
splinar Documentation

Loic Gouarin

May 24, 2019

CONTENTS

1	User manual	3
1.1	Cubic Spline	3
2	Tutorial	5
2.1	Splinart on a circle	5
3	Reference manual	9
3.1	splinart	9
4	Indices and tables	13
	Python Module Index	15

splinart is a package used for a tutorial which explains how to do the Python packaging using

- [PyPi](#)
- [conda build](#)
- [pytest](#)
- [Pylint](#)
- [Sphinx](#)

And automate the process to distribute this package using github.

The original idea of splinart is found on the great invonvergent website.

If you want to install splinart:

```
pip install splpinart
```

or:

```
conda install -c gouarin splinart
```


1.1 Cubic Spline

We consider here a cubic spline passing through the points (x_i, y_i) with $a = x_1 < \dots < x_n = b$, that is, a class function $\mathcal{C}^2$ on $[a, b]$ and each restriction at the interval $[x_{i-1}, x_i]$, $1 \leq i \leq n$, is a polynomial of degree less than 3. We will note S such a spline. His equation is given by

$$S_i(x) = Ay_i + By_{i+1} + Cy_i'' + Dy_{i+1}'', \quad x_i \leq x \leq x_{i+1},$$

where

$$A = \frac{x_{i+1} - x}{x_{i+1} - x_i} \quad \text{et} \quad B = \frac{x - x_i}{x_{i+1} - x_i},$$
$$C = \frac{1}{6} (A^3 - A) (x_{i+1} - x_i)^2 \quad \text{et} \quad D = \frac{1}{6} (B^3 - B) (x_{i+1} - x_i)^2.$$

If we derive this equation twice with respect to x , we get

$$\frac{d^2 S(x)}{dx^2} = Ay_i'' + By_{i+1}''.$$

Since $A = 1$ in x_i and $A = 0$ in x_{i+1} and conversely for B , we can see that the second derivative is continuous at the interface of the two intervals $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$.

It remains to determine the expression of y_i'' . To do this, we will calculate the first derivative and impose that it is continuous at the interface of two intervals. The first derivative is given by

$$\frac{dy}{dx} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{3A^2 - 1}{6} (x_{i+1} - x_i) y_i'' + \frac{3B^2 - 1}{6} (x_{i+1} - x_i) y_{i+1}''.$$

We therefore want the value of the first derivative in $x = x_i$ over the interval $[x_{i-1}, x_i]$ to be equal to the value of the first derivative in $x = x_i$ over the interval $[x_i, x_{i+1}]$; which gives us for $i = 2, \dots, n-1$

$$a_i y_{i-1}'' + b_i y_i'' + c_i y_{i+1}'' = d_i,$$

with

$$a_i = \frac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}}$$
$$b_i = 2$$
$$c_i = \frac{x_{i+1} - x_i}{x_{i+1} - x_{i-1}}$$
$$d_i = \frac{6}{x_{i+1} - x_{i-1}} \left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \right).$$

So we have $n - 2$ linear equations to calculate the n unknowns y_i'' for $i = 1, \dots, n$. So we have to make a choice for the first and last values and we will take them equal to zero. We can recognize the resolution of a system with a tridiagonal matrix. It is then easy to solve it by using the algorithm of Thomas which one recalls the principle

$$c'_i = \begin{cases} \frac{c_i}{b_i} & i = 1 \\ \frac{c_i}{b_i - a_i c'_{i-1}} & i = 2, \dots, n. \end{cases}$$

$$d'_i = \begin{cases} \frac{d_i}{b_i} & i = 1 \\ \frac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} & i = 2, \dots, n. \end{cases}$$

The solution is then obtained by the formula

$$\begin{aligned} y_n'' &= d'_n \\ y_i'' &= d'_i - c'_i y_{i+1}'' \quad \text{pour } i = n-1, \dots, 1. \end{aligned}$$

TUTORIAL

2.1 Splinart on a circle

In this tutorial, we will see how to use splinart with a circle.

First of all, we have to create a circle.

```
[34]: import splinart as spl
      center = [.5, .5]
      radius = .3
      theta, path = spl.circle(center, radius)
```

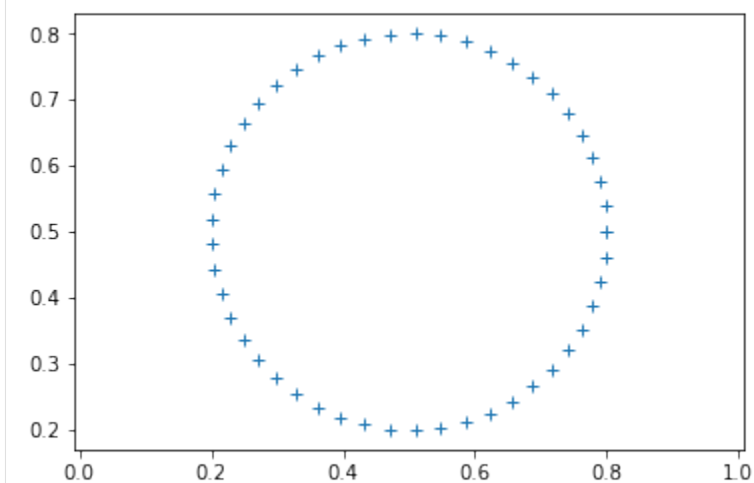
In the previous code, we create a discretization of a circle centered in $[0.5, 0.5]$ with a radius of 0.3. We don't specify the number of discretization points. The default is 30 points.

We can plot the points using matplotlib.

```
[2]: %matplotlib inline
```

```
[6]: import matplotlib.pyplot as plt
      plt.axis("equal")
      plt.plot(path[:, 0], path[:, 1], '+')
```

```
[6]: [<matplotlib.lines.Line2D at 0x7fc91dfe6048>]
```



2.1.1 The sample

In order to compute a sample on a given cubic spline equation, we need to provide a Python function that gives us the x coordinates. We can choose for example.

```
[12]: import numpy as np
def x_func():
    nsamples = 500
    return (np.random.random() + 2 * np.pi * np.linspace(0, 1, nsamples))%(2*np.pi)
```

We can see that the points are chosen between $[0, 2\pi]$ in a random fashion.

2.1.2 The cubic spline

Given a path, we can apply the spline function in order to compute the second derivative of this cubic spline.

```
[13]: yder2 = spl.spline.spline(theta, path)
```

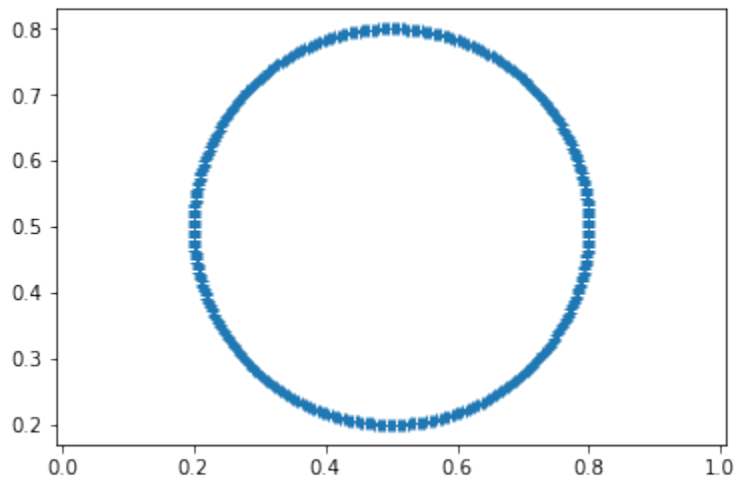
And apply the equation to the sample

```
[14]: xsample = x_func()
ysample = np.zeros((xsample.size, 2))
spl.spline.splint(theta, path, yder2, xsample, ysample)
```

which gives

```
[15]: import matplotlib.pyplot as plt
plt.axis("equal")
plt.plot(ysample[:, 0], ysample[:, 1], '+')
```

```
[15]: [<matplotlib.lines.Line2D at 0x7fc91e05feb8>]
```



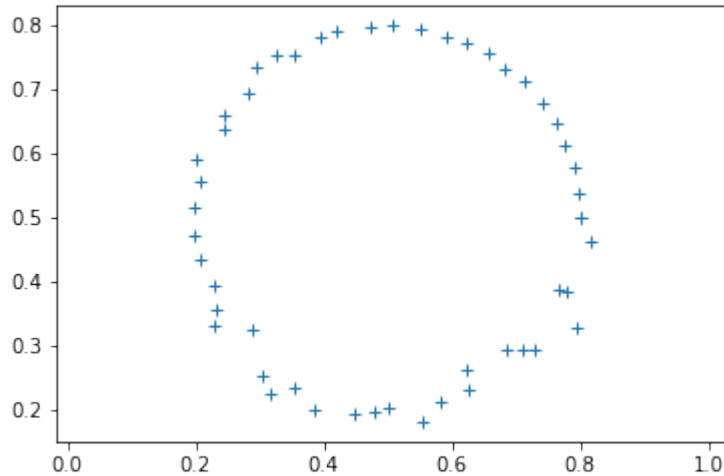
We can see the sample is well defined around the circle that we defined previously.

Now, assume that we move randomly the points of the circle with a small distance.

```
[35]: spl.compute.update_path(path, scale_value=.001, periodic=True)
```

```
[36]: import matplotlib.pyplot as plt
plt.axis("equal")
plt.plot(path[:, 0], path[:, 1], '+')

[36]: [<matplotlib.lines.Line2D at 0x7fc91da92e10>]
```

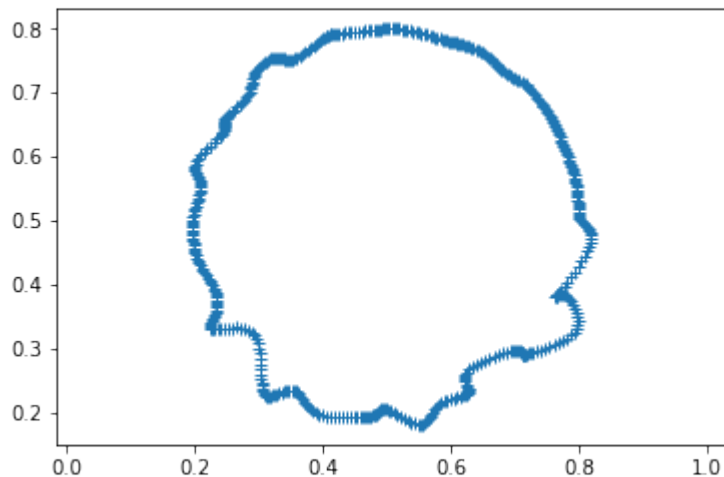


And we compute again the sample of the new cubic spline equation.

```
[37]: yder2 = spl.spline.spline(theta, path)
spl.spline.splint(theta, path, yder2, xsample, ysample)
spl.compute.update_path(path, scale_value=.001, periodic=True)

[39]: import matplotlib.pyplot as plt
plt.axis("equal")
plt.plot(ysample[:, 0], ysample[:, 1], '+')

[39]: [<matplotlib.lines.Line2D at 0x7fc91d94f748>]
```



The circle is deformed.

This is exactly how works splinart. We give a shape and at each step

- we perturb the points of this shape

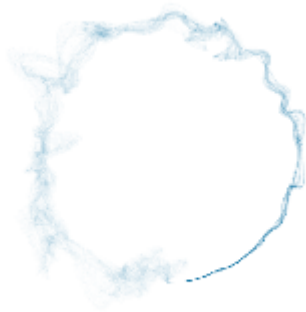
- we compute a sample an this new cubic spline equation
- we add the pixel with a given color on the output image

And we do that several time. We can have the following result

```
[40]: img_size, channels = 1000, 4
      img = np.ones((img_size, img_size, channels), dtype=np.float32)

      theta, path = spl.circle(center, radius)
      spl.update_img(img, path, x_func, nrep=4000, x=theta, scale_value=.00005)
```

```
[41]: spl.show_img(img)
```



```
[ ]:
```

REFERENCE MANUAL

3.1 splinart

3.1.1 splinart package

Subpackages

splinart.shapes package

Submodules

splinart.shapes.base module

Define basic shapes.

`splinart.shapes.base.circle` (*center, radius, npoints=50*)

Discretization of a circle.

Parameters

- **center** (*list(2)*) – 2d coordinates of the center.
- **radius** (*float*) – Radius of the circle.
- **npoints** (*int*) – Number of discretization points (the default value is 50).

Returns

- *np.ndarray* – The theta angle.
- *np.ndarray* – The 2d coordinates of the circle.

`splinart.shapes.base.line` (*begin, end, ypos=0.5, npoints=50*)

Discretization of a horizontal line.

Parameters

- **begin** (*float*) – The left point of the line.
- **end** (*float*) – The right point of the line.
- **ypos** (*float*) – The position of the y coordinate (the default value is 0.5).
- **npoints** (*int*) – Number of discretization points (the default value is 50).

Returns The 2d coordinates of the line.

Return type `np.ndarray`

Module contents

Shape package

splinar.spline package

Submodules

splinar.spline.spline module

Cubic spline

`splinar.spline.spline.spline(xs, ys)`

Return the second derivative of a cubic spline.

Parameters

- **xs** (*np.ndarray*) – The x coordinate of the cubic spline.
- **ys** (*np.ndarray*) – The y coordinate of the cubic spline.

Returns The second derivative of the cubic spline.

Return type *np.ndarray*

splinar.spline.splint module

Integration of a cubic spline.

`splinar.spline.splint.splint(xs, ys, y2s, x, y)`

Evaluate a sample on a cubic spline.

Parameters

- **xs** – The x coordinates of the cubic spline.
- **ys** – The y coordinates of the cubic spline.
- **y2s** – The second derivative of the cubic spline.
- **x** – The sample where to evaluation the cubic spline.
- **y** – The y coordinates of the sample.

See also:

splinar.spline.spline()

Module contents

Spline package

Submodules

splinar.color module

Define the default color of the output.

splinart.compute module

Material to update the output image using a cubic spline equation.

```
splinart.compute.update_img(img, path, xs_func, x=None, nrep=300, periodic=True,
                             scale_color=0.005, color=(0.0, 0.41568627450980394,
                             0.6196078431372549, 1.0), scale_value=1e-05)
```

Update the image using a cubic spline on a shape.

Parameters

- **img** (*np.ndarray*) – The output image.
- **path** (*np.ndarray*) – The y coordinate of the cubic spline if x is not None, the coordinates of the cubic spline if x is None.
- **x** (*np.ndarray*) – The x coordinates of the cubic spline if given. (the default value is None)
- **xs_func** (*function*) – The function that return the x coordinate of the sampling points where to compute the y coordinates given the spline equation.
- **nrep** (*int*) – Number of iteration (default is 300).
- **periodic** (*bool*) – Define if the first and last points of the path must be equal (default is True).
- **scale_color** (*float*) – Scale the given color (default is 0.005).
- **color** (*list(4)*) – Define the RGBA color to plot the spline.
- **scale_value** (*float*) – Rescale the random radius (default value is 0.00001).

See also:

[`update\_path\(\)`](#)

```
splinart.compute.update_path(path, periodic=False, scale_value=1e-05)
```

Update the path of the spline.

We move each point of the path by a random vector defined inside a circle where

- the center is the point of the path
- the radius is a random number between [-1, 1]

Parameters

- **path** (*np.ndarray*) – The y coordinate of the cubic spline.
- **periodic** (*bool*) – If True, the first and the last points of the path are the same (the default value is False).
- **scale_value** (*float*) – Rescale the random radius (default value is 0.00001).

splinart.draw module

Material to update the image with given points and save or plot this image.

```
splinart.draw.draw_pixel(img, xs, ys, scale_color=0.0005, color=(0.0, 0.41568627450980394,
0.6196078431372549, 1.0))
```

Add pixels on the image.

Parameters

- **img** (*np.ndarray*) – The image where we add pixels.
- **xs** (*np.ndarray*) – The x coordinate of the pixels to add.
- **ys** (*np.ndarray*) – The y coordinate of the pixels to add.
- **scale_color** (*float*) – Scale the given color (default is 0.0005).
- **color** (*list(4)*) – Define the RGBA color of the pixels.

`splinarth.draw.save_img(img, path, filename)`

Save the image in a png file.

Parameters

- **img** (*np.ndarray*) – The image to save.
- **path** (*str*) – The save directory.
- **filename** (*str*) – The file name with the png extension.

`splinarth.draw.show_img(img)`

Plot the image using matplotlib.

Parameters **img** (*np.ndarray*) – The image to save.

splinarth.version module

Module contents

Splinarth package

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

S

- `splinar`, [12](#)
- `splinar.color`, [10](#)
- `splinar.compute`, [11](#)
- `splinar.draw`, [11](#)
- `splinar.shapes`, [10](#)
- `splinar.shapes.base`, [9](#)
- `splinar.spline`, [10](#)
- `splinar.spline.spline`, [10](#)
- `splinar.spline.splint`, [10](#)
- `splinar.version`, [12](#)

INDEX

C

`circle()` (in module *splinar.shapes.base*), 9

D

`draw_pixel()` (in module *splinar.draw*), 11

L

`line()` (in module *splinar.shapes.base*), 9

S

`save_img()` (in module *splinar.draw*), 12

`show_img()` (in module *splinar.draw*), 12

`splinar` (module), 12

`splinar.color` (module), 10

`splinar.compute` (module), 11

`splinar.draw` (module), 11

`splinar.shapes` (module), 10

`splinar.shapes.base` (module), 9

`splinar.spline` (module), 10

`splinar.spline.spline` (module), 10

`splinar.spline.splint` (module), 10

`splinar.version` (module), 12

`spline()` (in module *splinar.spline.spline*), 10

`splint()` (in module *splinar.spline.splint*), 10

U

`update_img()` (in module *splinar.compute*), 11

`update_path()` (in module *splinar.compute*), 11