Curves, surfaces and splines with LAR *

Alberto Paoluzzi

April 24, 2014

Abstract

In this module we implement above LAR most of the parametric methods for polynomial and rational curves, surfaces and splines discussed in the book [**?**], and implemented in the PLaSM language and in the python package pyplasm.

Contents

1 Introduction

2 Transfinite Bézier

@D Multidimensional transfinite Bézier @""" Multidimensional transfinite Bezier """ def larBezier(U,d=3): def BEZIER0(controldata$_fn$) : $N = len(controldata_fn)-1 def map_fn(point)$: $t = U(point) controldata = [fun(point) if callable(fun) else fun for fun in controldata_fn] out =$ $[0.0 for i in range(len(controldata[0]))] for I in range(N+1) : weight = CHOOSE([N,I]) *$

*This document is part of the *Linear Algebraic Representation with CoChains* (LAR-CC) framework [**?**]. April 24, 2014

$math.pow(1-t, N-I) * math.pow(t, I) for K in range(len(out)) : out[K] += weight *$
$(controldata[I][K]) return out return(COMP([AA(COMP), DISTR]))([AA(SEL)(range(d)), map_f n]) return$
def larBezierCurve(controlpoints): dim = len(controlpoints[0]) return larBezier(S1,dim)(controlpoints)
@

3 Coons patches

@D Transfinite Coons patches @""" Transfinite Coons patches """ def larCoonsPatch
(args): $su0_f n, su1_f n, s0v_f n, s1v_f n = argsdef map_f n(point) : u, v = point su0 = su0_f n(point) if callable(su0_f n$
$su1_f n(point) if callable(su1_f n) else su1_f n s0v = s0v_f n(point) if callable(s0v_f n) else s0v_f n s1v =$
$s1v_f n(point) if callable(s1v_f n) else s1v_f n ret = [0.0 for i i in range(len(su0))] for K in range(len(ret)) :$
$ret[K] = ((1-u)*s0v[K]+u*s1v[K]+(1-v)*su0[K]+v*su1[K]+(1-u)*(1-v)*s0v[K]+$
$(1-u)*v*s0v[K]+u*(1-v)*s1v[K]+u*v*s1v[K]) return ret return(COMP([AA(COMP), DISTR]))([[S1, S2,$

4 Computational framework

4.1 Exporting the library

@O lib/py/splines.py @""" Mapping functions and primitive objects """ @¡ Initial import of modules @¿ @¡ Multidimensional transfinite Bézier @¿ @¡ Transfinite Coons patches @¿ @

5 Examples

Some examples of curves @O test/py/splines/test01.py @""" Example of Bezier curve """ import sys """ import modules from larcc/lib """ sys.path.insert(0, 'lib/py/') from splines import *
controlpoints = [[-0,0],[1,0],[1,1],[2,1],[3,1]] dom = larDomain([32],'simplex') obj = larMap(larBezierCurve(c
VIEW(STRUCT(MKPOLS(obj)))
obj = larMap(larBezier(S1,2)(controlpoints))(dom) VIEW(STRUCT(MKPOLS(obj)))
@

Transfinite cubic surface @O test/py/splines/test02.py @""" Example of transfinite surface """ import sys """ import modules from larcc/lib """ sys.path.insert(0, 'lib/py/') from splines import *
dom = larDomain([20],'simplex') C0 = larBezier(S1,3)([[0,0,0],[10,0,0]]) C1 = larBezier(S1,3)([[0,2,0],[8,3,0],
C2 = larBezier(S1,3)([[0,4,1],[7,5,-1],[8,5,1],[12,4,0]]) C3 = larBezier(S1,3)([[0,6,0],[9,6,3],[10,6,-
1]]) dom2D = larExtrude1(dom,20*[1./20]) obj = larMap(larBezier(S2,3)(AA(CONS)([C0,C1,C2,C3])))(dom2I
VIEW(STRUCT(MKPOLS(obj))) @

Coons patch interpolating 4 boundary curves @O test/py/splines/test03.py @"""
Example of transfinite Coons surface """ import sys """ import modules from larcc/lib
""" sys.path.insert(0, 'lib/py/') from splines import *

Su0 = larBezier(S1,3)([[0,0,0],[10,0,0]]) Su1 = larBezier(S1,3)([[0,10,0],[2.5,10,3],[5,10,-
3],[7.5,10,3],[10,10,0]]) Sv0 = larBezier(S2,3)([[0,0,0],[0,0,3],[0,10,3],[0,10,0]]) Sv1 = larBezier(S2,3)([[10,0,0],[10,
dom = larDomain([20],'simplex') dom2D = larExtrude1(dom,20*[1./20]) out = larMap(larCoonsPatch(AA(CO
VIEW(STRUCT(MKPOLS(out))) @

A Utility functions

Initial import of modules @D Initial import of modules @from pyplasm import *
from scipy import * import os,sys """ import modules from larcc/lib """ sys.path.insert(0,
'lib/py/') from lar2psm import * from simplexn import * from larcc import * from largrid
import * from mapper import * @