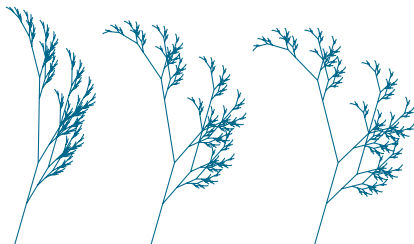


# Learning SciPy for Numerical and Scientific Computing

Francisco Blanco-Silva

University of South Carolina



# SOME BASIC PRINCIPLES IN SCIENTIFIC COMPUTING

## A PERSONAL PERSPECTIVE

**Scientific Computing** is concerned with constructing mathematical models and quantitative analysis techniques, and using computers to analyze and solve scientific problems.

# SOME BASIC PRINCIPLES IN SCIENTIFIC COMPUTING

## A PERSONAL PERSPECTIVE

**Scientific Computing** is concerned with constructing mathematical models and quantitative analysis techniques, and using computers to analyze and solve scientific problems.

- ▶ **Trust the Mathematicians:** High-level Mathematics do solve challenging research problems in *simple* ways—even if you don't understand why or how yet!

# SOME BASIC PRINCIPLES IN SCIENTIFIC COMPUTING

## A PERSONAL PERSPECTIVE

**Scientific Computing** is concerned with constructing mathematical models and quantitative analysis techniques, and using computers to analyze and solve scientific problems.

- ▶ **Trust the Mathematicians:** High-level Mathematics do solve challenging research problems in *simple* ways—even if you don't understand why or how yet!
- ▶ **Trust the Engineer, Biologist, Chemist, Physicist, ...:** There are different ways to solve any problem. Rather than dismiss a different point of view, embrace it, work it out, explore the source of the problem, and look for connections with other techniques.

# SOME BASIC PRINCIPLES IN SCIENTIFIC COMPUTING

## A PERSONAL PERSPECTIVE

**Scientific Computing** is concerned with constructing mathematical models and quantitative analysis techniques, and using computers to analyze and solve scientific problems.

- ▶ **Trust the Mathematicians:** High-level Mathematics do solve challenging research problems in *simple* ways—even if you don't understand why or how yet!
- ▶ **Trust the Engineer, Biologist, Chemist, Physicist, ...:** There are different ways to solve any problem. Rather than dismiss a different point of view, embrace it, work it out, explore the source of the problem, and look for connections with other techniques.
- ▶ **Trust the Computer Scientist:** Writing low-level code from scratch seldom guarantees best results.

# SOME BASIC PRINCIPLES IN SCIENTIFIC COMPUTING

## A PERSONAL PERSPECTIVE

**Scientific Computing** is concerned with constructing mathematical models and quantitative analysis techniques, and using computers to analyze and solve scientific problems.

- ▶ **Trust the Mathematicians:** High-level Mathematics do solve challenging research problems in *simple* ways—even if you don't understand why or how yet!
- ▶ **Trust the Engineer, Biologist, Chemist, Physicist, ...:** There are different ways to solve any problem. Rather than dismiss a different point of view, embrace it, work it out, explore the source of the problem, and look for connections with other techniques.
- ▶ **Trust the Computer Scientist:** Writing low-level code from scratch seldom guarantees best results.
- ▶ **Find a reliable way to communicate** through software.

# SOME BASIC PRINCIPLES IN SCIENTIFIC COMPUTING

## A PERSONAL PERSPECTIVE

**Scientific Computing** is concerned with constructing mathematical models and quantitative analysis techniques, and using computers to analyze and solve scientific problems.

- ▶ **Trust the Mathematicians:** High-level Mathematics do solve challenging research problems in *simple* ways—even if you don't understand why or how yet!
- ▶ **Trust the Engineer, Biologist, Chemist, Physicist, ...:** There are different ways to solve any problem. Rather than dismiss a different point of view, embrace it, work it out, explore the source of the problem, and look for connections with other techniques.
- ▶ **Trust the Computer Scientist:** Writing low-level code from scratch seldom guarantees best results.
- ▶ **Find a reliable way to communicate** through software.
- ▶ **Big guns:** Solving these problems usually require massive amounts of calculations and are often executed on supercomputers or distributed computing platforms.

# COMPUTATIONAL RESOURCES

## SOFTWARE

- ▶ Optimized Algebra Libraries
  - ▶ BLAS
  - ▶ LAPACK



# COMPUTATIONAL RESOURCES

## SOFTWARE

- ▶ Optimized Algebra Libraries
  - ▶ BLAS
  - ▶ LAPACK
- ▶ Computer Algebra systems
  - ▶ MATLAB
  - ▶ Mathematica
  - ▶ SciLab
  - ▶ GNU Octave

# COMPUTATIONAL RESOURCES

## SOFTWARE

- ▶ Optimized Algebra Libraries
  - ▶ BLAS
  - ▶ LAPACK
- ▶ Computer Algebra systems
  - ▶ MATLAB
  - ▶ Mathematica
  - ▶ SciLab
  - ▶ GNU Octave
- ▶ Computer Languages
  - ▶ C
  - ▶ Fortran
  - ▶ Perl
  - ▶ R
  - ▶ julia
  - ▶ Python

# COMPUTATIONAL RESOURCES

## SOFTWARE

- ▶ **Optimized Algebra Libraries**
  - ▶ BLAS
  - ▶ LAPACK
- ▶ **Computer Algebra systems**
  - ▶ MATLAB
  - ▶ Mathematica
  - ▶ SciLab
  - ▶ GNU Octave
- ▶ **Computer Languages**
  - ▶ C
  - ▶ Fortran
  - ▶ Perl (with PDL)
  - ▶ R
  - ▶ julia
  - ▶ Python

# COMPUTATIONAL RESOURCES

## SOFTWARE

- ▶ Optimized Algebra Libraries

- ▶ BLAS
- ▶ LAPACK

- ▶ Computer Algebra systems

- ▶ MATLAB
- ▶ Mathematica
- ▶ SciLab
- ▶ GNU Octave

- ▶ Computer Languages

- ▶ C
- ▶ Fortran
- ▶ Perl (with PDL)
- ▶ R
- ▶ julia
- ▶ Python

- ▶ with PyLab: `ipython + NumPy + SciPy + matplotlib`

# COMPUTATIONAL RESOURCES

## SOFTWARE

- ▶ Optimized Algebra Libraries

- ▶ BLAS
- ▶ LAPACK

- ▶ Computer Algebra systems

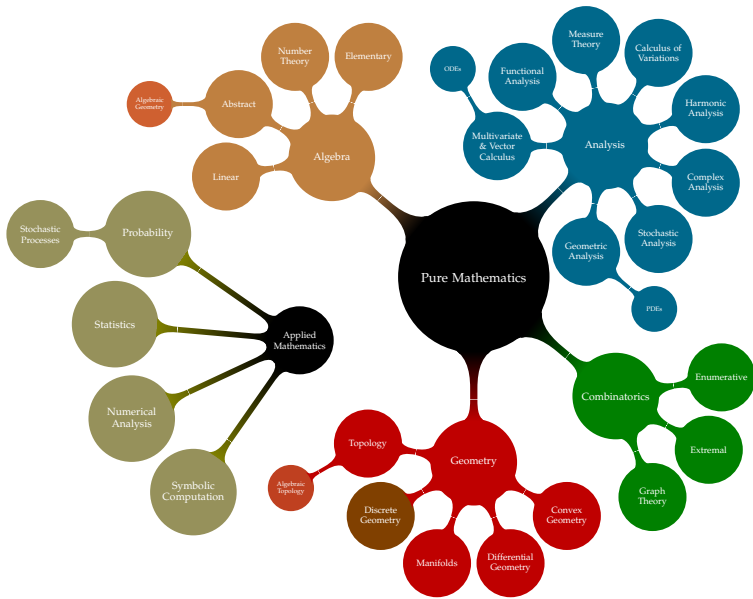
- ▶ MATLAB
- ▶ Mathematica
- ▶ SciLab
- ▶ GNU Octave

- ▶ Computer Languages

- ▶ C
- ▶ Fortran
- ▶ Perl (with PDL)
- ▶ R
- ▶ julia
- ▶ Python
  - ▶ with PyLab: `ipython + NumPy + SciPy + matplotlib`
  - ▶ with `scikits` and `Pandas` on top of that

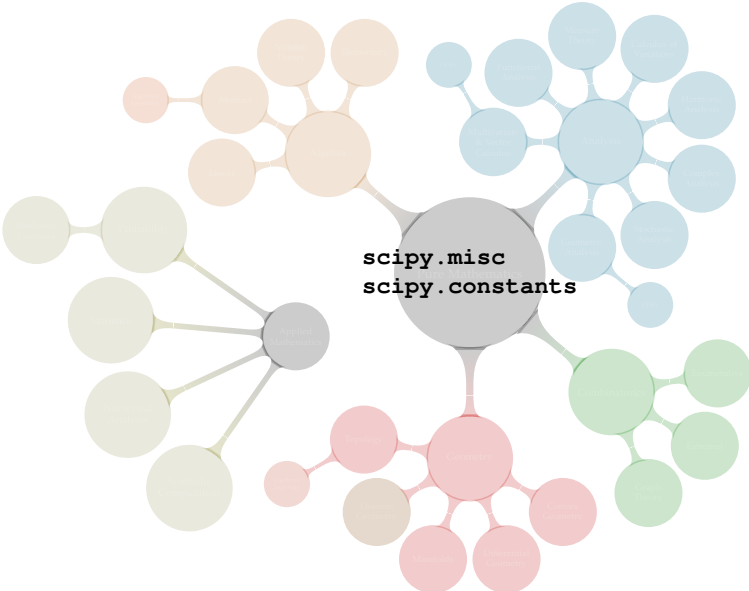
# THE STRUCTURE OF SCIPY

SIMILARITY TO THE DIFFERENT AREAS OF MATHEMATICS



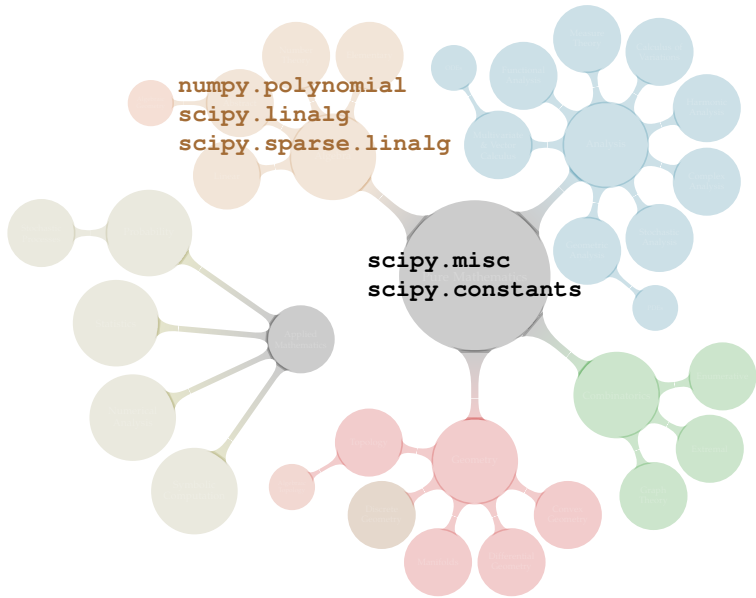
# THE STRUCTURE OF SCIPLY

## SIMILARITY TO THE DIFFERENT AREAS OF MATHEMATICS



# THE STRUCTURE OF SCIPY

SIMILARITY TO THE DIFFERENT AREAS OF MATHEMATICS



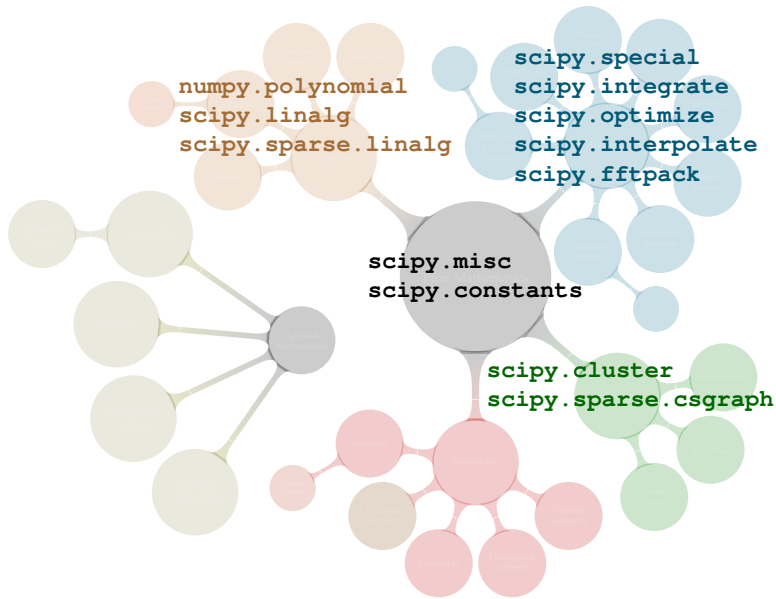


# THE STRUCTURE OF SCIPY

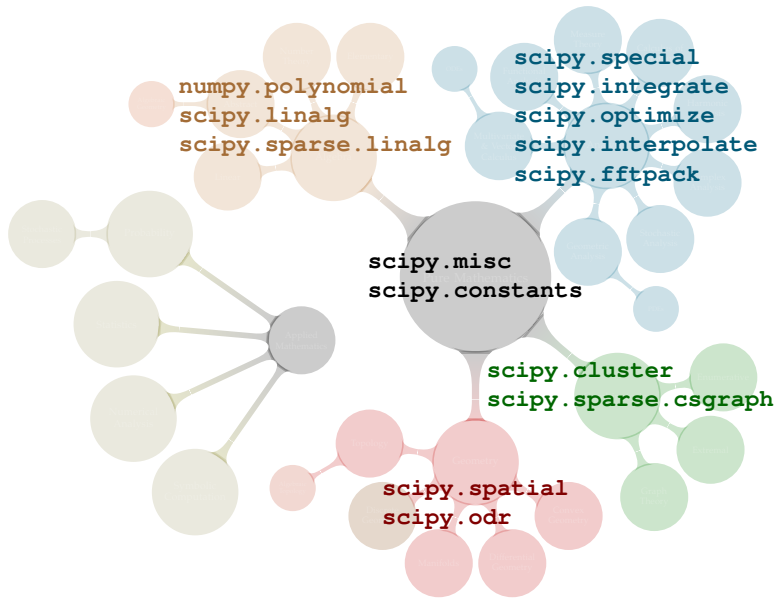
SIMILARITY TO THE DIFFERENT AREAS OF MATHEMATICS



## SIMILARITY TO THE DIFFERENT AREAS OF MATHEMATICS

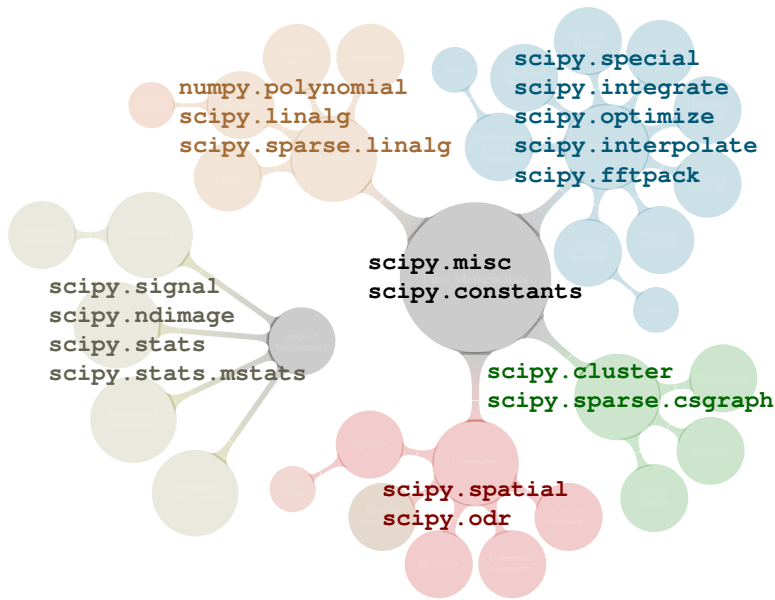


## SIMILARITY TO THE DIFFERENT AREAS OF MATHEMATICS



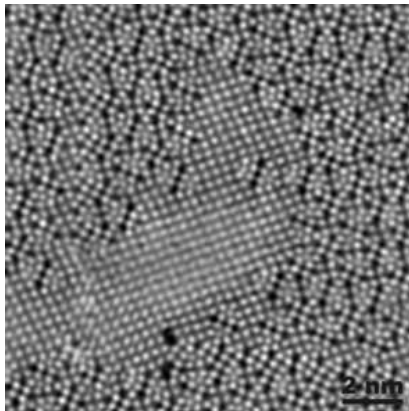
# THE STRUCTURE OF SCIPY

SIMILARITY TO THE DIFFERENT AREAS OF MATHEMATICS



## SCI-PY IN ACTION

EXTRACT THE STRUCTURAL MODEL OF A MOLECULE OF  $Nb_4W_{13}O_{47}$

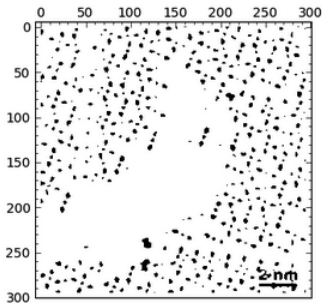


# SCI-PY IN ACTION

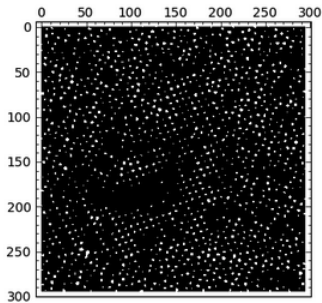
## COMPUTATION OF STRUCTURAL MODELS

We take the following (naïve) approach:

- **Segmentation** of the atoms by thresholding and morphological operations.



`img>0.2`



`img>0.7`

# SCIPY IN ACTION

## COMPUTATION OF STRUCTURAL MODELS

We take the following (naïve) approach:

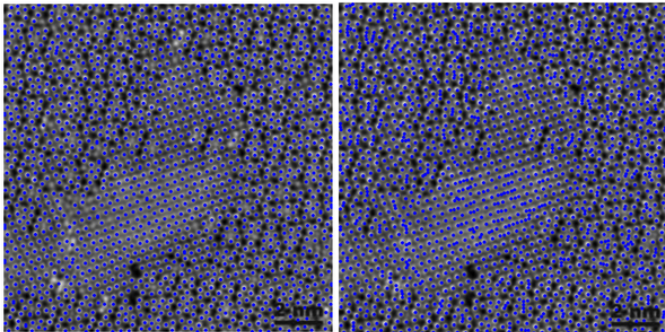
- ▶ **Segmentation** of the atoms by thresholding and morphological operations.
- ▶ **Connected component labeling** to extract each atom for posterior examination.

# SCIPLY IN ACTION

## COMPUTATION OF STRUCTURAL MODELS

We take the following (naïve) approach:

- ▶ **Segmentation** of the atoms by thresholding and morphological operations.
- ▶ **Connected component labeling** to extract each atom for posterior examination.
- ▶ **Computation of the centers of mass** of each label identified as an atom.



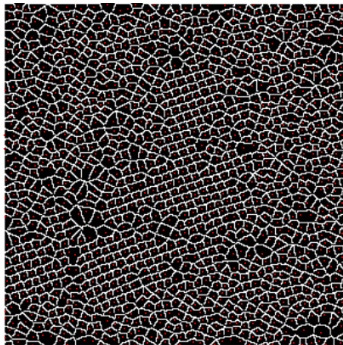


# SCIPLY IN ACTION

## COMPUTATION OF STRUCTURAL MODELS

We take the following (naïve) approach:

- ▶ **Segmentation** of the atoms by thresholding and morphological operations.
- ▶ **Connected component labeling** to extract each atom for posterior examination.
- ▶ **Computation of the centers of mass** of each label identified as an atom.
- ▶ **Computation of the Voronoi diagram** of the lattice formed by the previous points.



# SCI·PY IN ACTION

## COMPUTATION OF STRUCTURAL MODELS

```
1  # Preamble
2  import numpy
3  import scipy
4  from scipy.ndimage import binary_opening, label, center_of_mass,
    distance_transform_edt
5  # Load the image
6  img = scipy.misc.imread('NbW-STEM.png')
7  # Apply a threshold to segment atoms
8  BWatoms = (img > 0.62)
9  # Perform a binary operation to eliminate outliers
10 BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
11 # Segmentation of each atom
12 structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
13 segmentation, segments = label(BWatoms, structuring_element)
14 # Computation of centers of mass of each atom
15 coords = center_of_mass(img, segmentation, range(1, segments+1))
16 xcoords = array([x[0] for x in coords])
17 ycoords = array([x[1] for x in coords])
18 # Compute the Voronoi diagram of the lattice
19 L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
    return_indices=True)
20 Voronoi = segmentation[L1,L2]
```

# SCIPLY IN ACTION

## COMPUTATION OF STRUCTURAL MODELS

```
1  # Preamble
2  import numpy
3  import scipy
4  from scipy.ndimage import binary_opening, label, center_of_mass,
    distance_transform_edt
5  # Load the image
6  img = scipy.misc.imread('NbW-STEM.png')
7  # Apply a threshold to segment atoms
8  BWatoms = (img > 0.62)
9  # Perform a binary operation to eliminate outliers
10 BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
11 # Segmentation of each atom
12 structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
13 segmentation, segments = label(BWatoms, structuring_element)
14 # Computation of centers of mass of each atom
15 coords = center_of_mass(img, segmentation, range(1, segments+1))
16 xcoords = array([x[0] for x in coords])
17 ycoords = array([x[1] for x in coords])
18 # Compute the Voronoi diagram of the lattice
19 L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
    return_indices=True)
20 Voronoi = segmentation[L1,L2]
```

# SCI·PY IN ACTION

## COMPUTATION OF STRUCTURAL MODELS

```
1  # Preamble
2  import numpy
3  import scipy
4  from scipy.ndimage import binary_opening, label, center_of_mass,
    distance_transform_edt
5  # Load the image
6  img = scipy.misc.imread('NbW-STEM.png')
7  # Apply a threshold to segment atoms
8  BWatoms = (img > 0.62)
9  # Perform a binary operation to eliminate outliers
10 BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
11 # Segmentation of each atom
12 structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
13 segmentation,segments = label(BWatoms, structuring_element)
14 # Computation of centers of mass of each atom
15 coords = center_of_mass(img, segmentation, range(1,segments+1))
16 xcoords = array([x[0] for x in coords])
17 ycoords = array([x[1] for x in coords])
18 # Compute the Voronoi diagram of the lattice
19 L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
    return_indices=True)
20 Voronoi = segmentation[L1,L2]
```

# SCI·PY IN ACTION

## COMPUTATION OF STRUCTURAL MODELS

```
1  # Preamble
2  import numpy
3  import scipy
4  from scipy.ndimage import binary_opening, label, center_of_mass,
    distance_transform_edt
5  # Load the image
6  img = scipy.misc.imread('NbW-STEM.png')
7  # Apply a threshold to segment atoms
8  BWatoms = (img > 0.62)
9  # Perform a binary operation to eliminate outliers
10 BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
11 # Segmentation of each atom
12 structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
13 segmentation, segments = label(BWatoms, structuring_element)
14 # Computation of centers of mass of each atom
15 coords = center_of_mass(img, segmentation, range(1, segments+1))
16 xcoords = array([x[0] for x in coords])
17 ycoords = array([x[1] for x in coords])
18 # Compute the Voronoi diagram of the lattice
19 L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
    return_indices=True)
20 Voronoi = segmentation[L1,L2]
```

# SCI·PY IN ACTION

## COMPUTATION OF STRUCTURAL MODELS

```
1  # Preamble
2  import numpy
3  import scipy
4  from scipy.ndimage import binary_opening, label, center_of_mass,
    distance_transform_edt
5  # Load the image
6  img = scipy.misc.imread('NbW-STEM.png')
7  # Apply a threshold to segment atoms
8  BWatoms = (img > 0.62)
9  # Perform a binary operation to eliminate outliers
10 BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
11 # Segmentation of each atom
12 structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
13 segmentation,segments = label(BWatoms, structuring_element)
14 # Computation of centers of mass of each atom
15 coords = center_of_mass(img, segmentation, range(1,segments+1))
16 xcoords = array([x[0] for x in coords])
17 ycoords = array([x[1] for x in coords])
18 # Compute the Voronoi diagram of the lattice
19 L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
    return_indices=True)
20 Voronoi = segmentation[L1,L2]
```

# SCI·PY IN ACTION

## COMPUTATION OF STRUCTURAL MODELS

```
1  # Preamble
2  import numpy
3  import scipy
4  from scipy.ndimage import binary_opening, label, center_of_mass,
    distance_transform_edt
5  # Load the image
6  img = scipy.misc.imread('NbW-STEM.png')
7  # Apply a threshold to segment atoms
8  BWatoms = (img > 0.62)
9  # Perform a binary operation to eliminate outliers
10 BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
11 # Segmentation of each atom
12 structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
13 segmentation, segments = label(BWatoms, structuring_element)
14 # Computation of centers of mass of each atom
15 coords = center_of_mass(img, segmentation, range(1, segments+1))
16 xcoords = array([x[0] for x in coords])
17 ycoords = array([x[1] for x in coords])
18 # Compute the Voronoi diagram of the lattice
19 L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
    return_indices=True)
20 Voronoi = segmentation[L1,L2]
```

# SCI·PY IN ACTION

## COMPUTATION OF STRUCTURAL MODELS

```
1  # Preamble
2  import numpy
3  import scipy
4  from scipy.ndimage import binary_opening, label, center_of_mass,
    distance_transform_edt
5  # Load the image
6  img = scipy.misc.imread('NbW-STEM.png')
7  # Apply a threshold to segment atoms
8  BWatoms = (img > 0.62)
9  # Perform a binary operation to eliminate outliers
10 BWatoms = binary_opening(BWatoms, structure=numpy.ones((2,2)))
11 # Segmentation of each atom
12 structuring_element = [[0,1,0],[1,1,1],[0,1,0]]
13 segmentation,segments = label(BWatoms, structuring_element)
14 # Computation of centers of mass of each atom
15 coords = center_of_mass(img, segmentation, range(1,segments+1))
16 xcoords = array([x[0] for x in coords])
17 ycoords = array([x[1] for x in coords])
18 # Compute the Voronoi diagram of the lattice
19 L1, L2 = distance_transform_edt(segmentation==0, return_distances=False,
    return_indices=True)
20 Voronoi = segmentation[L1,L2]
```



## FOR MORE INFORMATION, EXAMPLES, IDEAS, ...

**Francisco Blanco-Silva**  
Useless math is useful math. It can be used to generate more useless math.

About me Curriculum Vitae Research Teaching Problem Solving LaTeX

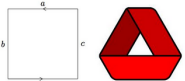
Type text to search here

### Möbius bands and Klein bottles

Go to comments Leave a comment Edit

Consider a long and narrow strip of paper, in which we glue the two narrower edges together after performing a half-twist of the strip, hence forming a funny-looking loop. Topologically, this is equivalent to identifying two parallel edges of a square in the proper way. Therefore, we can see the Möbius band as the quotient space of  $\mathbb{C}_2$  with the following equivalence relation: Given  $(x_1, x_2), (y_1, y_2) \in \mathbb{C}_2$ , we write  $(x_1, x_2) \sim (y_1, y_2)$  if

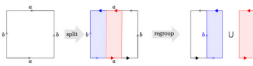
- $x_1 = y_1$  and  $x_2 = y_2$ , or
- $x_2 = -1$  and  $x_1 = 1 - y_1$ .



In a similar fashion, we construct a new surface by identification of two parallel edges of a square to form a cylinder, and a posterior identification of the remaining two edges (which turned into two circles after the first identification). The second identification will be performed after a "half-twist" (which unfortunately can only be realized in a fourth dimension!). The equivalence relation in the square  $\mathbb{C}_2$  is as follows: Given  $(x_1, x_2), (y_1, y_2) \in \mathbb{C}_2$ , we define  $(x_1, x_2) \sim (y_1, y_2)$  provided one of the following conditions are satisfied:

- $x_1 = y_1$  and  $x_2 = y_2$ , or
- $x_1 y_1 = -1$  and  $x_2 = y_2$ , or
- $x_2 y_2 = -1$  and  $x_1 = 1 - y_1$ .


A diagram representing this quotient space—which we denote  $\mathbb{K}$  and call Klein bottle—is shown below, together with an interesting way to split and recover the space: Notice how by cutting these strips in the manner shown, and adjoining two of the strips through the proper edge, we can see the Klein's bottle as the union of two Möbius bands.



**Miscellaneous**

All the images included in this presentation are generated with the aid of the package tikz in

blancosilva.wordpress.com



COMMUNITY EXPERIENCE DISTILLED

## Learning SciPy for Numerical and Scientific Computing

A practical tutorial that guarantees fast, accurate, and easy-to-code solutions to your numerical and scientific computing problems with the power of SciPy and Python

Francisco J. Blanco-Silva

**PACKT** open source  
PUBLISHING