#orbitN

Symplectic integrator for near-Keplerian planetary systems

#==#

README.TXT

*** orbitN comes with ABSOLUTELY NO WARRANTY ***
*** Use at your own risk. DO NOT DISTRIBUTE ***
*** except under GNU General Public License ***

Richard E. Zeebe
University of Hawaii at Manoa
1000 Pope Road, MSB 629
Honolulu, HI 96822, USA
correspondence to:
orbitN.code@gmail.com

___ ___ ___

Copyright (c) 2023 Richard E. Zeebe
This file is part of orbitN

This program 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, version 3.

This program 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 this program. If not, see <http://www.gnu.org/licenses/>.

___ ___ ___

When using orbitN, cite as (ZeebeAJ23 hereafter):

Zeebe, R. E. orbitN: A symplectic integrator for planetary systems
dominated by a central mass - Insight into long-term solar system
chaos. The Astronomical Journal, 2023.

#==#

(0) orbitN

orbitN is a second order symplectic integrator developed
with the primary goal of generating accurate and reproducible
long-term orbital solutions for near-Keplerian planetary systems
with a dominant mass M0. Among other features, orbitN includes
M0's quadrupole moment, a lunar contribution, and post-Newtonian
corrections (1PN) due to M0 (fast symplectic implementation).
To reduce numerical roundoff errors, orbitN features Kahan
compensated summation. orbitN version 1.0 focuses on hierarchical
systems without close encounters but can be extended to include
additional features in future versions. orbitN version 1.0
uses Gaussian units, i.e., length, time, and mass are expressed
in units of au, days, and fractions of M0.

(1) Installation

orbitN should run under OSs including Linux and Mac OS.
To compile and run orbitN, a C compiler is required. Once a
C compiler is available, installing and running orbitN should
be fairly simple. Many Linux distributions include, for
instance, gcc or provide convenient options to install required
packages (see below for Mac OS).

When the C compiler and tools such as 'make' are available,
proceed as follows. On github, click on orbitN-x.x.x.tar.gz,
then on the symbol "Download raw file". If instead you download
the full github main zip package, first extract orbitN-x.x.x.tar.gz
from it. Save the compressed file (orbitN-x.x.x.tar.gz) to the
directory of your choice, open a terminal, cd to that directory
and run (replace x.x.x by actual version number):

```
$ tar -xvf orbitN-x.x.x.tar.gz
$ cd orbitN-x.x.x
$ make
$ ./orbitN
```

Step 3 should produce a number of object files (.o) in the
'src' folder and the executable orbitN in the root folder.
Step 4 runs the program.

By default, orbitN output will be written to .dat and .log
files in orbitN's root directory, see below for content.

orbitN includes a utility program for post-run conversion
of state vectors to Keplerian elements (xv2elm). If the selected
conversion involves the masses of individual bodies, the user
needs to provide the original mass/coordinate input file of the
run. To use xv2elm, edit the input section of xv2elm.c and
execute:

```
$ make allelm
$ ./xv2elm
```

(1a) Mac OS

Installing Xcode, for instance, should provide the necessary
tools, including a C compiler. The steps to install and run orbitN
are then analogous to the Linux steps described above.
Xcode may be found on the system install disk or can be
downloaded. However, note that the file size for the latest
Xcode version may be very large. Alternatively, depending
on the Mac OS version, command line tools only may be installed
that require less disk space.

(2) Running orbitN

Running orbitN integrations to solve a specific user problem
can be accomplished by adjusting orbitN settings, i.e.,
through user modifications of code and/or input files
via different options. (1) By editing the main source code
files or (2) by adding custom simulations under ./sim/..
(see below).

For example, the files orbitN.h and orbitN.c include top-level
input sections that allow modifying input parameters
(C preprocessor macros, see below). Once set, the parameter
values take effect after recompilation (use make).

(2a) orbitN Input Sections

The input macros either have values or turn ON/OFF certain
options. To turn options off (disable): add a U in front of
the macro string. For example,

```
#define PN
```

turns Post-Newtonian corrections ON.

```
#define UPN
```

turns Post-Newtonian corrections OFF.

The input macros of orbitN.h and orbitN.c are included
below and should be self-explanatory (for more details,
see ZeebeAJ23). Simulations with advanced settings that
go beyond the options provided in the input sections will
require the user to modify the source code and/or other
(non-input) macros.

orbitN.h

```
/*==================== Input Section =========================*/

#define JJ 10     /* No. of Bodies incl. central mass      */

/* turn macros off (disable): add U in front of string     */
#define FASTDRIFT /* merge drift steps                     */
#define PN        /* Post-Newton (requires FAST)           */
#define J2        /* Solar Quadrupole Moment               */
#define LUNAR     /* Lunar: EMB quadrupole (Quinn91)       */

#define CORR      /* Symplectic corrector ON/OFF           */
#define STAGE 6   /* Symplectic corrector stage: 2, 4, 6   */

#define KAHAN     /* compensated summation (requires FAST) */

/* FOR ACCURACY AND STORAGE USE STATE VECTORS X V !         */
#define ORB_XV    /* output state xv. else: orb elements   */
#define ANG_DEG   /* output angle units: deg. else: rad    */
#define ELM_MJ 1  /* 0/1 use M0 or M0+mj to calc elements  */

/* CPU info (log file). disable in case of warnings/errors */
#define CPUINFO

/*============================================================*/
```

orbitN.c

```
/*==================== Input Section =========================*/

 /* set time step (days), t0, tend (days), save interval (steps) */
 /* Y2D = 365.25 (years -> days)                                 */
 const double in_dt = -2.0;
 double       in_t0 =  0.e3*Y2D;
 double     in_tend = -1.e4*Y2D;
 const int in_sstep =  73050;

/*==============================================================*/
```

(2b) orbitN Input for Simulations (./sim)

The orbitN package includes several example simulations,
see directory ./sim. These can be engaged by running shell
scripts in the ./sim/.. subdirectories.

Compiling default- vs. simulation settings is controlled by
the macro "SIM", which is set in the Makefile. For example,
the command:

```
$ make SIM=1
```

as used in the run/compile scripts under ./sim/.. invokes input
parameter values defined in the sim.h and sim.c files in the
respective ./sim/.. folder. These parameter values are then used
instead of the default parameter values defined in the input
sections of orbitN.h and orbitN.c

To run the simulation examples, the user should inspect and, if
applicable, modify the scripts in ./sim/.. Note: the simulation
./sim/n-cluster-ensemble may require advanced knowledge of cluster
computing, bash, ssh, etc. All scripts need to have executable
permission, e.g., to activate and run:

```
$ chmod +x ./run
etc.
$ ./run
```

To activate all scripts, run actvtScripts in ./sim.

Hence the user has at least two options to solve a specific user-
defined problem with orbitN. (1) Adjust orbitN settings through
modification of input parameter values in the input sections of
orbitN.h and orbitN.c. (2) Add custom user simulations, following
the examples provided under ./sim.

(2c) orbitN Input File: Masses/Coordinates

orbitN is set to read the input coordinates of the bodies
in the simulation from an input file (orbitN-coord.inp).
The input includes the body masses (in units of the central mass
M0; M0 = index 0) and initial state vectors = position and
velocity (in au and au/day) in bodycentric coordinates at time
t0. The coordinate input list of bodies starts at index 1.

!!! The input file must maintain the original bit-by-bit format exactly !!!

If the input file is modified (beyond adding/removing bodies and
changing masses/state vectors) and after modifying includes, e.g.,
additional characters and/or spaces, reading of the input file
may fail. Note: the backslash character '\' signifies line
continuation. Do not add spaces after '\', at end of lines,
etc. The text in lines following '#' is ignored.

input: state vectors = initial position & velocity (bodycentric)
orbitN-coord.inp

```
-----------------------------------------------------------------
# body 1
m    x0    y0    z0    u0    v0    w0

# body 2
m    x0    y0    z0    u0    v0    w0

etc.,
------------------------------------------------------------------
```

where

```
-----------------------------------------------------------------
SYMBOL    m     x0    y0    z0    u0    v0    w0
UNIT      M0    [.... au ....]    [... au/d ...]
VARIABLE  mass     position          velocity
------------------------------------------------------------------
```

(3) orbitN Output

By default, orbitN coordinate output is written to orbitN-j.dat
files (j = body index) in orbitN's root directory with the
following columns.

output option: state vectors = position & velocity (bodycentric)

```
-----------------------------------------------------------------
SYMBOL    t     x    y    z    u    v    w
UNIT      day   [....au...]    [...au/d..]
VARIABLE  time   position       velocity
------------------------------------------------------------------
```

NOTE: FOR ACCURACY AND STORAGE, STATE VECTORS ARE RECOMMENDED.

output option: Keplerian elements

```
-----------------------------------------------------------------
SYMBOL    t     a    e    i    om    oom    vpi    mn
UNIT      day   au   -    [....... deg or rad ......]
VARIABLE  time  [.......... see below .............]
------------------------------------------------------------------
```

where:

```
a    /* semimajor axis */
e    /* eccentricity   */
i    /* inclination    */
om   /* ArgPerihelion  */
oom  /* LongAscNode    */
vpi  /* LongPerihelion */
mn   /* mean anomaly   */
```

Additional output files are generated as follows:

orbitN-erg.dat

```
-----------------------------------------------------------------
SYMBOL    t     (E-E0)/E0 (Lx-Lx0)/L0 (Ly-Ly0)/L0 (Lz-Lz0)/L0
UNIT      day       -                     -
VARIABLE  time  d(Energy)          d(Angular Momentum)
------------------------------------------------------------------
```

where d(y) is the fractional change in y.

```
FILE                CONTENT
-----------------------------------------------------------------
orbitN.log       input values, run log, wallclock time estimates
orbitN-final.sve final coordinates for potential restart
-----------------------------------------------------------------
```

The shell script 'cleanup' can be used to clean the orbitN root
directory.

!!! CAUTION: all executables, output, object files etc. will be deleted !!!
Only source (.c,.h), input, and a few other text files remain.

(4) Checks

The orbitN package includes a check option to test whether orbitN
produces accurate output on the user's local machine. For the
test, install and compile orbitN out-of-the-box (without any
changes). Next, run:

```
$ ./orbitN
$ ./checkDo
```

and inspect check.log. The log file should show no differences.

Despite different OS/machines/optimizations, orbitN should produce
identical outcome to machine precision. However, *ONLY* when output
in STATE VECTORS XV is selected in orbitN.h:

```
#define ORB_XV
```

The above has been tested using gcc up to optimization
level 3 ('-O3'). When output in Keplerian elements is selected,
agreement to machine precision can not be guaranteed because
of issues with trigonometric functions (see ZeebeAJ23).

(5) Subdirectories

```
'./check'  : check files

'./coords' : example input coordinate files

'./docs'   : pdf copy of ZeebeAJ23, README, license, etc.

'./sim'    : example simulations

'./src'    : C source code files
```

```
                                    $ soffice --convert-to pdf README.txt
```