Computational topology: Lecture 3

Alberto Paoluzzi

March 7, 2019

1. Line segment intersection

2. Introduction to LAR (Linear Algebraic Representation)

Line segment intersection

Logic and data structures: segments, events, sweep-line

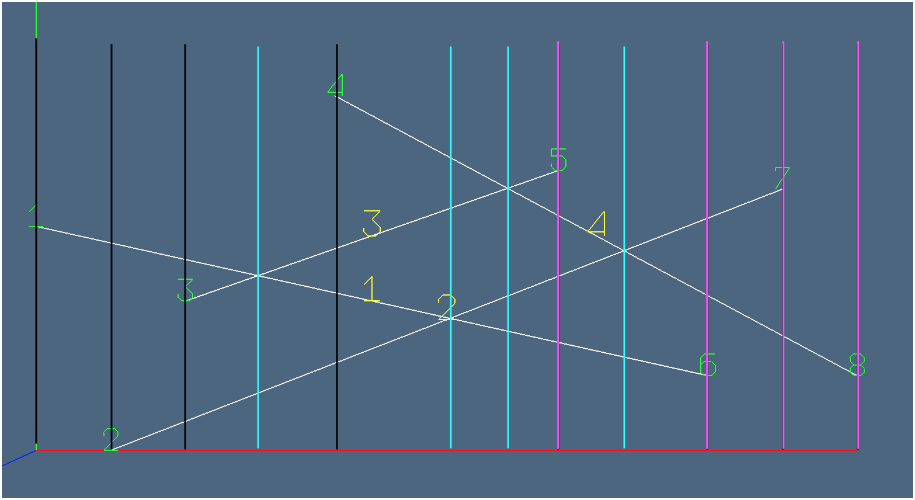


Figure 1: A simple example

Test data preparation

Same as the original article

```
# EXAMPLE 0

# data generation
lines = [[[1,3],[10,5]],[[2,6],[11,2.5]],
[[3,4],[8,2.25]],[[5,1.25],[12,5]]]
V,EV = lines2lar(lines)
Plasm.view(Plasm.numbering(2.)((V,[[[k] for k=1:size(V,2)], EV

# data sorting
V = Plasm.normalize(V,flag=true)
W,EW = presorted(V,EV)
Plasm.view(Plasm.numbering(.25)((W,[[[k] for k=1:size(W,2)], E
```

Pseudocode

```
Initialize event queue ξ = all segment endpoints;
Sort ξ by increasing x and y;
Initialize sweep line SL to be empty;
Initialize output intersection list Λ to be empty;

While (ξ is nonempty) {
    Let E = the next event from ξ;
    If (E is a left endpoint) {
        Let segE = E's segment;
        Add segE to SL;
        Let segA = the segment above segE in SL;
        Let segB = the segment below segE in SL;
        If (I = Intersect( segE with segA) exists)
            Insert I into ξ;
        If (I = Intersect( segE with segB) exists)
            Insert I into ξ;
    }
    Else If (E is a right endpoint) {
        Let segE = E's segment;
        Let segA = the segment above segE in SL;
        Let segB = the segment below segE in SL;
        Remove segE from SL;
        If (I = Intersect( segA with segB) exists)
            If (I is not in ξ already) Insert I into ξ;
    }
    Else {  // E is an intersection event
        Add E to the output list Λ;
        Let segE1 above segE2 be E's intersecting segments in SL;
        Swap their positions so that segE2 is now above segE1;
        Let segA = the segment above segE2 in SL;
        Let segB = the segment below segE1 in SL;
        If (I = Intersect(segE2 with segA) exists)
            If (I is not in ξ already) Insert I into ξ;
        If (I = Intersect(segE1 with segB) exists)
            If (I is not in ξ already) Insert I into ξ;
    }
    remove E from ξ;
}
return Λ;
```

Figure 2: Some examples

Algorithm bootstrap

"docs/algorithm.jl"

docs/algorithm.jl

Same structure than pseudocode, but data structures already choosen

Data structures

aaaaa

DataStructures.jl

latest ▼

Search docs

» DataStructures.jl

Edit on GitHub

DataStructures.jl

This package implements a variety of data structures, including

- Deque (based on block-list)
- CircularBuffer
- CircularDeque (based on a circular buffer)
- Stack
- Queue
- Priority Queue
- Accumulators and Counters
- Disjoint Sets
- Binary Heap
- Mutable Binary Heap
- Ordered Dicts and Sets
- Dictionaries with Defaults
- Trie
- Linked List
- Sorted Dict, Sorted Multi-Dict and Sorted Set
- DataStructures.IntSet

Contents

Data structures

aaaaa

Current status

aaaaa

Current status execution

aaaaa

Introduction to LAR (Linear Algebraic Representation)

aaaa

aaaaa

With increased complexity of geometric data, topological models play an increasingly important role beyond boundary representations, assemblies, finite elements, image processing, and other traditional modeling applications. While many graph- and index-based data structures have been proposed, no standard representation has emerged as of now. Furthermore, such representations typically do not deal with representations of mappings and functions and do not scale to support parallel processing, open source, and client-based architectures. We advocate that a proper mathematical model for all topological structures is a (co)chain complex: a sequence of (co)chain spaces and (co)boundary mappings. This in turn implies all topological structures may be represented by a collection of sparse matrices. We propose a Linear Algebraic Representation (LAR) scheme for mod 2 (co)chain complexes using CSR matrices and show that it supports a variety of topological computations using standard matrix algebra, without any overhead in space or running time. A full open source implementation of LAR is available and is being used for a variety of applications.

Figure 3: Some examples

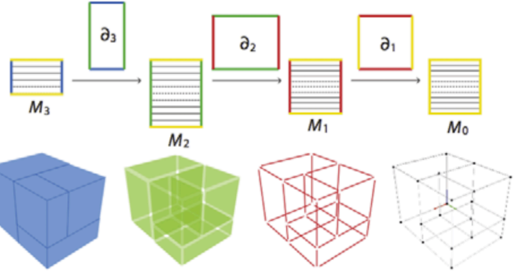
aaaa

aaaaa

A **complex** C is a sequence $\cdots \longrightarrow C_{d+1} \xrightarrow{\partial_{d+1}} C_d \xrightarrow{\partial_d} C_{d-1} \longrightarrow \cdots$

Chain and cochain complex

A chain complex C is a complex of chain spaces and boundary maps:



Unit d-chains (single d-cell subsets), are the standard bases (M_d rows) of d-chain

aaaa

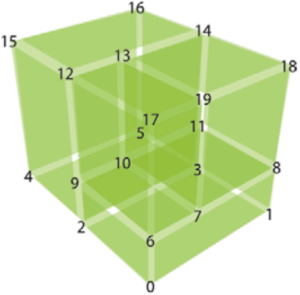
aaaaa

linear spaces C_d and **linear** boundary maps ∂_d, where $\partial_{d+1} \circ \partial_d = 0$, for all d

Characteristic matrices in CSR matrix form

$$M_3 = \left(\begin{array}{c}
0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0 \\
1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1 \\
0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1
\end{array}\right)
\qquad
\left(\begin{array}{l}
2\ 3\ 4\ 5\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16 \\
0\ 1\ 2\ 3\ 6\ 7\ 8\ 9\ 10\ 11 \\
6\ 7\ 9\ 10\ 12\ 13\ 17\ 19 \\
7\ 8\ 10\ 11\ 13\ 14\ 18\ 19
\end{array}\right)$$

$$M_2 = \left(\begin{array}{c}
1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0 \\
0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0 \\
0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1 \\
0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0 \\
0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1 \\
0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1 \\
0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0 \\
0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0 \\
0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1 \\
0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 1
\end{array}\right)
\qquad
\left(\begin{array}{l}
0\ 1\ 2\ 3 \\
0\ 1\ 6\ 7\ 8 \\
0\ 2\ 6\ 9 \\
1\ 3\ 8\ 11 \\
2\ 3\ 4\ 5 \\
2\ 3\ 9\ 10\ 11 \\
2\ 4\ 9\ 12\ 15 \\
3\ 5\ 11\ 14\ 16 \\
4\ 5\ 15\ 16 \\
6\ 7\ 9\ 10 \\
6\ 7\ 17\ 19 \\
6\ 9\ 12\ 17 \\
7\ 8\ 10\ 11 \\
7\ 8\ 18\ 19 \\
7\ 10\ 13\ 19 \\
8\ 11\ 14\ 18 \\
9\ 10\ 12\ 13 \\
10\ 11\ 13\ 14 \\
12\ 13\ 14\ 15\ 16 \\
12\ 13\ 17\ 19 \\
13\ 14\ 18\ 19
\end{array}\right)$$



aces *d*-cells as subsets of vertices

CSR form of a binary matrix

aaaa

aaaaa

aaaa

aaaaa