Paper AAS 01-176

Web-Based Tools for Spacecraft Attitude Dynamics and Control Simulation

Daniel Quock and **Jordi Puig-Suari**

Aerospace Engineering Department
California Polytechnic State University

11th AAS/AIAA Space Flight Mechanics Meeting

Santa Barbara, CA **11-15 February 2001**

AAS Publications Office, P.O. Box 28130, San Diego, CA 92129

AAS 01-176

WEB-BASED TOOLS FOR SPACECRAFT ATTITUDE DYNAMICS AND CONTROL SIMULATION

Daniel Quock[*] and Jordi Puig-Suari[†]

The web-based application presented in this paper provides attitude dynamics and control simulation tools utilizing the advantages of the World Wide Web. A user can access the program from the office, classroom, home, or local Internet cafe without the hassle of single-machine licenses. The program features an easy-to-use interface, allowing the user to quickly alter spacecraft characteristics, orbit parameters, and initial conditions. Different levels of complexity, ranging from rigid-body motion to active control of a spacecraft are available to the user. A number of output options are also available to the user, ranging from simple graphs to three-dimensional movies.

INTRODUCTION

The Internet and the World Wide Web have become an integral part of our society. Recently more and more web tools have become available to the public, ranging from multimedia to online games. Some experts predict that, in the near future, web-based application programs will become the primary means to access computer software. Many such applications are already available to users through the web, providing people access to word processors, spreadsheets and data storage from anywhere around the world.

The work presented here investigates the feasibility of using a Java-written web program, an applet, for mathematically intensive calculations. The primary concerns are that the program must be both small enough to download quickly and fast enough to solve the differential equations in reasonable time. Applets can run on any Java-capable device. Given Java's increasing popularity and continued growth, soon Palm pilots or cell phones might have the ability to download and run Java programs, giving a user the ability to run these simulations from any location without even needing a computer.

The software presents unique opportunities to the application in education. Professors can allow students access as a check to ensure that their simulations are

[*] Graduate Student, Aerospace Engineering Department, California Polytechnic State University, San Luis Obispo, CA 93407; Student member AIAA
[†] Associate Professor, Aerospace Engineering Department, California Polytechnic State University, San Luis Obispo, CA 93407; (805) 756-6479 (voice), (805) 756-2376 (fax); jpuigsua@calpoly.edu; Member AIAA

correct. In addition, the 3D animation feature provides the ability to easily visualize the vehicle's motion. This visual representation is a great aid in understanding spacecraft dynamics.

SOFTWARE DEVELOPMENT

Programming for the Web

Though Java is a relatively new computer language, it quickly sparked great interest in the computer world as the World Wide Web became more and more prominent in the early 1990's.[1] Java's main feature is its ability to run on multiple computer platforms. Much of Java's appeal comes from the fact that special Java-based programs, called Java applets, can run in web browsers. A user sitting at home on a Macintosh computer and a student on an IBM machine at school can both run a program developed on a UNIX workstation without the need for any software beyond their web browser. Java applets also help make web pages more attractive, animated, and interactive. In addition, Sun Microsystems, the developer of Java, makes Java development tools free, allowing users to download a Java compiler and development toolkits from their website.[2]

This portability means that users can run programs directly from their computer at any location with a connection to the Internet. Note that users download the applet onto their computer and run it locally. This means that all processes are run on the user's machine and do not slow the server down. By running the application locally, users also have control over the accuracy of the results, spending their personal computational resources as needed. Having the users download the program with each access to the web page also allows the software developer to ensure that the most recent version of the software is being used. In addition, there is no need for users to install other programs like MatLab or Satellite Tool Kit, which require licenses and are expensive. The convenience to users is very unique.

Java has the capability to include different Application Programming Interfaces, or APIs. These APIs are essentially a set of pre-constructed routines and functions. The program presented here uses the Java 3D API. Java 3D lets the programmer create a three-dimensional environment and easily manipulate the rotation and translation of the objects inside it.

The web-based tools presented here are the initial developments of a program for simulating spacecraft attitude dynamics and control. The program lets a user choose from a number of different scenarios and accepts the appropriate input. The program then provides graphs of the output. The Java 3D API also provides the groundwork for an animation of the satellite's motion. This 3D visualization is invaluable as it provides a very clear description of how the satellite rotates.

The program is highly modular making it easy to add more scenarios with a minimum of effort. Each new scenario merely requires the addition of the appropriate input/output fields and the new equations of motion.

Numerical Integrator

In order to solve the differential equations of motion, a numerical integration algorithm had to be implemented in Java. The numerical integrator is based on the fourth-order Runge-Kutta-Fehlberg method. This algorithm uses six evaluations for each time step and provides an estimated error at each step. However, instead of using Fehlberg's constants, the program uses parameters developed by Cash-Karp which are found to be more efficient[3]. The integration routine includes a time varying step algorithm. If the error of the numerically integrated step is larger than the user supplied error tolerance, the program can decrease the time step and re-run the integration. The algorithm can also increase the time step to speed up the integration.

Satellite Animations

The Java 3D API provides the groundwork to create animations of the spacecraft's motion. The animation window, shown in Figure 1, creates a box-shaped representation of the satellite. Originally, the satellite's shape was to be based on the ellipsoid of inertia but was discarded because the rotation of smooth surfaces is more difficult to visualize. The lengths of the box are determined using the moments of inertia equations (Eqs. 1 - 3) for a rectangular prism:

$$I_{xx} = \frac{1}{12} m \left(b^2 + c^2 \right) \tag{1}$$

$$I_{yy} = \frac{1}{12} m \left(a^2 + c^2 \right) \tag{2}$$

$$I_{zz} = \frac{1}{12} m \left(a^2 + b^2 \right) \tag{3}$$

The m/12 is dropped from each equation, which results in three equations and three unknowns. The values a, b and c are found and normalized to a length appropriate to fit in the animation window.

The body and inertia axes are color-coordinated to aid in the visualization of the orientation of the satellite. The user can use the mouse buttons to rotate, translate and zoom the scene. The reset axes button returns the orientation and zoom level to their original values. Additional buttons and a slider give the user control to start and stop the animation, alter the animation speed, and move to a desired time step. A counter displays the time for each frame.

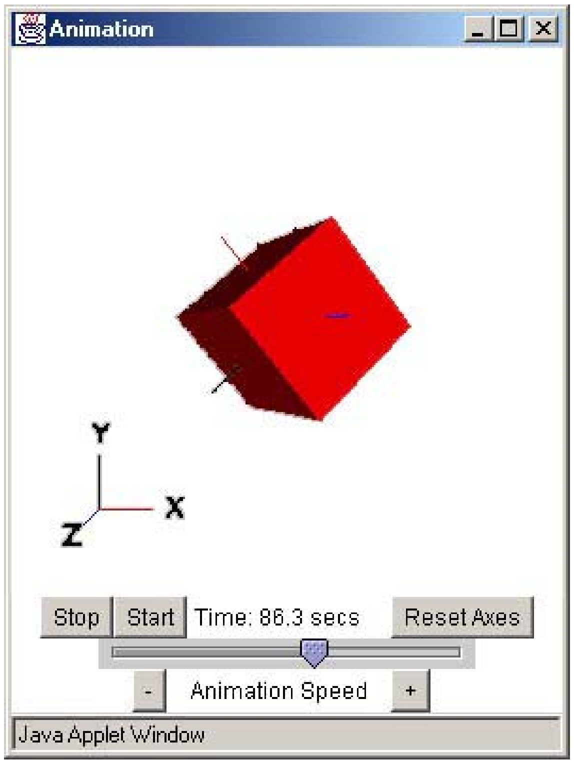


Figure 1 Satellite Animation Window

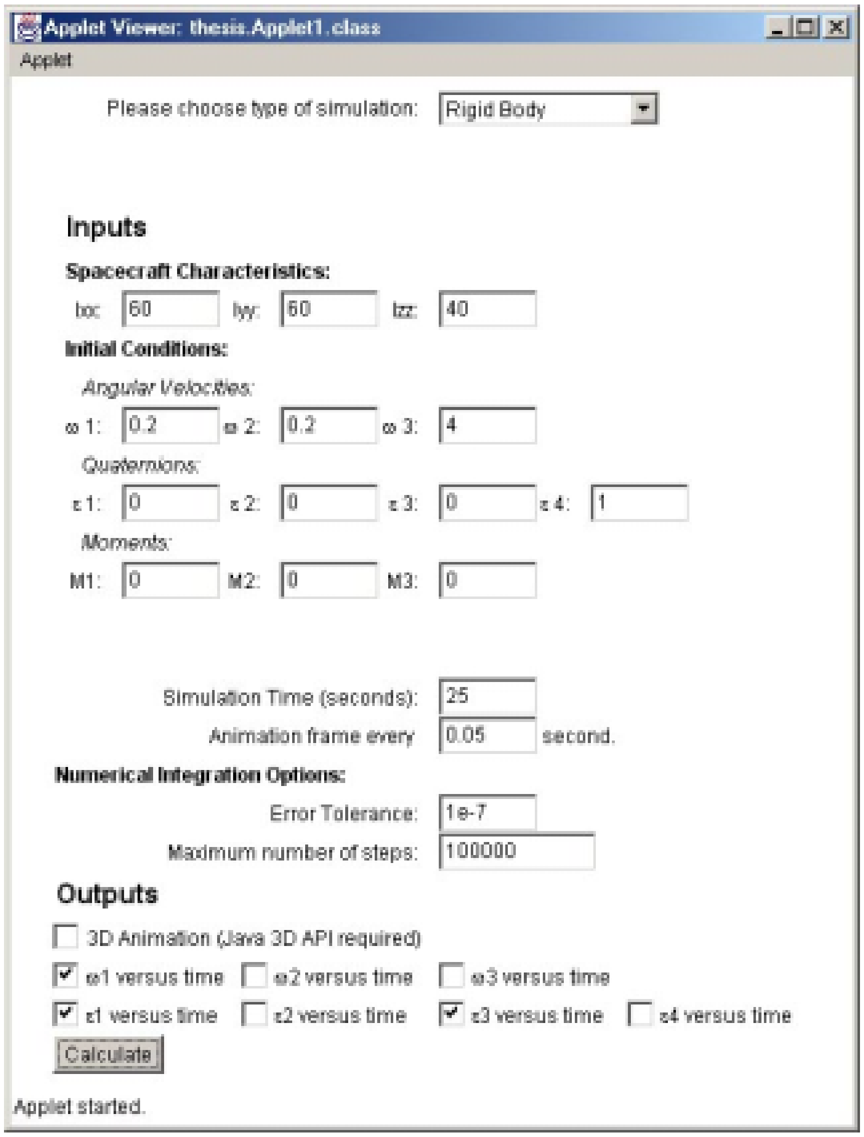
Running the Applet

The applet is accessed through a java-capable web browser. The user goes to the appropriate website and enters (if necessary) a password. The web page loads with the applet in the browser window. The first step is to select a scenario from a pull-down menu. The user is then prompted for required inputs and is presented with a choice of possible outputs, graphs and animation, as shown in Figure 2. When the user is satisfied with the input values and numerical constraints, clicking on the calculate button executes the applet. After the simulation is completed, the requested outputs appear in additional windows.

SAMPLE SCENARIOS

Rigid Body

One of the simplest simulations is a rigid-body spacecraft with constant body-fixed torques. The rigid body equations of motion implemented in principal axes are well known[4]:



Please choose type of simulation: Rigid Body

Inputs

Spacecraft Characteristics:

Ixx:	60	Iyy:	60	Izz:	40

Initial Conditions:

Angular Velocities:

ω 1:	0.2	ω 2:	0.2	ω 3:	4

Quaternions:

ε 1:	0	ε 2:	0	ε 3:	0	ε 4:	1

Moments:

M1:	0	M2:	0	M3:	0

Simulation Time (seconds): 25

Animation frame every 0.05 second.

Numerical Integration Options:

Error Tolerance: 1e-7

Maximum number of steps: 100000

Outputs

[ ] 3D Animation (Java 3D API required)

[✓] ω1 versus time [ ] ω2 versus time [ ] ω3 versus time

[✓] ε1 versus time [ ] ε2 versus time [✓] ε3 versus time [ ] ε4 versus time

Figure 2 Rigid Body Scenario Interface

$$\frac{d\omega_1}{dt} = \frac{\left(I_{yy} - I_{zz}\right)\omega_2\omega_3 + M_1}{I_{xx}} \tag{4}$$

$$\frac{d\omega_2}{dt} = \frac{\left(I_{zz} - I_{xx}\right)\omega_1\omega_3 + M_2}{I_{yy}} \tag{5}$$

$$\frac{d\omega_3}{dt} = \frac{\left(I_{xx} - I_{yy}\right)\omega_1\omega_2 + M_3}{I_{zz}} \tag{6}$$

where the ω's are the angular velocities, the I's the principal moments of inertia, and the M's the body-fixed torques.

Quaternions, ε, are chosen as kinematic variables due to the lack of singularities. The differential equations for the quaternions are:

$$\frac{d\varepsilon_1}{dt} = -0.5\left(-\omega_3\varepsilon_2 + \omega_2\varepsilon_3 - \omega_1\varepsilon_4\right) \tag{7}$$

$$\frac{d\varepsilon_2}{dt} = -0.5\left(\omega_3\varepsilon_1 - \omega_1\varepsilon_3 - \omega_2\varepsilon_4\right) \tag{8}$$

$$\frac{d\varepsilon_3}{dt} = -0.5\left(-\omega_2\varepsilon_1 + \omega_1\varepsilon_2 - \omega_3\varepsilon_4\right) \tag{9}$$

$$\frac{d\varepsilon_4}{dt} = -0.5\left(\omega_1\varepsilon_1 + \omega_2\varepsilon_2 + \omega_3\varepsilon_3\right) \tag{10}$$

The applet interface for the rigid body simulation (as shown in Figure 2) requires the input of the spacecraft principal moments of inertia and initial conditions (ω's and ε's). The user also inputs the simulation time, animation frame time steps, error tolerance and the maximum number of steps allowed in the numerical simulation. If the maximum number of steps is exceeded, the program notifies the user, stops the simulation, and produces results to that point. If desired, the user can increase the number of steps and re-run the program. Finally, the user can select the desired outputs. Graphs for the angular velocities and quaternions are available. The user can also choose to display a 3D animation. Note that creating the animation requires the Java 3D API plug-in and significantly increases computation time.

Sample output graphs for the simulation inputs in Figure 2 are shown in Figures 3-4. These graphs appear as new windows on the user's screen and can be easily printed.

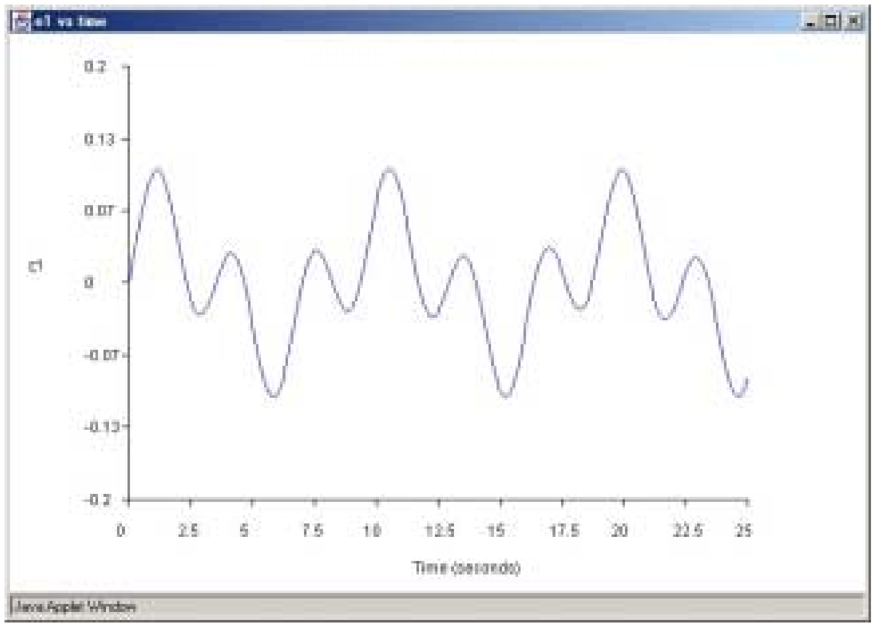


Figure 3 Rigid Body - 1st Quaternion Value

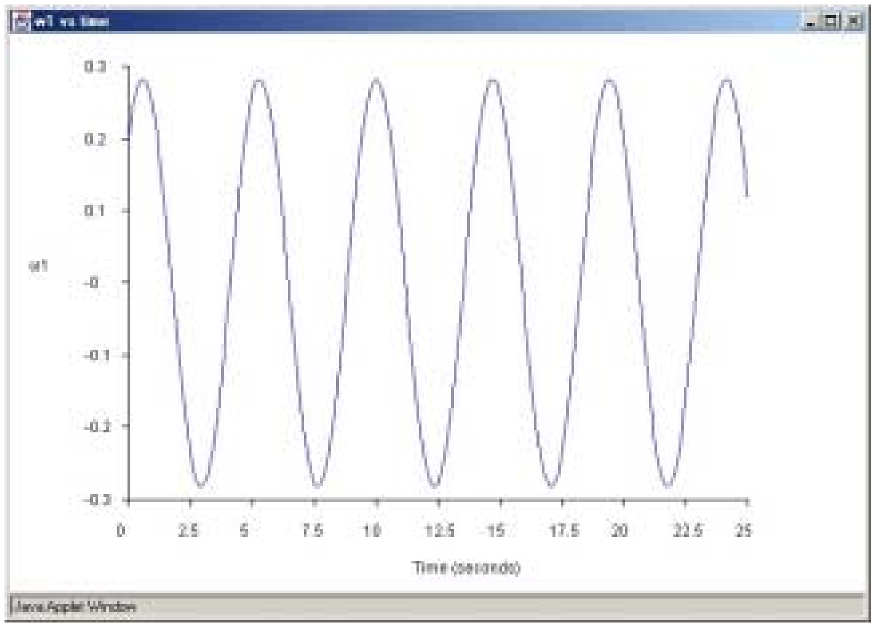


Figure 4 Rigid Body - Angular Velocity About X-Axis

If the user is short on memory or time, then they can reduce the number of steps taken in the numerical integration by lowering the error tolerance. A sample output of the angular velocity in the x-body frame of the same simulations shows the solution if an error greater than $1\text{x}10^{-3}$ is acceptable (see Figure 5).

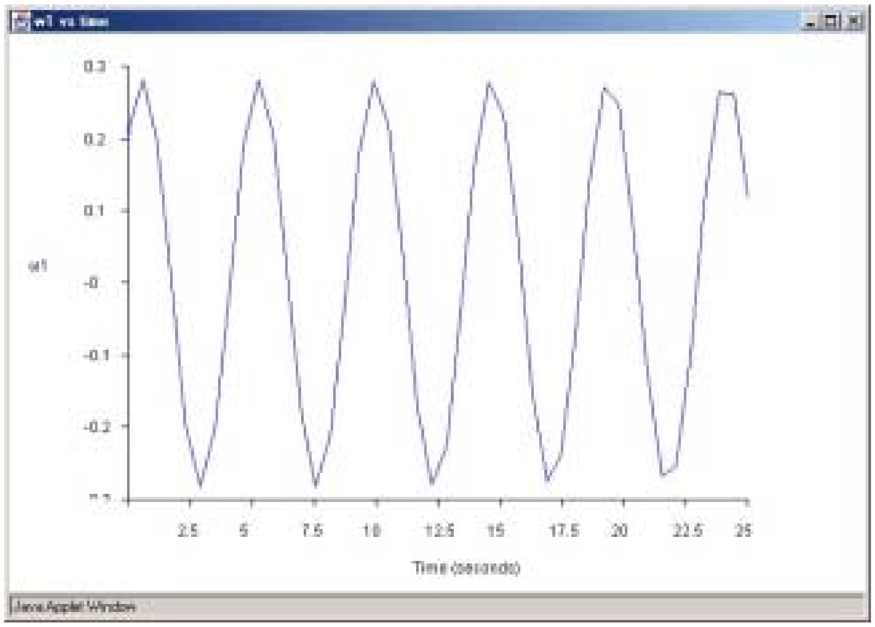


Figure 5 Rigid Body Angular Velocity About X-Axis with 1x10^{-3} Error Tolerance

Active Control

A more complex simulation is the active control of a satellite's attitude. This scenario involves three reaction wheels with inertias J_a, J_b, and J_c aligned with each of the principal axes of the satellite. (The program can be easily modified to include four wheel configurations.) The reaction wheels are used to orient the satellite from an initial quaternion vector set to a commanded (target) quaternion vector. The feedback law is given by[5]:

$$\mathbf{M} = k\mathbf{I}\,\boldsymbol{\varepsilon}_\mathbf{e} + \mathbf{C}\omega \qquad (11)$$

where **M** is the input torque vector generated by the reaction wheels, k is the proportional feedback gain, **I** is the identity matrix, **C** is a diagonal matrix with three derivative gains**,** and $\boldsymbol{\varepsilon}_\mathbf{e}$ is the vector of quaternion errors given by:

$$\varepsilon_{1e} = \varepsilon_{4c}\varepsilon_1 + \varepsilon_{3c}\varepsilon_2 - \varepsilon_{2c}\varepsilon_3 - \varepsilon_{1c}\varepsilon_4 \qquad (12)$$

$$\varepsilon_{2e} = -\varepsilon_{3c}\varepsilon_1 + \varepsilon_{4c}\varepsilon_2 - \varepsilon_{1c}\varepsilon_3 - \varepsilon_{2c}\varepsilon_4 \qquad (13)$$

$$\varepsilon_{3e} = \varepsilon_{2c}\varepsilon_1 - \varepsilon_{1c}\varepsilon_2 + \varepsilon_{4c}\varepsilon_3 - \varepsilon_{3c}\varepsilon_4 \qquad (14)$$

where ε_c are the quaternions for the target orientation.

The equations of motion for the angular velocities of the reaction wheels are also computed and are given by:

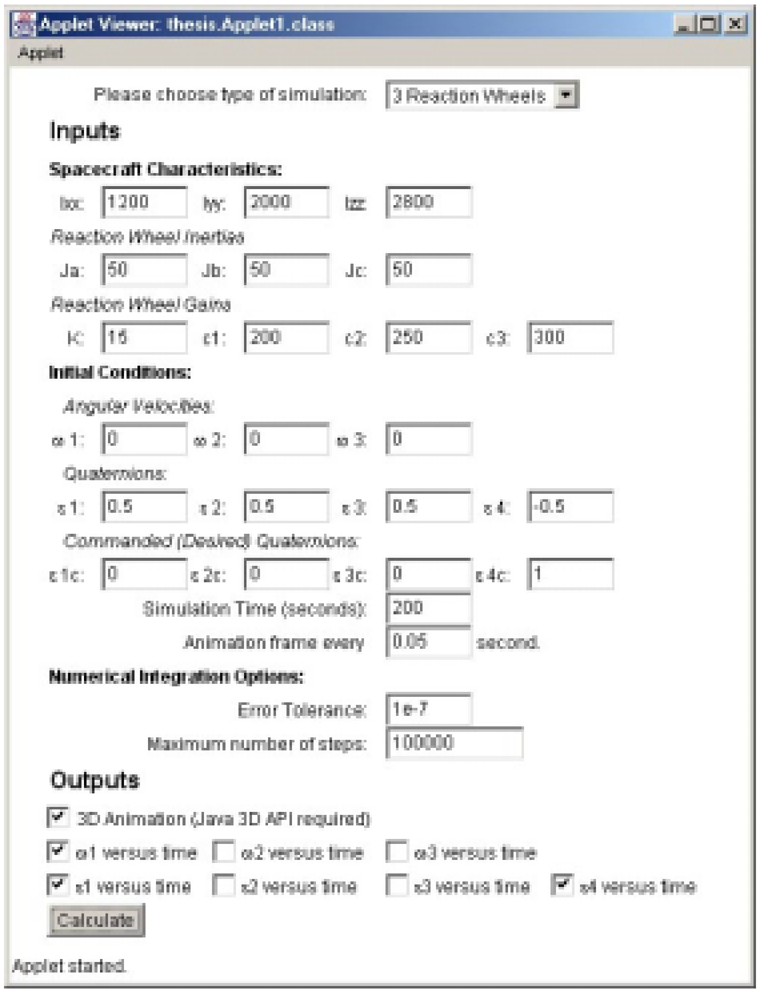
$$\frac{d\omega_a}{dt} = \frac{-\left(M_1 - J_b\omega_3\,\omega_b + J_b\omega_2\,\omega_c\right)}{J_a} \tag{15}$$

$$\frac{d\omega_b}{dt} = \frac{-\left(M_2 - J_a\omega_3\,\omega_a + J_c\omega_1\,\omega_c\right)}{J_b} \tag{16}$$

$$\frac{d\omega_c}{dt} = \frac{-\left(M_3 - J_a\omega_2\,\omega_a + J_b\omega_1\,\omega_b\right)}{J_c} \tag{17}$$

The equations of motion for the angular velocities and quaternions are the same used in the rigid body scenario. (Eqs. 4 – 10)

Since this scenario requires more inputs, the interface for this scenario includes the reaction wheel inertias, reaction wheel gain, derivative gains, and the commanded quaternions as shown in Figure 6.



Please choose type of simulation: 3 Reaction Wheels

Inputs

Spacecraft Characteristics:

Ixx: 1200 Iyy: 2000 Izz: 2800

Reaction Wheel Inertias

Ja: 50 Jb: 50 Jc: 50

Reaction Wheel Gains

K: 16 c1: 200 c2: 250 c3: 300

Initial Conditions:

Angular Velocities:

ω 1: 0 ω 2: 0 ω 3: 0

Quaternions:

ε 1: 0.5 ε 2: 0.5 ε 3: 0.5 ε 4: -0.5

Commanded (Desired) Quaternions:

ε 1c: 0 ε 2c: 0 ε 3c: 0 ε 4c: 1

Simulation Time (seconds): 200

Animation frame every 0.05 second.

Numerical Integration Options:

Error Tolerance: 1e-7

Maximum number of steps: 100000

Outputs

[✓] 3D Animation (Java 3D API required)

[✓] ω1 versus time [ ] ω2 versus time [ ] ω3 versus time

[✓] ε1 versus time [ ] ε2 versus time [ ] ε3 versus time [✓] ε4 versus time

Applet started.

Figure 6 Three Reaction Wheel Scenario Interface

Sample output graphs from the scenario in Figure 6 are shown in Figures 7 and 8. Note that the angular velocity shown in Figure 7 has the satellite starting and ending at rest. Modifications to the program can allow for final, user-defined, angular velocities.

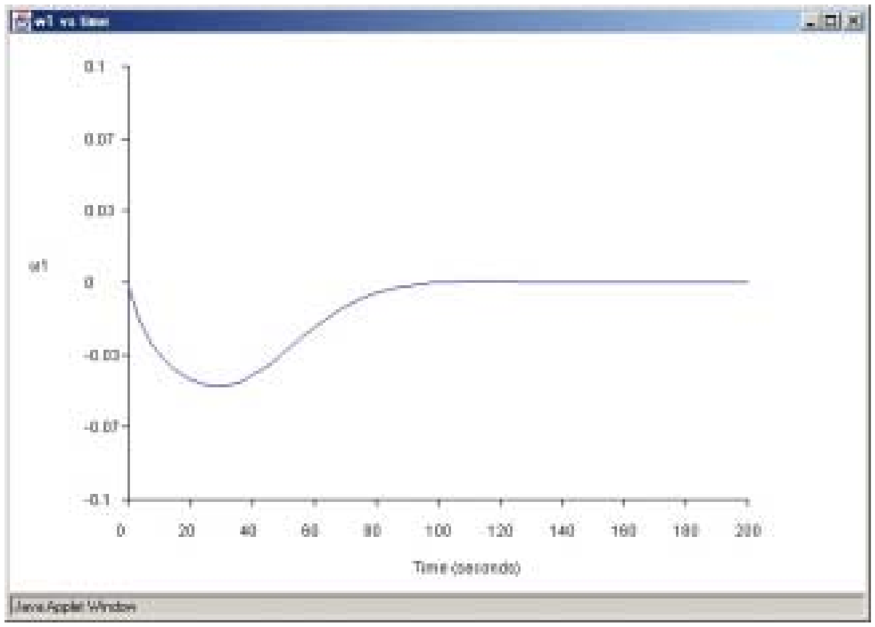


Figure 7 3-Reaction Wheel Active Control - Angular Velocity About X-Axis

The sample quaternion graphs in Figure 8 shows how ε_1 starts from 0.5 to the commanded value 0.

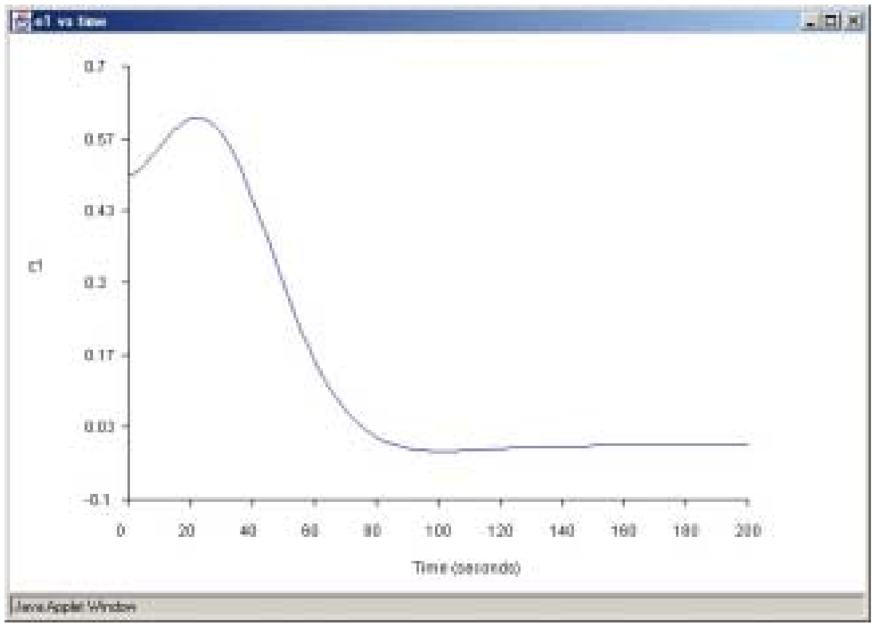


Figure 8 3-Reaction Wheel Active Control - 1st Quaternion Value

Figures 9 and 10 show the screen captures of the 3D animation at the beginning and the end of the scenario.

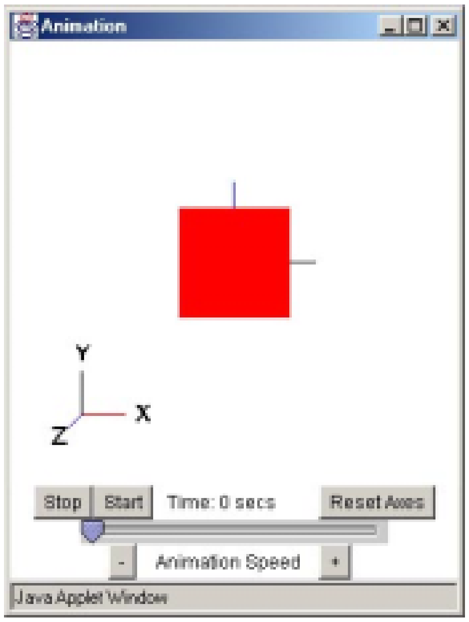


Figure 9 3-RW Active Control – Sat Orientation at t = 0 seconds

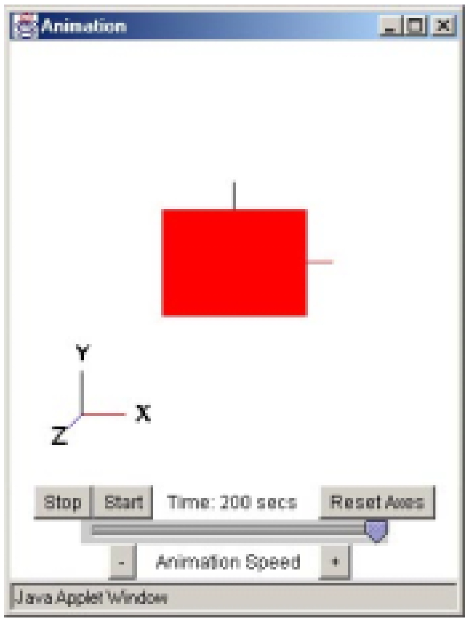


Figure 10 3-RW Active Control – Sat Orientation at t = 200 seconds

CONCLUSIONS AND FUTURE DEVELOPMENT

The purpose of the work presented here is to provide a convenient web-based satellite attitude simulation tool which takes advantage of Java features, including the ability for a user to access the applet from virtually any location with minimal to no software installation. Even the required Java 3D API is expected to become a standard feature of web browsers in the near future. The applet's current size, with the 2 scenarios shown here, is less than 30 kilobytes, which downloads in less than 15 seconds on a 56k modem. The applet also runs quickly: requiring less than 10 seconds to output the graphs and construct the animation on a Celeron 800 MHz machine.

Now that the feasibility of a Java applet for spacecraft simulation has been verified, more scenarios can be added. Additional output options can be included for all scenarios, such as the option to superimpose sets of graphs into one plot or the option to output reaction wheel velocities in the attitude control scenario.

This software presents a unique opportunity in education. Students can access the simulation from any computer with Internet access and are not required to purchase any software. Professors can use the applet as a supplement for assignments, allowing students to check their results or run additional scenarios not covered in class. In addition, the Java 3D based animation provides a great visualization tool to assist with the understanding of spacecraft dynamics.

ACKNOWLEDGMENT

This work is funded by the CSU/Lockheed Martin Grant program, (Grant #50920) under the direction of John Nelson and Jeff Fisher. The authors would like to express their sincere appreciation to both of them for helping make this project become a reality.

REFERENCES

1. Patrick Niemeyer and Joshua Peck, *Exploring Java, 2nd ed.*, O’Reilly & Associates, Inc., Sebastopol, 1996, pp. 1-4.

2. “java.sun.com – The Source for Java™ Technology”, January 2001, <http://java.sun.com>.

3. William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical Recipes in C. The Art of Scientific Computing, 2nd ed.*, Cambridge University Press, New York, 1992, pp.714-717.

4. William Tyrrell Thomson, *Introduction to Space Dynamics*, Dover Publications, Inc., New York, 1986, pp. 195.

5. Bong Wie, *Space Vehicle Dynamics and Control*, AIAA Education Series, Reston, 1998, pp. 436-444.