OpenOrb Tutorial v.0.2

Mikael Granvik (mgranvik@iki.fi)
Jenni Virtanen (jenni.virtanen@fgi.fi)

April 3, 2012

```
#=============================================================================#
#                                                                             #
# Copyright 2002-2011,2012                                                    #
# Mikael Granvik, Jenni Virtanen, Karri Muinonen, Teemu Laakso,               #
# Dagmara Oszkiewicz                                                          #
#                                                                             #
# This file is part of OpenOrb.                                               #
#                                                                             #
# OpenOrb is free software: you can redistribute it and/or modify it          #
# under the terms of the GNU General Public License as published by           #
# the Free Software Foundation, either version 3 of the License, or           #
# (at your option) any later version.                                         #
#                                                                             #
# OpenOrb is distributed in the hope that it will be useful, but              #
# WITHOUT ANY WARRANTY; without even the implied warranty of                  #
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU           #
# General Public License for more details.                                    #
#                                                                             #
# You should have received a copy of the GNU General Public License           #
# along with OpenOrb. If not, see <http://www.gnu.org/licenses/>.             #
#                                                                             #
#=============================================================================#
```

Contents

Chapter 1

Short introduction to OpenOrb

1.1 What OpenOrb does and what it doesn't

OpenOrb does

- compute an orbit for an object if the user provides a list of astrometric positions for that object.
- integrate orbits.
- dynamically classify orbits.
- a slew of other tasks related to asteroids and orbit computation.

OpenOrb does not

- do image processing. OpenOrb does not do *anything* with images.

1.2 Orbital inversion theory

In the Bayesian framework, the parameters to be estimated are treated as random variables and the complete solution to the inverse problem is contained in the parameters' posterior probability densities. The posterior distribution $p_{\rm p}$ is proportional to the a priori ($p_{\rm pr}$) and the observational error (p_ϵ) PDFs:

$$p_{\rm p}(\mathbf{P}) \propto p_{\rm pr}(\mathbf{P}) p_\epsilon(\Delta\psi(\mathbf{P})) \tag{1.1}$$

where $\mathbf{P}$ refers to the orbital elements and $\Delta\psi(\mathbf{P})$ stands for the observed minus computed ($O-C$) residuals [MB93, Vir05]. The a priori often used is Jeffreys' a priori which secures the invariance of the results. For example,

the collision probability does not depend on the type of orbital elements used in the analysis [VM06].

If you want to transform the computed orbital-element PDF to another set of variables such as other types of orbital elements or ephemerides, you must also transform (that is, propagate) the weights to the new variables [MB93]:

$$p(\mathbf{F}) = \int dP p(\mathbf{P}) \delta_{\mathrm{D}}(\mathbf{F} - \mathbf{F}(\mathbf{P})) = \frac{1}{\det(\frac{\delta \mathbf{F}}{\delta \mathbf{P}})_{\mathbf{F}}} p(\mathbf{P}(\mathbf{F})), \tag{1.2}$$

where $\mathbf{F}(\mathbf{P}) = (F_1(\mathbf{P}), \ldots, F_k(\mathbf{P}))^T$ is a set of functions of the orbital elements, δ_{D} is Dirac's function, and $\det(\frac{\delta \mathbf{F}}{\delta \mathbf{P}})_F$ is the determinant of the Jacobian matrix for the transformation $\mathbf{P} \rightarrow \mathbf{F}$. The propagation of the weights is done automatically for the PDFs produced by the "official" OpenOrb executable, `oorb`.

1.3 Numerical Methods

1.3.1 Orbital inversion

Ranging

Ranging maps the non-Gaussian orbital-element PDF with a given number of sample orbits [VMB01, MVB01]. Each sample orbit is computed using the following scheme: Two observations are chosen from the data set and a random deviate is added to all four coordinates to mimic observational noise. Next, a random topocentric distance is generated for the first observation date using a, typically, broad interval. The topocentric distance for the second observation date is generated from an interval relative to the topocentric distance on the first observation date. Since the location of the observatory with respect to the Sun is usually known, the four plane-of-sky coordinates and the two topocentric distances can be transformed into two heliocentric positions corresponding to the two observation dates. Using well-established methods in celestial mechanics, an orbit can be computed using the two heliocentric positions. The generated sample orbit is then used to compute ephemerides for the other observation dates. If the residuals are acceptable and the PDF value is good enough with respect to the until-then best-fit orbit, the sample orbit is accepted. The inversion can be sped up by iteratively adjusting the intervals for the topocentric distance either by performing the simulation for a smaller number of sample orbits before performing the full-scale inversion [VTMB03] or by starting with only two observations and adding more observations step by step [GM05].

MCMCRanging

Markov-Chain Monte Carlo version of the Ranging method.

Least-squares with linearized covariances

Least-squares with linearized covariances (LSL)...

1.3.2 Integration

- Bulirsch-Stoer extrapolation method
  - n-body dynamical model contains 8 planets + Moon + Pluto
  - leading relativistic term by the Sun (perihelion shift)

Chapter 2

Installing OpenOrb

2.1 Software requirements

In addition to basic Unix tools such as `tar` and `make`, using OpenOrb only requires a Fortran 90/95 compiler—we are mostly using the free `gfortran` compiler (http://gcc.gnu.org/)—and `gnuplot` (http://www.gnuplot.info) for automatic generation of plots.

2.2 Producing the executable

Extracting the source code archive will produce a directory called `OpenOrb`:

```
tar xvzf OpenOrb-vN.N.tar.gz
```

Change directory to the `OpenOrb` directory:

```
cd OpenOrb
```

Run configuration script to set up the chosen compiler and its command-line switches:

```
./configure [COMPILER] [opt | deb]
```

Here `COMPILER` can be one of the following: gfortran, g95, lahey, intel, sun, compaq, or absoft. The option `opt` will produce optimized code for production use, whereas `deb` will produce code suitable for debugging. The compiler commands and switches are explicitly given in (and should be modified through) the `make.config` file. Next change to the `main` directory and

produce the `oorb` executable:

```
cd main
make oorb
```

Assuming everything compiled without problems[1] the `oorb` executable is now in the `main` directory.

To allow a flexible usage of the executable, you may want to make the following changes or additions to your shell's configuration file:

```
export PATH=$PATH:/path/to/oorb
```

If you have access to `gnuplot` and want to make automatic plots of the orbital-element PDFs, you should define the `$OORB_GNUPLOT_SCRIPTS_DIR` environment variable:

```
export OORB_GNUPLOT_SCRIPTS_DIR=/path/to/OpenOrb/gnuplot/
```

2.3 Getting and processing required data files

Access to the following data files is required to run `oorb`:

- OBSCODE.dat http://www.cfa.harvard.edu/iau/lists/ObsCodes.html
- TAI-UTC.dat http://hpiers.obspm.fr/eop-pc/
- ET-UT.dat
- de405.dat

The most recent versions of the three first files are incorporated in the OpenOrb package. OBSCODE.dat is updated every night by the Minor Planet Center. However, if you do not use observations from observatories that have recently acquired an observatory code, there is no need to update this file more often than, say, once every year or so. Note that the format of the OBSCODE.dat is identical to the HTML page with the exception that all HTML tags and the header line have been deleted. TAI-UTC.dat and ET-UT.dat have to be updated manually. A suitable interval to check the availability of updates is every six months or so.

[1] The only known (compatibility) issue can be found at the end of modules/cl_options.f90 and requires commenting certain lines in/out. If you are using the latest gfortran compiler, everything should compile without errors.

To generate the JPL planetary ephemeris file `de405.dat`, go to the `OpenOrb/data/JPL_ephemeris/` directory

```
cd ../data/JPL_ephemeris/
```

and issue the following command:

```
make
```

If successful, the ASCII versions of the de405 ephemerides will first be automatically downloaded from the JPL FTP server, and then built to the binary file `de405.dat`. Finally, `de405.dat` is copied to the `data` directory and the intermediate files are deleted. Note that you do not have to rebuild the `de405.dat` file every time you get a new version of OpenOrb. However, since `de405.dat` is a binary file, it may have to be built separately for every operating system. The correctness of the generated ephemeris file can be checked with

```
make test
```

Finally, the `OORB_DATA` environment variable, which contains the path to the data directory (usually `OpenOrb/data/`), should be added to the environment:

```
export OORB_DATA=/path/to/data/directory/
```

Again, consider adding the `OORB_DATA` variable to your shell's configuration file.

2.4 Configuration file

The explanations and possible values for the configuration parameters are given in Table 2.4. The path to the configuration file `FILE` (path to default file is `OpenOrb/main/oorb.conf`) can either be given using the command line option `--conf=FILE` or using an environment variable:

```
export OORB_CONF=/path/to/configuration/FILE
```

If neither of these are used, it is assumed that the configuration file is called `oorb.conf` and resides in the working directory. Note that speci-

Option	Explanation
MANY OPTIONS	NOT MUCH TIME

Table 2.1: See configuration file. The options should be self-explanatory...

fying the path as a command-line parameter (`--conf=FILE`) overrides the environment variable (`$OORB_CONF`) which in turn overrides the default path (`./oorb.conf`).

2.5 Sample configuration file

If you are using Bash, it makes life quite a bit easier if the `.bashrc` file contains the following entries:

```
export PATH=$PATH:/path/to/oorb
export OORB_CONF=/path/to/oorb/configuration/file
export OORB_DATA=/path/to/data/directory/
export OORB_GNUPLOT_SCRIPTS_DIR=/path/to/OpenOrb/gnuplot/
```

Chapter 3

Using OpenOrb

The basic options (input, output, etc.) are given as command-line parameters when running `oorb`, but more detailed configuration parameters are defined using a configuration file.

3.1 File formats

3.1.1 Observations

- Current MPC format (.mpc).
- The Data Exchange Standard format used by, e.g., Pan-STARRS (.des).
- New MPC format (.mpc3).

3.1.2 Orbits

- The OpenOrb format (.orb).
- The Data Exchange Standard format used by, e.g., Pan-STARRS (.des).

3.1.3 Ephemeris

Current OpenOrb format (.eph).

3.2 Computing orbital-elements based on observed positions

The `oorb` executable currently has to ways of obtaining the orbital-element PDF based on the observed positions, Ranging (`--task=ranging`) and LSL (`--task=lsl`). Whereas Ranging maps the PDF rigorously, that is, without prior assumptions of its shape, LSL assumes that the PDF is Gaussian and derives a so-called single-point estimate for the orbital-element PDF. In practice, the single-point estimate consists of the nominal orbit and the corresponding hyperellipsoid which describes the uncertainty of the orbit. In general, Ranging is optimized for situations with scarce data. LSL, on the other hand, should only be used when there is enough data available to assume that the *true* shape of the orbital-element PDF is indeed more or less Gaussian. The orbital inversion is performed by using either one of the followong commands:

```
oorb --task=ranging --obs-in=FILE.mpc [ --orb-out=OUTFILE ]
   [ --separately ]

oorb --task=lsl --obs-in=FILE.mpc --orb-in=INFILE
   [ --orb-out=OUTFILE ] [ --separately ]
```

Here `FILE.mpc` contains the observed positions in the Minor Planet Center (MPC) format, and the file may contain more than one object in which case the number or temporary designation separates different sets. The observational uncertainty is specified in the configuration file. Currently, OpenOrb also allows observations to be fed using the "New MPC Format" the fate of which is still uncertain. In the New MPC Format (OpenOrb recognizes it by the suffix `.mpc3`) observations are, among other things, allowed to have individual uncertainties. For LSL, `oorb` needs at least one starting point which is supplied using INFILE. If `--orb-out=OUTFILE` is omitted, the resulting orbital-element PDF is written to standard out. With the `--separately` option the orbit or orbits for each separate observation set is written to separate file the name of which is defined by the designation or number of the observation set.

3.2.1 Tips for using Ranging

- Make a reasonable assumption for the observational uncertainty σ_{obs}, e.g., the observatory's historical RMS for $O - C$ residuals.

- If using relative weights for sample orbits (non-uniform sampling), make sure the acceptance window ($[-c_{\mathrm{acc}} \times \sigma_{\mathrm{obs}}, +c_{\mathrm{acc}} \times \sigma_{\mathrm{obs}}]$, where c_{acc} is the acceptance-window multiplier) is large enough to allow sampling without cut-offs due to residuals. In practice, make sure the residual stamps figure do not show cut-offs.

- The generation window ($[-c_{\mathrm{gen}} \times \sigma_{\mathrm{obs}}, +c_{\mathrm{gen}} \times \sigma_{\mathrm{obs}}]$, where c_{gen} is the generation-window multiplier) should not be larger than the acceptance window. Typically, they have the same size, that is, $c_{\mathrm{acc}} = c_{\mathrm{gen}}$.

- Use a number of sample orbits which is enough to do statistics, but is still not overkill considering running time. Typically, 2,000–10,000 sample orbits is enough.

- As rule of thumb, use the two-body approximation for Ranging. The n-body corrections are usually only required for special cases such as collision-probability estimation.

3.3 Propagation of orbital elements from one epoch to another

The orbital-element PDF can be propagated to another epoch using the command

```
oorb --task=propagation --epoch-mjd-tt=MJD --orb-in=INFILE
   [ --orb-out=OUTFILE ]
```

The mode of propagation, that is, analytical two-body propagation or numerical n-body propagation using an integrator, can be defined in the configuration file. If `--orb-out=OUTFILE` is omitted, the resulting orbital-element PDF is written to standard out.

3.4 Computing ephemerides

Topocentric ephemerides and their uncertainties are computed based on the orbital-element PDF using the command

```
oorb --task=ephemeris --obs-code=CODE [ --epoch-mjd-utc=MJD |
     --epoch-mjd-tt=MJD | --timespan=DT1 --step=DT2 ]
     --orb-in=INFILE
```

where `CODE` is the observatory code assigned by MPC and MJD is the Modified Julian Date of the desired ephemerides either in UTC or TT. DT1 is the timespan (in days) over which ephemerides should be computed when starting from the epoch of the orbital-elements and DT2 is the frequency (in days) of the ephemerides. Positive values for DT1 and DT2 indicate going forwards in time from the epoch, whereas negative values indicate going backwards. The ephemeris prediction based on the sampled PDF is a discrete set of predicted positions on the sky and their individual weights, and the ephemeris prediction based on the single-point estimate is the nominal position with a 3-σ uncertainty ellipse. Note that the non-Gaussian uncertainty region equivalent to the Gaussian 3-σ region can also be obtained for the sampled prediction by normalizing the sum of the weights to unity and requiring that 99.73002% of the total weight (probability mass) is within the boundaries. The ephemerides are written to standard out.

3.5 Dynamical classification

For cases where the orbital-element PDF has been sampled, it is possible to classify the object in a dynamical sense by using the relative weight of sample orbits. The classification

```
oorb --task=classification --orb-in=INFILE
```

The results are written to standard output. The classification criteria can be changed by modifying the `getGroupProbabilities` subroutine in `StochastiOrbit_class.f90`. Note that the current implementation of the dynamical classification is only done in the osculating (a, e, i, q, Q) space, which means that, e.g., the classification of Jupiter Trojans is not carried out rigorously.

3.6 Conversion between file types

Use

```
oorb --task=mpctompc3 --obs-in=FILE1 [ --obs-out=FILE2]
```

to convert an observation file (FILE1) formatted according to the current MPC format to a file (FILE2) formatted according to the proposed new

MPC format. Observational uncertainties are taken from the configuration file. If they are not defined, the default uncertainty is zero. Use

```
oorb --task=mpc3tompc --obs-in=FILE1 [ --obs-out=FILE2]
```

to convert an observation file (FILE1) formatted according to the proposed new MPC format to a file (FILE2) formatted according to the current MPC format. Note that information regarding observational uncertainties is lost during the conversion. Use

```
oorb --task=astorbtoorb --astorb=FILE1 [ --orb-out=FILE2]
```

to convert the orbital elements and the (H,G) values in Ted Bowell's astorb.dat file to the .orb format. Here FILE1 is the astorb.dat file and FILE2 is the output .orb file. Again, if FILE2 is omitted the output is sent to standard out.

3.7 Example

In the `OpenOrb/test/` directory you can find two almost identical observation files in the MPC format, `K08K42V_ranging.mpc` and `K08K42V_lsl.mpc`. The difference between these files is that in the previous file all but the three first observations have been commented out, whereas in the latter none are commented out.

First, compute a Ranging solution for observation set #1:

```
oorb --task=ranging --obs-in=K08K42V_ranging.mpc \
     --orb-out=K08K42V_ranging.orb
```

The output is written both in the form of an orbital-element PDF file called `K08K42V_ranging.orb` and a more general output file called `K08K42V_ranging.sor`. Moreover, two separate plots were also produced: `K08K0042V_0.08_sor_keplerian_results.eps.gz` shows the orbital-element PDF and `K08K0042V_0.08_sor_residual_stamps.eps.gz` shows stamps of the $O - C$ residual distributions corresponding to each observation included in the inversion.

Then, use the orbits produced by Ranging to initiate the differential correction (LSL) using observation set #2:

```
oorb --task=lsl --obs-in=K08K42V_lsl.mpc \
```

```
     --orb-in=K08K42V_ranging.orb --orb-out=K08K42V_lsl.orb
```

Again, the output is written in the form of an orbital-element file called `K08K42V_lsl.orb` which contains the nominal orbit and the standard deviations and correlations for the elements, and a general output file called `K08K42V_lsl.ls`. The nominal orbit and the 1-σ and the 3-σ uncertainty ellipsoids are shown in plot `K08K0042V_37.80_ls_keplerian_results.eps.gz`.

To make a dynamical classification of the object, do

```
oorb --task=classification --orb-in=K08K42V_ranging.orb
```

The result strongly suggests that the object is something unknown to the current classification scheme and, in fact, the object is the first retrograde TNO discovered.

To propagate the orbital-element PDF to the current date (2008-08-12) epoch, do

```
oorb --task=propagation --orb-in=K08K42V_ranging.orb \
     --epoch-mjd-tt=54690 --orb-out=K08K42V_ranging_54690.orb
```

The file `K08K42V_ranging_54690.orb` contains the orbital-element PDF at the new epoch.

To compute follow-up ephemerides (topocentric distance, R.A., Dec., and their corresponding time derivatives) for 10 dates with one-day intervals starting at 2008-08-12.0 TT, use

```
oorb --task=ephemeris --orb-in=K08K42V_ranging_54690.orb \
     --timespan=10.0 --step=1.0
```

The ephemerides are written to standard out.

Chapter 4

Developing your own applications based on OpenOrb

It is reasonably straightforward to develop your own applications in the OpenOrb framework due to the object-oriented approach. Although Fortran 90/95 is not a pure object-oriented language, we have mimicked object-oriented concepts in the software [DNS97].

More to be added later if there is interest...

Chapter 5

Communication

5.1 Getting help

Due to manpower limitations, we cannot promise to give personal assistance in all situations. It is assumed that the user is familiar with state-of-the-art orbit computation methods and Bayesian inversion theory through published literature [Vir05, BVMB02, VTB+08, for reviews, see]. However, if you intend to publish results obtained with OpenOrb and have doubt regarding the interpretation of the results, we encourage you to contact either Mikael Granvik (mgranvik@iki.fi) or Jenni Virtanen (jenni.virtanen@fgi.fi) for assistance.

If needed, there may be help available for developing source code based on OpenOrb to solve more complicated orbit-computation-related problems. However, the help has a price, which is typically to be paid in the currency which our funding agencies understand and accept, that is, as a co-authorship in the resulting publication(s).

5.2 Reporting bugs

OpenOrb has been tested during the years by comparing the results to the results obtained with either our older software or by software by colleagues. If you nevertheless find a bug, please report it to us so that we can fix the problem. The report should be concise and contain everything needed for reproducing the erroneous result. Send the report to MG (mgranvik@iki.fi) with the text "OpenOrb bug" in the subject line.

5.3 Acknowledging the use of OpenOrb

If you find OpenOrb useful in your work, please acknowledge our efforts by citing, in addition to the relevant original papers, the following paper on the OpenOrb software package:

Granvik, M., Virtanen, J., Oszkiewizc, D., Muinonen, K. (2009). *OpenOrb: Open-source asteroid-orbit-computation software including statistical ranging*, Meteoritics and Planetary Science **44**(12), 1853–1861.

Bibliography

[BVMB02] Edward Bowell, Jenni Virtanen, Karri Muinonen, and Andrea Boattini. Asteroid Orbit Computation. In William Bottke, Alberto Cellino, Paolo Paolicchi, and Richard P. Binzel, editors, *Asteroids III*, pages 27–43. University of Arizona Press, 2002.

[DNS97] Viktor K. Decyk, Charles D. Norton, and Boleslaw K. Szymanski. Introduction to Object-Oriented Concepts using Fortran90. `http://exodus.physics.ucla.edu/Fortran95/`, 1997.

[GM05] Mikael Granvik and Karri Muinonen. Asteroid identification at discovery. *Icarus*, 179(1):109–127, dec 2005.

[MB93] Karri Muinonen and Edward Bowell. Asteroid orbit determination using Bayesian probabilities. *Icarus*, 104(2):255–279, August 1993.

[MVB01] Karri Muinonen, Jenni Virtanen, and Edward Bowell. Collision probability for Earth-crossing asteroids using orbital ranging. *Cel. Mech. Dyn. Astron.*, 81(1–2):93–101, September 2001.

[Vir05] Jenni Virtanen. *Asteroid orbital inversion using statistical methods*. PhD thesis, University of Helsinki, June 2005.

[VM06] Jenni Virtanen and Karri Muinonen. Time evolution of orbital uncertainties at discovery for the impactor candidate 2004 AS_1. *Icarus*, 184(2):289–301, October 2006.

[VMB01] Jenni Virtanen, Karri Muinonen, and Edward Bowell. Statistical Ranging of Asteroid Orbits. *Icarus*, 154(2):412–431, December 2001.

[VTB+08] Jenni Virtanen, Gonzalo Tancredi, Gary M. Bernstein, Timothy Spahr, and Karri Muinonen. Transneptunian Orbit Computation. In M. A. Barucci, H. Boehnhardt, D. P. Cruikshank, and

A. Morbidelli, editors, *The Solar System Beyond Neptune*, pages 25–40. University of Arizona Press, Tucson, 2008.

[VTMB03] Jenni Virtanen, Gonzalo Tancredi, Karri Muinonen, and Edward Bowell. Orbit computation for transneptunian objects. *Icarus*, 161(2):419–430, February 2003.