UNIVERSIDAD DE LA RIOJA

Scientific Computing Group

The differential algebra least squares toolbox

User manual

Matteo Losacco

Keywords: differential algebra, astrodynamics, orbit determination

Consultancy service on the study of future algorithms and procedures for the Spanish Space Surveillance and Tracking Operation Centre (S3TOC). Set 1 - Orbit determination (OD) CDTI. (Deimos Space and the University of La Rioja).

Contents

Abstract

This manual provides a detailed illustration of the Differential Algebra based Least Squares (DALS) toolbox. The software allows the user to estimate the state and associated covariance matrix of a transiting object detected by a radar or optical sensor starting from a set of measurements and an apriori estimate of the state of the object. The estimation is done with a least squares process formulated in the DA framework and can be done by relying on either a numerical or a semi-analytical propagator. The output of the program is an orbit parameter message (OPM) file, which contains the obtained state estimate, the associated covariance matrix and, if required, an estimate of two object parameters, namely the"Bfactor" parameter and the solar radiation pressure coefficient (SRPC).

The manual is organized as follows. Chapter 1 describes in details the input files and utilities required by the program and gives an example of a possible output OPM. Chapter 2 describes the two available propagators, namely the numerical Accurate Integrator for Debris Analysis (AIDA) and the semi-analytical SADA software, whereas Chapter 3 illustrates the performance function of the estimation process. Chapter 4 describes in details all the steps performed by the DALS tool, from the software initialization to the OPM generation, whereas Chapter 5 finally illustrates how to build and install the toolbox.

1. Input and output files

This chapter provides a detailed illustration of the input and output files for the DALS toolbox. Section 1.1 describes the four input files required by the tool, whereas Section 1.2 illustrates the single output of the process.

1.1. Input files

The DALS toolbox requires four input files: a JSON environment file; a JSON input file, summarizing all the variables and data required by the process; a set of tracking data message (TDM) files, which contain all the epochs and measurements collected by the set of sensors considered; a .txt file including the epoch and the available a priori state estimate for the observed object and, if any, the associated state covariance matrix, the "Bfactor" coefficient and the solar radiation pressure coefficient ("SRPC"). The JSON environment file points to a set of SPICE kernel file and a space weather file that shall be included. In case of selection of the AIDA dynamical model, the gravitational model files addressed in the JSON input file shall be included too. Details about these extra files are given in Section 1.1.1 and 1.1.2. In addition, the external SPICE utility "pinpoint" is required. A detailed description is provided hereafter.

1.1.1. The JSON environment file

The JSON environment file contains the location of essential files required by the DALS process. Two objects are present:

- ["spiceKernels"]: path of the .txt file listing all the basic SPICE kernels required by the tool. The list of kernels shall contain:
  - naif0012.tls: leapseconds kernel file.
  - pck0010.tpc: orientation and size/shape data for natural bodies.
  - gm_de431.tpc: mass parameters for planets and satellites.
  - de432s.bsp: planetary ephemeris.
  - MOD.tf: mean equator and equinox of date frame kernel.
  - TOD.tf: true equator and equinox of date frame kernel.
  - ECLIPJ2000_DE405.tf: ecliptic J2000 frame kernel.

  The leapseconds kernel is the only non-constant kernel. The latest version can be found on the NASA SPICE website.
- ["spaceWeatherFile"]: path of the space weather file. Also this file shall be continuously updated, and can be downloaded from the CELESTRAK website.

1.1.2. The JSON input file

The JSON input file collects all the variables required by the DALS tool and provides the paths of extra input files required by the process. An example of JSON input file is given in Fig. 1.1. The file can be ideally divided into six blocks (or "objects"):

- ["DALS"]: definition of all the control parameters for the DALS tool.

- ["Dynamics"]: orbital dynamics selection and perturbation flags setup.
- ["Earth"]: Earth-fixed reference frame declaration and definition of all the required extra SPICE kernels.
- ["Measurements"]: TDM file names declaration.
- ["OD"]: state estimates file name declaration and flags for data availability definition.
- ["Observer"]: observer parameters setup.

The ["DALS"] block contains all the control parameters for the DALS solver, along with the data required for the generation of the orbit parameters message (OPM) output file. An example is given in Fig. 1.1a. The data included are the following:

- ["DALS"]["order"]: DA expansion order. The value must be greater or equal to 2.
- ["DALS"]["max_iter"]: maximum number of iterations for the DALS process. A value between 10 and 20 is suggested.
- ["DALS"]["date_est"]: the epoch at which the state estimate of the DALS is computed, expressed in UTC. This input is generally set equal to the epoch of the available a priori estimate, but can be arbitrarily selected by the user.
- ["DALS"]["flag_Bfactor_est"]: if set equal to "true", it allows for the estimation of the "Bfactor" parameter, which is defined as

$$\text{Bfactor} = C_D \frac{A}{m} \tag{1.1}$$

  where C_D is the drag coefficient, A the cross sectional area and m the object mass. On the contrary, if the flag is set equal to "false", the value provided in the state estimates file is maintained fixed. Note that, should the flag ["OD"][flag_Bfactor_av] be equal to "false" (i.e., no Bfactor estimate is provided in the state estimates file), ["DALS"]["flag_Bfactor_est"] is automatically set equal to "true".
- ["DALS"]["flag_SRPC_est"]: if set equal to "true", it allows for the estimation of the "SRPC" parameter during the DALS process, defined as

$$\text{SRPC} = \frac{A}{m}(1+\varepsilon) \tag{1.2}$$

  where ε is the object reflectivity. On the contrary, if the flag is set equal to "false", the value provided in the state estimates file is maintained fixed. Also in this case, if the flag ["OD"][flag_SRPC_av] is equal to "false" (i.e., no SRPC estimate is provided in the state estimates file), ["DALS"]["flag_SRPC_est"] is automatically set equal to "true".
- ["DALS"]["fun_tol"]: residuals tolerance. It is expressed as

$$\frac{|r_{(k)} - r_{(k+1)}|}{1 + |r_{(k)}|} \tag{1.3}$$

  where r represents the objective function (the norm of the measurements residuals, see Chapter 3), whereas the subscripts indicate the iteration level. It is used as one of the stopping criteria for the DALS process (see Chapter 4).
- ["DALS"]["step_tol"]: stepsize tolerance. It is expressed as

$$\frac{||\boldsymbol{x}^a_{(k)} - \boldsymbol{x}^a_{(k+1)}||}{1 + ||\boldsymbol{x}_{(k)}||} \tag{1.4}$$

  where $\boldsymbol{x}^a$ is the state (or augmented state) vector. It is used as one of the stopping criteria for the DALS process (see Chapter 4).

- ["DALS"]["grad_tol"]: tolerance on the residuals gradient. It is used as one of the stopping criteria for the DALS process (see Chapter 4).
- ["DALS"]["map_tol"]: DA map accuracy tolerance (see Chapter 4 for further details).
- ["DALS"]["stop_crit"]: stopping criterion. If set to "relative", DALS is run by considering residuals, stepsize and DA map tolerances. If "gradient" is selected, gradient and DA map tolerances are considered.
- ["DALS"]["output"]: data and flags for the generation of the output OPM file. The object includes
  - ["DALS"]["output"]["path"]: OPM file path.
  - ["DALS"]["output"]["name"]: OPM file name.
  - ["DALS"]["output"]["obj_ID"]: object ID (if known).
  - ["DALS"]["output"]["obj_name"]: object name (if known).
  - ["DALS"]["output"]["flag_Cov"]: flag for printing the estimated state covariance matrix ("true" or "false").
  - ["DALS"]["output"]["flag_KeplPar"]: flag for printing the estimated osculating Keplerian parameters ("true" or "false").

The ["Dynamics"] object contains all the data required for selecting and setting up the orbital propagators. It contains three mandatory entries, one of which changes according to the selected method. An example is given in Fig. 1.1b. The entries are

- ["Dynamics"]["method"]: "AIDA" or "SADA" (see Chapter 2).
- ["Dynamics"]["tolerance"]: integration tolerance. The value must be lower or equal to 1e-11.
- ["Dynamics"]["AIDAparam"]: this object is included if the "AIDA" propagator is selected. The object contains
  - ["Dynamics"]["AIDAparam"]["flag_SRP"]: flag for solar radiation pressure perturbations. Valid entries are:
    0 : none.
    1 : active (no shadow).
    2 : active, cylindrical Earth shadow.
    3 : active, biconical Earth shadow.
    4 : active, cylindrical Earth and Moon shadows.
    5 : active, biconical Earth shadow and cylindrical Moon shadow.
    6 : active, biconical Earth and Moon shadows.
  - ["Dynamics"]["AIDAparam"]["flag_drag"]: flag for atmospheric drag perturbation. Valid entries are:
    0 : none.
    1 : active, non rotating atmosphere.
    2 : active, rotating atmosphere.
  - ["Dynamics"]["AIDAparam"]["flag_thirdbody"]: flag for third body perturbations. Valid entries are:
    0 : none.
    1 : active, Moon considered.
    2 : active, Sun and Moon considered.
  - ["Dynamics"]["AIDAparam"]["gravmodel"]: gravitational model file input data. In particular
    - ["Dynamics"]["AIDAparam"]["gravmodel"]["path"]: gravitational model path.
    - ["Dynamics"]["AIDAparam"]["gravmodel"]["name"]: gravitational model file name ("egm96" or "egm2008").

Figure 1.1: *Example of JSON input file: (a) DALS portion; (b) orbital dynamics and Earth modelling portions; (c) input files portion; (d) observer setup portion.*

- ["Dynamics"]["SADAparam"]: this object is included if the "SADA" propagator is selected. It contains
  - ["Dynamics"]["SADAparam"]["flag_SRP"]: flag for solar radiation pressure perturbations. Valid entries are

    0 : none.

    1 : active (no shadow).
  - ["Dynamics"]["SADAparam"]["flag_drag"]: flag for atmospheric drag perturbations. Valid entries are

    0 : none.

    1 : active, rotating atmosphere.
  - ["Dynamics"]["SADAparam"]["flag_SunMoon"]: flag for third body perturbations. Valid entries are

    0 : none.

    1 : Moon and Sun, analytical ephemeris.

2 : Moon and Sun, SPICE ephemeris.

- ["Dynamics"]["SADAparam"]["flag_tesseral"]: flag for tesseral perturbations. Valid entries are

  0 : none.

  1 : active, 1:1 and 2:1 resonances if necessary (see Chapter 2).

- ["Dynamics"]["SADAparam"]["flag_shortPeriodics"]: flag for short periodics perturbations. Valid entries are

  0 : J_2 only.

  2 : J_2, Moon and Sun.

- ["Dynamics"]["SADAparam"]["flag_linear"]: flag for performing uncertainty propagation within single sets of observations with a linear approximation ("true" or "false").

The ["Earth"] object contains the specifications for the modelling of the Earth-fixed reference frame. An example is given in Fig. 1.1b. It contains

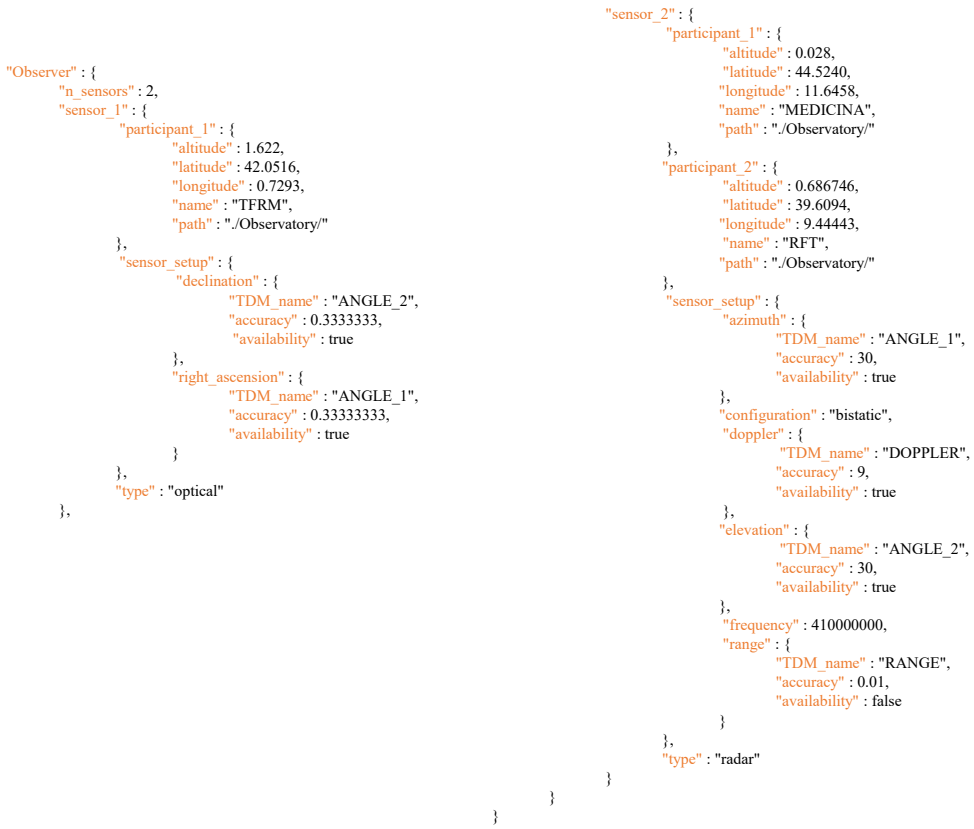
- ["Earth"]["Earth_model"]: SPICE-based Earth-fixed model. Valid entries are:
  - "IAU_EARTH": low accuracy reference frame.
  - "ITRF93": high accuracy reference frame.
- ["Earth"]["kernels"]: list of extra kernels for the characterisation of the Earth-fixed reference frame. It includes
  - ["Earth"]["kernels"]["path"]: kernels path.
  - ["Earth"]["kernels"]["IAU_assoc"]: "IAU_EARTH" association kernel name.
  - ["Earth"]["kernels"]["ITRF93"]: high precision Earth orientation kernel name. This entry is required only if the Earth model is set to "ITRF93" and shall be continuosly updated. The latest version can be found on the SPICE website.
  - ["Earth"]["kernels"]["ITRF93_assoc"]: "ITRF93" association kernel name. This file is required only if the Earth model is set to "ITRF93".
  - ["Earth"]["kernels"]["pck"]: orientation and size/shape data for natural bodies.

The ["Measurements"] object contains the name and location of the TDM files for all the involved sensors and a flag for measurements simulation. An example is given in Fig. 1.1c. In particular

- ["Measurements"]["flag_LTS"]: flag for light-time and stellar aberration corrections ("true" or "false", optical sensors only).
- ["Measurements"]["sensor_1"]: it is a sub-object containing name (["Measurements"]["sensor_1"]["name"]) and path (["Measurements"]["sensor_1"]["path"]) of the TDM file generated by sensor_1. If more than one sensor is considered, extra ["Measurements"]["sensor_2"],["Measurements"]["sensor_3"], etc. are required, each specifying the data of the associated TDM.

The ["OD"] object contains name and type of the state estimates input file. An example is given in Fig. 1.1c. It contains

- ["OD"]["path"]: location of the state estimates file.
- ["OD"]["name"]: name of the state estimates file.
- ["OD"]["flag_Cov_av"]: flag that indicates whether the state covariance is available or not ("true" or "false").
- ["OD"]["flag_Bfactor_av"]: flag that indicates whether an estimate for the Bfactor parameter is available or not ("true" or "false").
- ["OD"]["flag_SRPC_av"]: flag that indicates whether an estimate for the SRPC parameter is available or not ("true" or "false").



(a) **(b)**

Figure 1.2: *Example of JSON input file for a two-sensor configuration (observer portion): (a) optical sensor portion; (b) bistatic radar portion.*

Finally, the ["Observer"] object contains all the data required to characterise all the sensors. The object is organized in n ["Observer"]["sensor_i"] sub-blocks, one for each sensor considered. An example is given in Fig. 1.1d, where a single optical sensor is considered. For each sensor, three different configurations are possible: optical, monostatic radar and bistatic radar. Each ["Observer"]["sensor_i"] object includes

- ["Observer"]["sensor_i"]["participant_1"]: it is the first participant of the sensor architecture. In case of optical sensor or monostatic radar, it is also the only one. For a monostatic radar, it coincides with the receiver. The object collects all the data required to generate the SPICE kernel associated with the station in the Earth model previously selected. The entries are:
  - ["Observer"]["sensor_i"]["participant_1"]["path"]: location where to place the station kernel.
  - ["Observer"]["sensor_i"]["participant_1"]["name"]: name of the station kernel.
  - ["Observer"]["sensor_i"]["participant_1"]["longitude"]: station longitude (WGS84 reference frame, deg).
  - ["Observer"]["sensor_i"]["participant_1"]["latitude"]: station latitude (WGS84 reference frame, deg).
  - ["Observer"]["sensor_i"]["participant_1"]["altitude"]: station altitude (WGS84 reference frame, km).
- ["Observer"]["sensor_i"]["sensor_setup"]: a list of all the available measurements and related precision. This block is made of sub-objects with the name of the considered measurements, each of them with the following entries

```
META_START
TIME_SYSTEM = UTC
START_TIME = 2020-04-18T13:53:47.350053
STOP_TIME  = 2020-04-18T13:59:47.350053
PARTICIPANT_1 = MEDICINA
PARTICIPANT_2 = UNKNOWN
PARTICIPANT_3 = RFT
MODE = SEQUENTIAL
PATH = 3,2,1
ANGLE_TYPE = AZEL
REFERENCE_FRAME = ECIJ2000
META_STOP

DATA_START
ANGLE_1 = 2020-04-18T13:53:47.350053  158.9997166675095
ANGLE_2 = 2020-04-18T13:53:47.350053  20.93618646329778
DOPPLER = 2020-04-18T13:53:47.350053  -1060.060514198538
ANGLE_1 = 2020-04-18T13:54:27.350053  154.3309061572179
ANGLE_2 = 2020-04-18T13:54:27.350053  21.95996125644937
DOPPLER = 2020-04-18T13:54:27.350053  -2135.668029663795
ANGLE_1 = 2020-04-18T13:55:07.350053  149.6413450822842
ANGLE_2 = 2020-04-18T13:55:07.350053  22.73810841830675
DOPPLER = 2020-04-18T13:55:07.350053  -3156.677374075705
ANGLE_1 = 2020-04-18T13:55:47.350053  144.9864464637025
ANGLE_2 = 2020-04-18T13:55:47.350053  23.26639147726421
DOPPLER = 2020-04-18T13:55:47.350053  -4111.765974213338
```

Figure 1.3: *Example of TDM input file generated by a bistatic radar sensor (portion).*

- ["Observer"]["sensor_i"]["sensor_setup"]["measurement"]["availability"]: flag that indicates if the measurement is included in the TDM ("true" or "false").
- ["Observer"]["sensor_i"]["sensor_setup"]["measurement"]["accuracy"]: 1σ accuracy.
- ["Observer"]["sensor_i"]["sensor_setup"]["measurement"]["TDM_name"]: TDM associated name.

In case of optical sensor, the ["Observer"]["sensor_i"]["sensor_setup"] object contains two sub-objects, namely ["right_ascension"] and ["declination"]. They are typically both available, and their accuracy is expressed in arcseconds. In case of radar sensor, four blocks are present, i.e. ["azimuth"], ["elevation"], ["doppler"] and ["range"], and the user can decide whether they are available or not depending on the sensor characteristics. Similarly to before, the accuracy of the two angular measurements is expressed in arcseconds, whereas Doppler shift and range accuracy are expressed in Hz and km, respectively.

In case of radar sensor, two extra informations, labelled as ["frequency"] and ["configuration"] are included in the ["Observer"]["sensor_i"]["sensor_setup"] object. The ["configuration"] field indicates whether the radar is monostatic or bistatic. In case of bistatic configuration, the ["Observer"]["sensor_i"] object shall contain an extra entry, called ["participant_2"], which describes the location of the transmitter similarly to what is done for the ["participant_1"]. The whole procedure is then repeated for all the involved sensors. An example of how to compile the JSON input file in case of two sensors (optical and bistatic radar) is shown in Fig. 1.2.

1.1.3. The TDM input file

The set of time epochs and associated sensor measurements are provided to the DALS by means of a classic TDM file. An example of TDM file generated by a bistatic radar sensor providing azimuth, elevation and Doppler shift measurements is shown in Fig. 1.3. As previously mentioned, the name and location of all the TDM files are provided in the ["Measurements"] object of the JSON input file, one for each sensor, whereas the TDM names associated with each single measurement can be defined in the ["sensor_setup"] entry of the ["Observer"]["sensor_i"] object of all sensors. The tool can handle any combination and number of sensors, i.e. it can work with a single TDM or multiple TDMs, and the measurements can be obtained by a single sensor or different sensors, of the same type or different types, e.g. optical and radar.

1.1.4. The state estimates input file

The available a priori state estimate and associated epoch are provided in a simple .txt file whose name and location are specified in the ["OD"] object of the JSON input file. The first line of the file contains the estimate time epoch (UTC time system), whereas the second line includes the state estimate (ECIJ2000). The content of the rest of the file is variable, depending on the availability of state

```
16 APR 2020 20:30:24.307339 (UTC)
-5.3071420165346569e+02 7.6900797483692458 6e+03 4.03977996114627331e+03 -6.1071352179788496 2e+00 -5.1495610074422038 0e-01 2.40861943963528446e+00
2.5000000000000046e-05  -4.2351647362715017 0e-22  8.4703294725430033 9e-22  0.0000000000000000e+00  0.0000000000000000e+00 0.0000000000000000e+00
4.2351647362715017 0e-22  2.5000000000000012e-05 -3.3881317890172013 6e-21  0.0000000000000000e+00  0.0000000000000000e+00 0.0000000000000000e+00
0.0000000000000000e+00 -3.3881317890172013 6e-21  2.5000000000000046e-05  0.0000000000000000e+00  0.0000000000000000e+00 0.0000000000000000e+00
0.0000000000000000e+00  0.0000000000000000e+00 0.0000000000000000e+00  2.5000000000000042e-07  0.0000000000000000e+00 1.3234889800848442 8e-23
0.0000000000000000e+00  0.0000000000000000e+00 0.0000000000000000e+00  0.0000000000000000e+00  2.4999999999999989e-07 -3.9704669402545328 4e-23
0.0000000000000000e+00  0.0000000000000000e+00 0.0000000000000000e+00  1.3234889800848442 8e-23  -2.6469779601696885 6e-23 2.5000000000000042e-07
2.1731675274888487 6e-03
1.3996419826373487 4e-03
```

Figure 1.4: *Example of state estimates input file (covariance, Bfactor and SRPC available).*

```
CCSDS_OPM_VERS   = 2.0
CREATION_DATE    = 2020-06-01T20:07:34
ORIGINATOR       = DEIMOS SPACE
COMMENT          GEOCENTRIC, CARTESIAN, INERTIAL
OBJECT_NAME      = unknown
OBJECT_ID        = 00000
CENTER_NAME      = EARTH
REF_FRAME        = ECIJ2000
TIME_SYSTEM      = UTC

COMMENT  State vector
EPOCH            = 2020-04-16T20:30:24.307339
X                = -530.714332016609 [km]
Y                = 7690.079090718089 [km]
Z                = 4039.779489192777 [km]
X_DOT            = -6.10719278358436 [km/s]
Y_DOT            = -0.5151038014626561 [km/s]
Z_DOT            = 2.408877156961109 [km/s]

COMMENT  State covariance matrix
CX_X             = 0.01243294634432581 [km**2]
CY_X             = 0.0008424108317903097 [km**2]
CY_Y             = 0.01755122030507929 [km**2]
CZ_X             = -0.002370801281241992 [km**2]
CZ_Y             = 0.0003774610147756085 [km**2]
CZ_Z             = 0.007687883363682129 [km**2]
CX_DOT_X         = 0.0003267584993190924 [km**2/s]
CX_DOT_Y         = -0.0001949937490336081 [km**2/s]
CX_DOT_Z         = 0.0001154716647405686 [km**2/s]
CX_DOT_X_DOT     = 2.3843551371077 17e-05 [km**2/s**2]
CY_DOT_X         = -0.0001523828765365797 [km**2/s]
CY_DOT_Y         = 2.45980860031731e-05 [km**2/s]
CY_DOT_Z         = 7.3594508312775 57e-07 [km**2/s]
CY_DOT_X_DOT     = -6.433550011905295e-06 [km**2/s**2]
CY_DOT_Y_DOT     = 2.2543155893880 34e-06 [km**2/s**2]
CZ_DOT_X         = -0.0001990184250555748 [km**2/s]
CZ_DOT_Y         = 3.1569262801173 21e-05 [km**2/s]
CZ_DOT_Z         = 1.5637743786201 71e-05 [km**2/s]
CZ_DOT_X_DOT     = 1.5637743786201 71e-05 [km**2/s**2]
CZ_DOT_Y_DOT     = 4.3987840641648 42e-06 [km**2/s**2]
CZ_DOT_Z_DOT     = 4.0011696103318821e-05 [km**2/s**2]
```

Figure 1.5: *Example of OPM output file.*

covariance matrix, Bfactor estimate and SRPC estimate. If all quantities are available, then the .txt file is made of 10 lines, otherwise the number of lines is lower. The tool allows for any combination, i.e. one may have all three quantities, no quantity, the covariance matrix but no parameters, no covariance and all the parameters, covariance and Bfactor or covariance and SRPC. The availability of each single quantity shall be specified in the related entries of the ["OD"] object of the JSON input file.

1.1.5. The SPICE pinpoint utility

In addition to the aforementioned input files, the DALS toolbox requires the NASA SPICE "pinpoint" utility. The tool is used by the DALS for generating the SPICE kernels associated with the involved observer ground stations. The correct version of the utility that suits the user's operative system can be found on the SPICE website. Further details can be found in Chapter 4.

1.2. The OPM output file

The DALS tool provides as output the estimated state vector, covariance matrix and eventually the Bfactor and SRPC parameters, if required. The data are organized in a classic OPM file, whose name and location are set in the ["OD"] object of the JSON input file. An example is shown in Fig. 1.5. The epoch of the estimate and the associated state vector are always included, and they are expressed in the UTC time system and ECIJ2000 reference frame, respectively. The covariance matrix can be included or not,

depending on the flag set in ["OD"]["output"]["flag_Cov"]. The flag ["OD"]["output"]["flag_KeplPar"], instead, allows the user to include the osculating Keplerian parameters of the object.

If the Bfactor or the SRPC parameters are estimated, they are included as "USER_DEFINED" parameters, i.e. "USER_DEFINED_BFACTOR" and "USER_DEFINED_SRPC". In addition, the associated standard deviations are provided as "USER_DEFINED_BFACTOR_SIGMA" and "USER_DEFINED_SRPC_SIGMA". It may happen that the DALS process cannot provide a value for these uncertainties (e.g., if the residual does not depend on the object parameter). In that case, the two uncertainties are set to "NOT AVAILABLE".

2. Dynamical models and propagation

This chapter briefly illustrates the two dynamical models used by the DALS toolbox for propagating the available state estimates and performing the least squares process. The description of the numerical propagator AIDA follows the analysis offered in Morselli (2014), whereas Section 2.2 describes the semi-analytical SADA model following the illustration given by Gondelach (2019). Finally, Section 2.3 describes the propagator.

2.1. AIDA

The propagator AIDA (Accurate Integrator for Debris Analysis) is a numerical propagator tailored for space debris analysis within a DA framework. The perturbations included in AIDA are the geopotential acceleration, atmospheric drag, solar radiation pressure, and third body gravity. Details on the modelling of these sources of perturbation are given hereafter.

2.1.1. Geopotential acceleration

The acceleration due to the Earth's gravity potential is expressed as

$$\boldsymbol{a}_{grav} = \boldsymbol{\nabla}\frac{\mu_{\oplus}}{r}\sum_{n=0}^{\infty}\sum_{m=0}^{n}\frac{R_{\oplus}^{n}}{r^{n}}\bar{P}_{nm}(\sin\phi)(\bar{C}_{nm}\cos(m\lambda)+\bar{S}_{nm}\sin(m\lambda)) \tag{2.1}$$

where $\mu_{\oplus}$ is the Earth's gravitational parameter, $R_{\oplus}$ is the Earth's radius, $\bar{C}_{nm}$ and $\bar{S}_{nm}$ the normalized geopotential coefficients, $\bar{P}_{nm}$ the normalized associated Legendre functions, r is the object distance from the centre of the Earth, and ϕ and λ are the geocentric latitude and longitude. Two possible gravitational models can be selected, i.e. the EGM96 and the EGM2008 (Pavlis et al., 2012). Both the model and the gravitational order can be set in the ["Dynamics"]["AIDAparam"]["gravmodel"] field of the JSON input file.

2.1.2. Atmospheric drag

Th perturbing acceleration due to the atmospheric drag is modelled as

$$\boldsymbol{a}_{drag} = -\frac{1}{2}C_D\frac{A}{m}\rho v_r\boldsymbol{v}_r = -\frac{1}{2}\text{Bfactor}\rho v_r\boldsymbol{v}_r \tag{2.2}$$

where C_D is the drag coefficient, A is the object cross-sectional area, m the object mass, whereas $\boldsymbol{v}_r$ is the object relative velocity with respect to the Earth atmosphere, and ρ is the atmospheric density at the object's position. The atmospheric density is computed using the Naval Research Laboratory's Mass Spectrometer and Incoherent Scatter Radar of year 2000 (NRLMSISE-00) model (Picone et al., 2002). The model requires as inputs the solar and geomagnetic activity, geodetic altitude and latitude, longitude, year, day, and time of day in UT. These data are provided by the space weather file included by the ["spaceWeatherFile"] object of the JSON environment file.

AIDA allows for the definition of two different atmospheric models, namely with non rotating and rotating atmosphere. The desired model is set up with the related flag of the ["Dynamics"]["AIDAparam"]["flag_drag"] object of the JSON input file.

2.1.3. Third body perturbations

The AIDA propagator allows the user to include Sun and Moon third body perturbations. The perturbing acceleration on the orbiting object is modelled as

$$\boldsymbol{a}_{3rd} = \mu_{3rd}\left(\frac{\boldsymbol{s}}{s^3} - \frac{\boldsymbol{r}_{3rd}}{r_{3rd}^3}\right) \tag{2.3}$$

where

$$\boldsymbol{s} = \boldsymbol{r}_{3rd} - \boldsymbol{r}_{obj} \tag{2.4}$$

is the relative position of the third body with respect to the object. The position of the third body is computed using NASA's SPICE toolkit. Three different scenarios can be considered: no perturbation, Moon-only perturbation or Moon and Sun perturbations. The desired setup is obtained by selecting the appropriate flag for the ["Dynamics"]["AIDAparam"]["flag_third"] object of the JSON input file.

2.1.4. Solar radiation pressure

Assuming that the detected object can be treated as a sphere, the solar radiation pressure is modelled as

$$\boldsymbol{a}_{SRP} = \frac{L_S}{4\pi c}\frac{A}{m}(1+\epsilon)\frac{\boldsymbol{r}_{obj}-\boldsymbol{r}_S}{||\boldsymbol{r}_{obj}-\boldsymbol{r}_S||^3}\nu = \frac{L_S}{4\pi c}\mathrm{SRPC}\frac{\boldsymbol{r}_{obj}-\boldsymbol{r}_S}{||\boldsymbol{r}_{obj}-\boldsymbol{r}_S||^3}\nu \tag{2.5}$$

where c is the speed of light, L_S the solar luminosity, ϵ the body reflectivity, whereas ν is the shadow function. The tool allows the user to consider both Sun and Moon shadow, using either a cylindrical or biconical model. The desired setup is obtained by selecting the appropriate flag for the ["Dynamics"]["AIDAparam"]["flag_SRP"] object of the JSON input file.

2.2. SADA

The SADA propagator used in the DALS tool is a DA reformulation of HEOSAT, a semi-analytical propagator developed by Lara et al. (2017) to study the long-term evolution of objects in Highly Elliptical Orbits (HEO). The perturbation model, which is propagated in time in the True of Date reference system, takes into account the gravitational effects due to zonal terms and lunisolar perturbations, solar radiation pressure (SRP) and atmospheric drag. The gravitational terms are expressed in Hamiltonian form to obtain the mean elements equations of motion using Deprit's perturbation algorithm (Deprit, 1969) based on Lie transformations. The equations of motion due to SRP and drag perturbations are averaged over the mean anomaly via Gauss equations. The averaging techniques applied for developing HEOSAT are described in detail by Lara et al. (2017). A brief explanation is provided hereafter.

2.2.1. Gravitational perturbations

The zonal-term Hamiltonians are simplified by removing parallactic terms (via elimination of the parallax, see Deprit (1981) and Lara et al. (2014)) and short-periodic terms are eliminated by Delaunay normalization. This is carried out up to second order of the second zonal harmonic J_2 and to first order for J_3-J_{10}.

2.2.2. Atmospheric perturbations

Mean element rates due to atmospheric drag are computed by numerically averaging Gauss equations over the mean anomaly assuming a spherical object and a rotating atmosphere. The atmospheric density is taken from the Harris-Priester atmospheric density model (Harris and Priester, 1962). Atmospheric drag perturbations can be activated by setting the proper flag for the ["Dynamics"]["SADAparam"]["flag_drag"] object of the JSON input file.

2.2.3. Third body perturbations

The disturbing potentials of the Sun and Moon (point-mass approximation) are expanded using Legendre series to obtain the Hamiltonians for averaging out the short-periodic terms (Lara et al., 2012). Second and sixth-order Legendre polynomials are taken for the Sun and Moon potentials, respectively. Third body perturbations can be activated by setting the proper flag for the ["Dynamics"]["SADAparam"]["flag_third"] object of the JSON input file.

2.2.4. Solar radiation pressure

For averaging the equations of motion due to SRP, a spherical object and constant solar flux along the orbit (i.e. no shadow) are assumed. Kozai's analytical expressions for perturbations due to SRP (Kozai, 1963) are then used to analytically average Gauss equations over the mean anomaly. Solar radiation pressure perturbations can be activated or not by setting the proper flag for the ["Dynamics"]["SADAparam"]["flag_third"] object of the JSON input file.

2.2.5. Tesseral resonances

The averaged equations of motion due to tesseral resonance are obtained by averaging the tesseral Hamiltonian terms over the mean anomaly in the rotating frame to preserve the resonant terms (Lara et al., 2013). For this, Kaula's expansion of the geopotential in orbital elements is used. In the current version of the code, only the 1:1 and 2:1 tesseral resonances, relevant for 24-hour and 12-hour orbits, respectively, are considered. These resonances can be activated by setting ["Dynamics"]["SADAparam"]["flag_tesseral"] equal to 1. Then, at each integration step, the SADA propagator computes the resonance period of each resonance, and the perturbation is considered only if this period is larger than 5 days (i.e. if the rate of change of the resonant angle is smaller than $2\pi/5$days) (Morand et al., 2013).

2.3. Propagator

Regardless of the dynamical model selected by the user, the state estimates and uncertainty propagation process is performed with a DA implementation of the 8th-order variable-stepsize Runge-Kutta integrator (RK78) by Prince and Dormand (1981), with an 8th-order solution for propagation and 7th-order solution used for the stepsize control. The tolerance of the propagator can be set by the user by changing the ["Dynamics"]["tolerance"] entry of the JSON input file.

3. Residuals computation

The DALS toolbox aims at identifying the state estimate that minimizes the difference between the lists of measurements included in the available TDM files and the measurements that would be obtained with the considered state estimate. This difference is expressed in terms of a specific performance index, named residual (r), that takes into account both the accuracy of the measurements and the availability of an apriori state estimate. The estimation of a set of measurements starting from a state estimate is here referred to with the function f and runs in three steps:

1. Propagate the available state estimate $\boldsymbol{x}_{est}$ from t_{est} to the set of observation instants $T = \{t_1, \dots, t_N\}$.
2. Compute, for each time instant, the required set of measurements.
3. Compute the residuals.

In the previous notation, the set T includes all the sets provided in the available TDMs, i.e. it is the result of the merging of all the different observation windows. All the described operations are performed in the DA framework, therefore one can write

$$[r] = f([\boldsymbol{x}_{est}]) \tag{3.1}$$

While the last two steps change according to the sensor type considered, the propagation process is the same and provides a set of DA state vectors $\{[\boldsymbol{x}]\} = \{[\boldsymbol{x}(t_1)], \dots, [\boldsymbol{x}(t_N)]\}$. This set of state vectors is then used to compute the estimated measurements. Section 3.1 describes the measurements estimation process for optical and radar sensors, while the residuals computation is illustrated in Section 3.2.

3.1. Measurements estimation

The measurements estimation process changes according to the sensor considered. Optical sensors generally provide two angles per time instant, namely the topocentric right ascension α and declination δ. If an optical sensor is selected, the availability and accuracy of these measurements is governed by the ["availability"] and ["accuracy"] flags of the ["right_ascension"] and ["declination"] entries of the ["Observer"]["sensor_i"]["sensor_setup"] object for the i-th sensor. Starting from the available set of DA state vectors $\{[\boldsymbol{x}]\}$, the steps for computing, for a generic time instant t_k, the DA expansions of topocentric right ascension and declination are the following:

1. Compute the relative position object-observer

$$[\boldsymbol{\rho}(t_k)] = [\boldsymbol{rr}(t_k)] - \boldsymbol{rr}_{GS}(t_k) \tag{3.2}$$

   where $[\boldsymbol{rr}(t_k)]$ is the DA expansion of the object position vector at time t_k, whereas $\boldsymbol{rr}_{GS}(t_k)$ is the ground station position at time t_k. This position is computed using a dedicated SPICE kernel generated by the tool (see Chapter 4 for further details).

2. Compute the topocentric right ascension

$$[\alpha(t_k)] = \text{atan2}\left(\frac{[\rho(t_k)]_2}{[\rho(t_k)]_1}\right) \tag{3.3}$$

   where the subscripts "1" and "2" refer to the components of the vector.

3. Compute the topocentric declination

$$[\delta(t_k)] = \text{asin}\left(\frac{[\rho(t_k)]_3}{|||[\boldsymbol{\rho}(t_k)]|||}\right) \tag{3.4}$$

The tool allows the user for including light-time and stellar aberration (LTS) corrections. The dedicated flag (["Measurements"]["flag_LTS"]) is by default set to "true". The corrections are made with a DA-based version of SPICE LTS correction routines.

The set of measurements provided by a radar sensor is potentially larger. Both monostatic and bistatic radars theoretically can provide azimuth and elevation of the object as measured by the receiver, along with Doppler shift and range (or better slant range) measurements. Similarly to optical sensors, the availability and accuracy of these measurements is governed by the ["availability"] and ["accuracy"] flags of the ["azimuth"], ["elevation"], ["doppler"] and ["range"] entries of the ["Observer"]["sensor_i"]["sensor_setup"] object. Starting from the available set of DA state vectors $\{[\boldsymbol{x}]\}$, the steps for computing, for a generic time instant t_k, the DA expansions of azimuth, elevation, Doppler shift and slant range are the following:

1. Compute the relative position object-receiver

$$[\boldsymbol{\rho}_{RX}(t_k)] = [\boldsymbol{rr}(t_k)] - \boldsymbol{rr}_{RX}(t_k) \tag{3.5}$$

where $\boldsymbol{rr}_{RX}(t_k)$ is the receiver (RX) position at t_k.

2. Compute the rotation matrix $A_{eci2RX}(t_k)$ from the ECIJ2000 to the RX topocentric reference frame and rotate $[\boldsymbol{\rho}_{RX}(t_k)]$

$$[\boldsymbol{\rho}_{RX}(t_k)]^{RX} = A_{eci2RX}(t_k)[\boldsymbol{\rho}_{RX}(t_k)] \tag{3.6}$$

3. Compute the azimuth of the object at the RX

$$[Az_{RX}(t_k))] = -\text{atan2}\left(\frac{[\boldsymbol{\rho}_{RX}(t_k)]_2^{RX}}{[\boldsymbol{\rho}_{RX}(t_k)]_1^{RX}}\right) \tag{3.7}$$

4. Compute the elevation of the object at the RX

$$[El_{RX}(t_k)] = \text{asin}\left(\frac{[\boldsymbol{\rho}_{RX}(t_k)]_3^{RX}}{|||[\boldsymbol{\rho}_{RX}(t_k)]|||}\right) \tag{3.8}$$

5. Compute the slant range DA expansion $[SR(t_k)]$. If the radar is monostatic, then

$$[SR(t_k)] = 2|||[\boldsymbol{\rho}_{RX}(t_k)]||| \tag{3.9}$$

If the radar is bistatic, then compute the relative position object-transmitter

$$[\boldsymbol{\rho}_{TX}(t_k)] = [\boldsymbol{rr}(t_k)] - \boldsymbol{rr}_{TX}(t_k) \tag{3.10}$$

where $\boldsymbol{rr}_{TX}(t_k)$ is the transmitter (TX) position at t_k. Then the slant range becomes

$$[SR(t_k)] = |||[\boldsymbol{\rho}_{RX}(t_k)]||| + |||[\boldsymbol{\rho}_{TX}(t_k)]||| \tag{3.11}$$

6. Compute the rotation matrix $A_{eci2ecef}(t_k)$ from the ECIJ2000 to the Earth-fixed reference frame

7. Compute the Doppler shift expansion $[DS(t_k)]$. In case of monostatic radar

$$[DS(t_k)] = -\frac{2f}{c}A_{eci2ecef}\left([\boldsymbol{vv}(t_k))] \cdot \frac{[\boldsymbol{\rho}_{RX}]}{|||[\boldsymbol{\rho}_{RX}]|||}\right) \tag{3.12}$$

where f is the radar frequency and c is the speed of light, whereas for a bistatic radar

$$[DS(t_k)] = -\frac{f}{c}A_{eci2ecef}\left([\boldsymbol{vv}(t_k))] \cdot \frac{[\boldsymbol{\rho}_{RX}]}{|||[\boldsymbol{\rho}_{RX}]|||} + [\boldsymbol{vv}(t_k))] \cdot \frac{[\boldsymbol{\rho}_{TX}]}{|||[\boldsymbol{\rho}_{TX}]|||}\right) \tag{3.13}$$

3.2. Residual computation

Once computed the set of estimated measurements, the residuals are computed by comparing estimated and real measurements and considering the possible availability of an apriori estimate for state and covariance. The residual expansion can be essentially divided in two parts

$$[r] = [r_{meas}] + [r_{apr}] \tag{3.14}$$

that is, a portion due to the measurement difference and a portion due to a priori estimates. The computation of the first part depends on the sensor type. For optical sensors, the residual expansion computed at time epoch t_k can be expressed as

$$[r_{meas}(t_k)] = \frac{f_\alpha}{\sigma_\alpha^2}\left([\alpha(t_k)] - \alpha_{real}(t_k)\right)^2 + \frac{f_\delta}{\sigma_\delta^2}\left([\delta(t_k)] - \delta_{real}(t_k)\right)^2 \tag{3.15}$$

where the subscript "real" refers to the real available measurements, σ_α and σ_δ define the accuracy of the sensor in measuring right ascension and declination and they are set in the ["accuracy"] field of the ["right_ascension"] and ["declination"] entries of the ["Observer"]["sensor_i"]["sensor_setup"] object, whereas f_α and f_δ govern the availability of each type of measurement, and they can be either 0 or 1. In case of radar measurements, the expression becomes

$$[r_{meas}(t_k)] = \frac{f_{Az}}{\sigma_{Az}^2}\left([Az_{RX}(t_k)] - Az_{RX,real}(t_k)\right)^2 + \frac{f_{El}}{\sigma_{El}^2}\left([El_{RX}(t_k)] - El_{real}(t_k)\right)^2 + \tag{3.16}$$

$$\frac{f_{SR}}{\sigma_{SR}^2}\left([SR(t_k)] - SR_{real}(t_k)\right)^2 + \frac{f_{DS}}{\sigma_{DS}^2}\left([DS(t_k)] - DS_{real}(t_k)\right)^2 \tag{3.17}$$

where σ_{Az}, σ_{El}, σ_{SR} and σ_{DS} define the accuracy of the sensor in measuring azimuth, elevation, slant range and Doppler shift and they are set in the ["accuracy"] field of the ["azimuth"], ["elevation"],["range"] and ["doppler"] entries of the ["Observer"]["sensor_i"]["sensor_setup"] object, whereas f_{Az}, f_{El}, f_{SR} and f_{DS} govern the availability of each type of measurement. The complete residual expansion can be then expressed as

$$[r_{meas}] = \sum_{k=1}^{N} [r_{meas}(t_k)] \tag{3.18}$$

where, at each time instant t_k, the optical or radar expression for $[r_{meas}(t_k)]$ are used depending on the sensor used at time t_k.

The computation of $[r_{apr}]$ is instead insensitive to the sensor type. By defining with t_{est} the estimation epoch of the DALS process, $[\boldsymbol{x}_{est}] = [\boldsymbol{x}](t_{est})$ the estimate at t_{est}, $\boldsymbol{x}_{est,apr}$ the apriori estimate and $\boldsymbol{C}_{est,apr}$ the associated covariance, one obtains

$$[r_{apr}] = \left([\boldsymbol{x}_{est}] - \boldsymbol{x}_{est,apr}\right)^{\mathrm{T}} \boldsymbol{C}_{est,apr}^{-1}\left([\boldsymbol{x}_{est}] - \boldsymbol{x}_{est,apr}\right) \tag{3.19}$$

The name and location of the file containing apriori estimates are expressed in the ["path"] and ["name"] fields of the ["OD"] object. While $\boldsymbol{x}_{est,apr}$ is always available, the presence of the apriori covariance is not granted and it is governed by the ["flag_Cov_av"] flag: if it is set to "true", then it means that the state estimates file contains an apriori estimate for the covariance, thus $[r_{apr}]$ can be computed, otherwise this step is skipped and $[r] = [r_{meas}]$.

4. The DALS toolbox

The DALS toolbox is organized as a single C++ class which receives as input the files described in Chapter 1 and performs a high order least least squares process to obtain an estimate of the state of the detected object at a desired time epoch. The process is performed in eight steps, which are illustrated hereafter.

4.1. Initialization

The first step of the DALS toolbox consists in initializing the private variables the are later used in the process. The initialization is done by reading the input JSON file described in Chapter 1 and checking the validity of each input.

4.2. Observer kernels generation

Once defined the DALS private variables, the program generates the required SPICE kernels for all the involved ground stations of all the considered sensors. Optical and monostatic radar sensors require a single kernel per sensor. Two different kernels per sensor are instead generated in case of bistatic radars. For each ground station, the starting point of the kernel generation process is represented by the geodetic latitude, longitude and altitude provided by the user for the considered participant of the ["Observer"]["sensor_i"] object in the JSON input file. Then, according to the Earth model defined in ["Earth"]["Earth_model"], these coordinates are either maintained in the WGS84 reference frame or converted in the ITRF93 system. A so-called "definitions file" is then generated, and used as input for the SPICE pinpoint utility. This function generates a binary file that provides the position of the ground station in any reference frame and the associated frame kernel, which defines the topocentric reference system associated with the observer. A complete documentation on the pinpoint utility can be found here.

4.3. TDM reading

Starting from the data written in the ["Measurements"] object of the JSON input file, the program then stores all the available observation epochs and associated sensor measurements. The parsing of all the TDM files is done on the basis of the sensor specifications defined in ["Observer"]["sensor_i"]["sensor_setup"], i.e. the sensor type and the availability of the single measurements.

4.4. State estimates reading

Once defined the set of available measurements, the program considers the state estimates file defined in ["OD"] and retrieves the apriori state estimate $\boldsymbol{x}_{data}$ and the associated time epoch t_{data}. Then, depending on the ["OD"]["flag_Cov_av"],["OD"]["flag_Bfactor_av"] and ["OD"]["flag_SRPC_av"] availability flags, the program stores the apriori covariance matrix $\boldsymbol{C}_{data}$ and Bfactor and SRPC estimates. While the availability of the covariance is not essential for the process, a first guess for the two parameters is required, thus they are arbitrarily set to 1e-3 in case no estimate is provided. This value is later refined during the DALS process.

4.5. State estimate propagation

The available state estimate is then propagated to the estimation epoch t_{est} defined by the user in ["DALS"]["date_est"], thus obtaining the apriori state estimate $\boldsymbol{x}_{est,apr}$. If t_{est} coincides with t_{data}, no propagation is required. Otherwise, two different scenarios are possible. If no apriori covariance is available, then $\boldsymbol{x}_{data}$ is propagated from t_{data} to t_{est}, thus obtaining $\boldsymbol{x}_{est,apr}$. Otherwise, a DA state vector of arbitrary domain is built on $\boldsymbol{x}_{data}$, i.e.

$$[\boldsymbol{x}_{data}] = \boldsymbol{x}_{data} + \delta\boldsymbol{x} \tag{4.1}$$

The DA vector is then propagated to t_{est} using the DA-based RK78 propagator described in Chapter 2, and the expansion of the state vector at t_{est}, $[\boldsymbol{x}_{est}]$, is thus obtained. This DA map is then used to compute the derivatives of the final state with respect to the initial conditions, i.e. the high order expansion of the state transition matrix

$$[\Phi]_{ij} = \frac{\partial [x_{est}]_i}{\partial \delta x_j} \tag{4.2}$$

with $i, j = 1, \ldots, 6$. By extracting the constant part of this matrix, the state transition matrix is obtained, and it used to propagate the covariance matrix from t_{data} to t_{est}

$$\boldsymbol{C}_{est,apr} = \boldsymbol{\Phi}\boldsymbol{C}_{data}\boldsymbol{\Phi}^{\mathrm{T}} \tag{4.3}$$

The state vector $\boldsymbol{x}_{est,apr}$ is instead obtained by extracting the constant part of $[\boldsymbol{x}_{est}]$.

4.6. DA guess initialization

Once computed $\boldsymbol{x}_{est,apr}$, a first guess for the DALS solver is generated. Once again, the process changes according to the availability of $\boldsymbol{C}_{est,apr}$. If the covariance matrix is available, this matrix is used to infer the search interval of the DALS, and thus initialize the DA state vector. Starting from $\boldsymbol{x}_{est,apr}$, the first step consists in computing the rotation matrix $\boldsymbol{A}_{eci2RSW}(t_{est})$ from the ECIJ2000 to the RSW reference frame (Hill et al., 2008). This matrix is then used to rotate $\boldsymbol{C}_{est,apr}$ in the RSW reference frame, thus obtaining

$$\boldsymbol{C}^{RSW}_{est,apr} = \boldsymbol{A}_{eci2RSW}(t_{est})\boldsymbol{C}_{est,apr}\boldsymbol{A}^{\mathrm{T}}_{eci2RSW}(t_{est}) \tag{4.4}$$

A DA state error $[\boldsymbol{\varepsilon}]$ vector is then built as

$$[\boldsymbol{\varepsilon}]^{RSW}_i = 3\sqrt{\left(C^{RSW}_{est,apr}\right)_{ii}}\delta x_i \tag{4.5}$$

with $i = 1, \ldots, 6$. The vector is then rotated in the ECI reference frame

$$[\boldsymbol{\varepsilon}] = \boldsymbol{A}^{\mathrm{T}}_{eci2RSW}(t_{est})[\boldsymbol{\varepsilon}]^{RSW} \tag{4.6}$$

and used to initialize a DA state at t_{est}, that is

$$[\boldsymbol{x}_{est}] = \boldsymbol{x}_{est,apr} + [\boldsymbol{\varepsilon}] \tag{4.7}$$

In case no covariance is available the DA state vector is directly initialized in the ECIJ2000 reference frame by setting an arbitrary scaling factor for each single DA variable, that is

$$[x_{est}]_i = \left(x_{est,apr}\right)_i + 10^{-3}\delta x_i \tag{4.8}$$

with $i = 1, \ldots, 6$.

The user can decide to estimate also the Bfactor or the SRPC parameter or both. This is done by setting to "true" the ["DALS"]["flag_Bfactor_est"] and ["DALS"]["flag_SRPC_est"] flags, or if no a priori estimate for one parameter is available. By defining with β an arbitrary parameter, the initialization is done by selecting a scaling factor equal to ten percent of the value of the parameter, i.e.

$$[\beta] = \beta + 0.1\beta\delta x_k \tag{4.9}$$

with k equal to 7 or 8, depending on how many parameters must be initialized (1 or 2) and which parameter is considered.

At the end of the initialization process, an augmented DA state vector $[\boldsymbol{x}^a_{est}]$ is built. This vector coincides with $[\boldsymbol{x}_{est}]$ in case neither Bfactor nor SRPC need no be estimated, otherwise $[\boldsymbol{x}^a_{est}]$ can be

$$[\boldsymbol{x}^a_{est}] = \{[\boldsymbol{x}_{est}], [\text{Bfactor}]\} \tag{4.10}$$

or

$$[\boldsymbol{x}^a_{est}] = \{[\boldsymbol{x}_{est}], [\text{SRPC}]\} \tag{4.11}$$

or

$$[\boldsymbol{x}^a_{est}] = \{[\boldsymbol{x}_{est}], [\text{Bfactor}], [\text{SRPC}]\} \tag{4.12}$$

The length of the augmented state vector defines the number of required DA variables and is here defined as n_{DA}.

4.7. The DALS solver

The DALS solver starts from the DA first guess initialized in the previous step and iteratively changes its constant part so that the value of the measurements residual is minimized. The process is run in the DA framework and makes use of the find_min_box_constrained function of the DLib C++ optimization library. The function requires a search and stop strategy, a function to minimize and its gradient, a starting point and the search domain. The DALS solver adopts the DLib bfgs_search_strategy search strategy and the objective_delta_stop_strategy with a tolerance of 1e-16 as stopping criterion, whereas the DA expansions of the residual and of its gradient are used. The centre of the DA domain is used as starting point, whereas the scaling strategy adopted to build the DA variables allows the solver to always use $[-1, 1]^{n_{DA}}$ as search domain. During the iteration process, the DALS solver constantly controls the accuracy of the resulting residual map. A dedicated performance index is computed, and if it is lower than the ε^{lim}_{DA} value set in ["DALS"]["map_tol"], then the map is considered accurate. In parallel, two possible different convergence criteria can be considered. If the ["DALS"]["stop_crit"] flag is set to "relative", a check on much the residual changes and how far the new point is with respect to the previous one is performed. If the flag is set to "gradient", then the norm of the gradient function as computed in the optimal solution is considered. In the first case, the threshold parameters are defined as ε^{lim}_{r} and $\varepsilon^{lim}_{\delta\boldsymbol{x}}$, and they correspond to the values set in ["DALS"]["fun_tol"] and ["DALS"]["step_tol"], respectively. In case the gradient is controlled, the $\varepsilon^{\lim}_{\nabla r}$ is defined as equal to ["DALS"]["grad_tol"].

The process starts by setting $[\boldsymbol{x}^a]_{(k)} = [\boldsymbol{x}^a_{est}]$, $\delta\boldsymbol{x}^a_{opt,(k-1)} = \mathbf{0}$ and $k = 1$, where the subscript (k) refers to the iteration index. Then, the process runs as follows

1. Compute the DA expansion of the residuals $[r]_{(k)} = f([\boldsymbol{x}^a]_{(k)})$, where f is the performance function described in Chapter 3.
2. Compute the gradient of the residuals. The operation is done in the DA framework, that is

$$[\nabla r]_{(k)} = \frac{\partial [r]_{(k)}}{\partial \delta\boldsymbol{x}} \tag{4.13}$$

   where $\delta\boldsymbol{x} \in \mathbb{R}^{n_{DA}}$
3. Run the minimization process using the find_min_box_constrained function. The function provides the optimal deviation $\delta\boldsymbol{x}^a_{opt,k}$.
4. Compute the value of the residual in the optimal point

$$r_{opt,(k)} = [r]_{(k)}(\delta\boldsymbol{x}^a_{opt,(k)}) \tag{4.14}$$

5. Update the DA map of the state vector

$$[\boldsymbol{x}^a]_{(k+1)} = [\boldsymbol{x}^a]_{(k)} + [\boldsymbol{x}^a]_{(k)}(\delta\boldsymbol{x}^a_{opt,(k)}) - [\boldsymbol{x}^a]_{(k)}(\mathbf{0}) \tag{4.15}$$

6. Compute the new DA expansion of the residuals

$$[r]_{(k+1)} = f([\boldsymbol{x}^a]_{(k+1)}) \tag{4.16}$$

7. Estimate the residuals map accuracy

$$\varepsilon_{DA,(k)} = \frac{|[r]_{(k+1)}(\mathbf{0}) - r_{opt,(k)}|}{1 + r_{opt,(k)}} \tag{4.17}$$

8. If $\varepsilon_{DA,(k)} < \varepsilon_{DA}^{lim}$, continue, otherwise set $k = k + 1$ and go back to step 1.

9. If the convergence criterion is set to "relative", compute residual and step relative variations

$$\varepsilon_{r,(k)} = \frac{|[r]_{(k+1)}(\mathbf{0}) - [r]_{(k)}(\mathbf{0})|}{1 + [r]_{(k)}(\mathbf{0})} \tag{4.18}$$

$$\varepsilon_{\delta\boldsymbol{x},(k)} = \frac{||\delta\boldsymbol{x}^a_{opt,(k)} - \delta\boldsymbol{x}^a_{opt,(k-1)}||}{(1 + ||\delta\boldsymbol{x}^a_{opt,(k-1)}||)} \tag{4.19}$$

If $\varepsilon_{r,(k)} < \varepsilon_r^{lim}$ or $\varepsilon_{\delta\boldsymbol{x},(k)} < \varepsilon_{\delta\boldsymbol{x}}^{lim}$ stop, otherwise set $k = k + 1$ and go back to step 1.

If the convergence criterion is set to "gradient", then evaluate the new gradient map in the optimal point and compute its norm

$$\varepsilon_{\nabla r,(k)} = ||[\nabla r]_{(k+1)}(\mathbf{0})|| \tag{4.20}$$

If $\varepsilon_{\nabla r,(k)} < \varepsilon_{\nabla r}^{lim}$ stop, otherwise set $k = k + 1$ and go back to step 1.

At the end of the process, a refined estimate of the augmented state vector $\boldsymbol{x}^a_{est,opt} = [\boldsymbol{x}^a_{est,opt}](\mathbf{0})$ is obtained. In addition, the availability of the residual map allows the solver to provide an estimate of the covariance matrix associated with the estimated state vector and object parameters, if any. Let us define with $[r]$ the residuals DA map associated with the optimal solution of the DALS process. By relying on differential algebra, one can compute the DA expansion of the normal matrix as

$$[N]_{i,j} = \frac{1}{2}\frac{\partial^2[r]}{\partial\delta x_i \partial\delta x_j} \tag{4.21}$$

where $i, j = 1, \ldots, n_{DA}$ The covariance matrix can be then computed as

$$\boldsymbol{C}^a_{est,opt} = \frac{[r](\mathbf{0})}{n_{meas} - n_{DA}}\boldsymbol{N}^{-1} \tag{4.22}$$

where n_{meas} is the overall number of measurements. The so-obtained matrix is an “augmented” covariance matrix, that is it is a (n_{DA} x n_{DA}) matrix and includes the contribution of both the estimated state and object parameters, if any.

4.8. OPM writing

Once estimated the state and associated covariance matrix, the results are organized in an OPM file, whose name and location are set in the ["OD"] object of the JSON input file. The epoch of the estimate and the associated state vector are always included, and they are expressed in the UTC time system and ECIJ2000 reference frame, respectively. The covariance matrix can be included or not, depending on the flag set in ["OD"]["output"]["flag_Cov"]. The flag ["OD"]["output"]["flag_KeplPar"], instead, allows the user to include the osculating Keplerian parameters of the object.

If the Bfactor or the SRPC parameters are estimated, they are included as "USER_DEFINED" parameters, i.e. "USER_DEFINED_BFACTOR" and "USER_DEFINED_SRPC". In addition, the associated standard deviations are provided as "USER_DEFINED_BFACTOR_SIGMA" and "USER_DEFINED_SRPC_SIGMA". It may happen that the DALS process cannot provide a value for these uncertainties (e.g., if the residual does not depend on the object parameter). In that case, the two uncertainties are set to "NOT AVAILABLE".

5. Installing the software

This chapter offers a brief description of the software organization and provides detailed guidelines for building and installing the tool from scratch.

5.1. Repository organization

The software repository, named cdtisw, can be found at https://github.com/UNIVERSIDAD-DE-LA-RIOJA/cdtisw and is divided in different folders each one grouping a parts of code with common characteristics:

- modules: it contains the various libraries (e.g. orbital dynamics)
- tools: it contains the executables
- tests: it includes all the unit tests
- cmake_modules: it stores all the CMake functions and modules.
- examples: it is the location where some example of inputs, outputs and configuration files are collected for the most relevant tools.

The project can be built using CMake, therefore each folder (except the examples) contains a CMakeLists.txt with the instructions for compiling and linking.

5.1.1. Modules

The project modules are the following:

- astro: a library for astronomical routines.
- atmos: a library of atmospheric models (NRLMSISE-00, exponential).
- cfg: a collection of configuration utilities.
- dynorb: a library for orbital dynamics. It contains the core of the numerical propagator AIDA.
- geco: a generic collection of classes and utilities.
- odeprop: a library for the integration/propagation of ODEs.
- sada: a library for semi-analitical propagator using differential algebra.

Each module is treated by CMake as an interface (when made of templates only) or shared library.

5.1.2. Tools

The tools folder contains all the executables

- binarygrav: it converts an ASCII gravity file to a binary file for AIDA propagations.
- aida: it runs a propagation with AIDA (double only).
- dals: it is the least squares solver described in this manual.

```yaml
name: CI

on: [push]

jobs:
  build-and-test:
    runs-on: ubuntu-latest
    env:
      dace-co-directory: ./dace

    steps:
    - name: Get CppUnit and JsonCpp
    # Ubuntu package decides to put json include files nested into jsoncpp folder
      run: sudo apt-get install -y libcppunit-dev libjsoncpp-dev libdlib-dev libeigen3-dev libpng-dev libsqlite3-dev libblas-dev liblapack-dev libboost-random-dev libboost-system-dev
    # Download latest version of DACE from GitHub
    - name: Get DACE
      uses: actions/checkout@v2
      with:
        repository : dacelib/dace
        path: ${{env.dace-co-directory}}
    # Perform compilation and install dace
    - name: Compile and install dace
      run: |
        mkdir _build && cd _build
        cmake -DWITH_ALGEBRAICMATRIX=ON ..
        make
        sudo make install
      working-directory : ${{env.dace-co-directory}}
    # Download CSPICE, extract files from the archive and remove unnecessary files
    - name: Download and extract CSPICE
      run: |
        mkdir naif && cd naif
        wget naif.jpl.nasa.gov/pub/naif/toolkit//C/PC_Linux_GCC_64bit/packages/cspice.tar.Z
        gzip -d cspice.tar.Z
        tar xfv cspice.tar
        rm -r cspice.tar cspice/etc cspice/data cspice/src cspice/makeall.csh cspice/doc cspice/exe
    # Install CSPICE includes and library
    - name: Install CSPICE
      run: |
        sudo mkdir /usr/local/include/cspice
        sudo cp -r naif/cspice/include/* /usr/local/include/cspice
        sudo cp naif/cspice/lib/cspice.a naif/cspice/lib/csupport.a /usr/local/lib
    # Now checkout our cdtisw repository
    - uses: actions/checkout@v2
    # Run CMake
    - name: Create configuration
      run:  |
        mkdir _build && cd _build
        cmake -DCMAKE_INSTALL_PREFIX:PATH=$HOME/scratch -DCMAKE_BUILD_TYPE=Release ..
    # Perform make
    - name: Make library
      run: make -C_build/
    # Perform unit tests
    - name: Run CTest
      run: ctest -VV
      working-directory : ./_build
    # Install the software
    - name: Install library
      run: sudo make install


```

Figure 5.1: *Instructions for building and installing the DALS software*

5.1.3. Unit tests

The unit tests are placed in the tests folder, and are further divided into folders whose name is the same as the modules they are testing. Currently these are:

- atmos: unit tests for atmospheric models.

- cfg: unit tests for the cfg library.
- dynorb: unit tests for AIDA.
- geco: unit tests for geco classes.

5.2. Building and installing the software

The DALS tool requires additional libraries for compiling, linking and running the software

- JSONcpp
- DLib
- Eigen
- CSPICE
- DACE

To generate unit tests (optional), the dependency on CPPunit is also required.

The repository is set up to perform continuous integration. The instructions required to build and install the software are shown in Fig. 5.1. At the end of the process, a "DALS" executable is generated in "cdtisw/_build/tools/dals". Then, after moving to the cdtisw/examples/dals/ directory, the command to launch the program is

./dals "./PATH/DALS.input.json"

The file "./PATH/DALS.input.json" is the JSON input file described in Chapter 1. The folder cdtisw/examples/dals/Tests/Combined contains several cases that can be tested, for three different orbital regimes (LEO, HEO, GEO).

The system has been successfully tested on Ubuntu 18.04.1

Bibliography

Deprit, A. (1969). Canonical transformations depending on a small parameter. *Celestial Mechanics*, 1:12–30.

Deprit, A. (1981). The elimination of the parallax in satellite theory. *Celestial mechanics*, 24:111–153.

Gondelach, D. (2019). *Orbit Prediction and Analysis for Space Situational Awareness*. PhD thesis, University of Surrey.

Harris, I. and Priester, W. (1962). Time-Dependent Structure of the Upper Atmosphere. *Journal of the Atmospheric Sciences*, 19:286–301.

Hill, K., Alfriend, K. T., and Sabol, C. (2008). Covariance-based Uncorrelated Track Association. In *Proc. AIAA/AAS Astrodynamics Specialist Conference and Exhibit*.

Kozai, Y. (1963). Effects of Solar Radiation Pressure on the Motion of an Artificial Satellite. *Smithsonian Contributions to Astrophysics*, 6:109–112.

Lara, M., López-Ochoa, L. M., Folcik, Z. J., and Cefola, P. J. (2013). Deep Resonant GPS-Dynamics Due to the Geopotential. *The Journal of the Astronautical Sciences* , 58:661–676.

Lara, M., San-Juan, J., and Hautesserres, D. (2017). HEOSAT: a mean elements orbit propagator program for highly elliptical orbits. *CEAS Space Journal*, 10:3–23.

Lara, M., San-Juan, J. F., and López-Ochoa, L. M. (2014). Delaunay variables approach to the elimination of the perigee in Artificial Satellite Theory. *Celestial Mechanics and Dynamical Astronomy*, 120:39–56.

Lara, M., San-Juan, J. F., López-Ochoa, L. M., and Cefola, P. J. (2012). On the third-body perturbations of high-altitude orbits. *Celestial Mechanics and Dynamical Astronomy*, 113:435–452.

Morand, V., Caubet, A., Deleflie, F., Daquin, J., and Fraysse, H. (2013). Semi analytical implementation of tesseral harmonics perturbations for high eccentricity orbits. In *Proc. AAS/AIAA Astrodynamics Specialist Conference*.

Morselli, A. (2014). *High order methods for Space Situational Awareness*. PhD thesis, Politecnico di Milano.

Pavlis, N. K., Holmes, S. A., Kenyon, S. C., and Factor, J. K. (2012). The development and evaluation of the Earth Gravitational Model 2008 (EGM2008). *Journal of Geophysical Research: Solid Earth*, 117(B4).

Picone, J. M., Hedin, A. E., Drob, D. P., and Aikin, A. C. (2002). NRLMSISE-00 empirical model of the atmosphere: Statistical comparisons and scientific issues. *Journal of Geophysical Research: Space Physics*, 107(A12).

Prince, P. J. and Dormand, J. R. (1981). High order embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 7(1):67–75.