Orbit prediction and ground station network design

JASON-2, NASA-CNES Ocean Surface Topography Mission

Daniel Gavrilov

Table of Contents

Introduction

This report explains the code produced for the coursework. References are often made to files and folders and they are surrounded by square brackets, e.g. `[/propagations/keplerian.js]`.

The easiest way to follow along is by browsing the code on Github:

https://github.com/danielgavrilov/satellites/tree/master/public/src

The second easiest way is using the PDF of the source code.

Language choice

All the code is written in JavaScript. This is not because JavaScript is particularly suitable for this type of task, but because JavaScript runs on every computer with a web browser. This means you can run the program with a click of a link:

http://gavrilov.co.uk/satellites/

Conventions

To make understanding the code easier to understand, some conventions are followed. Throughout the code:

- **All distances are in kilometres**, and **velocities** in **kilometres per second**.
- **All angles are in radians.** The only time angles are converted to degrees is when they are passed onto the plotting library, which works with degrees.
- **Vectors are 3 element number arrays**, since this is written in JavaScript, which does not have native types for vectors.

Code structure

The functions are categorised and grouped in folders, depending on their nature. There are four categories:

- **Propagations** `[/propagations]`
  Contains the three propagators.
- **Transforms** `[/transforms]`
  Contains functions for transforming between Keplerian and Cartesian elements, different coordinate systems, matrix rotations and transforming the data for plotting.
- **Utilities** `[/utils]`
  Contains utility functions for normalising angles, vectors operations, manipulating arrays, etc.
- **Plots** `[/plots]`
  Contains functions that produce plots and graphs.

There is also the root directory `[/]` which contains files that don't fit into any of these categories:

- **Constants** `[/constants.js]`
  Contains values like the average radius of the Earth, J2000 reference date, etc.
- **Controller** `[/controller.js]`
  The controller is responsible for initialising the visualisation and keeping it in sync by handling interaction events (like clicks and drags).
- **Data** `[/data.js]`
  Contains the initial orbital elements for four satellites (of which JASON-2 is one, the others are used for testing).
- **Events** `[/events.js]`
  Exposes a single common object that can be used to publish and subscribe to events from anywhere in the program.
- **Main** `[/main.js]`
  The entry point of the program. It imports and starts up everything, including the controller.

Some of these are explained in more detail below.

Propagations

All three types of propagators are in this folder. They all have an identical interface—they require the same parameters and produce the same type of output.

KEPLERIAN

The Keplerian propagator `[/propagations/keplerian.js]` is mostly straightforward. Given the initial conditions, it can give a prediction of the position of the satellite at an arbitrary time (although, not a very accurate one).

RUNGE-KUTTA 4

There are two Runge-Kutta 4 propagators: one with oblateness correction `[/propagations/rk4-j2.js]` and one without `[/propagations/rk4.js]`. The only difference between them is the calculation of the *k* value, hence in both files you will only see the `calc_k` function.

The `calc_k` function gets passed to an `rk4_generator` `[/propagations/rk4-generator.js]` to produce a propagator. This prevents the repetition of code, but makes the `rk4_generator` a bit harder to understand, as it is a *higher order function*. The key point is that any `calc_k` function can be injected and used inside the propagator.

Transforms

The `cart2kep` `[/transforms/cart2kep.js]` & `kep2cart` `[/transforms/kep2cart.js]` functions are straightforward as they are simply the equations from the notes expressed in JavaScript.

The **coordinate transforms** `[/transforms/coordinates.js]` contain functions for converting between earth-centred-inertial (ECI), earth-centred-earth-fixed (ECEF), east-north-up (ENU) and height, cross-track, along-tract (HCL) bases, as well as to and from latitude, longitude and height.

The **rotations** `[/transforms/rotations.js]` contain functions for rotating about X, Y and Z axes.

The **plots** `[/transforms/plots.js]` contain functions for transforming the data into parameters the plotting library accepts. The plotting library works with degrees, so all angles need to be converted before getting passed on, and it also requires GeoJSON geometries to plot lines and shapes on a geographic projection.

There are other convenience transform functions which are explained individually in the code.

Utilities

Utilities includes functions for manipulating **angles** `[/utils/angles.js]`, **arrays** `[/utils/arrays.js]`, **dates** `[/utils/dates.js]` and computing **vector operations** `[/utils/vectors.js]`.

Plots

There are five different plots/graphs implemented:

- **World map** `[/plots/world-map.js]` which shows the Earth in an equirectangular projection.
- **Station view** `[/plots/station-view.js]` which shows satellite tracks in an ENU basis in a flipped stereographic projection relative to a ground station station.
- **Rise and set times** `[/plots/rise-and-set.js]` which show the periods of time the satellite is visible from a specific ground station.
- **Multi-line series graph** `[/plots/multi-line.js]` which have been used to plot various quantities.

These plots are used throughout this report.

Task 1

Differences between Keplerian and Runge-Kutta 4 monopole prediction

The vector differences of positions and velocities by the two propagators are plotted below in an ECI frame:

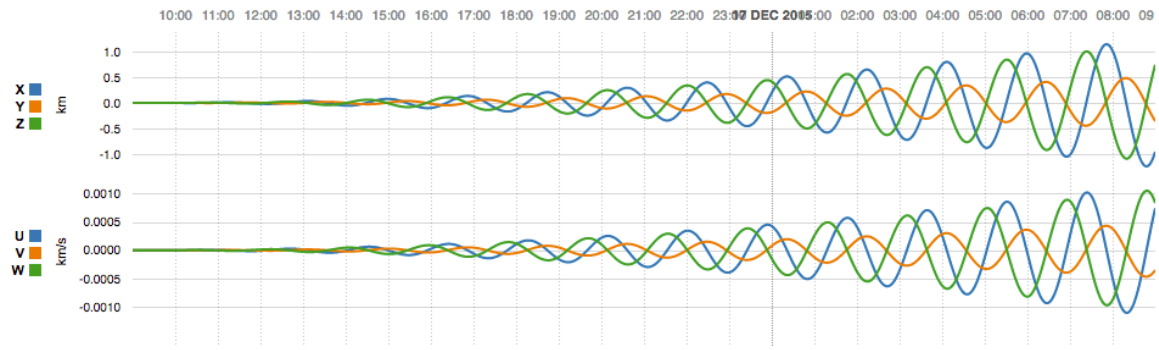


Figure 1 Differences between Keplerian and RK4 orbits in ECI over 24 hours

We can see that they oscillate with a steadily-increasing amplitude. The velocity difference is roughly phase-shifted by 90° compared to the position difference, due to the eccentricity of the orbit being close to 0, therefore the orbit being *almost* a perfect circle, making the position and velocity vectors *almost* normal to each other.

The oscillations are due to the rotation of the difference vector—on the opposite side of the Earth, the direction of the difference is switched. The differences can be better characterised in HCL frame:

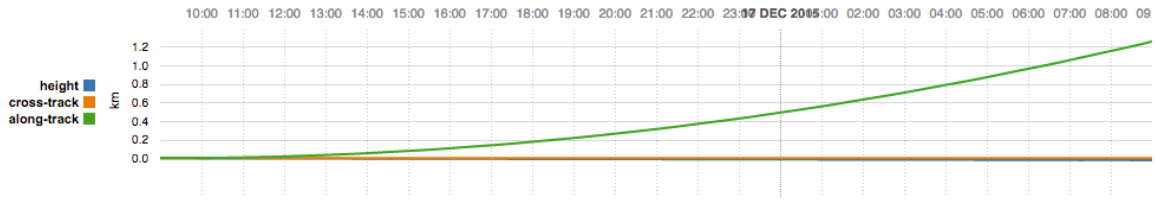
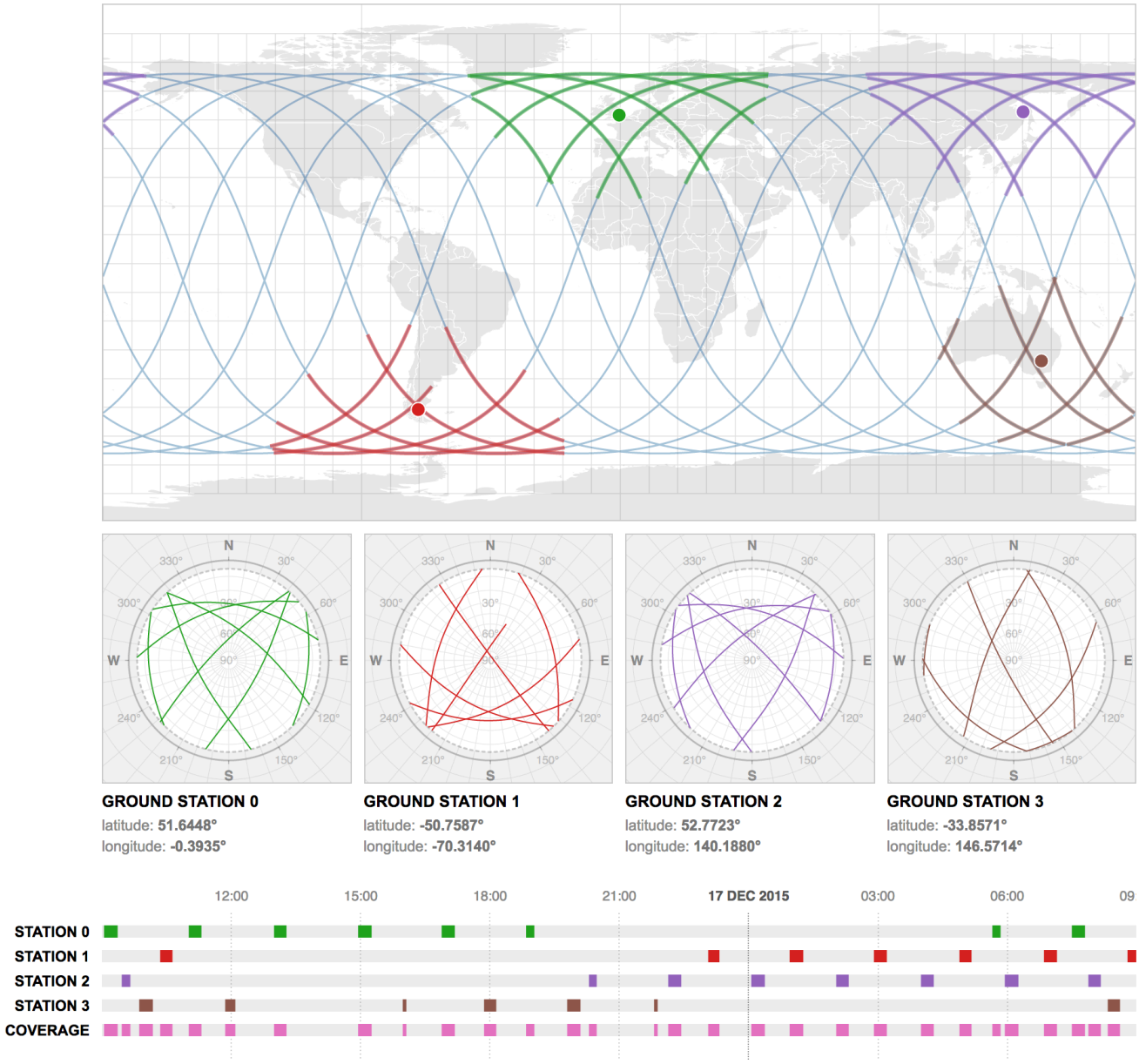


Figure 2 Differences between Keplerian and RK4 orbits in HCL over 24 hours

From this, it is clear that the two trajectories differ only *along-track*, which is most likely due to the quantisation of the numerical (Runge-Kutta) method. As it projects each time-step using the instantaneous position and velocity, it consistently makes a small error which aggregates over time. The position error grows to over 1.2 km over the course of 24 hours.

Ground station network



Above are the locations of the four ground stations, as well as their visibility trajectories over 24 hours. The locations were picked by hand—locations at higher latitudes had longer coverage, so higher latitudes were picked.

The visualisation shows the ground tracks on a world map, but also shows four station specific visibility trajectories which are plotted with azimuth and elevation.

Below them is a timeline showing intervals during which the satellite is visible from a specific ground station. The rise and set intervals are also presented in the table below:

STATION	RISE (S)	SET (S)	DURATION (S)	AZIMUTH (°)	ELEVATION (°)
0	180	1320	1140	-131.8572351	5.001807403
0	7260	8320	1060	-88.01944619	5.032241838
0	14380	15420	1040	-55.83758597	5.277829018
0	21400	22530	1130	-42.33134089	5.06765515
0	28380	29480	1100	-42.39320022	5.473180454
0	35430	36120	690	-58.15946322	5.212605755
0	74370	75060	690	135.2585139	5.271114166
0	81010	82110	1100	-165.5378659	5.213214793
1	4870	5900	1030	-135.1103153	5.008599173
1	50640	51580	940	18.08568854	5.110333172
1	57440	58580	1140	-34.11539433	5.359486574
1	64480	65560	1080	-80.37126695	5.227697395
1	71620	72630	1010	-118.0217241	5.356334295
1	78680	79770	1090	-136.9432676	5.413052129
1	85660	86390	730	-139.9339321	5.484474902
2	1660	2380	720	-56.6754093	5.101858438
2	40680	41330	650	131.5987355	5.139743444
2	47300	48390	1090	-168.588675	5.232703688
2	54240	55370	1130	-121.1532587	5.551502201
2	61330	62380	1050	-79.9047654	5.45230645
2	68410	69480	1070	-53.07535951	5.130678668
2	75410	76550	1140	-42.77492076	5.052080206
2	82390	83430	1040	-45.10731654	5.29748033
3	3120	4260	1140	-30.57986062	5.208306894
3	10300	11150	850	-89.03039499	5.086077746
3	25130	25430	300	171.8835437	5.098032604
3	31920	32940	1020	-165.4632693	5.115103404
3	38860	39970	1110	-146.3676454	5.524940257
3	46120	46420	300	-99.1297242	5.099413709
3	84010	85030	1020	8.380511941	5.353062753

All time measurements start from the initial epoch, which is in this case 16 December 2015 at 09:00:32.184000.

One inaccuracy that affects these results is the latitude/longitude to ECEF vector conversion. All four ground stations are assumed to be at a height of 0, however, the conversion model assumes a perfectly spherical earth.

Task 2

Denormalising the coefficient

The spherical harmonic coefficient c_{nm} was denormalised with the following formula:

$$\begin{bmatrix} c_{nm} \\ s_{nm} \end{bmatrix} = \left[\frac{(n-m)!\,(2n+1)(2-\partial m)}{(n+m)!} \right]^{1/2} \begin{bmatrix} \bar{c}_{nm} \\ \bar{s}_{nm} \end{bmatrix}$$

The calculation is below:

$$c_{20} = \left[\frac{(2-0)!\,(2\times 2+1)(2-1)}{(2+0)!} \right]^{1/2} \times \bar{c}_{nm} = \left[\frac{10}{2} \right]^{1/2} \times \bar{c}_{nm} = \sqrt{5} \times \bar{c}_{nm}$$

Differences between Runge-Kutta 4 with and without oblateness correction

The differences between the RK4 with and without oblateness correction is significant:

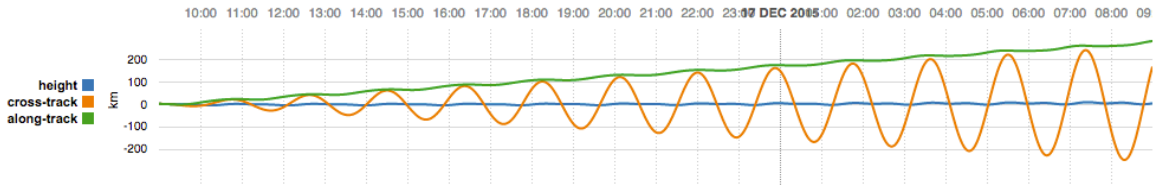


Figure 3 Differences between RK4 with and without oblateness correction in HCL over 24 hours

After 24 hours the along-track difference is 200 km, and the cross-track oscillates with the same amplitude of 200 km. The differences are even discernable on a world map:

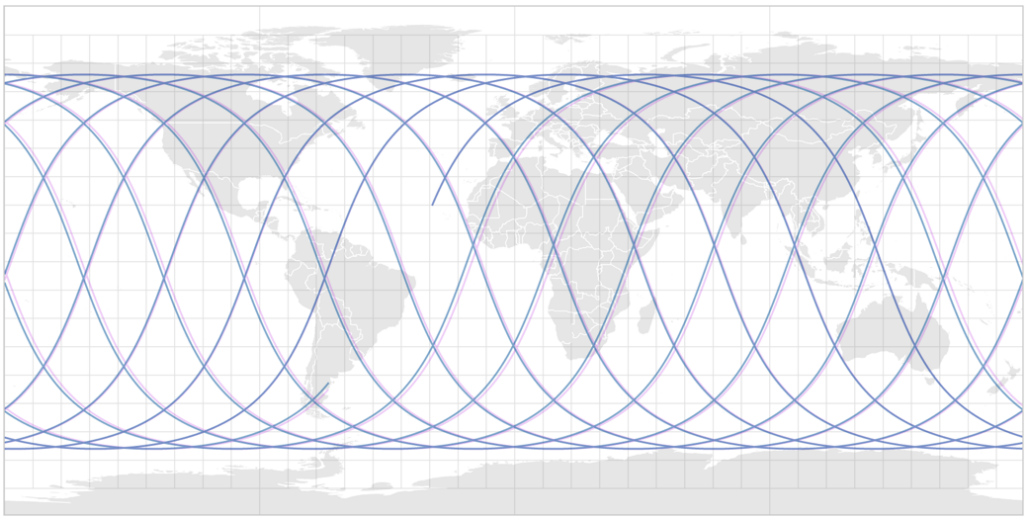


Figure 4 World map, showing RK4 with oblateness correction in blue and without oblateness correction in pink. Recorded over 24 hours.

The oscillating difference in *cross-track* is most likely due to the oblateness of the Earth. If we take into account oblateness, the direction of the force of gravity is not always to the centre of mass, but it oscillates as the satellite travels between the northern and southern hemispheres. This has the effect of changing (oscillating) the right ascension of the ascending node (Ω) of the orbit, which means the orbit does not lie on a single plane.

The increasing difference in *along-track* is also likely due to oblateness and the changing force of gravity, although this author hasn't managed to *quite* figure out why.

Correctness

The approach to checking the output of the functions is correct was to visualise as many outputs as possible and also to calculate the orbits to longer periods of time.

For example, the JASON-2 orbit repeats roughly every 10 days. A plot for 20 days:

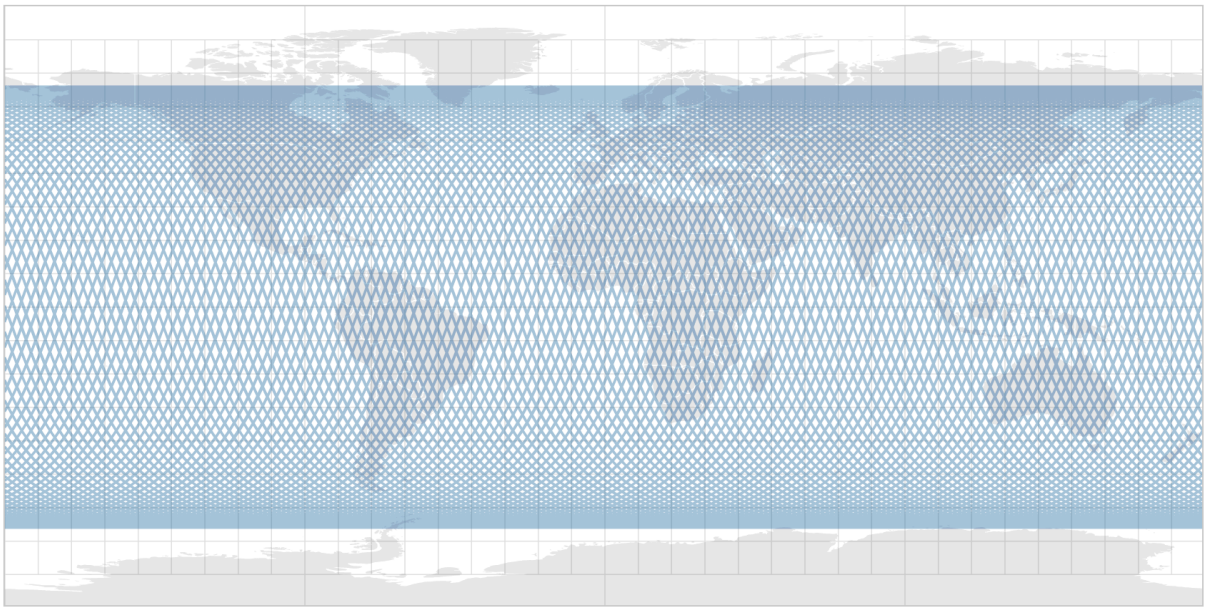


Figure 5 Ground tracks over 20 days for JASON-2 using RK4 with oblateness correction

If we zoom in, we can start to see some differences:

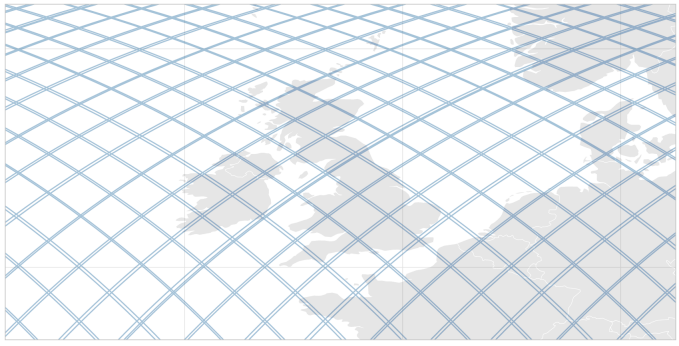


Figure 6 Zoomed in ground tracks over 20 days for JASON-2 using RK4 with oblateness correction

We can see the ground tracks don't *quite* repeat. This is most likely because we don't accurately model the oblateness of the Earth, and likely other effects.

Credits

Several JavaScript libraries were used to develop the program:

- **D3.js** — An SVG plotting library, every visualisation was made with this library.
- **TopoJSON** — A JavaScript library (and standard) for encoding and decoding topology in JSON.
- **Lodash** — A utility library that has useful functions for manipulating JavaScript arrays and objects.
- **Moment.js** — A library for parsing and manipulating dates.