Calculation of Volume of Any Solid

Sourabh P. Bhat
https://spbhat.in

December 15, 2019

1 Theory

The need for calculation of volume of solids arises in many computations. For primitive solids, like sphere, cylinder, cone, pyramids and prisms, we have well-defined analytical expressions for calculation of their volume. However, for an arbitrarily shaped solid, such a calculation can be daunting and it could be extremely difficult to derive an analytical closed form expression. But, if we are able to break down the surface of the solid into triangles, then the volume can be computed using the Stokes theorem. Given that any surface in three-dimensions can be approximated by a set of triangles, the volume of any solid can be calculated to a required degree of accuracy by using enough number of triangles for approximating the enclosing surface of the solid.
The volume of a solid is defined as,

$$\iiint_{\text{vol}} dx\, dy\, dz \,.$$

We can re-write this as,

$$\iiint_{\text{vol}} \frac{\partial x}{\partial x}\, dx\, dy\, dz \,.$$

Using the Stokes theorem (the divergence theorem) we can write,

$$\iint_{\text{Area}} x\, n_x\, dA \,,$$

where, n_x is the x-component of the unit normal. This means that now we can calculate the volume by just carrying out surface integrals over enclosing volume surfaces. We can easily break-down a enclosing surface into to a set of triangles. We must be cautious however that the **normal must be pointing outwards** and the vertices of the triangle must be in the **anti-clockwise direction**.

Let us compute the contribution of a single triangle to the volume. Since a triangle is a planar surface, n_x is a constant and can be moved out of the integral, which results in,

$$n_x \iint_{\text{Area}} x\, dA \,.$$

This is basically a formula for x-coordinate of the centroid of the triangle, times n_x times Area of the triangle. For a triangle, with vertices a, b and c, the x-coordinate of the centroid can be easily calculated as,

$$\bar{x} = \frac{xa + xb + xc}{3} .$$

Also, let us assume that the vertices are in an anti-clockwise direction. The two vectors formed by points a-b and a-c (vector $\vec{ab}$ and vector $\vec{ac}$) can be used for calculation of the area of the triangle as,

$$\vec{A} = \frac{1}{2}\left(\vec{ab} \times \vec{ac}\right) .$$

This will result in a vector, whose length is equal to the area of the triangle and direction is such that it points outwards from the solid. Scaling the vector to unit length and taking its x-component will give us n_x. Taking the magnitude will give area, A. Adding up the contributions from all the triangles, making up the surface, will give the volume of the solid as,

$$\sum_{i=0}^{n-1} n_{xi} \ |A_i| \ \bar{x}_i \ ,$$

which can be simplified to,

$$\sum_{i=0}^{n-1} \vec{A}_{ix} \ \bar{x}_i \ ,$$

where, $\vec{A}_{ix}$ is the x-component of the area vector, $\vec{A}$. Therefore, the volume of a solid made-up of n enclosing triangles can be computed as,

$$\text{Volume} = \sum_{i=0}^{n-1} \vec{A}_{ix} \ \bar{x}_i \ .$$

2 Code

Below is a program written in Java, to calculate volume of any solid.

Listing 1: Volume.java

```
/*
 * @author Sourabh Bhat (https://spbhat.in)
 *
 * Calculates the volume of a solid formed by a set of triangles.
 * The triangles must have points either all-clockwise or all-anti-clockwise,
 * looking from outside of the solid.
 *
 * @param triangles array of triangles
 * @return non-negative volume enclosed by the set of triangles
 */
public class Volume {
    public static double volume(Triangle[] triangles) {
        double volume = 0.0;
        for (Triangle t : triangles) {
            volume += volumeUnder(t);
        }
```

```
        return Math.abs(volume);
    }

    private static double volumeUnder(Triangle tri) {
        Point[] p = tri.points();
        double xbar = (p[0].x + p[1].x + p[2].x) / 3.0;
        Vector vab = new Vector(p[0], p[1]);
        Vector vac = new Vector(p[0], p[2]);

        // x-component of cross product
        double ax = vab.y * vac.z - vac.y * vab.z;

        return ax * xbar * 0.5;
    }
}
```

A simple implementation of required classes is provided below. For more elaborate implementation look at the code in my CFDSolver repository.

Listing 2: Triangle.java

```
/**
 * @author Sourabh Bhat (https://spbhat.in)
 */
public class Triangle {
    private final Point[] points;

    public Triangle(Point p0, Point p1, Point p2) {
        points = new Point[]{p0, p1, p2};
    }

    Point[] points() {
        return points;
    }
}
```

Listing 3: Point.java

```
/**
 * @author Sourabh Bhat (https://spbhat.in)
 */
public class Point {
    public final double x, y, z;

    public Point(double x, double y, double z) {
        this.x = x;
        this.y = y;
        this.z = z;
    }
}
```