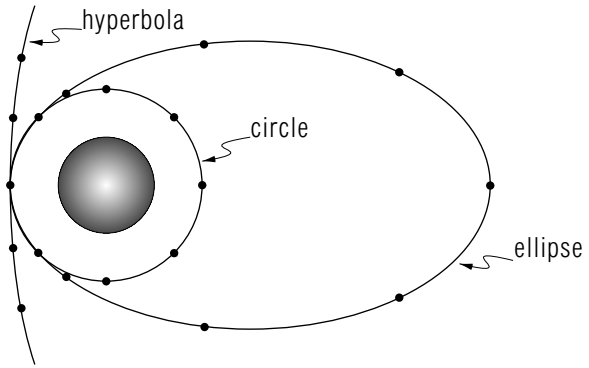
PHYSNET

MISN-0-105

ORBITS IN AN INVERSE-SQUARE-LAW FORCE FIELD: A COMPUTER PROJECT



Project PHYSNET Physics Bldg. Michigan State University East Lansing, MI

ORBITS IN AN INVERSE-SQUARE-LAW FORCE FIELD:
A COMPUTER PROJECT

by
Peter Signell

Title: **Orbits in an Inverse-Square-Law Force Field: A Computer Project**

Author: Peter Signell, Dept. of Physics, Mich. State Univ

Version: 2/1/2000 Evaluation: Stage 0

Length: 1 hr; 16 pages

Input Skills:

1. Vocabulary: apogee, orbit, perigee (MISN-0-102), mirror symmetry (Glossary).
2. State the expression for the orbital angular momentum of a point object and state the circumstances under which it is conserved (MISN-0-41).
3. Calculate the velocity of a mass traveling in a circular orbit in an inverse-square gravitational field (MISN-0-102).
4. State Kepler's three laws of planetary motion (MISN-0-102).

Output Skills (Project):

P1. On a computer, use a packaged version of the Numerov algorithm to calculate and graph circular, elliptical, and hyperbolic orbits in an inverse-square-law force field.

P2. Interpret the various characteristics of the resulting orbits and trajectories in terms of Kepler's laws, symmetries, and conservation of angular momentum.

External Resources (Required):

1. A calculator, graph paper, a microcomputer that accesses the program "ORBIT," and the hand-out "CBI on University Micros," (MISN-8-100).

Post-Options:

1. "Derivation of the Numerov Algorithm in Two Dimensions for Satellite and Planetary Orbits" (MISN-0-104).
2. "Derivation of Orbits and Trajectories in an Inverse-Square-Law Force Field" (MISN-0-106).

THIS IS A DEVELOPMENTAL-STAGE PUBLICATION
OF PROJECT PHYSNET

The goal of our project is to assist a network of educators and scientists in transferring physics from one person to another. We support manuscript processing and distribution, along with communication and information systems. We also work with employers to identify basic scientific skills as well as physics topics that are needed in science and technology. A number of our publications are aimed at assisting users in acquiring such skills.

Our publications are designed: (i) to be updated quickly in response to field tests and new scientific developments; (ii) to be used in both classroom and professional settings; (iii) to show the prerequisite dependencies existing among the various chunks of physics knowledge and skill, as a guide both to mental organization and to use of the materials; and (iv) to be adapted quickly to specific user needs ranging from single-skill instruction to complete custom textbooks.

New authors, reviewers and field testers are welcome.

PROJECT STAFF

Andrew Schnepp	Webmaster
Eugene Kales	Graphics
Peter Signell	Project Director

ADVISORY COMMITTEE

D. Alan Bromley	Yale University
E. Leonard Jossem	The Ohio State University
A. A. Strassenburg	S. U. N. Y., Stony Brook

Views expressed in a module are those of the module author(s) and are not necessarily those of other project participants.

ORBITS IN AN INVERSE-SQUARE-LAW FORCE FIELD: A COMPUTER PROJECT

by

Peter Signell

1. Computer Program "Orbit"

1a. Overview. The computer program ORBIT uses the second-order Numerov algorithm,[1] to solve Newton's second law for the case of a mass m in the gravitational field of the earth.

You specify the mass's initial coordinates x and y (in km), taking the origin of the coordinate system to be at the center of the earth, and also specify the mass's initial velocity vector in terms of its angular orientation and speed. The direction of the velocity vector must be specified in degrees, measured counterclockwise from the positive direction of the x-axis, and the speed must be in km/s. A typical input and its corresponding pictorial representation are shown in Fig. 1. The program ORBIT auto-

[1]For details see "Derivation of the Numerov Algorithm in Two Dimensions for Satellite and Planetry Orbits" (MISN-0-104).

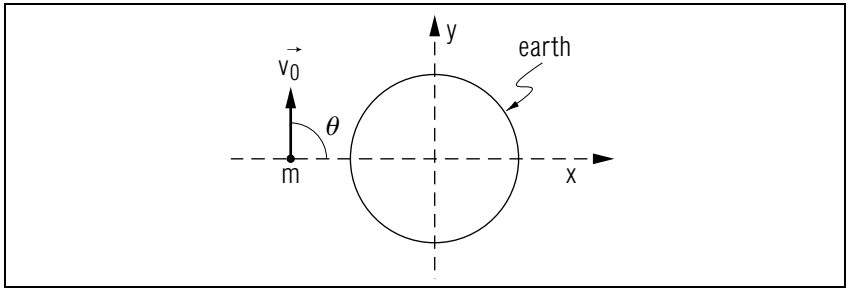


Figure 1. Graphed output for these input values:

ENTER X(KM)?	-6667
ENTER Y(KM)?	0
ENTER THETA(DEG)?	90
ENTER V(KM/SEC)?	8.357924
ENTER DELTA(SEC)?	2
MAX T(SEC)?	5040
PRINT EVERY Nth: N?	100

matically puts in the earth's mass and the gravitational constant in SI units and then solves the differential equation of motion,

$$\frac{\vec{F}}{m} = \vec{a} = \frac{d^2\vec{r}}{dt^2} = -\frac{Gm_E}{r^2}\hat{r}\,, \tag{1}$$

for the position of the mass at the ends of successive time-intervals Δ ("DELTA" in Fig. 1). The program ORBIT does not (in general) print out all the net-point time values you make it calculate. Rather, it tells you it will print out each n^{th} value, then asks you for your choice of "n." The time interval Δ determines the accuracy of the numerical solution:[2]

- If Δ is too large, the solution will be inaccurate due to the finite-interval computer algorith being a poor approximation to the differential equation.
- If Δ is too small, the solution will be inaccurate due to the subtraction of numbers have too many agreeing digits because of finite computer word lengths.

A typical value of Δ usually good for almost-3-digit accuracy is shown in Fig. 1.

1b. Input and Output. We ask you to input whatever initial values for x, y, θ, v, n, and Δ you wish. You should make 3 runs which differ only in the values used for v. Choose your three values to produce:

1. a circular orbit: $v^2 = G\,m_E/r$,
2. an elliptical orbit: $G\,m_E/r < v^2 < 2\,G\,m_E/r$, and
3. a hyperbolic trajectory: $2\,G\,m_E/r < v^2 < \infty$.[3]

Be sure that your output for case (2) really looks like an eccentric ellipse, not a circle. Plot each of the three sets of points on the same piece of graph paper and connect the points in each set with a smooth curve. Extend the hyperbolic curve backwards in time by making an identical run but with negative values of Δ or with the velocity vector reversed.

[2]For a more detailed discussion of how Δ is related to the accuracy, see MISN-0-104.

[3]The shape of orbits is discussed in "Derivation of Orbits in Inverse Square Law Force Fields" (MISN-0-106).

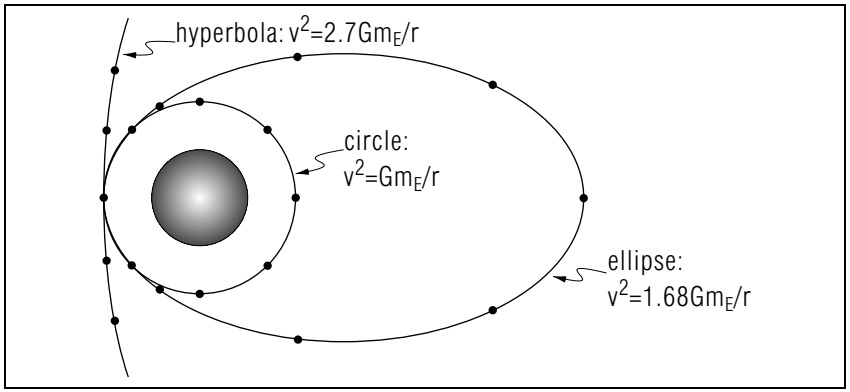


Figure 2. The three types of orbit about the earth. The distance scale is approximately 1: 2,500,000. The time increments are 18-1/3 minutes.

1c. Running the program. To actually run the program ORBIT you must first download it to your computer. Do that by going to the CBI Web site:

`physnet2.pa.msu.edu`

and then follow this path:

Reference Shelf ⇒ Optional Units ⇒ m105-p.bas

Clicking on the link `M105-p` will cause a window to appear in which you select the directory on your computer to which you want the program to be downloaded. A `temp` directory might be a good choice. After selection, go ahead and cause the program to be downloaded. Then go to that directory and run the program using the directions in MISN-8-100 (also downloadable).

2. Interpret the Graphs

2a. Circular Orbit is Circular. Check the radius of the various points along the trajectory and comment on the extent to which the radius stays constant, as it must for a circle.

2b. Elliptical Orbit is Symmetric. Holding the graph paper so the earth is at the left focus of the ellipse, the elliptical orbit has exact left-

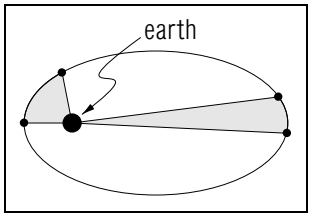


Figure 3. The two sectors are equal for equal time intervals.

right mirror symmetry. One way to show this is to make a vertical fold through the center of the ellipse and hold the graph paper up to the light.

2c. Velocities Do Not Show Mirror Symmetry. Despite the left-right spatial symmetry of the orbit, the velocity is very different on the left and right sides of the orbit. The speed of the satellite is greatest at the point closest to the earth, the perigee[4] and is least at the point farthest from the earth, the apogee.[5] The speed is inversely proportional to r because the angular momentum of the satellite ($\vec{L} = m\,\vec{r} \times \vec{v}$) is conserved in this isolated system (there are no external torques).

2d. Orbit is Indeed an Ellipse. The elliptical-looking orbit really is an ellipse with the earth at one focus (Kepler's first law). One way to show this is to locate the other focus by use of the left-right symmetry, then show that the total distance measured from one focus to any point on the orbit, and then from there to the other focus, is the same for all points on the orbit (using a ruler, check several such paths from one focus to the other via a point on the orbit). Another way is to show that the points on the orbit obey the equation

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1\,, \tag{2}$$

which is the Cartesian form for the equation of the ellipse centered at the origin. Show that the equation holds for several points on the orbit. Note that this x and y are not the same as those printed out by ORBIT, since they refer to a different origin. Note that for this equation: $y = 0$ when $x = \pm a$ and $x = 0$ when $y = \pm b$.

2e. Equal Areas Swept Out in Equal Times. According to Kepler's second law, the position vector sweeps out equal areas in equal times. One way to show this is to draw two sectors, a short fat one near the earth and a long thin one for the part of the orbit farthest from the

[4] Pronounced "pear′-ah-jee".
[5] Pronounced "ap′-a-jee".

earth (see Fig. 3). Count the graph paper squares in each sector to determine the two areas, estimating fractions of squares. Show that the two equal-time sectors have equal areas.

2f. Kepler's Third Law. Kepler's third law states that, for the circular and elliptical orbits, the squares of the periods are proportional to the cubes of the semi-major axes[6] of the orbits. You can estimate the satellite's period from the number of time intervals it takes for a complete orbit of the earth.

Acknowledgments

Preparation of this module was supported in part by the National Science Foundation, Division of Science Education Development and Research, through Grant #SED 74-20088 to Michigan State University.

Glossary

- **mirror symmetry**: a property of an object or a system where there exists a plane bisecting the object or system for which a point on one side of the plane has an equivalent point on the other side.

A. Basic to Calculate Orbits

```
'
'   PROGRAM TO TRACE AN ORBIT IN AN INVERSE SQUARE FORCE
'               (9/30/99)
'-----------------------------------------------------------
'
'        --> LIST OF PROCEDURES <--
'
80      GOSUB 200 'INITIALIZATIONS
        GOSUB 320 'INPUT INITIAL VALUES
        GOSUB 1050 'OUTPUT QUERY
        IF PRINT$ = "Y" THEN GOSUB 510 'PRINT INIT VALS
        GOSUB 620 'LOOP OVER TIMES
        '
```

[6]The semi-major axis of an ellipse is the distance from its center to either of the two furthest points from the center.


```
        INPUT "WANT ANOTHER RUN (Y/N)"; ANS$
        IF ANS$ = "y" OR ANS$ = "Y" THEN
            CLS : PRINT : GOTO 80
        '
        END
,
'-----------------------------------------------------------
200 'INITIALIZATIONS
    DIM A(4), B(4), C(4)
    DEFINT N
    PI = 3.14159
    SP = SQR(.5): G.M = 6.979 * 6.6732 * 10 ^ 4
    RAD.TO.DEG = 180 / 3.14159
    A(1) = .5:      B(1) = 2: C(1) = .5
    A(2) = 1 - SP: B(2) = 1: C(2) = 1 - SP
    A(3) = 1 + SP: B(3) = 1: C(3) = 1 + SP
    A(4) = 1 / 6:  B(4) = 2: C(4) = .5
    '
    RETURN
'-----------------------------------------------------------
320 'INPUT INITIAL VALUES
    '
    CLS
    PRINT "REFER TO UNIT 105 FOR TERMINOLOGY"
    PRINT
    PRINT "IN RESPONSE TO EACH OF THE FOLLOWING QUERIES,"
    PRINT "          TYPE IN YOUR NUMBER,"
    PRINT "          THEN HIT THE <RETURN> KEY."
    PRINT
    INPUT "     x(km)"; X0
    INPUT "     y(km)"; Y0
    INPUT "theta(deg)"; V.ANGLE.IN
    INPUT "  v(km/sec)"; V0
    INPUT "delta(sec)"; DELTA.T
    INPUT "max t(sec)"; MAX.TIME
    INPUT "Print every N-th point.  What is N"; N.PRINT
    V.ANGLE = V.ANGLE.IN / RAD.TO.DEG
    '
    RETURN
'-----------------------------------------------------------
510 'PRINT THE INITIAL VALUES
```

```
'
LPRINT "      x(km):"; X0
LPRINT "      y(km):"; Y0
LPRINT "theta(deg):"; V.ANGLE.IN
LPRINT "  v(km/sec):"; V0
LPRINT "delta(sec):"; DELTA.T
LPRINT "max t(sec):"; MAX.TIME
'
RETURN
'-------------------------------------------------------------------
'BEGIN LOOP OVER TIMES
'
X = X0: Y = Y0: VX = V0 * COS(V.ANGLE):
VY = V0 * SIN(V.ANGLE)
QX = 0: QY = 0: RX = 0: RY = 0: T = 0
CLS
PRINT "    T        X        Y        R      ANGLE"
IF PRINT$ = "Y" THEN
   LPRINT "    T        X        Y        R      ANGLE"
NO.POINT = 0
GOSUB 1160 ' get polar angle 'R.ANGLE' from X and Y.
R = SQR(X * X + Y * Y)
GOSUB 1250 'PRINT OUTPUT
'
FOR TIME = DELTA.T TO MAX.TIME STEP DELTA.T
  NO.POINT = NO.POINT + 1
  FOR J = 1 TO 4
    KX = A(J) * (VX - B(J) * QX)
    KY = A(J) * (VY - B(J) * QY)
    X = X + DELTA.T * KX
    Y = Y + DELTA.T * KY
    '
    GOSUB 1160 ' get polar angle 'R.ANGLE' from X and Y
    '
    R = SQR(X * X + Y * Y)
    G.M.OVER.R.CUBE = G.M / (R * R * R)
    AX = -G.M.OVER.R.CUBE * X
    AY = -G.M.OVER.R.CUBE * Y
    QX = QX + 3 * KX - C(J) * VX
    QY = QY + 3 * KY - C(J) * VY
    PX = A(J) * (AX - B(J) * RX)
```



```
    PY = A(J) * (AY - B(J) * RY)
    VX = VX + DELTA.T * PX
    VY = VY + DELTA.T * PY
    RX = RX + 3 * PX - C(J) * AX
    RY = RY + 3 * PY - C(J) * AY
  NEXT J
'
  IF TRACE = 0 THEN LOCATE , 1
'
  N.OUT = INT(NO.POINT / N.PRINT) * N.PRINT
  IF N.OUT = NO.POINT THEN GOSUB 1250 'PRINT OUTPUT
'
  NEXT TIME
PRINT
'
RETURN
'-------------------------------------------------------------------
'OUTPUT QUERY
'
PRINT
PRINT "OUTPUT WILL GO TO SCREEN AUTOMATICALLY."
INPUT "      OUTPUT TO PRINTER AS WELL (Y/N)?"; PRINT$
PRINT
IF PRINT$ = "y" THEN PRINT$ = "Y"
IF PRINT$ = "n" THEN PRINT$ = "N"
IF PRINT$ <> "Y" AND PRINT$ <> "N" THEN GOTO 1080
'
RETURN
'-------------------------------------------------------------------
'CALCULATE POLAR ANGLE FROM X AND Y
'
IF X > 0 AND Y > 0 THEN R.ANGLE = ATN(Y / X)
IF X < 0 AND Y >= 0 THEN R.ANGLE = PI + ATN(Y / X)
IF X < 0 AND Y < 0 THEN R.ANGLE = PI + ATN(Y / X)
IF X > 0 AND Y < 0 THEN R.ANGLE = 2 * PI + ATN(Y / X)
'
RETURN
'-------------------------------------------------------------------
'PRINT OUTPUT
'
R.ANGLE.OUT = R.ANGLE * RAD.TO.DEG
```

```
PRINT USING "######␣␣␣"; TIME; X; Y; R; R.ANGLE.OUT
IF PRINT$ = "Y" THEN
    LPRINT USING "######␣␣␣"; TIME; X; Y; R; R.ANGLE.OUT
'
RETURN
'--------------- END OF PROGRAM LINES ----------------
```


LOCAL GUIDE

1. Before going to the microcomputer, compute the velocity ranges for the three types of orbit. You will probably wish to use a scientific calculator for this. Also write down the values of the other parameters you will have to enter into the micro (see Fig. 1).

2. Follow the directions given in "CBI on University Microcomputers," (MISN-8-100). The name of the computer program is ORBIT.EXE. Be sure to use values for the initial position and speed of the satellite that are different than the sample input used in Fig. 1.

3. When you have finished the computer runs, plot the x-y points on the piece of graph paper and connect the points with smooth curves. Mark all of the time net-points on the graphs as in Fig. 2.

4. Interpret the various characteristics of the resulting orbits and trajectories in terms of symmetries, conservation of angular momentum and Kepler's laws, as detailed in the module text.

5. Submit your computer output, single sheet of graph paper with the three plots, and the appropriate interpretations described in Step 4, when you take the exam on this unit. There will be no other requirements on the exam.