foliation-sim

Release 1.0.0

Florian Babisch

Sep 04, 2022

CONTENTS:

1 foliation-sim **1**

1.1 foliationSimulation module 1

1.2 tangentSimulation module 11

2 Indices and tables **13**

Python Module Index **15**

Index **17**

CHAPTER

ONE

FOLIATION-SIM

1.1 foliationSimulation module

class foliationSimulation.**ClassRoots**

Bases: object

bisect(*f*, *x1*, *x2*, *switch=0*, *epsilon=1e-09*)

roots(*f*, *a*, *b*, *eps=1e-06*)

rootsearch(*f*, *a*, *b*, *dx*)

foliationSimulation.**angle_at_sign_flip**(*surface*)

Returns the angle between the two normal vectors of a 1-surface at which the x-component changes sign.

surface
[tuple] The x- and y-coordinates of a 1-surface.

float
Angle between two normal vectors that have different sign in the x-component.

foliationSimulation.**calculate_self_intersections**(*surface*)

Find all the points where a 1-surface self-intersects.

Parameters
surface (*tuple*) – x- and y-coordinates of some surface in 2 dimensional space.

Returns
x- and y-coordinates of self-intersections.

Return type
tuple

foliationSimulation.**circle**(*t*, *center*, *radius*)

Calculate values of a circle with given radius centered around a given point.

Parameters

- **t** (*list*) – Values that parametrize the circle.
- **center** (*tuple*) – Point around which the circle is centered.
- **radius** (*float*) – Radius of the circle.

Returns
The x- and y-values of the points of the unit circle.

Return type
tuple

foliationSimulation.**closest_value**(*list*, *value*)

Returns value closest to the input value in a given list.

Parameters

- **lst** (*list*) – List of numbers.
- **value** (*float*) – Value of which the closest element in the list is searched for.

Returns
Index of element of array that is closest to the given value and the corresponding value in the list.

Return type
tuple

foliationSimulation.**curvature**(*func*)

Calculate signed curvature of the given function.

Parameters
func (*SymPy type*) – Function of which the curvature is calculated.

Returns
Signed curvature of given function.

Return type
SymPy type

foliationSimulation.**draw_centers_osc_circ**(*surface*, *func*, *xValues*, *connectedCircles*)

Draw centers of osculating circles and print the minimal radius from all osculating circles, the point at which the radius is minimal and the corresponding center of the osculating circle. The printed values are rounded to three decimal places.

Note: Only centers with x- and y-value smaller than the calculated xmax and ymax value are drawn.

Parameters

- **surface** (*tuple*) – The x- and y-coordinates of the given surface of which the osculating circles are calculated.
- **func** (*SymPy type*) – Function that characterizes the given surface.
- **valueLimit** (*integer*) – Maximal x-value, which is used to determine the largest allowed value for the x- and y-coordinate of the centers of osculating circles that will still be drawn.
- **xValues** (*list*) – List of x-Values.
- **connectedCircles** (*boolean*) – If True the centers of osculating circles are plotted with the plot function, which yields a connected line. Otherwise they are ploted with the scatter function, which gives separate points.

foliationSimulation.**draw_curvature**(*xValues*, *func*)

Draws signed curvature.

Parameters

- **xValues** (*list*) – List of x-coordinates.
- **func** (*SymPy type*) – Function of which the curvature is drawn.

foliationSimulation.**draw_extrema**(*interval*, *func*)

Draws points where given function has extrema.

Parameters

- **interval** (*list*) – First entry is lower interval limit and second entry is uper interval limit. This gives the interval in which extrema are searched for.
- **func** (*SymPy type*) – Function for which the extrema are searched for.

foliationSimulation.**draw_modified_surfaces**(*func*, *valueLimit*, *numberOfSurfaces*, *spacing*, *initialSurface*, *unitNormalVectors*, *direction*, *sameSurfaceColor*)

Draws modified surfaces, that is the surfaces where the swallow tail is removed. It also prints the parameter interval of the swallow tails that are drawn.

Parameters

- **func** (*SymPy expression*) – Function that characterizes the initial surface.
- **valueLimit** (*integer*) – Value for left and right interval limit for the x-coordinates.
- **numberOfSurfaces** (*integer*) – The number of Surfaces that are drawn.
- **spacing** (*integer*) – The distance between two surfaces is by standard 1. Using this parameter the distance can be scaled as 1 / spacing.
- **initialSurface** (*tuple*) – The x- and y-coordinates of the initial surface.
- **unitNormalVectors** (*tuple*) – The x- and y-coorindates of the unit normal vectors of the initial surface.
- **direction** (*integer*) – Is either +1 or -1 depending on direction in which the initial surfaces is pushed along the normals.
- **sameSurfaceColor** (*boolean*) – Whether the surfaces are drawn with the same color or with individual colors.

foliationSimulation.**draw_normics**(*s*, *initialSurface*, *unitNormalVectors*, *normicSpacing*)

Draws the normal lines of the initial surface.

Parameters

- **s** (*list*) – Values for which the normal line is drawn along the normal starting from the initial surface.
- **initialSurface** (*tupel*) – The x- and y-coordinates of the initial surface.
- **unitNormalVectors** (*tupel*) – The x- and y-coordinates of the unit normal vectors of the initial surface.
- **normicSpacing** (*integer*) – Only every n-th element of the lists of coordinates is used to draw the normal lines.

foliationSimulation.**draw_self_intersections**(*numberOfSurfaces*, *spacing*, *initialSurface*, *unitNormalVectors*, *direction*)

Draw all points where evolved surfaces self-intersect and print the x- and y-values of the self-intersection.

Parameters

- **numberOfSurfaces** (*integer*) – The number of Surfaces that are drawn.
- **spacing** (*integer*) – The distance between two surfaces is by standard 1. Using this parameter the distance can be scaled as 1 / spacing.
- **initialSurface** (*tuple*) – The x- and y-coordinates of the initial surface.

- **unitNormalVectors** (*tuple*) – The x- and y-coorindates of the unit normal vectors of the initial surface.
- **direction** (*integer*) – Is either +1 or -1 depending on direction in which the initial surfaces is pushed along the normals.

foliationSimulation.**draw_surfaces**(*numberOfSurfaces*, *spacing*, *initialSurface*, *unitNormalVectors*, *direction*, *sameSurfaceColor*)

Draws the surfaces that arise when pushing the initial surface along the normal lines.

Parameters

- **numberOfSurfaces** (*integer*) – The number of Surfaces that are drawn.
- **spacing** (*integer*) – The distance between two surfaces is by standard 1. Using this parameter the distance can be scaled as 1 / spacing.
- **surface** (*initial*) – The x- and y-coordinates of the initial surface.
- **unitNormalVectors** (*tuple*) – The x- and y-coorindates of the unit normal vectors of the initial surface.
- **direction** (*integer*) – Is either +1 or -1 depending on direction in which the initial surfaces is pushed along the normals.
- **sameSurfaceColor** (*boolean*) – Whether the surfaces are drawn with the same color or with individual colors.

foliationSimulation.**draw_tangents**(*xValues*, *func*, *s*)

Draws the tangent vector to a surface.

Note: It might look like the tangent vector is not actually tangent. This is due to scaling of the plot and is something that has to be fixed.

Parameters

- **xValues** (*list*) – List of x-coordinates.
- **func** (*SymPy expression*) – Function of which the tangent vector is calculated.
- **s** (*float*) – Parameter value of the surface.

foliationSimulation.**drawing**(*inputFunction*, *inputSaveFileName*, *valueLimit=1*, *numberOfFutureSurfaces=0*, *numberOfPastSurfaces=0*, *spacing=1*, *posXLim=1*, *negXLim=1*, *posYLim=1*, *negYLim=1*, *normicSpacing=1*, *tangent_xValues=0*, *metric=True*, *showCentersOscCircle=False*, *connectedCenters=True*, *showNormics=False*, *showSurface=True*, *changeLimits=False*, *tangentVector=False*, *showSelfIntersections=False*, *showTails=False*, *sameSurfaceColor=True*, *saveFile=False*)

Is the method regularly called by the interact method. Modifies the initial figure by giving it a title etc., handles the user inputs, controls the different choices, displays the plot.

Parameters

- **inputFunction** (*str*) – User input that is used as the function that characterizes the initial surface.
- **inputSaveFileName** (*str*) – Name that is used when saving a plot.
- **valueLimit** (*integer*) – Interval limit for the x-values.

- **numberOfFutureSurfaces** (*integer*) – Number of surfaces that are drawn into the 'future', i.e., that have direction +1.
- **numberOfPastSurfaces** (*integer*) – Number of surfaces that are drawn into the 'future', i.e., that have direction -1.
- **spacing** (*integer*) – The distance between two surfaces is by standard 1. Using this parameter the distance can be scaled as 1 / spacing.
- **posXLim** (*integer*) – Limit for positive x-axis.
- **negXLim** (*integer*) – Limit for negative x-axis.
- **posYLim** (*integer*) – Limit for positive y-axis.
- **negYLim** (*integer*) – Limit for negative y-axis.
- **normicSpacing** (*integer*) – Inverse density of normal lines that are drawn. If normicSpacing = n, then only every nth normal line is drawn.
- **tangent_xValues** (*integer*) – The x-value that parameterizes the surface at which the tangent vector is drawn.
- **showCentersOscCircle** (*boolean*) – If True the centers of the osculating circles of the initial surface are drawn, otherwise they are not drawn.
- **connectedCircles** (*boolean*) – If True the centers of osculating circles are plotted with the plot function, which yields a connected line. Otherwise they are ploted with the scatter function, which gives separate points.
- **showNormics** (*boolean*) – If True the normal lines are drawn, otherwise they are not drawn.
- **showSurface** (*boolean*) – If True the surfaces that arise from pushing the initial surface along the normal lines are drawn, otherwise they are not drawn.
- **changeLimits** (*boolean*) – If True the limits of the axes are changed according to the settings of the user, otherwise the axes change according to the size of the plot.
- **tangeVector** (*boolean*) – If True the tangent vector of a surface at the chosen parameter x-value is drawn, otherwise the tangent vector is not drawn.
- **showSelfIntersections** (*boolean*) – If True then at the points of self-intersection a red dot is drawn, otherwise no red dots are drawn.
- **showTails** (*boolean*) – If True the tails of the swallow tails are shown drawn, otherwise they are removed by the draw_modified_surfaces method if possible (see note of that method).
- **sameSurfaceColor** (*boolean*) – If True all surfaces are drawn with the same color (blue), otherwise every surface has an individual color.
- **saveFile** (*boolean*) – If True the plot that it shown after pressing the 'Run interact' button is saved under the name written in the 'Name of Image' textbox.

foliationSimulation.**euclidean_angle_between_vectors**(*v*, *w*)

Calculates angle between two 2 dimensional vectors in Euclidean space.

Parameters

- **v** (*tuple*) – Vector with two components.
- **w** (*tuple*) – Vector with two components.

Returns

Angle between the vectors v and w.

Return type
float

foliationSimulation.**euclidean_inner_product**(*v*, *w*)

Calculates the Eucldiean inner product of two 2 dimensional vectors.

Parameters

- **v** (*tuple*) – Vector with two components.
- **w** (*tuple*) – Vector with two components.

Returns
Inner product of v and w.

Return type
float

foliationSimulation.**euclidean_norm**(*vector*)

Calculates Euclidean norm of 2 dimensional vector.

Parameters
vector (*tuple*) – Vector with two components.

Returns
Euclidean norm of the given vector.

Return type
float

foliationSimulation.**euclidean_normal_values**(*xValues*, *func*)

Create x- and y-values for the Euclidean normal vector of the given function.

Parameters

- **xValues** (*list*) – List of x-coordinates.
- **func** (*SymPy type*) – Function for which the values of the unit normal vectors are created.

Returns
The x- and y-values of the normal vector of the given function.

Return type
tuple

foliationSimulation.**euclidean_normal_vector**(*func*)

Calculates normal vector with Euclidean metric.

Parameters
func (*SymPy type*) – Function of which the normal vector is calculated.

Returns
Normal vector of given function.

Return type
tuple

foliationSimulation.**euclidean_unit_vector**(*vector*)

Normalize 2 dimensional vector from Euclidean space.

Parameters
vector (*tuple*) – Vector with two components.

Returns
Normalized components of the given vector.

Return type
tuple

foliationSimulation.**extrema**(*interval*, *func*)

Finds extrem points of function.

Parameters

- **interval** (*tuple*) – Interval limits in which extrema are searched for.
- **func** (*SymPy type*) – Function of which the extram are determined.

Returns
Roots of first derivative of the given function.

Return type
list

foliationSimulation.**find_duplicates**(*lst*)

Returns duplicates in a given list.

Parameters
lst (*list*) – List of values.

Returns
Values that appeared more than once.

Return type
list

foliationSimulation.**index_normal_sign_flip**(*surface*)

Returns index of x-coordinate where a sign flip of the normal vectors appears.

Parameters
surface (*tuple*) – The x- and y-coordinates of a 1-surface.

Returns
Index of x-coordinate where the normal vector of the surfaces changes sign.

Return type
integer

foliationSimulation.**intersection**(*x1*, *x2*, *x3*, *x4*, *y1*, *y2*, *y3*, *y4*)

Calculate the point where two lines constructed from 4 points intersect.

Parameters

- **xi** (*float*) – The x-coordinate of a point in 2 dimensional space.
- **yi** (*float*) – The y-coordinate of a point in 2 dimensional space.

Returns
The x- and y-coordinate of the point where the two lines intersect.

Return type
tuple

foliationSimulation.**isfloat**(*input*)

Check if input is float or not.

Parameters
input (*any*) – Input that is analyzed.

Returns
Tells whether the input is float or not.

Return type
boolean

foliationSimulation.**main**()

foliationSimulation.**minkowski_inner_product**(*v*, *w*)

Calculates the Minkowski inner product of two 2 dimensional vectors.

Parameters

- **v** (*tuple*) – Vector with two components.
- **w** (*tuple*) – Vector with two components.

Returns
Inner product of v and w.

Return type
float

foliationSimulation.**minkowski_norm**(*vector*)

Calculates Minkowski norm of 2 dimensional vector with (+-) signature.

Parameters
vector (*tuple*) – Vector with two components.

Returns
Minkowski norm of the given vector.

Return type
float

foliationSimulation.**minkowski_normal_values**(*xValues*, *func*)

Create x- and y-values for the Minkowski normal vector of the given function.

Parameters

- **xValues** (*list*) – List of x-coordinates.
- **func** (*SymPy type*) – Function for which the values of the unit normal vectors are created.

Returns
The x- and y-values of the normal vector of the given function.

Return type
tuple

foliationSimulation.**minkowski_normal_vector**(*func*)

Calculates normal vector with Minkowski metric.

Parameters
func (*SymPy type*) – Function of which the normal vector is calculated.

Returns
Normal vector of given function.

Return type
tuple

foliationSimulation.**minkowski_unit_vector**(*vector*)

Normalize 2 dimensional vector from Minkowski space.

Parameters
vector (*tuple*) – Vector with two components.

Returns
Normalized components of the given vector.

Return type
tuple

foliationSimulation.**osculating_circle**(*point*, *func*)

Calculates the radius and center of the osculating circles of a function at a given point.

Parameters

- **point** (*tuple*) – The x- and y-coordinate of the point of a function at which the osculating circle is calculated.
- **func** (*SymPy type*) – Function for which the osuclating circle is calculated at the given point.

Returns
Radius and x- and y-coordinate of center of the osculating circle.

Return type
list

foliationSimulation.**parameter_self_intersection**(*xValues*, *func*, *numberOfSurfaces*, *spacing*, *initialSurface*, *unitNormalVector*, *direction*)

Returns list of parameter intervals of the swallow tails of the surfaces obtained by pushing the initial surface along the normals.

Notes

Currently this does not generally work, e.g., for the exponential function, because there the swallow tails are tilted. This does also not work when the swallow tails get so large that they intersect another swallow tail. Also, if some swallow tail is only partly drawn, in the sense that the point of self-intersection is not included, then the swallow tail will still be visible. In that case it advised to reduce the number of surfaces drawn by one.

Parameters

- **xValues** (*list*) – List of x-coordinates.
- **func** (*SymPy type*) – Function that characterizes the initial surface.
- **numberOfSurfaces** (*integer*) – Number of surfaces that are drawn.
- **spacing** (*integer*) – The distance between two surfaces is by standard 1. Using this parameter the distance can be scaled as 1 / spacing.
- **initialSurface** (*tuple*) – The x- and y-coordinates of initial surface.
- **unitNormalVector** (*tuple*) – The x- and y-coordinates of normal vectors of initial surface.
- **direction** (*integer*) – Is either +1 or -1 depending on direction in which the initial surfaces is pushed along the normals.

Returns
Parameter intervals of swallow tails.

Return type
list

foliationSimulation.**push_surface**(*initialSurface*, *unitNormalVector*, *s*)

Push initial surface along normals.

Parameters

- **initialSurface** (*tuple*) – The x- and y-coordinates of the initial 1-surface.
- **unitNormalVector** (*tuple*) – The x- and y-coordinates of the unit normal vectors of the initial surface.

Returns
The x- and y-coordinates of the surface pushed along the normals.

Return type
tuple

foliationSimulation.**save_plot**(*filename*)

Saves the current full figure and the first axis as .pdf and .eps in the figures folder which should be located in the same folder as this notebook.

Parameters
filename (*str*) – Name under which the images are saved.

foliationSimulation.**smallest_distance**(*lst*)

Find smallest distance between neighboring elements of a list.

Parameters
lst (*list*) – List of numbers.

Returns
Smallest distance between neighboring elements.

Return type
float

foliationSimulation.**transform_function**(*xValues*, *func*)

Transforms expression of a function into a numpy expression and returns two lists of values.

Parameters

- **xValues** (*list*) – List of x-coordinates.
- **func** (*SymPy type*) – Function that is to be evaluated at the given x-coordinates.

Returns
The x-coordinates and the function evaluated at the coordinates.

Return type
tuple

foliationSimulation.**transform_vector**(*xValues*, *vector*)

Transforms tuple of sympy expressions into a tuple of numpy expression and evaluates at the given x-coordinates.

xValues
[list] List of x-coordinates.

vector
[tuple] Vector with two components that is to be evaluated at the given x-coordinates.

tuple
Components of the vector evaluated at the given x-coordinates.

foliationSimulation.**unit_circle**(*t*, *center*)

Calculate values of a unit circle around a given point.

Parameters

- **t** (*list*) – Values that parametrize the circle.
- **center** (*tuple*) – Point around which the unit circle is centered.

Returns
The x- and y-values of the points of the unit circle.

Return type
tuple

1.2 tangentSimulation module

tangentSimulation.**critical_coeff**(*xValues*, *func*, *s*)

Calculates the values of the critical coefficient as well as the the signed curvature and chi which is the critical coefficient divided by its absolute value.

Parameters

- **xValues** (*list*) – List of x-values.
- **func** (*SymPy type*) – Function that characterizes the initial surface.
- **s** (*float*) – Value of how far the initial surface is pushed along the normals.

tangentSimulation.**main**()

tangentSimulation.**plotting**(*input*, *inputSaveFileName*, *s=0*, *valueLimit=5*, *saveFile=False*)

Is the method regularly called by the interact method. Modifies the initial figure by giving it a title etc., handles the user inputs, controls the different choices, displays the plot.

Parameters

- **input** (*str*) – User input that is used as the function that characterizes the initial surface.
- **inputSaveFileName** (*str*) – Name that is used when saving a plot.
- **s** (*float*) – Parameter of how far the initial surface should be pushed along the normals.
- **valueLimit** (*integer*) – Interval limit for the x-values.
- **saveFile** (*boolean*) – If True the plot that it shown after pressing the 'Run interact' button is saved under the name written in the 'Name of Image' textbox.

tangentSimulation.**save_plot**(*fig*, *axis*, *filename*)

Save the current figure as .pdf and .eps.

Parameters

- **fig** (*matplotlib.figure.Figure*) – Holds all plot elements.
- **axis** (*matplotlib.axes._subplots.AxesSubplot*) – The subplot which will be saved.
- **filename** (*str*) – Name under which the images are saved.

tangentSimulation.**unit_tangent_vector**(*xValues*, *func*, *s*)

Calculates the components of the normalized tangential vector of an initial surfaces pushed along its normals.

Parameters

- **xValues** (*list*) – List of x-values.
- **func** (*SymPy type*) – Function that characterizes the initial surface.
- **s** (*float*) – Value of how far the initial surface is pushed along the normals.

Returns

Normalized components of the tangent vector.

Return type

tuple

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

foliationSimulation, 1

t

tangentSimulation, 11

INDEX

A

angle_at_sign_flip() (*in module foliationSimulation*), 1

B

bisect() (*foliationSimulation.ClassRoots method*), 1

C

calculate_self_intersections() (*in module foliationSimulation*), 1
circle() (*in module foliationSimulation*), 1
ClassRoots (*class in foliationSimulation*), 1
closest_value() (*in module foliationSimulation*), 2
critical_coeff() (*in module tangentSimulation*), 11
curvature() (*in module foliationSimulation*), 2

D

draw_centers_osc_circ() (*in module foliationSimulation*), 2
draw_curvature() (*in module foliationSimulation*), 2
draw_extrema() (*in module foliationSimulation*), 2
draw_modified_surfaces() (*in module foliationSimulation*), 3
draw_normics() (*in module foliationSimulation*), 3
draw_self_intersections() (*in module foliationSimulation*), 3
draw_surfaces() (*in module foliationSimulation*), 4
draw_tangents() (*in module foliationSimulation*), 4
drawing() (*in module foliationSimulation*), 4

E

euclidean_angle_between_vectors() (*in module foliationSimulation*), 5
euclidean_inner_product() (*in module foliationSimulation*), 6
euclidean_norm() (*in module foliationSimulation*), 6
euclidean_normal_values() (*in module foliationSimulation*), 6
euclidean_normal_vector() (*in module foliationSimulation*), 6
euclidean_unit_vector() (*in module foliationSimulation*), 6
extrema() (*in module foliationSimulation*), 7

F

find_duplicates() (*in module foliationSimulation*), 7
foliationSimulation
    module, 1

I

index_normal_sign_flip() (*in module foliationSimulation*), 7
intersection() (*in module foliationSimulation*), 7
isfloat() (*in module foliationSimulation*), 7

M

main() (*in module foliationSimulation*), 8
main() (*in module tangentSimulation*), 11
minkowski_inner_product() (*in module foliationSimulation*), 8
minkowski_norm() (*in module foliationSimulation*), 8
minkowski_normal_values() (*in module foliationSimulation*), 8
minkowski_normal_vector() (*in module foliationSimulation*), 8
minkowski_unit_vector() (*in module foliationSimulation*), 8
module
    foliationSimulation, 1
    tangentSimulation, 11

O

osculating_circle() (*in module foliationSimulation*), 9

P

parameter_self_intersection() (*in module foliationSimulation*), 9
plotting() (*in module tangentSimulation*), 11
push_surface() (*in module foliationSimulation*), 10

R

roots() (*foliationSimulation.ClassRoots method*), 1

rootsearch() (*foliationSimulation.ClassRoots method*), 1

S

save_plot() (*in module foliationSimulation*), 10
save_plot() (*in module tangentSimulation*), 11
smallest_distance() (*in module foliationSimulation*), 10

T

tangentSimulation
    module, 11
transform_function() (*in module foliationSimulation*), 10
transform_vector() (*in module foliationSimulation*), 10

U

unit_circle() (*in module foliationSimulation*), 11
unit_tangent_vector() (*in module tangentSimulation*), 11