An Open-Source Simulation Tool for Study and Design of Spacecraft Attitude Control Systems

Eric Stoneking
Feb 2, 2018
Three Audiences for This Talk?

• The User
  – "How do I solve today's problem, today?"

• The Developer
  – "What does a sim look like on the inside?"

• The Modeler
  – "Okay, fine, but what can it do?"
42 from the User's Perspective
The User Experience

• 42 is a command-line program
• Setup performed with plain text input files
  – Simulation parameters and settings
  – Spacecraft, orbit parameters and initial conditions
• Runs with or without graphics
  – Graphics adds situational awareness
  – Sim runs faster without graphics
• Plain-text output files produced for post-run analysis
• Graphics frames may be captured
  – Stitched together into movies using other software (eg. ffmpeg)
Rapid Prototyping

- Some studies may be conducted without any C coding
- Simple attitude command profiles may be specified in Inp_Cmd.txt
- "Prototype" control law follows that profile
- Sufficient for many concept studies
  - Evaluate instrument fields of regard
  - Size wheels, magnetic torquers for environment
Example Inp_Cmd.txt

```
0.0 SC[0] qrl = [0.0 0.0 0.0 1.0]
0.0 Point SC[1].B[0] Primary Vector [0.0 0.0 -1.0] at SUN
0.0 Align SC[1].B[0] Secondary Vector [0.0 1.0 0.0] with L-frame Vector [0.0 1.0 0.0]
0.0 Point SC[2].B[0] Primary Vector [0.0 0.0 -1.0] at SC[1]
0.0 Align SC[2].B[0] Secondary Vector [0.0 1.0 0.0] with L-frame Vector [0.0 1.0 0.0]
0.0 Align SC[3].B[0] Primary Vector [0.0 0.0 1.0] with L-frame Vector [1.0 0.0 1.0]
0.0 Point SC[3].B[0] Secondary Vector [1.0 0.0 0.0] at SUN
0.0 SC[4] Cmd Angles = [-90.0 -90.0 0.0] Seq = 131 wrt N Frame
```
In-Depth Studies

• More in-depth studies will require C coding
  – Write your own control laws, "flight software"
    • Some examples provided as a jumping-off point
  – Add custom sensor and actuator models
  – Add output to files to support your analysis needs
Matlab + 42 = Monte Carlo

- 42 can be called from within Matlab using the `system` command
- Use Matlab as the MC executive
  - Generate initial conditions, parameters
  - Write to 42's input files
  - Run 42
  - Process and save data
  - Repeat
- Use 42 as the high-speed, high-fidelity component
Matlab/42 Example

for Irun=1:Nrun,

% Compute initial attitude
CRN = TRIAD(tvn(Irun,:),svn,[0 0 1],[1 0 0]);
qrn = C2Q(CRN);

% Write target to file
Outdata = [TrgRA(Irun) TrgDec(Irun)];
save -ascii ./MOMBIAS/TargetRaDec.inp Outdata

% Write initial attitude to file
line = sprintf('%f %f %f %f ! Quaternion\n', qrn(1),qrn(2),qrn(3),qrn(4));
OverwriteLineInFile('./MOMBIAS/GLAST.inp',21,line);

% Run 42 for three days.
system('./42 MOMBIAS');

% Record pointing histogram.
load ./MOMBIAS/AngleToGo.42
[HistCount(Irun,:),HistAng(Irun,:)] = hist(AngleToGo,20);
end
Flight Software Testing to Operations

• Eventually, the control laws become flight software, running outside 42 on some other computer

• 42 can communicate over sockets
  – Sim "engine" <-> Flight software
  – Sim "engine" -> Sim "display"

• Splitting engine and display enables multiple displays

• For operations support, displays may be driven by flight telemetry instead of engine
Example: Hardware-in-the-Loop Sim with Multiple Spacecraft
Will It Run On My Computer?

- Most likely
- 42 is open-source, available for download from Sourceforge.net/projects/fortytwospacecraftsimulation
- For MacOS and linux, installation is very easy
  - Unzip archive
  - Put 42 folder wherever you want it
  - Edit Makefile to make sure it has your platform correct
  - make and run
- For Windows, there are some external dependencies
  - MinGW, msys provide a linux-style terminal window
  - glew, freeglut required to support graphics
  - Full instructions provided in 42/Docs/Install-msys.txt
42 from the Developer's Perspective
A Basic Simulation Loop

- **Initialize**
  - Read user inputs
  - Set up
- **Ephemeris: Where is everything?**
  - Sun, Earth, Moon, etc
  - Orbits
  - Spacecraft
- **Environment Models: What forces and torques exerted by the environment?**
- **Sensor Models**
  - Input truth
  - Output measurements
- **Flight Software Models**
  - Input Measurements
  - Process Control Laws, etc
  - Output Actuator Commands
- **Actuator Models**
  - Input Commands
  - Output Forces and Torques
- **Dynamics: How does S/C respond to forces and torques?**
  - Integrate dynamic equations of motion over a timestep
  - Advance time to next step
42's Software Architecture

- Ephemeris Models
- Environment Models
- Flight Software Models
- Sensor and Actuator Models
- Dynamics Models
- Global data structures
- FSW data structure
- SimStep()

Input:
- Simulation Initial Conditions and Control (ASCII Input)
- Truth (ASCII Output)

Output:
- OpenGL Graphics Output
- Mouse, Keyboard Input

Models:
- Environmental Forces & Torques
- Control Forces & Torques
- Dynamic States
Good Conventions Make Code Readable, Debuggable

- Choose standard notation to make code readable, unambiguous
  - Think about how notation morphs from the written page to code
- Make code document itself
  - It's much easier to debug

Table 1: Common Reference Frames

<table>
<thead>
<tr>
<th>Frame</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Inertial Frame (N = Newton)</td>
</tr>
<tr>
<td>L</td>
<td>Local Vertical-Local Horizontal</td>
</tr>
<tr>
<td>R</td>
<td>Command Frame (R = Reference)</td>
</tr>
<tr>
<td>B</td>
<td>Body Frame</td>
</tr>
</tbody>
</table>

Table 2: Commonly-used Expressions

<table>
<thead>
<tr>
<th>Notation</th>
<th>Written</th>
<th>Spoken</th>
<th>Coded</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N^B$</td>
<td>Angular velocity of $B$ in $N$</td>
<td>wbn, SC[i].B[j].wn</td>
<td></td>
</tr>
<tr>
<td>$N^B$</td>
<td>Velocity of $B^*$ in $N$</td>
<td>SC[i].B[j].vn</td>
<td></td>
</tr>
<tr>
<td>$B^N$</td>
<td>DCM of $B$ in $N$ (or from $N$ to $B$)</td>
<td>CBN, SC[i].B[j].CN</td>
<td></td>
</tr>
<tr>
<td>$q^N$</td>
<td>Quaternion of $B$ in $N$ (or from $N$ to $B$)</td>
<td>qbn, SC[i].B[j].qn</td>
<td></td>
</tr>
<tr>
<td>$v^A$</td>
<td>Components of $v$ in $A$, $v$ expressed in $A$</td>
<td>va</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Common Constructions

<table>
<thead>
<tr>
<th>Written</th>
<th>Coded</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A^v = A^C B^v$</td>
<td>MxV(CAB, vb, va)</td>
</tr>
<tr>
<td>$A^v = B^v R^A$</td>
<td>VxM(vb, CBA, va)</td>
</tr>
<tr>
<td>$A^v = (B^A)^T B^v$</td>
<td>MxV(CBA, vb, va)</td>
</tr>
<tr>
<td>$A^v = B^v (A^C)^T$</td>
<td>VxMT(vb, CAB, va)</td>
</tr>
<tr>
<td>Convert $B^C$ to $B^N$</td>
<td>C2Q(CBN, qbn)</td>
</tr>
<tr>
<td>Convert $B^q$ to $B^C$</td>
<td>Q2C(qbn, CBN)</td>
</tr>
<tr>
<td>Convert Euler Angles (2-1-3 Sequence) to DCM</td>
<td>A2C(213, ang1, ang2, ang3, C)</td>
</tr>
<tr>
<td>$N^R = (N^C)^T$</td>
<td>MT(CRN, CNR)</td>
</tr>
<tr>
<td>$H^C = H^C N^C A^R$</td>
<td>MxMT(CBN, CRN, CBR)</td>
</tr>
<tr>
<td>$B^q = B^q \otimes N^q$</td>
<td>QxQT(qbn, qrn, qbr)</td>
</tr>
</tbody>
</table>

from 42/Docs/Nomenclature.pdf
Reference Frames are Important!

• In any dynamics problem beyond the spinning top, a systematic approach to reference frames and the relationships between them is vital
• For 42, we define several fundamental reference frames, and notational conventions to keep quaternions and direction cosines sorted out
Reference Frames (1 of 2)

• Heliocentric Ecliptic (H)
  – Planet positions expressed in this frame
• Each world has an inertial (N) and rotating (W) frame
  – For Earth, N = ECI (True of date), W = ECEF
  – N is the bedrock for orbits, S/C attitude dynamics
  – Full Disclosure: Although True-of-Date <-> J2000 conversions are provided, the distinction is not always rigorously made
    • Star vectors provided in J2000 (from Skymap), converted to H
    • Planet ephemerides are assumed given in true-of-date H
    • Transformation from N to W is simple rotation, implying N is True-of-Date
    • TOD ↔ J2000 conversions in envkit.c
Reference Frames (2 of 2)

• Each reference orbit has a reference point R
  – For two-body orbit, R moves on Keplerian orbit
  – For three-body orbit, R propagates under influence of both attracting centers (as point masses)
  – S/C orbit perturbations integrated with respect to R

• Associated with each R is a LVLH frame (L) and a formation frame (F)
  – F is useful for formation-flying scenarios
  – F may be offset from R, may be fixed in N or L

• Each spacecraft has one or more Body (B) frames and one LVLH frame (L)
  – L(3) points to nadir, L(2) points to negative orbit normal
  – SC.L is distinct from Orb.L, since SC may be offset from R
Representing Attitude

• There are several ways to represent the rotation between two reference frames
  – Direction Cosines
  – Euler Angles
  – Quaternions (aka Euler Parameters)
  – and more

• They all have their strengths and weaknesses
  – Learn them all!
## Strengths and Weaknesses of Attitude Representations

<table>
<thead>
<tr>
<th>Representation</th>
<th>Strengths</th>
<th>Weaknesses</th>
<th>Best Used For</th>
</tr>
</thead>
</table>
| Direction Cosines    | • Work well with vectors  
                        • Easy to catenate rotations  
                        • Moderately intuitive (dot products)  
                        • No singularities                | • 9 params for 3 DOF                          | • Transforming Vectors                      |
| Quaternions          | • Efficient (4 params for 3 DOF)  
                        • No singularities              | • Not intuitive                        | • Propagating Equations of Motion           |
| Euler Angles         | • Intuitive  
                        • 3 params for 3DOF                | • Singularities  
                        • 24 Variants                      | • Input, Output  
                        • Gimballed Joints                |
Notation for Quaternions, DCMs

- The rotation from frame $A$ to frame $B$ may be described by the direction cosine matrix
  \[ B C^A_{ij} = \hat{b}_i \cdot \hat{a}_j \]

- Given the components of a vector in $A$, its components in $B$ may be found by the multiplication
  \[ B \mathbf{v} = B C^A A \mathbf{v} \]

- In C, we write the DCM as $CBA$ to preserve order of superscripts, e.g.
  \[ \text{MxV}(CBA, \mathbf{v}_A, \mathbf{v}_B) \]

- Quaternions are another way to describe rotations. We use a parallel notation:
  \[ QxV(qba, \mathbf{v}_A, \mathbf{v}_B) \]

- These and similar conventions promote concise, unambiguous code
42 from the Modeler's Perspective
Features

• Multiple spacecraft, anywhere in the solar system
  – Two-body, three-body orbit dynamics
  – One sun, nine planets, 45 major moons
  – Minor bodies (comets and asteroids) added as needed
    • Bennu, Eros, Itokawa, Wirtanen, etc

• Supports precision formation flying
  – Several S/C may be tied to a common reference orbit
  – Encke’s method or Euler-Hill equations used to propagate relative orbit states
    • Precision maintained by judicious partitioning of dynamics
      – Add big things to big things, small things to small things

• Clean FSW interface facilitates FSW validation
  – As flight software matures, it can be migrated out of 42
  – Used by GLAST project for independent validation of vendor’s (autocoded) GNC flight software
Environment Models

• Planetary Ephemerides
  – From Meeus, “Astronomical Algorithms”
  – Good enough for GNC validation, not intended for mission planning
    • Use GMAT or ODTBX for that
• Gravity Models have coefficients up to 18th order and degree
  – Earth: EGM96
  – Mars: GMM-2B
  – Luna: GLGM2
• Planetary Magnetic Field Models
  – IGRF up to 10th order (Earth only)
  – Tilted offset dipole field
• Earth Atmospheric Density Models
  – MSIS-86 (thanks to John Downing)
  – Jacchia-Roberts Atmospheric Density Model (NASA SP-8021)
  – NRLMSISE00 (Update to MSIS-86, extended down to ground)
• Simple exponential Mars atmosphere density model
  – New models easily incorporated as the state of the art advances
Dynamics Models

• Full nonlinear “6DOF” (actually N-DOF) dynamics

• Attitude Dynamics
  – One or many bodies
    • Tree topology (no kinematic loops)
  – Each body may be rigid or flexible
  – Joints may combine rotational and translational DOFs
    • May be gimballed or spherical
  – Slosh may be modeled as a pendulum (lo-fi, quick to implement and run)
    • 42 may run concurrently with Star-CCM CFD software for hi-fi slosh
  – Wheels embedded in Body[0]
  – Torques from actuators, aerodynamic drag, gravity-gradient, solar radiation pressure, joint torques

• Orbit Dynamics
  – Two- or three-body orbits
  – Encke or Euler-Hill (Clohessy-Wiltshire) for relative orbit motion (good for formation flying, prox ops)
  – Forces from actuators, aerodynamic drag, non-spherical gravity, third-body gravity, solar radiation pressure
The Bleeding Edge

• 42 is under constant development

• Here are some capabilities that are still provisional or under development
  – Contact forces (provisional)
    • Applied to some problems, not robust
  – Self-shadowing (provisional)
    • Passed first sanity checks, but some bugs persist
  – Flight in atmosphere (provisional)
    • Pieces in place, no rigorous test problem yet
  – Fluid slosh using Smoothed Particle Hydrodynamics (under development)
    • Needs parallelization to be practical
  – Interfaces to cFS, COSMOS (under development)
    • cFS is open-source flight software system from GSFC
    • COSMOS is open-source ops (cmd/tlm, etc) from Ball
Conclusion

• 42 is intended to support the ACS design cycle from concept to operations
  – Rapid prototyping for concept studies
  – High fidelity for validation, design
  – Plays well in integration, ops ecologies
• Notation, conventions are the key to building a large software tool over time
• \( F = ma \). All the rest is just accounting.