69th International Astronautical Congress, Bremen, Germany, 1-5 October 2018.

IAC-18,C1,1,2,x44215

Hybrid SGP4 propagator based on machine-learning techniques applied to GALILEO-type orbits

Juan F. San-Juan[a*], **Iván Pérez**[b], **Eliseo Vergara**[c], **Montserrat San Martín**[d], **Rosario López**[e], **Alexander Wittig**[f], **Dario Izzo**[g]

[a] Scientific Computing Group, University of La Rioja, Logroño, Spain, juanfelix.sanjuan@unirioja.es
[b] Scientific Computing Group, University of La Rioja, Logroño, Spain, ivan.perez@unirioja.es
[c] Scientific Computing Group, University of La Rioja, Logroño, Spain, eliseo.vergara@unirioja.es
[d] Scientific Computing Group, University of Granada, Melilla, Spain, momartin@ugr.es
[e] Scientific Computing Group, University of La Rioja, Logroño, Spain, rosario.lopez@unirioja.es
[f] University of Southampton, Southampton, U.K., a.wittig@soton.ac.uk
[g] Advanced Concepts Team, European Space Agency, Noordwijk, The Netherlands, dario.izzo@esa.int
[*] Corresponding Author

Abstract

Space Situational Awareness current needs demand innovative solutions to the orbit propagation problem, so as to find new algorithms which are simultaneously accurate and fast. The hybrid methodology for orbit propagation constitutes a recent approach based on modeling the error of any orbit propagator with the aim of complementing its calculations and hence enhancing its precision. Diverse sources of inaccuracy can exist in propagators, such as incomplete perturbation models, forces not considered, low-order of the series expansions, etc. The creation of a time series with the differences between ephemerides computed with low-accuracy propagators and their corresponding real observations (or precisely computed ephemerides) allows applying time-series forecasting techniques so as to create a model that includes any dynamics not contained in the original propagator. Then, the adjusted model can be used in order to correct other future predictions. We present an application of the hybrid methodology, in which the time-series forecasting process is performed by means of machine-learning techniques, to the well-known SGP4 propagator. We have adjusted the resulting Hybrid SGP4 propagator, HSGP4, to the case of Galileo-type orbits. We will show how the use of HSGP4 can reduce the position error of SGP4, hence extending the validity of Two-Line Elements (TLE) from Galileo satellites.

1. Introduction

The maintenance of a running catalog of space objects orbiting the Earth is an unavoidable duty in the management of the space environment close to the Earth, which requires the orbital propagation of tens of thousands of objects. Currently, these ephemerides are publicly available through the NORAD catalog, yet other organizations, like ESA, may make their own data, obtained from observations, accessible.

Due to the huge number of objects to be propagated, a compromise between accuracy and efficiency must be established, depending on a variety of criteria. High-fidelity propagation models usually require step-by-step propagation by using numerical methods, which are computationally intensive because they rely on small step sizes. On the other hand, simplified models may admit analytical solutions, in this way notably alleviating the computational burden. In either case, the orbit propagation program relies only on the initial conditions, as well as on the propagation model, to make its predictions. However, the collection of past ephemerides provided by the catalog can be used to improve orbit predictions by taking non-modeled effects into account.

Indeed, the use of machine-learning techniques allows improving the predictions of traditional orbit propagators by combining their output with forecasting methods. These kinds of augmented orbit propagators have been named *hybrid propagators* [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]. Basically, a hybrid propagator assumes that the current trajectory may be split into two different parts of quite different nature: an approximate solution that takes account of the main dynamical effects, on the one hand, and a statistical error function that is able to emulate the non-modeled dynamics, on the other.

Originally, the first hybrid propagators relied on statistical time-series forecasting methods. Therefore, it seems quite a natural progression to try applying more innovative techniques, such as those based on machine learning, to the enhancement of this kind of propagators.

The remainder of this paper is organized as follows. Section 2 gives an overview of the hybrid methodology and the orbit propagators used in this study, SGP4 and AIDA. Section 3 briefly introduces the tools, R and H2O, as well as the machine-learning algorithm, deep learning, that have been used in the forecasting part of the hybrid propagators. The result of applying the machine-learning technique, specially adapted to the Galileo constellation, to the SGP4 propagator is described in Section 4. Finally, the conclusions and final remarks are presented in Section 5.

2. Hybrid methodology

2.1. Concept

A hybrid propagator, like any other type of orbit propagator, is aimed at estimating the position and velocity of any artificial satellite or piece of space debris at a future instant t_f, $\hat{\mathbf{x}}_f$, from its initial conditions $\mathbf{x}_1$ at an initial instant t_1. Any set of canonical or non-canonical variables can be used for this purpose. First of all, an initial approximation to $\hat{\mathbf{x}}_f$ can be calculated in the first stage of the hybrid propagator through the application of any type of numerical, analytical or semi-analytical integrating method, which can be named $\mathcal{I}$, to the initial conditions $\mathbf{x}_1$:

$$\mathbf{x}_f^{\mathcal{I}} = \mathcal{I}(t_f, \mathbf{x}_1). \qquad (1)$$

$\mathbf{x}_f^{\mathcal{I}}$ is an approximate value due to the fact that, in order to allow for an affordable integrating process, the considered models include some simplifications. The objective of the hybrid propagator second stage, the forecasting technique, is the modeling and reproduction of missing dynamics. In order to achieve it, the forecasting technique has to be adjusted to the real dynamics of the orbiter. A set of precise observations, or accurately determined positions and velocities, during a *control interval* $[t_1, t_T]$, with $t_T < t_f$, is necessary for that purpose. By means of those values, the error of the integrating technique, that is, its difference with respect to the orbiter real behavior, can be determined for any instant t_i in the control interval as

$$\varepsilon_i = \mathbf{x}_i - \mathbf{x}_i^{\mathcal{I}}. \qquad (2)$$

The time series of ε_i data in the control interval, $\varepsilon_1, \ldots, \varepsilon_T$, which we call *control data*, contains the dynamics that the forecasting technique must model and reproduce, and thus it is the data used to adjust it. Once that process has been performed, an estimation of the error at the final instant t_f, $\hat{\varepsilon}_f$, can be determined, which allows for the calculation of the desired value of $\hat{\mathbf{x}}_f$ as

$$\hat{\mathbf{x}}_f = \mathbf{x}_f^{\mathcal{I}} + \hat{\varepsilon}_f. \qquad (3)$$

2.2. Orbit propagators: SGP4 and AIDA

We use two orbit propagators in this work: the analytical SGP4, whose accuracy we want to improve, and the numerical AIDA, whose more precise ephemerides we take as *pseudo-observations*, that is, as the accurate data from which to determine the SGP4 error that we need to model in order to achieve our objective. Next, we briefly summarize the main characteristics of both propagators.

The *Simplified General Perturbations 4* propagator, SGP4, is originally based on Brouwer's theory [15] of satellite motion perturbed by the first five Earth zonal harmonics, although SGP4 only includes from J_2 to J_4. The description of the original Fortran code can be found in [16], although the complete documentation of all mathematical equations was published in 2004 [17].

In this work, we use the most updated code developed by Vallado [18], which merges SGP4/SDP4 models and is simply referred to as SGP4. This propagator includes the following force models: J_2 to J_4 zonal harmonics, air drag, and lunar and solar perturbations, as well as long-period resonant harmonics for the so-called deep-space satellites.

The input to the SGP4 propagator is the Two-Line-Element (TLE) set, which provides position and velocity vectors at a given time. The TLE includes information about the satellite and its orbit, such as satellite number, orbit inclination, eccentricity, argument of perigee, derivatives of the mean motion, BSTAR drag, mean anomaly, and others. TLEs can be obtained from CelesTrak and AFSPC[*].

The other propagator that we use is the *Accurate Integrator for Debris Analysis*, AIDA, [14], which includes the following force models:

- geopotential acceleration computed using the EGM2008 model, up to an arbitrary degree and order for the harmonics;
- atmospheric drag, modeled using the NRLMSISE-00 air density model;
- solar radiation pressure with dual-cone shadow model;
- third body perturbations from Sun and Moon.

The SPICE toolbox[†] from NASA is used both for Moon and Sun ephemerides (DE405 kernels) and for reference frame and time transformations (ITRF93 and J2000 reference-frame and leap-second kernels). Space-weather data is obtained from CelesTrak[‡] and Earth orientation parameters from IERS[§].

[*] www.space-track.org
[†] https://naif.jpl.nasa.gov/naif/index.html
[‡] http://www.celestrak.com/SpaceData/sw19571001.txt
[§] ftp://ftp.iers.org/products/eop/rapid/standard/finals.data

3. Machine-learning technique and tools

We take deep learning as the machine-learning technique [19, 20, 21, 22] aimed at modeling the forces not included in the SGP4 propagator. The deep-learning models developed in this study have been implemented using the H2O library [23], a Java open-source, distributed, fast, and scalable machine-learning platform, which we run through R [24] as the interface and programming environment.

3.1. Deep learning in H2O

The feedforward architecture is the theoretical framework used by H2O for developing the deep-learning algorithm.

The basic unit in the model is the neuron, represented in Fig. 1, which is a biologically inspired model of the human neuron. In humans, the varying strength of the neuron output signals travel along the synaptic junctions, and are then aggregated as an input for the activation of a connected neuron.

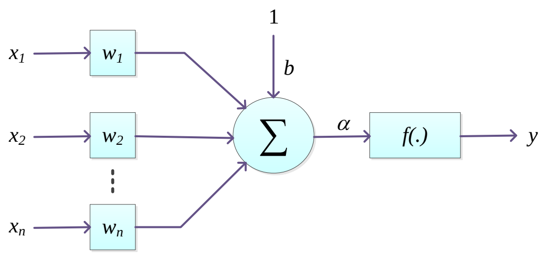

Fig. 1. Single neuron diagram

This neuron model is a computational unit that takes $x_1, \ldots, x_n$ as inputs, then calculates the weighted combination $\alpha = \sum_{i=1}^{n} w_i x_i + b$ of input signals, and finally produces an output signal $y = f(\alpha)$, which is transmitted to the following connected neurons. The function f represents the activation function used throughout the network, and the bias b represents the neuron activation threshold.

Multi-layer, feedforward neural networks consist of many layers of interconnected neuron units, as shown in Fig. 2, starting with an input layer to match the feature space, followed by multiple layers of nonlinearity, and ending with a linear regression or classification layer to match the output space. The neurons in the input, hidden, and output layers follow the basic logic of the single neuron described in Fig. 1.

Bias units are included in each non-output layer of the network. The weights linking neurons and biases with other neurons fully determine the output of the entire network. Learning occurs when these weights are adapted to minimize the error on training data.

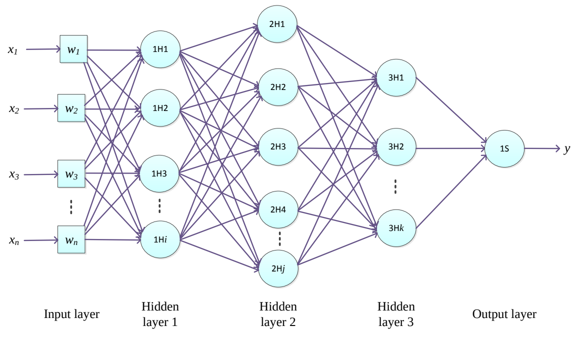

Fig. 2. Multi-layer neural network with 3 hidden layers. Squares denote neurons in the input layer, whereas circles represent neurons in the hidden and output layers

3.2. Programmed operations within the R environment

Figure 3 presents the sequence of operations that have been programmed within the R environment in order to model and forecast the error of SGP4. In this flowchart, the processes that are executed by the H2O kernel are marked in red.

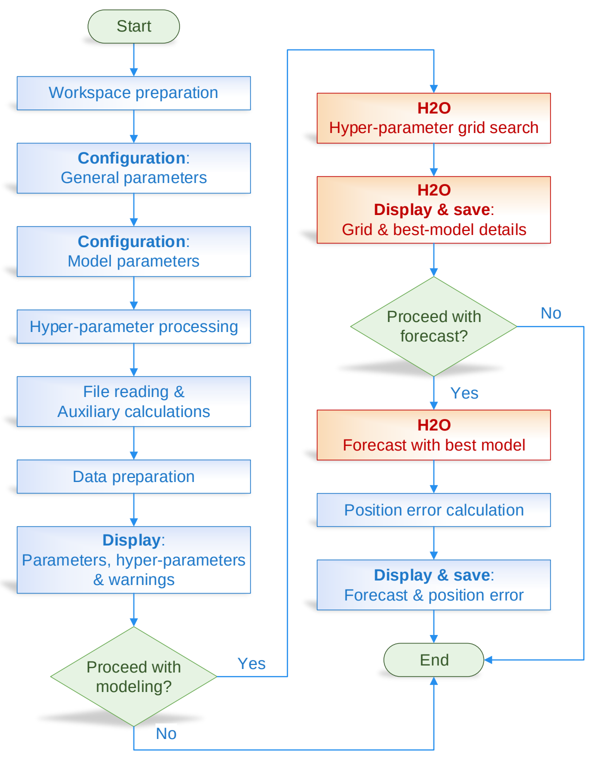
Start
Workspace preparation
Configuration: General parameters
Configuration: Model parameters
Hyper-parameter processing
File reading & Auxiliary calculations
Data preparation
Display: Parameters, hyper-parameters & warnings
Proceed with modeling?
Yes
No
H2O Hyper-parameter grid search
H2O Display & save: Grid & best-model details
Proceed with forecast?
No
Yes
H2O Forecast with best model
Position error calculation
Display & save: Forecast & position error
End

Fig. 3. Flowchart showing the modeling and forecasting sequence of operations. Processes in blue take place within the R environment, whereas the red ones are executed by the H2O kernel

First, the R workspace is initialized by loading the libraries and auxiliary scripts whose functions will be called by the program. Then, two sections are dedicated to the configuration of all the necessary parameters. The first one includes some general parameters, such as the size and appearance of the figures, the instants at which the position error has to be determined, or the names of the files. The second collects other group of parameters that are specific to the model to be built, such as those concerning the set of variables that will be used, the number of past values from which the model has to predict future values, the length of the training, validation, and test data sets, the choice between detrending or not the time series, the data resolution to be used, or the hyper-parameter values for the deep-learning algorithm. Next, these hyper-parameters are processed in order to create two separate lists, one integrated by the hyper-parameters for which several possible values have been specified, and the other with the single-valued hyper-parameters. The reason for doing this is that the multi-valued hyper-parameters will be used later to perform a grid search for the best model among different combinations of their values.

After that, two ephemeris files are read, one generated by SGP4 and the other by AIDA. Different sets of variables are admitted, namely Delaunay, polar-nodal, and equinoctial variables, as well as orbital elements and cartesian coordinates. Once the ephemerides are known, some auxiliary calculations are done so as to deduce their time resolution and keplerian period, so that the instants to be included in each of the training, validation, and test data sets can be determined. Then, these data sets are prepared, which implies that everything is ready to start the modeling stage. Nevertheless, since that can be a very long process, a summary of the main parameters and hyper-parameters is displayed, including some deduced values, plots and possible warnings, so that the user can abort the execution in case some changes are deemed appropriate.

Next, data are transferred from the R environment to the H2O kernel, and a grid search starts the modeling process by trying different combinations of hyper-parameter values until some stopping criterion, based on either a maximum number of models to try or a time limit, is met. Then, the model that has achieved the best results is selected, and a summary of its characteristics, together with some relevant information from the grid-search results, is both displayed and saved. In view of this information, the user can choose to abort the execution before starting the forecasting stage in case the best model is not satisfactory. Otherwise, forecasting with the best model starts. It is worth mentioning that this process can be slower than it would be expected, due to the multiple data conversions that need to be done between the R and H2O formats, which actually constitutes the main drawback of using H2O for the machine-learning operations.

Finally, with the aim of assessing the quality of the obtained model, the position error of the resulting Hybrid SGP4 (HSGP4) ephemerides, that is, the SGP4 ephemerides corrected with the forecast of their error, is calculated for the complete time span, and the maximum position error is also determined at some particular instants that were specified in the configuration section. When the position error has been displayed and saved, together with the forecast from the best model, the program execution stops.

4. HSGP4 orbit propagator

This section presents the development of the forecasting part of some Hybrid SGP4 propagators, HSGP4, for the case of Galileo-type orbits. An effort has been made to achieve parsimonious models, that is, the simplest models with the least assumptions and variables but with the greatest explanatory power.

In order to facilitate the understanding of the following subsections, it should be recalled that ε_i represents the time series of the position and velocity errors, which can be expressed using any canonical or non-canonical system of variables.

4.1. Exploration of the time series

The first step in applying the hybrid methodology to SGP4 is understanding its behavior through the study of the distance error, as well as its three components in the tangent, normal, and binormal directions, the along-track, cross-track, and radial errors, obtained when SGP4 is compared with AIDA, the high-accuracy numerical propagator that is used to generate pseudo-observations. In order to do so, 20 TLEs from 10 different satellites of the Galileo constellation [25] were propagated during a 60-day span using both SGP4 and AIDA. By default, the output file of each propagator provides orbital elements, although it can be converted into Delaunay, polar-nodal, and equinoctial variables (the definition of the Delaunay, polar-nodal, and equinoctial variables in function of the orbital elements is given in Appendix A). Then, we calculated the time series

$$\varepsilon_t^{\mathbf{x}} = \mathbf{x}_t^{\mathrm{AIDA}} - \mathbf{x}_t^{\mathrm{SGP4}}, \tag{4}$$

where $\mathbf{x}$ represents each of the Delaunay, polar-nodal, and equinoctial variables, $\mathbf{x}_t^{\mathrm{AIDA}}$ is the pseudo-observation at epoch t, and $\mathbf{x}_t^{\mathrm{SGP4}}$ is the data obtained from SGP4 at the same epoch. Any of the three sets of six time series contains the complete information related to SGP4 errors, which are caused by both the perturbation forces not taken into account, on the one hand, and the truncation of the analytical theory, on the other.

To understand the real influence that each variable, as well as some of their combinations, may have on the accuracy of the SGP4 propagation of Galileo-type orbits, an exploratory data analysis was performed, so that the different variables and their combinations could be ranked in terms of their capability to reduce the distance error of SGP4 propagations. Modeling only the most influential variables allows achieving parsimonious models, which is always a desirable goal to pursue in machine-learning models.

In all the studied cases, the most suitable variables were the polar-nodal ones. Improving the accuracy of just one of the variables of this set, the argument of the latitude θ, allowed reducing significantly the position error. For example, in the case of the Galileo FM2 satellite, we checked that the total distance, along-track, cross-track, and radial errors after a 60-day propagation could be reduced ideally from 69.00, 68.30, 61.79, and 23.82 km to 3.83, 3.76, 2.39, and 3.41 km, respectively. Using any other set of variables, such as Delaunay variables, orbital elements or equinoctial variables, would have required the improvement of the combination of at least two variables to reach a comparable degree of position-error reduction.

4.2. Behavior of the ε_t^θ time series

Figure 4 shows the ε_t^θ time series for 10 TLEs from different satellites of the Galileo constellation. As can be noticed, the time series do not present a uniform behavior. This situation can also be observed when different TLEs from the same satellite are considered. Figure 5 plots ε_t^θ time series for 53 different TLEs from the Galileo 8 satellite. TLE dates span from March 2015 to December 2016.

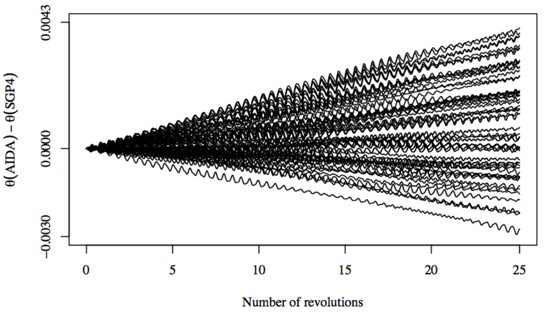

Fig. 5. ε_t^θ time series for 53 different TLEs from the Galileo 8 satellite

To illustrate our study, we use the ε_t^θ time series obtained from propagating two TLEs from Galileo 8 that we consider sufficiently representative of the qualitative behavior of the ε_t^θ time series in Fig. 5:

```
TLE 1
1 40545U 15017B   15087.10529976  .00000015  00000-0  00000+0 0  9997
2 40545 055.0895 094.8632 0005535 231.4671 034.4229 01.67457620    08

TLE 2
1 40545U 15017B   15109.17522156  .00000008  00000-0  00000+0 0  9994
2 40545 055.0839 094.3125 0004535 349.7734 010.1702 01.69234987   378
```

Figures 6 and 7 plot both time series for a propagation span of 30 days, which represents approximately 50 satellite revolutions. In the first case, Fig. 6, ε_t^θ shows a trend component, which decreases with time, a seasonal component, which shows a cyclic behavior with a period of about 14 hours, and data variance with an irregular behavior; in the second case, Fig. 7, the time series does not have a perceptible trend, although there are two seasonal components, the first of which seems to have a period of about 14 hours, whereas the second shows a considerably slower seasonal variation.

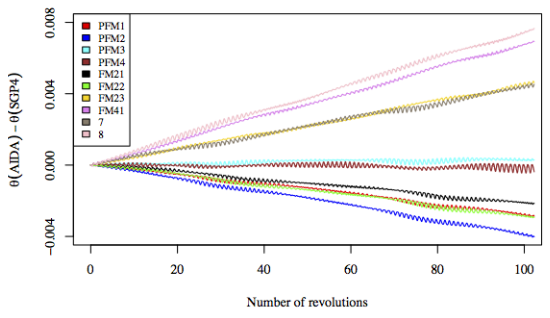

Fig. 4. ε_t^θ time series for 10 TLEs from different satellites of the Galileo constellation

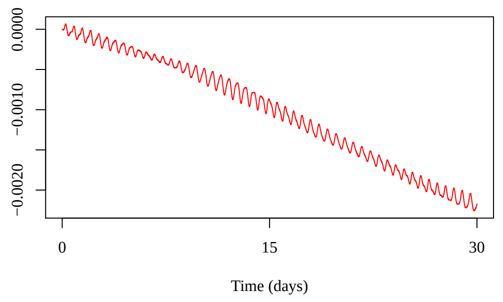

Fig. 6. TLE 1 corresponds to March 28, 2015 at 02:31:37

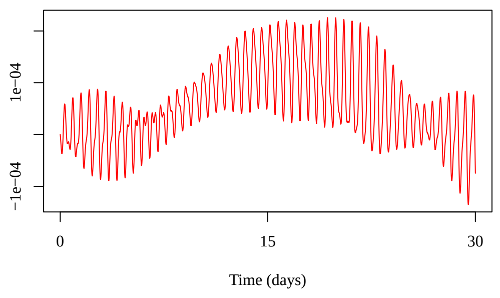

Fig. 7. TLE 2 corresponds to April 19, 2015 at 04:12:19

The periods of the seasonal components of both TLEs were calculated through ACF functions and periodograms, and in both cases a similar value to the time spanned by a satellite revolution, which is approximately 14 hours, was obtained.

Both time series were generated with a time resolution of 30 seconds, which amounts to 86 401 samples per series, so that a sufficient level of detail were available during the modeling process.

4.3. Deep-learning models of ε_t^θ for TLE 1 and TLE 2

In this subsection, we summarize the conclusions drawn from the application of deep learning to the modeling of the argument-of-the-latitude error that SGP4 makes in Galileo-satellite propagations. The study was conducted through trial and error on the two ε_t^θ time series presented in the previous subsection, whose qualitative behavior was considered to be representative of the rest of the series, one with both trend and seasonal components and the other with only a seasonal component, though highly variable. More than 190 tests were conducted to support these conclusions.

To start this study, we generated the training, validation, and test sets. These sets were created from the corresponding complete time series by using a sliding window of length $n + 1$, where n is the number of previous time-series values from which the next value has to be predicted. The number of neurons in the input layer is equal to n, while the number of neurons in the output layer is equal to 1. Each instance of the sliding window corresponds to a row in the matrix that constitutes the data set. It is worth noting that the validation set is unnecessary when cross-validation is used; in this case the training set is divided into k subsets, each serving as a validation subset for one of the k partial models to be developed, the one that makes use of the rest of subsets for training.

First, preliminary tests were dedicated to defining the hyper-parameter space to be explored during grid searches so that computing time remained under a reasonable limit of 3 days. As a result, we limited grid searches to a maximum number of 100 models per test, and considered variations in the following hyper-parameters: number of hidden layers, number of hidden neurons, activation functions, and L1-L2 regularization parameters.

Then, we focused on clarifying the effect of the number of inputs to the algorithm, which represents the number of previous time-series samples on which the algorithm bases its forecast for the next value. It is worth noting that this number is equivalent to the number of neurons n in the input layer, which has a significant influence on model complexity and therefore on computing time. We ascertained a close connection between n and the number of observations in a satellite revolution, p; therefore we started considering complete numbers of satellite revolutions to provide the inputs to the algorithm, and, by extension, also to create the training, validation, and test data sets.

It is worth noting that, although it could seem that it holds little interest, we tested all the developed models during the training interval, that is, we used them to make predictions over data previously used for training the models. On the one hand, that served as an initial filter to discard certain models that were not able to predict even those data. On the other hand, it allowed us to detect overfitting situations, whenever the fit was good during the training interval, but bad during the test interval.

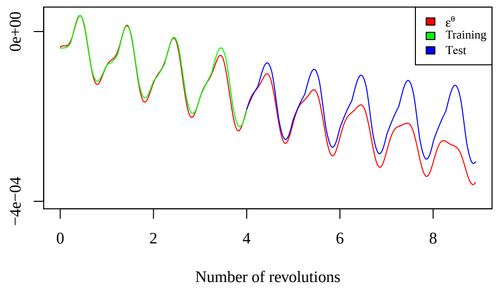

Fig. 8. NN[1] case. The ε_t^θ time series is displayed in red, green represents the forecast during the training interval, and blue during the test interval

After that, we concentrated our attention on modeling the time series that includes a trend component. We tried to ascertain what the minimum number of satellite revolutions for training should be when 1 revolution was considered as the input interval. In the test

NN^1, performed with a data resolution of 30 seconds, we verified that 4 revolutions using cross-validation was enough for training and validation. A model with 1 hidden layer provided with 74 neurons, and Maxout as activation function, showed very good performance during the following 5-revolution test interval (Fig. 8), approximately 3 propagation days, during which the distance error was reduced from 9.93 to 2.51 km.

Another test, NN^2, with a double number of satellite revolutions for training and cross-validation, 8, was conducted. Results were very similar to those of NN^1: the best model, provided with 1 hidden layer with 73 neurons and the Maxout activation function, was capable of reducing distance error from 19.9 to 3.3 km after a 9-revolution propagation, approximately 5 days.

In tests NN^3 and NN^4 (Fig. 9) we verified that standard validation outperforms cross-validation when using the same interval, 6 satellite revolutions, for training and validation. In both cases, the best model was provided with 1 hidden layer and the Maxout activation function. The distance error was reduced from 12.85 to 3.38 km (cross-validation) or to 2.15 km (standard validation) after a 7-revolution propagation, approximately 4 days.

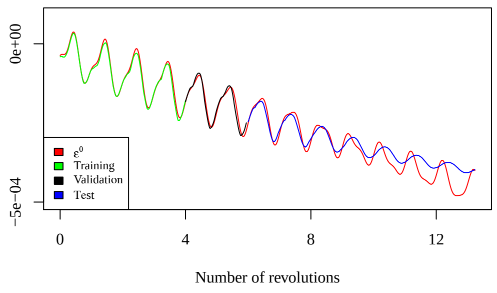

Fig. 9. NN^4 case. The ε_t^θ time series is displayed in red, the forecast during the training interval in green, during the validation interval in black, and during the test interval in blue

From test NN^5 to NN^8 data resolution was reduced, and observations were considered every 60 seconds instead of every 30. In general, that reduction led to worse results. In the case NN^5, the models provided by a hyper-parameter grid search did not reveal any relevant information. The same happened when we increased the number of neurons in the input layer to 1720, 2 satellite revolutions, case NN^6. Then, we doubled the number of revolutions in the training data set from 4 to 8, test NN^7 (Fig. 10). The 2-hidden-layer architecture, with Maxout activation function and 32+1 hidden neurons, was able to reproduce the behavior of the trend component of the time series but not the seasonal component. The distance error was reduced from 26.88 to 5.75 km after 13 satellite revolutions, approximately 8 propagation days. Finally, in the test NN^8 we changed from cross-validation to standard validation, and verified that the resulting models were unable to predict the trend component of the ε_t^θ time series, and therefore did not improve the results of the classical SGP4 propagator.

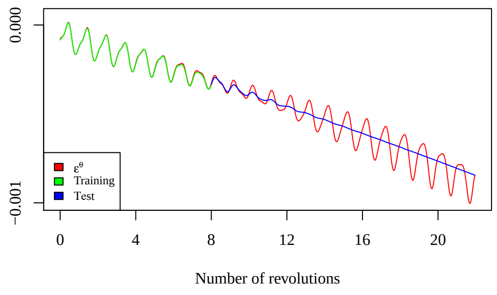

Fig. 10. NN^7 case. The ε_t^θ time series is displayed in red, green represents the forecast during the training interval, and blue during the test interval

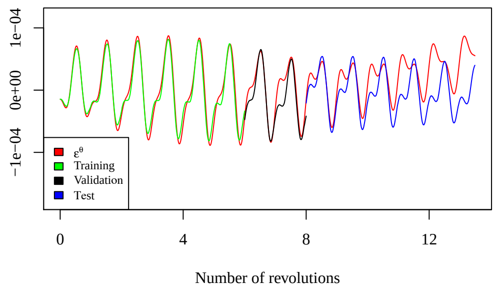

Fig. 11. NN^{10} case. The ε_t^θ time series is displayed in red, the forecast during the training interval in green, during the validation interval in black, and during the test interval in blue

In tests NN^9 and NN^{10} we analyzed the other ε_t^θ time series, the one which corresponds to TLE 2. Although it does not have a trend component, which should make it easier to be modeled, the fact is that its seasonal component is highly variable, with some changes in amplitude that complicate the task. Both models were capable of reproducing quite a complex pattern during the test interval, but only during the first 2 revolutions (Fig. 11).

In both cases the best model was the one provided with 1 hidden layer and the Maxout activation function.

Finally, we analyzed the effect of detrending the time series before modeling it, for which we worked on the TLE-1 ε_t^θ time series, since the other one lacked a significant trend. We decomposed the time series into trend, seasonal, and remainder components by means of the R function `stl()`, which is an implementation of Loess algorithm [26]. Figure 12 shows the three components obtained from the TLE-1 ε_t^θ time series. As can be observed, the seasonal and remainder components are roughly the same order.

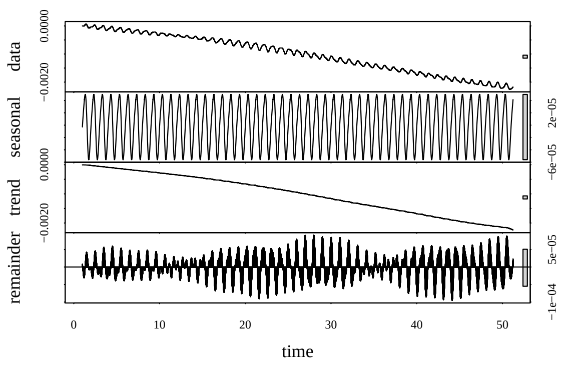

Fig. 12. STL decomposition of the TLE-1 ε_t^θ time series

Our forecasting strategy is based on modeling separately the trend, on the one hand, and the combination of the seasonal and remainder components, which we refer to as the *detrended* time series, on the other hand. In the first case, we considered three different statistical methods for forecasting the trend: linear regression, exponential smoothing, and Holt-Winters applied to the complete time series. It is worth noting that the trend component obtained from Loess decomposition is not necessary for the last method. In the second case, the combination of the seasonal and remainder components, that is, the detrended time series, was processed with deep learning techniques.

In the case of the trend, both linear regression and exponential smoothing methods, applied to modeling spans of 3, 4, and 5 revolutions (Fig. 13), showed good performance for long prediction horizons. For example, the distance error of SGP4 was reduced from 67.08 to approximately 7 km after 50 satellite revolutions by simply considering one of the forecasts of the trend that were developed, not taking into account any predictions of the rest of the time-series components (Fig. 14).

For the other part, the detrended time series, the best model corresponded to a 2-hidden-layer architec-

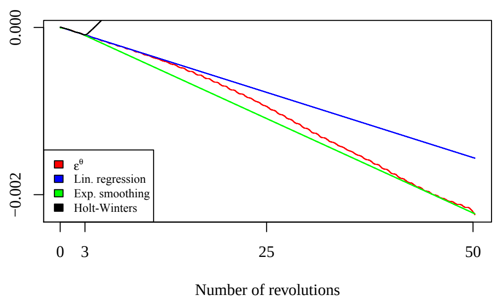

Fig. 13. Comparison between the TLE-1 ε_t^θ time-series trend generated by Loess decomposition, in red, and the trend forecasts obtained through linear regression, exponential smoothing, and Holt-Winters, in blue, green, and black, respectively, for a 3-revolution trend modeling span

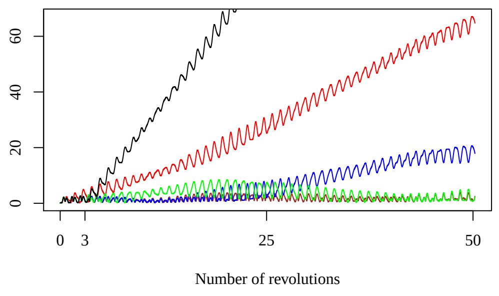

Fig. 14. Distance error (km) for the different forecasts of the TLE-1 ε_t^θ time-series trend shown in Fig. 13, following the same color convention, except for red, which represents the standard SGP4 propagation, and brown, which corresponds to the distance error of a hybrid propagation when the original trend, as generated by Loess decomposition, is considered

ture with 38+1 hidden neurons and Maxout activation function, NN[11] (Fig. 15), which used 4 revolutions for training. The 11.12-km distance error of SGP4 after 7 revolutions, approximately 4 days, which could be reduced to 2.48 km with just a hypothetical perfect prediction of the trend as generated by Loess decomposition, was finally lowered to 1.07 km with real predictions of both the trend and the rest of components.

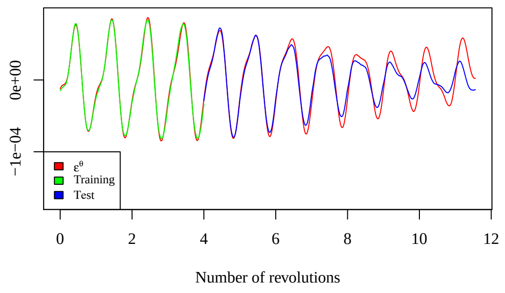

Fig. 15. NN[11] case. The ε_t^θ time series is displayed in red, green represents the forecast during the training interval, and blue during the test interval

5. Conclusions

A Hybrid SGP4 propagator based on machine-learning techniques has been developed for Galileo-type orbits. It applies deep learning in order to model the error of the well-known SGP4 propagator, with the aim of reproducing it later, and thus helping improve its precision. The numerical propagator AIDA has been used as the source of pseudo-observations, that is, the highly accurate ephemerides that have been used as the reference for determining the error of SGP4 propagations.

Initially, an exploratory data analysis has been undertaken in order to ascertain which set of variables is best suited for our goal. Since trying to achieve parsimonious models is always recommendable in machine-learning applications, a ranking of the most influential variables has been developed, with the aim of focusing only on the most essential. It has been verified that using polar-nodal variables allows obtaining a significant improvement by concentrating on just one variable, the argument of the latitude, whereas any other set of variables would require the combination of at least two variables to reach a comparable level of enhancement.

Then, attention has been focused on modeling the time series of the argument-of-the-latitude error. An effort has been made to try to characterize the qualitative behavior of those time series for the case of the SGP4 propagation of Galileo TLEs. As a result, two representative cases have been selected, their main difference being the presence or not of a trend component.

With the aim of facilitating the machine-learning operations that are necessary for modeling the time series, a program has been developed in the R environment. It is highly parameterized, therefore allowing for the launch of multiple executions with agility within a parallel computing environment, which permits trying more configurations in less time. After performing some preparatory tasks, it calls the H2O kernel, where the bulk of the computing is done. Finally, it displays and saves multiple plots and tables that help assessing the quality of the developed model.

This preliminary study shows that this technique is capable of reproducing the full time-series dynamics in the short term, especially when the time-series resolution is dense enough. That makes it a suitable technique for on-board applications. Concerning long-term propagation, it is essential to extract the time-series trend, so that a mixed approach can be applied: the trend can be modeled by means of conventional statistical methods, whereas deep-learning techniques can achieve acceptable models for the detrended time series.

Appendix A

The definition of the Delaunay, polar-nodal, and equinoctial variables in function of the orbital elements $(a, e, i, \omega, \Omega, M)$ is given by the following expressions:

- Delaunay variables:

$$\begin{aligned} l &= M & L &= \sqrt{\mu a} \\ g &= \omega & G &= L\sqrt{1-e^2} \\ h &= \Omega & H &= G\cos i \end{aligned}$$

  where μ represents the gravitational parameter of the Earth.

- Polar-nodal variables:

$$\begin{aligned} r &= \tfrac{a\,(1-e^2)}{1+e\cos f} & R &= \tfrac{e\,G\sin f}{a\,(1-e^2)} \\ \theta &= \omega + f & \Theta &= G \\ \nu &= \Omega & N &= H \end{aligned}$$

  where f corresponds to the true anomaly.

- Equinoctial elements:

$$\begin{aligned} \lambda &= M + \omega + \mathrm{I}\Omega & a &= a \\ h &= e\sin(\omega + \mathrm{I}\Omega) & k &= e\cos(\omega + \mathrm{I}\Omega) \\ p &= \left[\tan\left(\tfrac{i}{2}\right)\right]^{\mathrm{I}}\sin\Omega & q &= \left[\tan\left(\tfrac{i}{2}\right)\right]^{\mathrm{I}}\cos\Omega \end{aligned}$$

  with

$$\mathrm{I} = \begin{cases} +1 & \text{for direct equinoctial elements,} \\ -1 & \text{for retrograde equinoctial elements.} \end{cases}$$

Acknowledgments

This work has been funded by the Spanish State Research Agency and the European Regional Development Fund under Project ESP2016-76585-R (AEI/ERDF, EU). Support from the European Space Agency through Project Ariadna Hybrid Propagation (ESA Contract No. 4000118548/16/NL/LF/as) is also acknowledged. Computing has been done in the Beronia cluster (University of La Rioja), under support from ERDF-MINECO grant number UNLR-094E-2C-225.

References

[1] J. F. San-Juan, M. San-Martín, New families of hybrid orbit propagators based on analytical theories and time series models, Advances in the Astronautical Sciences 136 (2010) 547–565, paper AAS 10-136.

[2] J. F. San-Juan, M. San-Martín, D. Ortigosa, Hybrid analytical-statistical models, Lecture Notes in Computer Science 6783 (2) (2011) 450–462. doi:10.1007/978-3-642-21887-3_35.

[3] J. F. San-Juan, I. Pérez, M. San-Martín, Hybrid perturbation theories based on computational intelligence, in: Proceedings 5th International Conference on Astrodynamics Tools and Techniques, ICATT 2012, European Space Agency (ESA), Noordwijk, The Netherlands, 2012.

[4] J. F. San-Juan, M. San-Martín, I. Pérez, An economic hybrid J_2 analytical orbit propagator program based on SARIMA models, Mathematical Problems in Engineering 2012 (2012) 15 pages, article ID 207381. doi:10.1155/2012/207381.

[5] I. Pérez, J. F. San-Juan, M. San-Martín, L. M. López-Ochoa, Application of computational intelligence in order to develop hybrid orbit propagation methods, Mathematical Problems in Engineering 2013 (2013) 11 pages, article ID 631628. doi:10.1155/2013/631628.

[6] M. San-Martín, Métodos de propagación híbridos aplicados al problema del satélite artificial. Técnicas de suavizado exponencial, Ph.D. thesis, University of La Rioja, Spain (2014).

[7] I. Pérez, Aplicación de técnicas estadísticas y de inteligencia computacional al problema de la propagación de órbitas, Ph.D. thesis, University of La Rioja, Spain (2015).

[8] J. F. San-Juan, M. San-Martín, I. Pérez, Hybrid perturbation methods: modelling the J2 effect through the Kepler problem, Advances in the Astronautical Sciences 155 (2015) 3031–3046, paper AAS 15-207.

[9] M. San-Martín, I. Pérez, J. F. San-Juan, Hybrid methods around the critical inclination, Advances in the Astronautical Sciences 156 (2016) 679–693, paper AAS 15-540.

[10] J. F. San-Juan, M. San-Martín, I. Pérez, Application of the hybrid methodology to SGP4, Advances in the Astronautical Sciences 158 (2016) 685–696, paper AAS 16-311.

[11] J. F. San-Juan, M. San-Martín, I. Pérez, R. López, Hybrid SGP4: tools and methods, in: Proceedings 6th International Conference on Astrodynamics Tools and Techniques, ICATT 2016, European Space Agency (ESA), Darmstadt, Germany, 2016.

[12] J. F. San-Juan, M. San-Martín, I. Pérez, R. López, Hybrid perturbation methods based on statistical time series models, Advances in Space Research 57 (8) (2016) 1641–1651, Advances in Asteroid and Space Debris Science and Technology - Part 2. doi:10.1016/j.asr.2015.05.025.

[13] J. F. San-Juan, I. Pérez, M. San-Martín, E. P. Vergara, Hybrid SGP4 orbit propagator, Acta Astronautica 137 (2017) 254–260. doi:10.1016/j.actaastro.2017.04.015.

[14] A. Morselli, R. Armellin, P. Di Lizia, F. Bernelli-Zazzera, A high order method for orbital conjunctions analysis: sensitivity to initial uncertainties, Advances in Space Research 53 (3) (2014) 490–508. doi:10.1016/j.asr.2013.11.038.

[15] D. Brouwer, Solution of the problem of artificial satellite theory without drag, The Astronomical Journal 64 (1274) (1959) 378–397. doi:10.1086/107958.

[16] F. R. Hoots, R. L. Roehrich, Models for propagation of the NORAD element sets, Spacetrack Report #3, U.S. Air Force Aerospace Defense Command, Colorado Springs, CO, USA (1980).

[17] F. R. Hoots, P. W. Schumacher, Jr., R. A. Glover, History of analytical orbit modeling in the U.S. space surveillance system, Journal of Guidance, Control, and Dynamics 27 (2) (2004) 174–185. doi:10.2514/1.9161.

[18] D. A. Vallado, P. Crawford, R. Hujsak, T. S. Kelso, Revisiting spacetrack report #3, in: Proceedings 2006 AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Vol. 3, American Institute of Aeronautics and Astronautics, Keystone, CO, USA, 2006, pp. 1984–2071, paper AIAA 2006-6753. doi:10.2514/6.2006-6753.

[19] A. K. Palit, D. Popovic, Computational Intelligence in Time Series Forecasting: Theory and Engineering Applications, Advances in Industrial Control, Springer-Verlag, London, UK, 2005. doi:10.1007/1-84628-184-9.

[20] D. T. Larose, C. D. Larose, Discovering Knowledge in Data: An Introduction to Data Mining, 2nd Edition, Wiley Series on Methods and Applications in Data Mining, John Wiley & Sons, Hoboken, NJ, USA, 2014. doi:10.1002/9781118874059.

[21] M. A. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015, online book. URL http://neuralnetworksanddeeplearning.com/

[22] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, Adaptive Computation and Machine Learning series, MIT Press, Cambridge, MA, USA, 2016.

[23] H2O user guide.

[24] R Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

[25] M. Lara, J. F. San-Juan, L. M. López-Ochoa, P. J. Cefola, Long-term evolution of Galileo operational orbits by canonical perturbation theory, Acta Astronautica 94 (2) (2014) 646–655. doi:10.1016/j.actaastro.2013.09.008.

[26] R. B. Cleveland, W. S. Cleveland, J. E. McRae, I. Terpenning, STL: a seasonal-trend decomposition procedure based on Loess, Journal of Official Statistics 6 (1) (1990) 3–33.