Finite domain integration of polynomials *

Alberto Paoluzzi

November 16, 2015

Abstract

In order to plan or control the static/dynamic behaviour of models in CAD applications, it is often necessary to evaluate integral properties ot solid models (i.e. volume, centroid, moments of inertia, etc.). This module deals with the exact evaluation of inertial properties of homogeneous polyhedral objects.

Contents

1 Introduction

A finite integration method from[CP90] is developed here the computation of various order monomial integrals over polyhedral solids and surfaces in 3D space. The integration method can be used for the exact evaluation of domain integrals of trivariate polynomial forms.

*This document is part of the *Linear Algebraic Representation with CoChains* (LAR-CC) framework [CL13]. November 16, 2015

2 Integration of polynomials over polyhedral domains

Here we summarize from [CP90] an exact and symbolic solution both to the surface and volume integration of polynomials, by using a triangulation of the volume boundary. The evaluation of surface and volume integrals is achieved by transforming them into line integrals over the boundary of every 2-simplex of a domain triangulation. A different approach to integration, using a decomposition into volume elements induced by a boundary triangulation is given in [**?**] where a closed formula for volume integration over polyhedral volumes, by decomposing the solid into a set of solid tetrahedra, but such a method cannot be used for surface integrations.

Problem statement The finite method [CP90] to compute double and triplet integrals of monomials over linear regular polyhedra in $\mathbb{R}^3$ is discussed. In particular, this method enables practical formulae for the exact evaluation of integrals to be achieved:

$$II_S \equiv \int\!\!\int_S f(\mathbf{p})\, dS, \qquad III_P \equiv \int\!\!\int\!\!\int_P f(\mathbf{p})\, dV, \tag{1}$$

where S, and P are linear and regular 2- or 3-polyhedra in $\mathbb{R}^3$, dS and dV are the differential surface and the differential volume. The integrating function is a trivariate polynomial

$$f(\mathbf{p}) = \sum_{\alpha=0}^{n} \sum_{\beta=0}^{m} \sum_{\gamma=0}^{p} a_{\alpha\beta\gamma} x^\alpha y^\beta z^\gamma,$$

where α, β, γ are non-negative integers.

Since the extension to $f(\mathbf{p})$ is straightforwardly given by the linearity of integral operator, we may focus on the calculation of integrals of monomials:

$$II_S^{\alpha\beta\gamma} \equiv \int\!\!\int_S x^\alpha y^\beta z^\gamma\, dS, \qquad III_P^{\alpha\beta\gamma} \equiv \int\!\!\int\!\!\int_P x^\alpha y^\beta z^\gamma\, dV. \tag{2}$$

Algorithm preview Surface integrals are computed as a summation of integrals over a triangulation of the surface. Any triangle is mapped into the unit triangle in the 2-space of parameters, where integrals of monomials become particularly simple. Then formulae for integrals over polyhedral volumes are given. They are easily derived by transforming volume integrals in surface integrals. It is possible to show that such integrals are computable in polynomial time, and that inertia moments are computable in $O(E)$ time, E being the number of edges of the solid model of the integration domain.

A very important feature of the integration formulae presented here is that they can also be used with a partial model of a polyhedron, consisting of the collection of its face loops. Loops are oriented counter-clockwise if external, clockwise if internal to another loop. Such a model, without explicit storage of face adjacencies, is very frequently adopted in Computer Graphics.

In this case it is sufficient to consider any $n+1$-sided (also unconnected or multiply connected) face as topological sum of $n-1$ oriented triangles t_i, with vertices $\langle v_0, v_i, v_{i+1}\rangle$, where $1 \leq i \leq n-1$. In applying formulae (12) or (15) to such a set of triangles, any edge that does not belong to the original polygon will be computed twice, in the two opposite directions. These contributions to the whole integral will mutually cancel each other out, as they correspond to pairs of line integrals evaluated along opposite paths.

Surface integration We call *structure product* the integral of a monomial over a simplicial complex. Exact formulae for structure products over n-sided polygons in 2-space, the unit triangle in 2-space, and an arbitrary triangle in 3-space, are derived in the following. Structure products are a generalization of the usual products and moments of inertia, that can be obtained from (2) by assuming $\alpha+\beta+\gamma \leq 2$.

Polygon integrals A structure product over a polygon π in the plane xy is

$$II_{\pi}^{\alpha\beta} = \int\!\!\int_{\pi} x^{\alpha} y^{\beta}\, dS, \qquad \alpha, \beta \geq 0, integers. \tag{3}$$

Such integrals can be exactly expressed, when π is a polygon with n oriented edges, as:

$$II_{\pi}^{\alpha\beta} = \frac{1}{\alpha+1} \sum_{i=1}^{n} \sum_{h=0}^{\alpha+1} \binom{\alpha+1}{h} x_i^{\alpha+1-h} X_i^h \sum_{k=0}^{\beta} \frac{\binom{\beta}{k}}{h+k+1} y_i^{\beta-k} Y_i^{k+1} \tag{4}$$

where $\mathbf{p}_i = (x_i, y_i)$, $X_i = x_{i+1} - x_i$, $Y_i = y_{i+1} - y_i$ and $\mathbf{p}_{n+1} = \mathbf{p}_1$. The derivation of the formula (4) is based on the application of Green's theorem and on Newton's expression for binomial powers.

Unit triangle integrals The general formula (4) can be specialized for the unit triangle $\tau' = \langle \mathbf{w}_o, \mathbf{w}_a, \mathbf{w}_b \rangle$, with vertices

$$\mathbf{w}_o = (0,0), \qquad \mathbf{w}_a = (1,0), \qquad \mathbf{w}_b = (0,1), \tag{5}$$

getting a very simplified expression. With some algebraic manipulations, we obtain[1]

$$II^{\alpha\beta} = \frac{1}{\alpha+1} \sum_{h=0}^{\alpha+1} \binom{\alpha+1}{h} \frac{(-1)^h}{h+\beta+1}, \tag{6}$$

which reduces, for $\alpha = \beta = 0$, to the area of the triangle (5): $II^{00} = 1/2$.

[1] $II_{\pi}^{\alpha\beta}$ is substituted, when referred to the unit triangle, by the symbol $II^{\alpha\beta}$.

Triangle integrals In the following we derive the general expression for structure products evaluated on an arbitrary triangle $\tau = \langle \mathbf{v}_o, \mathbf{v}_a, \mathbf{v}_b \rangle$ of the 3-space xyz, defined by $\mathbf{v}_o = (x_o, y_o, z_o)$ and by the vectors $\mathbf{a} = \mathbf{v}_a - \mathbf{v}_o$ and $\mathbf{b} = \mathbf{v}_b - \mathbf{v}_o$. The parametric equation of its embedding plane is:

$$\mathbf{p} = \mathbf{v}_o + u\,\mathbf{a} + v\,\mathbf{b}, \tag{7}$$

where the area element is

$$d\tau = |J|\,du\,dv = \left|\frac{\partial \mathbf{p}}{\partial u} \times \frac{\partial \mathbf{p}}{\partial v}\right| du\,dv = |\mathbf{a} \times \mathbf{b}|\,du\,dv. \tag{8}$$

A structure product over a triangle τ in 3-space can be transformed by a coordinates transformation, as follows:

$$II_\tau^{\alpha\beta\gamma} = \iint_\tau x^\alpha y^\beta z^\gamma\,d\tau = |\mathbf{a} \times \mathbf{b}| \iint_{\tau'} x^\alpha(u,v) y^\beta(u,v) z^\gamma(u,v)\,du\,dv, \tag{9}$$

where τ' is the uv domain that corresponds to τ under the coordinate transformation (7). In this case we have (the proof is given in [**?**]):

$$\begin{aligned} II_\tau^{\alpha\beta\gamma} \;=\; & |\mathbf{a} \times \mathbf{b}| \sum_{h=0}^{\alpha} \binom{\alpha}{h} x_o^{\alpha-h} \sum_{k=0}^{\beta} \binom{\beta}{k} y_o^{\beta-k} \sum_{m=0}^{\gamma} \binom{\gamma}{m} z_o^{\gamma-m} \cdot \\ & \cdot \sum_{i=0}^{h} \binom{h}{i} a_x^{h-i} b_x^i \sum_{j=0}^{k} \binom{k}{j} a_y^{k-j} b_y^j \sum_{l=0}^{m} \binom{m}{l} a_z^{m-l} b_z^l\, II^{\mu\nu}, \end{aligned} \tag{10}$$

where $\mu = (h+k+m) - (i+j+l), \nu = (i+j+l)$, and $II^{\mu\nu}$ is a structure product over the triangle (5), as given by formula (6). Of course the area of a triangle τ is:

$$II_\tau^{000} = \iint_\tau d\tau = |\mathbf{a} \times \mathbf{b}|\,II^{00} = \frac{|\mathbf{a} \times \mathbf{b}|}{2}. \tag{11}$$

Surface integrals In conclusion, a structure product over a polyhedral surface S, open or closed, is a summation of structure products (10) over the 2-simplices of a triangulation K_2 of S:

$$II_S^{\alpha\beta\gamma} = \iint_S x^\alpha y^\beta z^\gamma\,dS = \sum_{\tau \in K_2} II_\tau^{\alpha\beta\gamma}. \tag{12}$$

Volume integration Let P be a three-dimensional polyhedron bounded by a polyhedral surface ∂P. The regularity of the integration domain and the continuity of the integrating function enable us to apply the divergence theorem, which can be briefly summarized, for a vector field $\mathbf{F} = \mathbf{F}(\mathbf{p})$ as:

$$\iiint_P \nabla \cdot \mathbf{F}\,dx\,dy\,dz = \iint_{\partial P} \mathbf{F} \cdot \mathbf{n}\,dS = \sum_{\tau \in K_2} \iint_\tau \mathbf{F} \cdot \mathbf{n}_\tau\,d\tau, \tag{13}$$

where $\mathbf{n}$ is the outward vector normal to the surface portion dS, and hence $\mathbf{n}_\tau = \mathbf{a} \times \mathbf{b}/|\mathbf{a} \times \mathbf{b}|$.

As the function $x^\alpha y^\beta z^\gamma$ equates the divergence of the vector field $\mathbf{F} = (x^{\alpha+1} y^\beta z^\gamma/\alpha + 1, 0, 0)$, an expression for $III_P^{\alpha\beta\gamma}$ is easily derived, which depends only on the 1-simplices of a triangulation of the domain boundary and on the structure products over its 2-simplices.

As a matter of fact, we have:

$$\begin{aligned} III_P^{\alpha\beta\gamma} &= \iiint_P x^\alpha y^\beta z^\gamma \, dx \, dy \, dz \\ &= \iiint_P \frac{\partial}{\partial x} \left(\frac{1}{\alpha+1} x^{\alpha+1} y^\beta z^\gamma \right) dx \, dy \, dz \\ &= \frac{1}{\alpha+1} \sum_{\tau' \in K'_2} (\mathbf{a} \times \mathbf{b})_x \iint_{\tau'} x^{\alpha+1} y^\beta z^\gamma \, du \, dv. \end{aligned} \tag{14}$$

Taking into account equations (8) and (9), we can substitute the integral in the previous equation, getting finally:

$$III_P^{\alpha\beta\gamma} = \frac{1}{\alpha+1} \sum_{\tau \in K_2} \left[\frac{(\mathbf{a} \times \mathbf{b})_x}{|\mathbf{a} \times \mathbf{b}|} \right]_\tau II_\tau^{\alpha+1,\beta,\gamma} \tag{15}$$

where the surface integrals are evaluated by using the formula (10).

Computation of inertia has linear complexity Surface and volume integrals over linear polyhedra are computable in linear time. In particular, *surface and volume integrals of a monomial $x^\alpha y^\beta z^\gamma$ over a linear 2-or 3-polyhedron are computable in $O(\alpha^3\beta^2\gamma^2 E)$ time, E being the number of edges of the polyhedron.*

In fact for the surface and volume integrals it is very easy to see, from inspection of the given equations, that both integrals may be evaluated in $O(\alpha^3\beta^2\gamma^2 T)$ time, T being the number of triangles of a minimal triangulation of the domain. It is easy to show that the relation $T = 2E - 2F < 2E$ holds between the number T of triangles of a minimal triangulation of a polyhedron boundary and the numbers E and F of its original edges and faces respectively. When all triangle faces are triangular, the relation reduces to $T = \frac{2}{3}E$.

This property is very important for Computer-Aided Design and Robotics applications: it demonstrates that the inertia tensor of a linear polyhedral solid is easy to compute. It directly implies that *the inertia tensor of a linear polyhedron is computable in $O(E)$ time.* As a matter of fact the elements of the inertia matrix of a homogeneous object B, namely its *mass* M, *first moments* M_x,M_y,M_z, *products of inertia* M_{xy}, M_{yz}, M_{zx}, and *second moments* M_{xx}, M_{yy}, M_{zz}, can be all expressed as

$$\rho \int_B x^\alpha y^\beta z^\gamma \, dV, \tag{16}$$

where ρ is the constant density, and where $0 \leq \alpha + \beta + \gamma \leq 2$. Being α, β, γ bounded, the assertion follows from the previous claim.

3 Implementation

3.1 High-level interfaces

Surface and volume integrals

⟨ Surface and volume integrals 5 ⟩ ≡

```
    """ Surface and volume integrals """
    def Surface(P, signed=False):
        return II(P, 0, 0, 0, signed)
    def Volume(P):
        return III(P, 0, 0, 0)
```

◇

Macro referenced in 9a.

Terms of the Euler tensor

⟨ Terms of the Euler tensor 6a ⟩ ≡

```
    """ Terms of the Euler tensor """
    def FirstMoment(P):
        out = [None]*3
        out[0] = III(P, 1, 0, 0)
        out[1] = III(P, 0, 1, 0)
        out[2] = III(P, 0, 0, 1)
        return out

    def SecondMoment(P):
        out = [None]*3
        out[0] = III(P, 2, 0, 0)
        out[1] = III(P, 0, 2, 0)
        out[2] = III(P, 0, 0, 2)
        return out

    def InertiaProduct(P):
        out = [None]*3
        out[0] = III(P, 0, 1, 1)
        out[1] = III(P, 1, 0, 1)
        out[2] = III(P, 1, 1, 0)
        return out
```

◇

Macro referenced in 9a.

Vectors and covectors of mechanical interest

⟨ Vectors and covectors of mechanical interest 6b ⟩ ≡

```
""" Vectors and covectors of mechanical interest """
def Centroid(P):
    out = [None]*3
    firstMoment = FirstMoment(P)
    volume = Volume(P)
    out[0] = firstMoment[0]/volume
    out[1] = firstMoment[1]/volume
    out[2] = firstMoment[2]/volume
    return out

def InertiaMoment(P):
    out = [None]*3
    secondMoment = SecondMoment(P)
    out[0] = secondMoment[1] + secondMoment[2]
    out[1] = secondMoment[2] + secondMoment[0]
    out[2] = secondMoment[0] + secondMoment[1]
    return out
```
◇

Macro referenced in 9a.

Basic integration functions

⟨ Basic integration functions 7a ⟩ ≡

```
""" Basic integration functions """
def II(P, alpha, beta, gamma, signed):
    w = 0
    V, FV = P
    for i in range(len(FV)):
        tau = [V[v] for v in FV[i]]
        term = TT(tau, alpha, beta, gamma, signed)
        w += term
    return w

def III(P, alpha, beta, gamma):
    w = 0
    V, FV = P
    for i in range(len(FV)):
        tau = [V[v] for v in FV[i]]
        vo,va,vb = tau
        a = VECTDIFF([va,vo])
        b = VECTDIFF([vb,vo])
        c = VECTPROD([a,b])
        w += (c[0]/VECTNORM(c)) * TT(tau, alpha+1, beta, gamma)
    return w/(alpha + 1)

def M(alpha, beta):
```

```
        a = 0
        for l in range(alpha + 2):
            a += CHOOSE([alpha+1,l]) * POWER([-1,l])/(l+beta+1)
        return a/(alpha + 1)
    ◇
```

Macro referenced in 9a.

The main integration routine

⟨ The main integration routine 7b ⟩ ≡

```
    """ The main integration routine """
    def TT(tau, alpha, beta, gamma, signed=False):
       vo,va,vb = tau
       a = VECTDIFF([va,vo])
       b = VECTDIFF([vb,vo])
       sl = 0;
       for h in range(alpha+1):
          for k in range(beta+1):
             for m in range(gamma+1):
                s2 = 0
                for i in range(h+1):
                   s3 = 0
                   for j in range(k+1):
                      s4 = 0
                      for l in range(m+1):
                         s4 += CHOOSE([m, l]) * POWER([a[2], m-l]) \
                            * POWER([b[2], l]) * M( h+k+m-i-j-l, i+j+l )
                      s3 += CHOOSE([k, j]) * POWER([a[1], k-j]) \
                         * POWER([b[1], j]) * s4
                   s2 += CHOOSE([h, i]) * POWER([a[0], h-i]) * POWER([b[0], i]) * s3;
                sl += CHOOSE ([alpha, h]) * CHOOSE ([beta, k]) * CHOOSE ([gamma, m]) \
                   * POWER([vo[0], alpha-h]) * POWER([vo[1], beta-k]) \
                   * POWER([vo[2], gamma-m]) * s2
       c = VECTPROD([a, b])
       if not signed: return sl * VECTNORM(c)
       elif a[2]==b[2]==0.0: return sl * VECTNORM(c) * SIGN(c[2])
       else: print "error: in signed surface integration"
    ◇
```

Macro referenced in 9a.

Surface integration An exact evaluation of the surface of every cell of a cellular complex may be obtained by integration of the constant scalar field $x^0y^0z^0 = 1$ over the 2D polygonal cells of our hospital model, embedded in the subspace $z = 0$.

The integration over a generic planar polygon, either convex or non-convex, may by computed by summing the integrals over the coherently-oriented *triangles* generated by the

(oriented) edges of the (oriented) polygon and by its first vertex. An implicit assumption on the input: the vertices in each `FV` element (face) are counterclockwise ordered, so that the `triangles` are coherently oriented.

The `Surface` integration function is imported from the `lar-cc` module `integr`. The input LAR `model` is the triple of 2D vertices `V` and the two compressed characteristic matrices `FV` and `EV`, giving the set of vertices incident on every face and edge of the cellular decomposition, respectively. The `cochain` output is the numeric array of surface areas indexed by faces.

⟨ Surface integration 8 ⟩ ≡

```
from integr import *
""" Surface integration """
def surfIntegration(model,signed=False):
    V,FV,EV = model
    V = [v+[0.0] for v in V]
    cochain = []
    for face in FV:
        triangles = AA(C(AL)(face[0]))(TRANS([face[1:-1],face[2:]]))
        P = V,triangles
        area = Surface(P,signed)
        cochain += [area]
    return cochain
```

◇

Macro referenced in 9a.

3.2 Exporting the `integr` module

"larlib/larlib/integr.py" 9a ≡

```
# -*- coding: utf-8 -*-
"""Module for integration of polynomials over 3D volumes and surfaces"""
from larlib import *

⟨ Surface and volume integrals 5 ⟩
⟨ Terms of the Euler tensor 6a ⟩
⟨ Vectors and covectors of mechanical interest 6b ⟩
⟨ Basic integration functions 7a ⟩
⟨ The main integration routine 7b ⟩
⟨ Surface integration 8 ⟩
```

◇

4 Test examples

Integrals on the standard triangle

"test/py/integr/test01.py" 9b ≡

```
""" Integrals on the standard triangle """
from larlib import *

V = [[0,0,0],[1,0,0],[0,1,0]]
FV = [[0,1,2]]
P = (V,FV)
print II(P, 0, 0, 0, False)
◇
```

Integrals on the standard tetrahedron

"test/py/integr/test02.py" 9c ≡

```
""" Integrals on the standard triangle """
from larlib import *

V = [[0.0, 0.0, 0.0], [1.0, 0.0, 0.0], [0.0, 1.0, 0.0], [0.0, 0.0, 1.0]]
FV = [[1,2,3],[0,3,2],[0,1,3],[0,1,2]]
P = (V,FV)
print Volume(P)
◇
```

Integrals on the standard 3D cube Here we give a *triangulation* of the boundary of the unit standard cube in 3D, positively oriented, i.e. with all the triangle normals pointing towards the exterior of the solid cube.

Notice that inverting the boundary orientation changes the sign of the volume, but does not change the centroid.

"test/py/integr/test03.py" 10 ≡

```
""" Integrals on the standard 3D cube """
from larlib import *

V = [[0,0,0],[1,0,0],[0,1,0],[1,1,0],[0,0,1],[1,0,1],[0,1,1],[1,1,1]]

FV = [[1,0,2],[0,1,4],[2,0,4],[1,2,3],[1,3,5],[4,1,5],[3,2,6],[2,4,6],
     [5,3,7],[3,6,7],[4,5,6],[6,5,7]]

P = (V,FV)
print Volume(P)
print Centroid(P)

""" changing the boundary orientation changes the sign of volume,
    but not changes the centroid """

P = (V,AA(REVERSE)(FV))
print Volume(P)
print Centroid(P)
◇
```

Integrals on a non-convex 2D polyline Some problem appear when using the integration method [CP90] on 2D models. In fact, the basic integration formulas refer to the three-variate monomial $x^\alpha y^\beta z^\gamma$, and work even for the integration over 2D domains embedded in 3D, but of course in this case they do not return the integral values as *signed* real numbers, but as positive numbers, since a positive orientation for a open 2D domain in 3D may be defined only locally.

The problem is solved by using the variable `signed`, set to `True` for non-convex polygons embedded in $z = 0$, as shown in the following example.

"test/py/integr/test04.py" 11a ≡

```
""" Integrals on 2D non-convex polyline """
from larlib import *

polyline = TRANS([[10,10,20,40,30,30,15,15],[10,20,30,20,10,15,15,10]])
model = polyline2lar([polyline])
VIEW(POLYLINE(polyline+[polyline[0]]))

V,FV,EV = model
tria = FV[0]
triangles = AA(C(AL)(0))(TRANS([tria[1:-1],tria[2:]]))
V = [v+[0.0] for v in V]
P = V,triangles

area = Surface(P,signed=True)
print "area =",area
```

◇

"test/py/integr/test05.py" 11b ≡

```
""" Integrals on 2D non-convex polyline """
from larlib import *

V,FV,EV = openCourt11.body[0]
tria = FV[0]
triangles = AA(C(AL)(0))(TRANS([tria[1:-1],tria[2:]]))
V = [v+[0.0] for v in V]
P = V,triangles

area = Surface(P,signed=True)
print "area =",area

P = V, AA(REVERSE)(triangles)
area = Surface(P,signed=True)
print "area =",area
```

◇

A Utilities

References

[CL13] CVD-Lab, *Linear algebraic representation*, Tech. Report 13-00, Roma Tre University, October 2013.

[CP90] C. Cattani and A. Paoluzzi, *Boundary integration over linear polyhedra*, Computer-Aided Design **22** (1990), no. 2, 130–135.