is coupled to an interactive netlist window (INW). That is, each element that is graphically configured in the GCW has its corresponding netlist text automatically generated in the INW; and vice versa, a line of netlist text entered into the INW immediately updates the system configuration display in the GCW.

Drawplanes are uniquely determined with 3 coordinates. These 3 coordinates are identified in PSugar by projecting the 2D cursor position upon the 3D xy-, zy-, or xz-planes of the coordinate axis; or other object. For example, Figures 2a-2c illustrate the 3-button-click sequence to configure a drawplane in 3D. Figure 2d shows an element placed on that drawplane using 2 button clicks. By dragging a node about its associated plane, both elements and drawplanes can be readily repositioned. The GCW also has snap-to-grid and snap-to-node options.

The elements or subsystems that are configured onto the GCW are selected through an element menu window (EMW). The set of elements from PSugar's library that are listed in these EMWs are user-definable. For example, a user may choose resistor, capacitor, inductor, opamp, integrator, diode, rectifier, and transistor as button choices for one EMW; and comb drive, beam, anchor, folded flexure, hinge, point mass, and carbon nanotube as button choices for another EMW. E.g., see Figure 3.

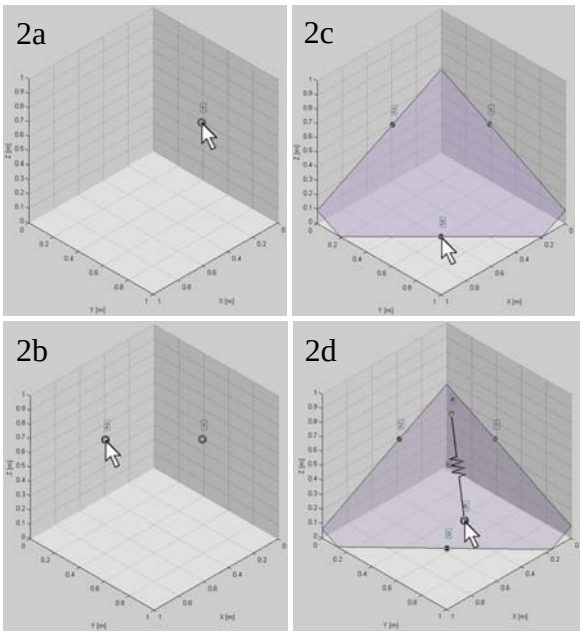

Figure 2: Configuring drawplanes in the GCW. 2a-2c shows a 3-button-click sequence to configure a drawplane in 3D. 2d shows the placement of a resistor element on the drawplane, using 2 button clicks. The 2a-2d sequence took a user a couple of seconds.

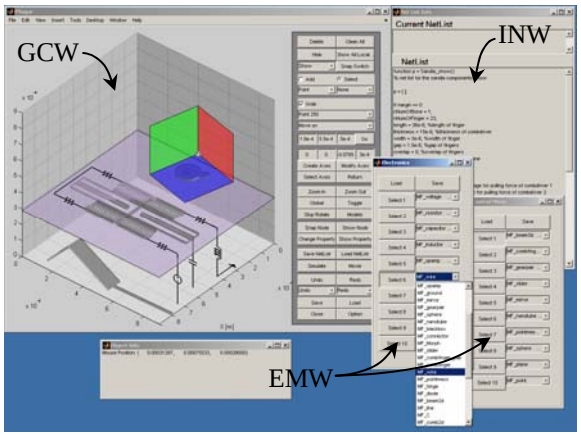

Figure 3: GUI. The GUI components shown are: GCW (graphic configuration window), INW (interactive netlist window), and a couple of EMWs (element menu windows). Elements shown include a hinged mirror with slider, folded flexure, and gear-pair in a local frame.

4. MODELING

In this section we discuss our systematic modeling method, which reduces the modeler's effort and readily accommodates systems subject to algebraic constraints.

In our previous work with Sugar, we represented microsystems as a set of second order differential equations (ODEs) of the form,

$$M\ddot{q} + B\dot{q} + Kq - F_{ext} = 0 \qquad (1)$$

where M, B, K are the system mass, damping, and stiffness matrices, and F_{ext} is the vector of externally

NSTI-Nanotech 2008, www.nsti.org, ISBN 978-1-4200-8505-1 Vol. 3

applied efforts such as electrostatic, gravitational, noninertial, voltage, stress, etc. [2]. However, ODE solvers are typically not equipped to solve a system of differential-algebraic equations (DAEs) [6], where the system mass matrix may be singular or zero. There are many complex microsystems that are amenable to DAEs, such as those comprising elements without inertia; elements with displacement-, flow-, dynamic variable-, or effort-constraints; or elements with inequality constraints.

The DAE form we use in PSugar is [7],

$$\frac{d}{dt}\nabla_f T^* + \nabla_f D + \nabla_q V - F_{ext} = \nabla_q T^* - \Phi_q^T \mu - \Psi_f^T \kappa \quad (2a)$$

$$0 = \Phi(q,t) \quad \text{vector of displacement constraints} \quad (2b)$$

$$0 = \Psi(f,q,t) \quad \text{vector of flow constraints} \quad (2c)$$

$$0 = \Gamma(e^\gamma, s, f, q, t) \quad \text{vector of effort constraints} \quad (2d)$$

$$0 = \dot{s} - \Lambda(e^\gamma, \dot{s}, s, f, q, t) \quad \text{vector of dynamic variables} \quad (2e)$$

where $T^*(f,q,t)$, $D(f,q,t)$, and $V(q,t)$ are kinetic co-energy, content, and potential; Φ, Ψ, Λ, and Γ are displacement-, flow-, dynamic variable-, and effort-constraints; $\mu(t)$ and $\kappa(t)$ are Lagrange multipliers; vector $q(t)$ represents generalized displacements such as the change in translation, rotation, charge, volume, entropy, etc. Comparing (2a) to (1), each term on the left-hand side of (2a) has the same meaning as each term in (1), respectively; and vector $f = \dot{q}$ is flow. However, (2a) allows kinetic co-energy to be a function of q and allows the dynamics to be constrained. Since vectors μ and κ are additional unknowns, relations (2b) and (2c) are required additional equations. (2d) is included to allow for effort e^γ constraints. And (2e) is included to allow for relations that are not represented within the energy functions; e.g., time derivatives of flow $\dot{s} = d\dot{q}/dt$ or time integrations of displacement $s = \int q\,dt$, etc.

Systematic modeling in PSugar is as follows. Each element has a representative parameterized model function containing its energy functions, constraints, and efforts. E.g. a linear 2-node, 12-DOF flexure model function returns the symbolic scalar $V_i = \frac{1}{2} q^{1\times 12} K^{12\times 12} q^{12\times 1}$. The assembler sums all energy functions, e.g. $V = \sum_{i=1}^{N} V_i$, then substitutes the functions into (2a)-(2e) for symbolic differentiation. Hence, in PSugar the modeler's effort is reduced to simply providing energy functions, constraints, and efforts. The common practice of rigorously manipulating a model into a particular form is eliminated.

5. SIMULATION

In general, the solution of a DAE involves solving a nonlinear algebraic equation at each time step.

For instance, applying linear approximations to (2), such as $\dot{q}_{n+1} \approx (q_{n+1} - q_n)/h_{n+1}$, Euler's method yields

$$\begin{pmatrix} \frac{q_{n+1} - q_n}{h_{n+1}} - f_{n+1} \\ M_{n+1} \frac{f_{n+1} - f_n}{h_{n+1}} + \Phi_q^T\big|_{n+1} \kappa_{n+1} + \Psi_q^T\big|_{n+1} \mu_{n+1} - \Upsilon_{n+1} \\ \Phi_{n+1} \\ \Psi_{n+1} \\ \Gamma_{n+1} \\ \frac{s_{n+1} - s_n}{h_{n+1}} - \Lambda_{n+1} \end{pmatrix} = 0 \quad (3)$$

where h is the step size in time, $M = \nabla_f^2 T^*$, $\Upsilon = Q - (\nabla_f T^*)_q f - (\nabla_f T^*)_t + \nabla_q T^* - \nabla_q V - \nabla_f D$, and the Jacobian $\partial F/\partial y_{n+1}$ can be evaluated using finite differences. There are several pubic domain solvers available for DAEs [6]. The choice of solver usually depends on the differential index of the DAE; that is, the minimum number of times that some or all of the equations would need to be differentiated in time to determine its underlying ODE. However, using an ODE solver to solve the resulting underlying ODE is not preferred, because the solution trajectory often drifts from the solution manifold that is defined by the explicit constraints in the original DAE. Methods such as BDF (Backward differentiation formula) and IRK (implicit Runge-Kutta) improve Euler's method by using higher-order approximations for $\dot{q}_{n+1}$, $\dot{f}_{n+1}$, and $\dot{s}_{n+1}$, and using variable step sizes.

Since Matlab's ode15s and ode23t solvers are only for index-1 DAE systems, we do not use them to directly solve (2), because its index can be as high as 3. Currently, we use any one of a collection of public domain high-index DAE solvers. Future work includes developing a DAE solver that exploits our particular DAE structure.

6. EXAMPLE

To exemplify our methodology, we simulate a microsystem similar to one that was developed by Sandia National Labs (Figure 5). We chose this particular system as an example because it is difficult model and simulate using conventional tools [8]. Multidisciplinary elements of the microsystem comprise resistors, capacitors, voltage sources, mechanical flexures, hinges, sliders, gears, and electrostatic comb drives. Its representation in PSugar is shown in Figure 6. We configure the system using both the GUI and netlist (e.g. the repetitive comb fingers). The system is represented by 4601 degrees of freedom. Using a Pentium-4, 3GHz processor with 1GB RAM, our BDF DAE solver averages 1.4 seconds per step in Matlab.

Identical ramp voltages applied to the two orthogonal sets of comb drives rotate the smallest gear counter-

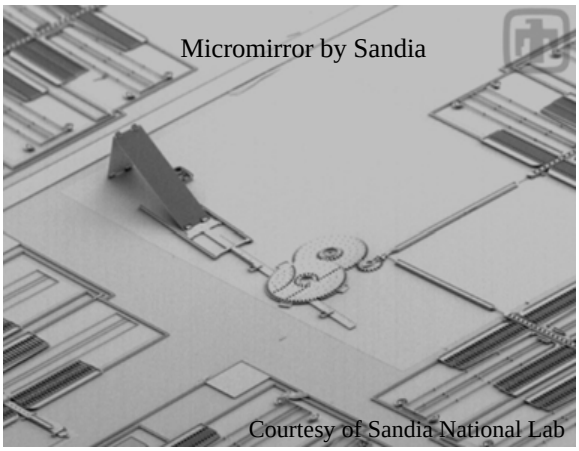

Figure 5: A complex microsystem.

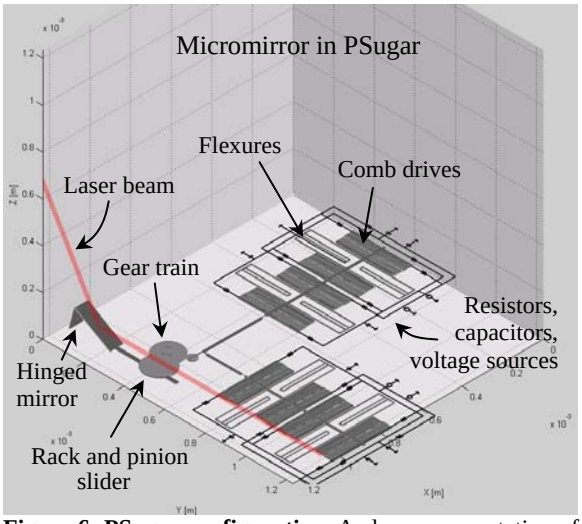

Figure 6: PSugar configuration. A close representation of the Sandia device from Figure 5 configured in PSugar. The red beam in the figure is a laser reflecting off the mirror.

clockwise $\sim \pi/2$ radians. As the voltage is removed, the system settles back to its initial state. See Figure 7. Slack between hinges and gear teeth are not modeled. The true geometry and material properties of actual microsystem shown in Figure 5 were not made available at the time of this writing. The actual friction in the hinges, gears, and slider are not known.

CONCLUSION

We presented a couple of design and modeling advancements toward an efficient system-level framework for complex engineered microsystems. Regarding design, we presented our interactive GUI that allows users to quickly and easily configure complex configurations; and we presented our powerful netlist language, which embraces the full flexibility and functionality of the Matlab language. Regarding modeling, we discussed the systematic method we use for representing complex, multidisciplinary

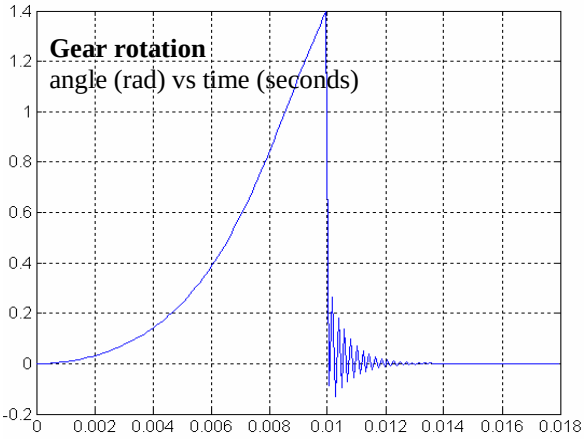

Figure 7: Transient analysis. Simulation of the configuration shown in Figure 6. A pair of linear voltage ramps applied across the two sets of orthogonal comb drives rotates the smallest gear a quarter turn. After the ramps end, the system settles back to its initial state.

components by their energy functions, constraints, and efforts. All code development was done in Matlab.

Some of our future work will involve verifying the performance of complex systems (qualitative results were presented here), developing a solver that fully exploits our DAE structure, and enhancing our library with components, subsystems, and complete systems.

ACKNOWLEDGEMENT

We thank R. A. Layton of Rose-Hulman Institute of Technology and B. Fabian of the University of Washington for insightful discussions on system dynamics.

REFERENCES

[1] S.D. Senturia, "CAD Challenges for Microsensors, Microactuators and Microsystems," *Proceedings of the IEEE*, Vol. 86, No. 8, August 1998, pp. 1611-1626.

[2] J. V. Clark and K. S. J. Pister, "Modeling, Simulation, and Verification of a Advanced Micromirror Using Sugar", *Journal of Microelectromechanical Systems*, Vol. 16, No. 6, 2007, pp. 1524-1536.

[3] G. K. Fedder and Q. Jing, "NODAS 1.3: Nodal Design of Actuators and Sensors", *Proc. IEEE/VIUF Int. Workshop on Behavioral Modeling and Simulation*, Orlando, FL, Oct 1998.

[4] Coventorware, 951 Mariners Island BLVD, Suite 205, San Mateo CA 94404. http://www.coventor.com

[5] IntelliSense Corporation, 600 W. Cummings Park Suite 2000, Woburn MA 01801. http://www.intellisense.com

[6] K. E. Brenan, S. L. Cambell, L. R. Petold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, 1996.

[7] R. A. Layton, *Principles of Analytical System Dynamics*, Springer, 1998.

[8] D. Sandison, "Keynote Address: Moving MEMS from Novelty to Necessity – a National Security Perspective", *TEXMEMS VII*, El Paso TX, 2005. http://www.uacj.mx/Texmems/Keynotes.htm