DODO

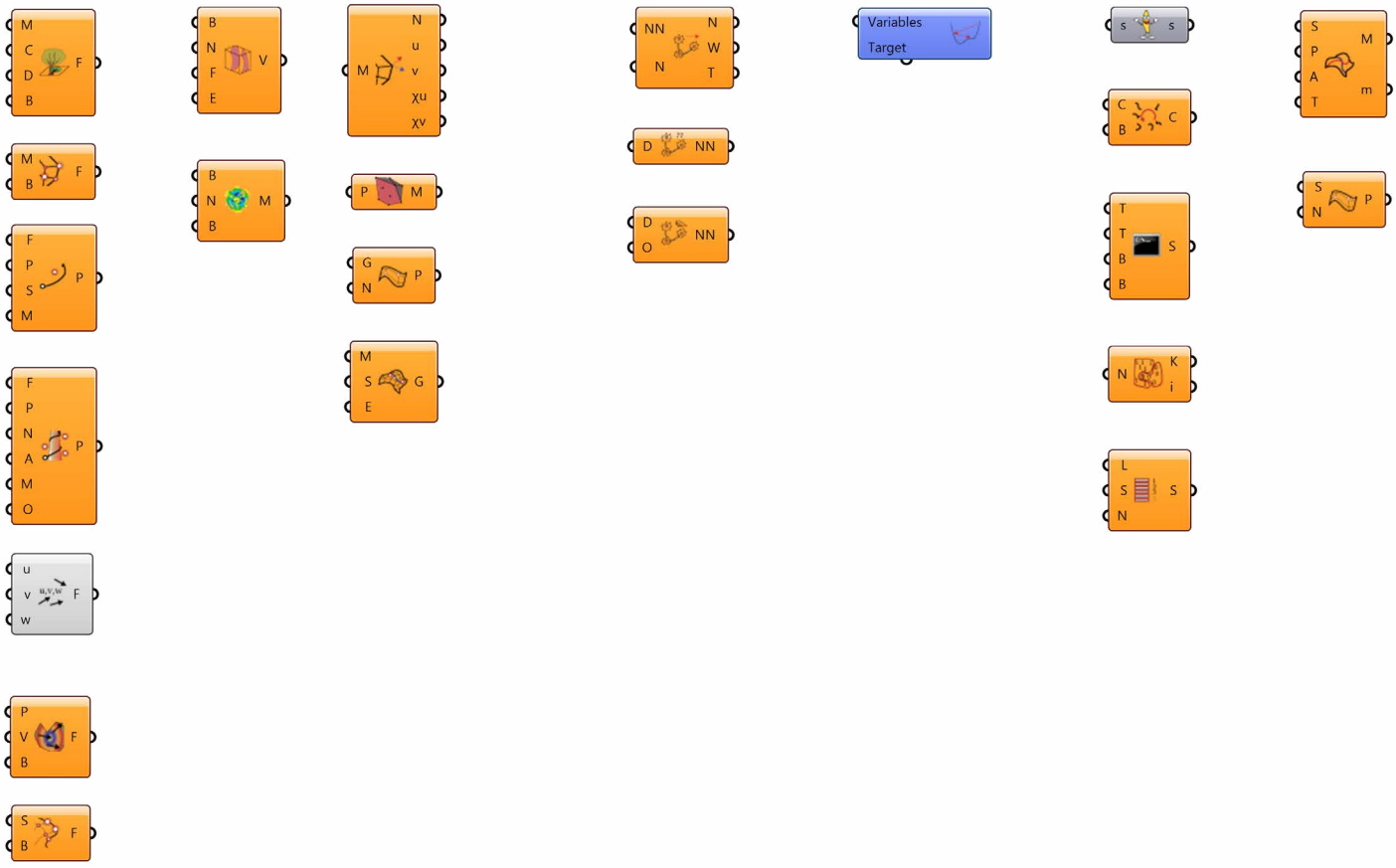
1. Introduction

Dodo is a Grasshopper plugin in which I'd like to collect tools matching my interests. They are mostly research oriented and their application in the design, AEC, automotive industries might not be straightforward but I believe that by giving everyone the possibility to play with them good ideas and implementations will surely arise :).

My main fields of interests are optimization, data analysis and visualization, mesh manipulation, structural analysis and many others that eventually you will discover by seeing the implementations I will hopefully be able to add to this plugin if I manage to have enough time.

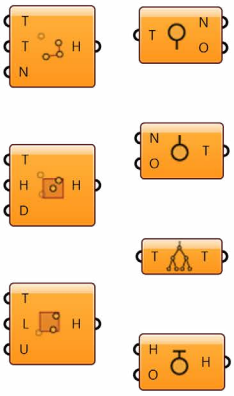
Also, if you have any idea to implement, suggestion or you want to report a bug, please contact me at:
lorenzogreco@parametricism.co.uk

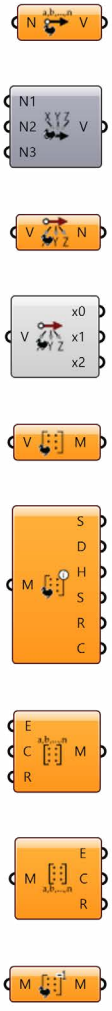
Figure 1
Dodo's components



Dodo
www.parametricism.co.uk

Index





2. NL Opt

Figure 2
Non-Linear Optimization component

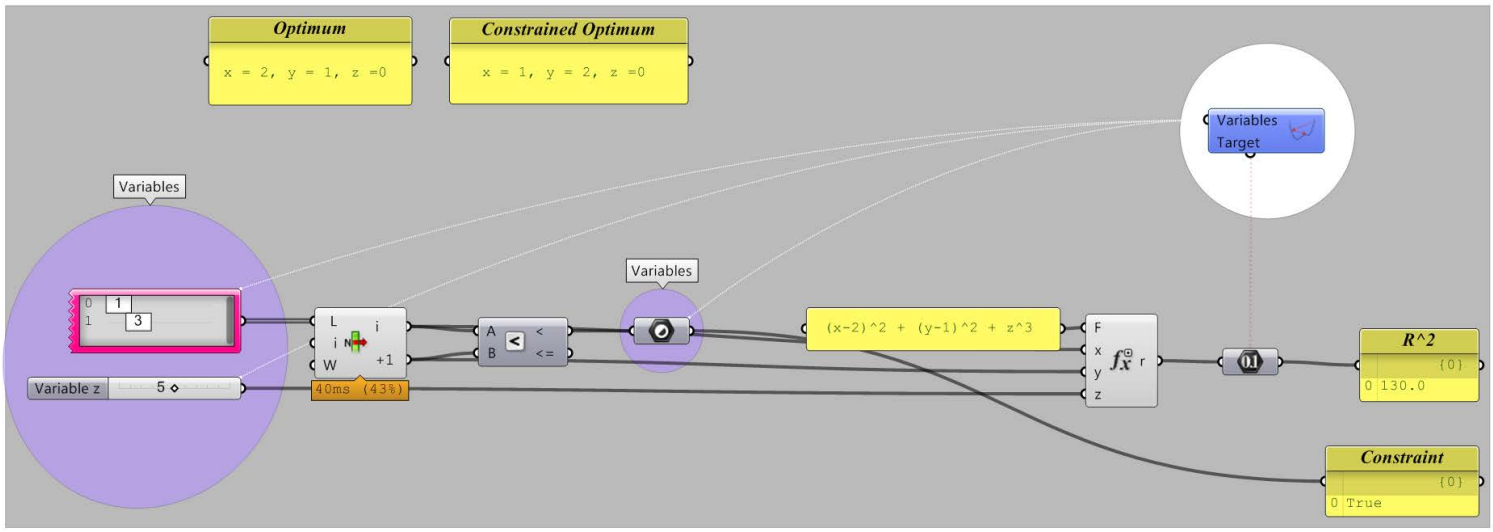
Non-linear optimization differs from linear optimization for the function it tries to minimize is non linear and/or the constraints it is subject to, which is the case for almost several engineering application. NL-opt makes use of gradient free algorithms to try to minimize the target function and in order to do that it needs to calculate an approximate value of the gradient for a given point. Once this is done, it tries to move to a neighbour point according to the interpretation of the gradient given by the specific engine criteria.

This plugin uses the .NET implementations of the famous NLOpt library that you can find here: *http://ab-initio.mit.edu/wiki/index.php/NLopt*, whilst the .NET implementation is here: *https://github.com/roryclune/NLOptDotNet*.
Due to the difficult interoperability of the dll, some engines are missings and non linear inequality constraints are harder to fulfill.

The GH component you need is the one in the picture and works similarly to Galapagos, Goat and Octopus, which I find very convenient.
Here's how to use it:

- Connect the input to any number of sliders and/or gene pools you want and also to a boolean - if the algorithm allows it - and the objective on the number you want to minimize/massimize.
- Double click on the component.
- Set the parameters.
- Click on RUN.

Figure 3
Optimization of a function with 3 unknowns and a constraint



A chart showing the goal trend is shown. You will notice that although the global trend is downwards there are spikes upwards. This is normal and the reason is that at each step the engine needs to compute the neighbouring points in order to calculate the local gradient. Of course these points can have (and most probably will) a higher value, hence the spikes.

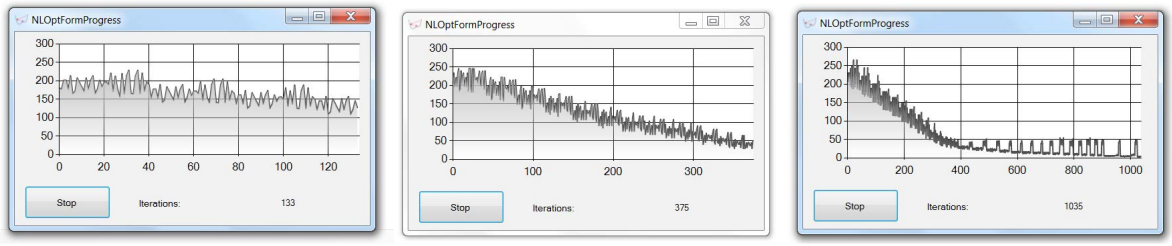


Figure 4
These three charts show the trend of an optimization process in time.

Due to the characteristics of the NLOpt library and its implementation for GH, some algorithms could not be implemented and not all of them support restrained solutions. Please check these algorithms on the table on the right.

In **bold** some of the most common.

Name	Can use bool
GN_DIRECT	
GN_DIRECT_L	
GN_DIRECT_L_RAND	
GN_DIRECT_NOSCAL	
GN_ORIG_DIRECT	✓
GN_ORIG_DIRECT_L	✓
LN_PRAXIS	
GN_CRS2_LM	
GN_MLSL	
GD_MLSL	
GN_MLSL_LDS	
GD_MLSL_LDS	
LD_MMA	✓
LN_COBYLA	✓
LN_NEWUOA	
LN_NEWUOA_BOUND	
LN_NELDERMEAD	
LN_SBPLX	
LN_AUGLAG	✓
LD_AUGLAG	✓
LN_AUGLAG_EQ	✓
LD_AUGLAG_EQ	✓
LN_BOBYQA	✓
AUGLAG	✓
AUGLAG_EQ	✓
G_MLSL	✓
G_MLSL_LDS	✓
LD_SLSQP	✓

Table 1
Non-Linear optimization engines implemented in Dodo. In **Bold** some of the most common

3. Marching Cubes/Tetrahedra

An isosurface is the collection of points in space mapping to the same value. In a 2D domain the isosurface is an isoline and an example of them can be found in topographical maps where mountains and depression are represented as closed curves each one indicating a different height.
Isosurfaces are the equivalent of isocurves but are the result of a 3D do-

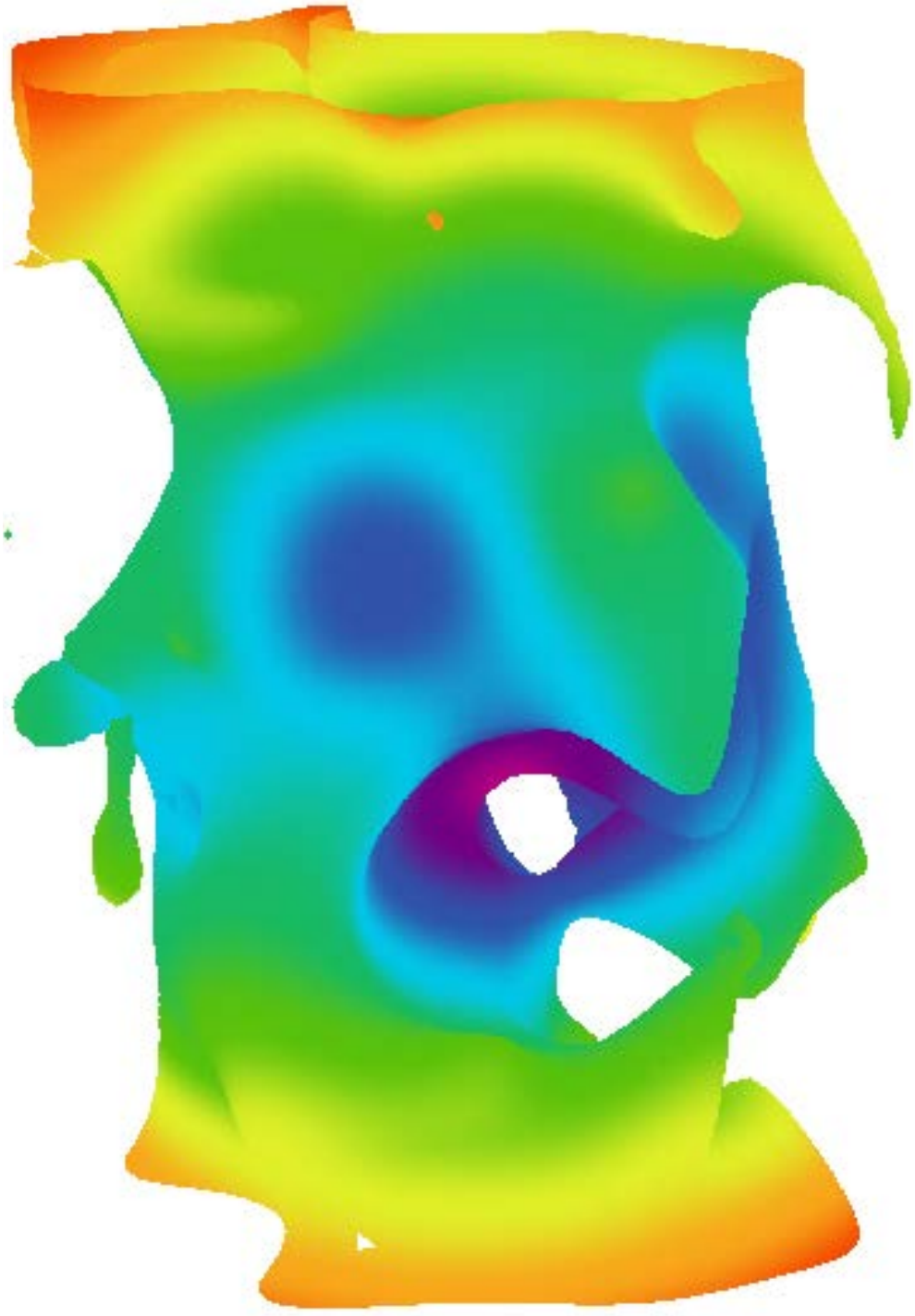


Figure 5
xample of GH definition for marching tetrahedra. A cylinder is disturbed by a point field

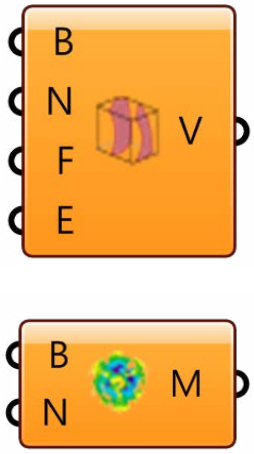
main where each point in space has a certain value. They connect all these points together thus forming one or multiple surfaces.

One of the main methods to produce these isosurfaces is using Marching Cubes or theory that generated from that one. In this plugin Marching Tetrahedra have been used since they can manage better singular points. Among the new implementations there are some making use of derivatives of the scalar field given, but unfortunately the grasshopper Field component does not calculate derivatives. Maybe in the future I will implement a new customized field component but for the time being I thought that using the already available tools is best.

Generating Isomesh with Dodo is pretty straightforward. First you need to generate the scalar field using the field components GH has, writing an analytical formula or both. In the example below a field is created using random points with random charges and is summed to the equation of the Barth Sextic. Just plug the field and the equation into the Voxel generator and set a value for the sampling of the scalar field. Then plug the voxel component into the Mesh generator and set the isovalue for the mesh. Enjoy!

In the Isomesh component you can switch from marching tetrahedra (true) marching cubes (false).

Figure 6
Voxel generator (top) and Isomesh generator (bottom)



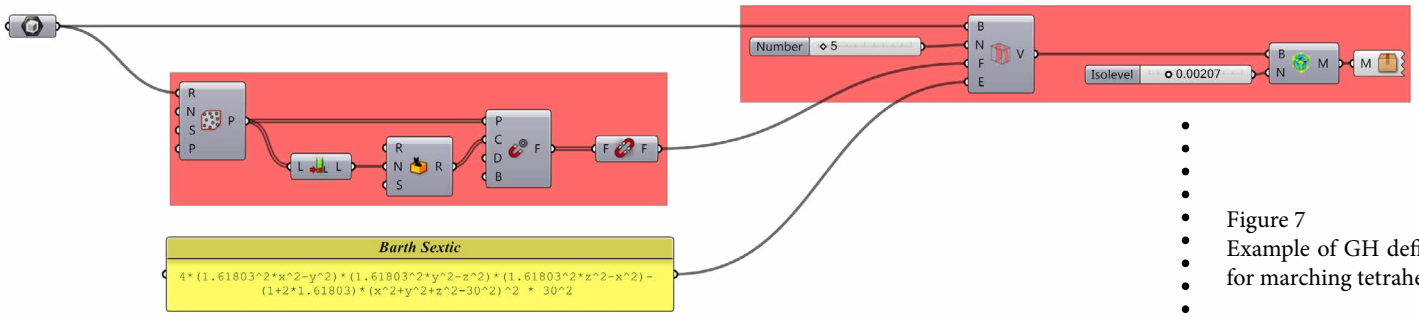


Figure 7
Example of GH definition for marching tetrahedra

4. Artificial Neural Network (ANN)

More info: https://en.wikipedia.org/wiki/Artificial_neural_network

Link to the .dll: http://www.codeproject.com/Articles/16447/Neural-Networks-on-C.

Artificial Neural Networks can date back to the second half of 1900 and take their name by their similarity with neuron interconnections in brain. ANN are composed by a series of layer each containing a number of neurons, each of which connects to its peers in the layer before and after, as shown in the following picture. The example in the picture shows an ANN mapping data from a 3D domain to a 2D on by making use of a hidden layer. Hidden layers are those which are not directly connected with the input data nor with the output. An ANN can have any number of neurons and hidden layer which can bring to completely different results as well as increased computational time. Neurons' connections are drawn as arrow pointing in the direction in which the data flows. Usually neurons perform very basic operations and they are connected through value functions which are the targets ANNs optimize in order to fit the input data to the expected results.
The field of application of these ANN is data fitting, prediction and classification and their implementation is treated below.

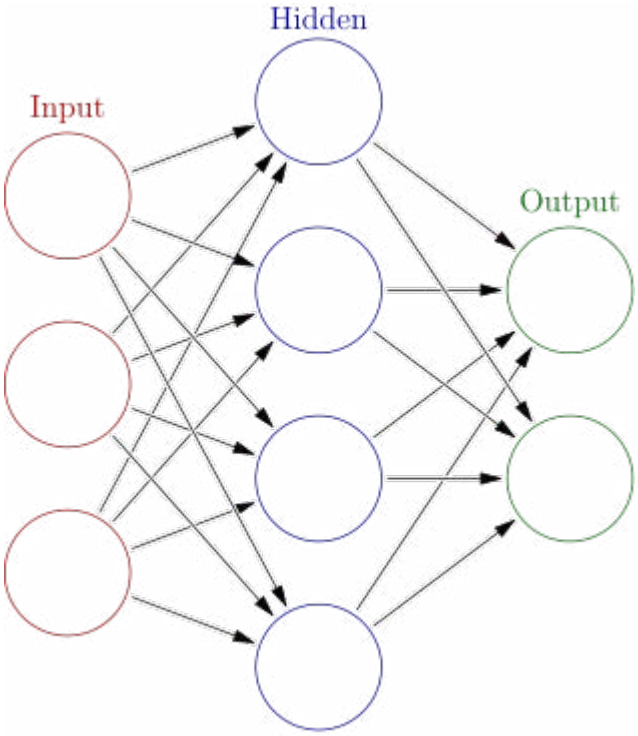


Figure 8
Scheme of an Artificial Neural Network with 1 input layer, 1 hidden layer with 4 neurons and 1 output layer. (*wikipedia*)

These mentioned are not the only applications though as ANN can be used with *unsupervised learning* as Self-Organizing Map (SOM)

ANN - Supervised Learning

This type of learning algorithms use sample inputs matched desired output values during the learning phase. The goal of this method is to shape to ANN so to provide a close fit output when given an input.

By double-clicking on the component on the right the form below appears where you can choose between three different learning algorithms:

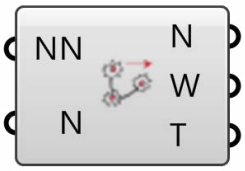


Figure 9
ANN Supervised Training

Learning Algorithm	Activation Function
Back-propagation	Sigmoid Bipolar Sigmoid
Delta-Rule	Threshold
Perceptron	Threshold

Table 2
Activation functions for supervised training

Supervised training form and paramters you can set are shown here:

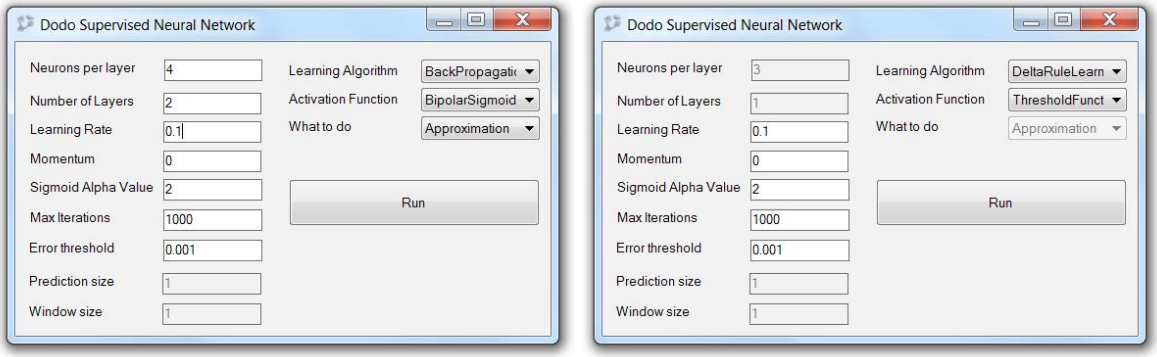


Figure 10
Supervised training Forms for (a) Back-Propagation and (b) Delta Rule and Perceptron

Paramter	Explanation
Neurons	Number of neurons per layer
Layers	Number of layers
Learning rate	Coefficient for updating the old weights according to the error.
Momentum	Adds inertia to the change in weights smoothing it down.
Sigmoid alpha value	Coefficient for the exponential in the sigmoid.
Max iterations	Maximum number of iterations before automatic stop.
Error threshold	Error value under which the learning ends.
Prediction size	Only for back-propagation for prediction. It defines how many vector inputs to use for learning.
Window size	Only for back-propagation for prediction. It defines how many output vectors to predict.

Table 3
Supervised training algorithms and their activation functions

Supervised Training - Delta Rule

Delta Rule learning is one of the two threshold finders in Dodo's ANN along with Perceptrons Rule.
In the example below the 8 verteces of a cube are assigned to two groups: group 0 above and group 1 below. The trained ANN finds the decision boundary depicted as a grey plane which splits the decision space in two. A sample point shown as a sphere changes colour according to the value the ANN gives to its position. Finally in picture (c) a grid of 11x11x11 have been sampled and only the points having calulated class of 1 have been shown demostrating that they are respctful of the boundary surface.

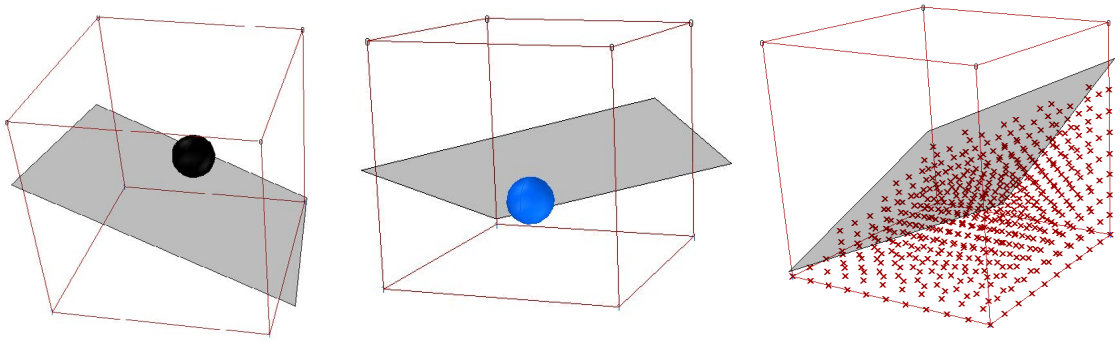


Figure 11
Decision boundary for a cube which upper verteces are in class 0 and the bottom are in class 1

Supervised Training - Perceptron

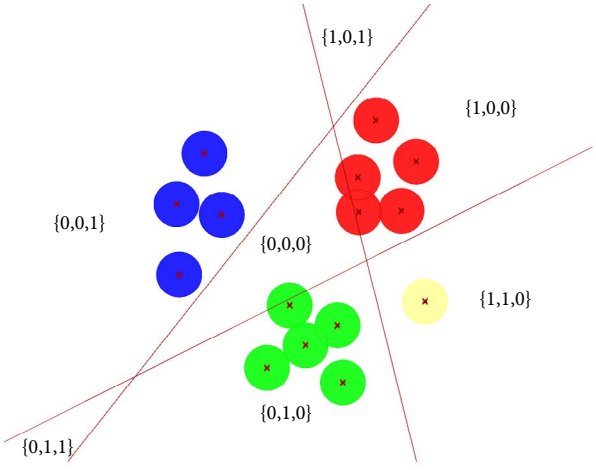


Figure 12
Decision boundaries are represented by red lines. They divide the specimen domain in 7 areas of which 3 of exclusive competence of each cluster, 3 shared and the central one unknown

In this example above the NN is fed with a number of points in the 2D space, being part of three distinct groups which leads to a 3D solution space identified. To represent the group to which they are part of, a 3D vector can be used having for each dimension either 0 or 1. Moreover, in order to have a better graphical understanding, the vectors are multiplied by 255 so to transform them into colors: {R,G,B}

Supervised Training - Aproximation

Supervised training can be used to make an ANN be able to predict values of a series. In the example shown below 8 couples of number were used and plotted in teal. In red it is shown the aproximating curve generated by the ANN trained using back-propagation learning. The two curves neatly superpose on the first part and diverge a bit on ending, but this behaviour can change widely playing with the learning coefficients.

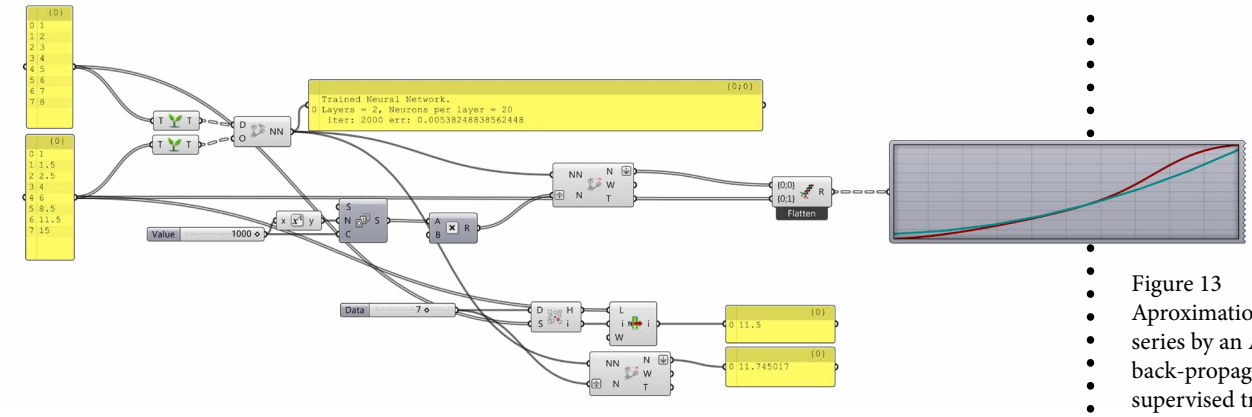


Figure 13
Aproximation of a series by an ANN using back-propagation as supervised training

Supervised Training - Prediction

For prediction the procedure is the same as before but we feed the ANN with one single series of number but making it use sub-series of 5 numbers to predict the 6th and use the difference between the latter and the expected value to rate the learning. In the example below 16 values from a cosine have been used and in the picture underneath one can see how well the ANN predicts the following 34 values without progressive increase of the error.

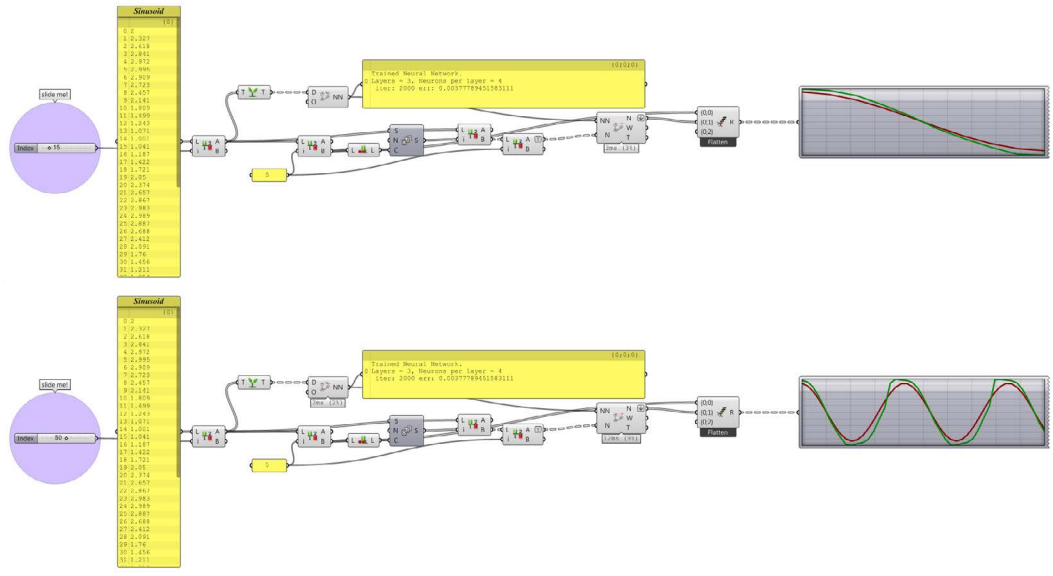


Figure 14
Prediction of a sine in the range [0; 5Pi] from a series [0;Pi/2]

ANN - Unsupervised Learning

Figure 15
ANN Supervised Training

The ANN is given sample inputs without expected results and it will organize itself in order to find patterns in this series.
Two learning algorithms are implemented in Dodo:

Learning Algorithm	Notes
Som	
Elastic Network	The neuron count needs to be a perfect square since it is used to produce a square network

Table 4
Unsupervised training algorithms

Unsupervised learning form and parameter explanations are explained in the following.

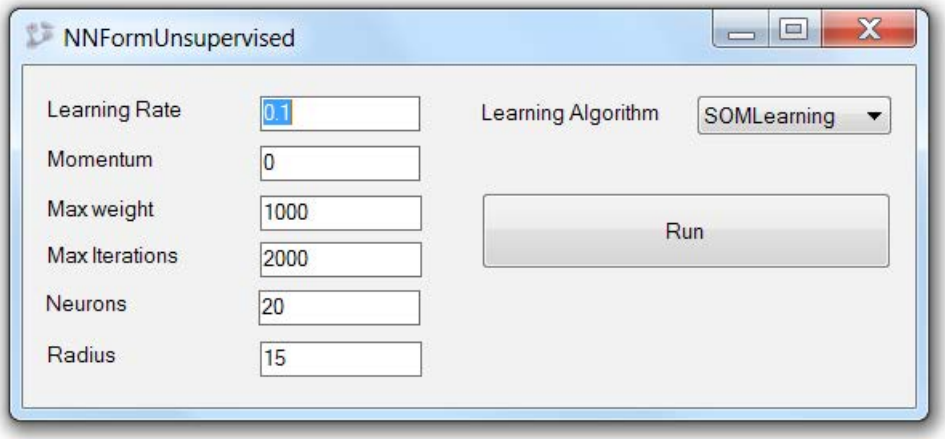


Figure 16
Unsupervised training Forms

Parameter	Explanation
Learning rate	Coefficient for updating the old weights according to the error.
Momentum	Adds inertia to the change in weights smoothing it down.
Max weight	Upper bound for neurons' weights. The lower bound is always 0 - rescale your inputs if this does not suit your data.
Max iterations	Maximum number of iterations before automatic stop.
Neurons	Number of neurons per layer.
Radius	Radius within which each neuron compares against the others.

Table 5
Paramters in Unsupervised training with their explanation

Unsupervised Training - Elastic Network

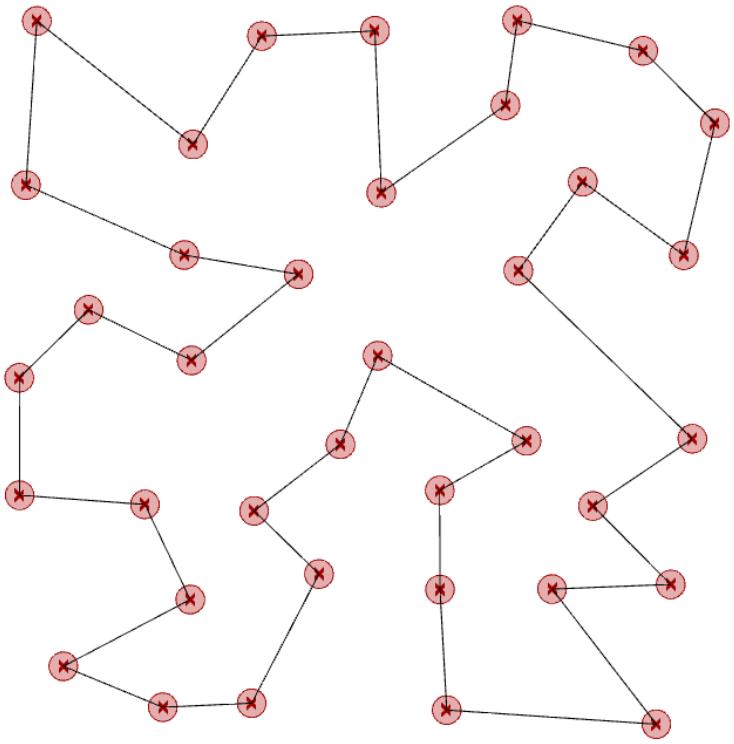


Figure 17
Minimum path between specimen

Unsupervised training and elastic network can be used to find a solution to the travelling salesman problem. The elastic network is initially brought to the center of the data set, then slowly tries to replicate the values (position) basically working as a dumped spring system. The resulting weights are the euclidean distances between the data points and the neurons' output can be used how well the NN fit to the data samples. The neurons' output can then be used to see how similar another dataset is to the first one and it is my understanding that this patter recognition strategies have been successfully used to recognize cancer cells from pictures.

Unsupervised Training - Self Organizing Map

[From wikipedia: https://en.wikipedia.org/wiki/Elastic_map]
The method of elastic maps has been systematically tested and compared with several machine learning methods on the applied problem of identification of the flow regime of a gas-liquid flow in a pipe. There are various regimes: Single phase water or air flow, Bubbly flow, Bubbly-slug flow, Slug flow, Slug-churn flow, Churn flow, Churn-annular flow, and Annular flow. The simplest and most common method used to identify the flow regime is visual observation. This approach is, however, subjective and unsuitable for relatively high gas and liquid flow rates. Therefore, the machine learning methods are proposed by many authors. The methods are applied to differential pressure data collected during a calibration process. The method of elastic maps provided a 2D map, where the area of each regime is represented. The comparison with some other machine learning methods is presented in Table 1 for various pipe diameters and pressure.

Here, ANN stands for the backpropagation artificial neural networks, SVM stands for the support vector machine, SOM for the self-organizing maps. The hybrid technology was developed for engineering applications.[13] In this technology, elastic maps are used in combination with Principal Component Analysis (PCA), Independent Component Analysis (ICA) and backpropagation ANN.

Algorithm	Calibration	Testing	Larger diameter	Higher pressure
Elastic map	100	98.2	100	100
ANN	99.1	89.2	76.2	70.5
SVM	100	88.5	61.7	70.5
SOM (small)	94.9	94.2	83.6	88.6
SOM (large)	100	94.6	82.1	84.1
SOM (large)	100	94.6	82.1	84.1

Table 6
Flow regime identification accuracy (%) of different machine learning algorithms

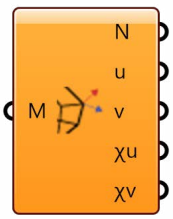
5. Mesh Tools

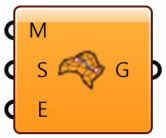
Finding the principal curvature of a continuous surface is relatively easy and is part of the domain of differential geometry. Meshes are discrete and therefore do not have continuous derivatives. Several papers have addressed this issue and the solution given by Szymon was adopted in Dodo. The paper also gives a very good look on the state of the art explaining pros and cons of other methods.

Paper from Szymon Rusinkiewicz: *http://gfx.cs.princeton.edu/pubs/_2004_ECA/index.php*

The grasshopper component only needs the mesh as input and returns the two principal directions - pointing toward increasing curvature - and principal curvature values at the mesh's vertices, whilst intermediate points can be derived by weighted average.

The images show an isomesh for which curvature has been calculated and the it has been integrated starting from points.





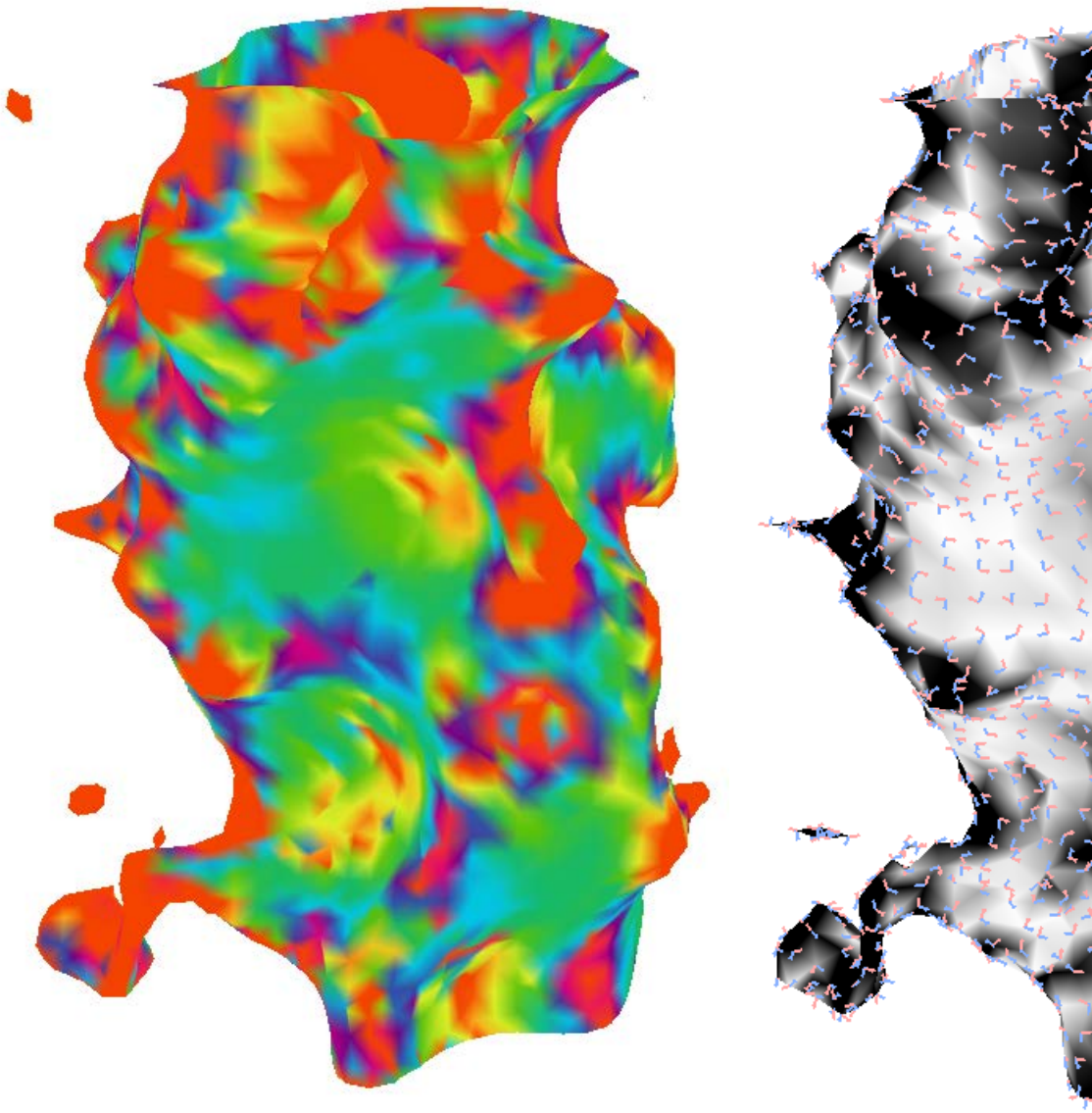
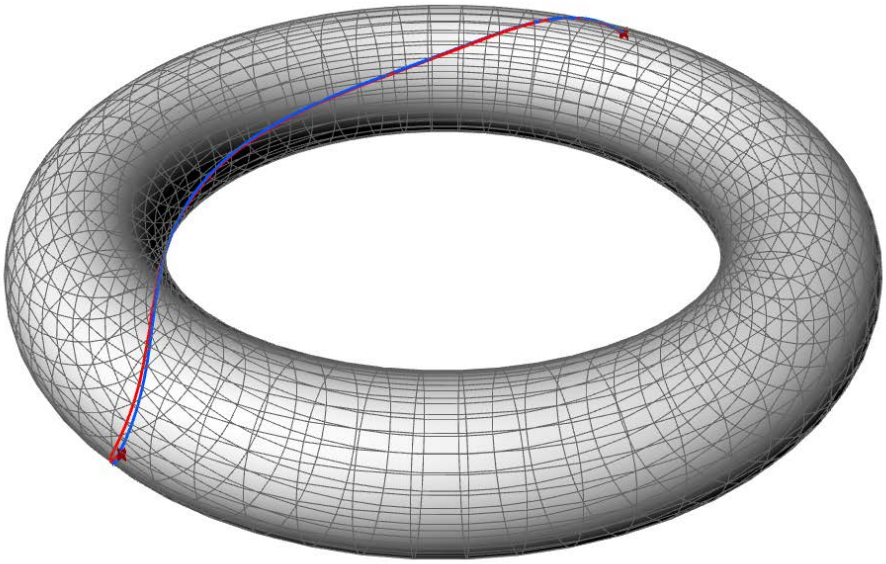


Figure 18
From top to bottom:
(a) Mesh Curvature,
(b) Convex Hull,
(c) Singular points



There are more refined strategies algorithms to find geodesics on meshes, but they applies to triangular meshes only. References can be found in the end.

Another useful tool consists in Geodesic on Mesh which exploits breadth-first and the midpoint subdivision to find the shortest path between two points.

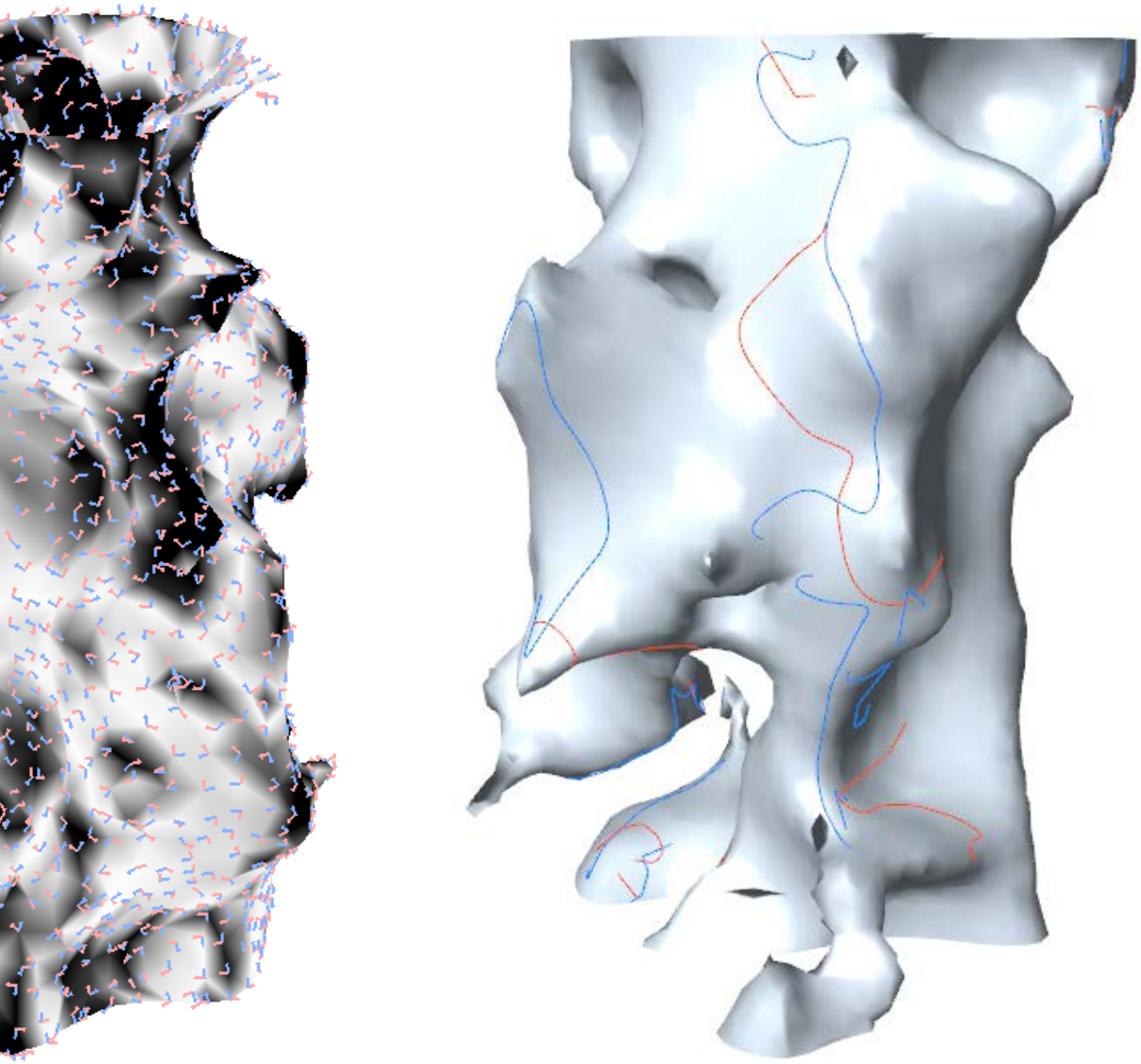
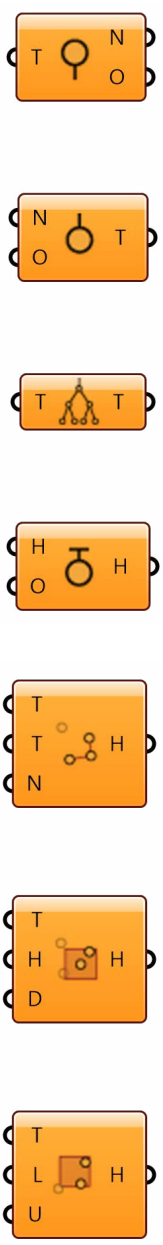


Figure 19
Color gradient showing curvature (a), false colors with principal directions (b) and curvature integration from given points (c)

6. kd-Tree

KD-Tree is a method to store and search multidimensional data very fast (although the speed decreases with the increase of dimensions), *here's the wikipedia article*.

Use component (a) to create an item of the dataset or component (b) to deconstruct it. Then you can build a datree by collecting all the items in component (c) and you are ready to search the dataset. Three type of search are implemented now: nearest neighbours (d), points in a domain (e) and points within a certain rectangular distance from a given source (f).



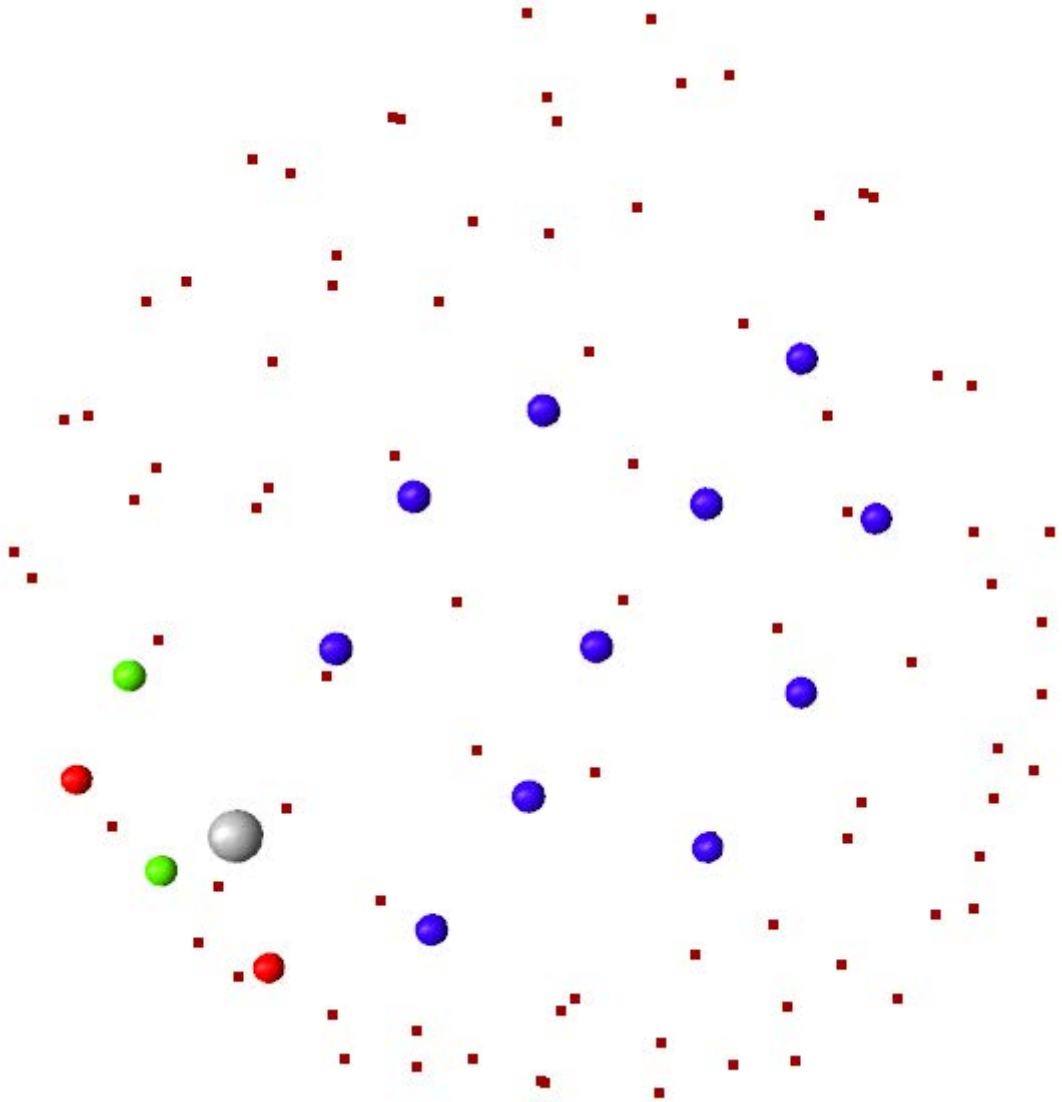


Figure 20
From top to bottom:
(a) Construct HPoint,
(b) Deconstruct HPoint,
(c) Construct tree, (d)
Deconstruct HPoint,
(e) Closest points, (f)
HPoints within distance,
(g) HPoints in region

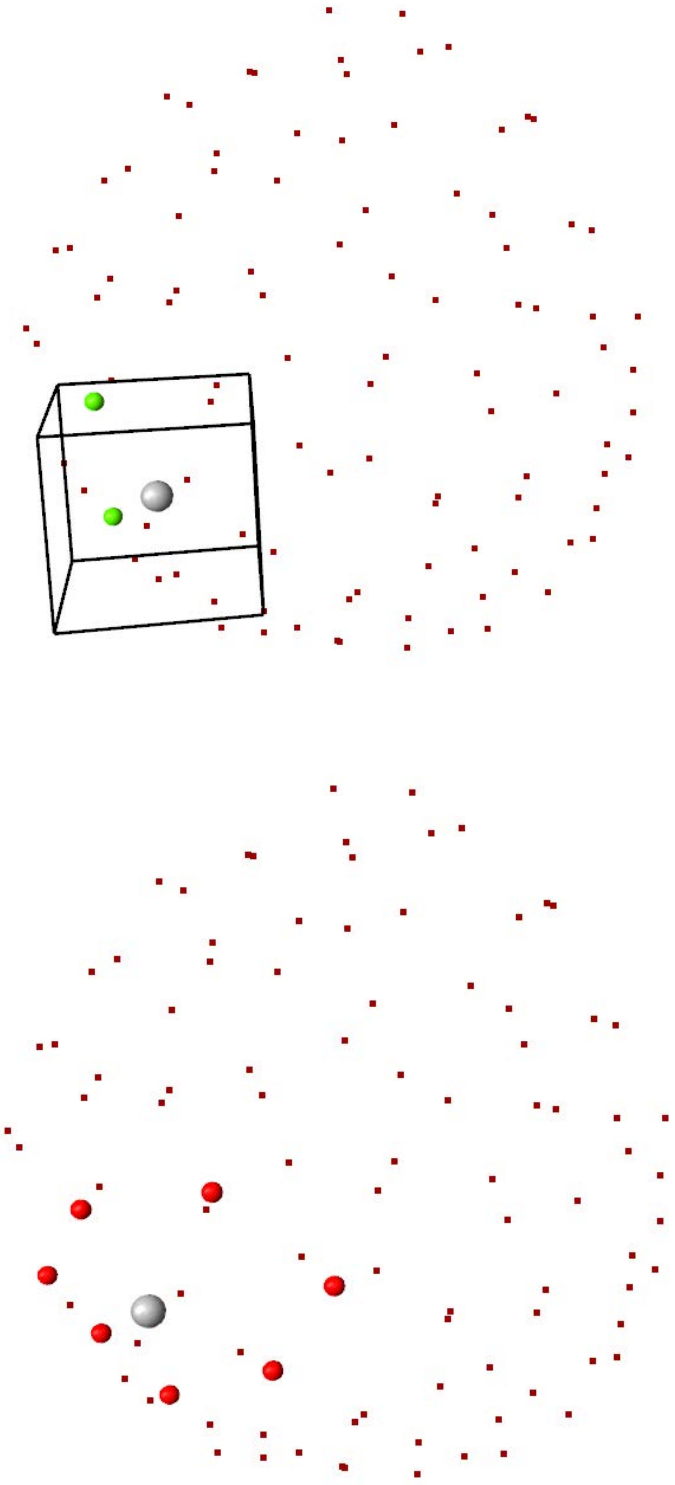


Figure 21
On the opposite page: (a) points within a given distance from seed.
On this page top to bottom: (b) 7 closest points, (c) points within a region of space.

7. MathNet

MathNet is the equivalent of numpy and scipy python libraries for .NET. It implements linear algebra, Fourier transform, matrix manipulation and decomposition, linear coefficients interpolation and solvers.

The utilization is pretty straightforward here the components only are shown.

External resources:

http://www.imagingshop.com/linear-and-nonlinear-least-squares-with-math-net/

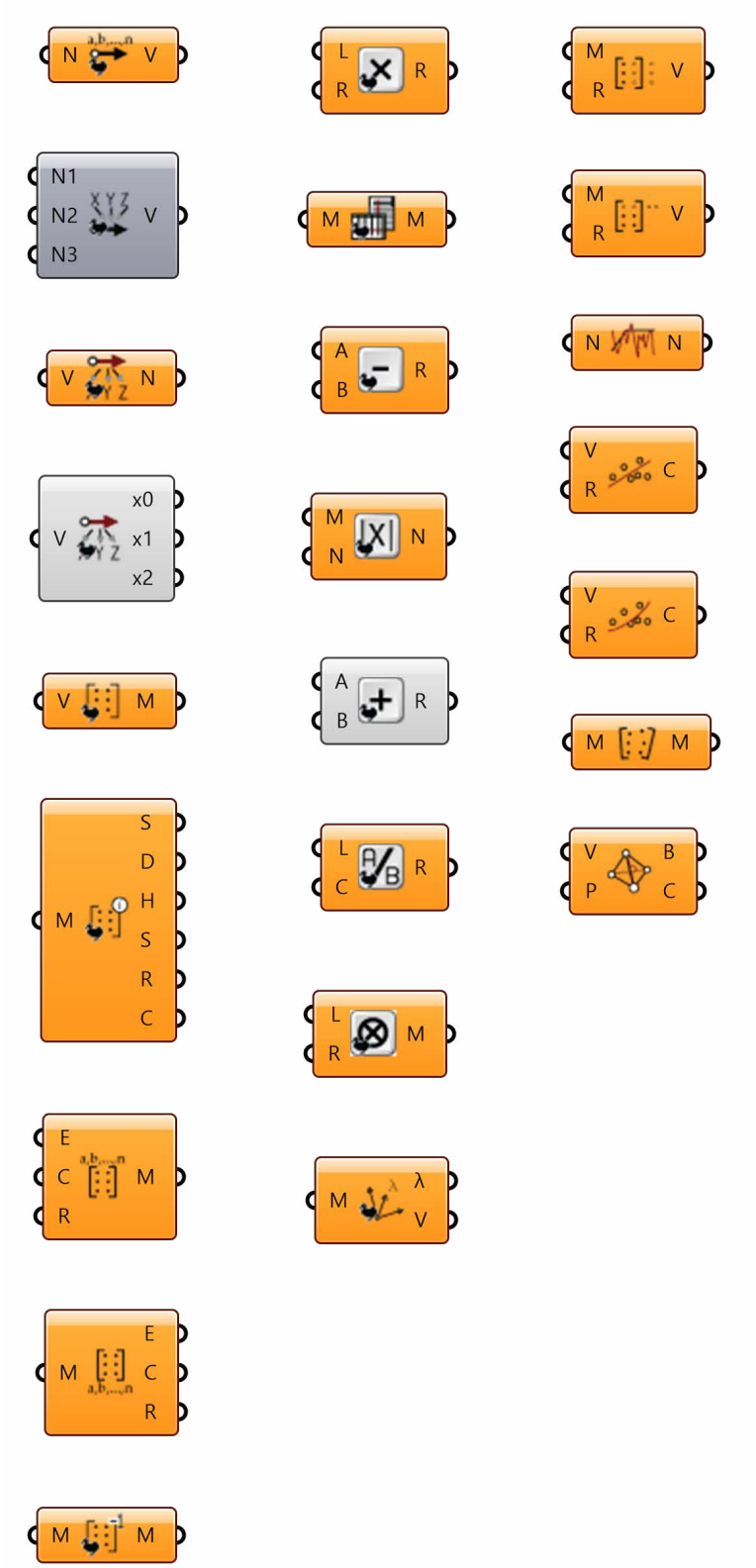
http://numerics.mathdot-net.com/Regression.html

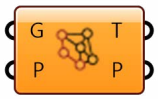
It's worth mentioning that:

- Everything is computed using complex numbers so to allow a broader range of operations.
- DVectors can be cast to/from HyperPoints for k-dtree search.
- DVectors can be cast to/from GH points, vectors, complex numbers and matrices.

Among the additional mathematical utilities Dodo features:

- Linear and NL-Least Squares;
- Fourier transform;
- Matrix decomposition;


N
V
N1
N2
N3
V
V
N
V
x0
x1
x2
V
M
M
S
D
H
S
R
C
E
C
R
M
M
E
C
R
M
M
L
R
R
M
M
M
A
B
R
M
N
N
A
B
R
L
C
B
R
L
R
M
M
λ
V
M
R
V
M
R
V
N
N
V
R
C
V
R
C
M
M
V
P
B
C



8. Graph

Dodo can generate graphs starting from lines, distances and weights, and then find minimum paths be there topological or shortest time/space -wise

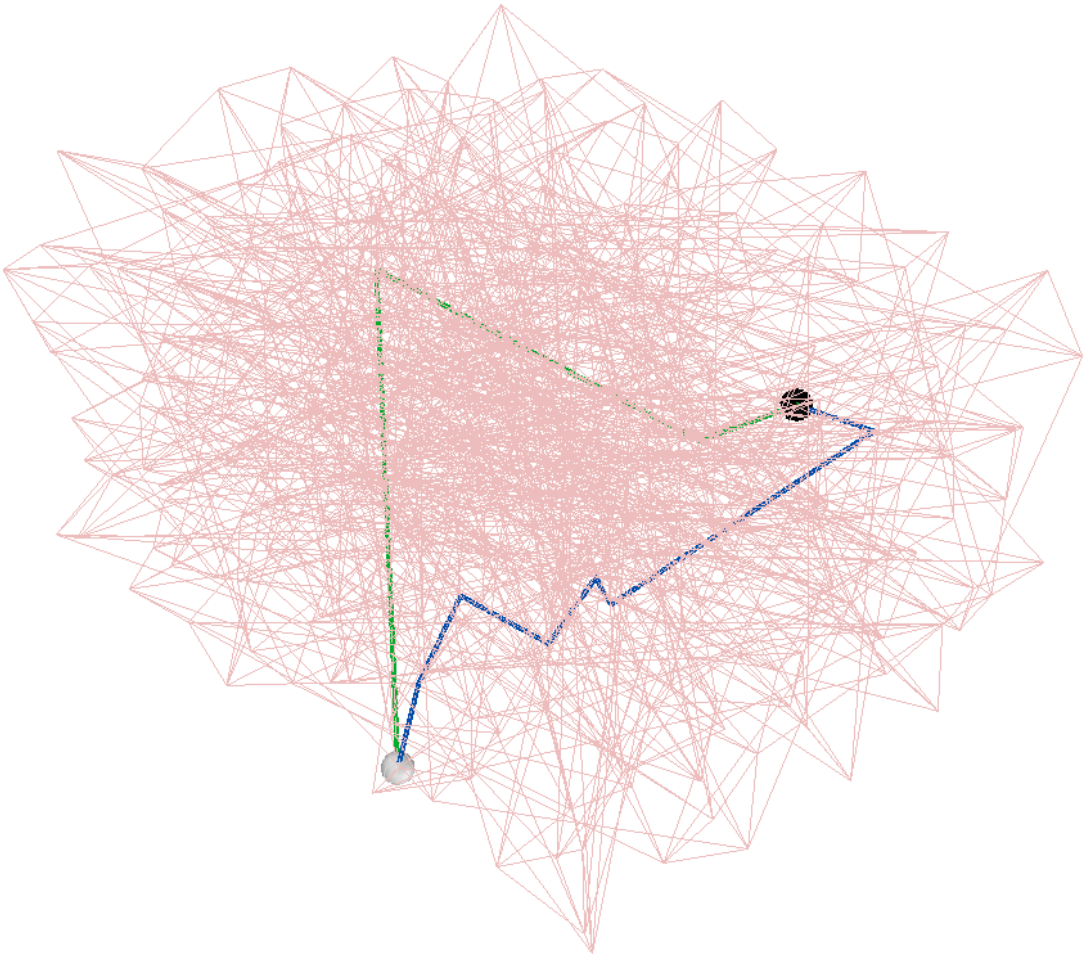
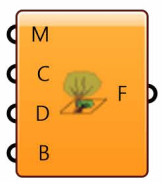
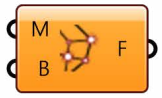
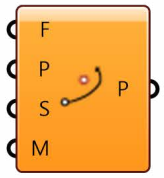


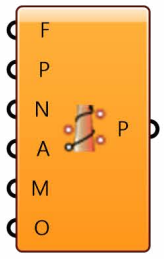
Figure 22
From top to bottom: (a) construct graph, (b) Calculate topological distances, (c) Show distance for point

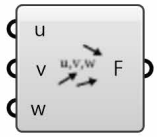
GREEN: Minimum topological path having distance = 3
BLUE: Shortest spatial path having compounded length = 57.3

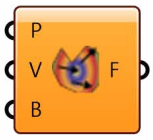












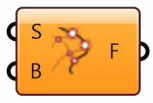
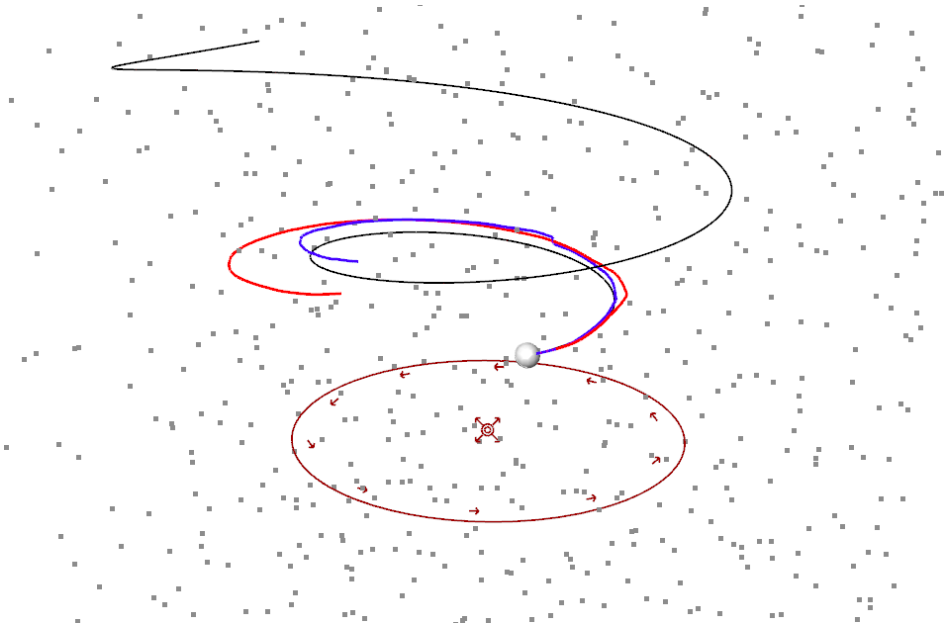


Figure 23
From top to bottom: (a) Field from image, (b) Field from mesh, (c) Point in vector field, (d) Path constrained on a surface, (e) Field from expression, (f) Field from vectors, (g) Field from surface

9. Fields

A few components to enrich the native Grasshopper ones.
1) generate a vector field from an image colors;
2) from a mesh curvature;
3) find the trajectory of an object in a field;
4) finds the field path using an object as constraint
5) create a vector field from a mathematical function
6) generate a continuous vector field from a discrete set of vectors in space



In the image above it is a comparison of a force path derived from a continuous field (black) and paths derived from densely and less densely sampled discrete fields (red and blue respectively).

In the image below the same vector field is integrated using a cone as constraint.

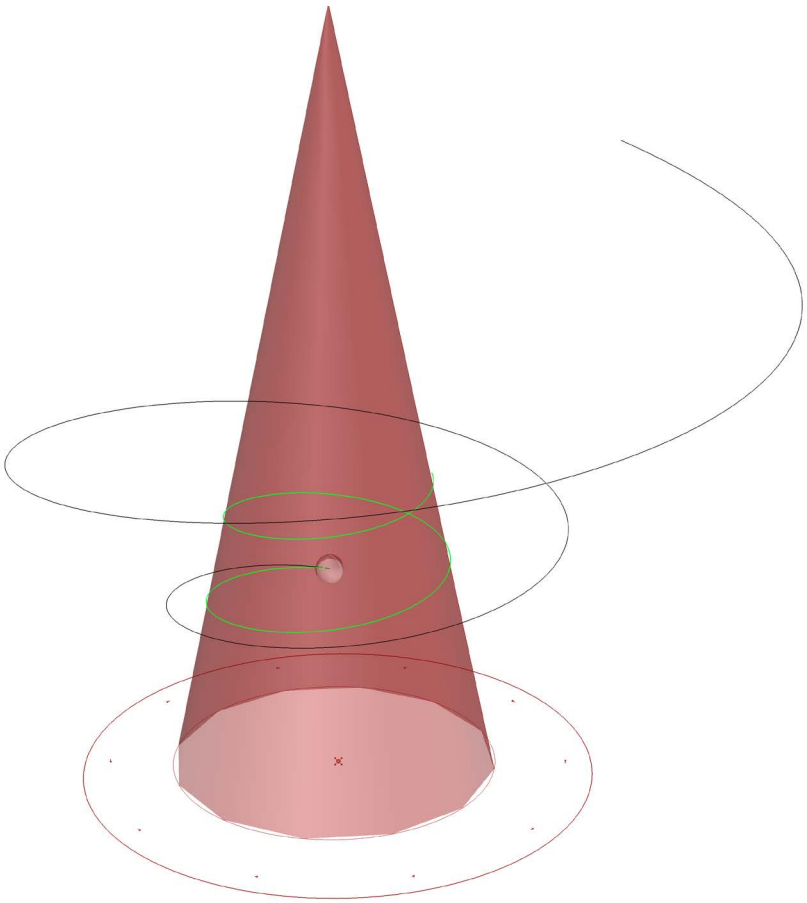


Figure 24
Force flow curves in a field. The green path has been constrained on a conical surface.

10. Other Tools

Convex 3D Hull
Creates a triangular mesh which is the smallest convex enclosure of a point cloud.

Surface Curvature
This tool generates paths along the maximum curvature direction.

Banana
A dancing banana.

Run Executable
Runs an external exacutable and returns the return value.

Series from List.
Generates a list of numbers using the length of the given list.

Group Numbers.
Groups a series of numbers.

11. References

Non-Linear Optimization

The NL-optimization component inherits from GalapagosComponent from David Rutten and is inspired by Goat from Simon Floery.

This plugin uses the .NET implementations of the famous NLOpt library that you can find here: *http://ab-initio.mit.edu/wiki/index.php/NLopt*, whilst the .NET implementation is here: *https://github.com/roryclune/NLOptDotNet*.

Numerical Optimization - J. Nocedal

_____

Artificial Neural Network

Artificial Intelligence MIT Open courseware
http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/

The Neural Network .dll is from Andrew Kirillov and can be found here: *http://www.codeproject.com/Articles/16447/Neural-Networks-on-C.*

_____

Mesh Curvature

The Mesh curvature component uses the algorithm described by Szymon Rusinkiewicz in the paper: *Estimating Curvatures and Their Derivatives on Triangle Meshes*

_____

Marching Cubes/Tetrahedra are based on the work of Paul Bourke: *http://paulbourke.net/geometry/polygonise/*

_____

Geodesics on Meshes

D. Martinez et al., Computing Geodesics on Triangular Meshes, 2005
V. Surazhsky et al, Fast Approximate Geodesics on Meshes

Computational Geometry

Mesh Parameterization: Theory and Practice, SIGGRAPH 2008.