Polyhedral Mass Properties (Revisited)

David Eberly
Magic Software, Inc.
http://www.magic-software.com

Created: December 31, 2002
Modified: January 25, 2003 (fixed incorrect pseudocode names, inertia now about center of mass)

1 Introduction

This document takes a second look at the article by Brian Mirtich, *Fast and accurate computation of polyhedral mass properties*, Journal of Graphics Tools, vol. 1, no. 2, pp. 31–50, 1996. The article discusses how to compute the mass and inertia tensor for a solid, simple polyhedron of constant mass density. The construction uses the Divergence Theorem from calculus for reducing volume integrals to surface integrals, a reduction from three–dimensional integrals to two–dimensional integrals. The polyhedron surface is a union of planar faces, so the surface integrals are effectively integrals in various planes. Projection of these faces onto coordinate planes are used to set up yet another reduction in dimension. Green's Theorem, the two–dimensional analog of the Divergence Theorem, is used to reduce the planar integrals to line integrals around the boundary of the projected faces.

Two important points emphasized in the paper are (1) the projection of the polyhedron faces onto the appropriate coordinate planes to avoid numerical problems and (2) the reduction using Green's Theorem to obtain common subexpressions (integrals of polynomials of one variable) to avoid redundant calculations. Item (2) occurs to handle polyhedron faces with four or more vertices. Item (1) is necessary in order to robustly compute what is required by item (2). When the polyhedron faces are triangles, neither items (1) nor (2) are necessary. A simpler construction is provided here when the polyhedron faces are triangles. A consequence of the formulas as derived in this document is that they require significantly less computational time than does Mirtich's formulas. I suspect that for nontriangular faces, Mirtich's formulas are reducible to simpler expressions.

2 Reduction of Volume Integrals

The mass, center of mass, and inertia tensor require computing volume integrals of the type

$$\int_V p(x,y,z)\,dV \tag{1}$$

where V is the volumetric region of integration and dV is an infinitesimal measure of volume. The function $p(x,y,z)$ is a polynomial selected from 1, x, y, z, x^2, y^2, z^2, xy, xz, and yz. We are interested in computing these integrals where V is the region bounded by a simple polyhedron. A volume integral may be converted to a surface integral via the Divergence Theorem from calculus:

$$\int_V p(x,y,z)\,dV = \int_V \boldsymbol{\nabla}\cdot\mathbf{F}\,dV = \int_S \mathbf{N}\cdot\mathbf{F}\,dS \tag{2}$$

where S is the boundary of the polyhedron, a union of triangular faces, and where dS is an infinitesimal measure of surface area. The function $\mathbf{F}(x,y,z)$ is chosen so that $\boldsymbol{\nabla}\cdot\mathbf{F} = p$. The vector $\mathbf{N}$ denotes outward–pointing, unit–length surface normals. The choices for $\mathbf{F}$ in the Mirtich paper are

p	$\mathbf{F}$	p	$\mathbf{F}$
1	$(x,0,0)$	y^2	$(0,y^3/3,0)$
x	$(x^2/2,0,0)$	z^2	$(0,0,z^3/3)$
y	$(0,y^2/2,0)$	xy	$(x^2y/2,0,0)$
z	$(0,0,z^2/2)$	xz	$(0,0,z^2x/2)$
x^2	$(x^3/3,0,0)$	yz	$(0,y^2z/2,0)$

(3)

The computational effort is now devoted to calculating the integrals $\int_S \mathbf{N}\cdot\mathbf{F}\,dS$. The boundary S is just the union of polyhedral faces $\mathcal{F}$. An outward–pointing, unit–length normal to face $\mathcal{F}$ is denoted by $\mathbf{N}_{\mathcal{F}}$. The surface integral decomposes to

$$\int_S \mathbf{N}\cdot\mathbf{F}\,dS = \sum_{\mathcal{F}\in S}\int_{\mathcal{F}} \mathbf{N}_{\mathcal{F}}\cdot\mathbf{F}\,dS \tag{4}$$

The integrals to be computed are now reduced to

$$\begin{array}{lcllcl}
\int_V dV &=& \sum_{\mathcal{F}\in S}(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\imath})\int_{\mathcal{F}} x\,dS & \int_V y^2\,dV &=& \frac{1}{3}\sum_{\mathcal{F}\in S}(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\jmath})\int_{\mathcal{F}} y^3\,dS \\
\int_V x\,dV &=& \frac{1}{2}\sum_{\mathcal{F}\in S}(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\imath})\int_{\mathcal{F}} x^2\,dS & \int_V z^2\,dV &=& \frac{1}{3}\sum_{\mathcal{F}\in S}(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{k})\int_{\mathcal{F}} z^3\,dS \\
\int_V y\,dV &=& \frac{1}{2}\sum_{\mathcal{F}\in S}(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\jmath})\int_{\mathcal{F}} y^2\,dS & \int_V xy\,dV &=& \frac{1}{2}\sum_{\mathcal{F}\in S}(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\imath})\int_{\mathcal{F}} x^2y\,dS \\
\int_V z\,dV &=& \frac{1}{2}\sum_{\mathcal{F}\in S}(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{k})\int_{\mathcal{F}} z^2\,dS & \int_V yz\,dV &=& \frac{1}{2}\sum_{\mathcal{F}\in S}(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\jmath})\int_{\mathcal{F}} y^2z\,dS \\
\int_V x^2\,dV &=& \frac{1}{3}\sum_{\mathcal{F}\in S}(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\imath})\int_{\mathcal{F}} x^3\,dS & \int_V xz\,dV &=& \frac{1}{2}\sum_{\mathcal{F}\in S}(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{k})\int_{\mathcal{F}} z^2x\,dS
\end{array} \tag{5}$$

3 Computation of Surface Integrals

We now need to compute integrals of the form

$$(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\ell})\int_{\mathcal{F}} q(x,y,z)\,dS \tag{6}$$

where $\boldsymbol{\ell}$ is one of $\boldsymbol{\imath}$, $\boldsymbol{\jmath}$, or $\boldsymbol{k}$ and where q is one of x, x^2, y^2, z^2, x^3, y^3, z^3, x^2y, y^2z, or z^2x. Let the triangular face be counterclockwise ordered and have vertices $\mathbf{P}_i = (x_i,y_i,z_i)$, $0\le i\le 2$. Two edges are

$$\mathbf{E}_i = \mathbf{P}_i - \mathbf{P}_0 = (x_i-x_0, y_i-y_0, z_i-z_0) = (\alpha_i,\beta_i,\gamma_i) \tag{7}$$

for $1\le i\le 2$. A parameterization of the face is

$$\begin{aligned}
\mathbf{P}(u,v) &= \mathbf{P}_0 + u\mathbf{E}_1 + v\mathbf{E}_2 \\
&= (x_0+\alpha_1u+\alpha_2v, y_0+\beta_1u+\beta_2v, z_0+\gamma_1u+\gamma_2v) \\
&= (x(u,v), y(u,v), z(u,v))
\end{aligned} \tag{8}$$

where $u \geq 0$, $v \geq 0$, and $u + v \leq 1$. The infinitesimal measure of surface area is

$$dS = \left| \frac{\partial \mathbf{P}}{\partial u} \times \frac{\partial \mathbf{P}}{\partial v} \right| du\, dv = |\mathbf{E}_1 \times \mathbf{E}_2|\, du\, dv \tag{9}$$

and the outer pointing unit–length face normal is

$$\mathbf{N}_{\mathcal{F}} = \frac{\mathbf{E}_1 \times \mathbf{E}_2}{|\mathbf{E}_1 \times \mathbf{E}_2|} = \frac{(\beta_1\gamma_2 - \beta_2\gamma_1, \alpha_2\gamma_1 - \alpha_1\gamma_2, \alpha_1\beta_2 - \alpha_2\beta_1)}{|\mathbf{E}_1 \times \mathbf{E}_2|} = \frac{(\delta_0, \delta_1, \delta_2)}{|\mathbf{E}_1 \times \mathbf{E}_2|} \tag{10}$$

The integrals in equation (6) reduces to

$$(\mathbf{N}_{\mathcal{F}} \cdot \boldsymbol{\ell}) \int_{\mathcal{F}} q(x, y, z)\, dS = (\mathbf{E}_1 \times \mathbf{E}_2 \cdot \boldsymbol{\ell}) \int_0^1 \int_0^{1-v} q(x(u, v), y(u, v), z(u, v))\, du\, dv \tag{11}$$

where $x(u, v)$, $y(u, v)$, and $z(u, v)$ are the components of the parameterization in equation (8).

The integrals on the right–hand side of equation (11) can be computed symbolically, either by hand or by a symbolic algebra package. The formulas listed later were computed using Mathematica. The instructions are

```
x = x0+u*(x1-x0)+v*(x2-x0)
y = y0+u*(y1-y0)+v*(y2-y0)
z = z0+u*(z1-z0)+v*(z2-z0)
intx = Together[Integrate[x,{v,0,1},{u,0,1-v}]]
intxx = Together[Integrate[x^2,{v,0,1},{u,0,1-v}]]
intyy = Together[Integrate[y^2,{v,0,1},{u,0,1-v}]]
intzz = Together[Integrate[z^2,{v,0,1},{u,0,1-v}]]
intxxx = Together[Integrate[x^3,{v,0,1},{u,0,1-v}]]
intyyy = Together[Integrate[y^3,{v,0,1},{u,0,1-v}]]
intzzz = Together[Integrate[z^3,{v,0,1},{u,0,1-v}]]
intxxy = Together[Integrate[x^2*y,{v,0,1},{u,0,1-v}]]
intyyz = Together[Integrate[y^2*z,{v,0,1},{u,0,1-v}]]
intzzx = Together[Integrate[z^2*x,{v,0,1},{u,0,1-v}]]
```

Common subexpressions may be obtained by some additional factoring. Define

$$s_n(w) = \sum_{i=0}^{n} w_0^{n-i} w_1^i, \quad f_0(w) = 1, \quad \text{and} \quad f_n(w) = s_n(w) + w_2 f_{n-1}(w) \quad \text{for } n \geq 1 \tag{12}$$

Think of these as macros where the input argument is a textual replacement wherever w occurs in the right–hand sides. Also define the macro

$$g_i(w) = f_2(w) + w_i f_1(w) + w_i^2 \tag{13}$$

The specific expressions required in the surface integrals are listed below in terms of w. Each macro is expanded three times, once for each of x, y, and z.

$$\begin{aligned}
f_1(w) &= w_0 + w_1 + w_2 = [w_0 + w_1] + w_2 \\
f_2(w) &= w_0^2 + w_0 w_1 + w_1^2 + w_2 f_1(w) = [[w_0^2] + w_1\{w_0 + w_1)\}] + w_2\{f_1(w)\} \\
f_3(w) &= w_0^3 + w_0^2 w_1 + w_0 w_1^2 + w_1^3 + w_2 f_2(w) = w_0\{w_0^2\} + w_1\{w_0^2 + w_0 w_1 + w_1^2\} + w_2\{f_2(w)\} \\
g_i(w) &= \{f_2(w)\} + w_i(\{f_1(w)\} + w_i)
\end{aligned} \tag{14}$$

The square brackets $[\cdot]$ indicate that the subexpression is computed and saved in temporary variables for later use. The curly braces $\{\cdot\}$ indicate that the subexpression was computed earlier and can be accessed from temporary variables. The number of subexpressions that must be stored at any one time is small, so cache coherence should not be an issue when enough floating point registers are available for storing the subexpressions (see the pseudocode later in this document).

The integrals are

$$\begin{array}{lcllcl}
(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\imath})\int_{\mathcal{F}} x\,dS &=& \frac{\delta_0}{6}f_1(x) & (\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\jmath})\int_{\mathcal{F}} y^3\,dS &=& \frac{\delta_1}{20}f_3(y)\\
(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\imath})\int_{\mathcal{F}} x^2\,dS &=& \frac{\delta_0}{12}f_2(x) & (\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{k})\int_{\mathcal{F}} z^3\,dS &=& \frac{\delta_2}{20}f_3(z)\\
(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\jmath})\int_{\mathcal{F}} y^2\,dS &=& \frac{\delta_1}{12}f_2(y) & (\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\imath})\int_{\mathcal{F}} x^2y\,dS &=& \frac{\delta_0}{60}(y_0g_0(x)+y_1g_1(x)+y_2g_2(x))\\
(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{k})\int_{\mathcal{F}} z^2\,dS &=& \frac{\delta_2}{12}f_2(z) & (\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\jmath})\int_{\mathcal{F}} y^2z\,dS &=& \frac{\delta_1}{60}(z_0g_0(y)+z_1g_1(y)+z_2g_2(y))\\
(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\imath})\int_{\mathcal{F}} x^3\,dS &=& \frac{\delta_0}{20}f_3(x) & (\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{k})\int_{\mathcal{F}} z^2x\,dS &=& \frac{\delta_2}{60}(x_0g_0(z)+x_1g_1(z)+x_2g_2(z))
\end{array} \tag{15}$$

4 Comparison to Mirtich's Formulas

Let us compare the formulas for $Q_x = (\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\imath})\int_{\mathcal{F}} x\,dS$. In this document the integral is

$$Q_x = \frac{\delta_0}{6}f_1(x) = \frac{\delta_0}{6}(x_0+x_1+x_2) \tag{16}$$

In Mirtich's formulas there are two possibilities for computing Q_x. One occurs when the projection variable is $\gamma = z$. Assembling the pieces of the corresponding formulas in the paper and switching to the notation used in this document,

$$\begin{array}{rcl}
Q_x &=& \frac{\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\imath}}{|\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{k}|}\pi_x\\
&=& \frac{\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\imath}}{|\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{k}|}\frac{\mathrm{sgn}(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{k})}{6}\sum_{i=0}^{2}(y_{i+1}-y_i)(x_{i+1}^2+x_{i+1}x_i+x_i^2)\\
&=& \frac{\delta_0}{6}\frac{(y_1-y_0)(x_1^2+x_0x_1+x_0^2)+(y_2-y_1)(x_2^2+x_1x_2+x_1^2)+(y_0-y_2)(x_0^2+x_0x_2+x_2^2)}{(x_1-x_0)(y_2-y_0)-(x_2-x_0)(y_1-y_0)}
\end{array} \tag{17}$$

The final formula requires much more computational time than the one derived in this document. In fact the numerator is exactly divisible by the denominator and the fraction reduces to $x_0+x_1+x_2$, as it should to be equivalent to the Q_x in equation (16). The reduction was verified using Mathematica. If the projection variable is $\gamma = x$,

$$\begin{array}{rcl}
Q_x &=& \frac{-1}{|\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\imath}|}\left((\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{\jmath})\pi_y+(\mathbf{N}_{\mathcal{F}}\cdot\boldsymbol{k})\pi_z-(\mathbf{N}_{\mathcal{F}}\cdot\mathbf{P}_0)\pi_1\right)\\
&=& -\frac{1}{\delta_0}\left(\frac{\delta_1}{6}\sum_{i=0}^{2}(z_{i+1}-z_i)(y_{i+1}^2+y_{i+1}y_i+y_i^2)-\frac{\delta_2}{6}\sum_{i=0}^{2}(y_{i+1}-y_i)(z_{i+1}^2+z_{i+1}z_i+z_i^2)-\right.\\
&& \left.\frac{\delta_0x_0+\delta_1y_0+\delta_2z_0}{2}\sum_{i=0}^{2}(z_{i+1}-z_i)(y_{i+1}+y_i)\right)
\end{array} \tag{18}$$

The correctness of this formula was verified using Mathematica; in fact it reduces to the one in equation (16). The computational requirements for this expression are enormous compared to that of equation (16).

Comparisons between the formulas for the other integrals is possible, but you will find that the differences in computational time become even greater than in the example shown here.

5 Pseudocode

The pseudocode for computing the integrals is quite simple. The polyhedron vertices are passes as the array `p[]`. The number of triangles is `tmax`. The array `index[]` has `tmax` triples of integers that are indices into the vertex array. The return values are the mass, the center of mass, and the inertia tensor relative to the center of mass.

```
MACRO Subexpressions(w0,w1,w2,f1,f2,f3,g0,g1,g2)
{
    temp0 = w0+w1;  f1 = temp0+w2;  temp1 = w0*w0;  temp2 = temp1+w1*temp0;
    f2 = temp2+w2*f1;  f3 = w0*temp1+w1*temp2+w2*f2;
    g0 = f2+w0*(f1+w0);  g1 = f2+w1*(f1+w1);  g2 = f2+w2*(f1+w2);
}

void Compute (Point p[], int tmax, int index[], Real& mass, Point& cm, Matrix& inertia)
{
    constant Real mult[10] = {1/6,1/24,1/24,1/24,1/60,1/60,1/60,1/120,1/120,1/120};
    Real intg[10] = {0,0,0,0,0,0,0,0,0,0};  // order: 1, x, y, z, x^2, y^2, z^2, xy, yz, zx
    for (t = 0; t < tmax; t++)
    {
        // get vertices of triangle t
        i0 = index[3*t];  i1 = index[3*t+1];  i2 = index[3*t+2];
        x0 = p[i0].x;  y0 = p[i0].y;  z0 = p[i0].z;
        x1 = p[i1].x;  y1 = p[i1].y;  z1 = p[i1].z;
        x2 = p[i2].x;  y2 = p[i2].y;  z2 = p[i2].z;

        // get edges and cross product of edges
        a1 = x1-x0;  b1 = y1-y0;  c1 = z1-z0;  a2 = x2-x0;  b2 = y2-y0;  c2 = z2-z0;
        d0 = b1*c2-b2*c1;  d1 = a2*c1-a1*c2; d2 = a1*b2-a2*b1;

        // compute integral terms
        Subexpressions(x0,x1,x2,f1x,f2x,f3x,g0x,g1x,g2x);
        Subexpressions(y0,y1,y2,f1y,f2y,f3y,g0y,g1y,g2y);
        Subexpressions(z0,z1,z2,f1z,f2z,f3z,g0z,g1z,g2z);

        // update integrals
        intg[0] += d0*f1x;
        intg[1] += d0*f2x;  intg[2] += d1*f2y;  intg[3] += d2*f2z;
        intg[4] += d0*f3x;  intg[5] += d1*f3y;  intg[6] += d2*f3z;
        intg[7] += d0*(y0*g0x+y1*g1x+y2*g2x);
        intg[8] += d1*(z0*g0y+z1*g1y+z2*g2y);
        intg[9] += d2*(x0*g0z+x1*g1z+x2*g2z);
    }
    for (i = 0; i < 10; i++)
        intg[i] *= mult[i];

    mass = intg[0];

    // center of mass
    cm.x = intg[1]/mass;
    cm.y = intg[2]/mass;
```

```
    cm.z = intg[3]/mass;

    // inertia tensor relative to center of mass
    inertia.xx = intg[5]+intg[6]-mass*(cm.y*cm.y+cm.z*cm.z);
    inertia.yy = intg[4]+intg[6]-mass*(cm.z*cm.z+cm.x*cm.x);
    inertia.zz = intg[4]+intg[5]-mass*(cm.x*cm.x+cm.y*cm.y);
    inertia.xy = -(intg[7]-mass*cm.x*cm.y);
    inertia.yz = -(intg[8]-mass*cm.y*cm.z);
    inertia.xz = -(intg[9]-mass*cm.z*cm.x);
}
```