### Ancillary Data Preprocessing Algorithm

#### Background/Introduction

The ancillary data preprocessing algorithm prepares the ancillary data provided by the spacecraft in the wideband data stream for use by subsequent geometric algorithms. This includes quality checking the incoming data to identify and remove outliers, applying scale factors from the CPF to convert the relevant ancillary data fields to engineering units, and processing the spacecraft attitude and ephemeris data to construct consistent attitude and ephemeris time histories for the data set. The baseline assumption is that the attitude and position/velocity estimates produced by the spacecraft will be sufficiently accurate to achieve L8/9 geolocation accuracy requirements. If this is the case, only basic quality checking and smoothing operations will be required. Since the ancillary data stream will also include the raw observable data used by the spacecraft to construct its attitude and ephemeris estimates, more sophisticated processing using the raw star tracker and SIRU data to construct a “definitive” attitude data set, and/or using the raw GPS pseudo-range and carrier phase observables to compute a “definitive” ephemeris data set, would be possible. These enhanced capabilities are currently considered a risk reduction contingency that would not be implemented unless needed.

The Landsat 8 Spacecraft to Ground Interface Control Document (70-P58230P, Rev C) defines the content and structure of the spacecraft ancillary data. This document clarifies several uncertainties regarding formats, coordinate systems, and sampling rates that required assumptions to be made in earlier versions of this algorithm description. For example, the rate at which the integrated spacecraft attitude estimates are provided was initially undecided. If the integrated spacecraft attitude estimate quaternions had been provided at a lower rate than the SIRU data, those estimates would have required densification using the raw SIRU data and its associated calibration parameters, status flags, and on-board bias, alignment, and scale estimates. Since the degree of smoothing that the SIRU measurements will be subjected to in the on-board attitude filter is, as yet, unknown, this algorithm still assumes that the raw SIRU measurements will be used to ensure that high-frequency content is preserved. The Smooth Euler and SIRU sub-algorithm will be used to perform this data blending and to replace quaternions flagged as outliers.

This algorithm was originally intended to support only imaging intervals that would be suitable for Level 1 processing – primarily Earth-view and lunar acquisitions. Subsequently, it was decided that ancillary data preprocessing would be valuable in other cases, particularly solar calibration intervals. Since these intervals tend to be quite short (only a few seconds), some special logic was added to allow processing to proceed under these conditions. These adjustments, mainly to the SIRU processing logic, are noted in the appropriate locations below.

#### Dependencies

The ancillary data preprocessing algorithm assumes that ancillary data for the full imaging interval, with (nominally) four seconds of extra data before and after the interval, is available to provide the required geometric support data, that a CPF containing the scale factors needed to convert the ancillary data to engineering units is available, and that the quality thresholds needed to detect and remove or repair outliers are provided either in the CPF or as processing parameters.

#### Inputs

The ancillary data preprocessing algorithm and its component sub-algorithms use the inputs listed in the following table. Note that some of these “inputs” are implementation conveniences (e.g., using an ODL parameter file to convey the values of and pointers to the input data; including data set IDs to provide unique identifiers for data trending).

|  |
| --- |
| **Algorithm Inputs** |
| ODL File (implementation) |
|  CPF File Name |
|  Relevant CPF contents: |
|  Ancillary data engineering unit conversion factors |
|  SIRU to ACS alignment matrix |
|  SIRU engineering unit conversion factors |
|  Leap Second Table |
|  Ancillary data thresholds and limits |
|  Orbital Radius Limits (nominal and max excursion) |
|  Ephemeris Angular Momentum Limits (nominal and max excursion) |
|  Quaternion normalization outlier threshold (max difference from 1) |
|  Level 0R Data Directory and File Root Name |
|  Relevant Level 0R spacecraft ancillary data contents: |
|  Spacecraft (S/C) time-tagged inertial to body quaternion estimate |
|  S/C time-tagged ephemeris estimate |
|  SIRU sampling delay (latency) estimate (see note #10) |
|  SIRU clock synchronization times – S/C clock |
|  SIRU clock synchronization times – SIRU clock |
|  SIRU time-tagged delta-angles |
|  SIRU status flags |
|  Output Preprocessed Ancillary Data File Name |
|  Acquisition Type (Earth, Lunar, Stellar, Cal) (optional, defaults to Earth) |
|  Trending on/off switch |
|  WRS Path/Row (for trending) |
|  Geometric Work Order Common Characterization ID (for trending) |
|  Work Order ID (for trending) |

#### Outputs

The following table contains the ancillary data preprocessing algorithm outputs. It is important to note that the algorithm outputs are independent of the details of SIRU/attitude data processing. Nor would the outputs change if any contingency definitive attitude and/or ephemeris capabilities were required. The ability to provide a stable interface at the output of this algorithm is a large part of the motivation for separating out these ancillary data validation and conversion preprocessing operations from the model creation logic.

|  |
| --- |
| Preprocessed Ancillary Data |
|  Attitude Data |
|  Attitude data UTC epoch: Year, Day of Year, Seconds of Day |
|  Time from epoch (one per sample, nominally 50 Hz) in seconds |
|  ECI2ACS quaternion (vector: q1, q2, q3, scalar: q4) (one per sample) |
|  ECEF2ACS quaternion (one per sample) |
|  Body rate estimate (roll, pitch, yaw rate) (one per sample) in radians/second |
|  Roll, pitch, yaw estimate (one per sample) in radians |
|  Ephemeris Data  |
|  Ephemeris data UTC epoch: Year, Day of Year, Seconds of Day |
|  Time from epoch (one per sample, nominally 1 Hz) in seconds |
|  ECI position estimate (X, Y, Z) (one set per sample) in meters |
|  ECI velocity estimate (Vx, Vy, Vz) (one set per sample) in meters/second |
|  ECEF position estimate (X, Y, Z) (one set per sample) in meters |
|  ECEF velocity estimate (Vx, Vy, Vz) (one set per sample) in meters/second |
| Trending Data |
|  WRS Path/Row |
|  Acquisition Date/Time |
|  Geometric Common Characterization ID |
|  Work Order ID  |
|  Ephemeris data start UTC time (year, day of year, seconds of day) |
|  Number of ephemeris data points  |
|  Number of out of limit ephemeris points |
|  Attitude data start UTC time (year, day of year, seconds of day) |
|  Number of attitude data points |
|  Number of out of limit attitude data points |

#### Options

Trending On/Off Switch

#### Prototype Code

Input to the executable is an ODL file; output is a HDF5 formatted preprocessed ancillary data file. Status messages and sample trending outputs are written to standard output. An ancillary data ASCII log file is also created to capture messages regarding detected data problems.

The prototype code was compiled with the following options when creating the test data files:

 -g -Wall –O2 -march=nocona -m32 –mfpmath=sse –msse2

The following text is a brief description of the main set of modules used within the prototype with each module listed, along with a very short description. It should be noted that not all library modules are referenced in the explanations below. The modules within the main “ancillary” directory of the prototype are discussed, and any library modules that were determined to be important to the explanation of either results, input parameters, or output parameters.

ancillary – Main procedure that retrieves the input parameters (using getpar), invokes the oli\_run\_preprocessing library module to perform ancillary data processing, and performs the final trending data output.

getpar – Retrieves the user-provided ODL parameters.

oli\_run\_preprocessing – Library routine that manages the output ancillary log file (anc.log), and invokes other library routines (see below) to load and process the spacecraft ancillary data from the Level 0R input.

oli\_ancillary\_log\_open – Opens the anc.log output file.

oli\_get\_header\_from\_l0r – Library routine that reads the L0R line headers to determine the scene start and stop times.

oli\_get\_ephemeris\_from\_l0r – Library routine that loads the ephemeris data from the input Level 0R data, quality checks the ephemeris using radius magnitude and angular momentum tests, corrects timing jitter (if any) in the ephemeris sample times, filters the ephemeris using a gravitational potential model, and creates the output ephemeris in both ECEF and ECI coordinates.

l8\_correct\_ephem\_time – Library routine that corrects timing jitter by comparing time differences to the corresponding position differences divided by the velocity.

l8\_kalman\_smooth\_gps – Library routine that uses a Kalman filter to smooth the ephemeris ensuring consistency with an Earth geopotential model.

geo\_earth\_second\_partial\_x – Compute gravitational acceleration for an ephemeris state vector, in the X direction.

geo\_earth\_second\_partial\_y – Compute gravitational acceleration for an ephemeris state vector, in the Y direction.

geo\_earth\_second\_partial\_z – Compute gravitational acceleration for an ephemeris state vector, in the Z direction.

geo\_ec2ic – Convert ECEF to ECI.

geo\_ic2ec – Convert ECI to ECEF.

exx\_calc\_gha – Compute Greenwich Apparent Sidereal Time (GAST), and the corresponding precession and nutation angles at the specified time.

exx\_precession – Apply the precession rotation to the input ECIJ2000 vector to convert it to ECI Mean Of Date (ECIMOD).

exx\_nutation – Apply the nutation rotation to the input ECIMOD vector to convert it to ECI True Of Date (ECITOD).

xxx\_rotatez – Apply the GAST rotation (including UT1-UTC offset) to convert ECITOD to ECEF of date (true pole).

exx\_polar\_motion – Apply the polar motion correction to convert ECEF of date to WGS84/mean ECEF (mean pole).

oli\_get\_attitude\_from\_l0r – Library routine that loads the attitude quaternion and SIRU data from the Level 0R input, windows the data to ensure that it falls within the ephemeris data interval, and creates an integrated output attitude data stream in both roll-pitch-yaw and quaternion form. This unit has been modified to support processing without SIRU data if the SIRU time synchronization process (see process\_siru\_times below) fails.

get\_siru2rpy – Construct matrix to convert 4 SIRU channel measurements to 3 orthogonal (roll-pitch-yaw) attitude measurements. Used only if SIRU processing is included.

process\_siru\_times – Process the SIRU clock and sync codes to create spacecraft epoch times for all SIRU samples. Used only if SIRU processing is included.

l8\_iru2acs – Library routine that applies the SIRU to spacecraft alignment correction (from the CPF) to convert SIRU data to the attitude control system coordinate system. Used only if SIRU processing is included.

l8\_sc\_attitude – Library routine that converts ECI quaternions to roll-pitch-yaw using ephemeris data.

l8\_init\_iru – Library routine that removes orbital motion, which is no longer simply a pitch thanks to off-nadir pointing and yaw steering, from the SIRU data for Earth acquisitions, using the ECI ephemeris. Used only if SIRU processing is included.

l8\_kalman\_smooth\_iru – Library routine that blends quaternion-derived absolute roll-pitch-yaw with SIRU-derived roll-pitch-yaw rates using a Kalman filter, to construct an integrated attitude data stream. Used only if SIRU processing is included.

l8\_movesat – Library routine that Lagrange interpolates the ephemeris data at the desired time.

geo\_eci2orb – Library routine that constructs the ECI-to-orbital rotation matrix from an ephemeris state vector.

euler2quat – Converts a rotation matrix to a quaternion.

oli\_ancillary\_log\_close – Closes the anc.log output file.

trend\_anc\_to\_database – Dummy routine to write the collected trending data to standard output. This would be replaced by a database call in the operational implementation.

#### Procedure

The primary tasks performed by the ancillary data preprocessing algorithm include the following:

1. Preprocess the ancillary ephemeris data:
	1. Load the spacecraft ephemeris data from the interval ancillary data stream.
	2. Validate the ephemeris points using orbital radius and angular momentum thresholds.
	3. Convert the ephemeris time codes from spacecraft time to a UTC time epoch at the first ephemeris data record time.
	4. Correct any time jitter in the ephemeris data samples.
	5. Repair any bad ephemeris points by interpolation/propagation.
	6. Perform a coordinate conversion to provide the ephemeris in both ECEF and ECI of epoch J2000.0 coordinates.
		1. Convert the incoming ECEF ephemeris state vectors to ECI J2000.
		2. Convert the ECI J2000 state vectors back to ECEF, removing the effects of Earth rotation from the velocity vectors, as described below.
2. Preprocess the ancillary attitude data:
	1. Load the spacecraft attitude data from the interval ancillary data stream.
	2. Validate the quaternion estimates by computing the magnitude of each and comparing it to 1.
	3. Window the attitude data to ensure that the attitude data are completely within the ephemeris data interval.
	4. Convert the attitude time codes from spacecraft time to a UTC time epoch at the first attitude data record time.
	5. Process the raw SIRU data time stamps to compute sample times relative to the spacecraft clock (if SIRU processing required). SIRU processing is suppressed if this process fails due to the lack of a valid SIRU time synchronization event in the ancillary data interval.
	6. Convert the raw SIRU integrated angle counts to angular rates (if SIRU processing is required and not suppressed).
	7. Rotate the SIRU data to the ACS coordinate system (if SIRU processing is required and not suppressed).
	8. Correct the SIRU data for the effects of orbital motion (Earth-view images only, lunar and stellar acquisitions are left in ECI). Only performed if SIRU processing is required and not suppressed.
	9. Convert the quaternions to roll, pitch, and yaw using the ECI ephemeris data.
	10. Filter the SIRU and quaternion-derived roll, pitch, and yaw values to generate an integrated roll, pitch, yaw and roll-rate, pitch-rate, yaw-rate attitude sequence at the full SIRU data rate. Note: This step will only be used if SIRU data processing is required. If SIRU data processing is not required or is suppressed, attitude estimates flagged as outliers will be replaced by linear interpolation.
	11. Convert the roll, pitch, yaw values to ECI quaternions using the ECI ephemeris data.
	12. Convert the roll, pitch, yaw values to ECEF quaternions using the ECEF ephemeris data.
3. Create the output ephemeris and attitude data set containing:
	1. Attitude Data
		1. Attitude interval UTC epoch as Year, Day of Year, Seconds of Day.
		2. Attitude sample time offsets from the UTC epoch (in seconds) – one per sample. There will nominally be 50 samples per second.
		3. Body/ACS to ECI quaternions (vector part q1, q2, q3, and scalar part q4) – one set per sample.
		4. Body/ACS to ECEF quaternions (vector part q1, q2, q3, and scalar part q4) – one set per sample.
		5. Body inertial rotation rates (roll rate, pitch rate, yaw rate) in radians/second – one set per sample.
		6. Roll, pitch, and yaw in radians – one set per sample.
	2. Ephemeris Data
		1. Ephemeris interval UTC epoch as Year, Day of Year, Seconds of Day.
		2. Ephemeris sample time offsets from the UTC epoch (in seconds) – one per sample. There will nominally be one sample per second.
		3. ECI X, Y, Z position in meters – one set per sample.
		4. ECI X, Y, Z velocity in meters/second – one set per sample.
		5. ECEF X, Y, Z position in meters – one set per sample.
		6. ECEF X, Y, Z velocity in meters/second – one set per sample. Note that these are actually ECI velocities rotated into the ECEF coordinate system, not true ECEF velocities that would include Earth rotation velocity.

Steps 1.a., 2.a., and 3 above are input/output functions and are not described further here. The remaining steps are described in detail in the sub-algorithms below.

##### Convert Spacecraft Time Code to UTC

The convert spacecraft time code to UTC is a general-purpose sub-algorithm that is used by the more specific ancillary data preprocessing sub-algorithms below. Spacecraft time codes are TAI offsets from the J2000 epoch. Since TAI and UTC differ only by leap seconds, the conversion to UTC amounts to a leap second correction. The spacecraft (J2000) epoch UTC date/time is hard coded (in a #define statement) to prevent it being changed inadvertently. See note #6 for more explanation.

1. Load the leap second table from the CPF. The leap second table is represented as the date that each leap second since 01 January 1972 was declared.
2. Scan the leap second table and compute the number of leap seconds prior to the J2000 spacecraft epoch.
3. Scan the leap second table and compute the number of leap seconds prior to the current spacecraft date/time. This is done by converting the spacecraft time code (TAI offset from J2000) to UTC (without any leap second correction) and then using the resulting UTC date to determine the number of leap seconds.
4. Subtract the leap second total for the spacecraft J2000 epoch (result of step 2) from the leap second total for the time code (result of step 3) to compute the number of leap seconds from the spacecraft epoch to the current spacecraft time. The resulting number of leap seconds since J2000 is stored the first time it is computed and used in each subsequent time code to/from UTC conversion operation.
5. Subtract the number of leap seconds since J2000 from the current spacecraft time code.
6. Add the adjusted time code to the UTC date/time for the spacecraft J2000 epoch to yield the UTC date/time corresponding to the spacecraft time code.

###### ECI to/from ECEF Coordinate Transformation

The transformation from ECI of epoch J2000 (mean equator and equinox of J2000.0) to ECEF (WGS84) coordinates is a time-varying rotation due primarily to the Earth’s rotation, but it also contains more slowly varying terms for precession, astronomic nutation, and polar wander. The ECI-to-ECEF rotation matrix can be expressed as a composite of these transformations:

**T**ecef/eci = **A B C D**

**A** = polar motion

**B** = sidereal time

**C** = astronomic nutation

**D** = precession

***Polar Motion***

The polar wander correction performs the transformation from the Earth's true spin axis (in the Terrestrial Intermediate Reference System) to the mean pole (in the International Terrestrial Reference System, or ITRS, here taken to be identical to WGS84). The polar motion corrections are tabulated in the CPF. The corrections for the current day are looked up from the CPF table and applied as described in section 6.5.2 of:

Kaplan, George H., United States Naval Observatory Circular No. 179, “The IAU Resolutions on Astronomical Reference Systems, Time Scales, and Earth Rotation Models - Explanation and Implementation,” U.S. Naval Observatory, Washington, D.C., October 20, 2005. This document will henceforth be referred to as Circular 179. This transformation is implemented using the *wobble* function in the NOVAS C3.1 library provided by the Naval Observatory.

***Sidereal Time***

The sidereal time correction performs the transformation from the inertial true-of-date system (true equator and equinox of date) to the Earth-fixed true-of-date (true pole or terrestrial intermediate reference) system. It applies the polar rotation due to GAST, as described in Circular 179. We use the “Equinox-Based” approach described in the Circular and implemented in the *sidereal\_time* function of NOVAS C3.1. Note that the sidereal time computation includes the time correction from UTC to UT1 for the current day. The “current day” would be defined by the data set UTC epoch (rather than being evaluated for each ephemeris or attitude point) to avoid the possibility of introducing leap seconds in the middle of an imaging interval. This correction is tabulated in the CPF, along with the polar wander corrections.

***Nutation***

The nutation correction performs the transformation from the inertial mean-of-date system (mean equator and equinox of date) to the inertial true-of-date system through nutation angles. The nutation model is based on the IAU 2000 theory of nutation, as described in Circular 179 and implemented in the *nutation* function of NOVAS C3.1.

***Precession***

The precession correction transforms the inertial of epoch J2000.0 system to the inertial mean-of-date system. The precession model is based on the IAU 2000 definition, as described in Circular 179 and implemented in the *precession* function of NOVAS C3.1. Note that we do not apply the (small) frame bias correction defined in Circular 179 because our target inertial coordinate system is the inertial system of epoch J2000 (ECIJ2000.0) rather than the Geocentric Celestial Reference System (GCRS) described in the Circular.

This transformation rotates a vector from the ECI J2000.0 system to the Earth-fixed (WGS84) system. For example, an ECIJ2000 position vector is converted to ECEF as follows:

Xecef = **T**ecef/eci Xeci = **A B C D** Xeci

When working with ephemeris state vectors containing both position and velocity terms, there can be ambiguity in the treatment of the velocity terms when converting between Earth-fixed and inertial coordinates. This ambiguity arises because the transformation is itself time varying. Taking the time derivative of the equation above yields the following:



The second term captures the time-varying effect of the transformation itself. The time-varying effects of precession, nutation, and polar motion transformations are negligible when compared to the orbital motion of a spacecraft, but the sidereal time transformation contributes a significant effect. Keeping this in mind and expanding **T**ecef/eci above yields the following:



 Where thematrix is defined as the following:



With: \* = Earth rotation rate in precessing reference frame

 GAST = Greenwich apparent sidereal time

For useful figures and additional explanation of this transformation, please reference DMA TR8350.2-A, “Supplement to the Department of Defense World Geodetic System of 1984 Technical Report – Part I: Methods, Techniques, and Data Used in WGS 84 Development,” Defense Mapping Agency (now National Geospatial Intelligence Agency), 1 December 1987.

This equation shows that the ECEF velocity is composed of the ECI velocity rotated into the ECEF coordinate system (the first term) plus the effect of Earth rotation (the second term). Note that Earth rotation is modeled by the rate of change of the sidereal time transformation () applied to the (ECI true-of-date) position vector.

Whether or not the second term (Earth rotation) is included in the ECI to ECEF velocity transformation depends upon the intended purpose. The original ECEF ephemeris data received from the spacecraft contains velocity estimates that include the Earth rotation effects (i.e., “true” ECEF state vectors). The Earth rotation term must therefore be taken into account when converting these state vectors to ECI J2000. This coordinate system conversion is used to accomplish step 1.f.i above. For most applications within the geometric model, however, we are only interested in the velocity vector as a direction in inertial space (e.g., when using position and velocity to define the orbital coordinate system, which is the attitude control system reference). In this case, we only want the first term – the inertial velocity rotated to ECEF coordinates. Therefore, we rotate the ECI J2000 ephemeris state vectors back to “pseudo” ECEF coordinates without including the Earth rotation term.

To summarize, the ECI/ECEF coordinate system transformations applied to the incoming ephemeris data from the ancillary data file are as follows:

ECEF to ECI:





Noting that the **A**, **B**, **C**, and **D** matrices are orthogonal so that their inverses are equal to their transposes.

ECI to (pseudo) ECEF:





Noting that the position vector is the same as the original value, but the velocity vector is not, as indicated by the prime notation.

##### Correct Ephemeris Sub-Algorithm

The correct ephemeris sub-algorithm performs steps 1.b., 1.c., and 1.d. above. This sub-algorithm quality checks the ephemeris data and corrects any timing jitter errors in the ephemeris solution. The ephemeris values are used to calculate satellite position in the WGS84 ECEF frame.

1. Extract the ephemeris data records from the ancillary data.
2. Search the ancillary data ephemeris records and find the first and last valid ephemeris records in the interval. Extract the time tags for these records.
3. Set the ephemeris epoch to the time associated with the start index found in step b) converted to UTC (see Convert Spacecraft Time Code to UTC sub-algorithm above). Retain the corresponding epoch spacecraft time, as it will be subtracted from the other ephemeris samples.
4. Loop on ephemeris, starting at and ending at indexes found in b.
	1. Set ephemeris sample time to the time code from ancillary data minus the ephemeris start time code, i.e., convert times to offsets from the ephemeris epoch defined in c).
	2. Convert the ECEF ephemeris position and velocity vectors to ECI J2000 so that the angular momentum check, and subsequent ephemeris smoothing algorithms, can operate in inertial space.
	3. Get angular momentum and orbital radius nominal values and allowable deviation thresholds from the CPF: angmo\_nom, angmo\_delta, orbrad\_nom, orbrad\_delta.
	4. Calculate orbital radius to compare against threshold:

radius = | p |

Where: p = ephemeris position vector

* 1. Calculate angular momentum of ephemeris to compare against threshold:



where:

 *p* = satellite positional vector

 *v* = satellite velocity vector

* 1. Check orbital radius and angular momentum against nominal values and thresholds from CPF:

 If | radius – orbrad\_nom | <= orbrad\_delta AND

 | angular momentum – angmo\_nom | <= angmo\_delta

Then store the ephemeris point for processing.

Otherwise, report the bad ephemeris point as an outlier.

If fewer than four (the minimum required to support Lagrange interpolation) valid ephemeris points are found, a fatal error is returned for Earth-view, lunar, and stellar acquisitions. For solar cal acquisitions, additional ephemeris points are propagated using the process model described in the Smooth Position and Velocity Sub-Algorithm below, until four points are available.

The ECI ephemeris is reinterpolated, using the following method, to remove any small time jitters that are present.

Let vx, vy, and vz be the measured velocity.

Let px, py, and pz be the measured position.

1. Loop through the ephemeris points (i = 0 to N-1), computing the distances between adjacent points:

If first ephemeris point (i = 0) set d0 = 0.

If ephemeris point is not first value, then

1. Calculate difference in ephemeris from point i and i-1

dxi = pxi - pxi-1

dyi = pyi - pyi-1

dzi = pzi – pzi-1

1. Calculate magnitudes of the delta position and the velocity vectors

si = sqrt( dxi \* dxi + dyi \* dyi + dzi \* dzi)

svi = sqrt( vxi \* vxi + vyi \* vyi + vzi \* vzi )

1. Calculate difference between the “predicted time” from the magnitudes calculated in a2 and the time measured difference between ephemeris points i+1 and i

Set di = si / svi – ephemeris timei + ephemeris timei-1

1. Calculate average difference of time differences from a.

Let 

Where N = number of ephemeris points

1. Loop through the ephemeris points, adjusting times by the “predicted time difference” (a3) and the average of “predicted time difference” (b).

ephemeris timei = ephemeris timei + di – *avg*

Using Lagrange interpolation, calculate satellite position and velocity at one-second intervals, correcting any sampling timing irregularities and filling in any missing outlier points. The time-adjusted satellite position and velocity from the previous step are taken as input.

1. Loop on ephemeris values
2. Calculate ephemeris interpolation time for current interval (multiple of one second).
3. Convert ephemeris time from seconds to year, day of year, and seconds of day.
4. Bracket ephemeris data for interpolation (4 valid points are needed)
	1. Use 2 points before and 2 after the interpolation time.
	2. If that would require points beyond the beginning or end of the ephemeris interval, use the first four or the last four points in the interval.
5. Interpolate ephemeris to current time (a1) using Lagrange interpolation and bracketed values (a3).

Use the Smooth Position and Velocity sub-algorithm (see below) to smooth the ECI ephemeris. It is then converted to ECEF so that the ECI and ECEF ephemeris representations are consistent (step 1.f.ii. above).