ISSN I044 - 9825

CELESTIAL COMPUTING

A Journal for Personal Computers and Celestial Mechanics

Volume 1 Number 4 Fall 1989

Copyright © 1989 by David Eagle. All rights reserved.

FEATURE ARTICLE

In this issue of *CELESTIAL COMPUTING*, we feature an interactive QuickBASIC computer program called **PLANETS.BAS** which can be used to predict special planetary events and positions. The planetary events which can be calculated with this software are

- ▹ Perihelion and aphelion of a planet
- ▹ Greatest elongation or opposition of a planet
- ▹ Conjunction in longitude between two planets
- ▹ Closest approach distance between two planets
- ▹ Minimum angular separation between two planets
- ▹ Time of the equinoxes and solstices of the Sun
- ▹ Position of a planet at a user specified epoch

Program **PLANETS** includes three different algorithms for computing planetary positions. It also illustrates a typical application of the theory of minima and maxima to solve astronomical problems. A discussion about *minmax* problems can be found in the NUMERICAL METHODS column for this issue, and a short tutorial about coordinate systems and transformations is provided in the FUNDAMENTAL ASTRONOMY column.

The following is a brief description of each of the main menu options of program **PLANETS**.

Perihelion and aphelion of a planet

This option determines the time when a planet is closest to the Sun and the time when a planet reaches its greatest distance from the Sun. The minimum distance of a planet to the Sun is called *perihelion* and the maximum heliocentric distance of a planet is called *aphelion*.

Greatest elongation or opposition of a planet

This option of program **PLANETS** determines the *greatest elongation* of an inner planet or the time of *opposition* of an outer planet. Greatest elongation is the instant when the angle between an inner planet and the Sun, when viewed from the Earth, reaches its maximum value. The time of opposition of an outer planet is the instant when the geocentric (Earth-centered) longitude of the Sun and planet differ by exactly 180 degrees.

Conjunction in longitude between two planets

This option determines the time of *conjunction in longitude* of two planets relative to a third body. Conjunction is the instant when two planets have the same apparent celestial longitude. The data computed by **PLANETS** will be conjunction in longitude of two planets relative to the Sun unless the Earth is one of the planets selected by the user. In this case, the data displayed is the *geocentric conjunction* of the Sun and the planet.

Closest approach distance between two planets

This option of program **PLANETS** determines the time of closest approach distance between two planets. This is the instant in time when the *separation distance* between two planets reaches a minimum value.

Minimum angular separation between two planets

This option determines the time of *minimum angular separation* between two planets. This is the instant when the angle between two planets, as seen from the Earth, reaches its minimum value.

Time of the equinoxes and solstices of the Sun

This option determines the time of the *equinoxes* and *solstices* of the Sun. An equinox is the point where the apparent geocentric longitude of the Sun is either 0 degrees (Spring) or 180 degrees (Fall). At a solstice, the Sun's longitude is either 90 degrees (Summer) or 270 degrees (Winter).

Position of a planet at a user specified epoch

This option determines the position of a planet or the Sun at a specific epoch. For this selection, the user inputs a valid calendar date and local civil time, and the program determines the geocentric and topocentric position of the planet or the Sun.

Please note that events computed by program **PLANETS** are evaluated in the *geocentric* frame of reference. Although the planet position is also computed in the topocentric frame, many celestial events may not be visible at the observer's location. However, this method of computation insures that any planetary event is always predicted. The possibility of visibility can be determined by the topocentric azimuth and elevation angles at the time of the planetary event. If the elevation angle is *positive*, the view in the azimuth direction is not obscured, and the weather cooperates, viewing the event is visible.

The following is a short discussion about the important numerical algorithms used in program **PLANETS.**

Position of the Sun

The position of the Sun is computed with the data and algorithms described in the book, *Planetary Programs and Tables* by Pierre Bretagnon and Jean-Louis Simon. The mathematical algorithm described here closely follows that given in the book.

The fundamental time argument is the number of days relative to the Julian epoch January 1, 2000 at 0 hours Ephemeris Time (ET) *normalized* with respect to 3652500 Julian days. This value can be calculated for any Julian Ephemeris Date (JED) with

$$U = \frac{JED - 2451545}{3652500} \qquad (1)$$

The *geocentric, ecliptic* <u>mean</u> longitude of the Sun is calculated with a trigonometric series of the form

$$\lambda_{\odot}^{m} = \lambda_0 + \lambda_1 U + \sum_{i=1}^{50} l_i \sin(\alpha_i + \nu_i U) \qquad (2)$$

The *geocentric* distance of the Sun is calculated with another series of the form

$$R_{\odot} = R_0 + R_1 U + \sum_{i=1}^{50} r_i \cos(\alpha_i + \nu_i U) \qquad (3)$$

The longitude of the Sun is corrected for the effect of *aberration* (in units of radians) with the following equation

$$\Delta\lambda_{\odot}^{a} = 10^{-7} \, (-993 + 17 \cos(3.10 + 62830.14 \, U)) \qquad (4)$$

The *nutation in longitude* (in units of radians) is calculated from

$$\Delta\psi = 10^{-7} \, (-834 \sin A1 - 64 \sin A2) \qquad (5)$$

where

$$A1 = 2.18 - 3375.70 \, U + 0.36 \, U^2$$

$$A2 = 3.51 + 125666.39 \, U + 0.10 \, U^2$$

The *apparent, geocentric ecliptic* longitude of the Sun is determined as the combination of these three components with the next equation

$$\lambda_{\odot} = \lambda_{\odot}^{m} + \Delta\lambda_{\odot}^{a} + \Delta\psi \qquad (6)$$

The three components of the *geocentric, ecliptic* position vector of the Sun are given by

$$X_{\odot} = R_{\odot} \cos \lambda_{\odot} \qquad (7a)$$

$$Y_{\odot} = R_{\odot} \sin \lambda_{\odot} \qquad (7b)$$

$$Z_{\odot} = 0 \qquad (7c)$$

We can compute the *apparent* geocentric, equatorial right ascension α and declination δ of the Sun from the two equations

$$\alpha_{\odot} = \tan^{-1} (\cos \lambda_{\odot}, \sin \varepsilon \sin \lambda_{\odot}) \qquad (8)$$

$$\delta_{\odot} = \sin^{-1} (\sin \varepsilon \sin \lambda_{\odot}) \qquad (9)$$

The inverse tangent used here is a two argument function which determines the value of an angle between 0 and 2π radians. ε is the true obliquity of the ecliptic.

Finally, we can compute the three components of the apparent, geocentric equatorial *unit* position vector $\hat{U}_{\odot}$ of the Sun with the following equations

$$U_{\odot x} = \cos \alpha_{\odot} \cos \delta_{\odot} \qquad (10a)$$

$$U_{\odot y} = \sin \alpha_{\odot} \cos \delta_{\odot} \qquad (10b)$$

$$U_{\odot z} = \sin \delta_{\odot} \qquad (10c)$$

The geocentric, equatorial position vector of the Sun can be determined from the distance and unit pointing vector with

$$\bar{R}_{\odot} = R_{\odot} \hat{U}_{\odot} \qquad (11)$$

Position of the Planets

Program **PLANETS.BAS** uses three different algorithms to compute planetary positions. This approach was chosen to demonstrate the computer subroutines necessary to implement each method. And besides, it was challenging and fun!

The position of Pluto is determined from the method described in *An Accurate Representation of the Motion of Pluto* by E. Goffin, J. Meeus, and C. Steyaert which appeared in Volume 155 of *Astronomy and Astrophysics*, pages 323-325, 1986. This algorithm is valid for the calendar years 1885-2099.

The fundamental time argument for this method is a function of the Julian Ephemeris Day JED according to the equation

$$T = \frac{JED - 2415020}{3652500} \qquad (12)$$

The heliocentric ecliptic coordinates are computed from series of the form

$$\lambda = \lambda^m + \sum_{i=1}^{43} A \sin \alpha + B \cos \alpha \qquad (13)$$

where

λ^m = coordinate mean value

$\alpha = iJ + jS + kP$

J, S, P = mean longitudes of Jupiter, Saturn and Pluto

i, j, k = integer constants

A, B = coefficients of periodic terms

The position of Mars is calculated with the method described in Chapters 23, 24 and 25 of *Astronomical Formulae for Calculators* by Jean Meeus. The orbital elements are represented by polynomials of the form

$$a_0 + a_1T + a_2T^2 + a_3T^3 \qquad (14)$$

where the time argument T is given by Eq. 12 above.

The orbital elements calculated are as follows.

L = mean longitude

a = semimajor axis

e = orbital eccentricity

ι = orbital inclination

ω = argument of perihelion

Ω = longitude of the ascending node

From these elements, the Martian longitude of perihelion is

$$\pi = \omega + \Omega \tag{15}$$

and the mean anomaly of Mars is given by

$$M = L - \pi = L - \omega - \Omega \tag{16}$$

An iterative *Newton-Raphson* method is used to compute the true anomaly ν of Mars. The heliocentric distance of Mars is calculated from

$$r_p = \frac{a\,(1 - e^2)}{1 + e \cos \nu} \tag{17}$$

From the argument of latitude

$$u = L + \nu - M - \Omega \tag{18}$$

the heliocentric, ecliptic longitude of Mars can be computed from

$$\lambda = \text{ATAN3}\,(\cos \iota \sin u, \cos u) \tag{19}$$

where ATAN3 is a four quadrant inverse tangent function.

The heliocentric, ecliptic latitude of Mars is

$$\beta = \sin^{-1} (\sin u \sin \iota) \tag{20}$$

The calculations for Mars also include long period corrections to the mean longitude and mean anomaly, and periodic corrections to the longitude and heliocentric distance.

The positions of the other planets are calculated with the method described in *Low-Precision Formulae for Planetary Positions*. This algorithm has been discussed in several earlier issues of *CELESTIAL COMPUTING* and the reader is referred to those articles.

The *geocentric equatorial* X, Y, and Z components and magnitude R of a planet's position vector are calculated with

$$X_{eq} = r_p \cos\beta_p \cos\lambda_p + r_\odot \cos\beta_\odot \cos\lambda_\odot \tag{21a}$$

$$\begin{aligned} Y_{eq} = {} & r_p \left[\cos\beta_p \sin\lambda_p \cos\varepsilon - \sin\beta_p \sin\varepsilon\right] \\ & + r_\odot \left[\cos\beta_\odot \sin\lambda_\odot \cos\varepsilon - \sin\beta_\odot \sin\varepsilon\right] \end{aligned} \tag{21b}$$

$$\begin{aligned} Z_{eq} = {} & r_p \left[\cos\beta_p \sin\lambda_p \sin\varepsilon + \sin\beta_p \cos\varepsilon\right] \\ & + r_\odot \left[\cos\beta_\odot \sin\lambda_\odot \sin\varepsilon - \sin\beta_\odot \cos\varepsilon\right] \end{aligned} \tag{21c}$$

$$R_{eq} = \sqrt{X_{eq}^2 + Y_{eq}^2 + Z_{eq}^2} \tag{22}$$

where

r_p	= heliocentric distance of a planet
$r_\odot$	= geocentric distance of the Sun
β_p	= ecliptic latitude of a planet
$\beta_\odot$	= ecliptic latitude of the Sun
λ_p	= ecliptic longitude of a planet
$\lambda_\odot$	= ecliptic longitude of the Sun
ε	= obliquity of the ecliptic

Program **PLANETS** utilizes what are called *objective functions* to predict planetary events or phenomena. These objective functions define the geometry of the planets relative to the Sun, Earth and each other as a function of time.

As an example, let's consider the *closest approach distance* between two planets. The objective function in this case is an equation which describes how the *heliocentric* separation distance between any two planets changes as a function of time. The planets are closest to one another whenever this separation distance is a *minimum*.

Mathematically, we calculate

$$\Delta = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2} \qquad (23)$$

In this equation, "Δ" is the objective function we seek and

X_1, Y_1, Z_1 = heliocentric position vector of first planet

X_2, Y_2, Z_2 = heliocentric position vector of second planet

The closest approach distance is the instant in time when this objective function reaches a *minimum* value. **PLANETS** searches for all the minimum values of the objective function contained in the time interval specified by the user.

Searching for minimum and maximum values of a function can be done in several ways. One method is a *brute force* technique which consists of stepping forward in time at small time steps looking for a minimum value of the objective function. This method is very slow and inefficient. Another technique is to use a numerical method which calculates a function's minima by using information about the behavior of the function itself. This is the technique used in program **PLANETS**.

The minimization method used by program **PLANETS** is described in Chapter 10 of *Numerical Recipes* by W. Press, B. Flannery, S. Teukolsky and W. Vetterling. The algorithm is due to Richard Brent and consists of a two step approach to the minimization problem. The first step of the algorithm *brackets* the function minima (or maxima), and the second step actually calculates the minima or maxima. A function *maxima* can be found by minimizing the negative value of the objective function. A unique feature of Brent's method is that it does not require calculation of the *derivatives* of the objective function. A more detailed discussion about Brent's method of bracketing and minimization is provided in the NUMERICAL METHODS column for this issue.

The following is a short discussion of the objective functions used in the different options of program **PLANETS**.

Perihelion and aphelion of a planet

Perihelion is the minimum value of the heliocentric distance of a planet, and aphelion is the maximum value of this distance. The objective function *F* is given by

$$F = R_p \qquad (24)$$

where

R_p = heliocentric distance of a planet

Greatest elongation or opposition of a planet

The objective function used here is the geocentric elongation angle of the planet. This option searches for the maximum value of the elongation angle which can be calculated from

$$E = \cos^{-1}\left\{\frac{R_\odot^2 + R_{pg}^2 - R_{ph}^2}{2\,R_\odot\,R_{pg}}\right\} \tag{25}$$

where

$R_\odot$ = geocentric position of the Sun

R_{pg} = geocentric position of the planet

R_{ph} = heliocentric position of the planet

Since we seek a *maximum*, the objective function is given by

$$F = -E$$

Conjunction in longitude between two planets

This objective function is the minimum value of the difference in apparent heliocentric or geocentric celestial longitude of two planets. This can be written as

$$F = \lambda_2 - \lambda_1 \tag{26}$$

where

λ_1, λ_2 = heliocentric or geocentric celestial longitude of each planet

The heliocentric longitudes are used unless the Earth is one of the planets selected by the user. Otherwise, geocentric longitudes are used in the objective function.

Closest approach distance between two planets

The objective function used in this option is the minimum value of the *heliocentric distance* between any two planets. The objective function is

$$F = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2} \tag{27}$$

where

X_1, Y_1, Z_1 = position vector of first planet

X_2, Y_2, Z_2 = position vector of second planet

Minimum angular separation between two planets

The objective function used here is the value of the *geocentric pointing angle* between two planets. This function is calculated by finding the angle between the geocentric unit position vectors of the two planets with the following *dot product* equation.

$$\theta = \cos^{-1}(\hat{U}_{p1} \cdot \hat{U}_{p2}) \tag{28}$$

where

$\hat{U}_{p1}$ = geocentric unit position vector of first planet

$\hat{U}_{p2}$ = geocentric unit position vector of second planet

Time of the equinoxes and solstices of the Sun

This option determines the time of the year when the apparent geocentric longitude of the Sun is an integer multiple of 90 degrees. For this option, we seek the minima of the objective function defined by

$$F = \lambda - \lambda_{\odot} \tag{29}$$

where

λ = 0, 90, 180, 270 degrees

$\lambda_{\odot}$ = apparent geocentric longitude of the Sun

The *four* cases for λ are as follows.

λ =   0 degrees ⇒ spring equinox

λ = 90 degrees ⇒ summer solstice

λ = 180 degrees ⇒ autumn equinox

λ = 270 degrees ⇒ winter solstice

Position of a planet at a user specified epoch

No special objective function is used for this option of program **PLANETS**. This main menu selection is a straightforward calculation of the geocentric and topocentric position of a planet or the Sun.

Program **PLANETS** computes what are called *local* minima or maxima for the time period requested by the user. For any time period, there may be a *global* minima or maxima. The global event time will be the *absolute* minima or maxima of all the objective function event times found during a search period. For example, in the following diagram points A and B are local minima and point C is the global minima. A detailed explanation of the minimization algorithm used in program **PLANETS** is given in the NUMERICAL METHODS column for this issue.

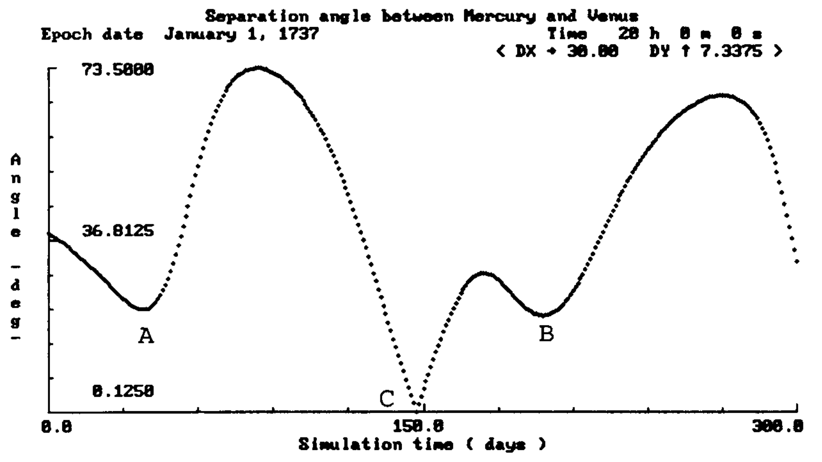


Figure 1 Local and Global Minima

Program Notes

During execution of program **PLANETS**, several menus will appear displaying choices and requesting information from the user. Any item on these menus can be selected in two different ways. The user may scroll through the list of menu items by either pressing the *number* key which corresponds to that item, or by using the *up* and *down* arrow keys on the keyboard. The software will highlight the user's choice, and the item can be input to the program by pressing the *[Enter]* key.

The following one line menu will also appear at several places during the execution of program **PLANETS**.

```
C)ontinue      P)rint      R)estart      Q)uit
```

Any item from this menu is selected by pressing the *first letter* of the selection (C, P, R or Q). Selection *C)ontinue* allows the user to continue the program after a data screen or message has been displayed. If the user chooses *P)rint* screen, the software will print the currently displayed screen to a printer. *R)estart* will start the program again from the beginning, and selection *Q)uit* will exit the program.

Program **PLANETS** will display a *main menu.* The screen menu will appear as follows.

```
1 ) Perihelion and aphelion of a planet
2 ) Greatest elongation or opposition of a planet
3 ) Conjunction in longitude between two planets
4 ) Closest approach distance between two planets
5 ) Minimum angular separation between two planets
6 ) Time of the equinoxes and solstices of the Sun
7 ) Position of a planet at a user specified epoch
```

At this point the user can make a choice and the program will display a short message describing the option selected. You if decide that the menu item you have selected is not what you really want, press *R)estart* to start the program over.

The next display will be a *planetary menu.* If the main menu selection involves two planets, or the Sun and a planet, the planetary menu will be displayed twice. The planet displayed in the third menu item may be either the Earth or the Sun depending on the main menu selection.

After the main menu and planetary menus are accommodated, **PLANETS** will request a calendar date to start the program with the prompt

Please enter the initial calendar date
Month [1 - 12], Day [1 - 31], Year [YYYY])
For example, October 21, 1948 is input as 10,21,1948

The user should respond with three integer numbers separated by commas. The full calendar year should be entered. For example, the calendar year 1987 is input as *1987* not *87*. The number input for the month should be an integer between 1 and 12 and the number input for the day is between 1 and 31. **PLANETS** accounts for the Gregorian reform and will display the following error message if an invalid calendar date is selected.

This date does not exist!!

If this error occurs, the user should press *C)ontinue* and the program will redisplay the calendar date prompt.

For all main menu options except selection **"7) Position of a planet at a user specified epoch"**, the software will prompt the user for the search duration of the simulation in days.

Main menu option 7 will also request the observer's local civil time with

Please input your local civil time
Hours [0 - 24], Minutes [0 - 60], Seconds [0 - 60]
For example, 9:30:45 p.m. is input as 21,30,45

The user should input three positive numbers separated by commas. Please note that civil time is input in *24 hour format*. The number for hours will be an integer between 0 and 24 and the input for minutes and seconds numbers between 0 and 60.

The user's time zone will be requested with this prompt

Please input your time zone
This is an integer number between 0 and 23.
Time zones are positive west of Greenwich.
For example, Eastern Standard Time is time zone 5.

The user response to this request should be an integer number between 0 and 23. Please note that times zones are measured *positive west* of Greenwich. For example, Eastern Standard Time is time zone 5, Central Standard Time is time zone 6, etc.

The program will ask for the status of Daylight Savings Time with

```
                    Daylight Savings Time ?

                      y = yes,    n = no
```

The user should respond with *y* for *yes* if Daylight Savings Time is in effect at his or her location, or *n* for *no* if it is not.

NOTE: If the user inputs 0 for the time zone and *no* for Daylight Savings Time, the time of a planetary event will be displayed in Universal Time (UT).

The program will ask for the observer's geographic latitude with

```
             Please input your geographic latitude
 Degrees [ -90 to +90 ], Minutes [ 0 - 60 ], Seconds [ 0 - 60 ]
    North latitudes are positive, south latitudes are negative
```

The user should respond to this request with three numbers separated by commas. The first number will be an integer between -90 and +90 and the second and third numbers should be positive. South latitudes are negative and north latitudes are positive.

The prompt for the observer's geographic *west* longitude appears as

```
           Please input your geographic west longitude
   Degrees [ 0 - 360 ], Minutes [ 0 - 60 ], Seconds [ 0 - 60 ]
            West longitude equals 360 - east longitude
```

This request requires three positive numbers separated by commas. The west longitude in degrees is an integer between 0 and 360, and the response for minutes and seconds are numbers between 0 and 60. Note that an observer's west longitude is equal to 360 degrees - the east longitude.

After program **PLANETS** has completed the main menu option requested, the software will prompt the user for another selection with

```
                     Another Calculation ?

                      y = yes,     n = no
```

At this point the user should input *y* for *yes* to redisplay the main menu or *n* for *no* to completely exit the program.

If the user responds with *yes*, **PLANETS** will display the prompt

Do you want to change
the observer's location ?

y = yes n = no

A user response of *n* for *no* will tell the software to use the current observer information for the next menu option. This allows the user to calculate a variety of events for the same observer location without reentering the information each time. If the user chooses to keep the current observer information, the observer's latitude, longitude, time zone and Daylight Savings Time status will not change.

Program **PLANETS** will display the following output.

- ▹ Calendar date of the planetary event
- ▹ Local civil time of the planetary event
- ▹ Julian Ephemeris Date (JED) of the planetary event
- ▹ Ephemeris Time (ET) of the planetary event
- ▹ The observer's local sidereal time
- ▹ The planet's topocentric azimuth and elevation
- ▹ The planet's geocentric right ascension and declination
- ▹ The planet's heliocentric and geocentric distances

Program **PLANETS** will print the Julian Ephemeris Date and the Ephemeris Time of the planetary event in hours, minutes and seconds. The observer's local sidereal time in hours, minutes and seconds is also provided. The software will then display the topocentric azimuth and elevation of the selected planet in degrees, minutes and seconds. The geocentric right ascension in hours, minutes and seconds, and the geocentric declination in degrees, minutes and seconds are another part of the screen display. The heliocentric and geocentric distances of the planet at the computed event time are printed in Astronomical Units.

PLANETS will also display additional data which is a function of the actual main menu option selected by the user. For example, the closest approach distance in Astronomical Units will be displayed for menu item **"4) Closest approach distance between two planets"**. For main menu options which involve two planets, the program will calculate and print the data for both planets except when the second planet is the Sun or Earth.

The following are several output screens from program **PLANETS.BAS**. The first example illustrates typical *closest approach conditions* between the Earth and Mars. It also shows the display format of the topocentric, geocentric and heliocentric coordinates.

```
          Closest approach distance between Earth and Mars

September 21, 1988                                 20 h  33 m  23 s

39°  45'  00"  North                          104°  58'  48"  West
Time Zone  7                             Daylight Savings Time = No

Julian Ephemeris Date                                2447426.64921
Ephemeris Time                                      3 h  34 m  52 s
Local sidereal time                                20 h  38 m  21 s

Closest approach distance                         0.39309621  A.U.

                              * Mars *

Topocentric azimuth                                 112°  30'  42"
Topocentric elevation                                22°  18'  11"

Geocentric right ascension                          0 h  33 m  25 s
Geocentric declination                               -1°  42'  02"

Heliocentric distance                             1.39315639  A.U.
Geocentric distance                               0.39309621  A.U.
```

This next example is a screen display of the *minimum angular separation conditions* between Mars and Jupiter.

```
         Minimum angular separation between Mars and Jupiter

June 14, 1991                                       8 h  41 m  32 s

39°  45'  00"  North                          104°  58'  48"  West
Time Zone  7                             Daylight Savings Time = No

Julian Ephemeris Date                                2448422.15491
Ephemeris Time                                     15 h  43 m  04 s
Local sidereal time                                 2 h  11 m  22 s

Separation angle                                      0°  36'  48"
```

FUNDAMENTAL ASTRONOMY

In this edition of **FUNDAMENTAL ASTRONOMY**, we present a QuickBASIC computer program called **CSYSTEMS.BAS** which can be used to understand fundamental astronomical coordinate systems. These systems are used to define the position and motion of celestial bodies. Program **CSYSTEMS** is a *coordinate conversion* utility which can be used to convert astronomical positions from one coordinate system to another.

The coordinate conversion options of program **CSYSTEMS** are

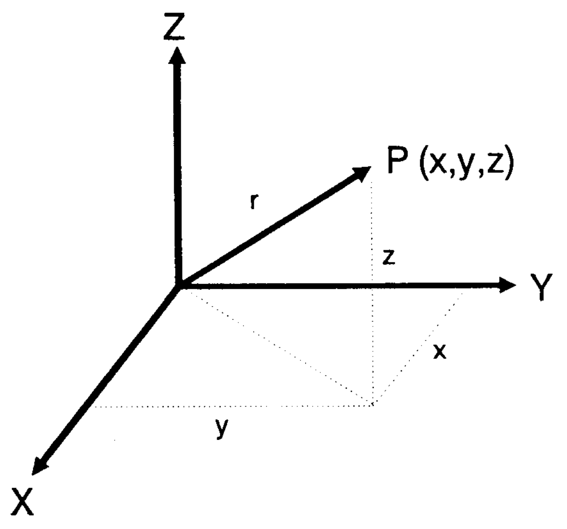
- ▷ Polar to rectangular coordinates
- ▷ Rectangular to polar coordinates
- ▷ Ecliptic to equatorial coordinates
- ▷ Equatorial to ecliptic coordinates
- ▷ Geocentric to topocentric coordinates
- ▷ Topocentric to geocentric coordinates
- ▷ Mean-of-date to True-of-date coordinates
- ▷ True-of-date to Mean-of-date coordinates

Fundamental Concepts

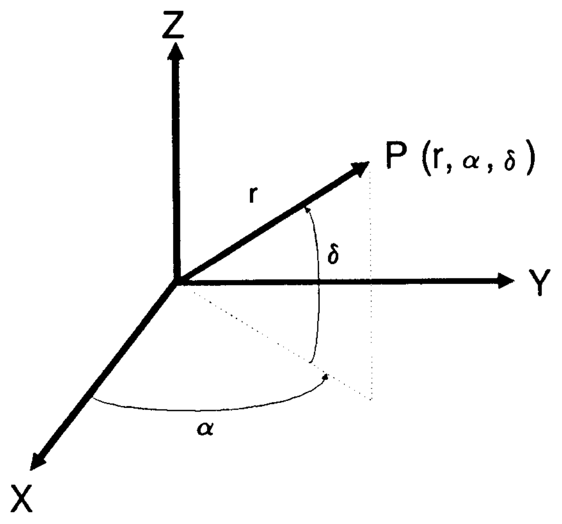
Program **CSYSTEMS** makes use of *vectors* and *matrices* when computing coordinate conversions. A quantity which has both *magnitude and direction* is called a *vector*. A vector can be represented by a *directed line segment*, where the length of this line segment is the magnitude. A matrix is an array of numbers arranged in *rows* and *columns*. A vector is a *column* matrix consisting of m rows and 1 column. The velocity of a planet in its orbit about the Sun is an example of a vector.

Any coordinate system can be specified in terms of its *origin*, the direction of its *principal axis*, and its *fundamental plane*. All of the coordinate systems we are about to describe are called *orthogonal* coordinate systems. For these coordinate systems, the three axes are mutually perpendicular, and meet at the origin.

Program **CSYSTEMS** is concerned with celestial positions which may be defined by two types of coordinates. The first type is called *rectangular* or *cartesian* coordinates, and the other type is called *polar* coordinates. The following diagram illustrates these two types of coordinates.



Rectangular Coordinates



Polar Coordinates

This diagram shows that the rectangular coordinates of a vector *P* consist of x, y, and z components which are the *projections* of *P* onto the three axes. The polar coordinates of vector *P* consist of an angle α measured in the fundamental plane, an angle δ measured above or below this plane, and the *scalar* length or *magnitude* *r* of the vector *P*.

The conversion of a vector $\overline{X}_1$ defined in coordinate system 1 can be converted to a vector $\overline{X}_2$ in another system 2 by a *coordinate transformation* defined as follows.

$$\overline{X}_2 = \left[\, M \,\right] \overline{X}_1 \tag{1}$$

The matrix **M** can be represented by different combinations of three *fundamental* matrices which involve angular rotations about each of the three coordinate axes.

The matrix for an angular rotation about the x-axis of angle ϕ is

$$M_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{2}$$

The matrix for an angular rotation about the y-axis of angle ϕ is

$$M_y(\phi) = \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix} \tag{3}$$

The matrix for an angular rotation about the z-axis of angle ϕ is

$$M_z(\phi) = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

A transformation matrix is usually composed of one or more *matrix multiplications.* Two matrices, **A** and **B**, can be multiplied together if the number of *columns* of matrix **A** is equal to the number of *rows* of matrix **B**. If matrix **A** has *m* rows and *n* columns, and matrix **B** has *n* rows and *p* columns, then the product of these two matrices **C** is given by

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj} \qquad i = 1 \ldots m, \qquad j = 1 \ldots p \tag{5}$$

The vector $\bar{X}_2$ defined by Eq. (1) above can be converted back to vector $\bar{X}_1$ by using the *inverse* of matrix **M** given by

$$\bar{X}_1 = \begin{bmatrix} M \end{bmatrix}^{-1} \bar{X}_2 \tag{6}$$

A very important (and convenient) property of orthogonal matrices is the fact that the inverse of an orthogonal matrix is equal to its *transpose.*

The transpose of an orthogonal matrix can be determined by rearranging the matrix so that the *rows* of the original matrix become the *columns* of the new matrix. For example, to calculate matrix **B** as the transpose of matrix **A**, the indices of the matrix **B** are arranged as follows:

$$\begin{bmatrix} B \end{bmatrix} = \begin{bmatrix} A \end{bmatrix}^{-1} = \begin{bmatrix} A \end{bmatrix}^{T} \Rightarrow \begin{bmatrix} B \end{bmatrix}_{ij} = \begin{bmatrix} A \end{bmatrix}_{ji} \tag{7}$$

Polar and Rectangular Coordinates

The x, y and z components of a rectangular position vector are

$$\bar{R} = R\,\hat{U} = \begin{Bmatrix} R_x \\ R_y \\ R_z \end{Bmatrix} \tag{8}$$

and the magnitude or length *R* of the position vector can be determined from the individual vector components with

$$R = \sqrt{R_x^2 + R_y^2 + R_z^2} \tag{9}$$

A rectangular *unit pointing vector* has a scalar magnitude or length equal to 1 and can be written in several ways as shown by this next expression.

$$\hat{U} = \begin{Bmatrix} U_x \\ U_y \\ U_z \end{Bmatrix} = \begin{Bmatrix} R_x/R \\ R_y/R \\ R_z/R \end{Bmatrix} = \begin{Bmatrix} \cos\beta\cos\lambda \\ \cos\beta\sin\lambda \\ \sin\beta \end{Bmatrix} = U_x\hat{i} + U_y\hat{j} + U_z\hat{k} \tag{10}$$

In the latter part of this expression, $\hat{i}$, $\hat{j}$, and $\hat{k}$ are unit pointing vectors along the x, y, and z axes respectively.

The polar coordinates can be computed from the components of the rectangular unit pointing vector with the next two equations.

$$\delta = \sin^{-1}(U_z) \tag{11a}$$

$$\alpha = \text{ATAN3}\ (U_y, U_x) \tag{11b}$$

Ecliptic and Equatorial Coordinates

In the ecliptic coordinate system, the origin is usually the center of the Sun or *heliocentric*, the fundamental plane is the *ecliptic* plane, and the *X* or principle axis is in the direction of the *Vernal Equinox*. The ecliptic plane is the plane described by the Earth's motion about the Sun. The Vernal Equinox is established at the moment when the *line of intersection* of the ecliptic and equatorial planes passes through the center of the Sun. This line of intersection is called the *line of equinoxes* and passes through the Sun twice a year. The positive direction of the principle occurs during the March equinox, and is also called the *first point of Aries*.

In the equatorial system, the origin is usually the center of the Earth or *geocentric*, the fundamental plane is the Earth's equatorial plane, and the *X* axis is along the Vernal Equinox.

The conversion of ecliptic to equatorial coordinates consists of a matrix rotation about the x-axis through an angle ε, called the *obliquity of the ecliptic*. The obliquity is the angle between the ecliptic plane and the Earth's equatorial plane.

This transformation can be expressed as

$$\bar{R}_{eq} = \left[M_x(-\varepsilon) \right] \bar{R}_{ec} \tag{12}$$

The conversion of equatorial to ecliptic coordinates is

$$\bar{R}_{ec} = \left[M_x(\varepsilon) \right] \bar{R}_{eq} \tag{13}$$

Geocentric and Topocentric Coordinates

In the geocentric coordinate system, the origin is the Earth's center, the fundamental plane is the Earth's equator, and the *X* axis is in the direction of the Vernal Equinox. In the topocentric system, the origin is the observer's geographic location on the Earth's surface, the fundamental plane is a plane *tangent* to the Earth's surface at that location, and the principle axis is along the direction of north.

The geocentric x, y, and z components of the observer's *inertial* position vector are calculated with the following three equations.

$$R_{ox} = C_x \cos\phi \cos\theta \tag{14a}$$

$$R_{oy} = C_y \cos\phi \sin\theta \tag{14b}$$

$$R_{oz} = C_y \sin\phi \tag{14c}$$

where

ϕ = observer geographic latitude $\begin{cases} \text{positive north} \\ \text{negative south} \end{cases}$

θ = observer local sidereal time

The calculation of an observer's local sidereal time was described in the Spring 1989 issue of *CELESTIAL COMPUTING*.

The *geodetic* constants C_x and C_y are functions of the observer's latitude, altitude, and the shape and size of the Earth. They are determined with the next two equations.

$$C_x = \frac{r_{eq}}{\sqrt{1 - (2f - f^2)\sin^2\phi}} + h \tag{15a}$$

$$C_y = \frac{r_{eq}(1 - f^2)}{\sqrt{1 - (2f - f^2)\sin^2\phi}} + h \tag{15b}$$

where

r_{eq} = equatorial radius of the Earth

f = flattening factor of the Earth

$$h = \text{observer's altitude} \begin{cases} \text{positive above sea level} \\ \text{negative below sea level} \end{cases}$$

The matrix which transforms position in the geocentric system to position in the topocentric system is a function of the observer's geographic latitude on the Earth and local sidereal time. The conversion of a geocentric unit vector $\hat{U}_G$ to a topocentric unit position vector $\hat{U}_T$ can be accomplished with the following matrix transformation.

$$\hat{U}_T = \left[\, A \,\right] \hat{U}_G \tag{16}$$

where the 9 components of the 3 by 3 transformation matrix **A** are determined from the following equations.

$$\begin{array}{lll} A_{11} = \sin\phi\cos\theta & A_{12} = \sin\phi\sin\theta & A_{13} = -\cos\theta \\ A_{21} = -\sin\theta & A_{22} = \cos\theta & A_{23} = 0 \\ A_{31} = \cos\phi\cos\theta & A_{32} = \sin\phi\cos\theta & A_{33} = \sin\phi \end{array} \tag{17}$$

Can you derive the individual transformation matrices which define this transformation?

This coordinate transformation also involves a *translation* of the origin from the Earth's center to the observer's location. This translation consists of a vector subtraction given by

$$\bar{R}_T = \bar{R}_G - \bar{R}_O \qquad (18)$$

where

$\bar{R}_T$ = topocentric position vector

$\bar{R}_G$ = geocentric position vector

$\bar{R}_O$ = observer's geocentric position vector

The topocentric unit position vector is calculated by *normalizing* the three components of the position vector.

Mean-of-date and True-of-date Coordinates

The conversion of Mean-of-date coordinates (MOD) to True-of-date coordinates (TOD), and vice versa involves corrections for both *precession* and *nutation*. The *names* of the fundamental plane and principle axis are the same in both systems. However, the mean-of-date system does not include the perturbations of the fundamental plane and principle axis due to both precession and nutation.

The coordinate conversion can be accomplished with the following two matrix transformations.

$$\bar{R}_{TOD} = \left[\,N\,\right]\left[\,P\,\right]\bar{R}_{MOD} \qquad (19)$$

$$\bar{R}_{MOD} = \left[\left[\,N\,\right]\left[\,P\,\right]\right]^T \bar{R}_{TOD} \qquad (20)$$

The precession matrix **P** is computed with the following three matrix multiplications.

$$\left[\,P\,\right] = M_z(-z - 90^\circ)\; M_x(\theta)\; M_z(90^\circ - \zeta) \qquad (21)$$

A numerical method for calculating this matrix was described in the FUNDAMENTAL ASTRONOMY column of the Summer 1989 issue of *CELESTIAL COMPUTING*.

The nutation matrix **N** is also a combination of three matrix multiplications as follows.

$$\left[\, N \, \right] = M_x(-\varepsilon)\; M_z(-\Delta\psi)\; M_x(\bar{\varepsilon}) \qquad (22)$$

where

ε = true obliquity of the ecliptic

$\Delta\psi$ = nutation in obliquity

$\bar{\varepsilon}$ = mean obliquity of the ecliptic

The calculation of the obliquity terms in this matrix was described in the FUNDAMENTAL ASTRONOMY column of the Spring 1989 issue of *CELESTIAL COMPUTING*.

Program Notes

Each coordinate conversion option of program **CSYSTEMS** will allow the user to input either polar or rectangular coordinates. Both of these two options will also request the *distance* of the celestial object.

If the user wants to transform angles *only*, he or she should input the number 0 in response to the distance prompt. In this situation, the software will assume the celestial object is very far away, and transform the pointing angles only. This same effect can be accomplished by inputting the components of a *unit position vector* in response to the software's request for rectangular coordinates.

The following is a short discussion of additional user inputs which are unique to each coordinate conversion option.

Polar to rectangular coordinates

This option will allow the user to input either azimuth and elevation, or right ascension and declination. Please note that azimuth is input in the units of *degrees*, *minutes* and *seconds*, and right ascension should be input in the units of *hours*, *minutes* and *seconds*.

Ecliptic to/from equatorial coordinates

This option will also request the calendar date and Universal Time of the coordinate conversion. The calendar date prompt requires all *four* digits of the calendar year, and the Universal Time should be input in 24 hour format.

Geocentric to topocentric coordinates

This option of program **CSYSTEMS** will also ask for the geographic latitude, west longitude and altitude of the observer. Please note that north latitudes are positive and south latitudes are negative. An observer's west longitude is equal to 360 degrees minus the east longitude. Observer altitudes above sea level are positive, and observer locations below sea level are negative. The observer's altitude must be input in the units of *meters*, and the distance of the celestial object must be input in the units of *kilometers*.

Mean-of-date to/from True-of-date coordinates

This menu selection will also request the calendar dates and Universal Times of both the mean-of-date epoch and the true-of-date epoch.

The following is the **CSYSTEMS** solution to *Example 8.a* in the book, *Astronomical Formulae for Calculators*, page 45.

```
                  Equatorial to ecliptic coordinates

   Calendar date                             January 1, 1950
   Universal time                            0 h  00 m  0.000 s
   Julian Date                               2433282.5

                       Ecliptic Coordinates

   Latitude                                    6°  40'  46.870" North
   Longitude                                 112°  31'  31.186"

   X-component of unit position vector      -.3804913815415064
   Y-component of unit position vector       .9174400744149217
   Z-component of unit position vector       .1163186074116204

                      Equatorial Coordinates

   Right ascension                          7 h  42 m  15.525 s
   Declination                              28°  08'  55.110" North

   X-component of unit position vector      -.3804913815415064
   Y-component of unit position vector       .7954044540011024
   Z-component of unit position vector       .4717605993804951
```

APPLIED ASTRODYNAMICS

In this edition of APPLIED ASTRODYNAMICS, we present a QuickBASIC program called **VSAT.BAS** which can be used to determine visibility conditions of Earth satellites. The software is valid for satellites in both circular and elliptical orbits, and the observer can be located anywhere on an *oblate* Earth. Program **VSAT** also computes Earth shadow conditions during visibility, and provides a graphics display of pointing angles and groundtrack.

Program **VSAT** uses an *analytic* method to propagate a satellite's orbit while searching for *line-of-site visibility* between the satellite and an Earth observer. The method of orbit propagation is described in Chapter 10 of *Methods of Orbit Determination* by P. R. Escobal, and includes the first order effect of the Earth's oblateness on the satellite's motion.

The perturbations of mean anomaly M, right ascension of the ascending node Ω, and argument of perigee ω are calculated as a function of time with the following set of equations

$$M = M_0 + \tilde{n}\, t \qquad (1)$$

$$\Omega = \Omega_0 + \dot{\Omega}\, t \qquad (2)$$

$$\omega = \omega_0 + \dot{\omega}\, t \qquad (3)$$

In these three equations, M_0, Ω_0 and ω_0 are the initial values of the orbital elements.

The Keplerian *mean motion* is described by the equation

$$n_k = \frac{2\pi}{\tau_k} \qquad (4)$$

where τ_k is the Keplerian or *unperturbed* period of the satellite.

The *perturbed* mean motion includes the oblateness effect and is computed from the next equation

$$\tilde{n} = n_k \left[1 + 1.5\, J_2 \frac{\sqrt{1 - e^2}}{p^2} (1 - 1.5 \sin^2 \iota) \right] \qquad (5)$$

The perturbation of the satellite's argument of perigee is

$$\dot{\omega} = \frac{d\omega}{dt} = 1.5\ J_2\ \tilde{n}\ \frac{\sqrt{1 - e^2}}{p^2}\ (2 - 2.5\ \sin^2 \iota) \tag{6}$$

The perturbation of the right ascension of the ascending node is

$$\dot{\Omega} = \frac{d\Omega}{dt} = -1.5\ J_2\ \tilde{n}\ \frac{\sqrt{1 - e^2}}{p^2}\ \cos\ \iota \tag{7}$$

where

t	=	simulation time
a	=	semimajor axis
p	=	semilatus rectum = $a\ (1 - e^2)$
M	=	mean anomaly
ι	=	orbital inclination
Ω	=	right ascension of the ascending node
ω	=	argument of perigee
n	=	Keplerian mean motion
$\tilde{n}$	=	perturbed mean motion

Kepler's equation for an elliptic orbit is defined by

$$M = E - e\ \sin\ E \tag{8}$$

This is a *transcendental* equation which is solved iteratively in program **VSAT** for the value of *eccentric anomaly* E. The true anomaly of the satellite can be calculated with

$$\nu = \text{ATAN3}\left[\sqrt{1 - e^2}\ \sin\ E,\ \cos\ E - e\right] \tag{9}$$

As the satellite's orbit is propagated forward in time, program **VSAT** evaluates a *visibility function*. Whenever the value of this visibility function is *positive*, the satellite is visible to the Earth observer. The derivation of the visibility function is described in Chapter 5 of *Methods of Orbit Determination* by P.R. Escobal. This visibility function is defined by the following expression

$$F = a\left[(\cos\ E - e)\ \hat{P}\cdot\hat{Z} + (\sqrt{1 - e^2}\ \sin\ E)\ \hat{Q}\cdot\hat{Z}\right] - G \tag{10}$$

where

a = semimajor axis

E = eccentric anomaly

e = orbital eccentricity

ϕ = observer's geographic latitude

θ = observer's local sidereal time

J_2 = oblate Earth gravity coefficient

$G = C_x \cos^2\phi + C_y \sin^2\phi$

$P_x = \cos\omega \cos\Omega - \sin\omega \sin\Omega \cos\iota$

$P_y = \cos\omega \sin\Omega + \sin\omega \cos\Omega \cos\iota$

$P_z = \sin\omega \sin\iota$

$Q_x = -\sin\omega \cos\Omega - \cos\omega \sin\Omega \cos\iota$

$Q_y = -\sin\omega \sin\Omega + \cos\omega \cos\Omega \cos\iota$

$Q_z = \cos\omega \sin\iota$

$Z_x = \cos\phi \cos\theta$

$Z_y = \cos\phi \sin\theta$

$Z_z = \sin\phi$

The vectors **P**, **Q** and **Z** are called *orbit plane* unit vectors. The *geodetic* constants C_x and C_y are defined in the FUNDAMENTAL ASTRONOMY section of this issue.

Program **VSAT** also determines if a satellite is within the umbra or penumbra portion of the Earth's shadow. The best observing conditions occur when the observer is in darkness, the satellite is visible, and has not entered the Earth's shadow. The software first calculates an *umbra* and *penumbra* angle with the two equations given by

$$\psi_u = \eta - \theta_u \Rightarrow \text{umbra} \qquad (11a)$$

$$\psi_p = \eta + \theta_p \Rightarrow \text{penumbra} \qquad (11b)$$

where

$$\eta = \sin^{-1}\left[\frac{R_e}{R_{sc}}\right]$$

R_e = radius of the Earth

R_{sc} = geocentric distance of the satellite

In these equations, η is the size of a *cylindrically* shaped shadow.

The size of the umbra portion of the shadow at the satellite's distance is calculated from the following equation

$$\theta_u = \sin^{-1}\left[\frac{R_s - R_e}{R_{es}}\right] \tag{12}$$

where

R_s = radius of the Sun

R_{es} = distance from the Earth to the Sun

The size of the penumbra portion of the shadow is calculated from

$$\theta_p = \sin^{-1}\left[\frac{R_s + R_e}{R_{es}}\right] \tag{13}$$

The program then evaluates a *shadow parameter* defined by the equation

$$\varphi = -\frac{|\bar{R}_{sc} \times \bar{R}_{es}|}{R_{es}} \operatorname{sign}(\bar{R}_{sc} \cdot \bar{R}_{es}) \tag{14}$$

We can define a *critical value* of the shadow parameter for both the umbra and penumbra portions of the Earth's shadow. These are *geometric conditions* given by

$$\varphi_p = |\bar{R}_{sc}| \sin \psi_p \tag{15a}$$

$$\varphi_u = |\bar{R}_{sc}| \sin \psi_u \tag{15b}$$

The computer algorithm compares the value of the shadow parameter with these two critical values to determine shadow conditions. If the condition $\varphi_u < \varphi \le \varphi_p$ is satisfied, the satellite is in the penumbra portion of the Earth's shadow.

If the condition $0 \le \varphi \le \varphi_u$ is true, the satellite is in the umbra portion of the Earth's shadow. If the *absolute value* of the calculated shadow parameter is larger than the penumbra shadow value, the satellite is not in the Earth's shadow.

Program Notes

Program **VSAT** will request the following inputs from the user.

- ▹ The observer's geographic latitude
- ▹ The observer's geographic west longitude
- ▹ The observer's altitude above or below sea level
- ▹ The observer's time zone
- ▹ The status of Daylight Savings Time (yes or no)
- ▹ The orbital inclination of the satellite's orbit
- ▹ The satellite's orbital period
- ▹ The eccentricity of the satellite's orbit
- ▹ The satellite's argument of perigee
- ▹ The calendar date of a reference equatorial crossing
- ▹ The GMT of a reference equatorial crossing
- ▹ The west longitude of a reference equatorial crossing
- ▹ A visibility *search* step size
- ▹ A visibility *print* step size

Program **VSAT** will provide the following program output.

- ▹ The observer's local civil time during visibility
- ▹ Azimuth, elevation and slant range to the satellite
- ▹ Topocentric right ascension and declination
- ▹ Shadow conditions (umbra or penumbra)
- ▹ Azimuth and elevation graphics
- ▹ Right ascension and declination graphics
- ▹ Satellite groundtrack graphics

NASA Prediction Bulletins

These bulletins contain information about the classical orbital elements and reference equatorial crossings of satellites. NASA prediction bulletins are issued periodically as the satellite orbit changes, and are available free from the following address.

NASA Goddard Space Flight Center
Project Operations Branch
Code 513
Greenbelt, MD 20771

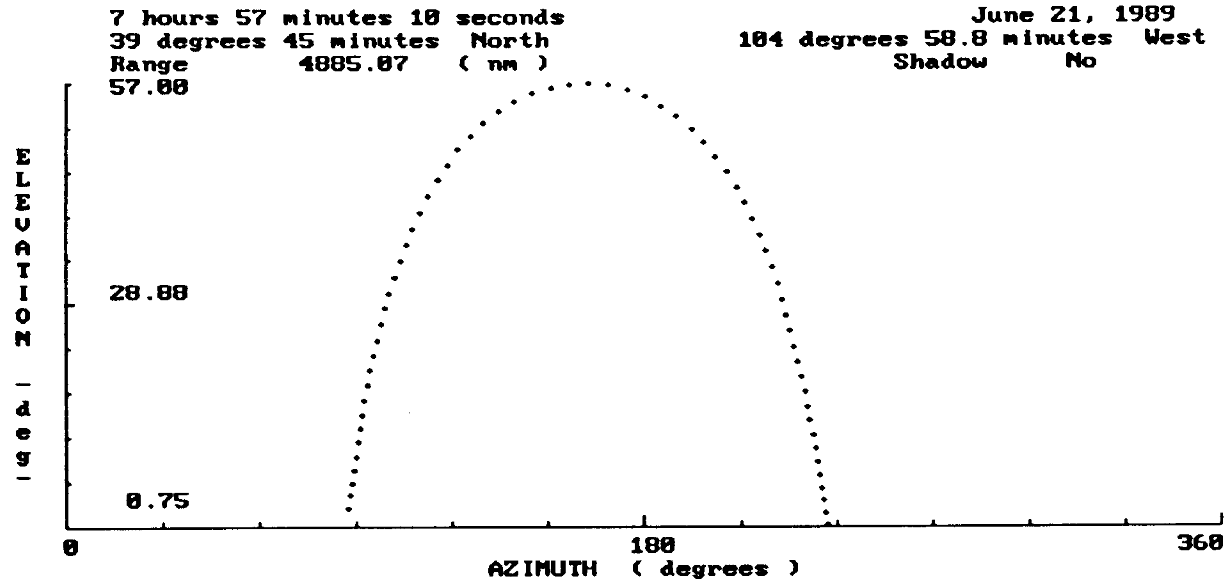
The bulletins are organized in a cross reference system which relates a NASA catalog number to the international designation of a satellite. The following is a list of several satellites.

Name	Catalog Number	International Designation
Mir	16609	86 17A
Salyut 7	13138	82 33A
Seasat	10967	78 64A

When requesting prediction bulletins from NASA, be sure to include the catalog number of the satellite(s) of interest.

Program **VSAT** also requires the orbital period of a satellite. The orbital period (in minutes) can be determined by dividing 1440 minutes by the number of orbits per day printed on the bulletin.

The following is a typical graphics display from program **VSAT**.



SYMBOLIC COMPUTING

In this session of SYMBOLIC COMPUTING, we present Eureka: The Solver and MathCAD equation files which can be used to determine the closest approach conditions between an observer on an oblate Earth and an Earth satellite. Let's discuss the MathCAD version of the equation file.

This report describes a method for finding closest approach conditions between an Earth satellite and a ground site. The analysis is valid for Earth satellites in either circular or elliptical orbits and the Earth is modelled as an *oblate spheroid*. Results from this analysis include the orbital true anomaly and slant range at closest approach.

All equations in this report are expressed in an *earth-centered-fixed* or ECF coordinate system. This system is centered at the Earth with the x-axis aligned with Greenwich, the z-axis aligned with the Earth's spin axis and the y-axis completes the right-handed orthogonal system.

We begin by calculating a utility constant which will be used to convert angles from degrees to radians.

$$dtr := \frac{\pi}{180}$$

The *equatorial radius* of the oblate Earth model is

$a_e := 3443.923$ (nautical miles)

and the *flattening factor* of this model is

$f := \frac{1}{298.257}$ (non-dimensional)

Next we define the *classical* orbital elements of a typical Earth satellite.

The *semimajor axis* in nautical miles is

$$a := 4143.923$$

and the *orbital eccentricity* is

$e := .0085$ (non-dimensional)

The *orbital inclination* is specified in degrees as

$i := 63.435$

and converted to radians with

$i := i \cdot dtr$

Next we input the *argument of perigee*

$\omega := 165.25$ (degrees)

$\omega := \omega \cdot dtr$ (radians)

and the *east longitude of the ascending node*

$\Omega := 125.989$ (degrees)

$\Omega := \Omega \cdot dtr$ (radians)

The *geodetic latitude* of the ground site is

$\phi := 39.75$ (degrees)

$\phi := \phi \cdot dtr$ (radians)

North latitudes are positive and south latitudes are negative.

The *east longitude* of the ground site is

$\lambda := 255.02$ (degrees)

$\lambda := \lambda \cdot dtr$ (radians)

The *ground site altitude* is initially input in feet and then converted to nautical miles. Site altitudes are positive above sea level and negative below sea level.

$H := 5280$ (feet)

$H := \frac{H}{6076.115486}$ (nautical miles)

The components of the ground site ECF position vector (r) can be calculated with the following set of equations.

$$G_1 = \frac{a_e}{\sqrt{1 - (2\,f - f^2)\,\sin^2\phi}} + H$$

$$G_2 = \frac{1 - f^2}{\sqrt{1 - (2\,f - f^2)\,\sin^2\phi}} + H$$

$$r := \begin{bmatrix} G_1 \cos\phi \cos\lambda \\ G_1 \cos\phi \sin\lambda \\ G_2 \sin\phi \end{bmatrix}$$

First we would like to calculate and plot the behavior of the *slant range* as a function of orbital true anomaly.

Let's define "j" as the true anomaly range variable.

$j := 0\,..72$ (degrees)

The *orbital true anomaly* in five degree steps is

$\theta_j := 5 \cdot j \cdot \mathrm{dtr}$ (radians)

and the *argument of latitude* in radians is

$$\sigma := \overline{(\omega + \theta)}$$

Here we use the MathCAD "vec" operator to calculate σ and other array elements elements we need.

The satellite *position magnitude* can be calculated from

$$rm := \overline{\left[a\,\frac{1 - e^2}{1 + e\,\cos(\theta)} \right]}$$

The x, y and z components of the satellite ECF position vector are computed with the following three equations.

$$rx := \overline{(rm \cdot (\cos(\Omega) \cdot \cos(\sigma) - \sin(\Omega) \cdot \cos(i) \cdot \sin(\sigma)))}$$

$$ry := \overline{(rm\cdot(\sin(\Omega)\cdot\cos(\ ) + \cos(\Omega)\cdot\cos(i)\cdot\sin(\sigma)))}$$

$$rz := \overline{(rm\cdot\sin(i)\cdot\sin(\sigma))}$$

The slant range from the ground site to the spacecraft is the magnitude of the vector difference of the two position vectors.

$$range := \overline{\left[\sqrt{(rx - r_0)^2 + (ry - r_1)^2 + (rz - r_2)^2}\right]}$$

The minimum and maximum slant ranges can be found using the built-in array functions of MathCAD. This information also helps us determine the proper y-axis scaling for plotting.

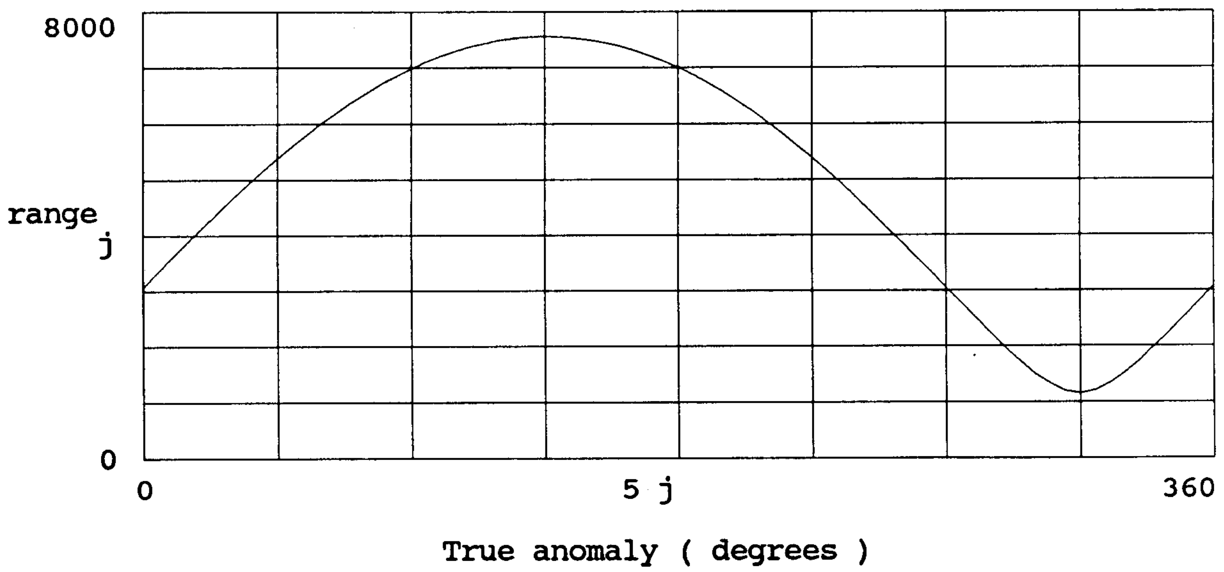
$rmax := max(range)$

$rmin := min(range)$

$rmax = 7.559\cdot10^3$ (nautical miles)

$rmin = 1.157\cdot10^3$ (nautical miles)

The following is a plot of ground site-to-spacecraft slant range in nautical miles as a function of true anomaly in the orbit.



Now we can calculate the true anomaly at closest approach using the built-in derivative and root functions of MathCAD.

First we need to define the satellite position vector in terms of the orbital true anomaly with the following equations.

$$rm(\theta) := a \cdot \frac{1 - e^2}{1 + e \cdot \cos(\theta)}$$

$$rx(\theta) := rm(\theta) \cdot (\cos(\Omega) \cdot \cos(\omega + \theta) - \sin(\Omega) \cdot \cos(i) \cdot \sin(\omega + \theta))$$

$$ry(\theta) := rm(\theta) \cdot (\sin(\Omega) \cdot \cos(\omega + \theta) + \cos(\Omega) \cdot \cos(i) \cdot \sin(\omega + \theta))$$

$$rz(\theta) := rm(\theta) \cdot \sin(i) \cdot \sin(\omega + \theta)$$

The relationship between true anomaly and slant range is given by

$$range(\theta) := \sqrt{(rx(\theta) - r_0)^2 + (ry(\theta) - r_1)^2 + (rz(\theta) - r_2)^2}$$

From the plot of slant range versus true anomaly, we can provide the MathCAD *root finder* with an initial guess for true anomaly at closest approach.

$\theta := 315$ (degrees)

$\theta := \theta \cdot dtr$ (radians)

The true anomaly at closest approach is the point where the derivative of slant range is zero (*a local minima*) near our initial guess.

$$\theta_{ca} := root\left(\frac{d}{d\theta} range(\theta), \theta \right)$$

Finally, the true anomaly at closest approach in degrees is

$$\theta_{ca} := \theta_{ca} \cdot \frac{\pi}{180}$$

$\theta_{ca} = 314.818$ (degrees)

and the slant range from the ground site to the satellite is

$$range := range(\theta_{ca} \cdot dtr)$$

$range = 1.157 \cdot 10^3$ (nautical miles)

This problem can also be solved with the use of the MathCAD *Solve blocks* capability. First we need to give the algorithm an *initial guess* for the true anomaly at closest approach.

$\theta := \pi$ (radians)

Next we need to define the fundamental equations and all constraints within the *Given/Find* construct of a MathCAD Solve block.

Given

$$\mathrm{range}(\theta) := \sqrt{(rx(\theta) - r_0)^2 + (ry(\theta) - r_1)^2 + (rz(\theta) - r_2)^2}$$

The fundamental equation is the derivative of the slant range which we will call *f(θ)*.

$$f(\theta) := \frac{d}{d\theta}\,\mathrm{range}(\theta)$$

We *constrain* the solution to the minimum by requiring that the *second derivative* of the range function f(θ) be *positive.*

$$\frac{d}{d\theta}\,f(\theta) > 0$$

We also require that the solution be within the true anomaly range of 0 to 360 degrees with the next two constraints.

$\theta \ge 0$ (radians)

$\theta \le 2\cdot\pi$ (radians)

Finally, we *Find* the value of orbital true anomaly which satisfies the requirement that the derivative of range is zero.

$f(\theta) \approx 0$

$\theta_{ca} := \mathrm{Find}(\theta)$

$\theta_{ca} := \theta_{ca}\,\dfrac{180}{\pi}$

$\theta_{ca} = 314.818$ (degrees)

RECREATIONAL COMPUTING

In this new column of *CELESTIAL COMPUTING*, we will present computer applications which are both fun and entertaining. Many of these programs will emphasis graphics to help the user visualize different types of astronomical concepts and celestial motions. Professor Danby has graciously given his permission to use several of the companion computer programs to his book, *Fundamentals of Celestial Mechanics*.

Our first RECREATIONAL COMPUTING article is a QuickBASIC program called **GALILEAN.BAS** which can be used to graphically display the position of the Galilean satellites relative to Jupiter. The software supports IBM-PC and true compatible computers with CGA, EGA and Hercules graphic displays. The program will also provide the numerical data used to generate the graphics. The computer algorithm is based on Chapter 36 of *Astronomical Formulae for Calculators* by Jean Meeus.

GALILEAN.BAS will request the following information from the user.

- ▹ the calendar date
- ▹ the observer's local civil time
- ▹ the observer's time zone
- ▹ the status of Daylight Savings Time

The *Galilean Menu* will prompt the user with the following options.

- ▹ Display data
- ▹ Display graphics
- ▹ Continue to next day
- ▹ End this session

The *Display data* option provides the following information.

- ▹ x-position of each satellite (units of Jupiter radii)
- ▹ y-position of each satellite (units of Jupiter radii)
- ▹ position angle of each satellite (relative to inferior conjunction with Jupiter)
- ▹ semidiameter of Jupiter (arc seconds)
- ▹ Julian Date of observation

NUMERICAL METHODS

In this session of NUMERICAL METHODS, we present one of the most powerful and useful numerical methods available for solving problems in Celestial Mechanics. Many important astronomical events can be predicted if we know when certain *geometric functions* reach a *minimum* or *maximum* value. Function minimization or *optimization* is the topic of this edition of NUMERICAL METHODS.

We will present two algorithms which can be used together to solve optimization problems. There are many celestial problems which can be formulated and solved as minimization problems, and we will feature them in several future issues of *CELESTIAL COMPUTING*. Representative problems and computer programs are as follows:

- ▹ The prediction of lunar occultations
- ▹ Calculating the circumstances of solar eclipses
- ▹ Predicting visibility of planetary features
- ▹ Closest approach conditions of comets and minor planets

DEMOMINI.BAS - demo program for subroutine MINIMA.BAS

This program demonstrates a procedure for calling the subroutine **BMINIMA**, which first brackets the function minima or maxima, and the subroutine **MINIMA** which then solves for a minima or maxima of a *scalar* function of one variable.

The demo program illustrates how we can use these two algorithms to find the time of apogee and perigee of the Moon. For this problem, we are searching for the values of minimum geocentric distance (perigee) and maximum geocentric distance (apogee). The Moon's geocentric distance is calculated with the algorithm described in *Low-Precision Formulae for Planetary Positions*.

Let's first describe the two subroutines, and then we will discuss the programming logic used to solve this particular problem.

<u>Syntax</u>

CALL BMINIMA(AX, BX, CX)

Where

AX, BX, CX = bracketing *triplet* of X values (input as the initial guess and output as bracketing triplet)

Comments

It is a good idea to determine an interval which you *know* contains a minima or maxima before actually beginning the search for the function minima or maxima. This technique is called *bracketing*, and involves finding three values of **x** (**a**, **b** and **c**) such that

$a < b < c$ <u>and</u> $f(b) < f(a)$ <u>and</u> $f(b) < f(c)$

If the bracketing triplet satisfies these three conditions, the function has a minimum in the interval (a,c).

Only values for **AX** and **BX** are actually needed in the call to **BMINIMA**. This subroutine will calculate an initial value for **CX** with the equation

CX = BX + GOLD * (BX - AX)

where the constant **GOLD** is called the *golden mean*.

Syntax

CALL MINIMA(AX, BX, CX, TOL, XMIN, FMIN)

Where

AX, BX, CX	= bracketing triplet of X values
TOL	= convergence tolerance
XMIN	= minimum (or maximum) X value
FMIN	= minimum (or maximum) function value

Comments

This algorithm requires a bracketing triplet which contains the function minima or maxima. This method is based on Brent's technique and no derivative calculations are required.

The user must provide an *objective function* for which the minima or maxima is desired. This function is defined in a QuickBASIC subroutine called **OFUNCTION** which is coded as

SUB OFUNCTION(X, FX)

In the subroutine parameter list, X is the function argument and FX is the negative <u>or</u> positive value of the function evaluated at X. The positive value should be returned if the algorithm is used to find a *minima*, and the negative function value returned when searching for a function *maximum*. The subroutine MINIMA provides all interaction with this subroutine.

Let's illustrate the computer code which actually *cycles* the search and minimizer procedure. The search subroutine is called from within the WHILE/WEND construct shown below. **EPOCH.TIME** is the current simulation time, **JD0** is the initial Julian Date, and **NDAYS** is the total number of days to search for lunar perigee and apogee conditions. The variable **JDATE** is set to the value -999 before entering the WHILE/WEND construct, and at the beginning of subroutine **SEARCH**.

If a valid minima or maxima is found, the corresponding time is returned as the Julian Date of perigee or apogee. The current epoch time is then set to this value plus one day in line [3]. If a valid minima or maxima is not found, the epoch time is incremented by one day in line [5].

```
[1]       WHILE EPOCH.TIME < JD0 + NDAYS
[2]          IF JDATE >=0# THEN
[3]             EPOCH.TIME = JDATE + 1#
[4]          ELSE
[5]             EPOCH.TIME = EPOCH.TIME + 1#
[6]          END IF
[7]          CALL SEARCH(TOL)
[8]       WEND
```

Within subroutine **SEARCH**, the processing which occurs is shown below. The initial bracketing triplet is defined by **T1** and **T2** in lines [1] and [2]. The minima (or maxima) is bracketed by the call to **BMINIMA** in line [3]. Note that this subroutine always looks in the *downhill* direction when bracketing a minima or maxima. It may actually bracket a minima which is *prior* to the current epoch time. If this occurs, the triplet is ignored in line [4] and we exit from the subroutine (with JDATE = -999).

The minima is calculated by the call to subroutine **MINIMA** in line [5]. If the time of minima **TX** is prior to the current epoch time, the subroutine is exited and the search continues. If the minima is valid, the Julian Date of apogee or perigee is returned in the variable **JDATE** and the geocentric distance is returned in **RMOON**.

```
[1]       T1 = EPOCH.TIME
[2]       T2 = EPOCH.TIME + .5#
[3]       CALL BMINIMA(T1, T2, T3)
[4]       IF (T1 < EPOCH.TIME AND T2 < EPOCH.TIME AND
          AND T3 < EPOCH.TIME THEN EXIT SUB
```

```
[5]       CALL MINIMA(T1, T2, T3, TOL, TX, FX)
[6]       IF TX < EPOCH.TIME THEN EXIT SUB
[7]       RMOON = FX
[8]       JDATE = TX
```

This programming method may be used to solve a variety of problems. However, the user should be aware of several important things when formulating a minimization problem.

- ▹ The step size used in both the search procedure and the bracketing triplet is a function of the *behavior* of the astronomical event. For events which occur frequently, small time increments are necessary, while events which occur less frequently can use larger increments. For example, the demo program computes events which occur twice a month and uses a step size of one day. The **PLANETS** program described in the FEATURE ARTICLE uses time increments of 30 days when searching for perihelion or aphelion of the outer planets, and a time step of 3 days for the inner planets.
- ▹ The time increment for the bracketing triplet should be approximately equal to half the search time increment.
- ▹ The convergence tolerance specified for the minimization subroutine should be about 1D-6 to 1D-10. Please note that smaller tolerances will require longer execution times.
- ▹ Finally, experiment! Try different time increments, etc.

Program Notes

Program **DEMOMINI** will request an initial calendar date and search interval. The lunar ephemeris used in **DEMOMINI** is valid for time periods between about 1680 to 2280 A.D. Be sure to keep the combination of initial calendar date and search interval within these dates.

The demo program will output the following information.

- ▹ Calendar date of apogee or perigee
- ▹ Universal time of apogee or perigee
- ▹ Geocentric distance (kilometers)
- ▹ Julian Date of apogee or perigee

CELESTIAL BOOK REVIEW

Publications of the U.S. Naval Observatory, 34th and Massachusetts Avenue, NW, Washington, DC 20392.

The many publications of the U.S. Naval Observatory (USNO) provide a unique, authoritative and inexpensive source of astronomical information. In this review we will briefly discuss several of these publications.

▹ *The Astronomical Almanac* - this publication is the successor to the *American Ephemeris and Nautical Almanac.* It is issued annually and contains precise ephemerides of the Sun, Moon, planets and their satellites. It also contains data for lunar and solar eclipses and other astronomical phenomena.

▹ *Explanatory Supplement* - this publication contains the physical and mathematical methods used to generate the information provided in the *Astronomical Almanac.* The *Supplement* is no longer being printed, but is available in college and technical libraries. It is planned to print a new version in the future.

▹ *The Nautical Almanac* - this publication is issued annually and contains the astronomical data required for *marine navigation.*

▹ *The Air Almanac* - this publication is also issued annually and contains the astronomical information required for *air navigation.*

▹ *Astronomical Phenomena* - this publication is a *preprint* of the data contained in the *Astronomical Almanac.* It is issued annually and contains such information as the calendar, phases of the Moon, visibility and configurations of the planets, eclipses, and rising and setting information for the Sun and Moon.

▹ *The Ephemeris* - this publication contains information used for surveying; ephemerides of the Sun, Polaris, and selected stars.

▹ *Planetary and Lunar Coordinates* - this publication contains low-precision ephemerides of the Sun, Moon and planets.

▹ *Astronomical Papers* - these unique papers contain fundamental astronomical data, observations, and theories which supplement the information provided in the *Astronomical Almanac.* They can be ordered by *Volume* number and *Part* number. Among these many papers are the classic works of Dirk Brouwer, G. M. Clemence and Wallace J. Eckert.

▹ *U.S. Naval Observatory Circulars* - these publications are issued periodically and contain fundamental astronomical data about solar and lunar eclipses, phases of the Moon, fundamental coordinate frames, etc. Many USNO circulars are free.

▹ *Tables of Moonrise and Moonset* - these are numeric tables of the times of Moonrise and Moonset for a particular geographic location. They are valid for one calendar year.

▹ *Sunrise and Sunset Tables* - these are numeric tables for the times of Sunrise and Sunset for selected cities in the continental United States, Alaska and Hawaii.

▹ *Sky with Ocean Joined* - this publication is a reprint of the proceedings of the Sesquicentennial Symposia of the U. S. Naval Observatory.

The publications of the Naval Observatory also include several computer programs.

▹ *The Almanac for Computers* - this annual publication contains ephemeris data in a compact form for use with calculators and small computers. This publication contains both low precision and full precision astronomical tables. It also includes stellar tables for 176 stars. Sections D and E of this publication are also available on magnetic media.

▹ *Floppy Almanac* - this is a computer implementation of the *Astronomical Almanac*. It is issued annually and is available for IBM-PC's, DEC VAX and MicroVAX II, and IBM 370, 43xx and 30xx mainframe computers. The main features are:

(1) Positions

(2) Physical Ephemeris of a Planet

(3) Sidereal Times

(4) Rise, Set and Transit Times

(5) Navigation

(6) Daily Configuration

(7) Topocentric Altitude and Azimuth

▹ *The Interactive Computer Ephemeris* - this computer program provides the same information as the *Floppy Almanac* for the time period from December 21, 1800 to June 7, 2049. This computer program was reviewed in the CELESTIAL SOFTWARE REVIEW column of the Summer 1989 issue of *CELESTIAL COMPUTING*.

CELESTIAL SOFTWARE REVIEW

Program *XonVu*, XonTech, Inc., 6862 Hayvenhurst Avenue, Van Nuys, CA 91406, $79. IBM-PC and true compatible computers, 5 1/4" floppy disk. CGA, EGA, or Hercules graphics. Version 1.0.

XonVu is an interactive computer program which will allow the user to graphically simulate the interplanetary missions of the Voyager and Giotto spacecraft. Seven *mission control files* are included which can simulate the following encounters:

- ▷ Voyager 1 at Jupiter and Saturn
- ▷ Voyager 2 at Jupiter, Saturn, Uranus and Neptune
- ▷ Giotto at Halley's Comet

The outer planets and their moons and rings are represented by *wireframe* graphics. The program will also draw the nucleus and tail of Halley's Comet, stars, the Sun and the spacecraft. This program was initially developed at the Jet Propulsion Laboratory (JPL) over a five year period from 1982 to 1987 for interplanetary mission analysis.

The software provides seven major areas of capability as follows:

(1) Program configuration

(2) Field of view

(3) Scene complexity

(4) Observer

(5) Pointing

(6) Time and mission sequencing

(7) Animation

The following is a brief discussion about each of these features.

Program Configuration

Program XonVu allows the user to control and adjust such program characteristics as screen dimensions or *aspect*, the sensitivity of control keys, and the parameter status summary. The status summary includes such things as star magnitude limit, field orientation, etc. A Help function is also available.

Field of View

The observer's field of view can be adjusted between 50 and .000001 degrees. The user can select keys for zooming in and out, rotate the field of view, and cycle through a set of field sizes. The program will let the user examine the current field of view and optionally input any field size.

Scene Complexity

The scene complexity keys allow the user to toggle body grids and landmarks on and off, input a star magnitude limit for displaying stars, or remove all stars. There are also 5 levels of resolution for drawing bodies. The software will run slower as the scene complexity is increased.

Observer

The origin of the observer may be at several different locations. For example, the observer may be on the spacecraft, in the vicinity of the Earth, or external to the planet or comet. One may also observe from the north pole or equator of the central body, and change the distance from the body. The user may also specify a central body latitude, longitude, and distance.

Pointing

The program will allow the user to point at the central body or at one of its moons. The user may also point at a specific right ascension and declination in space. The pointing direction may also be offset to the left, right, or up and down.

Time and Mission Sequencing

The time control keys allow the user to control the speed of the graphics simulation. The user may also input a specific calendar date for the simulation, and toggle the erasure of previous scenes on and off.

Animation

Animation can be achieved with XonVu by simplifying the graphics scene and experimenting with such things as the time and erasure features. A math coprocessor will make animation more effective. A neat animation to observe is the *gravity assist* effect of the outer planet on the trajectory of the Voyager spacecraft. Star and moon occultations by the outer planets can also be animated.

SUBSCRIPTION AND DISK INFORMATION

CELESTIAL COMPUTING is published four times per year, around the time of the solstices and equinoxes. Both journal and floppy disk subscriptions for the IBM-PC and compatibles are available.

The computer programs are available on both 5 1/4", 360K capacity floppy disks and 3 1/2", 720K capacity floppy disks. Please be sure to specify the format when ordering disk subscriptions. Both QuickBASIC *source* code and *executable* programs are provided on the disks. The Microsoft QuickBASIC compiler is not required to run these programs.

Please submit payment in the form of a personal check or money order, in U.S. dollars, payable to *Science Software*. The costs to countries other than the United States are shown in parentheses.

Send all orders and correspondence to

Science Software
7370 S. Jay Street
Littleton, CO 80123-4661

CELESTIAL COMPUTING ORDER FORM

☐ *CELESTIAL COMPUTING* journal subscription
(1 year, 4 issues) $24.95 ($34.95)

☐ *CELESTIAL COMPUTING* disk subscription
(1 year, 4 disks) $14.95 ($19.95)

☐ *CELESTIAL COMPUTING* back issues
(journal $6.95 ($8.95) disk $4.95 ($6.95))
indicate issue(s) by volume and number ________________

Please specify the disk format

☐ 5 1/4", 360K ☐ 3 1/2", 720K

Name ______________________________________

Street address ______________________________

City, state, zip ____________________________

DISK INFORMATION

The *CELESTIAL COMPUTING* floppy disk contains QuickBASIC source code and executable files, Eureka source and report files, and MathCAD equation files. The size of the individual programs for this issue made it necessary to provide two 5 1/4" floppy disks. The source files are on one disk and executable files are on the other diskette. The disks for this issue also contain an updated version of program **ECLIPSE** which appeared in the Spring 1989 issue of *CELESTIAL COMPUTING*.

The following is a directory listing of the files on the disk for this issue.

```
BMINIMA  BAS    2206   7-12-89    PLANETS  BAS   68557    8-13-89
CATALOG  TXT   26839   6-10-89    README   BAT      19   10-05-88
CSYSTEMS BAS   42183   8-22-89    RTS2SAT  EKA    3266    6-11-89
DEMOMINI BAS   13699   8-10-89    RTS2SAT  MCD    9324    4-09-88
ECLIPSE  BAS   35867   8-04-89    RTS2SAT  RPT    5360    6-11-89
GALILEAN BAS   11872   7-27-89    SCAN     EXE   44996    7-16-89
MINIMA   BAS    2760   7-12-89    VSAT     BAS   29490    8-15-89

BRUN45   EXE   77440   9-28-88    GALILEAN EXE   13830    7-27-89
CSYSTEMS EXE   34524   8-22-89    PLANETS  EXE   67482    8-13-89
DEMOMINI EXE   13288   8-10-89    QBHERC   COM    6749    9-28-88
ECLIPSE  EXE   35870   8-04-89    VSAT     EXE   28248    8-15-89
```

The floppy disk also contains the following files.

- ▷ **BRUN45.EXE** - this is the *run-time* program required for all QuickBASIC executable programs. When copying *CELESTIAL COMPUTING* QuickBASIC executable programs to other floppy and hard disks, be sure to copy this file also.

- ▷ **README.BAT** - this batch file allows the user to view the Science Software disk catalog. From the DOS command line, type README.

- ▷ **CATALOG.TXT** - this is an ASCII text file of the Science Software catalog.

- ▷ **SCAN.EXE** - this program allows the user to scan any ASCII file by using the cursor arrow keys and the PgUp and PgDn keys of the keyboard. To use this program, type SCAN *filename* from the DOS command line.

- **QBHERC.COM** - this file is required when using Hercules compatible graphics boards with QuickBASIC programs which display graphics. This program must be run before executing any graphics program. It can be invoked from the DOS command line by typing QBHERC.

IBM and **MS-DOS** are registered trademarks of the IBM Corporation.

Microsoft and **QuickBASIC** are registered trademarks of Microsoft Corporation.

Eureka: The Solver is a registered trademarks of Borland International, Inc.

MathCAD is a registered trademark of MathSoft, Inc.

Hercules is a registered trademark of Hercules Computer Technology

COMING NEXT ISSUE

- **Feature Article**

  The Stumpff/Weiss Solution of the Four-Body Problem

- **Fundamental Astronomy**

  Computing an Accurate Position of the Earth

- **Applied Astrodynamics**

  Calculating Mutual Visibility Between Two Earth Satellites

- **Symbolic Computing**

  Symbolic Computing Solutions of Kepler's Equation

- **Recreational Computing**

  A Computer Graphics Simulation of Three-Body Motion

- **Numerical Methods**

  Computer Programs for Solving Non-linear Equations