Determining the Orbit of Eros

Robert Arlt Jr.

12.410

November 18, 2009

Abstract

In order to ascertain if George R. Wallace, Astrophysical Observatory (WAO) is properly equipped for observation and calculation of Near-Earth Asteroid (NEA) orbits, three observations of 433 Eros were made and Eros's orbit calculated using Gauss's method of orbit determination. The results show that WAO is properly equipped to calculate orbits of NEAs with high brightness such as Eros to an accuracy of approximately two-percent. The results also indicate that Gauss's method would not be appropriate for use in further study. A more generalized orbit-fitting algorithm, such as that employed by California Institute of Technology's Jet Propulsion Laboratory (one that can provide error analysis and incorporate more than three observations) would be more appropriate for future use.

Introduction

Near-Earth Objects (NEOs) are comets and asteroids that have entered into orbits that bring them close to the earth. Comets were initially composed primarily of water ice that may have entirely dissipated or may still be frozen around either dust particles; formed billions of years ago at the same time as the solar system, offer a look into the creation of the solar system and the composition of the primordial mixture from which the planets formed.

Some NEOs are interesting for other reasons – Near-Earth Asteroids (NEAs) that are over one kilometer in diameter, for example, pose a potential risk for severe catastrophic events should one hit the Earth. Because of this, NASA is currently under a U.S. Congressional mandate to identify 90% of all NEAs of this size. After an NEA is identified, its orbit must be calculated in order to assess the possibility of it hitting the Earth.

Knowing accurate orbits of NEOs is also useful for further study of the objects. Accurate positional information allows the shape, size, and composition of NEOs to be determined with further observation.

The purpose of this project was to explore the process of NEA identification and orbit calculation. Unfortunately, the resources needed to identify NEAs, a process that is best accomplished with a large-scale sky-survey project, were unavailable. Therefore, the project would explore only the second half of NASA's mission — calculating the orbit of an NEA. The project also provided information on the feasibility of calculating the orbits of asteroids closer to the Earth than main belt asteroids and with higher eccentricities using data taken with the relatively small fourteen-inch diameter Schmidt-Cassegrain telescopes at George R. Wallace, Astrophysical Observatory (WAO).

433 Eros, an Amor-family asteroid that crosses the orbital path of Mars, was chosen for this project because of its rise and set times, its orbital period, and its relatively bright apparent magnitude (a range of ten to 12 over the duration of this project) among available asteroids, which resulted in good signal-to-noise ratios. Additionally, the orbit of Eros is known, providing a chance to assess the success of this project's techniques and to quantize the accuracy and degree of precision of the calculation.

Gauss's method was used to calculate the orbit because it requires only three positional observations to be made. The low amount of required data was especially desirable because of the limited availability of observation time due to the scheduling of telescopes and the weather patterns at WAO.

Gauss's method requires that approximately five percent of the object's period be observed in order to obtain a proper fit of the orbit. Thus, for Eros's period of approximately 650 days, the three observations had to be spread out across the time period of a month.

Observation

The three required observations took place on September 21, 2009; October 12, 2009; and October 26, 2009. The same fourteen-inch f/11 Celestron C14 Schmidt-Cassegrain telescope was used each night. The telescope was equipped with a SBIG STL-1001E imaging CCD camera with clear, B, V, R, I, and VR filters, and an Optec TCF-S3 autofocusser. The telescope sat upon a Software Bisque Paramount ME robotic German equatorial mount that was driven by Software Bisque's *TheSky6* installed on an Apple MacMini running Microsoft *Windows XP* operating system. The camera was controlled by Software Bisque's *CCDSoft* installed on the same MacMini.

TheSky6 utilized a 300 star ProTrack *TPoint* pointing model with periodic-error correction resulting in pointing that was accurate to approximately 29 arcseconds and tracking with a maximum exposure time of two to four minutes depending on the position of the object.

At the beginning of the night, the computer was turned on, the time synched to Haystack Observatory's time server, and the telescope turned on and homed. The camera was cooled to a temperature of -20 °C (excepting October 26 when the telescope was previously used for another project and the temperature was set at -15 °C).

Once the temperature of the camera stabilized, the telescope was focused for use with the R filter. Then calibration frames were taken. 40 bias images and 20 dark images of 30-second exposure times were taken each night.

After the calibration frames were taken, the coarse position of Eros was loaded through *TheSky6* from Harvard Minor Planet Center and the telescope pointed at the coordinates. A test frame was taken and the position of the asteroid ascertained by comparing the frame to a DSS-finder chart produced with the help of *koronisfamily.org*. Once the frame was verified, a suitable star was chosen for centering and telescope tracking.

A suitable star was one which placed the position of the asteroid slightly to the upper left of center in the image when viewed within *CCDSoft*. This position meant that Eros would drift toward the center of the frame as the night progressed. A sidereal-tracking rate was chosen for the project because it increased the accuracy of the plate solution as opposed to tracking Eros itself.

With the telescope pointed at the correct field, data acquisition began. An exposure time of 30 seconds through the R filter was determined to give the best signal-to-noise ratio of Eros while at the same time ensuring little drift of the object due to its difference in motion speed from sidereal rate. The exposure time also prevented the stars themselves from streaking from the imperfection of the mount's tracking ability.

At least 20 frames of the object were taken each night to ensure that at least one scientifically usable frame was available. Fortunately, the weather was not an issue on any of the observation nights and each night produced a host of suitable images.

Data Reduction

Calibration

The National Optical Astronomy Observatories' *Image Reduction and Analysis Facility* (*IRAF*) was used for the calibration of the images. The particular distribution of *IRAF* used was part of the *cygwin* installation of *Scisoft* (this package was developed by Robert Arlt Jr. and was available at *http://dukrat.net/public/scisoft at the time of writing*). The exact settings used for each package are available in Appendix A.

The first step in the calibration process was the creation of a bad-pixel mask. From previous work on bad-pixel masks at WAO, the best masks were produced from a set of long-exposure darks requiring an hour at the minimum. It is also known that the SBIG STL-1001E cameras at WAO can develop new bad pixels over time periods as short as 38 days (*http://occult.mit.edu/people/studentReports2009/robert/robert.htm*). However, observation time constraints dictated that a new bad-pixel mask could not be made for the project. Instead, a bad-pixel mask generated from data taken at WAO on the same equipment on August 04, 2009 was used. Figure 1 shows the bad-pixel

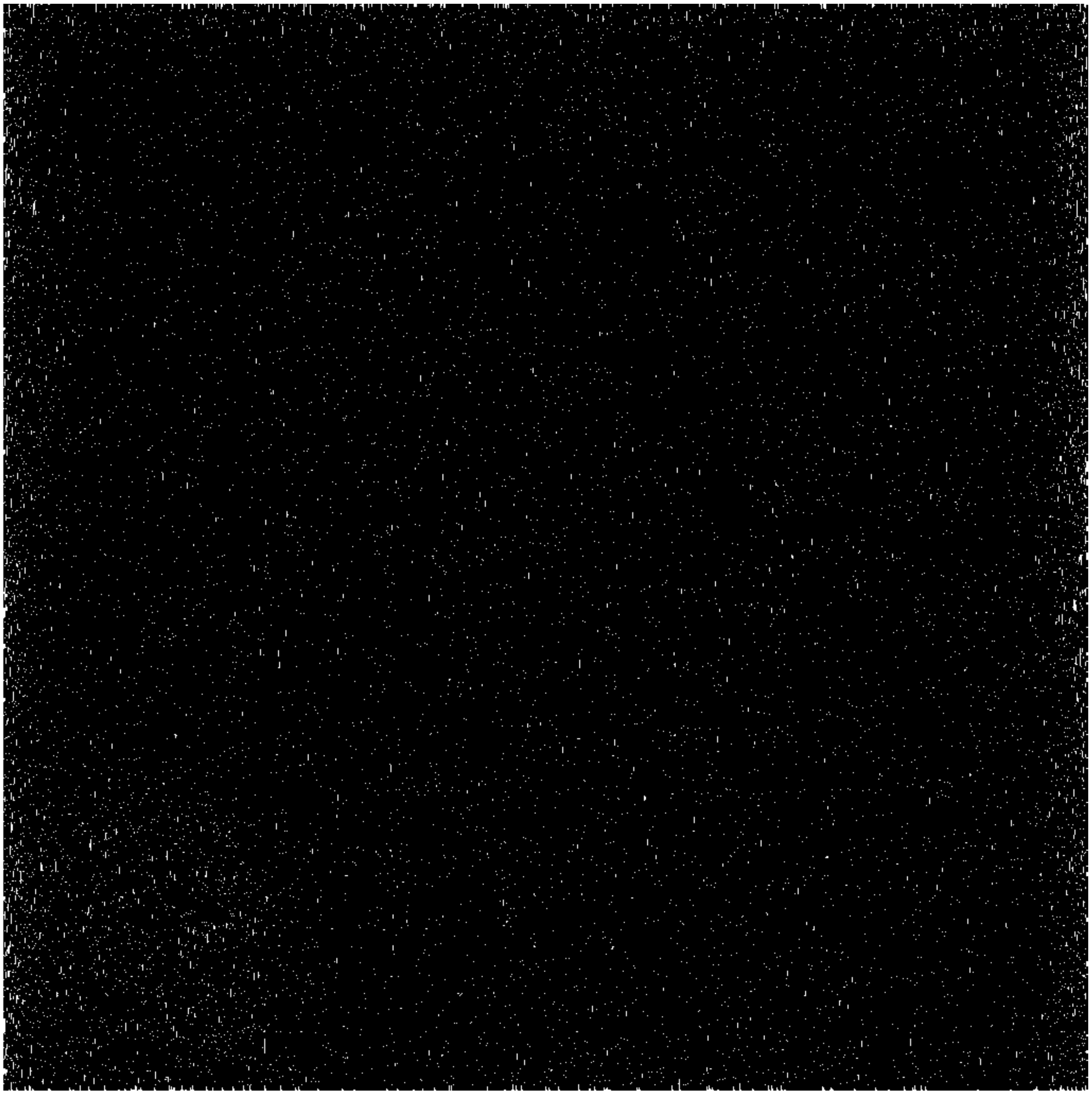

Fig. 1: The bad pixel mask used for the project.
Notice the uniformity of the bad pixels across most of the image with an increase near the edges. Each bad pixel presents the opportunity to negatively affect the centroiding process used to find the logical position of stars. Identifying each bad pixel and interpolating a new value for it from surrounding data points increases the accuracy of a centroid-produced position.

mask that was used.

The bad-pixel mask was generated from two flat fields, one with a large number of background counts (~35000) and one with a low number of background counts (~3500), and 11 darks of five-minute exposure time.

A master-bias image was created using the *IRAF* package *zerocombine*. The darks and flats were then bias corrected using the *IRAF* package *ccdproc*. The low-count flat was divided by the high-count flat using the *IRAF* package *imarith*. Then the first part of the bad-pixel mask was created with the *IRAF* package *ccdmask*. The second part of the bad-pixel mask was created by combining the darks with the *IRAF* package *imcombine* and then producing the mask with the *IRAF* package *ccdmask*. The two parts were then combined to make the final bad-pixel mask by converting the two parts to *fits* files with the *IRAF* package *imcopy* and then combining the two parts with the *IRAF* package *imcombine*. Finally, the bad-pixel mask was converted to a *pl* file with the *IRAF* package *imcopy*.

The second step in calibrating the images was creating the master-bias and master-dark images for each night. The bias images were bad-pixel-mask corrected and combined to make the master-bias image with the *IRAF* package *zerocombine*. The darks were bad-pixel-mask corrected, bias corrected, and combined to make the master dark using the *IRAF* package *darkcombine*.

Observation time constraints also dictated that flats could not be taken at the time of observing. Instead, a single set of flats from October 14, 2009 was used. The flats were bad-pixel corrected, bias corrected, and combined to make the master flat used for all nights; the *IRAF* package *flatcombine* was used.

The master dark images were then examined using the *IRAF* package *imexam*. The darks had low counts of mean approximately four and standard deviations of approximately four as well. This meant that dark-correcting the images would do little to actually correct the images and would only add noise. Therefore, it was decided that the images would not be dark-corrected. The science images were bad-pixel-mask corrected, nightly master-bias corrected, and master-flat corrected using the IRAF package ccdproc.

Astrometry

The next step of the data reduction process involved calculating the right ascension (RA) and declination (Dec) of Eros in each image. The selection process for the three images used included ensuring that Eros was not overlapping another object and that all the objects had crisp discs. Ten bright isolated stars were chosen in each frame. Then, the logical (on CCD chip) centroid coordinates of the ten stars and Eros were determined with the *IRAF* package *autoast* (the *autoast* package is developed by Eran Ofek and was available from *http://wise-obs.tau.ac.il/~eran/iraf/astrometry.html* at time of writing). The RA and Dec of the ten stars were then obtained from the USNO-A2.0 catalog of astrometric standard stars using *TheSky6* as an interface to the database.

From the RA, Dec, and logical coordinates of the ten stars, a plate solution was obtained. This was accomplished with javascript (available from *http://www.phys.vt.edu/~jhs/SIP/astrome-*

trycalc.html at the time of writing) based on a *BASIC* program written by Jordan D. Marche that calculated the plate constants of the transformation equations

$$u = x + ax + by + c, \qquad \text{(Eq. 1)}$$

$$v = y + dx + ey + f, \qquad \text{(Eq. 2)}$$

where x,y gives the logical centroid position of an object and a, b, c, d, e, and f are the plate constants. As u and v are related to RA and Dec by

$$RA = \mathrm{atan}(u / b), \qquad \text{(Eq. 3)}$$

$$Dec = \mathrm{atan}((\sin(d_O) + v\cos(d_O))(u^2 + v^2)^{-1/2}), \qquad \text{(Eq. 4)}$$

where d_O is a reference declination chosen to be approximately in the center of the image, the plate constants can be used to calculate RA and Dec coordinates from logical coordinates and vice-versa. Table 1, Table 2, and Table 3 summarize the inputs and outputs for each image used for the orbit calculation. The errors on the RA and Dec of Eros are derived from the root-mean-square residuals of the least-mean-square fitting process used to calculate the plate constants.

Orbit Determination

The heliocentric ecliptic position and velocity vectors were then calculated from the three sets of RA and Dec coordinates of Eros at known times observed from a known location. This was accomplished within *Matlab* with a modified version of Björn J. R. Davidsson's "Computer codes for Computer Exercise #1" (available from *http://www.astro.uu.se/~bjorn/celmechlab01codes.*

Table 1: Plate solution summary for September 21, 2009.

Input:				
UT Time (HH MM SS.SSS)			LST (HH MM SS.SSS)	
03 04 33.562			22 19 09.779	
Source	Column (x,px)	Row (y, px)	RA (HH MM SS.SS)	Dec (±DD MM SS.SS)
Center			21 47 54.21	+05 20 39.34
Star 1	301.903	154.508	21 48 10.50	+05 27 34.97
Star 2	347.785	314.125	21 48 06.28	+05 24 25.23
Star 3	384.771	467.93	21 48 02.87	+05 21 22.44
Star 4	87.007	924.966	21 48 25.40	+05 12 01.08
Star 5	512.812	521.693	21 47 52.46	+05 20 24.98
Star 6	810.705	658.526	21 47 28.09	+05 17 54.45
Star 7	888.923	688.273	21 47 21.65	+05 17 21.15
Star 8	928.028	170.553	21 47 20.14	+05 27 44.74
Star 9	433.218	967.991	21 47 57.44	+05 11 25.46
Star 10	184.553	655.298	21 48 18.27	+05 17 29.31
Eros	587.477	548.741		
Output:				
Eros			21 47 46.33±0.05	+05 19 55.36±0.50

Plate constant	value
a	-1.000
b	-2.226E-7
c	0.003
d	2.234E-7
e	-1.000
f	0.003

Table 2: Plate solution summary for October 12, 2009.

Input:				
UT Time (HH MM SS.SSS)			LST (HH MM SS.SSS)	
01 50 09.784			22 27 19.419	

Source	Column (x,px)	Row (y, px)	RA (HH MM SS.SS)	Dec (±DD MM SS.SS)
Center			21 31 19.86	+03 49 45.60
Star 1	959.985	870.299	21 30 39.51	+03 43 00.23
Star 2	540.263	816.305	21 31 13.32	+03 43 46.12
Star 3	227.238	290.124	21 31 40.03	+03 54 01.91
Star 4	777.374	566.559	21 30 55.10	+03 48 56.21
Star 5	863.461	346.311	21 30 48.91	+03 53 24.56
Star 6	057.506	871.774	21 31 51.83	+03 42 17.21
Star 7	146.636	668.091	21 31 45.28	+03 46 25.29
Star 8	126.640	201.751	21 31 48.35	+03 55 43.34
Star 9	617.051	138.251	21 31 09.32	+03 57 22.10
Star 10	264.035	153.505	21 31 37.55	+03 56 46.81
Eros	187.908	270.872		
Output:				
Eros			21 31 43.26±0.02	+03 54 23.19±0.34

Plate constant	value
a	-1.000
b	-2.289E-7
c	0.003
d	2.248E-7
e	-1.000
f	0.003

Table 3: Plate solution summary for October 26, 2009.

Input:				
UT Time (HH MM SS.SSS)			LST (HH MM SS.SSS)	
02 04 59.342			23 37 23.191	

Source	Column (x,px)	Row (y, px)	RA (HH MM SS.SS)	Dec (±DD MM SS.SS)
Center			21 34 38.47	+03 19 56.10
Star 1	866.439	842.763	21 33 53.78	+03 15 59.19
Star 2	607.313	812.852	21 34 14.92	+03 16 25.85
Star 3	542.427	872.081	21 34 20.00	+03 15 11.90
Star 4	518.367	960.990	21 34 21.58	+03 13 24.25
Star 5	960.886	497.609	21 33 47.63	+03 22 58.56
Star 6	676.007	600.547	21 34 10.16	+03 20 42.00
Star 7	115.146	543.006	21 34 55.17	+03 21 27.00
Star 8	793.910	388.225	21 34 01.39	+03 25 02.00
Star 9	138.228	328.822	21 34 54.10	+03 25 43.68
Star 10	183.535	192.275	21 34 50.80	+03 28 29.57
Eros	573.779	506.916		
Output:				
Eros			21 34 18.58±0.12	+03 22 30.00±0.73

Plate constant	value
a	-1.000
b	-2.379E-7
c	0.002
d	2.115E-7
e	-1.000
f	0.004

html at time of writing). The script (see Appendix B) utilized multiple iterations of Gauss's method of orbit determination in order to converge upon a least-mean-square fit of the state vectors.

The state vectors were then used to calculate the classical orbital elements assuming a simple two-body system. This calculation was performed by the *Java* program "Kepler Orbit" written by Dieter Egger (available at *http://math-ed.com/Resources/GIS/Geometry_In_Space/java1/Temp/TLOrbit.html* at time of writing).

Table 4: Classical orbital elements of Eros.

Input:

Julian date			
2455116.572			

Heliocentric ecliptic position vector (AU)		
1.572	-0.091	0.238

Heliocentric ecliptic velocity vector (AU/Day)		
-0.002	0.013	.001

Output:

Orbital element	Value	Error*	% error*
Semi-major axis (AU)	1.479	0.021	1.40%
Eccentricity	0.231	0.008	3.35%
Inclination (deg)	10.713	-0.116	-1.09%
Argument of perihelion (deg)	175.062	-3.697	-2.16%
Ascending node longitude (deg)	303.693	-0.678	-0.22%
Mean anomaly (deg)	263.020	6.324	2.46%
Revolution period (days)	656.890	13.786	2.06%

*Errors are absolute differences from JPL's orbital elements available from http://ssd.jpl.nasa.gov/sbdb.cgi?sstr=433;orb=1 at time of writing.

Results

Table 4 shows the results of the orbit calculation. The results show that the orbit calculation was successful. However, the error offsets of the calculated orbit from the actual orbit of Eros as calculated

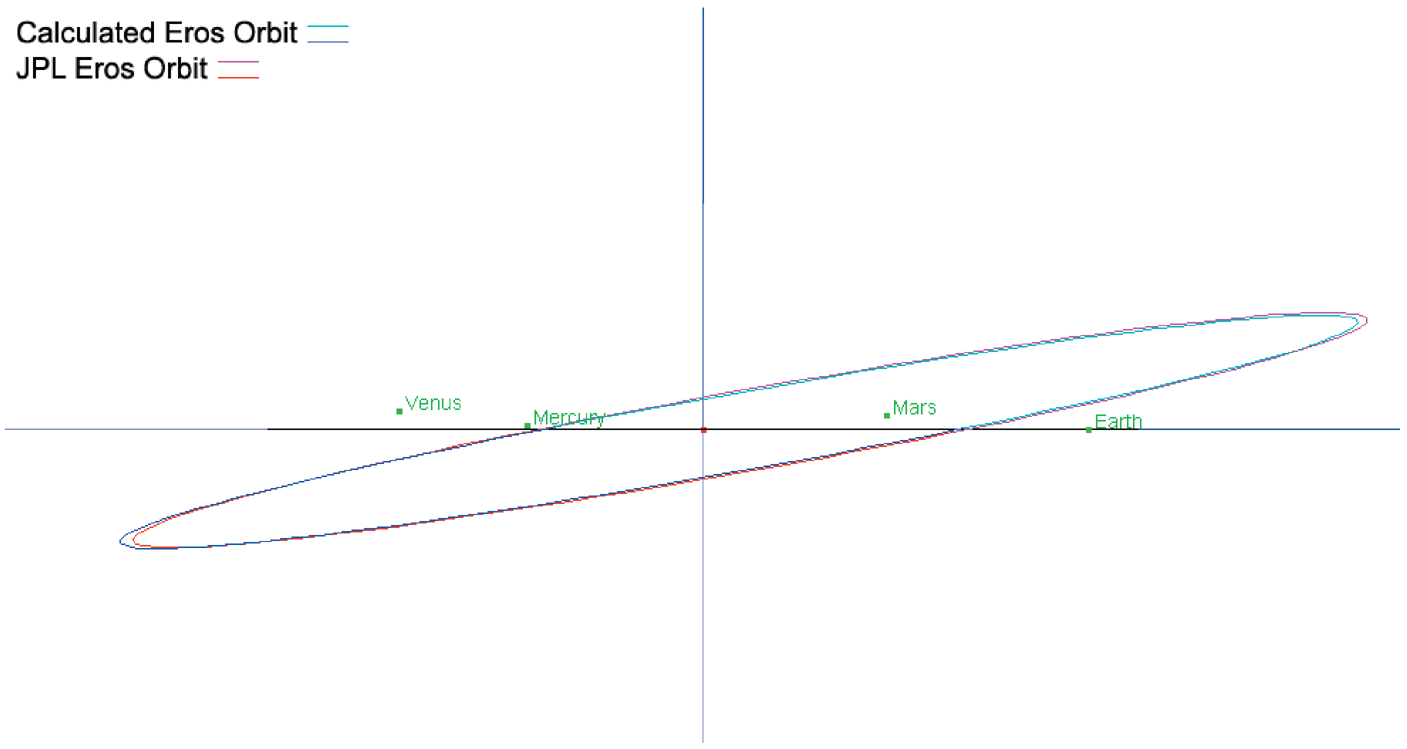

Fig. 2: Comparison of calculated and JPL orbits of Eros (seen from Earth's plane of orbit). Observe the closeness of the ascending node and inclination element between the calculated orbit of Eros and that from JPL. A change in colors indicates the path passing through an axis plane.

by California Institute of Technology's Jet Propulsion Laboratory (JPL) indicate that the calculation could be improved upon. Figure 2, Fig. 3, and Fig. 4 plot the JPL Eros orbit against the calculated Eros orbit. The plots were created using the java program "OrbitViewer" written by Osamu Ajiki and Ron Baalke (available from *http://www.astroarts.co.jp/products/orbitviewer/index.html* at time of writing).

Some amount of the error can be explained by the seeing conditions at WAO and the error in the plate solution. However, these two sources cannot account for all of the total error. A large part of the total error comes from the method of calculation used.

Conclusions

The use of Gauss's method of orbit determination was crucial for this project because it allowed the feasibility of using data taken at WAO to calculate the orbit of an NEA to be assessed with only three observations taken within the time-restricted period of a month. For this application and

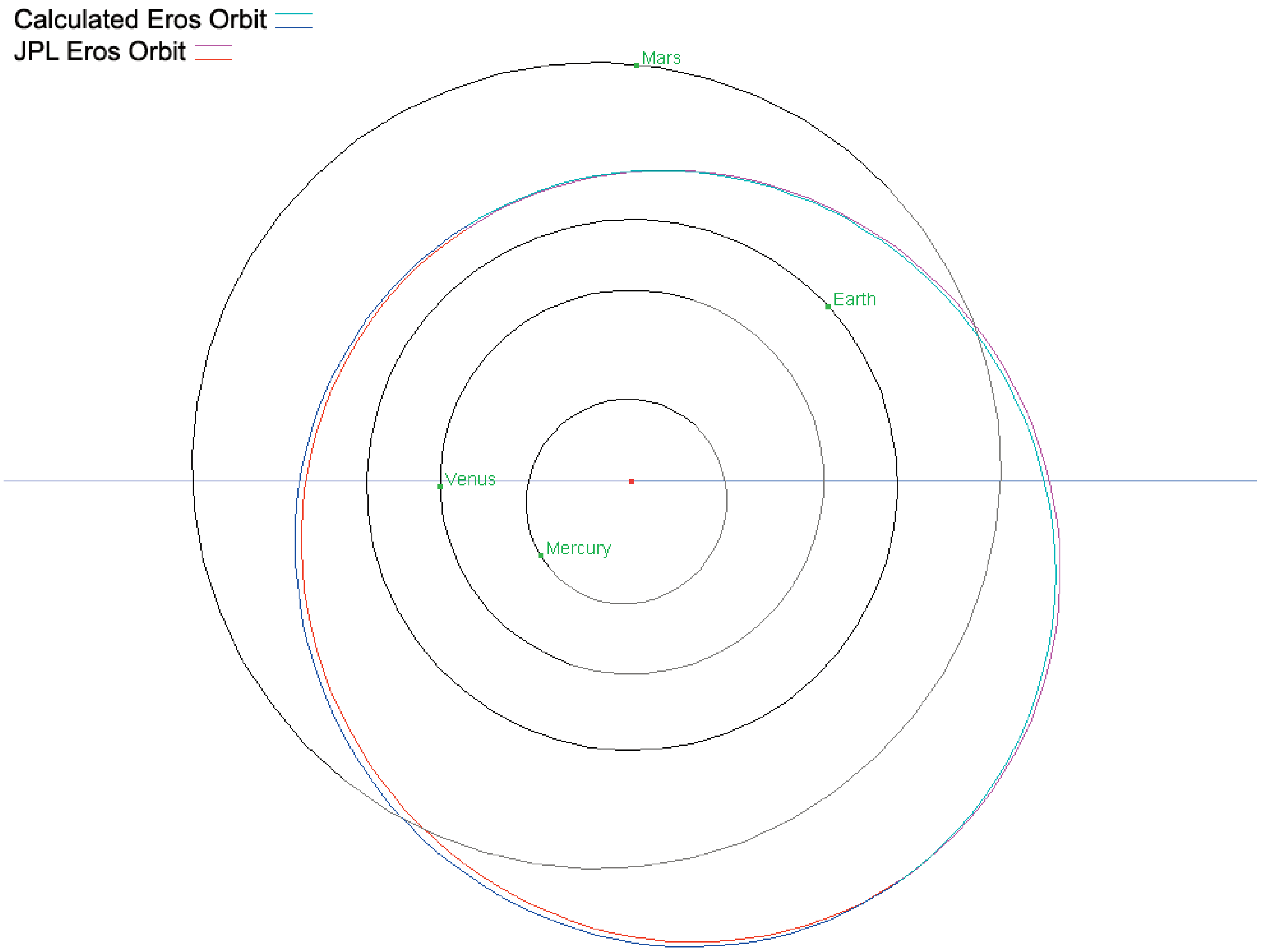

Fig. 3: Comparison of calculated and JPL orbits of Eros (seen perpendicularly from Earth's plane of orbit). Notice the two areas of the orbits that overlay each other. One of these areas is located near Eros's location during observations, the other would be the location of Eros a half-orbital period of time later.

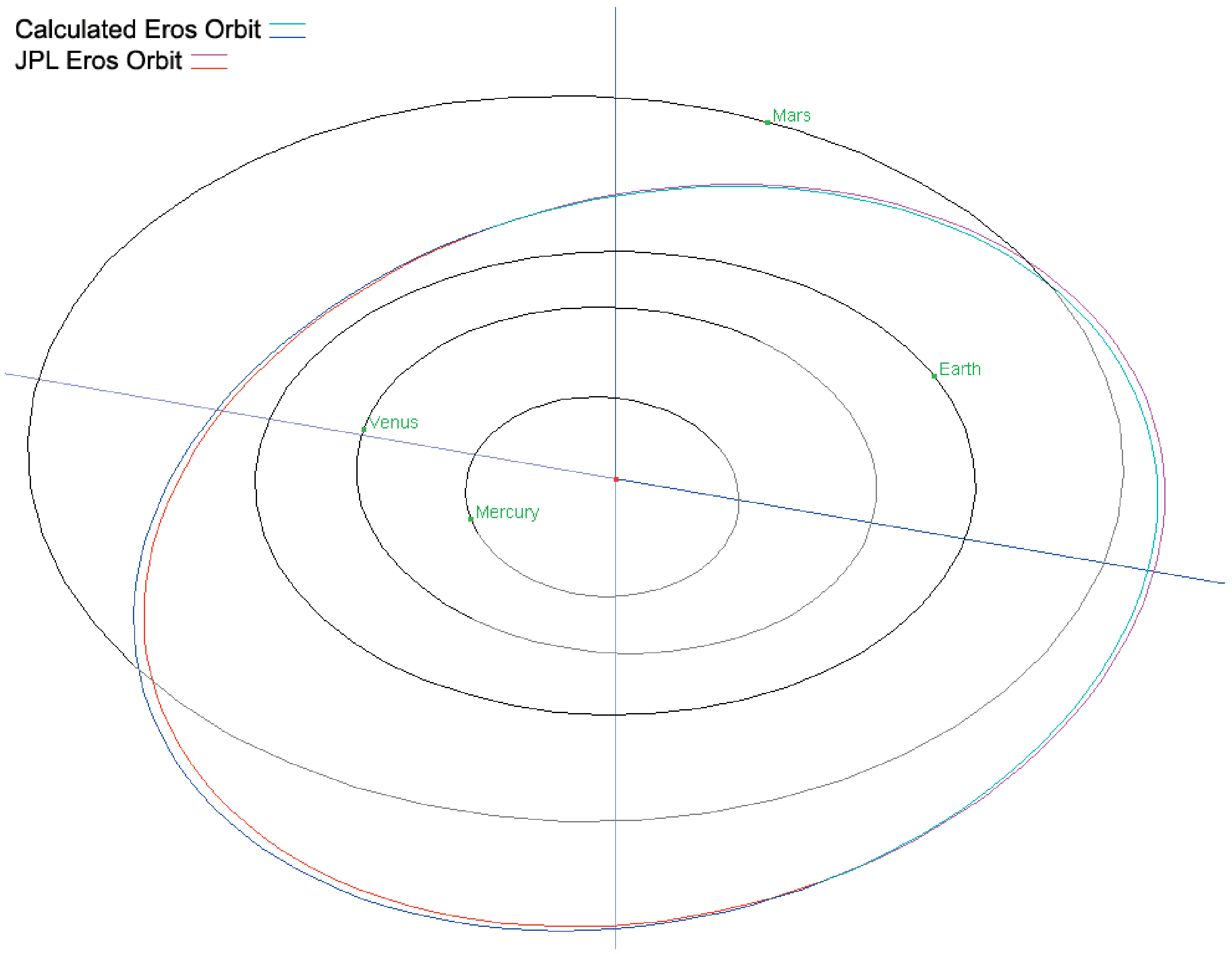

Fig. 4: Comparison of calculated and JPL orbits of Eros (seen out of Earth's plane of orbit).
This view of the orbits offers a better three dimensional understanding of the difference between the two orbits.

for rapid initial orbit determination, Gauss's method is an excellent choice. However, Gauss's method is a poor choice for calculating an accurate orbit of an NEA because it cannot offer the necessary precision needed to do so.

In order to calculate a highly precise and accurate orbit, a more robust method is needed that is capable of using more than three observations. A method of this type has the disadvantage of requiring observations throughout the full revolution period of the object. However, such a method has the advantage of being more accurate than Gauss's method as well as providing error analysis through the least-squares-fit residuals. The method employed by JPL is an example of such a method. Over 3500 observations were used in the calculation of JPL's orbit for Eros, and that calculation has errors on the order of only a millionth of a percent.

Appendix A

Bold font indicates that values are modified from default unlearn values.
Packages not listed here are run with default unlearn values.

```
PACKAGE = ccdred
   TASK = zerocombine

input   =                      List of zero level images to combine
(output =            bias.fits) Output zero level name
(combine=              average) Type of combine operation
(reject =               minmax) Type of rejection
(ccdtype=                     ) CCD image type to combine
(process=                   no) Process images before combining?
(delete =                   no) Delete input images after combining?
(clobber=                   no) Clobber existing output image?
(scale  =                 none) Image scaling
(statsec=                     ) Image section for computing statistics
(nlow   =                    0) minmax: Number of low pixels to reject
(nhigh  =                    2) minmax: Number of high pixels to reject
(nkeep  =                    1) Minimum to keep (pos) or maximum to reject (neg)
(mclip  =                  yes) Use median in sigma clipping algorithms?
(lsigma =                   3.) Lower sigma clipping factor
(hsigma =                   3.) Upper sigma clipping factor
(rdnoise=                   0.) ccdclip: CCD readout noise (electrons)
(gain   =                   1.) ccdclip: CCD gain (electrons/DN)
(snoise =                   0.) ccdclip: Sensitivity noise (fraction)
(pclip  =                 -0.5) pclip: Percentile clipping parameter
(blank  =                   0.) Value if there are no pixels
(mode   =                   ql)
```

```
PACKAGE = ccdred
   TASK = ccdproc

images  =                      List of CCD images to correct
(output =                     ) List of output CCD images
(ccdtype=                     ) CCD image type to correct
(max_cac=                    0) Maximum image caching memory (in Mbytes)
(noproc =                   no) List processing steps only?

(fixpix =                    *) Fix bad CCD lines and columns?
(oversca=                   no) Apply overscan strip correction?
(trim   =                   no) Trim the image?
(zerocor=                    *) Apply zero level correction?
(darkcor=                   no) Apply dark count correction?
(flatcor=                    *) Apply flat field correction?
(illumco=                   no) Apply illumination correction?
(fringec=                   no) Apply fringe correction?
(readcor=                   no) Convert zero level image to readout correction?
(scancor=                   no) Convert flat field image to scan correction?

(readaxi=                 line) Read out axis (column|line)
(fixfile=           abadpix.pl) File describing the bad lines and columns
(biassec=                     ) Overscan strip image section
(trimsec=                     ) Trim data section
```

```
(zero   =             bias.fits) Zero level calibration image
(dark   =             dark.fits) Dark count calibration image
(flat   =             flat.fits) Flat field images
(illum  =                      ) Illumination correction images
(fringe =                      ) Fringe correction images
(minrepl=                    1.) Minimum flat field value
(scantyp=             shortscan) Scan type (shortscan|longscan)
(nscan  =                     1) Number of short scan lines

(interac=                    no) Fit overscan interactively?
(functio=              legendre) Fitting function
(order  =                     1) Number of polynomial terms or spline pieces
(sample =                     *) Sample points to fit
(naverag=                     1) Number of sample points to combine
(niterat=                     1) Number of rejection iterations
(low_rej=                    3.) Low sigma rejection factor
(high_re=                    3.) High sigma rejection factor
(grow   =                    0.) Rejection growing radius
(mode   =                    ql)
```

*** indicates that these values change depending on whether or not the images are being bad pixel mask, bias, or flat corrected**

```
PACKAGE = ccdred
   TASK = ccdmask

image   =                         Input image
mask    =                         Output pixel mask
(ncmed  =                     7) Column box size for median level calculation
(nlmed  =                     7) Line box size for median level calculation
(ncsig  =                    15) Column box size for sigma calculation
(nlsig  =                    15) Line box size for sigma calculation
(lsigma =                   20.) Low clipping sigma
(hsigma =                   20.) High clipping sigma
(ngood  =                     5) Minimum column length of good pixel seqments
(linterp=                     2) Mask value for line interpolation
(cinterp=                     3) Mask value for column interpolation
(eqinter=                     2) Mask value for equal interpolation
(mode   =                    ql)
```

```
PACKAGE = immatch
   TASK = imcombine when creating the second part of the bad pixel mask

input   =                         List of images to combine
output  =                         List of output images
(headers=                      ) List of header files (optional)
(bpmasks=                      ) List of bad pixel masks (optional)
(rejmask=                      ) List of rejection masks (optional)
(nrejmas=                      ) List of number rejected masks (optional)
(expmask=                      ) List of exposure masks (optional)
(sigmas =                      ) List of sigma images (optional)
(imcmb  =                    $I) Keyword for IMCMB keywords
(logfile=                STDOUT) Log file

(combine=                   sum) Type of combine operation
```

```
(reject =                 minmax) Type of rejection
(project=                     no) Project highest dimension of input images?
(outtype=                   real) Output image pixel datatype
(outlimi=                       ) Output limits (x1 x2 y1 y2 ...)
(offsets=                   none) Input image offsets
(masktyp=                   none) Mask type
(maskval=                      0) Mask value
(blank  =                     0.) Value if there are no pixels

(scale  =                   none) Image scaling
(zero   =                   none) Image zero point offset
(weight =                   none) Image weights
(statsec=                       ) Image section for computing statistics
(expname=                       ) Image header exposure time keyword

(lthresh=                  INDEF) Lower threshold
(hthresh=                  INDEF) Upper threshold
(nlow   =                      0) minmax: Number of low pixels to reject
(nhigh  =                      0) minmax: Number of high pixels to reject
(nkeep  =                      *) Minimum to keep (pos) or maximum to reject (neg)
(mclip  =                    yes) Use median in sigma clipping algorithms?
(lsigma =                     3.) Lower sigma clipping factor
(hsigma =                     3.) Upper sigma clipping factor
(rdnoise=                     0.) ccdclip: CCD readout noise (electrons)
(gain   =                     1.) ccdclip: CCD gain (electrons/DN)
(snoise =                     0.) ccdclip: Sensitivity noise (fraction)
(sigscal=                    0.1) Tolerance for sigma clipping scaling corrections
(pclip  =                   -0.5) pclip: Percentile clipping parameter
(grow   =                     0.) Radius (pixels) for neighbor rejection
(mode   =                     ql)
```

*** indicates that this value is 1000*(number of darks used)**

```
PACKAGE = immatch
   TASK = imcombine when creating the bad pixel mask from the first and second parts

input   =                         List of images to combine
output  =                         List of output images
(headers=                       ) List of header files (optional)
(bpmasks=                       ) List of bad pixel masks (optional)
(rejmask=                       ) List of rejection masks (optional)
(nrejmas=                       ) List of number rejected masks (optional)
(expmask=                       ) List of exposure masks (optional)
(sigmas =                       ) List of sigma images (optional)
(imcmb  =                     $I) Keyword for IMCMB keywords
(logfile=                 STDOUT) Log file

(combine=                    sum) Type of combine operation
(reject =                   none) Type of rejection
(project=                     no) Project highest dimension of input images?
(outtype=                   real) Output image pixel datatype
(outlimi=                       ) Output limits (x1 x2 y1 y2 ...)
(offsets=                   none) Input image offsets
(masktyp=                   none) Mask type
(maskval=                      0) Mask value
(blank  =                     0.) Value if there are no pixels
```

```
(scale  =                  none) Image scaling
(zero   =                  none) Image zero point offset
(weight =                  none) Image weights
(statsec=                      ) Image section for computing statistics
(expname=                      ) Image header exposure time keyword

(lthresh=                 INDEF) Lower threshold
(hthresh=                 INDEF) Upper threshold
(nlow   =                     1) minmax: Number of low pixels to reject
(nhigh  =                     1) minmax: Number of high pixels to reject
(nkeep  =                     1) Minimum to keep (pos) or maximum to reject (neg)
(mclip  =                   yes) Use median in sigma clipping algorithms?
(lsigma =                    3.) Lower sigma clipping factor
(hsigma =                    3.) Upper sigma clipping factor
(rdnoise=                    0.) ccdclip: CCD readout noise (electrons)
(gain   =                    1.) ccdclip: CCD gain (electrons/DN)
(snoise =                    0.) ccdclip: Sensitivity noise (fraction)
(sigscal=                   0.1) Tolerance for sigma clipping scaling corrections
(pclip  =                  -0.5) pclip: Percentile clipping parameter
(grow   =                    0.) Radius (pixels) for neighbor rejection
(mode   =                    ql)



PACKAGE = ccdred
   TASK = darkcombine

input   =                        List of dark images to combine
```
`(output =             dark.fits) Output dark image root name`
```
(combine=               average) Type of combine operation
(reject =                minmax) Type of rejection
```
`(ccdtype=                      ) CCD image type to combine`
```
(process=                   yes) Process images before combining?
(delete =                    no) Delete input images after combining?
(clobber=                    no) Clobber existing output image?
```
`(scale  =                  none) Image scaling`
```
(statsec=                      ) Image section for computing statistics
(nlow   =                     0) minmax: Number of low pixels to reject
```
`(nhigh  =                     2) minmax: Number of high pixels to reject`
```
(nkeep  =                     1) Minimum to keep (pos) or maximum to reject (neg)
(mclip  =                   yes) Use median in sigma clipping algorithms?
(lsigma =                    3.) Lower sigma clipping factor
(hsigma =                    3.) Upper sigma clipping factor
(rdnoise=                    0.) ccdclip: CCD readout noise (electrons)
(gain   =                    1.) ccdclip: CCD gain (electrons/DN)
(snoise =                    0.) ccdclip: Sensitivity noise (fraction)
(pclip  =                  -0.5) pclip: Percentile clipping parameter
(blank  =                    0.) Value if there are no pixels
(mode   =                    ql)



PACKAGE = ccdred
   TASK = flatcombine

input   =                        List of flat field images to combine
```
`(output =             flat.fits) Output flat field root name`
`(combine=                median) Type of combine operation`
`(reject =              crreject) Type of rejection`

```
(ccdtype=                      ) CCD image type to combine
(process=                   yes) Process images before combining?
(subsets=                   yes) Combine images by subset parameter?
(delete =                    no) Delete input images after combining?
(clobber=                    no) Clobber existing output image?
(scale  =                  none) Image scaling
(statsec=                      ) Image section for computing statistics
(nlow   =                     1) minmax: Number of low pixels to reject
(nhigh  =                     1) minmax: Number of high pixels to reject
(nkeep  =                     1) Minimum to keep (pos) or maximum to reject (neg)
(mclip  =                   yes) Use median in sigma clipping algorithms?
(lsigma =                    3.) Lower sigma clipping factor
(hsigma =                    3.) Upper sigma clipping factor
(rdnoise=                    0.) ccdclip: CCD readout noise (electrons)
(gain   =                    1.) ccdclip: CCD gain (electrons/DN)
(snoise =                    0.) ccdclip: Sensitivity noise (fraction)
(pclip  =                  -0.5) pclip: Percentile clipping parameter
(blank  =                    1.) Value if there are no pixels
(mode   =                    ql)



PACKAGE = user
   TASK = autoast

imname  =                        image name
(db_pref=                ast.db) name prefix for solution parameters output file
(res_pre=               ast.res) name prefix for solution residuals output file
(racen  =                      ) Initial guess field center R.A. in sexagesimal format
(deccen =                      ) Initial guess field center Dec. in sexagesimal format
(equinox=                 2000.) Equinox of coordinates
(use_key=                   yes) take as initial guess the header coordinates
(ra_key =               objctra) R.A. image header keyword
(ha_key =                 telha) H.A. image header keyword
(dec_key=              objctdec) Dec. image header keyword
(equ_key=                 epoch) Equinox image header keyword
(st_key =                   lst) Sidereal Time image header keyword
(ut_key =              time-obs) Universal Time image header keyword
(createc=                   yes) run daofind and create *.coo file
(magfile=               default) phot .mag file
(fwhm   =                  2.62) PSF FWHM in pixels
(readnoi=                   15.) CCD read out noise in electrons
(gain   =                    2.) CCD gain in electrons per count
(scale  =                   1.2) Approximate CCD scale in arcsec/pix
(numpix =                  1024) CCD Width in pixels
(width_a=                 1000.) Initial search width in arcsec.
(xytol  =                   10.) matching tolerance for pgshift
(tol_xyx=                   10.) matching tolerance for xyxymatch
(objectn=                   100) Max. number of stars to match
(catalogue=                 usnosa2) catalogue to use (usnoa1 | usnoa2 | usnosa1 | usnosa2)
(fitgeo =               general) Fitting geometry
(functio=            polynomial) surface type to fit: (chebyshev, legendre, polynomial)
(xxorder=                     2) order of Xi fit in X axis
(xyorder=                     2) order of Xi fit in Y axis
(xxterms=                  half) Xi fit cross terms type, (none, half, full)
(yxorder=                     2) order of Eta fit in X axis
(yyorder=                     2) order of Eta fit in Y axis
(yxterms=                  half) Eta fit cross terms type, (none, half, full)
(wcsupda=                   yes) Update the Image WCS
(input_c=                      ) File name containing coordinates (X Y or RA Dec)
```

```
(pre_cf_=                    out) prefix for output file
(dir_xy2=                    yes) Transform X Y to RA Dec or vice versa
(interac=                     no) Interactive rejection of residuals
(succeed=                     no) succeeded to find astrometric solution
(scat_pa= /cygdrive/e/pal/iraf/wcstools/bin) path for the scat search program
(path_us=                       ) path for the USNO SA1.0 catalogue
(path_us= /cygdrive/e/pal/iraf/usnoa2) path for the USNO SA2.0 catalogue
(path_ng=                       ) path for the GSC North catalogue
(path_sg=                       ) path for the GSC South catalogue
(logfile=            autoast.log) logfile name
(errlog =              error.log) error log file
(no_line=                       ) Number of stars used in astrometric solution
(run_sam=                     no) Only for run on same field - Use old catalogue file
(lis2   =                       )
(lis3   =                       )
(list   =                       )
(lis_wc =                       )
(mode   =                     ql)
```

Note: most of these options must be set differently for each system and image

Appendix B

```
 % gaussmethod.m
%
% This program uses Gauss method to calculate the equatorial position- and velocity-
% vectors of a Solar System object at a particular instant of time, based
% on observed positions of the object. These vectors may then be used to calculate
% the orbital elements of the object.
% requires date2jd.m and Qfunk.m

%%%%%%%%%%%%%%%%%%%%%%%%% OBSERVATIONAL DATA %%%%%%%%%%%%%%%%%%%%%%%


% First observation
t(1)=date2jd(2009, 09, 21, 03, 04, 33.562);     % JD of observation OR UT of observation
date2jd(YYYY, MM, DD, HH, MM, SS.SSS)
RA1=[21 47 46.33];                              % Right ascension [h m s]
DEC1=[05 19 55.36];                             % Declination [deg arcmin arcsec]
% LST1=[22 19 07.779];                          % Local sidereal time [h m s]

% Second observation
t(2)=date2jd(2009, 10, 12, 01, 50, 09.784);
RA2=[21 31 43.26];
DEC2=[03 54 23.19];
% LST2=[22 27 19.419];

% Third observation
t(3)=date2jd(2009, 10, 26, 02, 04, 59.342);
RA3=[21 34 18.58];
DEC3=[03 22 30];
% LST3=[23 37 23.191];

% Site location
H =.1029;                                    % Observation elevation (Km)
lat = [42 36 36.8];                          % Observation latitude [deg arcmin arcsec]

% settings
iterate = 10000;                                %Set number of iterations

% Constants

Re = 6378;                                   % Earth's Radius
f = 1/298.26;                                % Earth’s flattening factor
kgauss=0.01720209895;                        % The Gauss gravitational constant
deg = pi/180;                                % Deg -> Rad
mu = 398600;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Angles to radians

alpha1=15*(RA1(1)+RA1(2)/60+RA1(3)/3600)*deg;
delta1=sign(DEC1(1))*(abs(DEC1(1))+DEC1(2)/60+DEC1(3)/3600)*deg;
%theta1=15*(LST1(1)+LST1(2)/60+LST1(3)/3600)*deg;

alpha2=15*(RA2(1)+RA2(2)/60+RA2(3)/3600)*deg;
delta2=sign(DEC2(1))*(abs(DEC2(1))+DEC2(2)/60+DEC2(3)/3600)*deg;
```

```
%theta2=15*(LST2(1)+LST2(2)/60+LST2(3)/3600)*deg;

alpha3=15*(RA3(1)+RA3(2)/60+RA3(3)/3600)*deg;
delta3=sign(DEC3(1))*(abs(DEC3(1))+DEC3(2)/60+DEC3(3)/3600)*deg;
%theta3=15*(LST3(1)+LST3(2)/60+LST3(3)/3600)*deg;

phi=sign(lat(1))*(abs(lat(1))+lat(2)/60+lat(3)/3600)*deg;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Calculate R
for l=1:3
    tp(l) = t(l) - 2451545.0;
    g(l) = 357.528*deg +.9856003*deg*tp(l);
    L(l) = 280.460*deg + .9856474*deg*tp(l);
    lambda(l) = L(l) + 1.915*deg*sin(g(l)) + .020*deg*sin(2*g(l));
    epsilon(l) = 23.439*deg - .0000004*deg*tp(l);
    dist(l) = 1.00014 - .01671*cos(g(l)) - .00014*cos(2*g(l));

    x(l) = dist(l)*cos(lambda(l));
    y(l) = dist(l)*cos(epsilon(l))*sin(lambda(l));
    z(l) = dist(l)*sin(epsilon(l))*sin(lambda(l));
end

R1 = [x(1); y(1); z(1)];
R2 = [x(2); y(2); z(2)];
R3 = [x(3); y(3); z(3)];

% Extract times

t1=t(1);
t2=t(2);
t3=t(3);

Deltahat1=[cos(alpha1)*cos(delta1);sin(alpha1)*cos(delta1);sin(delta1)];
Deltahat2=[cos(alpha2)*cos(delta2);sin(alpha2)*cos(delta2);sin(delta2)];
Deltahat3=[cos(alpha3)*cos(delta3);sin(alpha3)*cos(delta3);sin(delta3)];


% Cunningham system coordinate axes
% expressed in geocentric equatorial system

% xi unit vector equal to Deltahat1

% eta unit vector
eta=cross(Deltahat1,cross(Deltahat3,Deltahat1));
eta=eta/norm(eta);

% zeta unit vector
zeta=cross(Deltahat1,eta);
zeta=zeta/norm(zeta);



% Transformation matrix
% (geocentric equatorial system -> Cunningham system)
RM=zeros(3,3);
RM(1,:)=Deltahat1';
RM(2,:)=eta';
RM(3,:)=zeta';

% Solar position coordinates in Cunningham system
% "R prime"
R1P=RM*R1;
R2P=RM*R2;
R3P=RM*R3;

% Calculate the geocentric unit position vectors of asteroid
% in Cunningham system
% "Deltahat prime"

% Deltahat1P equals [1,0,0]

Deltahat2P=RM*Deltahat2;
Deltahat3P=RM*Deltahat3;

% Using the notations of the Compendium
% for clarity

xi2=Deltahat2P(1);
eta2=Deltahat2P(2);
zeta2=Deltahat2P(3);
xi3=Deltahat3P(1);
eta3=Deltahat3P(2);

% Print the zeta2 element to the screen
disp(' ');
disp(['zeta2 = ',num2str(zeta2)]);
disp(' ');

% Initial guess of c1 and c3
c1=(t3-t2)/(t3-t1);
c3=(t2-t1)/(t3-t1);

% Allocate space for the geocentric distances "Delta" of
% the asteroid at the time of the three observations

Delta=[0;0;0];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ITERATION OF GAUSS' METHOD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:iterate
    % Geocentric distances from current c1 & c3
    % Eq. (189)
    Delta(2)=(-c1*R1P(3)+R2P(3)-c3*R3P(3))/zeta2;
    Delta(3)=(Delta(2)*eta2+c1*R1P(2)-R2P(2)+c3*R3P(2))/(c3*eta3);
    Delta(1)=(Delta(2)*xi2-c3*Delta(3)*xi3+c1*R1P(1)-R2P(1)+c3*R3P(1))/c1;

    % Heliocentric equatorial coordinates of asteroid from current c1 & c3
    % Eq. (177)
    r1=Delta(1)*Deltahat1-R1;
    r2=Delta(2)*Deltahat2-R2;
    r3=Delta(3)*Deltahat3-R3;

    % Print geocentric distances to screen
    disp(['Iteration #',num2str(i),': Delta1 = ',num2str(Delta(1)),...
            'AU  Delta2 = ',num2str(Delta(2)),'AU  Delta3 = ',num2str(Delta(3)),'AU']);


    % Steffensen's method for calculating y1, y2, and y3 (see Eq. 180)
    % These are collected in the array y=[y1 y2 y3]
    % The quantities called y_(1), y_(2), and y_(3) in the Compendium
    % are here denoted yone, ytwo, and ythree.

    for j=1:3
      % Consider one observational occasion at a time
      if (j==1)
        % y1 (2->3)
        K=sqrt(2*(norm(r2)*norm(r3)+dot(r2,r3)));
        m2=(kgauss*(t3-t2))^2/K^3;
        L=(norm(r2)+norm(r3)-K)/(2*K);
      end;
      if (j==2)
        % y2 (1->3)
        K=sqrt(2*(norm(r1)*norm(r3)+dot(r1,r3)));
        m2=(kgauss*(t3-t1))^2/K^3;
        L=(norm(r1)+norm(r3)-K)/(2*K);
      end;
      if (j==3)
        % y3 (1->2)
        K=sqrt(2*(norm(r1)*norm(r2)+dot(r1,r2)));
        m2=(kgauss*(t2-t1))^2/K^3;
        L=(norm(r1)+norm(r2)-K)/(2*K);
      end;

      yone=1;
      xstar=(m2/yone^2)-L;
      ytwo=1+(m2/yone^2)*4*Qfunk(xstar)/3;
      xstar2=(m2/ytwo^2)-L;
      ythree=1+(m2/ytwo^2)*4*Qfunk(xstar2)/3;
      getout=0;
      % while(abs(yone-2*ytwo+ythree)>1e-14 && getout<50)
        getout=getout+1;
        y(j)=yone-(ytwo-yone)^2/(yone-2*ytwo+ythree);
        yone=y(j);
        xstar=(m2/yone^2)-L;
        ytwo=1+(m2/yone^2)*4*Qfunk(xstar)/3;
        xstar2=(m2/ytwo^2)-L;
        ythree=1+(m2/ytwo^2)*4*Qfunk(xstar2)/3;
      % end;
   end;

   % Based on y, calculate new values of c1 and c2
   c1=y(2)*(t3-t2)/(y(1)*(t3-t1));
   c3=y(2)*(t2-t1)/(y(3)*(t3-t1));
end;




% Correct for planetary abberation
t1=t1-0.005768*Delta(1);
t2=t2-0.005768*Delta(2);
t3=t3-0.005768*Delta(3);

if (1==0)
```

```
% Approximate calculation of
% velocity vector at second
% observation
v12=(r2-r1)/(t2-t1);
v23=(r3-r2)/(t3-t2);
v2=((t2-t1)*v12+(t3-t2)*v23)/((t2-t1)+(t3-t2));
end;


if (1==1)% Velocity from f and g series
tau1=kgauss*(t1-t2);
tau3=kgauss*(t3-t2);
f1=1-0.5*tau1^2/norm(r2)^3;
f3=1-0.5*tau3^2/norm(r2)^3;
g1=(tau1-tau1^3/(6*norm(r2)^3))/kgauss;
g3=(tau3-tau3^3/(6*norm(r2)^3))/kgauss;
v2a=(Delta(1)*Deltahat1-R1-f1*r2)/g1;
v2b=(Delta(3)*Deltahat3-R3-f3*r2)/g3;
v2=0.5*(v2a+v2b);
end;


% Transformation to ecliptic system
eps=(23.439-4e-7*(t2-2451545))*deg;
toequa=[1 0 0; 0 cos(eps) -sin(eps); 0 sin(eps) cos(eps)];
toecl=inv(toequa);
r2ecl=toecl*r2;
v2ecl=toecl*v2;

% Print position and velocity vectors to
% the screen

disp(' ');
disp(['Julian date: ',num2str(t2)]);
disp(['Heliocentric ecliptic position vector: [',num2str(r2ecl(1)),',
',num2str(r2ecl(2)),', ',num2str(r2ecl(3)),'] AU']);
disp(['Heliocentric ecliptic velocity vector: [',num2str(v2ecl(1),'%0.5e'),',
',num2str(v2ecl(2),'%0.5e'),', ',num2str(v2ecl(3),'%0.5e'),'] AU/day']);



% coe = coe_from_sv(transpose(r2),transpose(v2));
% fprintf('\n Angular momentum (km^2/s) = %g', coe(1))
% fprintf('\n Eccentricity = %g', coe(2))
% fprintf('\n RA of ascending node (deg) = %g', coe(3)/deg)
% fprintf('\n Inclination (deg) = %g', coe(4)/deg)
% fprintf('\n Argument of perigee (deg) = %g', coe(5)/deg)
% fprintf('\n True anomaly (deg) = %g', coe(6)/deg)
% fprintf('\n Semimajor axis (km) = %g', coe(7))
% fprintf('\n Periapse radius (km) = %g', coe(1)^2/mu/(1 + coe(2)))
% %...If the orbit is an ellipse, output the period:
% if coe(2)<1
% T = 2*pi/sqrt(mu)*coe(7)^1.5;
% fprintf('\n Period:')
% fprintf('\n Seconds = %g', T)
% fprintf('\n Minutes = %g', T/60)
% fprintf('\n Hours = %g', T/3600)
% fprintf('\n Days = %g', T/24/3600)
% end
% fprintf('\n-----------------------------------------------\n')
% Qfunk.m
%
% This function evaluates eq. 6.11.31-6.11.32 in Danby

function Q=Qfunk(x)

if (x>0 && x<=0.5)
    Q=2*(2*x-1)*sqrt(x-x^2)+asin(2*x-1)+pi/2;
    Q=3*Q/(16*(x-x^2)^(3/2));
end;

if (x<0)
    Q=2*(1-2*x)*sqrt(x^2-x)-log(1-2*x+2*sqrt(x^2-x));
    Q=3*Q/(16*(x^2-x)^(3/2));
end;

if (x==1)
    Q=1;
end;




% date2jd.m
function jd = date2jd(varargin)
%DATE2JD Julian day number from Gregorian date.
%
%   JD = DATE2JD(YEAR, MONTH, DAY, HOUR, MINUTE, SECOND) returns the Julian
%   day number of the given date (Gregorian calendar) plus a fractional part
%   depending on the time of day.
%
%   Any missing MONTH or DAY will be replaced by ones.  Any missing HOUR,
%   MINUTE or SECOND will be replaced by zeros.
%
%   If no date is specified, the current date and time is used.
%
%   Start of the JD (Julian day) count is from 0 at 12 noon 1 January -4712
%   (4713 BC), Julian proleptic calendar.  Note that this day count conforms
%   with the astronomical convention starting the day at noon, in contrast
%   with the civil practice where the day starts with midnight.
%
%   Astronomers have used the Julian period to assign a unique number to
%   every day since 1 January 4713 BC.  This is the so-called Julian Day
%   (JD).  JD 0 designates the 24 hours from noon UTC on 1 January 4713 BC
%   (Julian proleptic calendar) to noon UTC on 2 January 4713 BC.

%   Sources:  - http://tycho.usno.navy.mil/mjd.html
%             - The Calendar FAQ (http://www.faqs.org)

%   Author:      Peter J. Acklam
%   Time-stamp:  2002-05-24 13:30:06 +0200
%   E-mail:      pjacklam@online.no
%   URL:         http://home.online.no/~pjacklam

   nargsin = nargin;
   error(nargchk(0, 6, nargsin));
   if nargsin
```

```
      argv = {1 1 1 0 0 0};
      argv(1:nargsin) = varargin;
   else
      argv = num2cell(clock);
   end
   [year, month, day, hour, minute, second] = deal(argv{:});

   % The following algorithm is a modified version of the one found in the
   % Calendar FAQ.

   a = floor((14 - month)/12);
   y = year + 4800 - a;
   m = month + 12*a - 3;

   % For a date in the Gregorian calendar:
   jd = day + floor((153*m + 2)/5) + y*365 + floor(y/4) - floor(y/100) + floor(y/400) -
32045 + ( second + 60*minute + 3600*(hour - 12) )/86400;
```