Asteroid tetrahedron shape models from spud data

B. Khushalani
Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109-2140, USA

Gravitational potentials considering a body to be a sphere or an ellipsoid appear quite commonly in literature. To extract science from asteroids, the latter are treated as point-masses or ellipsoids. A closed form solution for gravitational potential of a tetrahedral-shaped body has appeared recently although in geophysical context. In this paper, we show how to obtain a tetrahedral shape model for an asteroid from a spud model which gives the radius of the asteroid at different locations on the asteroid. For the benefit of readers we give complete and explicit coding to create the shape model.

AS spacecraft explorations of minor bodies in the solar system takes momentum, we face a problem of navigation issues of the spacecrafts at or near these bodies. Gravity field obtained by assuming spherical or ellipsoidal shape models for asteroids proves to be inadequate because of divergence associated with them. One way around this is to obtain polyhedral gravity field by discretizing the shape of the asteroid into tetrahedrons which can be later combined to obtain a coherent polyhedron[1].

A latitude–longitude shape model can be created with the help of either the images from instruments on-board the spacecraft or earth-based instrument, a so-called spud (potato) model[2].

Given this spud data in the form of a *n*-row, 3-column matrix containing latitude, longitude and the radius at that point, we wish to obtain a tetrahedral shape model for the asteroid.

Vertex and face numbering

We begin by dividing the 90°S to 90°N latitude range into finite divisions, each division being latSpacing. And similarly, the 0 to 360° longitude range, each division being lonSpacing. As an example, let latSpacing and lonSpacing each be 10°. This is shown in Figure 1.

We start by numbering the 90°S vertex as 1, followed by 80°S, 0° lon vertex as 2, 80°S, 10° lon vertex as 3 and so on around the longitude ring. Each longitude ring then contains lonDivisions number of vertices where

$$\text{lonDivisions} = 360/\text{lonSpacing}.$$

Similarly,

$$\text{latDivisions} = (180/\text{latSpacing}) + 1.$$

For the surface normals to point outward, we now assign numbers to the faces, each face containing a 1×3 vector of vertices which go counter-clockwise as seen from outside the asteroid. The first face will connect v2, v3, v1; the second face v3, v4, v1 and so on, where v1 is the sole vertex at the 90°S latitude. This is shown in Figure 1 for

The total number of faces between 90 S and 80 S (and 90 N and 80 N) is lonDivisions. And that between any other pair of latitudes is two times lonDivisions. The explicit formulae then become,

$$\text{total faces} = (2 \times \text{lonDivisions}) \times (\text{latDivisions} - 2),$$

$$\text{total vertices} = \text{lonDivisions} \times (\text{latDivisions} - 2) + 2.$$

80 S–70 S faces, for example, can then be arranged as

face 1: $V_{1,70}V_{2,80}V_{1,80}$

face 2: $V_{1,70}V_{2,70}V_{2,80}$

face 3: $V_{2,70}V_{3,80}V_{2,80}$

face 4: $V_{2,70}V_{3,70}V_{3,80}$

$\vdots$

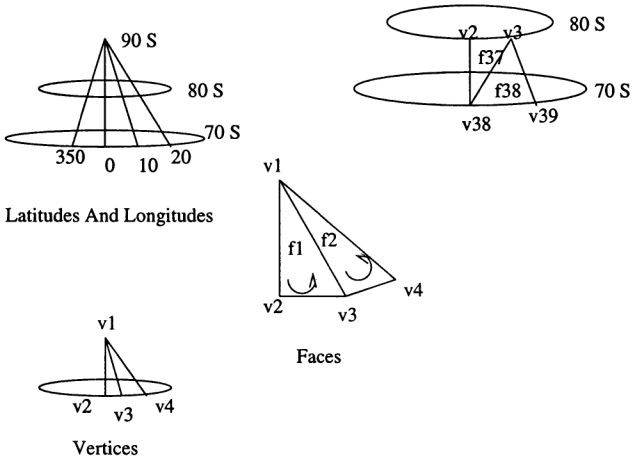

Figure 1.

e-mail: bharat@engin.umich.edu

face 69: $V_{35,70}V_{36,80}V_{35,80}$

face 70: $V_{35,70}V_{36,70}V_{36,80}$

face 71: $V_{36,70}V_{1,80}V_{36,80}$

face 72: $V_{36,70}V_{1,70}V_{1,80}$,

around the ring with (2 × lonDivisions) number of faces. Here, $V_{1,70}$ is the 0 lon vertex at lat 70°, $V_{2,80}$ is the 10 lon vertex at lat 80°, etc.

While for the two ends, the faces joining 90 S to 80 S and 90 N to 80 N are

face 1: $V_{1,80S}V_{2,80S}V_{90S}$

face 2: $V_{2,80S}V_{3,80S}V_{90S}$

$\vdots$

face 36: $V_{36,80S}V_{1,80S}V_{90S}$,

and

face 1: $V_{90N}V_{2,80N}V_{1,80N}$

face 2: $V_{90N}V_{3,80N}V_{2,80N}$

$\vdots$

face 36: $V_{90N}V_{1,80N}V_{36,80N}$,

respectively.

Removing the 90 S and 90 N vertices, the vertex matrix can be written

$$\begin{bmatrix} V_{1,80S} & V_{2,80S}\cdots & V_{\text{lonDiv},80S} \\ V_{1,70S} & V_{2,70S}\cdots & V_{\text{lonDiv},70S} \\ \vdots & & \\ V_{1,80N} & V_{2,80N}\cdots & V_{\text{lonDiv},80N} \end{bmatrix}.$$

The 10° spacing 17 × 37 augmented vertex matrix is,

$$\begin{bmatrix} 2 & 3\cdots & 37 & 2 \\ 38 & 39\cdots & 73 & 38 \\ \vdots & & & \\ 578 & 579\cdots & 613 & 578 \end{bmatrix}.$$

Explicit coding to generate the vertex number matrix then can be,

```
for i = 1: latDivisions – 2
  for j = 1: lonDivisions
    vertexNumberMatrix(i, j) = (lonDivisions*
    (i – 1)) + 2 + (j – 1);
  end
  vertexNumberMatrix(i, j + 1) =
  vertexNumberMatrix(i, 1);
end
```

Generating face file

To generate the faces from the vertex number matrix, we move inside this matrix as in Figure 2.

If we start with the first face, the face file containing 1 × 3 vector with connecting vertices as its elements, takes the form

```
for j = 1: lonDivisions
  faceFile(faceNumber, 1) = vertexNumberMatrix(1, j);
  faceFile(faceNumber, 2) = vertexNumberMatrix(1, j + 1);
  faceFile(faceNumber, 3) = 1;
  faceNumber = faceNumber + 1;
end

for i = 1: latDivisions – 3
 for j = 1: lonDivisions
  faceFile(faceNumber, 1) = vertexNumberMatrix(1, j);
  faceFile(faceNumber, 2) = vertexNumberMatrix(1, j + 1);
  faceFile(faceNumber, 3) = vertexNumberMatrix(1, j);

  faceFile(faceNumber+1,1)=vertexNumberMatrix(i + 1, j);
  faceFile(faceNumber + 1, 2)=
  vertexNumberMatrix(i + 1, j + 1);
  faceFile(faceNumber+1,3)=vertexNumberMatrix(i, j+1);
  faceNumber = faceNumber + 2;
 end
end

    for j = 1: lonDivisions
      faceFile(faceNumber, 1) =
      lonDivisions*(latDivisions – 2) + 2;
      faceFile(faceNumber, 2) =
      vertexNumberMatrix(latDivisions – 2j + 1);
      faceFile(faceNumber, 3) =
      vertexNumberMatrix(latDivisions – 2j);

      faceNumber = faceNumber + 1;
    end
```

This completes the inventory of faces.

Generating vertex file

Spud data[2], for an equal 10° lat–lon spacing takes the format as shown in Table 1. From our vertex definitions,

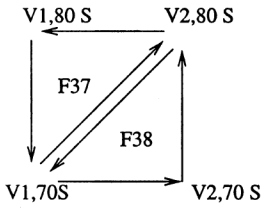

Figure 2.

Table 1

Lat	Lon	Rad
– 90	0	r_1
– 90	10	r_1
⋮		
– 90	360	r_1
– 80	0	r_2
– 80	10	r_3
⋮		
– 80	360	r_2
⋮		
90	0	$r_{\text{last vertex}}$
⋮		
90	360	$r_{\text{last vertex}}$

Table 2

Lat	Lon	Vertex number
– 90	0	1
⋮		
– 90	360	1
– 80	0	2
⋮		
– 80	350	37
– 80	360	2
⋮		
90	0	614
⋮		
90	360	614

we can now put the vertex numbers corresponding to each (lat, lon) pair as shown in Table 2. Each vertex in Table 2 is again a 1×3 vector containing its x, y, z components calculated from the spud table as

$$V_x = r \cos(\text{lat}) \cos(\text{lon})$$
$$V_y = r \cos(\text{lat}) \sin(\text{lon})$$
$$V_z = r \sin(\text{lat}).$$

We thus obtain the vertex file containing the vertex number and the three components of the corresponding vertices.

Figure 3. Asteroid *Bhishma* tetrahedrally discretized.

Generating shape models

Availability of the spud data for a couple of asteroids now allows us to obtain a precision-controlled shape model for that asteroid. This can be used in our further analysis relating to, for example, evolution of ejecta trajectories from these asteroids.

Figure 3 shows the discretized shape model of asteroid *Bhishma*. We see from the figure that a commonly assumed ellipsoidal shape of $33 \times 13 \times 13$ km is far from the true shape for this asteroid. In fact, clearly visible now is the punch in front of the asteroid and a hump behind it. Models assuming the ellipsoidal shape will not give true representation of *Bhishma* and a gravitational potential obtained by taking it to be a sphere or a point-mass will be outrightly wrong.

1. Scheeres, D. J., Khushalani, B. and Werner, R., Asteroids, Comets and Meteors Conference, Cornell University, Ithaca, 1999.
2. http://loring.jhuapl.edu/NEAR/SDC/ScienceTeamProducts/MSI/
3. Mortenson, M., *Geometric Modeling*, Wiley, New York, 1997.
4. Gurumoorthy, B., Class notes for the course on Geometric Modelling, Indian Institute of Science, Bangalore, India.

Received 12 April 2000; accepted 30 August 2000