Finite domain integration of polynomials *

Alberto Paoluzzi

January 9, 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.

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.

2 Algoritms

3 Implementation

3.1 High-level interfaces

Surface and volume integrals

⟨ Surface and volume integrals 1 ⟩ ≡

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

Macro referenced in 4a.

*This document is part of the *Linear Algebraic Representation with CoChains* (LAR-CC) framework [CL13]. January 9, 2015

Terms of the Euler tensor

⟨ Terms of the Euler tensor 2a ⟩ ≡

```
""" 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 4a.

Vectors and convectors of mechanical interest

⟨ Vectors and convectors of mechanical interest 2b ⟩ ≡

```
""" Vectors and convectors 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 4a.

Basic integration functions

⟨ Basic integration functions 3a ⟩ ≡

```
    """ Basic integration functions """
    def II(P, alpha, beta, gamma):
        w = 0
        V, FV = P
        for i in range(len(FV)):
            tau = [V[v] for v in FV[i]]
            w += T(tau, alpha, beta, gamma)
        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)) * T(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 4a.

The main integration routine

⟨ The main integration routine 3b ⟩ ≡

```
    """ The main integration routine """
    def T(tau, alpha, beta, gamma):
       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]);
    return sl * VECTNORM(c);
◇
```

Macro referenced in 4a.

3.2 Exporting the integr module

"lib/py/integr.py" 4a ≡

```
# -*- coding: utf-8 -*-
"""Module for integration of polynomials over 3D volumes and surfaces"""
from pyplasm import *
import sys; sys.path.insert(0, 'lib/py/')
from lar2psm import *

⟨ Surface and volume integrals 1 ⟩
⟨ Terms of the Euler tensor 2a ⟩
⟨ Vectors and convectors of mechanical interest 2b ⟩
⟨ Basic integration functions 3a ⟩
⟨ The main integration routine 3b ⟩
◇
```

4 Test examples

Integrals on the standard triangle

"test/py/integr/test01.py" 4b ≡

```
""" Integrals on the standard triangle """
import sys; sys.path.insert(0, 'lib/py/')
from integr import *
```

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

Integrals on the standard tetrahedron

"test/py/integr/test02.py" 5a ≡

```
""" Integrals on the standard triangle """
import sys; sys.path.insert(0, 'lib/py/')
from integr 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

"test/py/integr/test03.py" 5b ≡

```
""" Integrals on the standard 3D cube """
import sys; sys.path.insert(0, 'lib/py/')
from integr 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)
```
◇

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.