# A Python Module for Modeling and Control Design of Flexible Robots

Ryan W. Krauss and Wayne J. Book

September 5, 2006

## 1 Introduction

Flexibility in robot links or joints may come from demands for lightweight or long reach. Flexible robots can provide the benefit of being lighter, faster, and cheaper to actuate than their rigid counterparts. Robots with flexible links pose a considerable challenge because they maybe composed of discrete and distributed parameter elements, many links, complicated actuators, and multiple feedback loops. The example system analyzed in this work poses the additional challenges of being hydraulically actuated and having two feedback loops where the sensors and actuators are not precisely collocated.

Existing modeling approaches for flexible structures are inadequate for designing controllers for flexible robots. The transfer matrix method (TMM) [1, 2] can be an excellent approach that meets the needs of a controls engineer if some theoretical hurdles can be overcome and some new software package makes the approach accessible and user friendly.

This work discusses how the capabilities of the TMM have been expanded and a Python software module has been developed to make the TMM an excellent tool for modeling and control design of practical flexible robots [3].

## 2 System Description

Figure 1 shows a picture of the flexible robot used in the experimental part of this work. The robot is known as 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 be used to give SAMII a large workspace and move it into the general position desired. Once SAMII is in position, the joints of the large robot might be locked. The large robot must have long links to give SAMII a large workspace, but with that length comes inherent flexibility and vibration problems. Modeling approaches and control schemes are needed to deal with that flexibility and suppress or avoid vibrations.

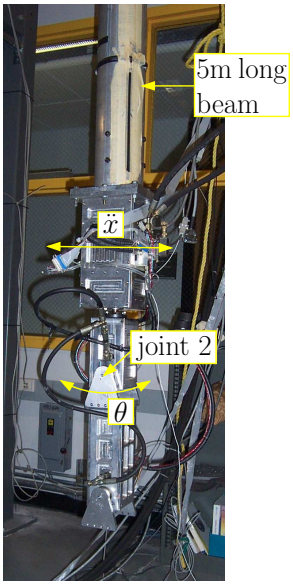


Figure 1: Picture of SAMII showing joint 2 and the two outputs: $\theta$ is the angular position of joint 2 and $\ddot{x}$ is the acceleration of SAMII's base (i.e. the end of the cantilever beam).

# 3 Problem Statement

A block diagram of the control scheme is shown in Figure 2. The control design goal is to develop transfer functions for $G_\theta$ and $G_a$. $G_\theta$ specifies how the system will respond to commands to move to a desired position $\theta_d$ and is referred to as the motion control portion of the control scheme. $G_a$ is the vibration suppression controller. The goal is to find the optimal $G_\theta$ and $G_a$ so that the robot moves quickly while also suppressing vibration.

# 4 The Need for Better Modeling Tools

Many approaches exist for modeling flexible structures. 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. However, it is not the ideal tool for control design of flexible robots. It is difficult to find an FEA software package that can model multiple feedback loops and hydraulic actuators. Even if a suitable package could be found that is capable of modeling the closed-loop response of hydraulically actuated flexible robots, such a model could not be used for control design without modal discretization. This would be a clumsy approach for very flexible robots where the feedback controller is affecting the mode shapes of the system.

The AMM has been applied to robotics [4], however the approach quickly grows unwieldy as the number of links increases. Correctly handling the element-connectivity conditions as more links are added to the model is quite burdensome. It would be very complicated to apply the approach to the robot analyzed in this work. This approach

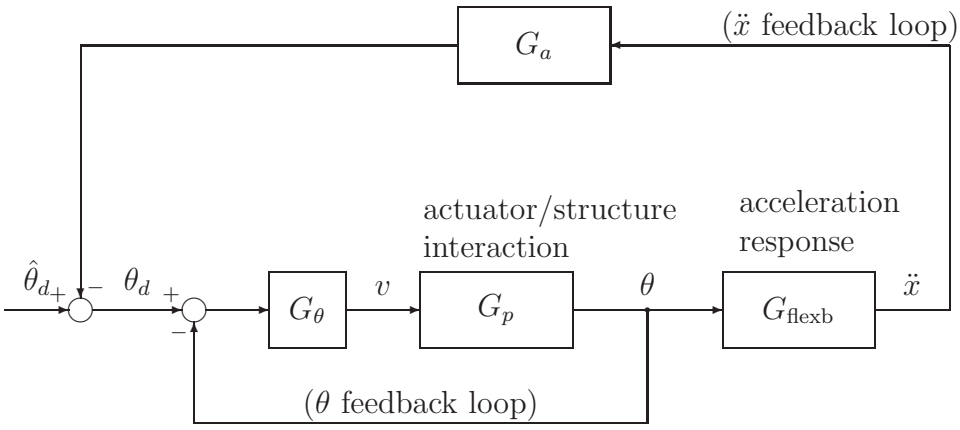


Figure 2: Block diagram of the system with motion control ($\theta$ feedback) and vibration suppression ($\ddot{x}$ feedback).

would also be clumsy for very flexible robots.

# 5 Expanding the Capabilities of the TMM

The TMM has the potential to overcome the shortcomings of other methods. It does not grow unwieldy as more links are added to the model and element-connectivity conditions are handled exactly and automatically. It can handle distributed parameter elements without discretization, so very flexible robots do not pose any additional challenges. The TMM lends itself to control design because the method outputs Bode plots very naturally and it is easy to incorporate feedback.

There were several hurdles that had to be overcome before the TMM could be used to model SAMII. The two biggest obstacles were hydraulic actuators and non-collocated feedback.

The TMM models each element in a system by 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

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

where $\mathbf{U}_{\text{bs}}$ is the transfer matrix for the basespring, $\mathbf{U}_{\text{beam}}$ is the transfer matrix for the beam, and so on for each element in the schematic of Figure 3. Because of this approach, there is no means for using a state from several elements back in the model. This mathematical limitation of the method prohibits accurate representation of the physical system. For practical reasons, the vibration suppression scheme is based on accelerometers mounted

on the end of the cantilever beam. The accelerometer signal is feedback into the control scheme through the actuator of joint 2 which is several elements away in the TMM model. A transfer matrix for non-collocated feedback was developed based on symbolically inverting a transfer matrix from the sensor location to the actuator location. This allows the sensor state to be determined based on the states at the actuator location (i.e. the states that will multiply the actuator transfer matrix). These matrices for non-collocated feedback have been experimentally verified in a model that correctly predicts the closed-loop response of the system.

A transfer matrix model was also developed for a hydraulic actuator interacting with the flexible structure. This model was also experimentally verified and accurately captures the interaction between the actuator and the structure at resonance.

# 6 A Python Module for Analyzing Flexible Robots

## 6.1 An Object-Oriented Model

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

A picture of the system is shown along with a schematic in Figure 3. Each element in the schematic is an object in the code. The entire system model is created in just 12 lines:

```
1  def olsamiimodel():
2      basespring=TorsionalSpringDamper4x4({'k':166358.0,'c':468.789},
           symlabel='base', unknownparams=['k','c'])
3      beam=samiiBeam()
4      link0=samiiLink0()
5      j1spring=TorsionalSpringDamper4x4({'k':4028.28,'c': 6.3058},
           symlabel='j1', unknownparams=['k','c'])
6      link1=samiiLink1()
7      avs=AngularVelocitySource4x4({'K':0.435489,'tau':173.833},
           symlabel='act', unknownparams=['K','tau'])
8      j2spring=TorsionalSpringDamper4x4({'k':1900.49,'c':21.6805},
           symlabel='j2', unknownparams='all')
9      bodeout1=bodeout(input='j2v', output='a1', type='abs', ind=beam,
           post='accel', dof=0, gain=0.35, gainknown=False)
10     bodeout2=bodeout(input='j2v', output='j2a', type='diff', ind=[
           j2spring, link1], post='', dof=1, gain=180.0/pi)
11     link2=samiiLink2()
12     return ClampedFreeTMMSystem([basespring, beam, link0, j1spring,
           link1, avs, j2spring, link2], bodeouts=[bodeout1, bodeout2])
```

A system model consists of a list of elements ([basespring, beam, ..., link2]) that are

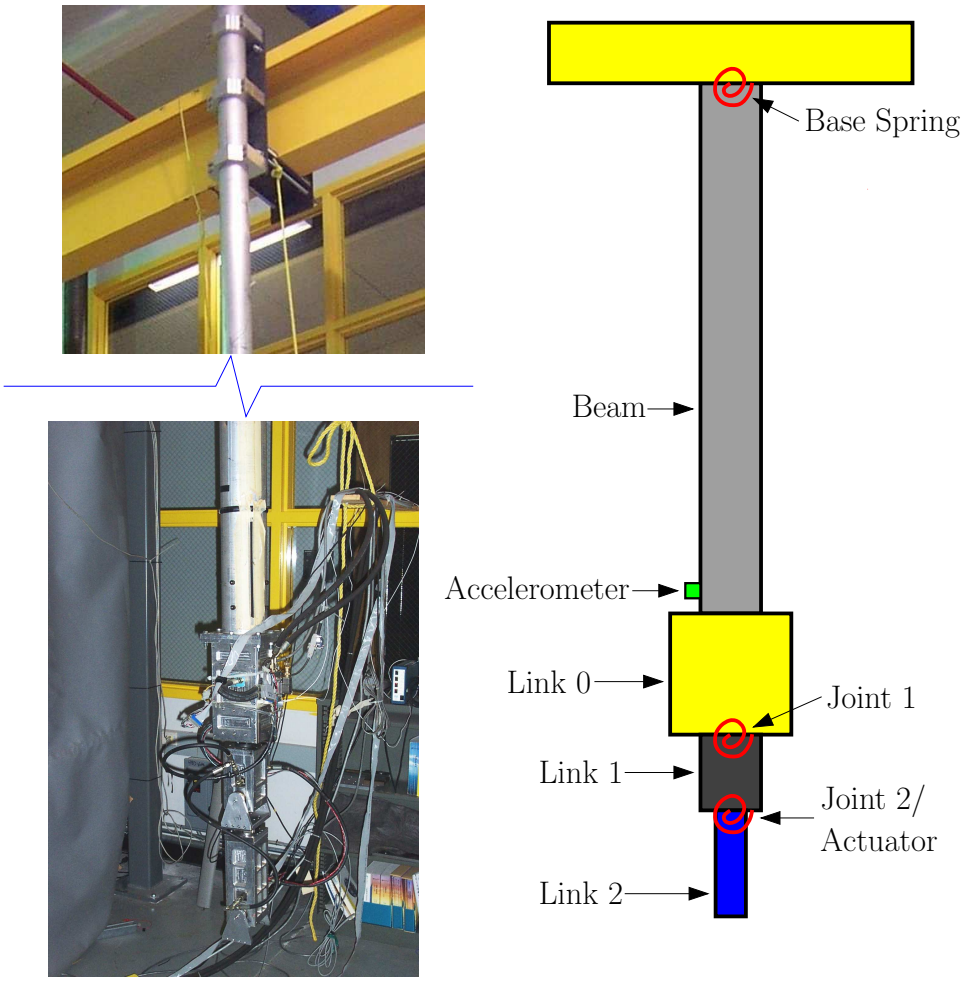


Figure 3: Picture and schematic of SAMII used to illustrate the TMM model.

connected serially along with a specification of the system boundary conditions and the output signals that should be calculated ([bodeout1, bodeout2]). The elements may be flexible links (beam elements), rigid links, torsional springs, hydraulic actuators, and feedback elements (possibly non-collocated). Users can develop new elements by deriving their new element from the base element class.

## 6.2 Capabilities

Once a system model has been created, the software provides many capabilities for analysis, system identification, and control design. Examples include finding the natural frequencies and mode shapes of the system (including visualizing three dimensional mode shapes), automated system identification, Bode analysis with user defined outputs, and two approaches to control design and optimization. Control design can be done by simultaneously optimiz-

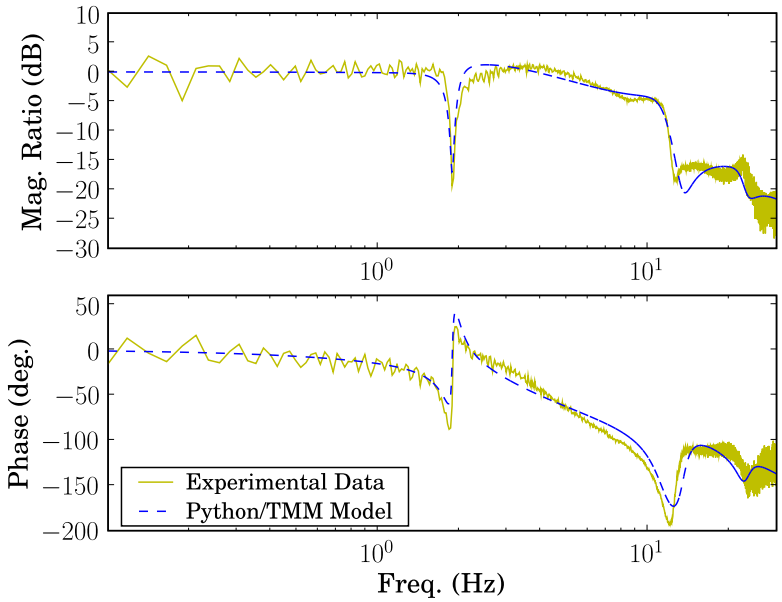


Figure 4: Closed-loop actuator Bode plot $\theta/\hat{\theta}_d$ for the system with $\theta$ and $\ddot{x}$ feedback (vibration suppression) comparing experimental data to the transfer matrix model and a transfer function model.

ing multiple Bode plots or by optimizing the closed-loop pole locations.

The software also has the ability to find closed-form symbolic expression for the closed-loop system response. This is done by using Python to automatically write an input script to Maxima. Maxima does the symbolic analysis and outputs its results to FORTRAN files. These FORTRAN files can be imported into Python in two ways: one option is to compile them and use f2py to connect to the compiled code; the second option is to automatically parse the FORTRAN code into Python modules that can be directly imported. All of this is done in such a way that the user does not need to learn Maxima of FORTRAN, or even know that they are being used.

## 6.3 Example Usage and Results

Once the system model as been created, generating Bode plots for the open or closed-loop response of the system is very straightforward. As an example of the predictive capabilities of the software, Figures 4 and 5 show Bode plots of the system with both the motion control and vibration suppression loops closed (i.e. the closed-loop system illustrated by the block diagram of Figure 2). There is excellent agreement between model and experiment.

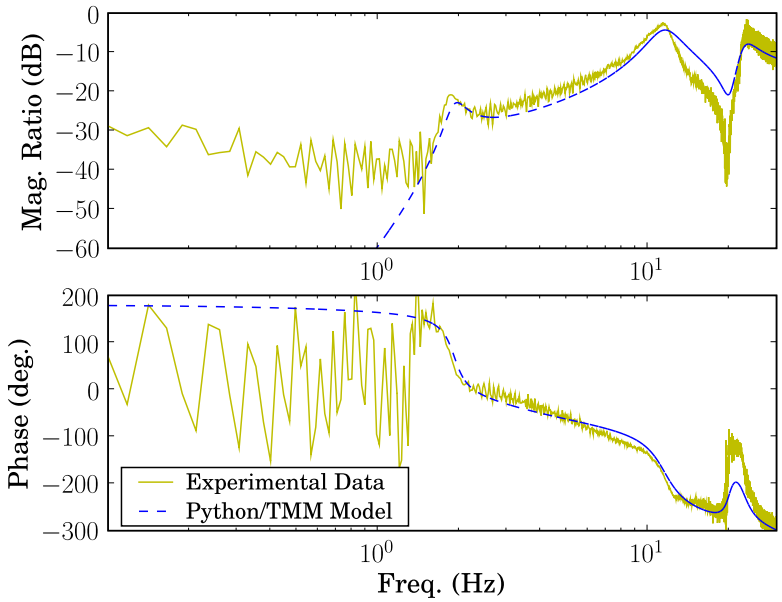


Figure 5: Closed-loop flexible base Bode plot $\ddot{x}/\hat{\theta}_d$ for the system with $\theta$ and $\ddot{x}$ feedback (vibration suppression) comparing experimental data to the transfer matrix model and a transfer function model.

### 6.4 Why Python?

There are many scientific and engineering modules available for Python. These modules make it fairly easy to give substantial technical capabilities to Python objects. This work uses NumPy for linear algebra capabilities, SciPy for optimization, f2py to speed up the optimizations with FORTRAN, Matplotlib for 2D plotting, and Mayavi for 3D visualization. Coding time is greatly reduced through interactive development with IPython.

The Python language is very easy to learn [5], even for a mechanical engineer with little formal computer training. Python syntax is very clean and readable. Python code is naturally structured into modules that keep code organized and make it easy to reuse or redistribute.

## 7 Conclusion

This work has expanded the capabilities of the TMM so that it can be applied to practical flexible robots. The primary theoretical expansions include the ability to model non-collocated feedback and hydraulic actuators. This work has also resulted in the creation of a Python module for modeling and control design of flexible robots. This module makes the power of the TMM accessible to engineers who are not experts in the TMM. The software includes the ability to model serial connections of flexible and rigid links and joints along

with various actuators. The software also includes built-in system identification capabilities, modal analysis tools, and two approaches to control design and optimization.

# References


[1] Eduard C. Pestel and Frederick A. Leckie. *Matrix Methods in Elastomechanics*. McGraw Hill, New York, 1963.

[2] Wayne J. Book. *Modeling, Design and Control of Flexible Manipulator Arms*. PhD thesis, Massachusetts Institute of Technology, Apr. 1974.

[3] Ryan W. Krauss. *An Improved Technique for Modeling and Control of Flexible Structures*. PhD thesis, Georgia Institute of Technology, Aug. 2006.

[4] A. Deluca and B. Siciliano. Closed-form dynamic-model of planar multilink lightweight robots. *IEEE Transactions on Systems Man and Cybernetics*, 21(4):826–839, Jul-Aug 1991.

[5] Mark Lutz and David Ascher. *Learning Python*. O'Reilly, Sebastopol, CA, second edition, 2003.