LOFTI: Orbit Fitting of Wide Stellar Binaries with Gaia

The LOFTI-Gaia package will fit orbital elements to the astrometry provided by Gaia DR2 only.

Written by Logan A. Pearce, 2019
If you use LOFTI in your work please cite Pearce et al. 2019

Fitting orbital parameters:

Begin by importing the "fitorbit" module

In [1]:
```
from lofti_gaia.lofti import fitorbit
```

Let's look at the arguements and what it writes out

In [2]:
```
help(fitorbit)
```

```
Help on function fitorbit in module lofti_gaia.lofti:

fitorbit(source_id1, source_id2, mass1=0, mass2=0, d=2015.5, verbose=Fals
e, output_directory='.', rank=0, accept_min=10000)
    Fit orbital parameters to binary stars using only the RA/DEC position
s and proper motions from
    Gaia DR2 by inputting the source ids of the two objects and their mas
ses only.
    Writes accepted orbital parameters to a file.

    Parameters:
    -----------
    source_id1, source_id2 : int
        Gaia DR2 source identifiers, found in the Gaia archive or Simbad.
Fit will be
        of source_id2 relative to source_id1.
    mass1, mass2 : tuple, flt [Msol]
        masses of primary and secondary objects, entered as a tuple with
the error.  For example:
        mass1 = (1.0,0.2) is a 1 solar mass star with error of \pm 0.2 so
lar masses.  If mass1 or mass2 = 0,
        script will prompt user to input a tuple mass.  Default = 0.
    d : flt [decimalyear]
        observation date.  Default = 2015.5, the DR2 obs date.
    verbose : bool
        if set to True, script will print constraints to screen, ask for
confrimation before proceeding,
        and print regular updates on number of accepted orbits.  If set t
o False, script will print a progress bar
        to screen.  Default = False.
    output_name : str
        directory to write output files to.  If verbose = True, script wi
ll prompt for directory, if
        verbose = False it will write files to current directly unless th
e name argument is specified.
    rank : int
        if running in parallel processing mode, set this keyword to the r
ank of each process.  Else it is NA.
    accept_min : int
        when the number of accepted orbits reaches this number, script wi
ll terminate

    Returns:
    --------
    output files :
        writes out accepted orbits to a file called name+rank+'_accepte
d'.  The columns of the file are:
            semi-major axis [arcsec]
            period [yrs]
            epoch of periastron passage [decimalyear]
            eccentricity
            inclination [deg]
            argument of periastron [deg]
            position angle of nodes [deg]
            chi-squared value of the orbit
```

```
                probability of orbit generating observations
                random uniform number to determine acceptance
            writes out a human-readable text file of the constraints it compu
ted from Gaia data, called "constraints.txt".
                deltaRA [mas]: relative RA separation
                deltaDec [mas]: relative DEC separation
                pmRA_kms [km/s]: relative proper motion in RA
                pmDec_kms [km/s]: relative proper motion in DEC
                deltarv [km/s]: relative radial velocity (if applicable)
                total_pos_velocity [mas/yr]: total velocity vector in the pla
ne of the sky
                total_velocity_kms [km.s]: total velocity vector in the plane
of the sky
                rho [mas]: separation
                pa [deg]: position angle
                delta_mag [mag]: contrast in magnitudes
                d_star [pc]: distance
            writes out the above parameters to a machine readable file called
"constraints.pkl"

    Notes:
    ------
    Future versions will adapt to new Gaia data releases and additional c
onstraints.  See
    Pearce et al. 2019 for more information, including a discussion of ho
w to determine if
    the Gaia DR2 solution is of adequate quality to provide meaningful an
d accurate constraints
    for orbit fitting.
    If you use this package, please cite Pearce et al. 2019.

    Written by Logan A. Pearce, 2019
```

Example: DS Tuc AB:

The first use of this technique was for DS Tuc AB, and published in Newton et al. 2019. Both components have well-defined solutions in Gaia DR2, including radial velocities. It makes a good demostration case.
Let's start by making a new directory to hold the output file

In [3]:
```
os.system('mkdir DSTucAB')
```

Out[3]: 256

All we need to give the fitter is the Gaia source ID numbers for the two components:

In [4]:
```
DSTucA = 6387058411482257536
DSTucB = 6387058411482257280
```

and their masses (masses are from Newton et al. 2019). fitorbit looks for the mass and its error to be entered as a tuple:

In [5]:
```
massA = (0.97,0.04)
massB = (0.87, 0.04)
```

Run the fitter by calling fitorbit. Let's tell it to output files to the directory we made, set a low minimum accepted orbit number for demonstration purposes, and set verbose to True. When verbose is set to True, the fitter pauses and asks you to check that the constraints it will use look reasonable and like you expect them to, and makes sure it will write out the file where you are expecting to find it. It will also print an update when it finds lower chi-squared values, and periodically prints the number of orbits it's found

In [6]:
```
fitorbit(DSTucA, DSTucB,
         mass1 = massA,
         mass2 = massB,
         output_directory = "DSTucAB",
         verbose = True,
         accept_min = 50
        )
```

```
Computing constraints.
Created TAP+ (v1.0.1) - Connection:
        Host: gea.esac.esa.int
        Use HTTPS: False
        Port: 80
        SSL Port: 443
Finished computing constraints:
Delta RA, err in mas: -1146.6531115571179 0.015914032442077848
Delta Dec, err in mas: 5240.634169821117 0.03132321306860027

pmRA, err in km/s: -0.30173712008430653 0.02059592026680047
pmDec, err in km/s: 0.3543703626789322 0.012171205850378666
deltaRV, err im km/s (pos towards observer): 1.8793611665844168 0.7222733
249905701

Total relative velocity [km/s]: 1.9361358521672773 +/- 1.8795134276320105
Total plane-of-sky relative velocity [mas/yr]: 2.2246301214145014 +/- 0.1
1376315413334326

sep,err [mas] 5364.611808922288 0.030782922107068112 pa,err [deg]: 347.65
815421242434 0.00018087678592866222
sep [AU] 236.7619843430118
sep, err [km] (35419088720.4213, 0.0) (203239.87792941494, 0.0)
D_star 44.1340385429632 +\- 0.06336868730526682
Delta Gmag -1.0800505

Does this look good? Hit enter to start the fit, n to exit
Yeehaw let's go
Chi-min: 11.56334261598355
Ok, starting loop
I will write files out to this directory: DSTucAB
Is that right? Hit enter to proceed, n for no:
I will be looking for  50  orbits.
Ok? Hit enter to proceed, n for no:
Found new chi min:  3.298175019792497
Loop count rank  0 :  0
Rank  0  has found  12 accepted orbits
Found new chi min:  1.8097436072769046
Found new chi min:  0.9433485180067218
Loop count rank  0 :  10
Rank  0  has found  11 accepted orbits
Loop count rank  0 :  20
Rank  0  has found  22 accepted orbits
Found new chi min:  0.7043729230309588
Loop count rank  0 :  30
Rank  0  has found  29 accepted orbits
Loop count rank  0 :  40
Rank  0  has found  34 accepted orbits
Loop count rank  0 :  50
```

```
Rank  0  has found  40 accepted orbits
Loop count rank  0 :  60
Rank  0  has found  46 accepted orbits
Found new chi min:  0.15922388514717503
Loop count rank  0 :  70
Rank  0  has found  42 accepted orbits
Loop count rank  0 :  80
Rank  0  has found  44 accepted orbits
Loop count rank  0 :  90
Rank  0  has found  51 accepted orbits

Found  51  orbits, finishing up...
This operation took 49.41786503791809 seconds
and 0.013727184732755025 hours
```

With verbose = False, the print statements are supressed, the script does not pause to check with the user before proceeding, and a progress bar reports the status of the fit.

In [7]:
```
from lofti_gaia.lofti import fitorbit

DSTucA = 6387058411482257536
DSTucB = 6387058411482257280
massA = (0.97,0.04)
massB = (0.87, 0.04)

fitorbit(DSTucA, DSTucB,
         mass1 = massA,
         mass2 = massB,
         output_directory = "DSTucAB",
         accept_min = 100
        )
```

```
Computing constraints.
Ok, starting loop

 88% (88 of 100) |####################   | Elapsed Time: 0:00:33 ETA:
0:00:07


Found  101  orbits, finishing up...
This operation took 39.62535095214844 seconds
and 0.011007041931152343 hours
```

If you forget to enter the masses, the script will prompt you to enter the mass and error with a space between them.

In [8]:
```
fitorbit(DSTucA, DSTucB,
         output_directory = "DSTucAB",
         accept_min = 100
        )
```

```
Computing constraints.
Enter mass of object 1 and error separated by a space (ex: 1.02 0.2):0.97
0.04
Enter mass of object 2 and error separated by a space (ex: 1.02 0.2):0.87
0.04
Ok, starting loop
 79% (79 of 100) |#################      | Elapsed Time: 0:00:05 ETA:
0:00:01


Found  158  orbits, finishing up...
This operation took 11.220418930053711 seconds
and 0.0031167830361260307 hours
```

Plotting the output

lofti_gaia offers and optional setting of plotting tools to inspect the results of fitorbit, called lofti_gaia.lofti.makeplots.

In [9]:
```
from lofti_gaia.lofti import makeplots
help(makeplots)
```

```
Help on function makeplots in module lofti_gaia.lofti:

makeplots(input_directory, rank=0, Collect_into_one_file=False, limit=0.
0, roll_w=False, plot_posteriors=True, plot_orbit_plane=True, plot_3d=Tru
e, axlim=6)
    Produce plots and summary statistics for the output from lofti.fitorb
it.

    Parameters:
    -----------
    input_directory : str
        Gaia DR2 source identifiers, found in the Gaia archive or Simbad.
Fit will be
        of source_id2 relative to source_id1.
    rank : int
        Set this parameter to iterate through processes if running on mul
tiple threads
    Collect_into_one_file : bool
        Set to true if running on multiple process and the script did not
terminate on its own.  This will
        tell the script to collect output from each multiple process and
put into one file.
    limit : int [au]
        Sometimes semi-major axis posteriors will have very long tails.
If you wish to truncate the sma histogram
        at some value for clarity, set the limit parameter to that value.
    roll_w : bool
        I you wish to have arg of periastron wrap around zero, set this t
o True
    plot_posteriors : bool
        set to True to make posterior histogram plots of X, Y, Z, dotX, d
otY, dotZ, ddotX, ddotY, ddotZ
    plot_orbit_plane : bool
        set to True to generate plot of 100 random orbits from the poster
ior in XY plane, XZ plane, and YZ plane
    plot_3d: bool
        set to True to generare a 3D plot of 20 random orbits from the po
sterior.
    axlim: flt [arcsec]
        if plot_orbits = True or plot_3d = True, set this parameter to se
t the axis limits (in arcsec) for the plots

    Returns:
    --------
    output files :
        stats: summary statistics for each orbital parameter + periastron
distance, including:
            Mean    Std    Mode    68% Min Cred Int    95% Min Cred Int
        hist.pdf: 1d histogram of orbital parameter posteriors
        observable_posteriors (if plot_posteriors = True): directory cont
aining 1d histograms of
            posteriors for X, Y, Z, dotX, dotY, dotZ, ddotX, ddotY, ddotZ
        orbits.png (if plot_orbits = True): plot of a selection of 100 ra
ndom orbits from fitorbit posterior in
```

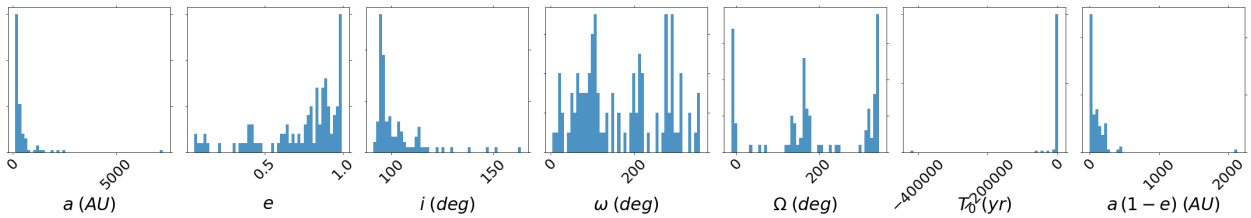
```
            RA/DEC, colored by orbital phase
        orbits_yz.png, orbits_xz.png (if plot_orbits = True): plots of th
e same 100 orbits in YZ and XZ planes
        orbits_3d.png (if plot_3d = True): 3d plot of 20 random orbits fr
om posterior

    Notes:
    ------
    These are suggested summary stats and plots.  For more versatility yo
u can use
    the fitorbit output with your own plotting scheme.
    If you use this package, please cite Pearce et al. 2019.

    Written by Logan A. Pearce, 2019
```

In [10]:
```
directory = 'DSTucAB'
makeplots(directory,
                  rank = 0,
                  Collect_into_one_file = False,
                  limit = 0.,
                  roll_w = False,
                  plot_posteriors = True,
                  plot_orbit_plane = True,
                  plot_3d = True
              )
```

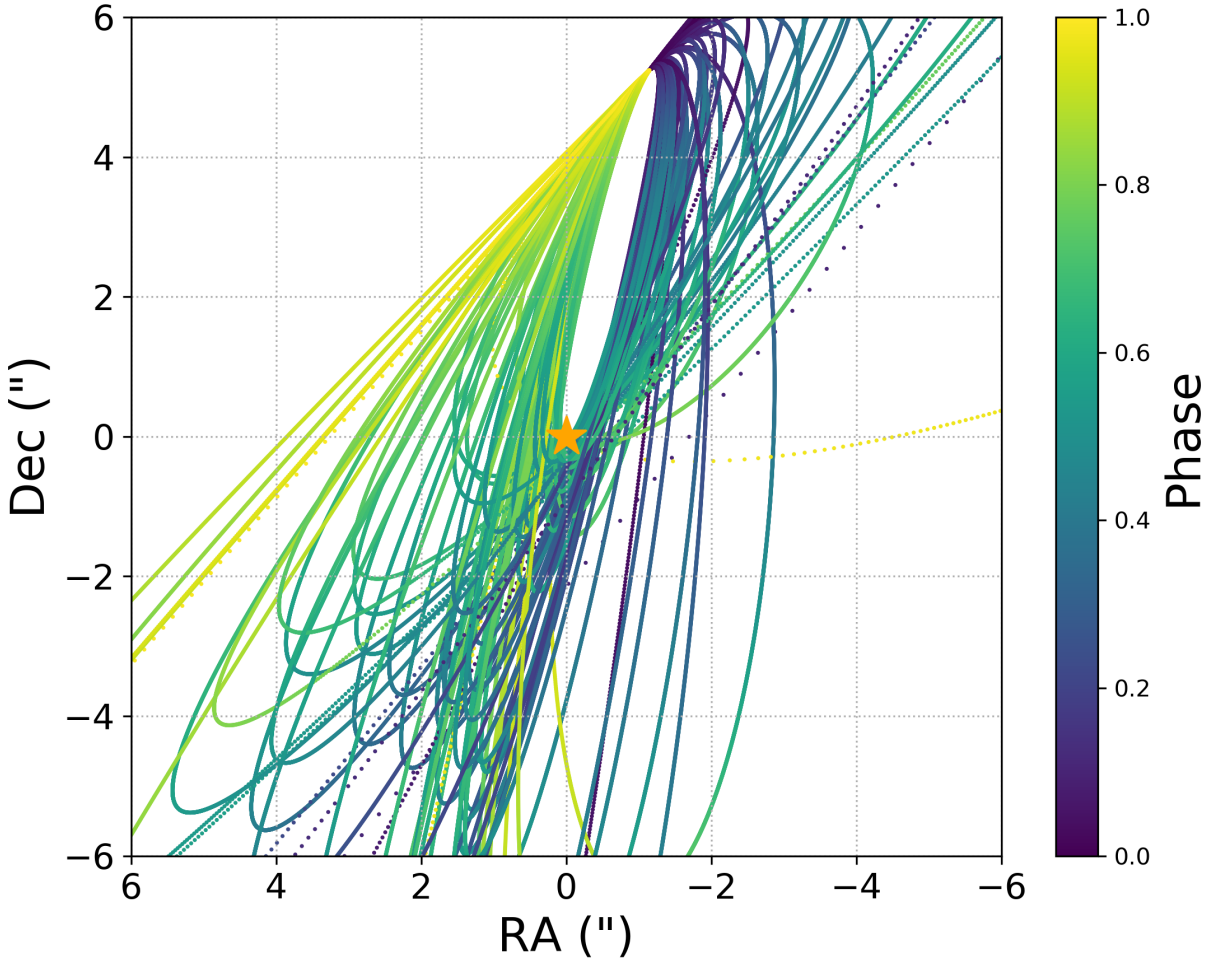
```
Writing out stats
Making histograms
Plotting observable posteriors
Plotting orbits
XY plane
XZ plane
YZ plane
3D
```

As with fitorbit, the output of this script is saved into the directory you set at the beginning. Figures are saved as pngs. Let's look at the figures this code generated:

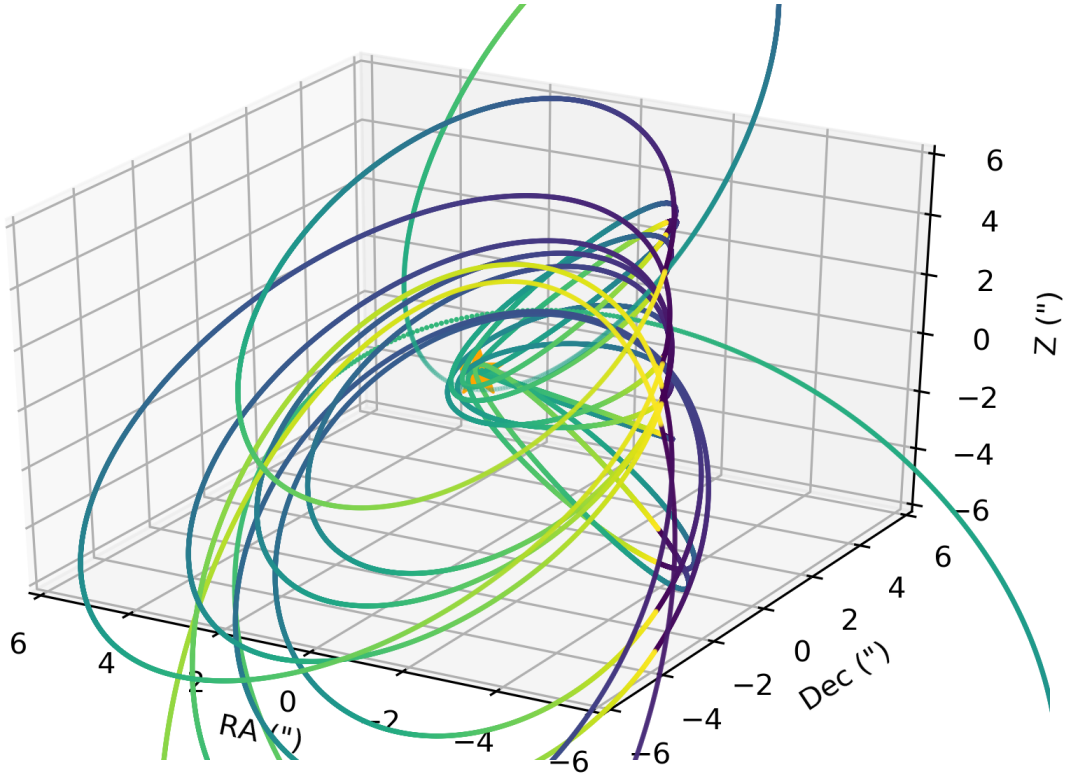
hists.png:



orbits.png

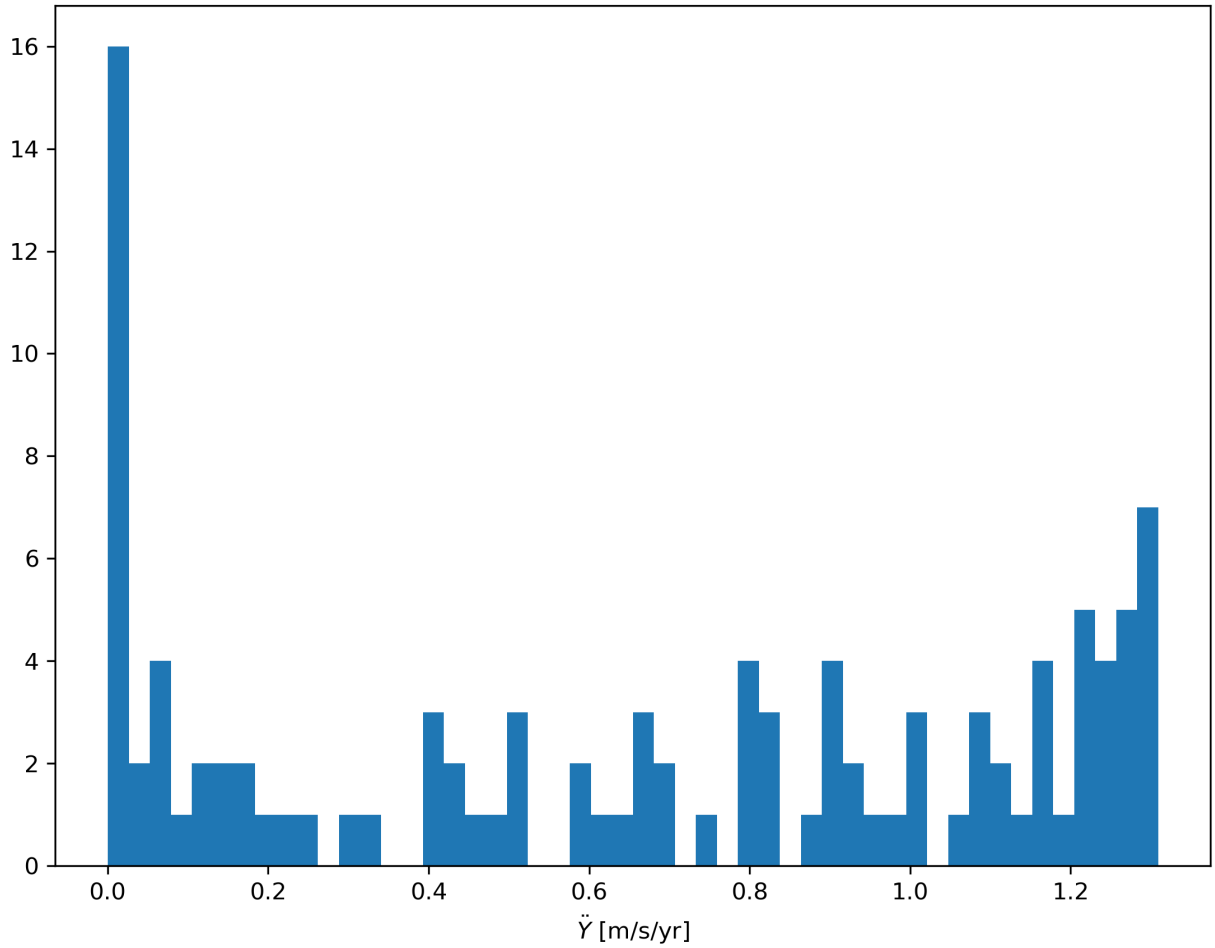


orbits_3d.png



One of the observables posteriors:

This is the distribution of acceleration in RA from the orbits in the posterior sample:



stats.txt:

The stats file looks like this inside (the parameters are already written in Latex math mode):

```
Parameter    Mean    Std    Mode    68% Min Cred Int    95% Min Cred Int
$a \; (AU)$    440.273860033696    778.7828994666231    167.91369902260953
   (120.78742643128001, 335.5722381365086)    (120.78742643128001,
1440.6607692140242)
$e$    0.7363533729151298    0.25417521870515686    0.9354707541002492
(0.7079602799311652, 0.9944789694374682)    (0.11403236837364339,
0.9944789694374682)
$ i \; (deg)$    102.21552571902514    12.315962182778556
96.93037897540756    (93.85757612975858, 104.21407925155971)
(91.20949059969848, 129.6667081849308)
$ \omega \; (deg)$    170.14767463219783    99.64182945955947
110.92869998371395    (13.407759062840405, 220.04147746284139)
(5.060697734275732, 336.04000373968086)
$\Omega \; (deg)$    37.72898415688386    94.40123009172831
-29.405473241926238    (-45.72234936546789, 143.51214034948345)
(-119.84686434942489, 179.66687380258452)
$T_0 \; (yr)$    -5266.072718030055    41405.54729054067
1390.9519132615283    (-387.5345674226005, 1637.8029937252832)
(-13979.8862767336, 1939.049329542796)
```

```
$a\,(1-e) \; (AU)$    109.41920890604749    219.0675705854163
26.869332294301532    (0.9391792185192334, 117.30940253868981)
(0.9391792185192334, 263.2353692706762)
```

In [ ]: