# SpliPy - Spline modelling in Python


**Kjetil Andre Johannessen, Eivind Fonn**

SINTEF Digital, Dept. of Mathematics and Cybernetics, Trondheim, Norway

Kjetil.Johannessen@sintef.no, Eivind.Fonn@sintef.no


SINTEF

NTNU

## 1. Introduction


**Abstract**

SpliPy is a pure python library for the creation, evaluation and manipulation of B-spline and NURBS geometries. It supports $n$-variate splines of any dimension, but emphasis is made on the use of curves, surfaces and volumes. The library is designed primarily for analysis use, and therefore allows fine-grained control over many aspects which is not possible to achieve with conventional CAD tools.



**Keywords:** NURBS, B-splines, CAD, Interpolation, Approximation


**Installation**

The package is distributed through the Python Package Index (PyPI) and can be installed by typing

```
> pip install splipy
```

into the commandline; or anaconda promt

The current SpliPy release version is 1.3

## 2. B-splines

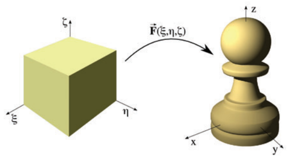
Given a knot vector of nondecreasing knots $\Xi = [\xi_1, \xi_2, \xi_3, ...\xi_{n+p+1}]$ we define the set of $n$ basisis functions by

**The basis**

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi), \qquad (1)$$

and special-casing for $p = 0$-functions

By creating a tensor product of two or three univariate splines weighted by their controlpoints, we are able to create surface and solid representations.



https://youtu.be/vtqdF97JmiM

Fig 1: A trivariate NURBS solid mapping

## 3. Structure

The class follows a simple structure with a Curve, Surface and Volume class which all inherit from a parent SplineObject class. Corresponding to each of these primtitives, we collect a number of generative methods in so-called factory classes.

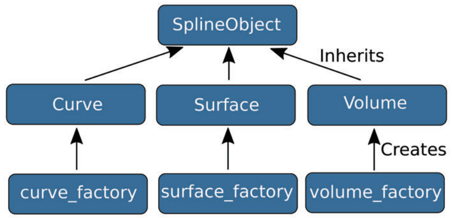



Fig 2: Primary classes and modules

## 4. Examples

Adaptive curve fitting for parametric curves. Uses a posteriori error estimate to refine where needed

```
from splipy import *
from numpy import pi,cos,sin,transpose,array

def trefoil(u):
  x = [    45*sin(u)- 30*cos(2*u)+113*sin(2*u)-11*cos(3*u)+27*sin(3*u),
 41*cos(u)-18*sin(u)- 83*cos(2*u)- 83*sin(2*u)-11*cos(3*u)+27*sin(3*u),
 36*cos(u)+27*sin(u)-113*cos(2*u)+ 30*sin(2*u)+11*cos(3*u)-27*sin(3*u)]
  return transpose(array(x))

knot_curve = curve_factory.fit(trefoil, 0, 2*pi)
```

Sweep operations where one curve is swept along another

```
# the square is scaled by a factor 15
square = 15*curve_factory.n_gon(4)
srf = surface_factory.sweep(knot_curve, square)
```

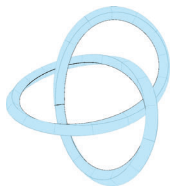


Fig 3: Swept surface

### 4.1 Wind turbine blade

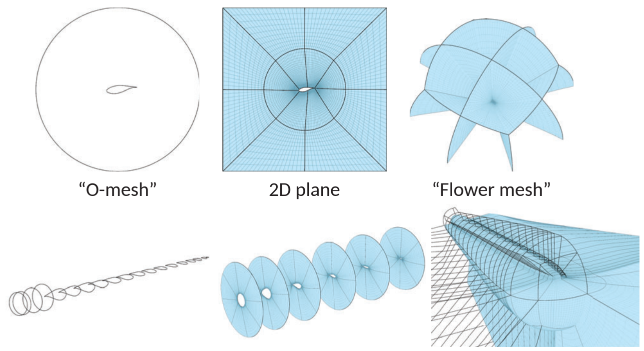



Fig 5: Line-to-volume construction of a full wind-turbine blade mesh.

**Integration with Nutils**

The package contains functions for converting to Nutils objects.

```
from splipy import *
from nutils import *

surf = surface_factory.disc(r=2, type='square')
surf = surf.rebuild(p=3, n=20)
domain, geom = utils.nutils.splipy_to_nutils(surf)
ns = function.Namespace()
ns.x   = geom
ns.phi = domain.basis('spline', degree=2)
A = domain.integrate(ns.eval_ij('phi_i,k phi_j,k'), degree=3)
b = domain.integrate(ns.eval_i('phi_i sin(7 x_0)'), degree=3)
cons = domain.boundary.project(0, ns.phi, ns.x,     degree=3)
ns.w = A.solve(b, constrain=cons)
```

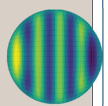


Fig 6: Nutils Solution

**Conclusion**

- SpliPy allows for fast scriptable isogeometric mesh generation
- It is especially suited for smooth lofted geometires, such as turbine blades
- Read more on website: https://github.com/sintefmath/splipy

**Disclaimer:** SpliPy does not contain a graphical user interface. All figures produced on this poster have been created using 3rd party visualizers. Splipy is to be considered an API ready to be integrated into other custom applications.