UNIVERSITY OF CAMBRIDGE

University at Buffalo
The State University of New York

Verification of the Draper Semianalytic Satellite Theory in OREKIT

Barry Bentley
University of Cambridge

Paul Cefola
University at Buffalo SUNY

Rev 5

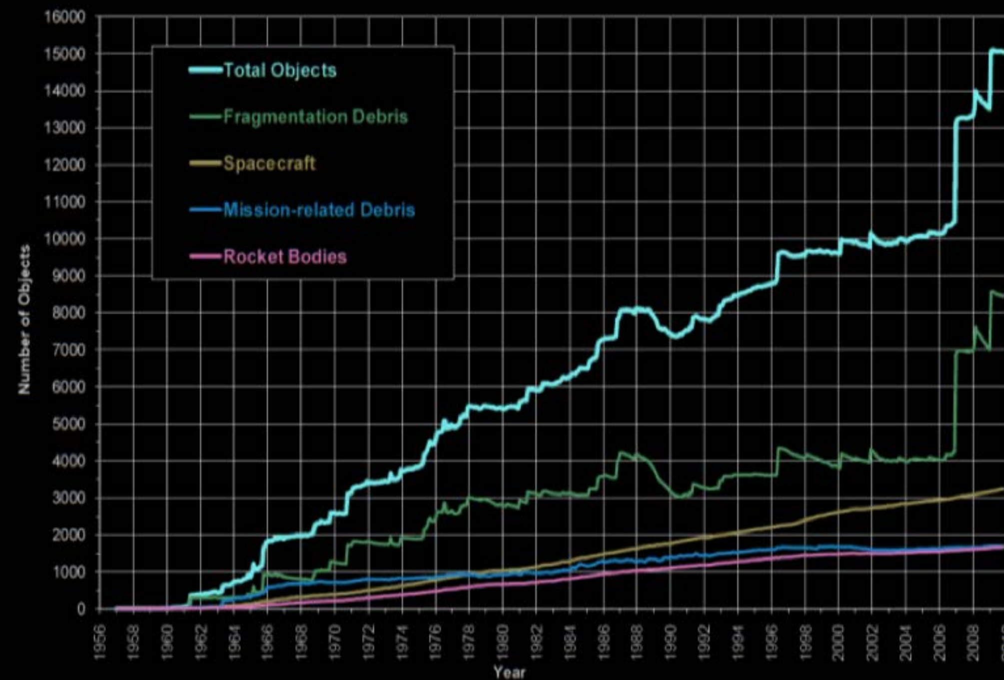
Outline

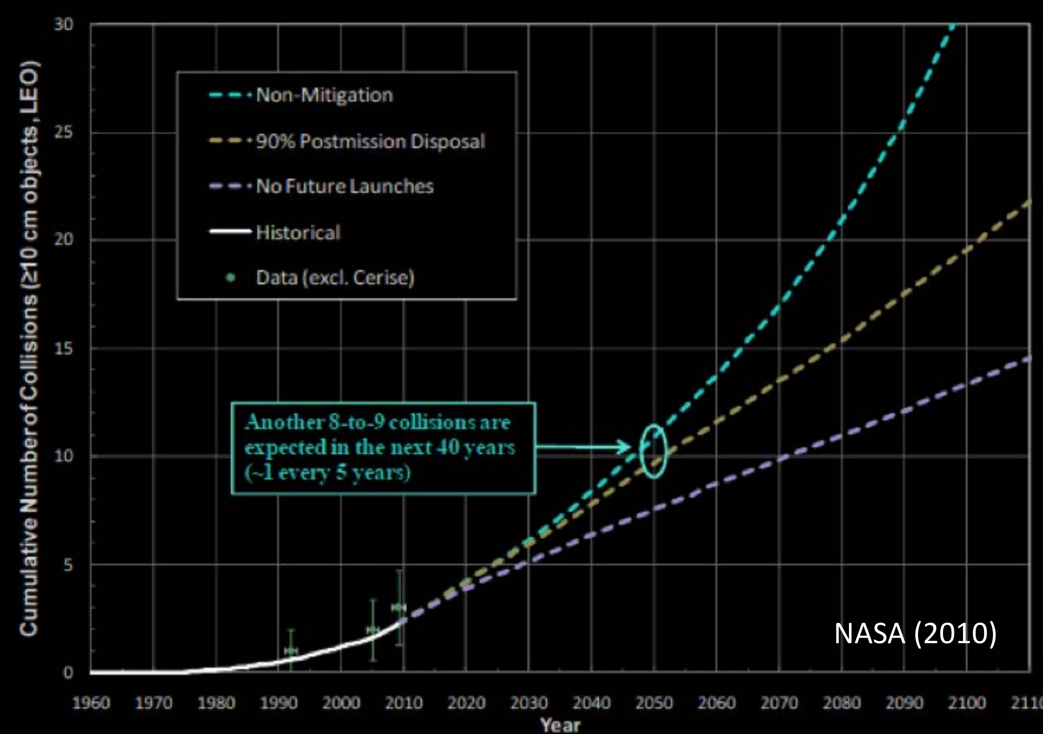
- Space Situational Awareness (SSA)
- Legacy astrodynamics software
- Orbit extrapolation kit (OREKIT)
- The Draper Semianalytic Satellite Theory (DSST)
- OREKIT implementation of DSST
- System testing

The Problem:

- Increasing space debris
- Estimated number of uncatalogued items: ~60 000

  Osiander & Ostdiek (2009)
- Legacy *(proprietary)* software systems





NASA (2010)

Astrodynamics software

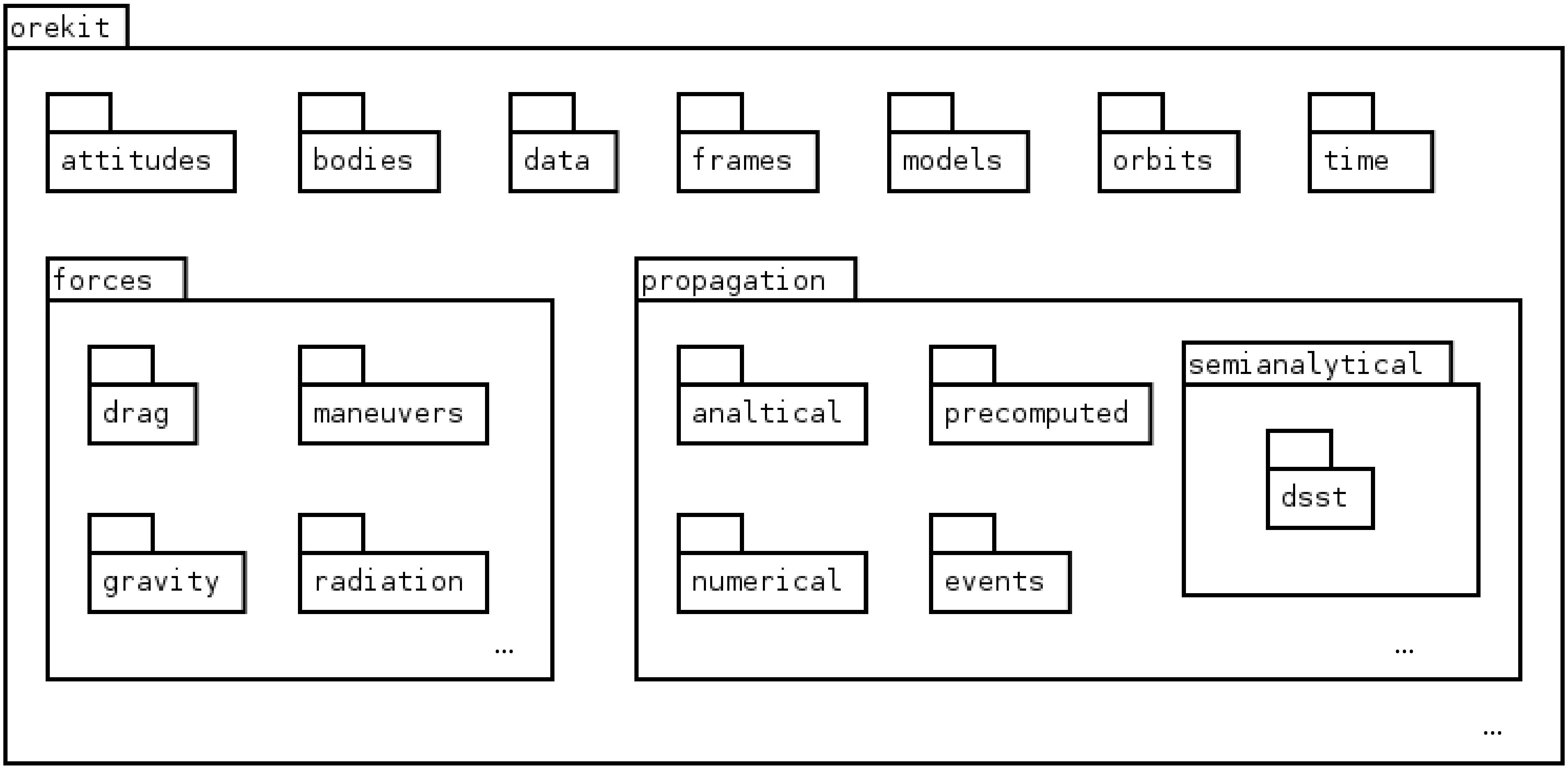
- Legacy software:
  - Increasing maintenance costs
  - Slow (or no) update lifecycle
  - Bound by out-dated hardware or design principles
- Proprietary software:
  - Less transparency
  - Difficult to audit or update
  - Less R&D
- Solution: Open Source astrodynamics software

OREKIT *(**OR**bit **E**xtrapolation **KIT**)*

- Open Source astrodynamics library in Java, developed at CS
- Tested on rendezvous between ATV and International Space Station
- Framework allows easy use of multiple:
  Time scales, Frames, Data formats, Propagators
- Growing developer base
- Can be modified for commercially use (Apache License)
- Free and *user extensible*

Functional decomposition in OREKIT

- **Object oriented:** focus on objects rather than processes
- **Encapsulation:** *plug-and-play* orbit construction



DSST migration

- Possible to migrate algorithms from legacy software to OREKIT framework
- Starting with Draper Semianalytic Satellite Theory (DSST)
- SST development started in 1970s at NASA GSFC. Continued at Draper Laboratory and recently MIT Lincoln
- Access to DSST is limited. Development of web interface, but source is closed and in Fortran 77

What is semianalytic satellite theory?

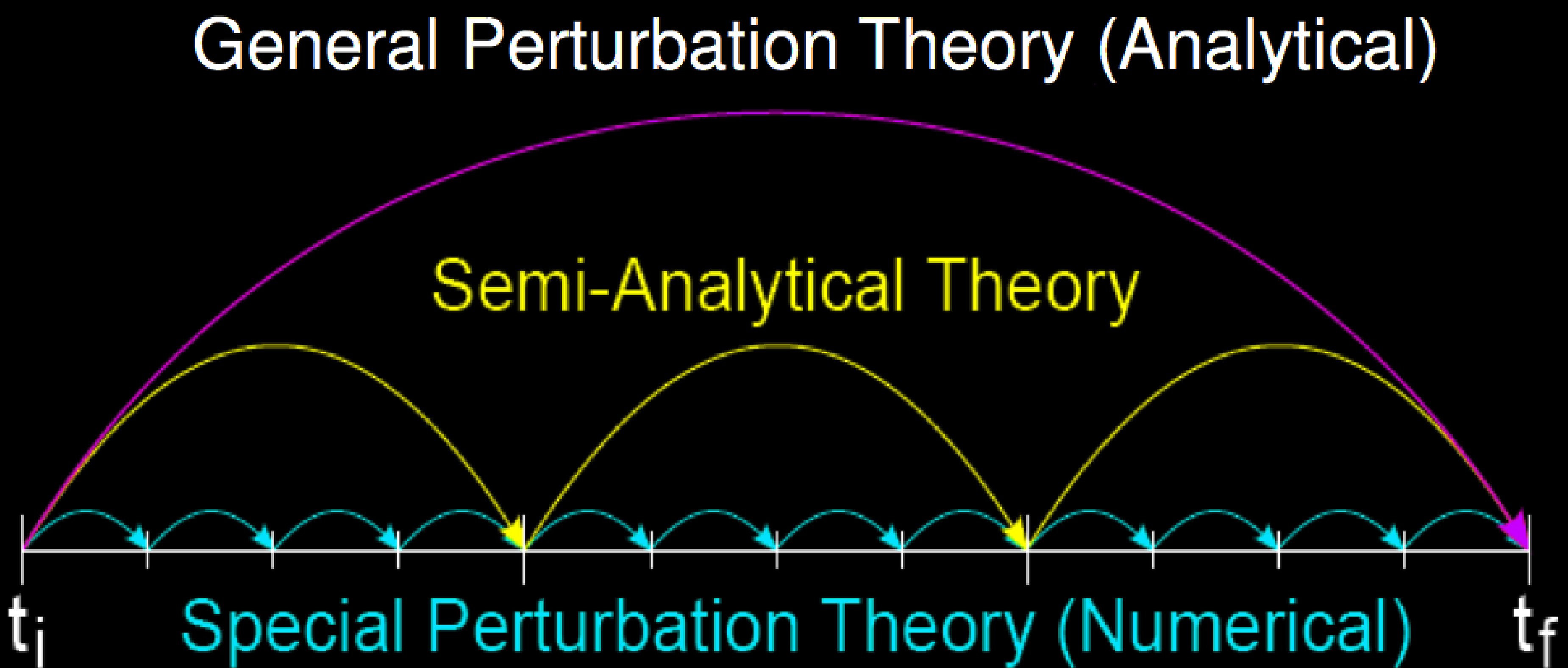
- Generalised Method of Averaging used to decouple long and short period motion
- Treat contributions separately and integrate long-period motion with larger step size
- Increases computational efficiency while retaining accuracy
- Useful for resource-intense tasks such as SSA

Draper Semi-analytical Satellite Theory (DSST)

Comprehensive force models:

- Zonal harmonics (thru J_{50})
- Tesseral harmonics (50x50 field)
- J_2-squared terms
- Lunar-solar point-masses
- Solid Earth tides
- Atmospheric drag
- Solar radiation pressure

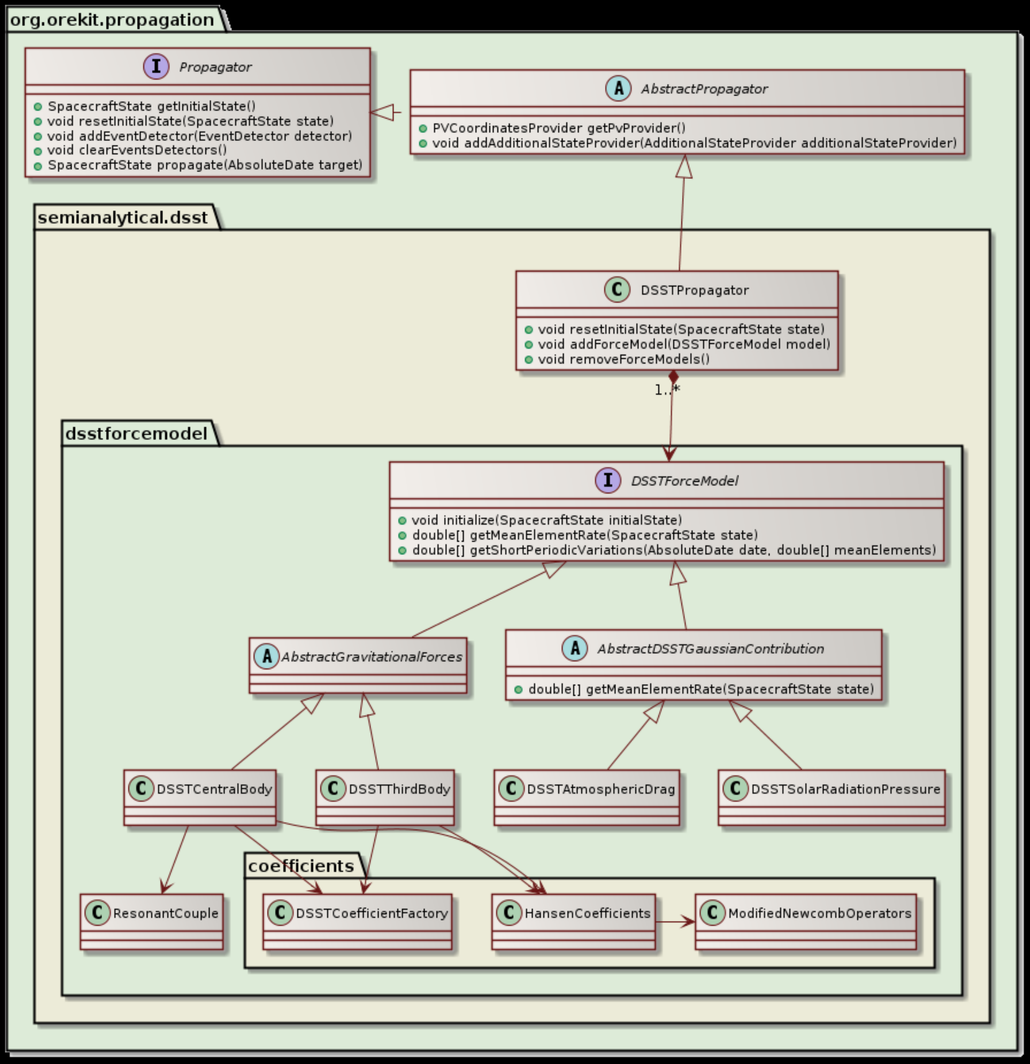
Comparison of integration grids



J. F. San Juan (2012)

OREKIT implementation

- Current implementation based on Danielson *et al.* (1995)
- Refactored to fit the OREKIT framework (Object Oriented Design)
- Interoperability with other OREKIT classes
- Undergoing verification



Semianalytic Satellite Theory Document–

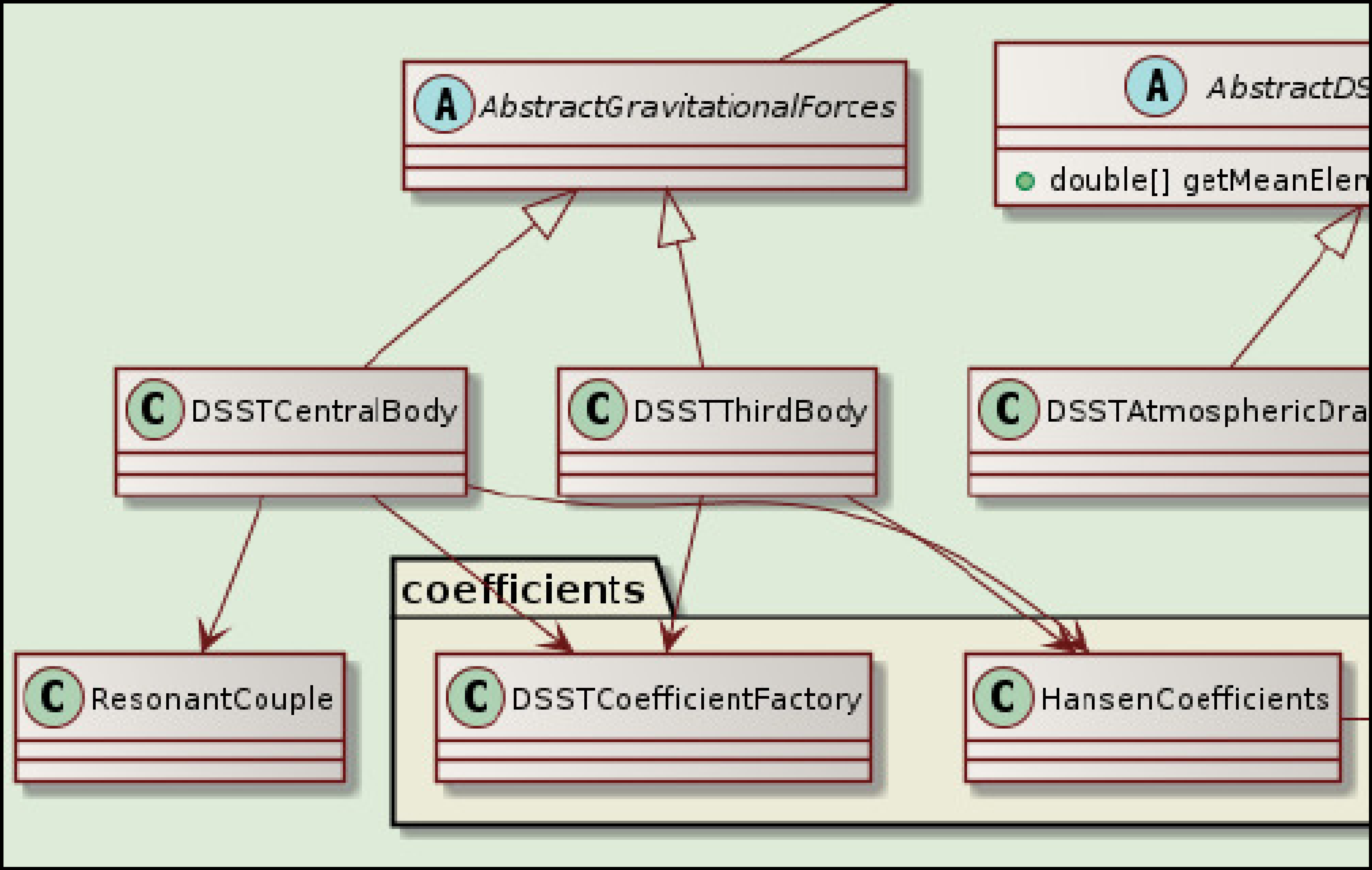
D. Danielsen, C. Sagovac, B. Neta, L. W. Early

- Naval Postgraduate School, Monterey, CA
- Dr. Steve Knowles of the Naval Space Command gave the initial impetus for the generation of this document
- The purpose was to simplify, assemble, unify, and extend the DSST theory whose documentation was only available in conference preprints, published papers, technical reports (including MIT theses), and private communications
- Report developed around 1993-95 (theory development from 1974).

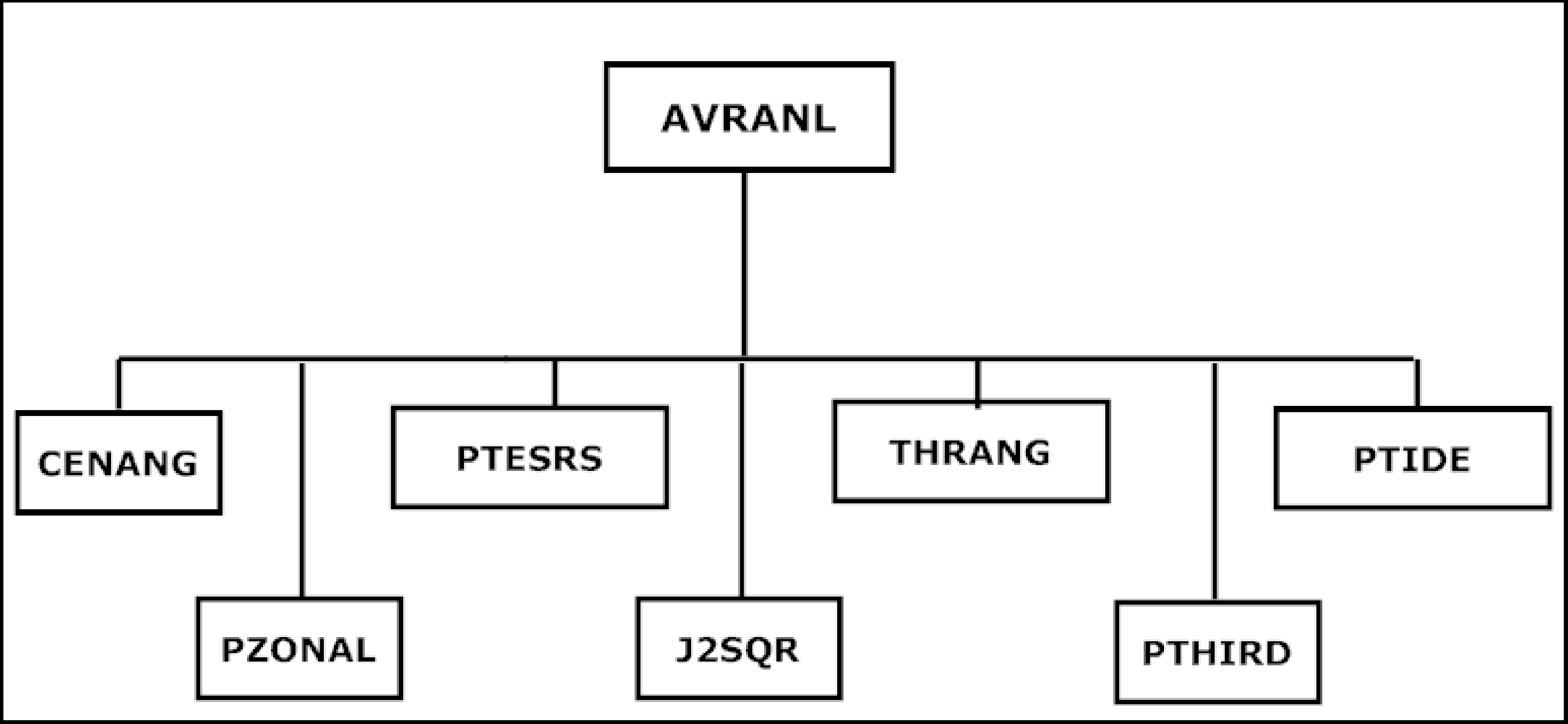
Semianalytic Satellite Theory Doc. Cont'd

- Outline
  - Mathematical Preliminaries
  - First-Order Mean Element Rates
  - First-Order Short-Periodic Variations
  - Higher-Order Terms
  - Truncation Algorithms
  - Numerical Methods
- The NPS report introduces modifications to the formulation at secondary levels
- The NPS report has not previously been used as the basis for software

Drilling Down into the Orekit dsst-propagator-class-diagram



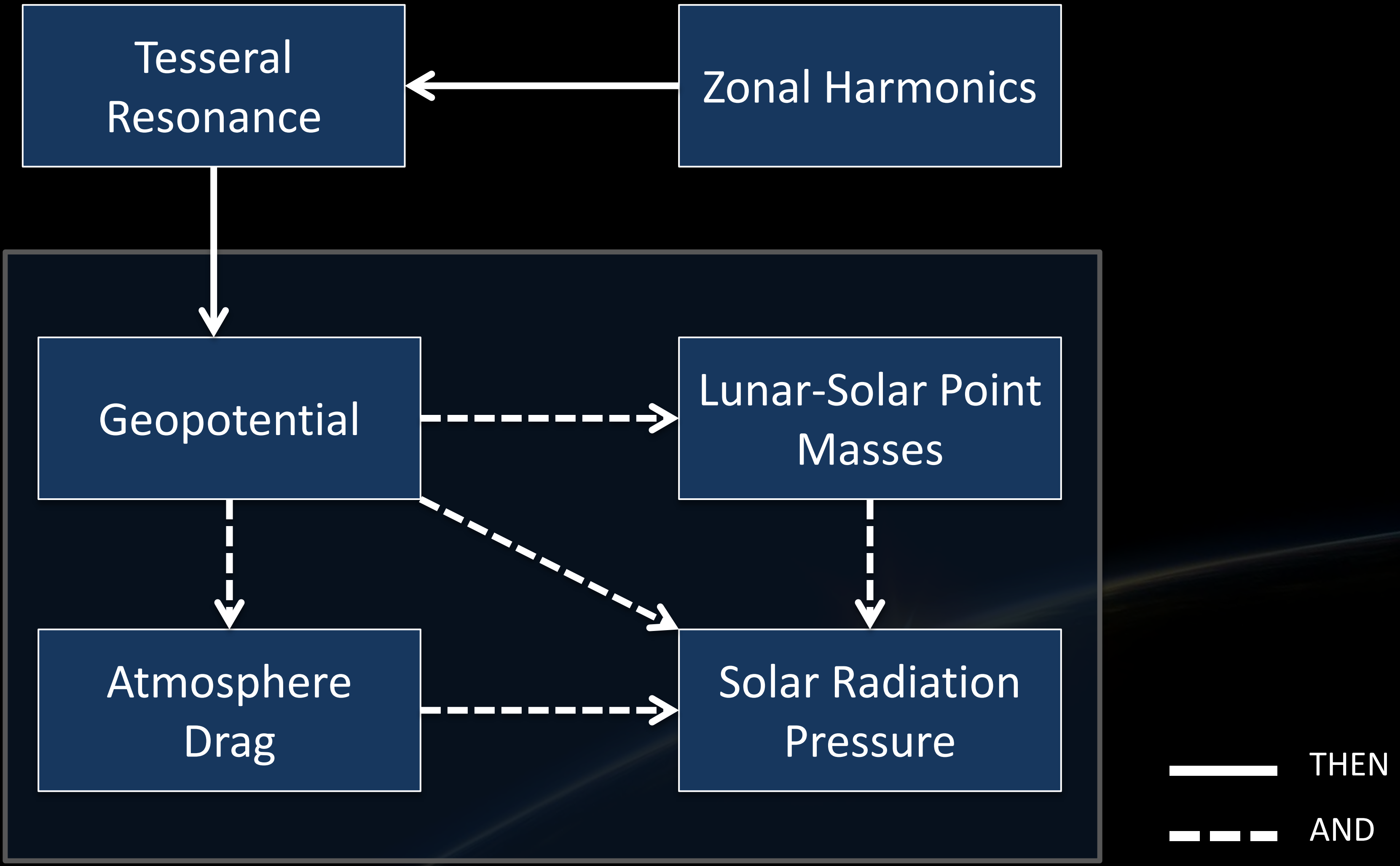
F77 DSST Standalone sequential decomposition of the mean element rates



Verification strategy

- Compare OREKIT against DSST F77 (DSST Standalone & GTDS DSST)
- Use interactive debuggers in Java and Fortran to locate any discrepancies
- Align physical models between systems
- Generate test cases for DSST functionality

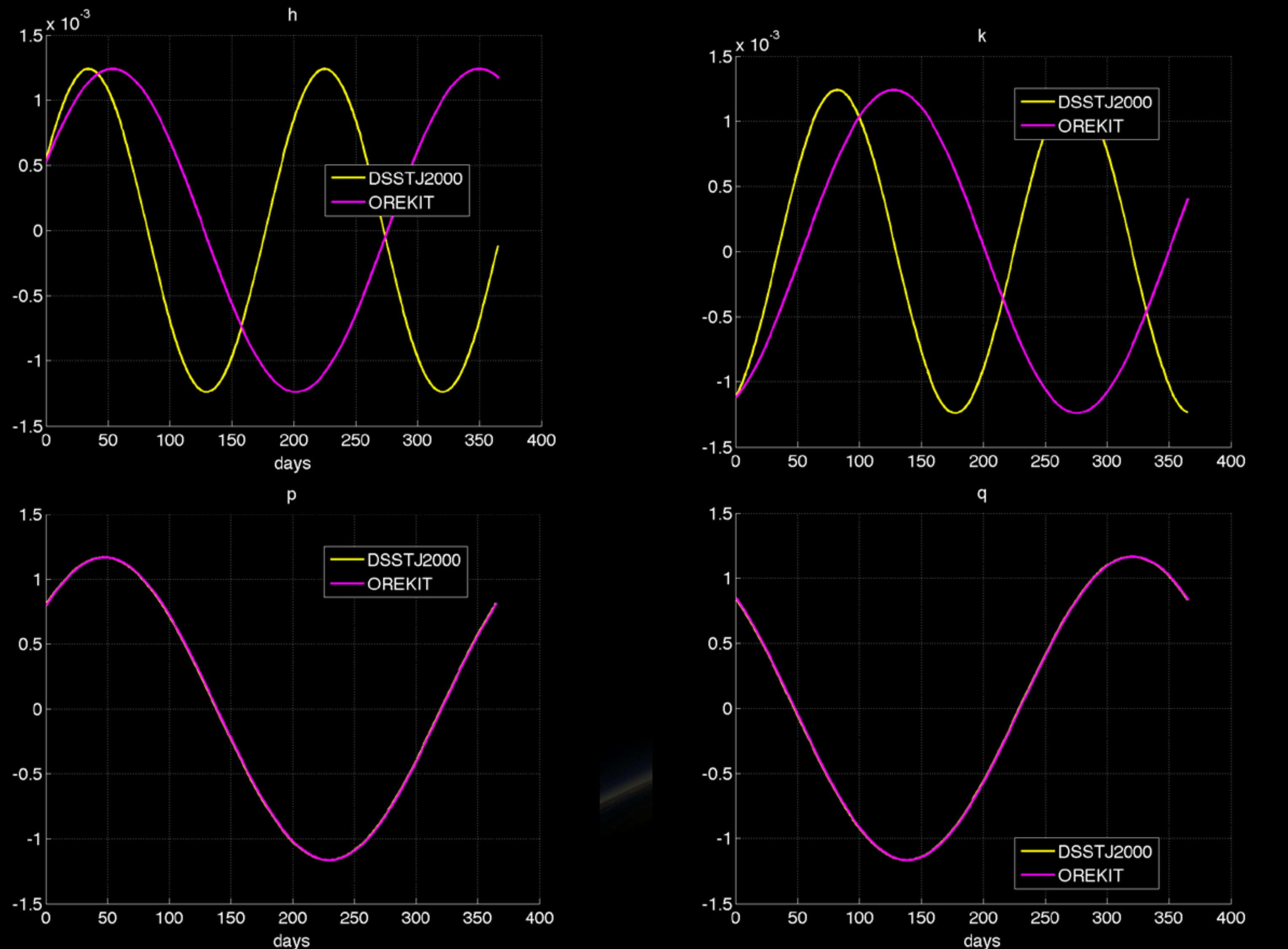
DSST test plan



Orekit DSST vs F77 DSST

- Input & output coordinate system: J2000
- Input parameters:
  - Mean Keplerian elements or
  - Mean Equinoctial elements (a, h, k, p, q, λ)
- Integration Coordinate System: J2000
- Integration Method: Runge-Kutta with Predictor-Corrector as backup
- Output: (1) mean Keplerian elements, (2) mean equinoctial elements, and (3) position and velocity

Test 1: LEO, 2 x 0
Equinoctial element histories over 365 days



Resolving the anomaly in the J2-only mean element case

- Comparison of debug output from Orekit routine DSSTCentralBody.java and from F77 DSST routines AVRANL/PZONAL
  - p dot and q dot approximately agree
  - h dot and k dot do not agree
- **Matlab** and **maxima** models for the equations of motion
- The partials of the potential wrt h and k are likely sources of the error

Danielsen Section 3.1, Eq (1): averaged eqs of motion for zonal harmonics

$$\frac{da}{dt} = 0$$

$$\frac{dh}{dt} = \frac{B}{A}\frac{\partial U}{\partial k} + \frac{k}{AB}(pU_{,\alpha\gamma} - IqU_{,\beta\gamma})$$

$$\frac{dk}{dt} = -\frac{B}{A}\frac{\partial U}{\partial h} - \frac{h}{AB}(pU_{,\alpha\gamma} - IqU_{,\beta\gamma})$$

$$\frac{dp}{dt} = -\frac{C}{2AB}U_{,\beta\gamma}$$

$$\frac{dq}{dt} = -\frac{IC}{2AB}U_{,\alpha\gamma}$$

$$\frac{d\lambda}{dt} = -\frac{2a}{A}\frac{\partial U}{\partial a} + \frac{B}{A(1+B)}(h\frac{\partial U}{\partial h} + k\frac{\partial U}{\partial k}) + \frac{1}{AB}(pU_{,\alpha\gamma} - IqU_{,\beta\gamma})$$

Partial derivatives of the potential due to zonals (Danielsen, Sect. 3, eq. 6)

$$
\begin{aligned}
\frac{\partial U}{\partial a} &= \frac{\mu}{a^2}\sum_{s,n}(2-\delta_{0s})(n+1)\left(\frac{R}{a}\right)^n J_n V_{ns} K_0^{-n-1,s} Q_{ns} G_s \\
\frac{\partial U}{\partial h} &= -\frac{\mu}{a}\sum_{s,n}(2-\delta_{0s})\left(\frac{R}{a}\right)^n J_n V_{ns} Q_{ns}\left(K_0^{-n-1,s}\frac{\partial G_s}{\partial h} + h\chi^3 G_s \frac{dK_0^{-n-1,s}}{d\chi}\right) \\
\frac{\partial U}{\partial k} &= -\frac{\mu}{a}\sum_{s,n}(2-\delta_{0s})\left(\frac{R}{a}\right)^n J_n V_{ns} Q_{ns}\left(K_0^{-n-1,s}\frac{\partial G_s}{\partial k} + k\chi^3 \frac{dK_0^{-n-1,s}}{d\chi}\right) \\
\frac{\partial U}{\partial \alpha} &= -\frac{\mu}{a}\sum_{s,n}(2-\delta_{0s})\left(\frac{R}{a}\right)^n J_n V_{ns} K_0^{-n-1,s} Q_{ns}\frac{\partial G_s}{\partial \alpha} \\
\frac{\partial U}{\partial \beta} &= -\frac{\mu}{a}\sum_{s,n}(2-\delta_{0s})\left(\frac{R}{a}\right)^n J_n V_{ns} K_0^{-n-1,s} Q_{ns}\frac{\partial G_s}{\partial \beta} \\
\frac{\partial U}{\partial \gamma} &= -\frac{\mu}{a}\sum_{s,n}(2-\delta_{0s})\left(\frac{R}{a}\right)^n J_n V_{ns} K_0^{-n-1,s}\frac{dQ_{ns}}{d\gamma} G_s
\end{aligned}
$$

Missing G_s term

Recursion for the derivatives of the Hansen coefficient kernels (Danielsen, Sect. 3.1, eq. 7)

$$\frac{dK_0^{-n-1,s}}{d\chi} = \begin{cases} 0 & \text{for } n = s \\ \dfrac{(1+2s)\chi^{2s}}{2^s} & \text{for } n = s+1 \\ \dfrac{(n-1)\chi^2}{(n+s-1)(n-s+1)}\left[(2n-3)\dfrac{dK_0^{-n,s}}{d\chi} - (n-2)\dfrac{dK_0^{-n+1,s}}{d\chi}\right] & \\ \quad + \dfrac{2}{\chi}K_0^{-n-1,s} & \text{for } n > s+1 \end{cases}$$

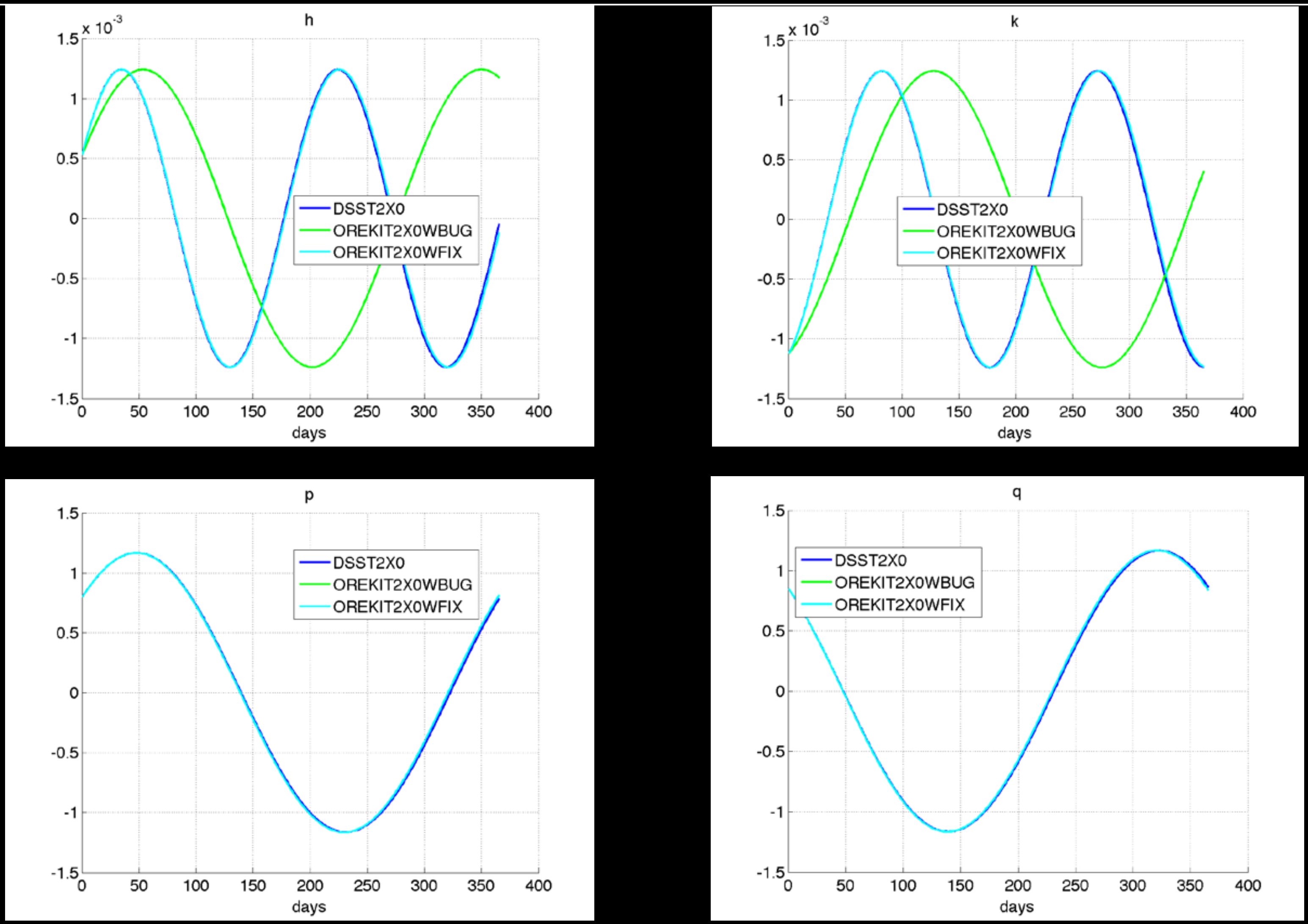
<u>Sign error</u>

- This recursion is obtained by differentiating the recursion for the Hansen kernels
- This algorithm is implemented in Orekit module **HansenCoefficients.java**
- Error in recursion verified by java debug

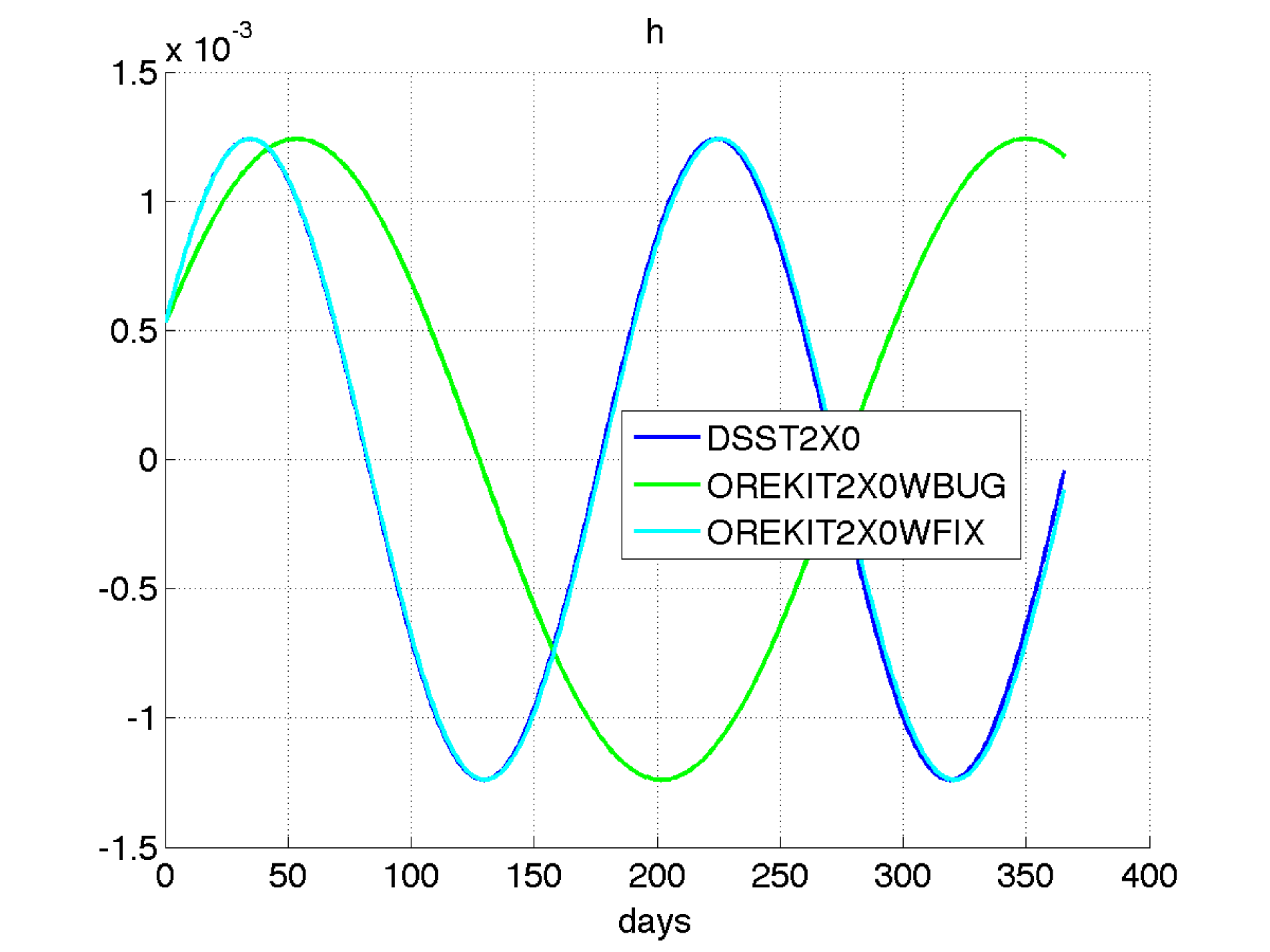
Modification to HansenCoefficients.java

```
private double computeHKDJ0NNegative(final int n, final int s) throws OrekitException {
    double dkdxns = 0.;
    final MNSKey key = new MNSKey(0, -(n + 1), s);

    if (n == FastMath.abs(s)) {
        HANSEN_KERNEL_DERIVATIVES.put(key, 0.);

    } else if (n == FastMath.abs(s) + 1) {
        dkdxns = (1. + 2. * s) * FastMath.pow(chi, 2 * s) / FastMath.pow(2, s);

    } else {

        final MNSKey keymN = new MNSKey(0, -n, s);
        double dkmN = 0.;
        if (HANSEN_KERNEL_DERIVATIVES.containsKey(keymN)) {
            dkmN = HANSEN_KERNEL_DERIVATIVES.get(keymN);
        } else {
            dkmN = computeHKDJ0NNegative(n - 1, s);
        }

        final MNSKey keymNp1 = new MNSKey(0, -n + 1, s);
        double dkmNp1 = 0.;
        if (HANSEN_KERNEL_DERIVATIVES.containsKey(keymNp1)) {
            dkmNp1 = HANSEN_KERNEL_DERIVATIVES.get(keymNp1);
        } else {
            dkmNp1 = computeHKDJ0NNegative(n - 2, s);
        }

        final double kns = getHansenKernelValue(0, -(n + 1), s);

        dkdxns = (n - 1) * chi2 * ((2. * n - 3.) * dkmN - (n - 2.) * dkmNp1) / ((n + s - 1.) * (n - s - 1.));
        dkdxns += 2. * kns / chi;
    }

    HANSEN_KERNEL_DERIVATIVES.put(key, dkdxns);
    return dkdxns;
}
```

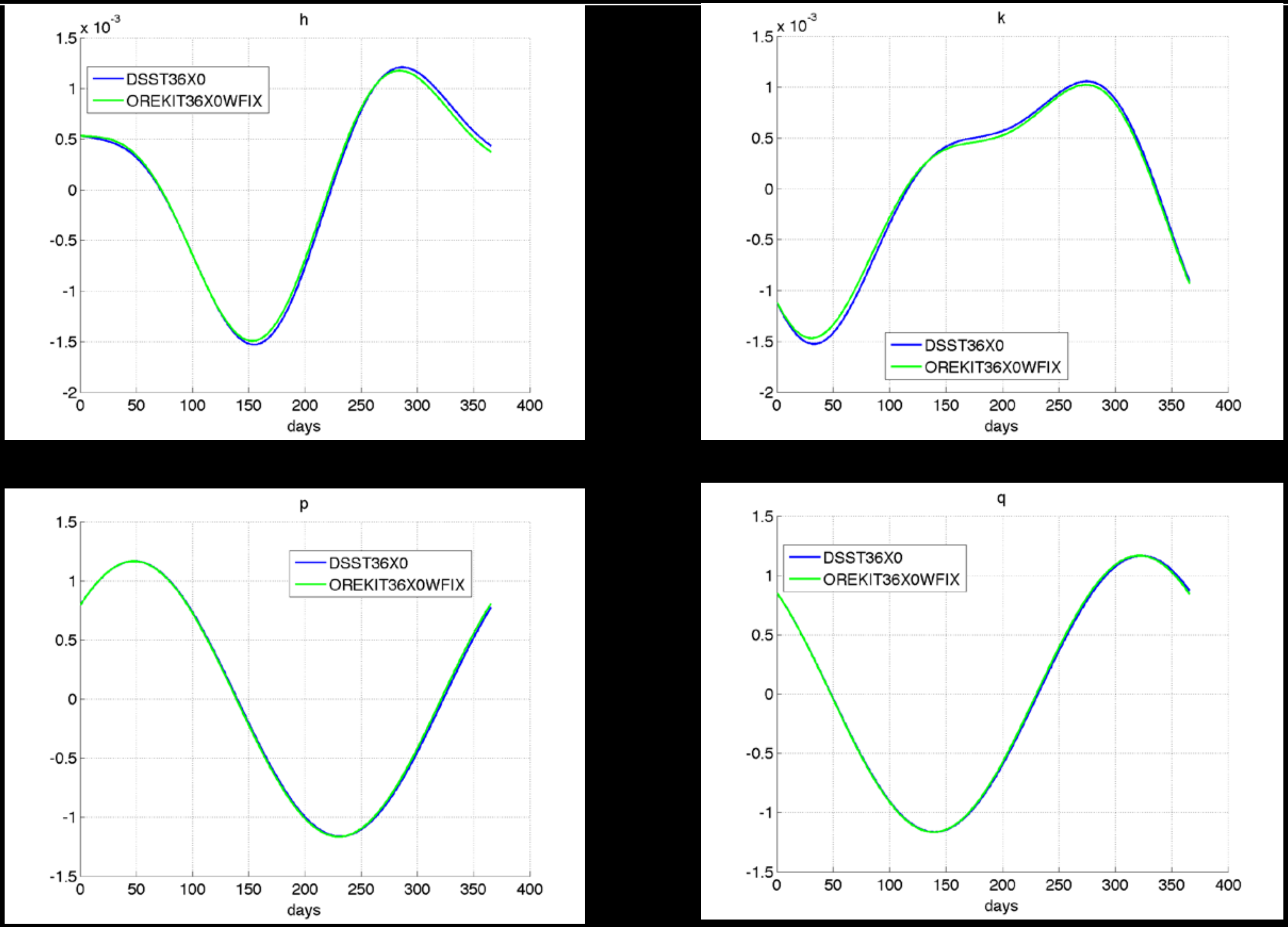
Test 1: LEO, 2 x 0 Equinoctial element histories with fix to Orekit module HansenCoefficients.java



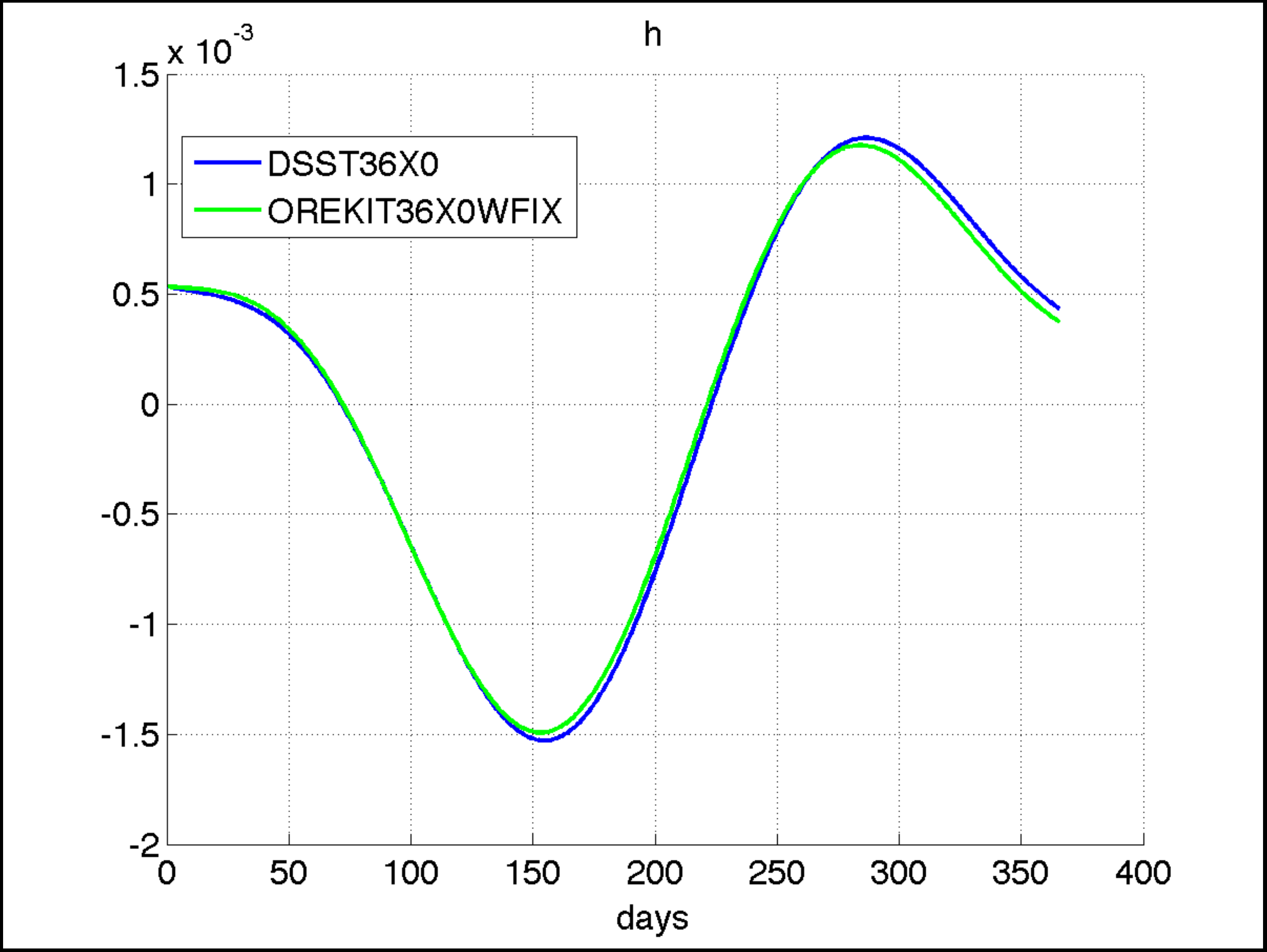
Drilling in on the equinoctial element h time history



Test 1: LEO, 36 x 0 Equinoctial element histories with fix to Orekit module HansenCoefficients.java



Drilling in on the equinoctial element h time history for the 36 x 0 case



Discussion of the Testing Process

- The capability to generate reference trajectories should be shared among the testing players (University of Cambridge, University at Buffalo, CS Systemes, MIT)
- The testing process must include review of the Danielsen document
- Explicit formulas for DSST functionality generated by symbolic algebra systems (Zeis, Kaniecki) can be important.
- It is desirable to create a symbolic algebra version of the Danielsen document
- Matlab can be very helpful
- Testers must be able to make their way through both the Orekit DSST java code and the F77 DSST code
- Testers must understand both the Eclipse java debugger and the Fortran 77 debugger

On-going work

- Debugging the J2-only case resulted in the identification of an error in a Hansen coefficient recursion; the software error propagated from an error in the NPG document
- Need to extend testing to the general zonal harmonic model
- Testing of the lunar-solar point mass model
- Testing of the tesseral resonance model
- Extend the Orekit DSST to include the short-periodic motion and the state transition matrix
- Improve documentation
- Improve the coordination between the Orekit DSST and F77 DSST in time and coordinate systems, physical models

Summary (OREKIT)

- OREKIT is a modern, open-source, extensible, and community-driven astrodynamics library, enabling:
  - Open collaboration
  - Algorithm development
  - Migration of legacy software
  - Standardisation and transparency in SSA

Summary (DSST)

- DSST is a computational efficient theory well suited to SSA
- Previously closed source, now updated and ported to OREKIT
- OREKIT implementation undergoing thorough audit and debugged
- Bugs detected early, and are being worked through to ensure accuracy and stability

Acknowledgements

- Zachary Folcik
- Romain Di Costanzo
- Pascal Parraud
- Luc Maisonobe
- Brian Weeden

Backup slides