Open-Source, Extensible Spacecraft Simulation And Modeling Environment Framework

Andrew Turner
Masters Thesis Presentation
July 17th, 2003
Outline of Presentation

- Introduction
- Background
- Purpose
- Architecture Design
- Software Implementation
- Verification and Validation
- Simulation Examples
- Conclusions
Introduction

- Open-SESSAME Framework
  - Open-Source
  - Extensible
  - Spacecraft
  - Simulation And Modeling
  - Environment
  - Framework

Section: Background

- Spacecraft Simulation
- Software Alternatives
- Project Background
- Purpose
- Object-Oriented Programming
Spacecraft Simulation

Aerospace Engineering
- Attitude dynamics
- Orbit dynamics
- Satellite environment
- Terminology
- Aerospace tools
- Typical use cases
- Satellite Community

Computer Science / Software Engineering
- Object-Oriented Design
- Computational Speed
- Networking
- Data Handling
- Extensibility
- Software development tools
- Open-Source Community
Project Background

- Developing attitude control algorithms
- No current solution
  - Performs necessary tasks
  - Understandable and inspectable
- Need to redevelop solution every time
# Software Alternatives

<table>
<thead>
<tr>
<th>Package</th>
<th>Manufacturer</th>
<th>Benefits</th>
<th>Negatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoCon</td>
<td>NASA Goddard</td>
<td>Heritage</td>
<td>Not maintained, not readily available</td>
</tr>
<tr>
<td>DSHELL dshell.jpl.nasa.gov</td>
<td>NASA JPL</td>
<td>Free to academic institutions</td>
<td>Not readily available</td>
</tr>
<tr>
<td>Formation Testbed</td>
<td>NASA Goddard</td>
<td>Well supported, ties in with hardware</td>
<td>Expensive, limited availability, aimed towards formations</td>
</tr>
<tr>
<td>FreeFlyer <a href="http://www.ai-solutions.com">www.ai-solutions.com</a></td>
<td>a.i. solutions</td>
<td>Scriptable, good user interface, heritage, 2D/3D visualization</td>
<td>Expensive, limited integration with existing software</td>
</tr>
<tr>
<td>MultiSatSim <a href="http://www.psatsatellite.com">www.psatsatellite.com</a></td>
<td>Princeton Satellite Systems</td>
<td>Good graphics, easy to use interface, scriptable</td>
<td>Apple hardware only, limited to 8 satellites, limited expandability</td>
</tr>
<tr>
<td>ORSA orsa.sourceforge.net</td>
<td>Open-Source</td>
<td>multiple platforms, active development</td>
<td>Not complete, limited functionality, orbits only</td>
</tr>
<tr>
<td>Open-SESSAME spacecraft.sourceforge.net</td>
<td>Open-Source</td>
<td>multiple platforms, extensible, well documented, active development, orbit and attitude, separate libraries</td>
<td>No graphical user interface, requires knowledge of C++ programming</td>
</tr>
<tr>
<td>SATCOS <a href="http://www.saic.com/space/">www.saic.com/space/</a></td>
<td>SAIC</td>
<td>Heritage</td>
<td>Orbit and constellations only</td>
</tr>
<tr>
<td>SaVi savi.sourceforge.net</td>
<td>Open-Source</td>
<td>Good user interface, in development</td>
<td>Only models orbits, made for constellations, single developer</td>
</tr>
<tr>
<td>SC Modeler <a href="http://www.avmdynamics.com">www.avmdynamics.com</a></td>
<td>AVM Dynamics</td>
<td>-</td>
<td>No longer supported</td>
</tr>
<tr>
<td>Spacecraft Control Toolbox <a href="http://www.psatsatellite.com">www.psatsatellite.com</a></td>
<td>Princeton Satellite Systems</td>
<td>Well developed, documented</td>
<td>MatLab only, attitude only</td>
</tr>
<tr>
<td>Satellite ToolKit <a href="http://www.stk.com">www.stk.com</a></td>
<td>Analytical Graphics</td>
<td>Easy to use, 2D/3D visualization, heritage, large number of modules, good support, external communications, variety of operating systems</td>
<td>Expensive, not extensible, complex</td>
</tr>
<tr>
<td>Swingby</td>
<td>Computer Sciences Corporation</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>WinOrbit <a href="http://www.satnet.com/winorbit/">www.satnet.com/winorbit/</a></td>
<td>Carl Gregory, Univ. Illinois</td>
<td>Free of cost, graphical user interface</td>
<td>Windows only, not extensible, user interface difficult</td>
</tr>
</tbody>
</table>
Purpose

- Provide a useful spacecraft simulation modeling tool to engineers
  - Complement existing engineering tools
  - Include underdeveloped methodologies
  - Allow for infinite extensibility
- Educate students and other users how to design and implement simulation in software
  - Readable and useful documentation and software
  - Logical architecture design
- Develop an open-source framework that can be maintained and extended longer than the original project’s lifetime
  - Host on a public repository and include developers in the community
  - Foster communication between developers and users
Object-Oriented Programming

- Objects = Classes
  - Encapsulate Physical Objects
  - Abstract Functionality
  - Data-Hiding

- UML Diagram
- Object-Oriented Design
  - Composition
  - Inheritance
Section: Architecture Design

- Software Architecture
- Component Libraries
- Building an Application
Architecture Design

- Collections of libraries
  - Interchangeable components
  - Application framework
  - ‘Assemble’ to build an application
  - Suggested Architecture
  - Reconfigurable

- Dynamics Library
- Attitude Library
- Orbit Library
- Rotation Library
- Environment Library
- Communications
- Data Handling
Rotation Library

- **Rotation** Class
  - Abstract
  - Conversion Routines
  - Stored as a **Quaternion**

<table>
<thead>
<tr>
<th>Rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Quaternion</td>
</tr>
<tr>
<td>+ SetDCM</td>
</tr>
<tr>
<td>+ SetEulerAngles</td>
</tr>
<tr>
<td>+ SetEulerAxisAngle</td>
</tr>
<tr>
<td>+ SetMRP</td>
</tr>
<tr>
<td>+ SetQuaternion</td>
</tr>
<tr>
<td>+ GetDCM</td>
</tr>
<tr>
<td>+ GetEulerAxisAngle</td>
</tr>
<tr>
<td>+ GetMRP</td>
</tr>
<tr>
<td>+ GetQuaternion</td>
</tr>
</tbody>
</table>

- **DirectionCosine Matrix**
  - **Matrix**

- **Quaternion**
  - **Vector**

- **ModifiedRodriguez Parameters**

- **Other Representations**
Attitude Library

- **AttitudeState**
  - Composition
  - Rotation, Frame, AngVel

- **Attitude**
  - Larger composition
  - AttitudeState
  - Kinematics & Dynamics
  - History
Orbit Library

- **OrbitState**
  - Local Composite
- **OrbitStateRep**
- **OrbitFrame**
- **Orbit**
  - Larger Composite
  - Dynamics
  - History

```
Orbit
  └── OrbitState
    ├── 1..n
    │    └── OrbitState-Representation
    │       ├── PositionVelocity
    │       │       └── OrbitFrame
    │       │           └── OrbitFrameECEF
    │       │               ├── OrbitFramePQW
    │       │               │       └── OrbitFrameIJK
    │       │               │           └── OrbitFrameRSW
    │       │               │               └── Other OrbitFrames
    └── Other State Representations
```

---

14
Dynamics Library

- **Propagator**
  - Abstract interface
  - Types of propagators
    - Numeric
    - Analytic

```
Propagator
  ├── NumericPropagator
  │    └── Combined-NumericPropagator
  └── AnalyticPropagator
      ├── EnkeCombined-NumericPropagator
      └── Other Numeric Propagators
```

AnalyticPropagator Schemes
Environment Library

- **Environment**
  - Composition and interface

- **CentralBody**
  - Physical parameters
  - Defines frames
  - Location with respect to other celestial bodies

- **EnvFunctions**
  - Interface to force and torque functions
  - Parameters: time, orbit state, attitude state
Data Handling Utilities

- **History**
  - Vectors of data
  - Associated with times
  - Interpolation
  - Data Conversion

![Diagram showing data handling utilities with History, AttitudeHistory, OrbitHistory, ControlHistory, and Other History Collections with associated data types and time parameters.](image)
Math Library

- Numerical Integration
  - Fixed step
  - Variable step
- Data Interpolation
Open-SESSAME Application

1. Choose Equations of Motion
2. Add Dynamic Components
3. Choose Orbit Characteristics
4. Choose Attitude Characteristics
5. Add Environment Effects
6. Add History
7. Choose Interpolators
8. Simulate!
Section: Software Implementation

- Software Implementation
- Code Metrics
- Implementation Artifacts
- Documentation
Software Implementation

- C++ software language
- Open-source libraries and toolkits
  - Matrix
  - Network Communications
- Documentation
  -Verbose coding
  - Doxygen
Code Metrics

- Lines of code: 4810
- Lines of documentation: 8112
- Number of classes: 44
- Functions: 498
- Avg executable size: ~4 MB

<table>
<thead>
<tr>
<th>Library</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attitude</td>
<td>335k</td>
</tr>
<tr>
<td>Data Handling</td>
<td>749k</td>
</tr>
<tr>
<td>Dynamics</td>
<td>697k</td>
</tr>
<tr>
<td>Environment</td>
<td>518k</td>
</tr>
<tr>
<td>Math</td>
<td>430k</td>
</tr>
<tr>
<td>Matrix</td>
<td>1.88M</td>
</tr>
<tr>
<td>Orbit</td>
<td>658k</td>
</tr>
<tr>
<td>Rotation</td>
<td>255k</td>
</tr>
<tr>
<td>Utils</td>
<td>223k</td>
</tr>
</tbody>
</table>
Implementation Artifacts

- Framework bundle
  - 9 libraries
  - Source code
  - Example applications
- Documentation
  - User’s Manual
  - Master’s Thesis
  - Conference paper
- Public repository
  - Forums, documentation, Source (CVS)
Documentation

- Doxygen
  - Automatic document generation
  - Produces html, xml, latex (pdf)
  - Includes equations
- Verbose coding
  - ‘Hungarian Notation’
  - Inline comments
- Cite references
- Programming primer
Section: Verification and Validation

- Verification
  - Purpose
  - Methods
  - Results

- Validation
  - Purpose
  - Methods
  - Results
Verification

- Show that the software is implemented correctly - Verify
  - Computational results from software match analytical analysis
  - Output is compared to alternative, industry accepted, solutions
Orbit Verification

Conservation of Angular Momentum

Keplerian Equation

\[ \mathbf{h} = \mathbf{r} \times \mathbf{v} = \text{constant} \]

<table>
<thead>
<tr>
<th>( r_i^j )</th>
<th>( [1131.34, -2282.343, 6672.423]^T \text{km} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_i^j )</td>
<td>( [-5.64305, 4.30333, 2.42879]^T \text{km/s} )</td>
</tr>
<tr>
<td>( t )</td>
<td>2400 sec</td>
</tr>
</tbody>
</table>

Integrator: RK-4(5), tol: \( 10 \times 10^{-9} \)

Output

<table>
<thead>
<tr>
<th>Running Time</th>
<th>0.59 sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_{Kep}(t) )</td>
<td>( [-4219.853, 4363.116, -3958.789]^T \text{km} )</td>
</tr>
<tr>
<td>( v_{Kep}(t) )</td>
<td>( [3.689736, -1.91662, -6.112528]^T \text{km/s} )</td>
</tr>
<tr>
<td>( r_{Sim}(t) )</td>
<td>( [-4219.752, 4363.029, -3958.766]^T \text{km} )</td>
</tr>
<tr>
<td>( v_{Sim}(t) )</td>
<td>( [3.689865, -1.916734, -6.112511]^T \text{km/s} )</td>
</tr>
<tr>
<td>RMS error</td>
<td>( \Delta r_{RMS} = 3.1493 \times 10^{-5} )</td>
</tr>
<tr>
<td></td>
<td>( \Delta v_{RMS} = 6.95115 \times 10^{-5} )</td>
</tr>
</tbody>
</table>
Closed-Form Attitude Solution

Euler’s Equations:

\[ \dot{\mathbf{\omega}} = \mathbf{g} - \mathbf{\omega} \times I\mathbf{\omega} \]

Assume: axisymmetric

\[ I_T = I_1 = I_2 \]

Closed-form solution:

\[
\begin{align*}
\omega_1 &= \omega_{01} \cos \omega_p t + \omega_{02} \sin \omega_p t \\
&\quad + \frac{1}{I_T} \int_0^t [g_1(\tau) \cos (\omega_p(t - \tau)) + g_2(\tau) \sin (\omega_p(t - \tau))] d\tau \\
\omega_2 &= -\omega_{01} \sin \omega_p t + \omega_{02} \cos \omega_p t \\
&\quad + \frac{1}{I_T} \int_0^t [-g_1(\tau) \sin (\omega_p(t - \tau)) + g_2(\tau) \cos (\omega_p(t - \tau))] d\tau \\
\omega_3 &= \omega_{03}
\end{align*}
\]
# Attitude Verification

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega_0$</td>
<td>$[0.3, -0.4, 0.7]^T \text{rad/s}$</td>
</tr>
<tr>
<td>$I_t$</td>
<td>100 kg m$^2$</td>
</tr>
<tr>
<td>$I_3$</td>
<td>150 kg m$^2$</td>
</tr>
<tr>
<td>$g(t)$</td>
<td>$[0.2 \sin t \quad -0.4 \sin t \quad 0]^T \text{Nm}$</td>
</tr>
<tr>
<td>$t$</td>
<td>300 sec</td>
</tr>
<tr>
<td>Integrator</td>
<td>RK-4(5), tol: $10 \times 10^{-8}$</td>
</tr>
</tbody>
</table>

## Output

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega(t)$</td>
<td>$[-0.46134984 \quad -0.194318263 \quad 0.7]^T \text{rad/s}$</td>
</tr>
<tr>
<td>$\omega_{Sim}$</td>
<td>$[-0.46134984100 \quad -0.193182629296 \quad 0.7]^T \text{rad/s}$</td>
</tr>
<tr>
<td>RMS error</td>
<td>$\sqrt{(-1.15712 \times 10^{-9})^2 + (2.942222 \times 10^{-9})^2} = 3.165158 \times 10^{-9} \text{rad/s}$</td>
</tr>
</tbody>
</table>
Validation

- Show that the correct software is implemented - Validate
  - Does the code meet requirements of use?
  - Does the implementation improve upon existing solutions?
  - Can the software easily demonstrate its purpose?
Section: Simulation Examples

- Attitude Integration
- Orbit Integration
- Coupled Simulation
- Hardware-in-the-loop
  - HokieSat Orbit/Attitude Coupled Simulation Server
  - HOACSS
1. Code the attitude dynamics equation
2. Code the disturbance torque function
3. Assign function parameters
4. Create an initial attitude state
5. Create and initialize integrator
6. Integrate the equations
7. Graph or output the state history
Attitude Simulation Output

- Gnuplot
- Save/Export
- Dynamics display
Orbit Simulation Output

- Similar application construction
- Output orbit position

![Graph showing orbit simulation output](image)
Hardware-in-the-Loop
HokieSat Simulation

Orbit Plot

Attitude Plot
Conclusions

- No prior solution for configurable, extensible, coupled spacecraft simulation
- Open-SESSAME fulfills goals and requirements
  - Speed and accuracy comparable to commercial packages
  - Used to ease development and testing of HokieSat flight code
- Future
  - Multiple extension points for added functionality
  - Integrate software into curriculum
  - Publish use cases and include in lab activities
  - Increase connectivity with other programs
  - Optimize operation
Thank you

- Space Systems Simulation Laboratory
  - Jana Schwartz
  - Dr. Christopher Hall
  - Sourceforge.net
  - ION-F & HokieSat

Virginia Tech Department of Aerospace & Ocean Engineering
Questions?
Accumulation of Error

High Eccentricity Error
## High Eccentricity

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Orbit State</td>
<td>( \mathbf{r}_0 = \begin{bmatrix} 7500 &amp; 0 &amp; 0 \end{bmatrix}^T )</td>
</tr>
<tr>
<td></td>
<td>( \mathbf{v}_0 = \begin{bmatrix} 0 &amp; 6.819333903 &amp; 6.819333903 \end{bmatrix}^T )</td>
</tr>
<tr>
<td>Disturbances</td>
<td>Two-Body Gravity</td>
</tr>
<tr>
<td>Integrator</td>
<td>Runge-Kutta-Fehlberg 4(5)</td>
</tr>
<tr>
<td>Simulation Time</td>
<td>20 days</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td><strong>O-SESSAME</strong></td>
</tr>
<tr>
<td>Final Orbit State</td>
<td>( \begin{bmatrix} -46724.94751 \ -8276.177337 \ -8276.177337 \ 1.339061791 \ -0.85741544 \ -0.85741544 \end{bmatrix} )</td>
</tr>
<tr>
<td>Running Time</td>
<td>15 sec</td>
</tr>
<tr>
<td>RMS Error</td>
<td>-</td>
</tr>
</tbody>
</table>