A Python Module for Modeling and Control Design of Flexible Robots

This article discusses the creation of a Python module for object-oriented modeling and control design of flexible robots using the transfer matrix method (TMM). The authors overcame several theoretical hurdles to apply the TMM to practical flexible robots and have experimentally validated the Python module's modeling capabilities.

Flexible robots offer the benefits of being lighter, faster, and cheaper to actuate than their rigid counterparts. However, robots with flexible links or joints also pose a considerable challenge because they might be composed of discrete and distributed parameter elements, many links, complicated actuators, and multiple feedback loops. The example system analyzed in this article poses two additional challenges: it's hydraulically actuated with two feedback loops in which the sensors and actuators aren't precisely collocated.

Existing modeling approaches for flexible structures are inadequate for designing controllers for flexible robots. The transfer matrix method (TMM) might be an excellent approach if some theoretical hurdles are overcome and if some new software package makes the approach more accessible and user friendly.[1–4]

This article discusses how we expanded the TMM's capabilities and developed a Python software module to make the TMM an excellent tool for the modeling and control design of practical flexible robots.[5]

1521-9615/07/$25.00 © 2007 IEEE
Copublished by the IEEE CS and the AIP

Ryan W. Krauss
Southern Illinois University Edwardsville
Wayne J. Book
Georgia Institute of Technology

System Description

Figure 1 shows a picture of the flexible robot we used in the experimental part of our work. The robot is called SAMII, which stands for small, articulated manipulator II. SAMII is a hydraulically actuated robot with rigid links mounted on the end of a cantilevered beam. The cantilevered beam represents a larger robot that would give SAMII a larger workspace and move it into the general position desired. Once SAMII is in position, the large robot's joints might be locked. Thus, the large robot must have long links to give SAMII a large workspace, even though this length inherently implies flexibility and vibration problems. We need modeling approaches and control schemes to deal with this flexibility and suppress or avoid vibrations.

Problem Statement

Figure 2 shows a block diagram of the control scheme. The goal is to develop transfer functions for G_θ and G_a, where G_θ specifies how the system will respond to commands to move to a desired position θ_d and is referred to as the motion control portion of the control scheme, and G_a is the vibration suppression controller. We want to find the optimal G_θ and G_a so that the robot moves quickly while also suppressing vibration.

The Need for Better Modeling Tools

Many approaches exist for modeling flexible structures, but two of the most prominent are finite element analysis (FEA) and the assumed modes method (AMM).

FEA is widely used in the analysis of flexible structures, but it isn't the ideal tool for control design of flexible robots. It's difficult, for example, to find an FEA software package that can model multiple feedback loops and hydraulic actuators. Moreover, even if we found a suitable package that could model the closed-loop response of hydraulically actuated flexible robots, we couldn't use such a model for control design without modal discretization. It would be a clumsy approach for very flexible robots in which the feedback controller affects the system's mode shapes.

Other researchers have applied the AMM to robotics,[6] but the approach quickly grows unwieldy as the number of links increases. Correctly handling element-connectivity conditions as we add more links to the model is quite burdensome, and it would quickly become very complicated if we applied this approach to the robot analyzed here. Additionally, this approach is clumsy for extremely flexible robots.

Expanding the TMM's Capabilities

The TMM has the potential to overcome other methods' shortcomings. It doesn't grow unwieldy as more links are added to the model, and it handles element-connectivity conditions exactly and automatically. It can also handle distributed parameter elements without discretization, so very flexible robots don't pose any additional challenges. The TMM lends itself to control design because the method outputs Bode plots (that is, magnitude and phase plots of the complex valued transfer functions between system inputs and outputs) very naturally, and it's easy to incorporate feedback.

However, we had two large obstacles to overcome before we could use the TMM to model SAMII: hydraulic actuators and non-collocated feedback.

The TMM models each element in a system with a matrix that transfers a state vector from one end of the element to the other. Each matrix is multiplied by the state vector at the end of the preceding element; the system transfer matrix comes from multiplying the element transfer matrices together. For example, SAMII's open-loop system transfer matrix is given by

$$\mathbf{U}_{sys} = \mathbf{U}_{bs}\,\mathbf{U}_{beam}\,\mathbf{U}_{l0}\,\mathbf{U}_{j1}\,\mathbf{U}_{l1}\,\mathbf{U}_{act}\,\mathbf{U}_{l2}\,\mathbf{U}_{j3}\,\mathbf{U}_{l3-6} \tag{1}$$

where $\mathbf{U}_{bs}$ is the transfer matrix for the basespring, $\mathbf{U}_{beam}$ is the transfer matrix for the beam, and so on for each element in Figure 3's schematic. Because of this approach, we can't use a state from several elements back in the model, which prohibits accurate representation of the physical system. For practical reasons, our vibration suppression scheme is based on accelerometers mounted on the end of the cantilever beam. The accelerometer signal is fed back into the control

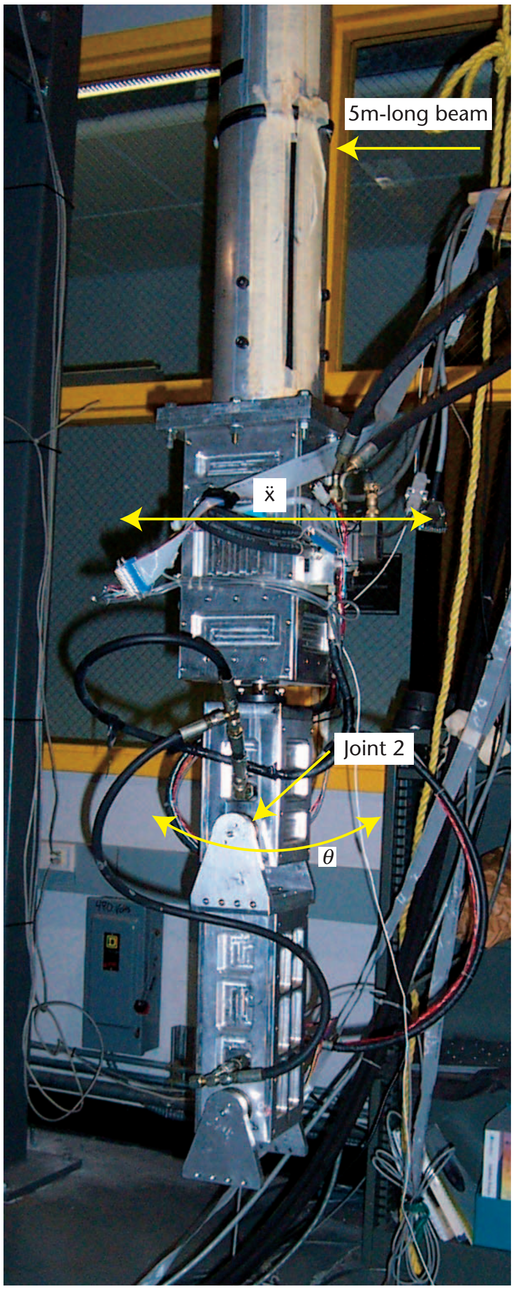

Figure 1. Picture of SAMII. Joint 2's angular position is θ, and $\ddot{x}$ is the acceleration of SAMII's base (that is, the end of the cantilever beam).

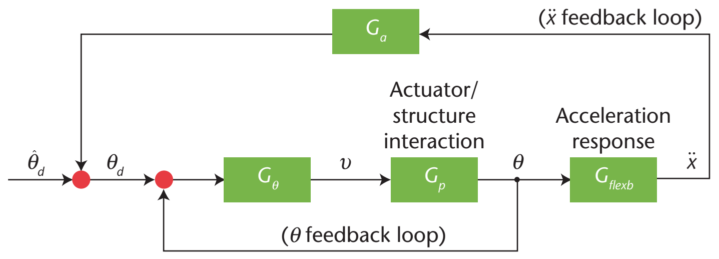

Figure 2. Block diagram. The system requires motion control (θ feedback) and vibration suppression ($\ddot{x}$ feedback).

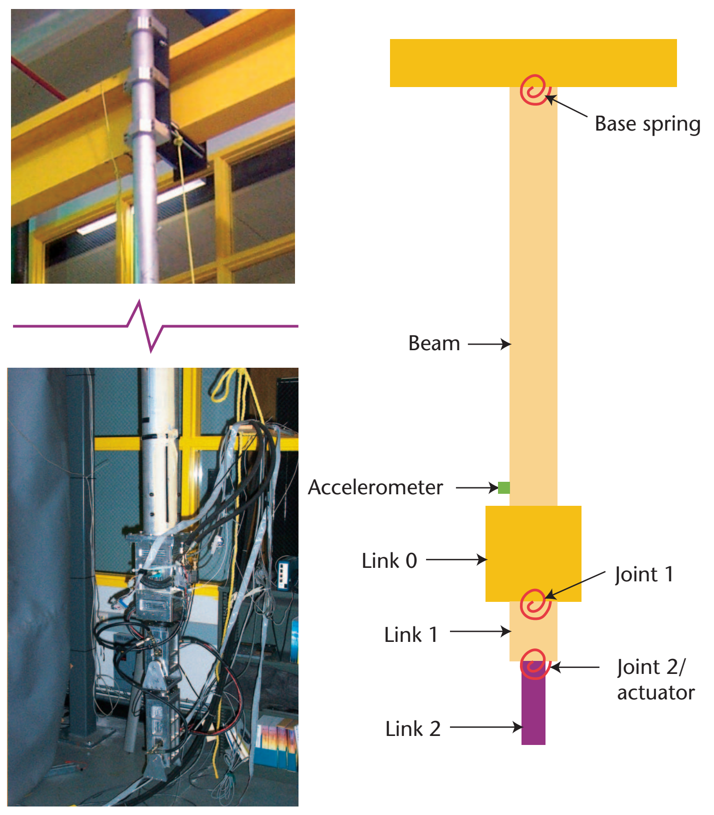

Figure 3. SAMII. Both the picture and schematic illustrate the transfer matrix model.

scheme through the actuator of joint 2, which is several elements away in the TMM model. We developed a transfer matrix for non-collocated feedback based on symbolically inverting a transfer matrix from the sensor location to the actuator location. This transfer matrix enables the TMM to determine the sensor state based on the states at the actuator location (that is, the states that will multiply the actuator transfer matrix). We've experimentally verified these matrices for non-collocated feedback in a model that correctly predicts the system's closed-loop response.

We also developed a transfer matrix model for a hydraulic actuator interacting with the flexible structure. Similarly, we experimentally verified the model and found that it accurately captures the interaction between the actuator and the structure at resonance.

A Python Module for Analyzing Flexible Robots

We created a Python module for objected-oriented analysis of flexible structures via the TMM. The objected-oriented nature of the software leads to clean, clear, and easy-to-maintain code and provides a framework for user extensibility through inheritance. A user can define a new transfer matrix element by deriving from the base class, which clearly defines what properties and methods the new element must have to be valid.

The module uses two primary classes for TMM analysis: `TMMElement` and `TMMSystem`. Examples of `TMMElelment` include flexible links (beam elements), rigid links, torsional springs, hydraulic actuators, and feedback elements (possibly non-collocated); users can develop them by deriving from the base class. The `TMMElement` is the TMM model's primary building block, and it has two primary methods: `GetMat` and `GetHT`. The former takes the frequency variable *s* as an input and returns the element transfer matrix, which the `TMMSystem` then uses to form the system transfer matrix according to Equation 1. `GetHT` returns the homogeneous transformation matrix for the element used in the 3D visualization of the mode shapes.

A `TMMSystem` consists of a list of serially connected `TMMElements` along with a specification of the system boundary conditions and the output signals to be calculated. The `TMMSystem` class has methods for finding a system's natural frequencies and mode shapes, generating Bode plots of specific outputs, system identification, and so on.

Figure 3 shows a picture of SAMII along with a schematic; each element in the schematic is an object in the code from Figure 4.

Each of Lines 2 through 9 in Figure 4 creates a `TMMElement` object from a derived class (`TorsionalSpringDamper4x4`, `samiiBeam`, `samiiLink0`, and so on). Each object models a specific physical piece of the robot by calculating a transfer matrix that captures that piece's dynamics. Line 7 creates an `AVSwThetaFB` element that models a hydraulic actuator under θ feedback, including the interaction between the structure and the actuator. Line 8 creates a `SAMIIAccelFB` element that models the non-collocated accelerometer feedback where the sensor is at the end of the beam and the

```
 1 def accfbsamiimodel(xf,Ga=1):
 2     basespring = TorsionalSpringDamper4x4({'k':xf[0],'c':xf[1]},
         unknownparams=['k','c'])
 3     beam = samiiBeam()
 4     link0 = samiiLink0()
 5     j1spring = TorsionalSpringDamper4x4({'k':xf[2],'c':xf[3]},
         unknownparams = ['k','c'])
 6     link1 = samiiLink1()
 7     clavs = AVSwThetaFB({'Ka':xf[4],'tau':xf[5],'ks':xf[6],'c':xf[7],'
         Gc':180.0/pi},unknownparams=['Ka','tau','ks','c'])
 8     accfb = SAMIIAccelFB(link0,j1spring,link1,clavs,Ga = Ga)
 9     link2 = samiiLink2()
10     bodeout1 = bodeout(input='j2dhat',output='j2a',type='diff',ind=[
         clavs,link1],dof=1)
11     bodeout2 = bodeout(input='j2dhat',output='a1',type='abs',ind=beam,
         post='accel',dof=0,gain=xf[8])
12     return ClampedFreeTMMSystem([basespring,beam,link0,j1spring,link1,
         accfb,clavs,link2],bodeouts=[bodeout1,bodeout2])
```

Figure 4. Python code to create the TMM model of SAMII corresponding to Figure 3.

actuator is at joint 2. Lines 10 and 11 specify the output signals to calculate, and Line 12 returns an object derived from `TMMSystem`, in which the system boundary conditions are specified (clamped-free). The entire system model is created in just 12 lines.

Capabilities

Once we've created a system model, the software provides many capabilities for analysis, system identification, and control design. Examples include finding the system's natural frequencies and mode shapes, automated system identification, Bode analysis with user-defined outputs, and control design and optimization. We can do control design by optimizing multiple Bode plots or optimizing the closed-loop pole locations.

The software also has the ability to find closed-form symbolic expressions for the closed-loop system response by using Python to automatically write an input script to Maxima, which does the symbolic analysis and outputs its results to Fortran files. We can import these Fortran files into Python in two ways: compile them and use `f2py` to connect to the compiled code, or automatically parse the Fortran code into Python modules that we can directly import. All of this can happen without forcing the user to learn Maxima or Fortran—or even knowing that they're being used.

Example Usage and Results

Now that we've created the system model, it's very straightforward to generate Bode plots for the system's open- or closed-loop response. As an example of the software's predictive capabilities, Figures 5 and 6 show Bode plots of the system with both the motion control and vibration suppression loops closed (that is, like the closed-loop system illustrated in Figure 2). We find excellent agreement between model and experiment.

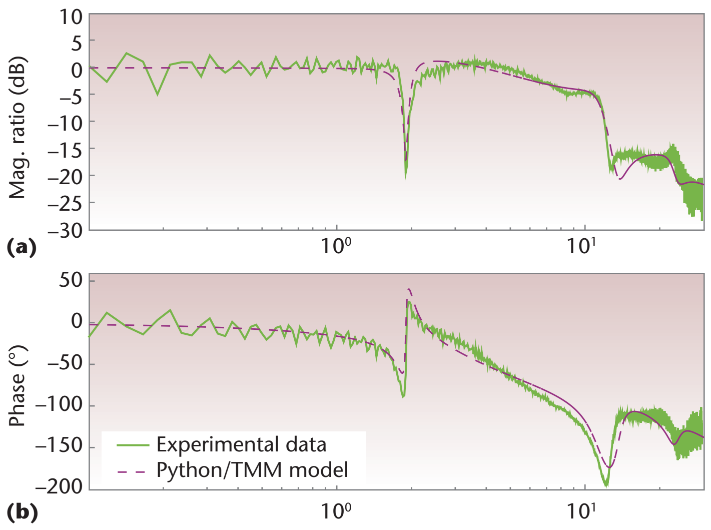

Figure 5. Closed-loop actuator Bode plot $\theta/\hat{\theta}_d$. We're comparing experimental data to the transfer matrix model for the system that has θ and $\ddot{x}$ feedback (vibration suppression).

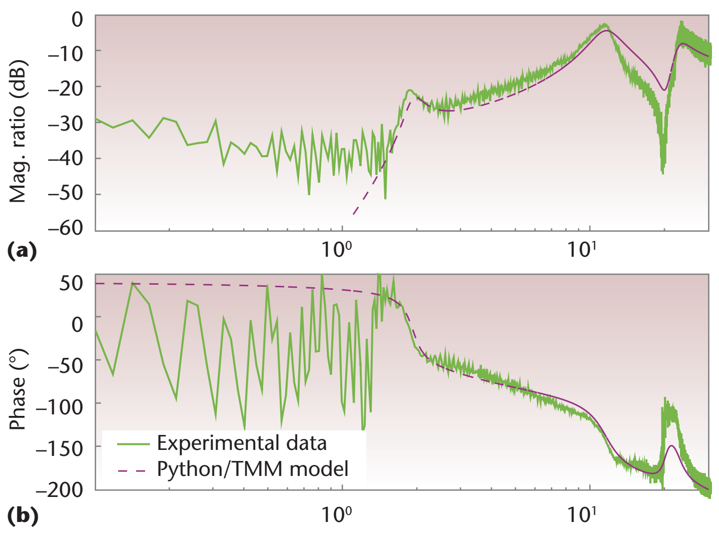

Figure 6. Closed-loop flexible base Bode plot $\ddot{x} / \hat{\theta}_d$. We're comparing experimental data to the transfer matrix model for the system that has θ and $\ddot{x}$ feedback (vibration suppression).

We chose Python for our work for two main reasons: Python makes object-oriented programming easy, and many scientific and engineering modules are available for us to build on. We also found that our coding time was greatly reduced through interactive development with IPython (http://ipython.scipy.org). We plan to continue this work by developing Python modules for simulating interconnected dynamic systems and for rapidly implementing the control schemes on real-time embedded hardware.

References

1. E.C. Pestel and F.A. Leckie, *Matrix Methods in Elastomechanics*, McGraw Hill, 1963.
2. W.J. Book, *Modeling, Design and Control of Flexible Manipulator Arms*, PhD dissertation, Dept. of Mechanical Eng., Massachusetts Inst. of Technology, Apr. 1974.
3. W.J. Book, O. Maizza-Neto, and D.E. Whitney, "Feedback Control of Two Beam, Two Joint Systems with Distributed Flexibility," *J. Dynamic Systems, Measurement and Control*, vol. 97, no. 4, 1975, pp. 424–431.
4. R.W. Krauss, O. Brüls, and W.J. Book, "Two Competing Linear Models for Flexible Robots: Comparison, Experimental Validation, and Refinement," *Proc. 2005 Am. Control Conf.*, IEEE CS Press, 2005, pp. 1963–1968; http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1470257.
5. R.W. Krauss, An Improved Technique for Modeling and Control of Flexible Structures, PhD dissertation, Dept. of Mechanical Eng., Georgia Inst. of Technology, Aug. 2006; http://etd.gatech.edu/theses/available/etd-06202006-185450/.
6. A. Deluca and B. Siciliano, "Closed-Form Dynamic-Model of Planar Multilink Lightweight Robots," IEEE Trans. Systems Man and Cybernetics, vol. 21, no. 4, 1991, pp. 826–839.

Ryan W. Krauss *is an assistant professor in the mechanical engineering department at Southern Illinois University Edwardsville. His research interests include control of flexible structures, applied controls, automotive crashworthiness, and technical computing. Krauss has a PhD in mechanical engineering from the Georgia Institute of Technology. Contact him at rkrauss@siue.edu.*

Wayne J. Book *is the Husco/Ramirez Distinguished Professor in Fluid Power and Motion Control in the George W. Woodruff School of Mechanical Engineering at the Georgia Institute of Technology. His research interests include fluid power, motion control, haptics, and hardware-in-the-loop simulations. Book has a PhD in mechanical engineering from the Massachusetts Institute of Technology. He is a fellow of the American Society of Mechanical Engineers and the IEEE. Contact him at wayne.book@me.gatech.edu.*