M A N U A L F O R T H E M E R C U R Y I N T E G R A T O R

P A C K A G E V E R S I O N 6

by John E. Chambers

(with some subroutines supplied by Hal Levison and Martin Duncan)

Dedicated to the memory of Fabio Migliorini

Many thanks to all of you for reporting bugs and suggesting improvements. Special thanks to David Asher, Scott Manley and Eugenio Rivera for your help.

< Last modified 1 March 2001 >

N.B. If you publish the results of calculations using MERCURY, please reference the package using J.E.Chambers (1999) ''A Hybrid Symplectic Integrator that Permits Close Encounters between Massive Bodies''. Monthly Notices of the Royal Astronomical Society, vol 304, pp793-799.

C O N T E N T S

---

(1) I N T R O D U C T I O N

MERCURY is a general-purpose software package for doing N-body integrations. It is designed to calculate the orbital evolution of objects moving in the gravitational field of a large central body. For example MERCURY can be used to simulate the motion of the planets, asteroids and comets orbiting the Sun; or a system of moons orbiting a planet; or a planetary system orbiting another star.

MERCURY is written in Fortran 77. The code is slightly non standard (e.g. it contains 'end do' statements) but it should work using most compilers.

MERCURY currently includes the following N-body algorithms:

- A second-order mixed-variable symplectic (MVS) algorithm incorporating simple symplectic correctors (see J.Wisdom et al. 1996, Fields Instit. Commun. vol 10 pp217) - this is very fast but it cannot compute close encounters between objects.

- A general Bulirsch-Stoer - slow but accurate in most situations. You can use this when all else fails, or to test whether the other algorithms are appropriate for the problem you want to study.

- Conservative Bulirsch-Stoer - twice as fast as the general BS routine, but it will only work for conservative systems, in which accelerations are a function of position only (e.g. Newtonian gravity, but not General Relativity).

- Everhart's RA15 (RADAU) - about 2-3 times faster than the general version of Bulirsch-Stoer. Usually reliable, except for very close encounters or very eccentric (e.g. Sun grazing) orbits.

- Hybrid symplectic/Bulirsch-Stoer integrator - very fast but only moderately accurate. This algorithm can compute close encounters.

N.B. The symplectic integrators may give spurious results if some objects have highly eccentric orbits during an integration.

MERCURY includes the effects of Newtonian gravitational forces between bodies that are assumed to be point masses. It can also calculate non-gravitational forces for comets (using equations by B.Marsden et al. 1973. Astron. J. vol 78, pp211). You can include the effects of other forces, using some algorithms, by modifying the subroutine mfo_user.for.

The MERCURY package consists of several drivers and subroutines, together with a number of input files that you may want to alter to suit your application.

The Drivers

1) mercury6_1.for

This is the basic integration programme. It contains all the subroutines you need to carry out integrations using any of the algorithms described above. mercury6_1.for produces some output files that are in a machine-independent compressed format, and you will need the following programmes to convert this output into a format you can read.

2) element6.for

This programme converts an output file created by mercury6_1.for into a set of files containing Keplerian orbital elements for each of the objects in the integration. These files allow you to see how object's orbits change with time, and can be used as the basis for making graphs or movies using a graphics package.

3) close6.for

This programme converts an output file produced by mercury6_1.for
into a set of files containing details of close encounters between
objects during the integration.

Other files
----

1) mercury.inc

This contains constants and general parameters used by programmes in
the mercury package. You may want to alter some of the parameters before
you compile and run mercury6_1.for

2) swift.inc

This contains constants and parameters used in the subroutines written
by H.Levison and M.Duncan (1994, Icarus, vol 108, pp18). These
subroutines have names that begin with either 'drift' or 'orbel'.

N.B. If you change mercury.inc or swift.inc, you must recompile the
=== driver programmes (mercury6_1.for etc) for the changes to take
effect.

---

(2) I N I T I A L P R E P A R A T I O N S

Before using the MERCURY package for the first time, you should do the
following:

a) Make sure you have copies of these files:
mercury6_1.for
mercury.inc
swift.inc
element6.for
close6.for
message.in
files.in
You will need some additional input files to run the programmes in the
MERCURY package, but you can create these yourself (see the following
sections for details).

b) Compile mercury6_1.for. The precise command you use will depend on
your Fortran compiler. I suggest you call the executable version
of the programme mercury6
e.g. On Linux systems, try g77 -o mercury6 mercury6_1.for
On DEC Unix systems, try f77 -o mercury6 mercury6_1.for

c) Compile element6.for. I suggest you call the executable element6

d) Compile close6.for. I suggest you call the executable close6

---

(3) H O W T O D O A N I N T E G R A T I O N

==========

a) Make sure the compiled version of mercury6_1.for exists.

b) Make sure each of the input files described below exists, and alter
   them to suit your needs.

c) Run the compiled version of mercury6_1.for
   e.g. On Unix or Linux systems, use the command: ./mercury6

   For long integrations, you may want to run the programme in the
   background or as a batch job.

d) Read the information summary file produced by mercury6_1.for to make
   sure that no problems occurred during the integration. You can read
   this file while mercury6_1.for is still running.

The Input Files

1) files.in

This should contain a list of 10 names for the input and output files
used by mercury6_1.for. List each file name on a separate line.

The first 3 names should be input files that already exist.
In order, these are:

 i) A file containing initial data for the Big bodies in the integration
    (e.g. planets in the Solar System).

 ii) A file containing initial data for the Small bodies in the integration
     (e.g. asteroids or comets that you want to include).

 iii) A file containing parameters used by the integrator (e.g. start
      and end times of the integration etc.).

The MERCURY package includes some sample versions of these files which
you can use as templates. I call these big.in, small.in and param.in
respectively.

The last 7 names in files.in are the names that mercury6_1.for will use
for its output files. In order, these will contain:

 iv) Position and velocity information for the objects in the
      integration, produced at periodic intervals.
 v)   Details of close encounters that occur during the integration.
 vi) A summary of the integration parameters used in by the integrator,
      and a list of any events that took place (e.g. collisions between
      objects).
 vii) A dump file containing data for the Big bodies. You can use this
      to continue the integration if your computer crashes or the
      programme is interupted.
 viii) A dump file containing data for the Small bodies.
 ix) A dump file containing the integration parameters.
 x)   An additional dump file containing other variables used by
      mercury6_1.for

I usually call these 7 files xv.out, ce.out, info.out, big.dmp,

small.dmp, param.dmp and restart.dmp

2) big.in

This file contains the initial data for all the Big bodies in the integration EXCEPT for the central body (i.e. the Sun if you are integrating the Solar System). A Big body is defined as one that perturbs and interacts with all other objects during the integration.

Any lines beginning with the) character are assumed to be comments, and will be ignored by mercury6_1.for, however, you should not delete the first comment line beginning with)O+_06

o The first non-comment line should end with a word that tells the programme what format you use for the initial data. You should specify either

Cartesian = for xyz coordinates and velocities. Distances should be in AU and velocities in AU per day (1 day = 86400 seconds).

Asteroidal = Keplerian orbital elements, in an 'asteroidal' format.
i.e. a e I g n M, where
a = semi-major axis (in AU)
e = eccentricity
I = inclination (degrees)
g = argument of pericentre (degrees)
n = longitude of the ascending node (degrees)
M = mean anomaly (degrees)

Cometary = Keplerian orbital elements in a 'cometary' format.
i.e. q e I g n T, where
q = pericentre distance (AU)
e,I,g,n = as above
T = epoch of pericentre (days)

o The next line should end with the epoch of osculation in days (i.e. the time at which the initial coordinates/elements are valid).

E.g. the first few lines of big.in might look like this:

```
)O+_06 Big-body initial data  (WARNING: Do not delete this line!!)
) Lines beginning with ')' are ignored.
)---------------------------------------------------------------------
 style (Cartesian, Asteroidal, Cometary) = Asteroid
 epoch (in days) = 2451544.5
```

o The remaining lines provide data for each Big body. The first line for each body should begin with the body's name, having up to 8 characters.
After that you can include any of the following on the same line:

m = X where X is a real number, to indicate the body's mass in Solar masses. If you don't specify a value the mass is assumed to be 0.

r = X where X is a real number, to indicate the maximum distance from the body (in Hill radii) that constitutes a close encounter. If you don't include this the default is r=1

d = X where X is a real number, to indicate the density of the body in g/cm^3. If you don't include this the default is d=1

a1 = X where X is a real number, to indicate the A1 non-gravitational force parameter for this body. Realistically this should be zero for Big bodies (the default is 0).

a2 = X where X is a real number, to indicate the A2 non-gravitational force parameter for this body (the default is 0).

a3 = X where X is a real number, to indicate the A1 non-gravitational force parameter for this body (the default is 0).

E.g. the line might look something like this:

```
MARS   m=3.2271514450538653OE-07 d= 3.94
```

The next line(s) for a body should contain the 6 initial coordinates and velocities or the 6 orbital elements, separated by one or more spaces or carriage returns. After these numbers you should give the 3 components of spin angular momentum for the body, in units of solar masses AU^2 per day (if in doubt enter these as 0).

3) small.in

This file contains the initial data for all the Small bodies in the integration. A Small body is defined as one that only perturbs and interacts with Big bodies during the integration. Hence, Small bodies ignore one another completely (i.e they do not perturb one another, and they cannot collide with each other).
If you give these objects zero mass they will behave as test particles.

Any lines beginning with the) character are assumed to be comments, and will be ignored by mercury6_1.for, however, you should not delete the first comment line beginning with)O+_06

o The first non-comment line should end with a word that tells the programme what format you use for the initial data. The possible formats are the same as those in big.in

o The remaining lines provide data for each Small body. These are exactly analogous to the lines in big.in, except that you can also specify an epoch of osculation for each Small body using

ep = X where X is a real number. If you don't include this the default is X = the same as the epoch for the Big bodies. Small bodies with differing epochs will be integrated to the same epoch prior to the main integration.

E.g. the line might look something like this:

```
HALLEY   Ep=2446480.5  a1=0.04d-8 A2 =0.0155d-8
```

Note that if any of the Small bodies have different epochs than the Large bodies, the Small bodies must all have zero mass.

4) param.in

This file contains parameters that control how an integration is carried out. Any lines beginning with the) character are assumed to be comments, and will be ignored by mercury6_1.for, however, you

should not delete the first comment line beginning with)O+_06

The file should contain the following items, one per line and in this order (the programme actually searches for information after an '=' sign, so you may change the text of the message before this).

- The integration algorithm. Choose one of the following:

  ```
  mvs     : second-order mixed-variable symplectic
  bs      : Bulirsch-Stoer (general)
  bs2     :    "       "   (conservative systems only)
  radau   : RA15 (RADAU)
  hybrid  : hybrid symplectic/Bulirsch-Stoer integrator
  ```

- The time that the integration should start (in days). This doesn't have to be the same as the epoch of the Big bodies. Rather, the integrator will start producing output at this date. If you are integrating objects in the Solar System, you may want to measure time in Julian Day numbers (e.g. 1st Jan 2000 = 2451544.5)

- The time at which the integration will finish (in days).

- The output interval (in days). This determines how often the programme will store orbital elements for the bodies.

- The timestep used by the integrator (in days). For variable timestep algorithms (e.g. Bulirsch-Stoer and Radau), this is the stepsize used for the first timestep only. After that the programme will choose its own timestep. Note that if you choose a large initial timestep, the variable timestep algorithms may reduce it in order to maintain the desired accuracy.

- A integration accuracy parameter. This is approximately how much error per step the variable-timestep algorithms will tolerate. It is also used by the hybrid algorithm during close approaches. This number is ignored by the MVS algorithm but you should provide a number anyway.

The next lines in param.in should contain options that you will only want to change occasionally. If in doubt, you can use the same options as the sample param.in file. The options are:

- Should the integrator stop if a close encounter occurs (yes or no)?

- Should the programme check for collisions and take appropriate action if they occur (yes or no)? If you answer no, all the bodies will behave as point masses.

- Should collisions result in fragmentation (yes or no)? This version of MERCURY does not include fragmentation, so this is ignored at present. You should still specify yes or no however.

- How should the time be expressed (days or years)? This option controls how information messages produced by mercury6_1.for are formatted.

- Should time be measured with respect to the integration start time (yes or no)?
  If you choose 'years' and 'no' for the previous option and this one, the time will expressed as a Julian date before October

1582, and as a Gregorian date for later dates.

- What level of output precision (low, medium or high)? This determines how many significant figures will be used to store the orbital elements (roughly 4, 9 or 15).

- This line is no longer used. However, for backwards compatibility, you should still include a line here in param.in, although it doesn't matter what you put on this line.

- Include the effects of general relativity (yes or no)? This version of MERCURY does not include relativity, so this is ignored at present.

- Include the effects of the user-defined force routine (yes or no). You can add additional forces to the integrator in the subroutine mfo_user in mercury6_1.for

The remaining lines in param.in should contain some other parameters that you will only need to change rarely. These are:

- The distance from the central body at which objects are removed (in AU). These bodies are assumed to be so far from the central body that they are no longer important. Note that this number is used to scale the output (on a log scale), so don't make it bigger than you need to.

- The radius of the central body (in AU). Objects coming closer than this are assumed to collide with the central body. This number is also used to scale the output (on a log scale).

- The mass of the central body (in solar masses).

- The J2 moment of the central body in units of its radius^2.

- The J4 moment of the central body in units of its radius^4.

- The J6 moment of the central body in units of its radius^6.

- A line which is not used at present. Write whatever you like on this line.

- Another line which is not used at present.

- The changeover distance used by the hybrid integrator (in Hill radii). This is the minimum separation between objects before the hybrid (close encounter) part of the integrator takes effect.

- The maximum number of timesteps between data dumps. This also controls how often mercury6_1.for notifies you of its progress.

- The number of timesteps between other periodic effects. At present this controls how often mercury6_1.for checks for ejections and recomputes objects' Hill radii.

5) message.in

N.B. Alter the contents of this file at your peril!!
===

This file contains the text of various messages output by MERCURY,
together with an index number and the number of characters in the string
(including spaces used for alignment).

---

(4) C O N V E R T I N G D A T A T O O R B I T A L E L E M E N T S

After doing an integration you can see how the objects' orbits
varied over time. To do so, follow these steps:

a) Make sure the output files produced by the original integration still
   exist.

b) Make sure the compiled version of element6.for exists.

c) Make sure each of the input files described below exists, and alter
   them to suit your needs.

d) Run the compiled version of element6.for
   e.g. On Unix/Linux systems, use the command: ./element6

The programme will produce a set of new files, one per object, containing
orbital elements. Each file has the name of the object with the extension
.aei

The Input Files

1) element.in

This file contains parameters and options used by element6.for
Any lines beginning with the) character are assumed to be comments,
and will be ignored by element6.for

- In the first non-comment line, give the number N of compressed
  data files you want to read from. Usually this will be 1.
  (If you specify more than one file, element6.for will combine
  the orbital elements from all the integrations into one set
  of output files).

- The next N lines should contain the name(s) of the compressed
  data file(s) produced by mercury6_1.for Put each file name on a
  different line.

- At the end of the next line, indicate what origin you want to
  use for your elements. Choose between Central (for elements
  with respect to the central body), Barycentric (for barycentric
  elements) or Jacobi (for Jacobi elements).

- On the next line, specify the minimum time interval between the
  times at which you would like orbital elements. For example,
  if the original integration stored data every 100 days, but you
  are only interested in seeing orbital elements every 500 days,
  put 500 on this line in elements.in

- On the next line say whether you would like time expressed in days
  or years (write years or days).

- Then, on a new line, state whether you want to express the time with respect to the integration start date (write yes or no).

- Next comes a line indicating which orbital elements you want, and in what format. Each element is indicated by a code letter, followed by a number indicating the desired number of digits and decimal places. If the number of figures is followed by 'e', then the programme will use exponential notation for that element.

  The code letters are:
  a = semi-major axis (in AU)
  b = apocentre distance (in AU, b is short for Big q)
  d = density (g per cm^3)
  e = eccentricity
  f = true anomaly (degrees)
  g = argument of perihelion (degrees)
  i = inclination (degrees)
  l = mean anomaly (degrees)
  m = mass (solar masses)
  n = longitude of ascending node
  o = obliquity (degrees)
  p = longitude of perihelion (degrees)
  q = pericentre distance (AU)
  r = radial distance (AU)
  s = spin period (days)
  x, y or z = Cartesian coordinates x, y or z
  u, v or w = Cartesian velocities vx, vy or vz

  E.g. a8.4 e8.6 i7.3 g7.3 n7.3 l7.3 m13e
  indicates that you want the semi-major axis (8 digits including 4 decimal places), eccentricity (8 digits including 6 decimal places) etc.... and mass (13 digits in exponential notation).

  Note that if you choose to express an element using a large number of significant figures, the last few digits might not be meaningful if the output precision of the original integation was low or medium.

- The remaining lines in elements.in should contain the names of the objects for which you want orbital elements. If you don't supply any names, the programme assumes that you want elements for all the objects.

2) message.in

This is the same file as used by mercury6_1.for

---

(5) EXAMINING DATA ON CLOSE ENCOUNTERS

To examine details of close encounters that occurred during an integration, follow these steps:

a) Make sure the output files produced by the original integration still exist.

b) Make sure the compiled version of close6.for exists.

c) Make sure each of the input files described below exists, and alter
   them to suit your needs.

d) Run the compiled version of close6.for
   e.g. On Unix/Linux systems, use the command: ./close6

The programme will produce a set of new files, one per object, containing
details of close encounters with that object. Each file has the name of
the object with the extension .clo

The Input Files
----

1) close.in

This file contains parameters and options used by close6.for
Any lines beginning with the) character are assumed to be comments,
and will be ignored by close6.for. Don't delete the first comment
line though.

  o In the first non-comment line, give the number N of compressed
    data files you want to read from. Usually this will be 1.
    (If you specify more than one file, close6.for will combine
    close-encounter details from all the integrations into one set
    of output files).

  o The next N lines should contain the name(s) of the compressed
     data file(s) produced by mercury6_1.for Put each file name on a
     different line.

  o On the next line say whether you would like time expressed in days
    or years (write years or days in close.in).

  o Then, on a new line, state whether you want to express the time
    with respect to the integration start date (write yes or no).

 o The remaining lines in m_close.in should contain the names of the
   objects for which you want close-encounter details. If you don't
   supply any names, the programme assumes that you want details for
   all the objects.

2) message.in

The same file as used by mercury6_1.for

N.B. When using the hybrid symplectic algorithm, only those close
=== encounters that are integrated using the Bulirsch-Stoer part of
     the integrator will be saved. In practice, this means that some
     distant ''close'' encounters will not be recorded.

---

(6) C O N T I N U I N G A N I N T E G R A T I O N F R O M D U M P F I L E S

If your computer crashes while MERCURY is doing an integration, all is
not lost. You can continue the integration from the point at which

mercury6_1.for last saved data in dump files, rather than having to
redo the whole calculation. Just follow these steps:

a) Make sure all of the input, output and dump files used by the original
   integration are still present.

b) Make sure the filenames listed in files.in correspond to these files.

c) Run the compiled version of mercury6_1.for

If for some reason one of the dump files has become corrupted, look
to see if you still have a set of files with the extension .tmp
produced during the original integration (if you have subsequently
used mercury6_1.for to do another integration in the same directory,
you will have lost these unfortunately). These .tmp files are duplicate
copies of the dump files. Copy each one so that they form a set of
uncorrupted dump files (e.g. copy big.tmp to big.dmp etc.), and then
run the compiled version of mercury6_1.for

N.B. It is important that you replace all the dump files with the .tmp
=== files in this way, rather than just the file that is corrupted.

---

(7) E X T E N D I N G A P R E V I O U S I N T E G R A T I O N

If you want to extend an old integration that finished successfully
(i.e. not one that crashed), follow these steps:

a) Make sure all of the input, output and dump files used by the original
   integration are still present.

b) Make sure the filenames listed in files.in correspond to these files.

c) Change the finish time in the parameter dump file (see section 3)
   to the end point of the extended integration.

d) Run the compiled version of mercury6_1.for

---

(8) C H A N G E S F R O M M E R C U R Y 5

The code has been restructured, primarily to make it easier to
incorporate new coordinate systems (binary-star coordinates, and
some others that I'm tinkering with, that are not included in this
version). All the coordinate change routines now accept the same
input and output (except going to/from Keplerian elements).
A major result of this is that the arrays for objects are now
indexed starting at 2 instead of 1 (the central body uses index 1).

The compressed output has changed again (!! sorry about that).
However, the new version is an improvement for a few reasons.
Firstly, it makes it possible to choose your coordinate origin
(central/barycentric/Jacobi) when you run the decompression
programme rather than when you do the original integration.
Secondly, it handles hyperbolic elements better - no more problems
with the hyperbolic mean anomaly. In addition, you no longer have

to specify emax or qmin in the param.in file - the new compressed format can handle all osculating values of q and e. Finally, it is now possible to produce .aei files when the number of objects in the integration exceeds the number of files that can be open at the same time for your operating system.

Rather than remodifying m_elem.for to cope with yet another type of compressed output, I have written a new programme called elements (which is also easier to pronounce). Use this for compressed output generated by Mercury6, and use m_elem5 for the earlier versions of Mercury.

A bug fix: Mercury5_2, and earlier, had a bug which occurred when integrating test particles using the MVS algorithm. This incorrectly modelled the perturbations due to the innermost massive body.

The central-body oblateness terms J2, J4 and J6 now function. These were also included in a few older versions of Mercury, but were not properly tested until now.

The MVS algorithm now incorporates simple symplectic correctors which should improve the accuracy using this algorithm. The correctors remove error terms proportional to eh^3 thru eh^6, where h is the timestep and e is the ratio of the object masses to the central mass. Hence, the leading error terms are proportional to e^2h^3 and eh^7.

Close encounter details are now output in batches, rather than after every encounter. Furthermore, the programme does not always do regular data dumps in addition to data dumps at every output interval. These changes are made in the hope that they will reduce delays caused by accessing the hard disk.