Computational Algebraic Topology: Lecture 4

Alberto Paoluzzi

March 17, 2017

Nuweb literate tool

Download, Compile and Build

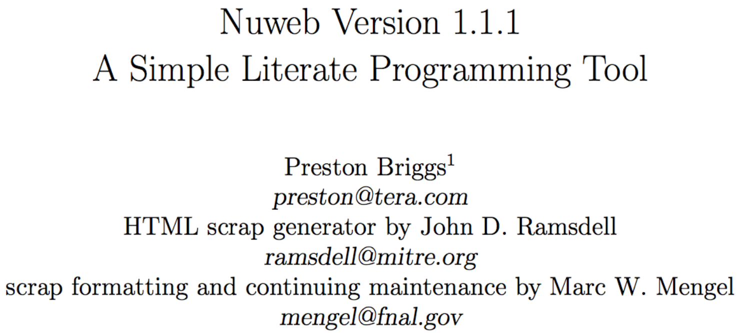
- Download Nuweb from Google code Archive
- Standard install procedure

```
$ cd <downloaded directory>
$ ./configure
$ make
$ sudo make install

$ which nuweb
/usr/local/bin/nuweb
$ nuweb
nuweb: expected a file name.
Usage is: nuweb [-cdmnopstv] file-name...
```

Major and minor commands

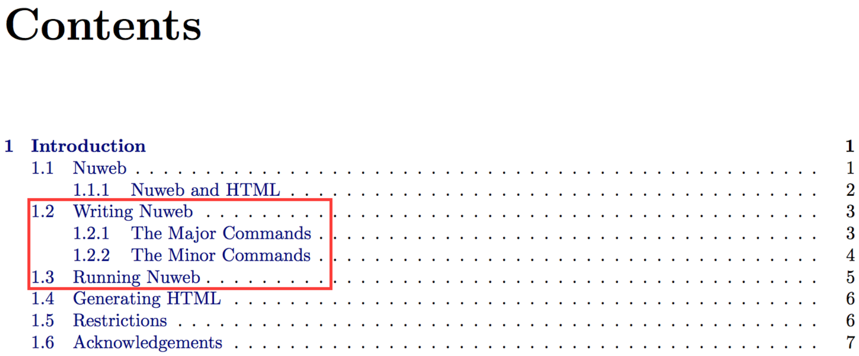
Open nuwebdoc.pdf in `<downloaded directory>`



Nuweb Version 1.1.1
A Simple Literate Programming Tool

Preston Briggs[1]
preston@tera.com
HTML scrap generator by John D. Ramsdell
ramsdell@mitre.org
scrap formatting and continuing maintenance by Marc W. Mengel
mengel@fnal.gov

Major and minor commands

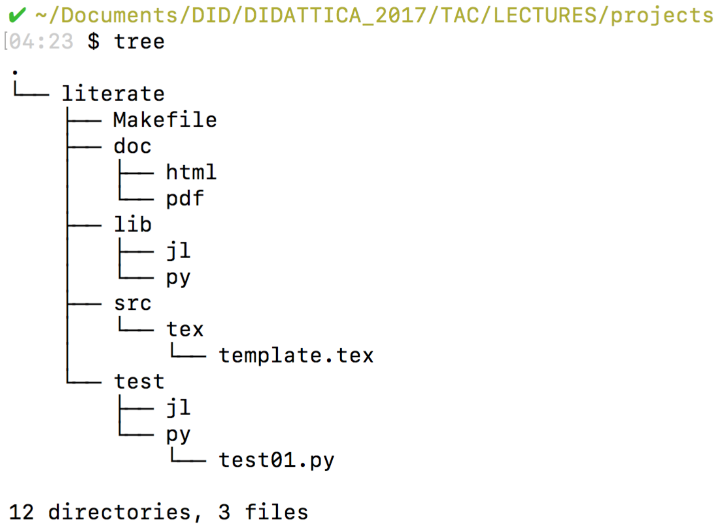


Contents

1 Introduction 1
1.1 Nuweb . . . 1
1.1.1 Nuweb and HTML . . . 2
1.2 Writing Nuweb . . . 3
1.2.1 The Major Commands . . . 3
1.2.2 The Minor Commands . . . 4
1.3 Running Nuweb . . . 5
1.4 Generating HTML . . . 6
1.5 Restrictions . . . 6
1.6 Acknowledgements . . . 7

Figure 2: nuwebdoc

Your directory structure



```
✔ ~/Documents/DID/DIDATTICA_2017/TAC/LECTURES/projects
[04:23 $ tree
.
└── literate
    ├── Makefile
    ├── doc
    │   ├── html
    │   └── pdf
    ├── lib
    │   ├── jl
    │   └── py
    ├── src
    │   └── tex
    │       └── template.tex
    └── test
        ├── jl
        └── py
            └── test01.py

12 directories, 3 files
```

Figure 3: Project's directory structure

Make & Makefile automation tool

Makefile

From Wikipedia

Makefile A `Makefile` is a file containing a set of directives used with the `make` build automation tool.

Makefiles originated on Unix-like systems, and are still a primary software build mechanism in such environments.

Makefiles contain:

- explicit rules,
- implicit rules (via the target of other rules),
- variable definitions,
- directives, and
- comments.

Rules

A makefile consists of "rules" in the following form:

```
target: dependencies
    recipe (system command(s))


. . .
```

A dependency (also called prerequisite) is a file (or files) used as input to create the target.

A recipe may have more than one commands, on subsequent lines, each starting with a `<tab>` character

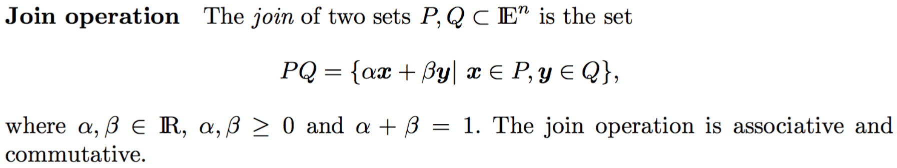
Example: the simplest Makefile (LaTeX)

```
# LaTeX Makefile
#
filename=start
pdf:
    pdflatex ${filename}
    bibtex ${filename}||true
    pdflatex ${filename}
    pdflatex ${filename}
md: html
    pandoc ${filename}.tex -o ${filename}.md
html:
    htlatex ${filename}
read:
    open ${filename}.pdf &
aread:
    acroread ${filename}.pdf &
clean:
    mv ${filename}.tex ${filename}
    rm -f ${filename}.*
    mv ${filename} ${filename}.tex
```

Simplicial Complexes

Join operation

From Chapter 4 of Geometric Programming for Computer-Aided Design (free download from `uniroma3.it` domain)



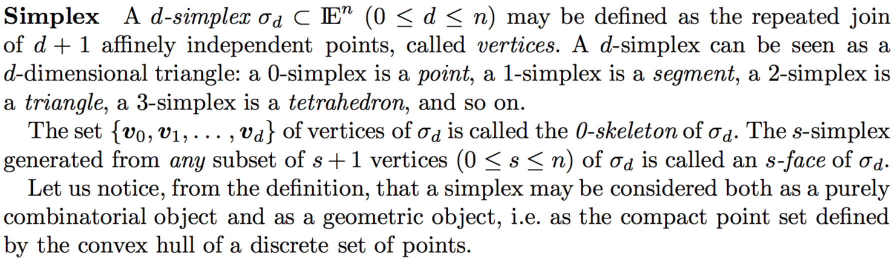
Join operation The *join* of two sets $P, Q \subset \mathbb{E}^n$ is the set

$$PQ = \{\alpha \boldsymbol{x} + \beta \boldsymbol{y} |\ \boldsymbol{x} \in P, \boldsymbol{y} \in Q\},$$

where $\alpha, \beta \in \mathbb{R}$, $\alpha, \beta \geq 0$ and $\alpha + \beta = 1$. The join operation is associative and commutative.

Figure 4: def

Simplex, skeleton, face



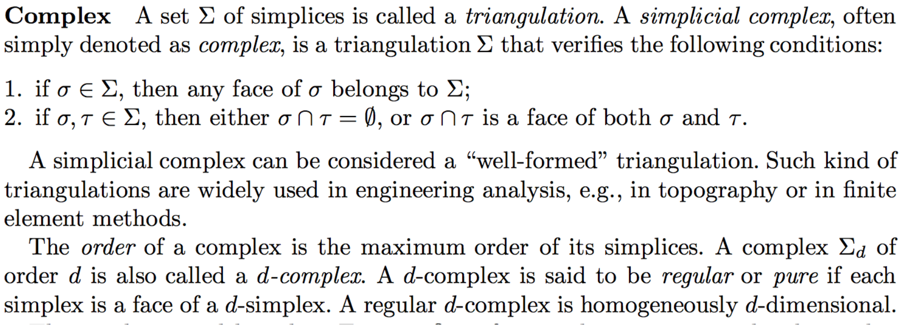
Simplex A *d-simplex* $\sigma_d \subset \mathbb{E}^n$ $(0 \le d \le n)$ may be defined as the repeated join of $d+1$ affinely independent points, called *vertices*. A d-simplex can be seen as a d-dimensional triangle: a 0-simplex is a *point*, a 1-simplex is a *segment*, a 2-simplex is a *triangle*, a 3-simplex is a *tetrahedron*, and so on.

The set $\{\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_d\}$ of vertices of σ_d is called the *0-skeleton* of σ_d. The s-simplex generated from *any* subset of $s+1$ vertices $(0 \le s \le n)$ of σ_d is called an *s-face* of σ_d.

Let us notice, from the definition, that a simplex may be considered both as a purely combinatorial object and as a geometric object, i.e. as the compact point set defined by the convex hull of a discrete set of points.

Figure 5: Simplex, skeleton, face

Triangulation, simplicial complex



Complex A set Σ of simplices is called a *triangulation*. A *simplicial complex*, often simply denoted as *complex*, is a triangulation Σ that verifies the following conditions:

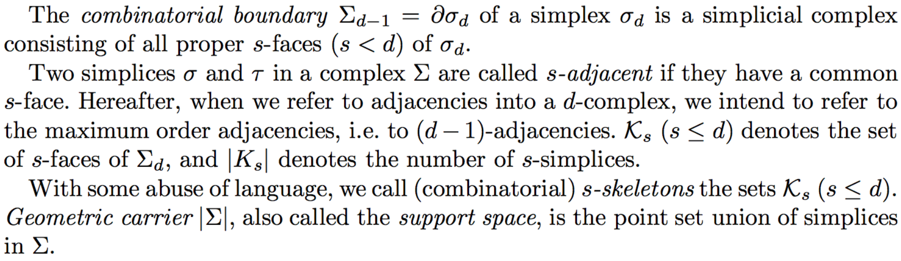
1. if $\sigma \in \Sigma$, then any face of σ belongs to Σ;
2. if $\sigma, \tau \in \Sigma$, then either $\sigma \cap \tau = \emptyset$, or $\sigma \cap \tau$ is a face of both σ and τ.

A simplicial complex can be considered a "well-formed" triangulation. Such kind of triangulations are widely used in engineering analysis, e.g., in topography or in finite element methods.

The *order* of a complex is the maximum order of its simplices. A complex Σ_d of order d is also called a *d-complex*. A d-complex is said to be *regular* or *pure* if each simplex is a face of a d-simplex. A regular d-complex is homogeneously d-dimensional.

Figure 6: Triangulation, simplicial complex

Combinatorial boundary, adjacency, support space



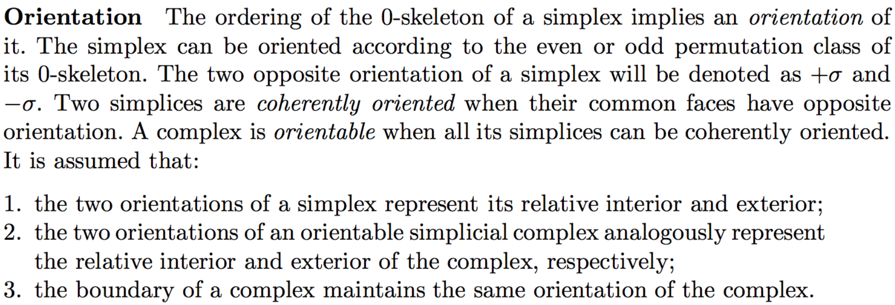
The *combinatorial boundary* $\Sigma_{d-1} = \partial\sigma_d$ of a simplex σ_d is a simplicial complex consisting of all proper s-faces $(s < d)$ of σ_d.

Two simplices σ and τ in a complex Σ are called *s-adjacent* if they have a common s-face. Hereafter, when we refer to adjacencies into a d-complex, we intend to refer to the maximum order adjacencies, i.e. to $(d-1)$-adjacencies. $\mathcal{K}_s$ $(s \leq d)$ denotes the set of s-faces of Σ_d, and $|K_s|$ denotes the number of s-simplices.

With some abuse of language, we call (combinatorial) *s-skeletons* the sets $\mathcal{K}_s$ $(s \leq d)$. *Geometric carrier* $|\Sigma|$, also called the *support space*, is the point set union of simplices in Σ.

Figure 7: Combinatorial boundary, adjacency, support space

Orientation

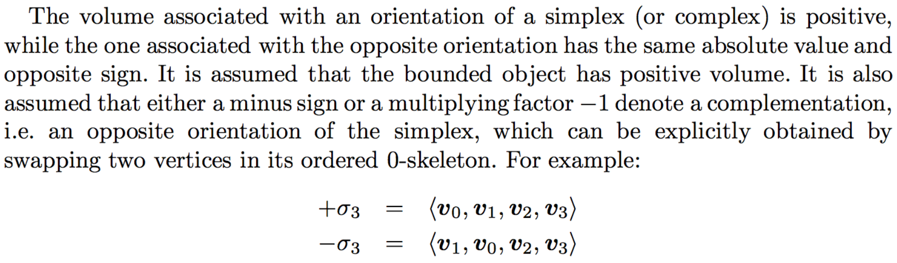


Orientation The ordering of the 0-skeleton of a simplex implies an *orientation* of it. The simplex can be oriented according to the even or odd permutation class of its 0-skeleton. The two opposite orientation of a simplex will be denoted as $+\sigma$ and $-\sigma$. Two simplices are *coherently oriented* when their common faces have opposite orientation. A complex is *orientable* when all its simplices can be coherently oriented. It is assumed that:

1. the two orientations of a simplex represent its relative interior and exterior;
2. the two orientations of an orientable simplicial complex analogously represent the relative interior and exterior of the complex, respectively;
3. the boundary of a complex maintains the same orientation of the complex.

Figure 8: Orientation

Orientation



The volume associated with an orientation of a simplex (or complex) is positive, while the one associated with the opposite orientation has the same absolute value and opposite sign. It is assumed that the bounded object has positive volume. It is also assumed that either a minus sign or a multiplying factor -1 denote a complementation, i.e. an opposite orientation of the simplex, which can be explicitly obtained by swapping two vertices in its ordered 0-skeleton. For example:

$$
\begin{aligned}
+\sigma_3 &= \langle \boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3 \rangle \\
-\sigma_3 &= \langle \boldsymbol{v}_1, \boldsymbol{v}_0, \boldsymbol{v}_2, \boldsymbol{v}_3 \rangle
\end{aligned}
$$

Figure 9: Orientation

Coherently oriented complex

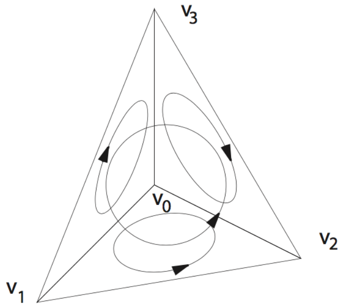
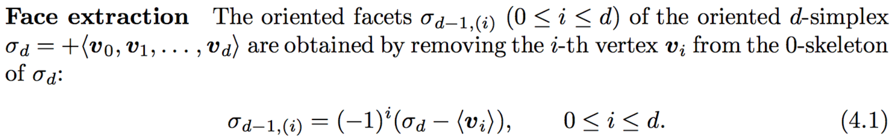


Figure 4.9 Coherent orientation of the faces of a 3-simplex

Figure 10: Coherently oriented complex

Oriented facet extraction

Combinatorial formula

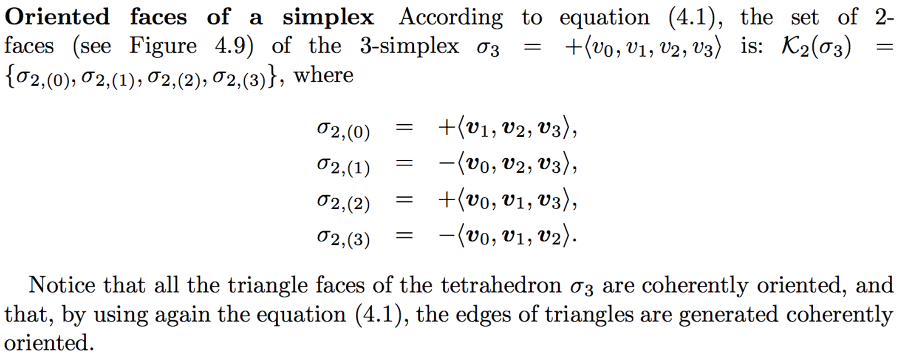


Face extraction The oriented facets $\sigma_{d-1,(i)}$ $(0 \leq i \leq d)$ of the oriented d-simplex $\sigma_d = +\langle \boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_d \rangle$ are obtained by removing the i-th vertex $\boldsymbol{v}_i$ from the 0-skeleton of σ_d:

$$\sigma_{d-1,(i)} = (-1)^i (\sigma_d - \langle \boldsymbol{v}_i \rangle), \qquad 0 \leq i \leq d. \tag{4.1}$$

Figure 11: Facet extraction

Example



Oriented faces of a simplex According to equation (4.1), the set of 2-faces (see Figure 4.9) of the 3-simplex $\sigma_3 = +\langle v_0, v_1, v_2, v_3\rangle$ is: $\mathcal{K}_2(\sigma_3) = \{\sigma_{2,(0)}, \sigma_{2,(1)}, \sigma_{2,(2)}, \sigma_{2,(3)}\}$, where

$$\begin{aligned}
\sigma_{2,(0)} &= +\langle \boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\rangle, \\
\sigma_{2,(1)} &= -\langle \boldsymbol{v}_0, \boldsymbol{v}_2, \boldsymbol{v}_3\rangle, \\
\sigma_{2,(2)} &= +\langle \boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_3\rangle, \\
\sigma_{2,(3)} &= -\langle \boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2\rangle.
\end{aligned}$$

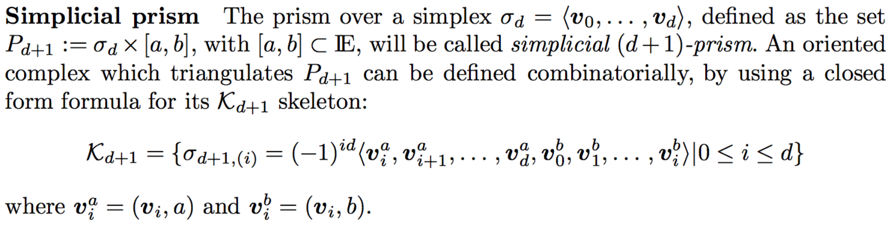
Notice that all the triangle faces of the tetrahedron σ_3 are coherently oriented, and that, by using again the equation (4.1), the edges of triangles are generated coherently oriented.

Figure 12: Example

Simplicial prism

Combinatorial extrusion formula

$(d+1)$-simplices generated by extrusion of a d-simplex



Simplicial prism The prism over a simplex $\sigma_d = \langle \boldsymbol{v}_0, \dots, \boldsymbol{v}_d \rangle$, defined as the set $P_{d+1} := \sigma_d \times [a, b]$, with $[a, b] \subset \mathbb{E}$, will be called *simplicial* $(d+1)$*-prism*. An oriented complex which triangulates P_{d+1} can be defined combinatorially, by using a closed form formula for its $\mathcal{K}_{d+1}$ skeleton:

$$\mathcal{K}_{d+1} = \{\sigma_{d+1,(i)} = (-1)^{id} \langle \boldsymbol{v}_i^a, \boldsymbol{v}_{i+1}^a, \dots, \boldsymbol{v}_d^a, \boldsymbol{v}_0^b, \boldsymbol{v}_1^b, \dots, \boldsymbol{v}_i^b \rangle | 0 \le i \le d\}$$

where $\boldsymbol{v}_i^a = (\boldsymbol{v}_i, a)$ and $\boldsymbol{v}_i^b = (\boldsymbol{v}_i, b)$.

Figure 13: Example

PyPlasm example

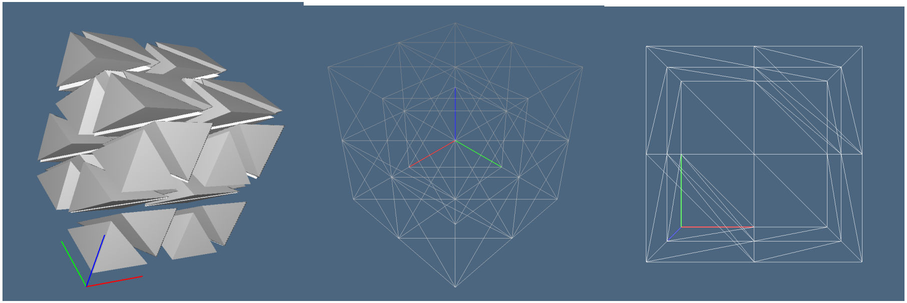
```
from pyplasm import *

s = SIMPLEX(2)
VIEW(s)

p = EXTRUDE([1,s,1])
VIEW(p)
VIEW(SKELETON(1)(p))
```

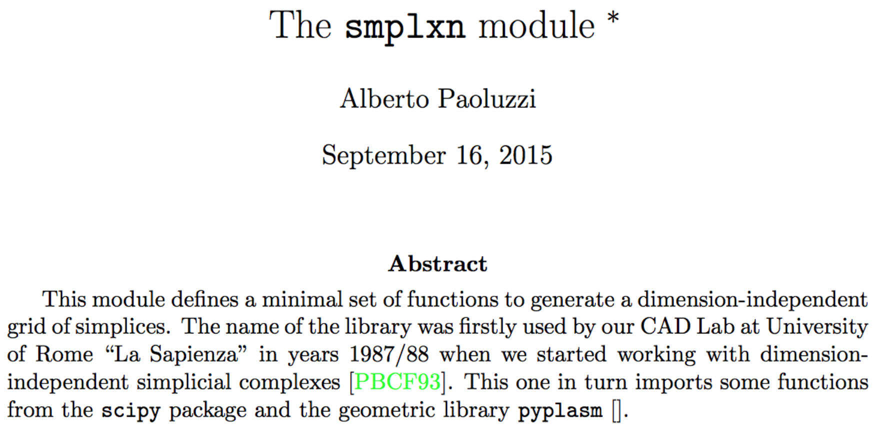
Python implementation in Larlib

Grid of 3-simplices



Simple$_X^n$ module of `Larlib`

Open simplexn.pdf



The **smplxn** module *

Alberto Paoluzzi

September 16, 2015

Abstract

This module defines a minimal set of functions to generate a dimension-independent grid of simplices. The name of the library was firstly used by our CAD Lab at University of Rome “La Sapienza” in years 1987/88 when we started working with dimension-independent simplicial complexes [PBCF93]. This one in turn imports some functions from the **scipy** package and the geometric library **pyplasm** [].

References