Boundary operators on LAR [*]

Alberto Paoluzzi

February 10, 2016

Abstract

The various versions of boundary operators on Linear Algebraic Representation of cellular complexes are developed in this module, in order to maintain under focus their proper development, including the possible special cases.

Contents

[*]This document is part of the *Linear Algebraic Representation with CoChains* (LAR-CC) framework [CL13]. February 10, 2016

1 Introduction

In the current LarLib implementation, we have to distinguish between between dimension-independent, dimension-dependent, signed and non-signed operators. Therefore a refactoring of the LarLib code related to boundary/coboundary operators started here, with the aim to provide a precise mathematical definition witin the LAR framework, and to simplify and generalise the related algorithms.

2 Implementation

We start this section by making a distinction between

2.1 Non-signed operators

2.1.1 Dimension-independent

Convex-cells

⟨ convex-cells boundary operator 2a ⟩ ≡

```
""" convex-cells boundary operator """
from larcc import *
def boundary(cells,facets):
    lenV = max(CAT(cells))+1
    csrCV = csrCreate(cells,lenV)
    csrFV = csrCreate(facets,lenV)
    csrFC = matrixProduct(csrFV, csrTranspose(csrCV))
    facetLengths = [csrCell.getnnz() for csrCell in csrCV]
    return csrBoundaryFilter(csrFC,facetLengths)
◇
```

Macro referenced in 3a.

Path-connected cells

⟨ path-connected-cells boundary operator 2b ⟩ ≡

```
""" path-connected-cells boundary operator """
import larcc
from larcc import *
def boundary1(CV,FV,EV):
    lenV = max(CAT(CV))+1
    csrCV = csrCreate(CV,lenV)
    csrFV = csrCreate(FV,lenV)
    csrFC = matrixProduct(csrFV, csrTranspose(csrCV))
    facetLengths = [csrCell.getnnz() for csrCell in csrCV]
    VV = AA(LIST)(range(lenV))
```

```
        csrBBMat = boundary(FV,EV)*boundary(CV,FV)
        FE = larcc.crossRelation(lenV,FV,EV,True)
        return csrBoundaryFilter1(csrBBMat,CV,FV,EV,lenV,FE,csrFC,facetLengths)
    ◇
```

Macro referenced in 3a.

2.2 Signed operators

3 Exporting

"larlib/larlib/boundary.py" 3a ≡

```
    """ boundary operators """
    from larlib import *
    ⟨convex-cells boundary operator 2a⟩
    ⟨path-connected-cells boundary operator 2b⟩
    ◇
```

4 Testing

5 Non-signed operators

"test/py/boundary/test01.py" 3b ≡

```
    """ testing boundary operators """
    from larlib import *

    filename = "test/svg/inters/boundarytest0.svg"
    lines = svg2lines(filename)
    VIEW(STRUCT(AA(POLYLINE)(lines)))

    V,FV,EV,polygons = larFromLines(lines)
    VV = AA(LIST)(range(len(V)))
    submodel = STRUCT(MKPOLS((V,EV)))
    VIEW(larModelNumbering(1,1,1)(V,[VV,EV,FV],submodel,0.2))
    VIEW(EXPLODE(1.2,1.2,1.2)(MKPOLS((V,[EV[e] for e in boundaryCells(FV,EV)],))))
    VIEW(EXPLODE(1.2,1.2,1.2)(MKTRIANGLES((V,FV,EV))))

    boundaryOp = boundary(FV,EV)

    for k in range(1,len(FV)+1):
        faceChain = k*[1]
        BF = chain2BoundaryChain(boundaryOp)(faceChain)
        VIEW(STRUCT(MKPOLS((V,[EV[e] for e in BF]))))
    ◇
```

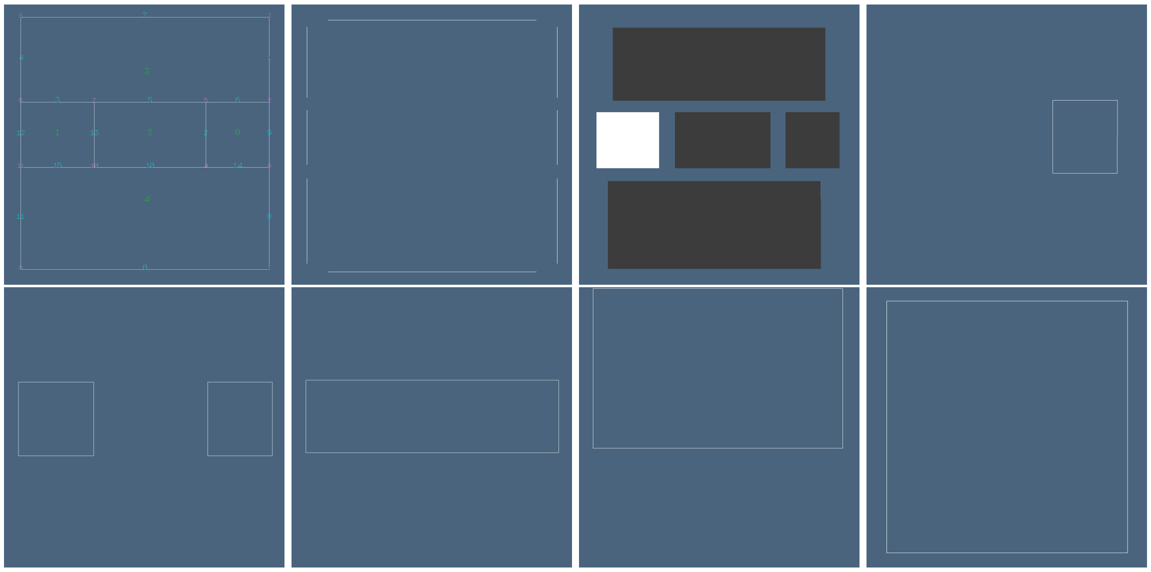

Figure 1: Convex-cell 2-complex. (a) Indexing of 0-,1-,and 2-cells; (b) exploded 2-boundary cells; (c) exploded 2-cells; (d) boundary of a singleton 2-chain; (e–f) boundaries of some 2-chains.

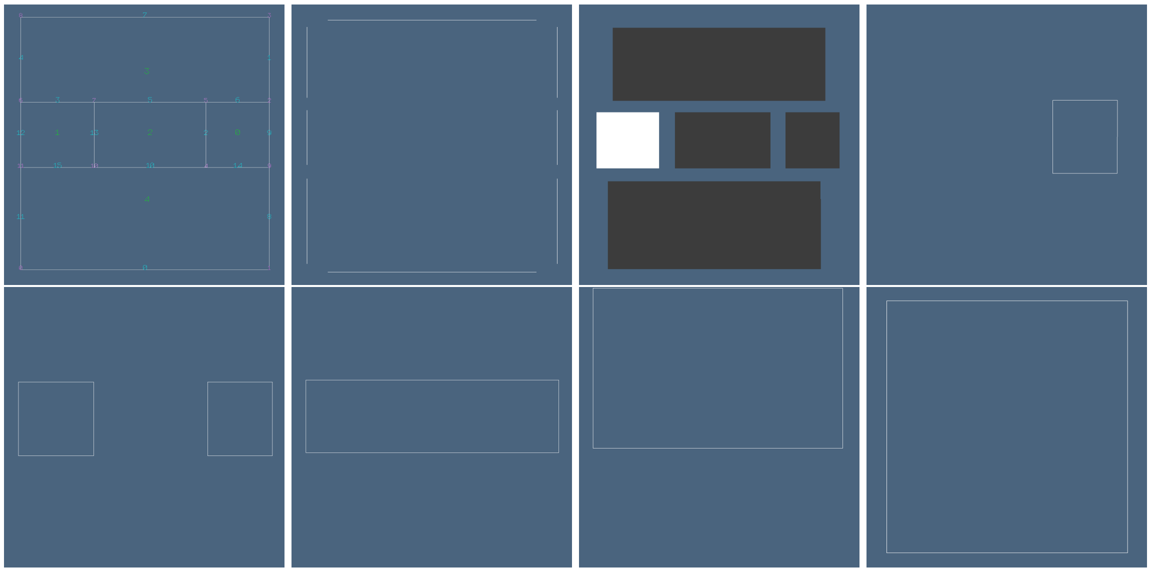

Figure 2: Convex-cell 2-complex. (a) Indexing of 0-,1-,and 2-cells; (b) exploded 2-boundary cells; (c) exploded 2-cells; (d) boundary of a singleton 2-chain; (e–f) boundaries of some 2-chains.

References

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