3D printed part in porcelain. See example at 'Lofts' chapter

PART V Parametric design

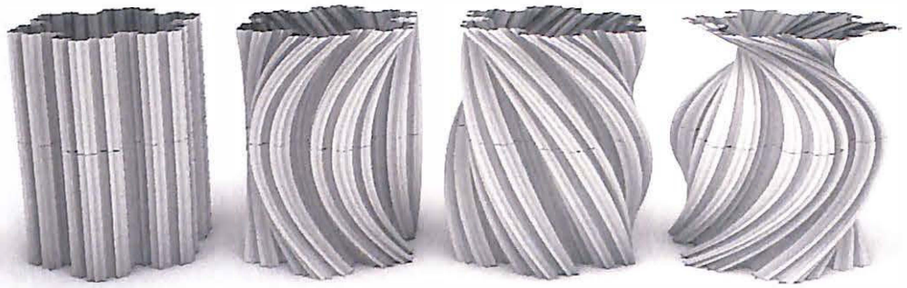
In this chapter you will find same examples and possibilities when controlling the path from Grasshopper®.

Lofts

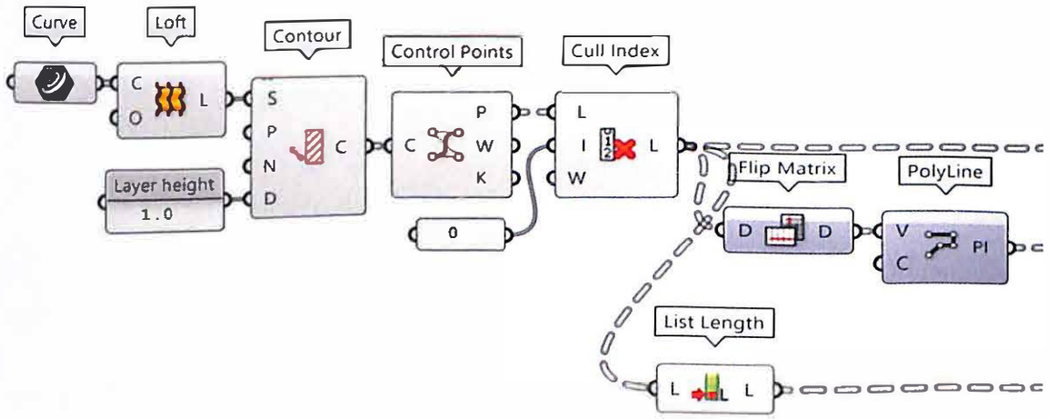
A very common way to start a project for clay 3D printing is using the tool 'Loft'. It is necessary to control four different aspects to make a proper loft: Order, Directions, Seam and Styles.

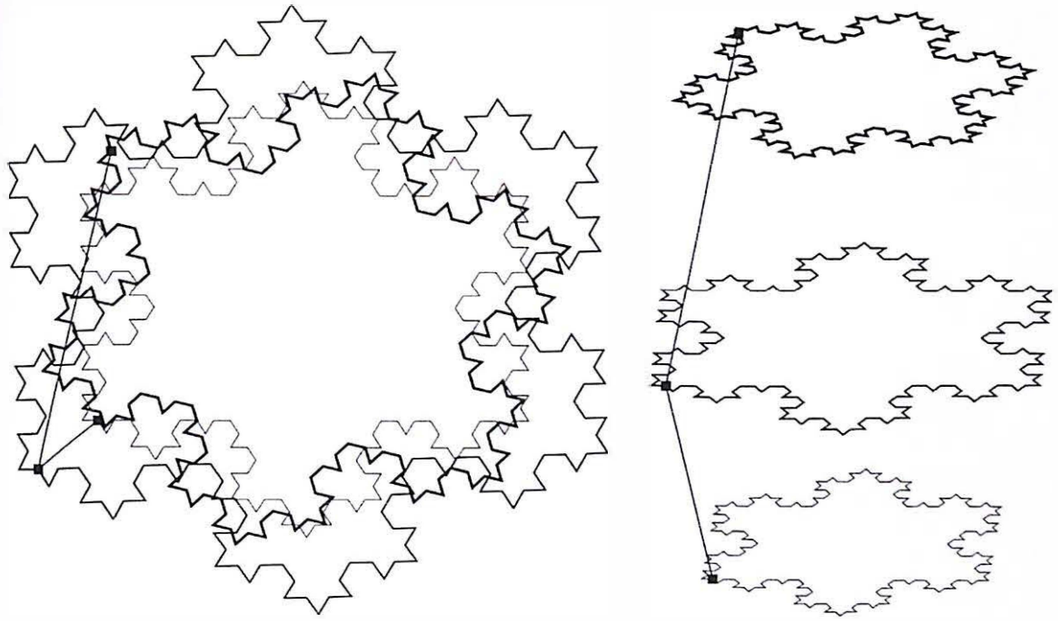
1. The curves' **order**. How to order the generative curves of a loft is the first thing we need to control before even creating a pattern or a texture on the surface. We can determine the order directly by selecting the curves in their proper order within Rhino using a 'Curve' parameter. Do not select them by window as it could change the expected order. It is better to select them one by one in the desired order.
2. The curves' **directions**. The direction of the curves relies on their own direction. This can be changed using the command 'Flip' in Rhino.
3. The **seam** of the resultant surface or polysurface. It will depend on the own curves' seams. If we use the same curve, interesting Breps can be created just by rotating them, as by default the loft's seam will follow the curves' seams.
4. The loft **styles**. This can be explored on either the Rhino or Grasshopper® options input.

The following example is based on the Koch curve. It is a fractal curve also known as 'snowflake curve'.

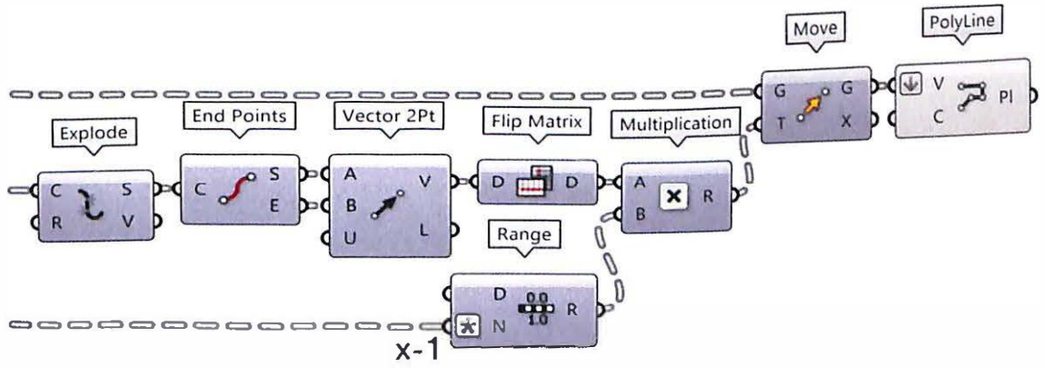


There are three curves in the model. They are rotated and scaled. We can see the origin of the curves displayed as dot. If the loft is created in Grasshopper®, the loft's seam will follow the curves' seams by default. If it is done in Rhinoceros®, the default loft will follow what has been defined as the 'automatic' option, which finds the most simple surface available. If we want the polysurface's seam to follow the origins of the curve, we must select 'Natural' in the options in the command line.

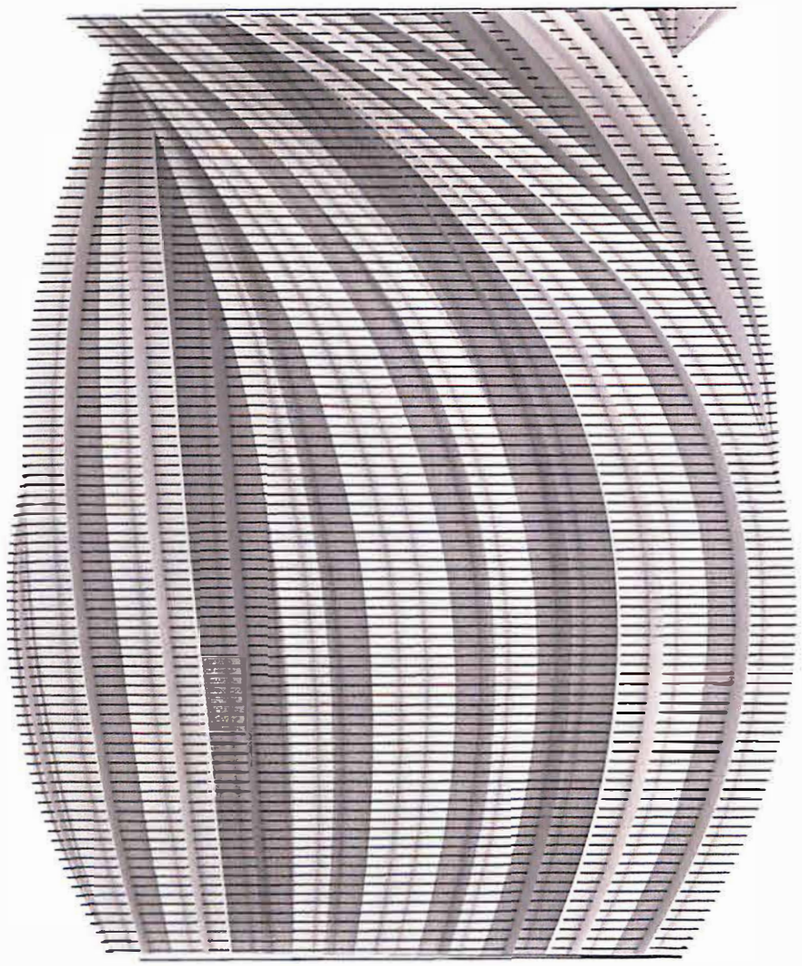




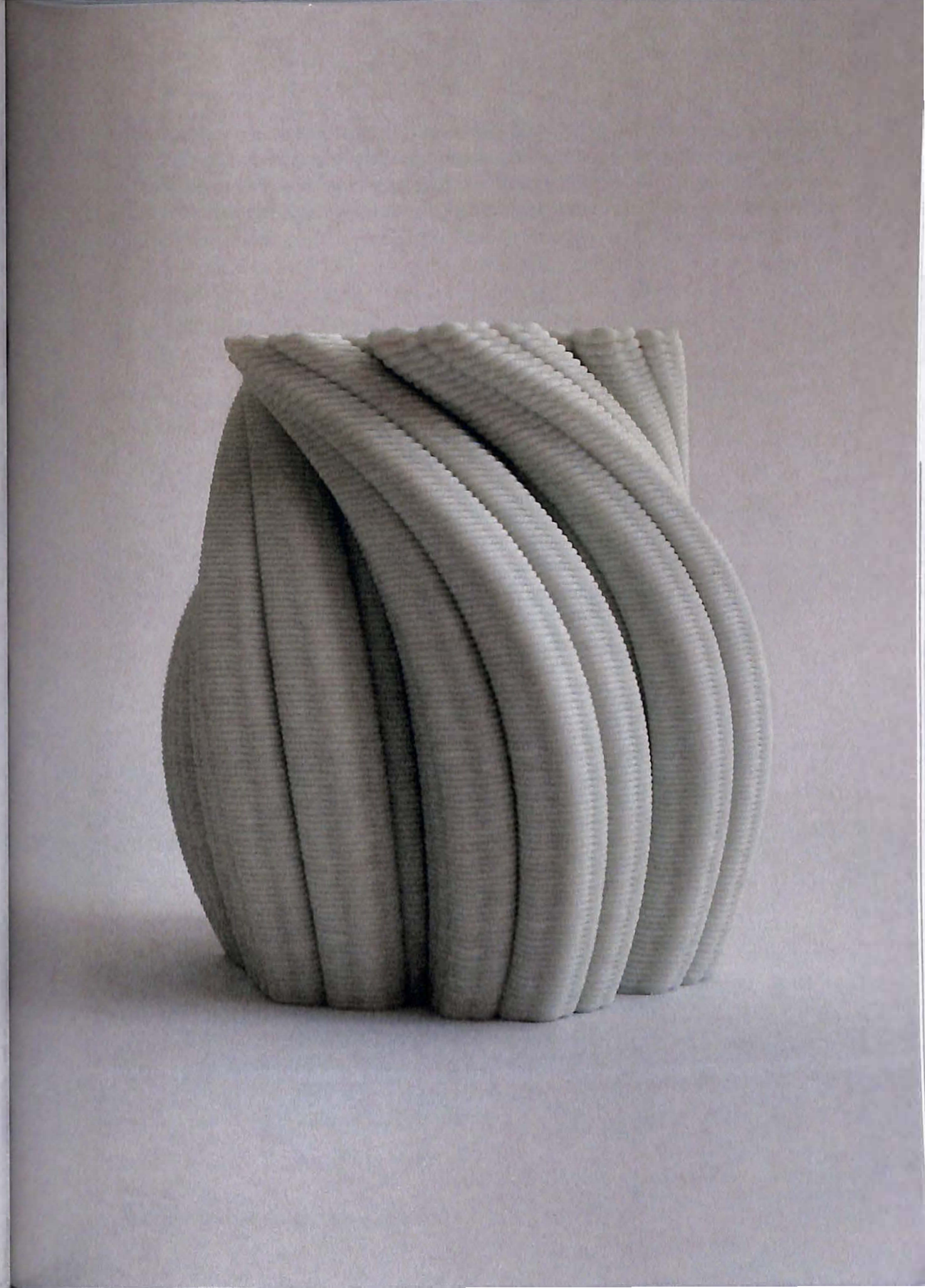
The result is a polysurface. For 3D printing it, we can create a continuous polyline with the definition explained at previous chapter: *Brep -> Curves -> Polyline -> Points -> X,Y,Z*. The result from 'Contour' are polylines. If we use 'Divide curve' as explained, the result will not be accurate. In this example and every time that the Brep is a polysurface with sharp edges, it is better to use the 'Control points' component in combination with 'Cull index' as explained at chapter *Mesh -> Polylines -> Points -> X,Y,Z* with closed meshes.



As the number of division points depends on the amount of control points, we can use 'List Length' to measure the list of points. The other components in the definition remain the same.



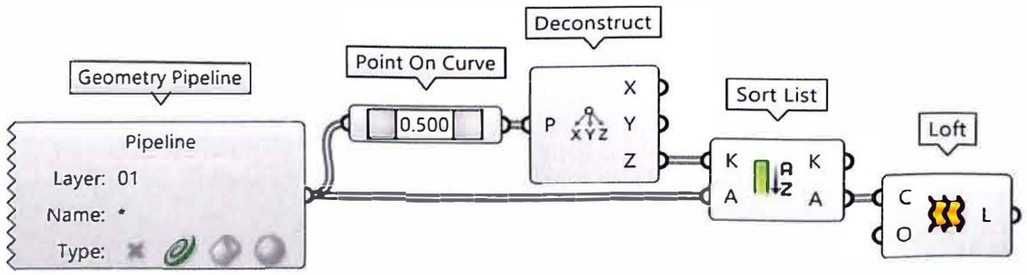
The result should be a continuous spiralized polyline. Above, the polysurface and the curve for the 3D printer's path. On the right, the curve, 3D printed in porcelain.



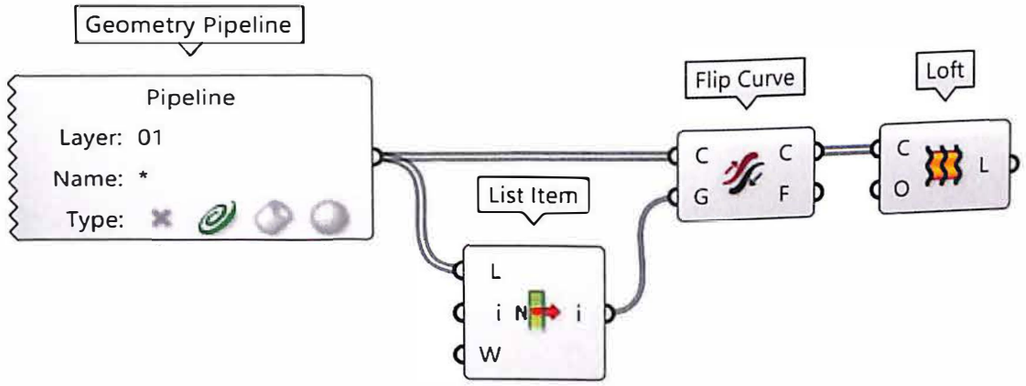
For more 'advanced' and dynamic lofts, there is the tool 'Geometry Pipeline'. This tool selects the curves previously drawn in Rhinoceros® according to the layer (e.g. layer '01'). In Grasshopper® right click on the component 'Geometry Pipeline', select 'Curve' filter and write the name of the layer were the shapes are '01'.

It is more than likely that the curves will not be in proper order, direction and/or alignment. As the curves are now referenced into Grasshopper®, we need tools to control these features inside it:

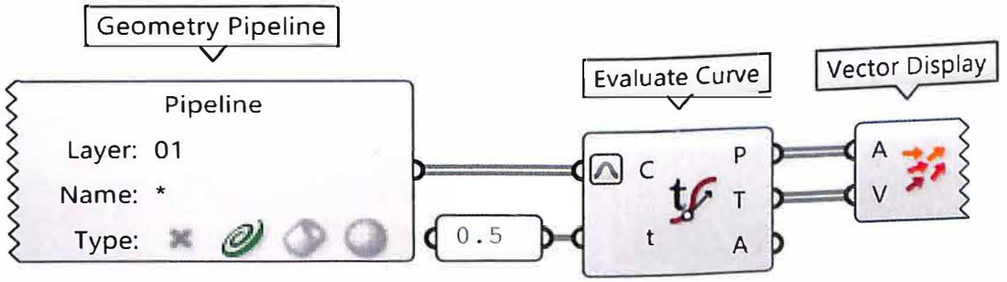
1. The curves' **order**. The main tool to use is 'Sort list'. It can give a new order to a list of 'Values' according to a list of synchronous numbers ('Keys'). To create numbers associated to the curves, use 'Point on Curve'. This will evaluate a curve at a specific point. Use 'Deconstruct' to deconstruct a point into its three component parts or coordinates. 'Sort List' will sort the curves according to their Z coordinates:



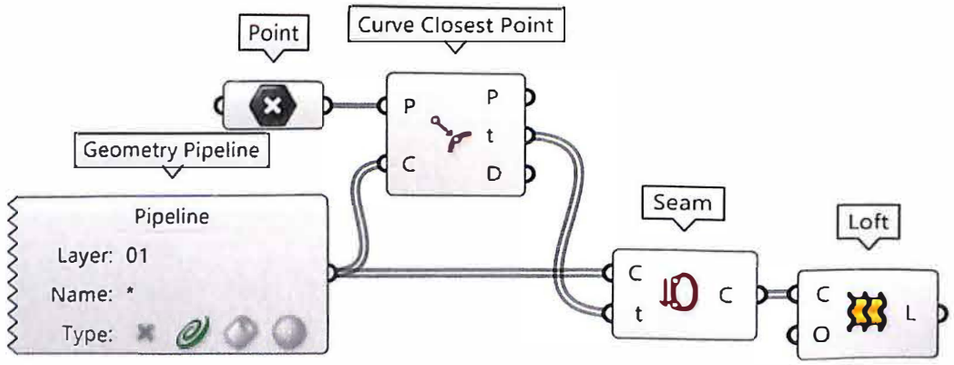
2. The curves' **directions**. If the curves have opposite directions, the loft will twist. Clockwise curves will create a Brep with the positive normal vector to the interior of the object. Counterclockwise curves, will create a Brep with the positive normal vector to the exterior of the object. The normal of the Brep could affect to ulterior operations but the most important thing is that all the curves are in the same direction otherwise the loft will twist. The component 'Flip Curve' has two inputs. Feed 'Curves' input with the collection of curves from 'Geometry pipeline' and as for 'Guide' we can use any curve. We can select one of them with 'List Item' component, then all the curves will get the same direction as that one. If no curve is used as a guide, then all will be flipped, but that will not solve the problem with the loft.



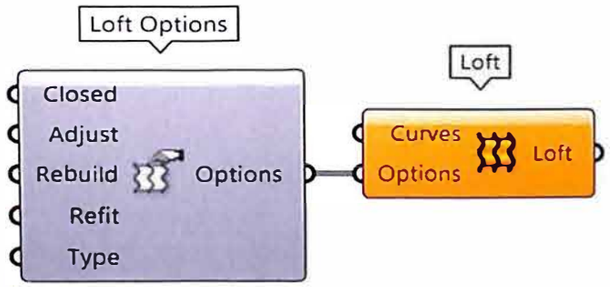
If the user is interested in checking out the curves' directions, a combination of 'Evaluate Curve' and 'Vector Display' should be used. If the curves are reparametrized in the 'Curve' input a value between 0 and 1 can be used to evaluate the curves.



3. The **seam** of the resultant surface or polysurface. By default in Grasshopper® , the seam of a loft will follow the origins of the curves. It is equal to the 'Natural' option in the Rhinoceros® command bar for 'Loft'. There is the option to 'Align sections' in the 'Options' input and the loft will be aligned in a similar way to the 'Automatic' option in Rhinoceros®. If none of these situations solve our loft, or we need to control the seams in a different way, there is the tool 'Seam'. It needs the curves and the 'Parameter' on each curve to reset the seam or new curve's origin. The parameter in the curves could be aligned using a reference point in Rhinoceros® and a 'Curve Closest Point' component or via other methods.

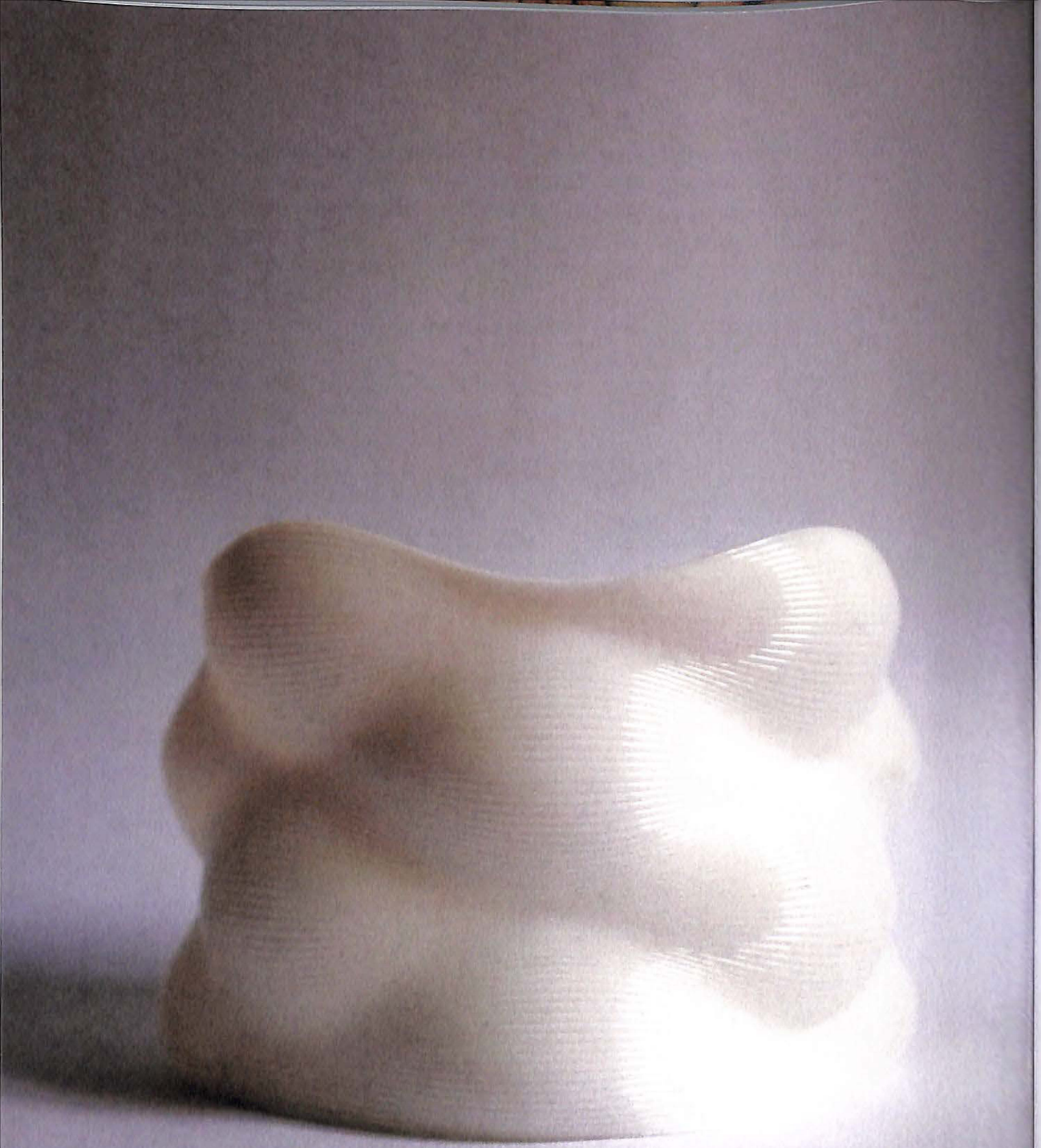


4. The loft **styles**. That can be explored on the Rhinoceros® or Grasshopper® 'Options' input. They can be also selected using the 'Loft Options' component were the types are defined by numbers (1=Normal, 2=Loose, 3=Tight, 4=Uniform, 5=Straight) or by a right click.



Sometimes it will be necessary to solve all these issues together moving the curves from one strategy to the next one:

Select curves → Sort → Flip → Seams → Loft Styles → Loft



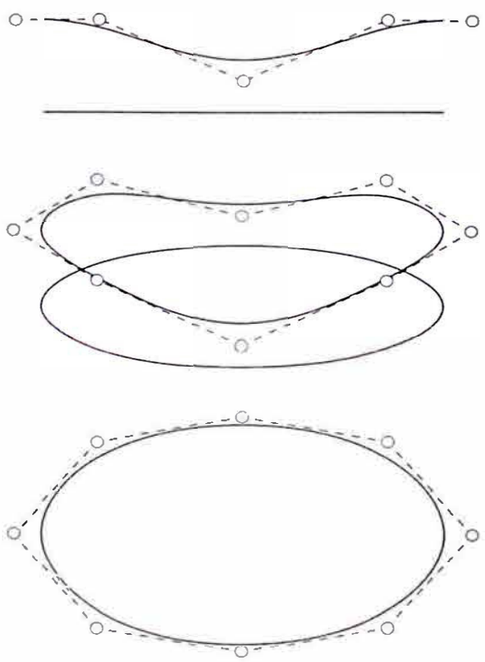
3D printed part in white PLA

Isocurves

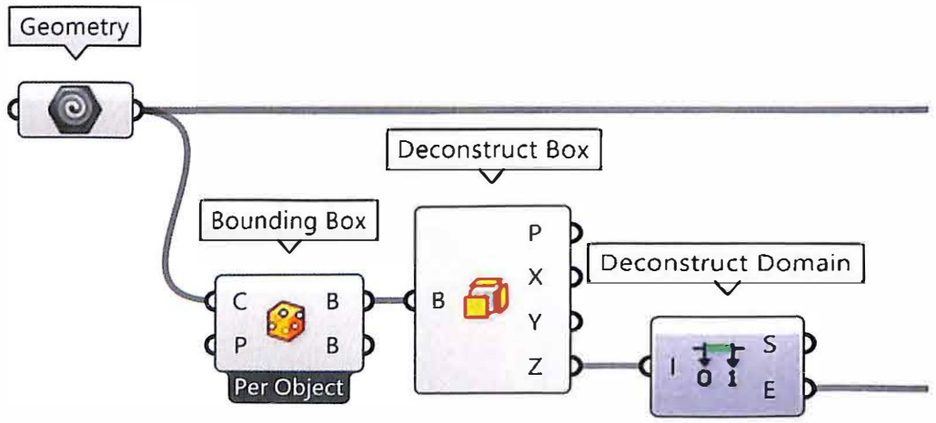
The isocurves of a surface are virtual curves that follow the same parametric information on the surface. They are also called isoparametric curves. By default, isocurves are displayed at NURBS' knots. If there are no intermediate knots, like in a rectangular simple surface, then they will follow the mid points of the surface for U and V directions. Those isocurves could be used to create paths for the 3D printer, related rather than the topology of the surface rather than with section planes as used in the previous contour strategies.

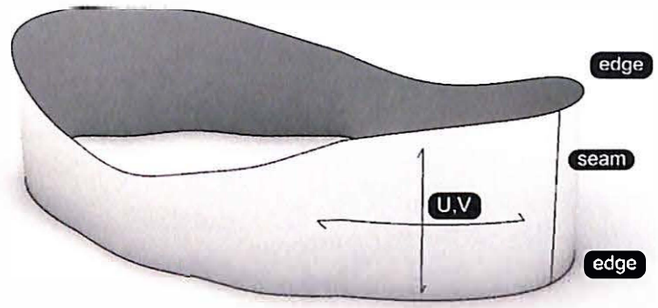
The following example uses two ellipses to create a loft. The paths for the 3D printer will be created using the surface isocurves. The steps as are following:

1. In Rhinoceros® draw an ellipse. Copy the ellipse in the Z axis using the function 'Copy' or using Gumball. The upper one is modified dragging the midpoints downwards. As an ellipse is a degree 2 curve, manipulating it by its control points could create kinks at some points. It is better to rebuild them with 'degree 3' to have continuous curves. 'Rebuild' allows you to modify the amount of control points as well as the curve's degree.

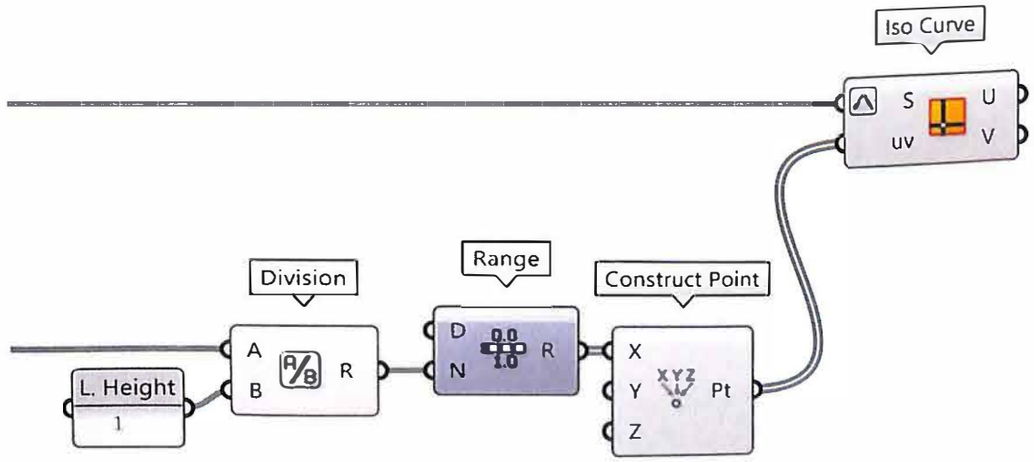


2. The 'Loft' can be made in Rhinoceros® or in Grasshopper®. It is important to understand the topology of this surface. It has four edges: the lower one, the upper one and two more collapsed into a single edge called seam. The U and V directions will correspond with the Z vector and the perimetral direction.



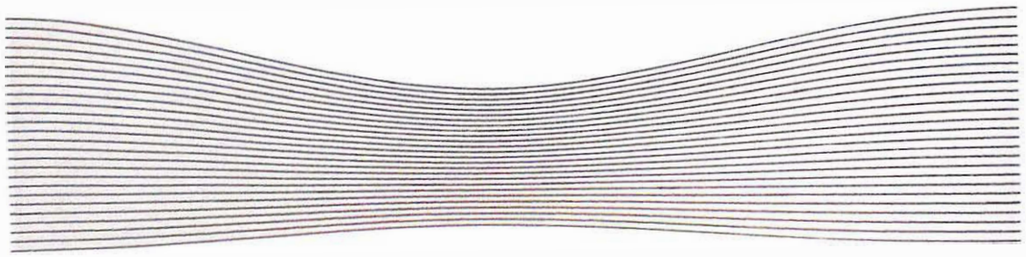


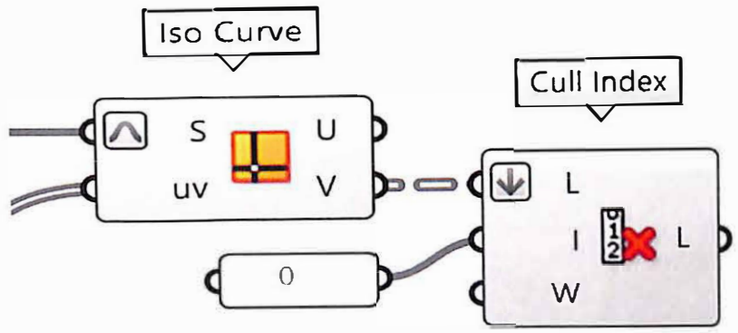
3. To begin creating isocurves from the loft, we use the component 'Iso Curve'. As they will not be parallel to the XY plane, there will be parts where they will be in closer proximity than in others. In the interest of getting a proper height for the set of isocurves, we need to find the greatest height of the surface in order to calculate the divisions of the surface. It is better to have as a reference the greatest and not the smallest height as that way the clay will be more compressed on the smallest distances, but never in the air. For this, use a 'Bounding Box' component followed by 'Deconstruct Box'. The 'Z' output will provide the box domain for the height of the loft. As usual, we need to divide it according to the layer height. As the 'Iso Curve' component needs to be fed with UV points of the surface, we will need to transform the amount of divisions into UV points. A possible method consists of dividing a 0 to 1

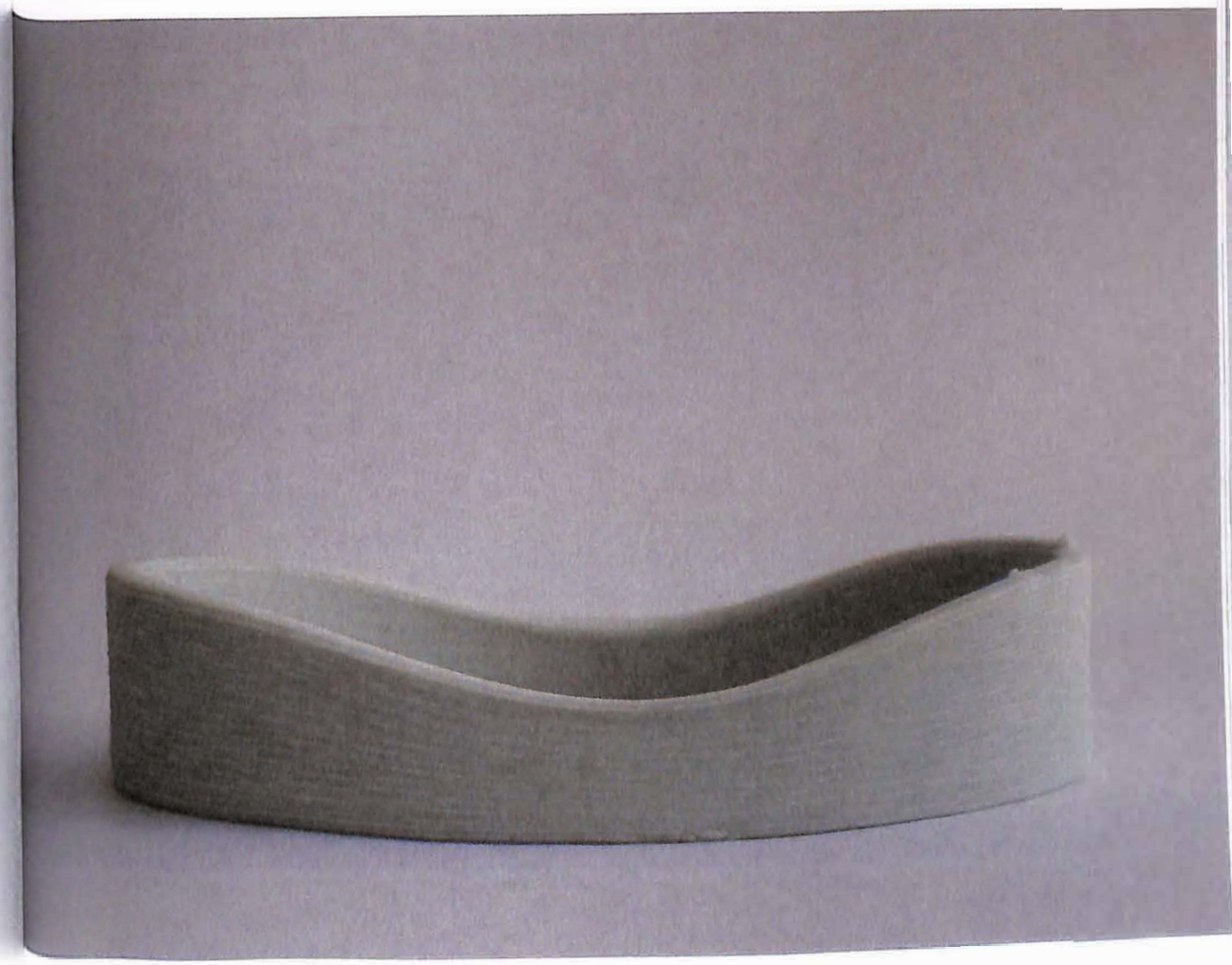


Domain with 'Range' and 'Construct Points' with the output from 'Range' as the 'X' coordinates of the points. As the values go from 0 to 1, the surface must be 'reparametrized' too. A surface, as well as a curve, can be reparametrized. That means that whatever the mathematical domains are, they can be re calculated into domains from 0 to 1. This makes some calculations easier as sometimes it is hard to deal with the default domain values. In order to reparametrize the surface, right click on 'Surface' and select 'Reparameterize'.

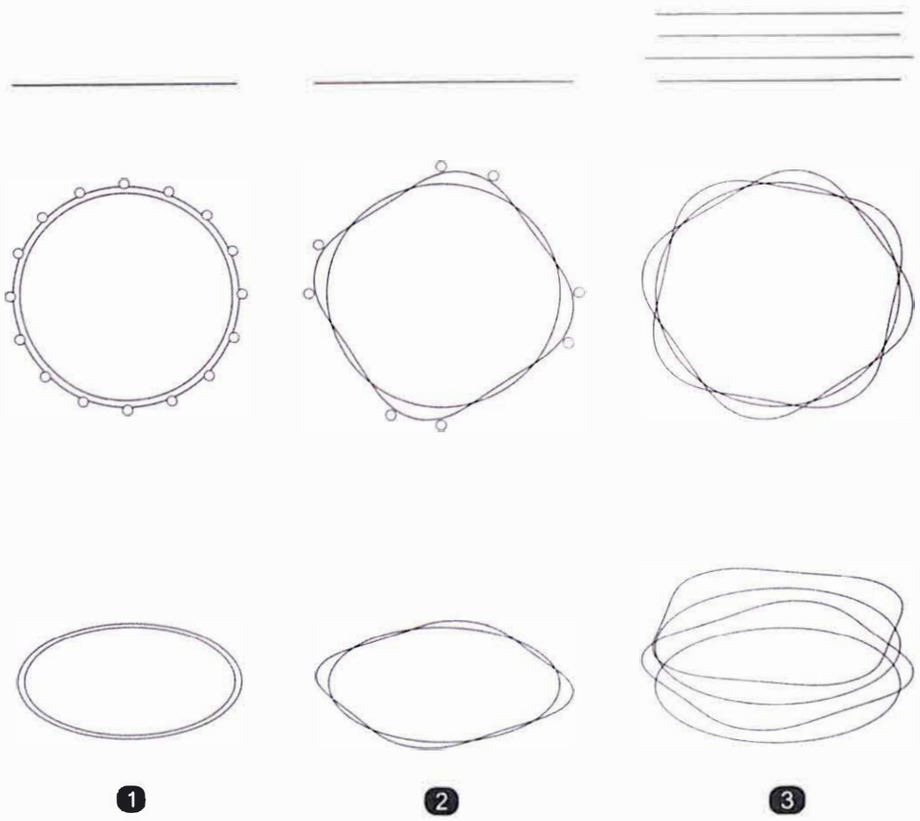
4. In this example the loft has a height of 25 mm and a layer thickness of 1 mm, therefore we need a total of 25 layers as previously stated. After 'Range', the total amount of values is 26. This is logical, as if we divide a domain in 25 parts, we will get 26 values. This means that we have curves coinciding with the lower and the upper edges of the surface. It is at the reader's choice to remove the first of them or not with a 'Cull index' component for example.
Note: 'Iso Curve' component only inputs surfaces. So, this method will not work with polysurfaces.



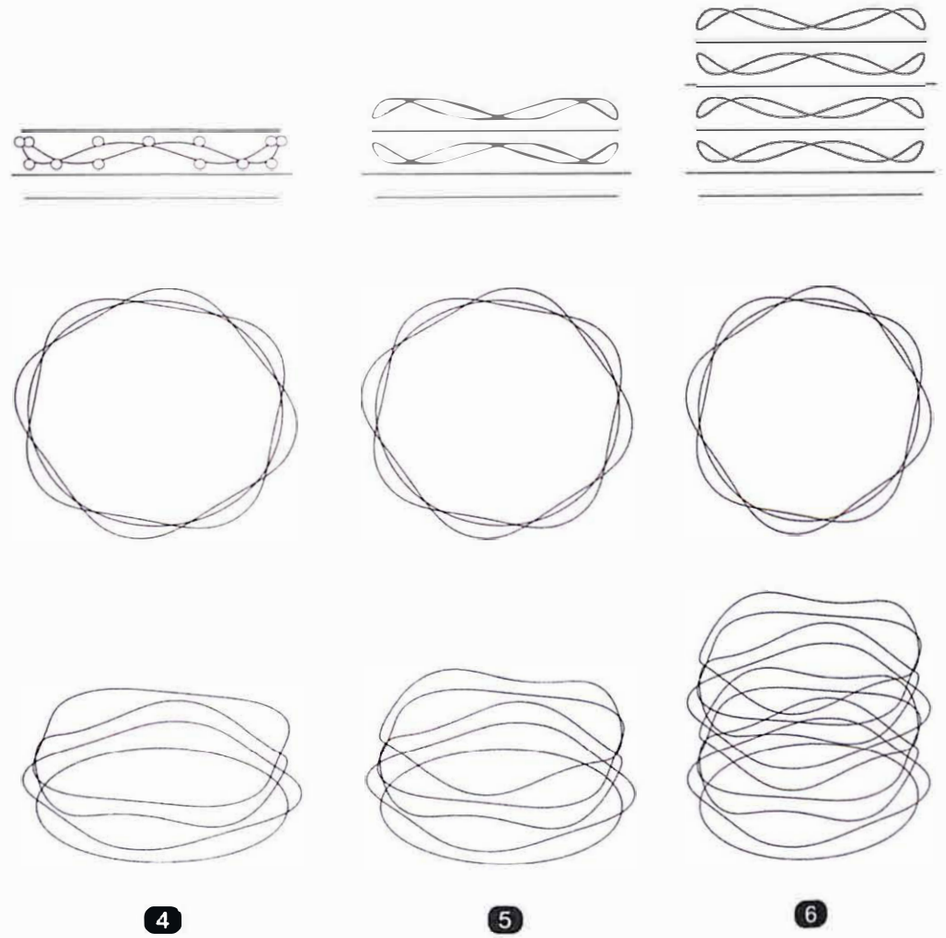




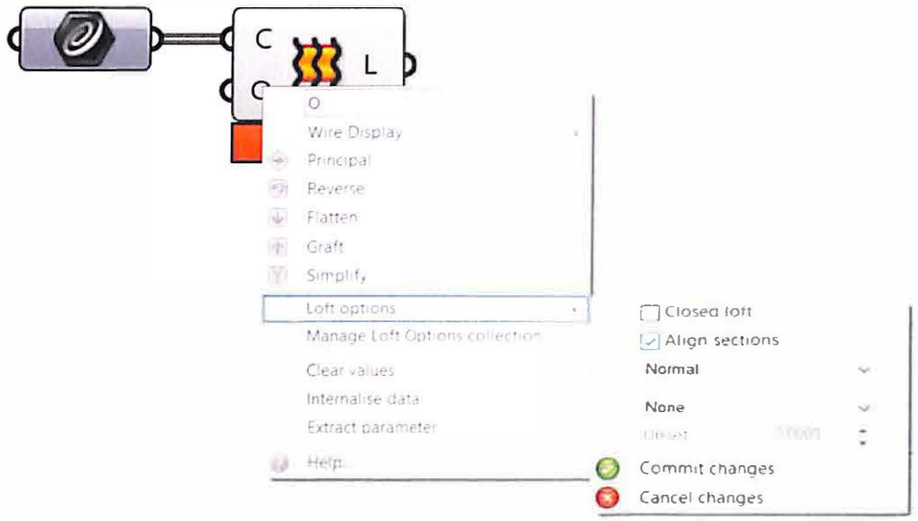
Many examples can be created and printed using isocurves. Below shows the process for one of them.



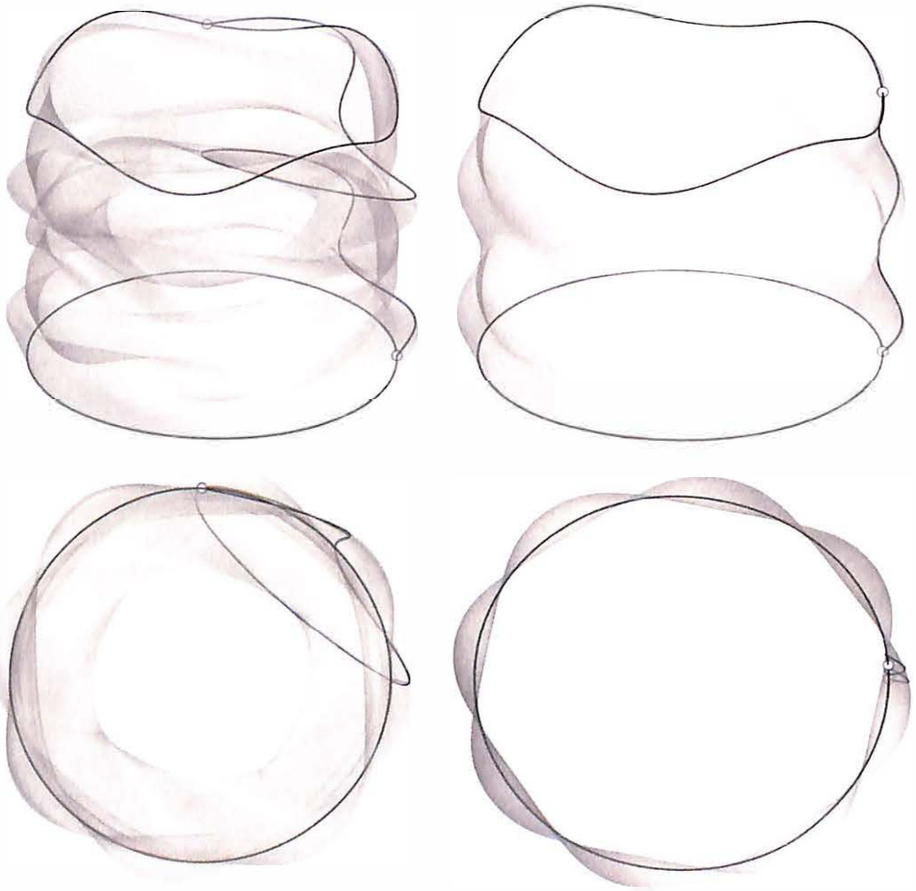
1. In Rhinoceros® draw a circle (e.g. 100mm. diameter) and offset it 3mm. towards the interior. As circles are quadratic curves, it is better to rebuild them into continuous 'degree 3' curves in order to distort them. 16 control points were chosen for this example.
2. Scale two control points every other two points.
3. Rotate the curve 45 degrees. Move the curves upward.
4. Select two control points every other two and move them downward. Select the other 8 points and move them in opposite direction creating a weaving curve.
5. Copy that weaving curve and rotate 45 degrees.
6. You can then copy that set of curves as many times as you want.



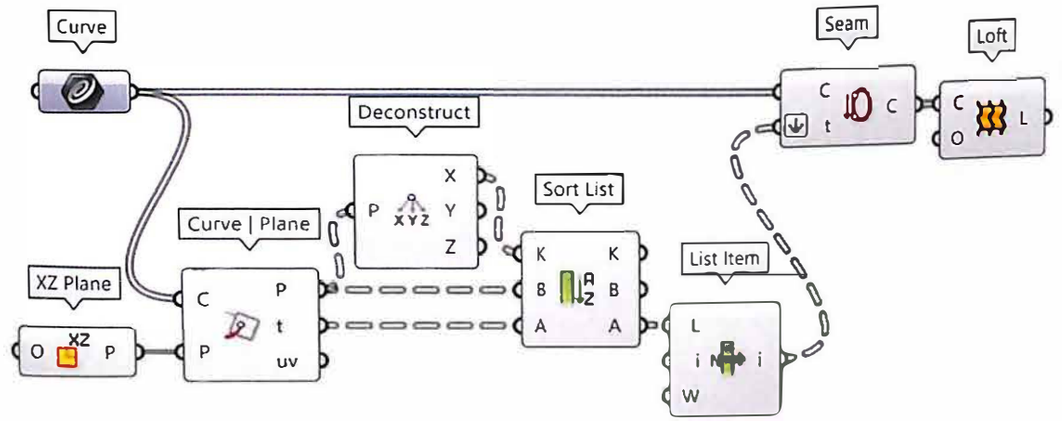
It is time to make the loft. We can reference the curves by order 'bottom → top' with a 'Curve' component as usual. Then, connect it to 'Loft'. According to the previous 'Loft chapter', the order will be fine as we have selected the curves manually, and the direction will be the same as they were mainly copies of the first circle, but the seam will not be straight. By default we get the first surface (see next page). There is the option 'Align sections' in loft options, which will improve the loft considerably, but the seam will become distorted.

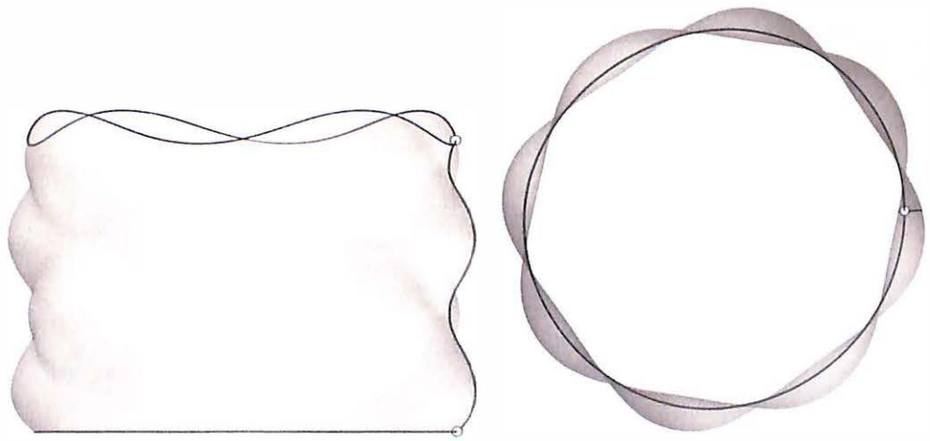


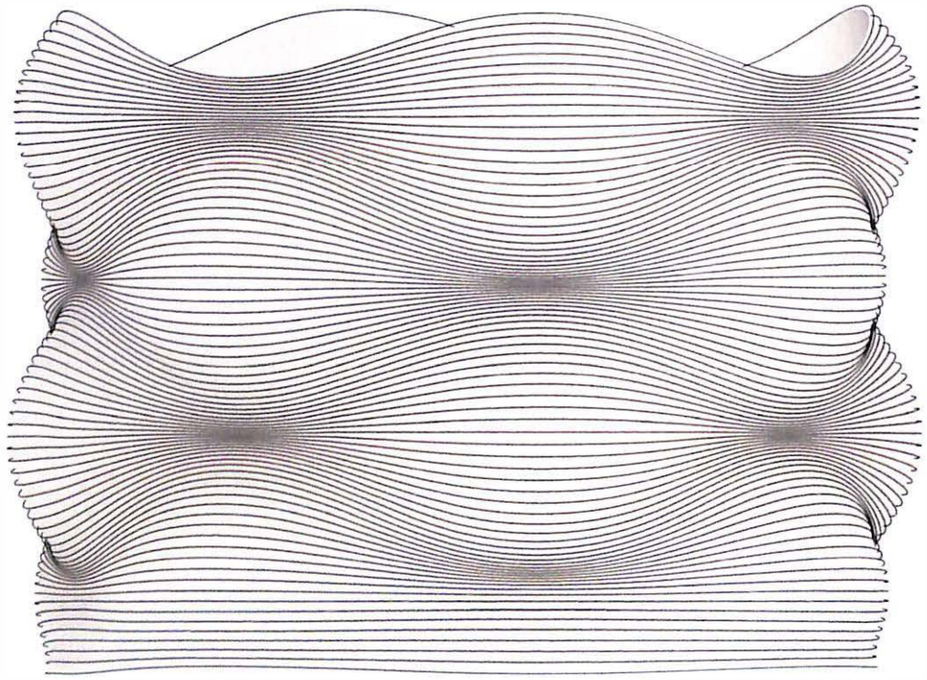
Before and after 'Align sections':



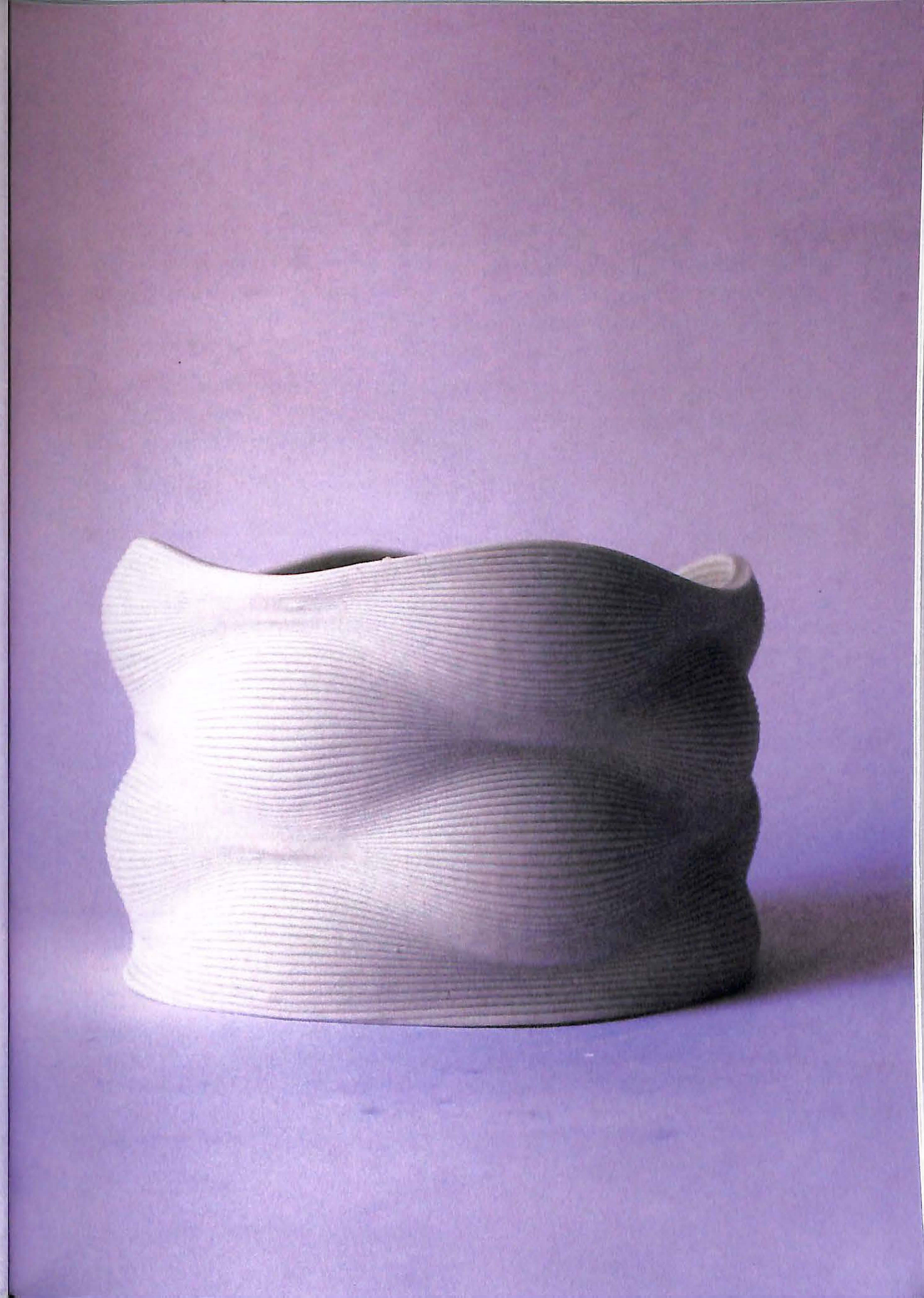
Use 'Seam' to create a straight vertical seam. A possible method is using an auxiliar 'XZ Plane' to intersect it with the curves. The parameters at the intersections will become the new origins of the curves and make a loft where the seam is straight following those exact points. The intersection 'Curve | Plane' outputs two points per curve. However, they are in opposite directions and sometimes the first of the list will be on the 'right' and sometimes on the 'left' of the curve. So before selecting one with 'List Item' we need to sort them according to the X coordinate of the points with 'Deconstruct Point'. Once the new parameters 't' feed the 'Seam', the loft will have a straight seam. Now it is important that the option 'Align sections' is disabled or there will be no effect on the loft's seam.







To achieve the isocurves, we must repeat the steps described in the previous example.On the right, the isocurves 3D printed in clay.



Flow (E)

In models were the layer height is not constant like in the previous examples, the amount of extruded material must be controlled in order to have a constant thickness of the wall. Otherwise we will get a larger amount of material in the layers with shorter height.

There are several ways to find out the flow according to the different layer height, but all of them require controlling the value for E. The one we are going to explain uses the method explained at chapter:

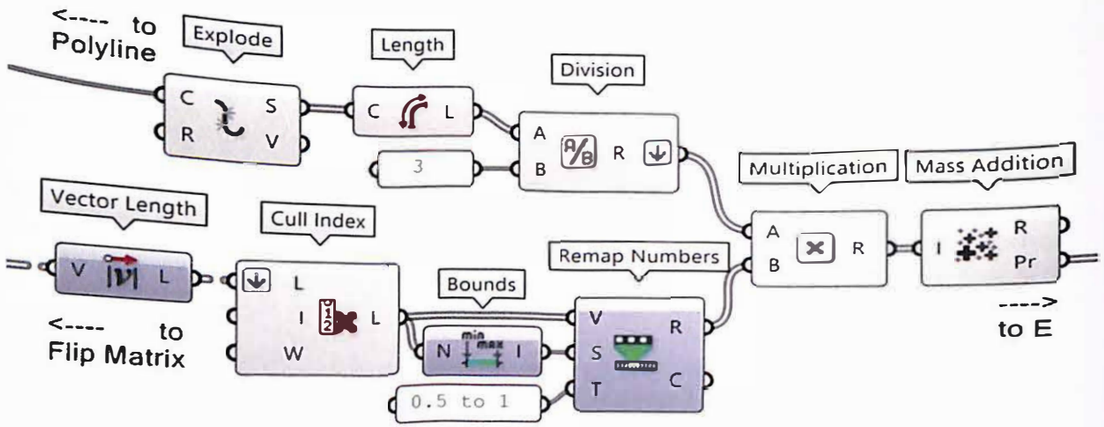
Brep -> Curves -> Polyline -> Points -> X,Y,Z

In that chapter we learnt how to create a continuous polyline avoiding the seam and most importantly, following the tangent to the surface.

To create the polyline, every point was moved in the direction of the corresponding point of the next contour. The vector for the movement was designated using a 'Vector 2Pt' component. It is important to review the definition and its explanation at chapter *Brep -> Curves -> Polyline -> Points -> X,Y,Z*

We use these vectors to determine a factor that will modify the E (flow) parameter. The component 'Vector 2Pt' has an output that outcomes the length of the vectors. Be careful, not to use it directly as in the definition we had a following 'Flip Matrix' component that rearranges the data tree. We must get the vectors from 'Flip Matrix' and measure their length. Use 'Vector Length' to get the length of the vector, also known as amplitude or modulus. Then we have to transform or remap those values into a percentage which is able to modify the values for E that come from the lengths of the polyline's segments, as explained in chapter '*Points -> X,Y,Z*'. There, the lengths of the polyline's segments produce the values for a 'Mass addition' component.

Well, if we want to modify those numbers by a percentage according to the layer height, the lengths of the vectors must be remapped into a value from 'something' to 1. A multiplication by 1 will not affect E, but a multiplication by a smaller number than 1 will decrease the value for E as shown in the following example:

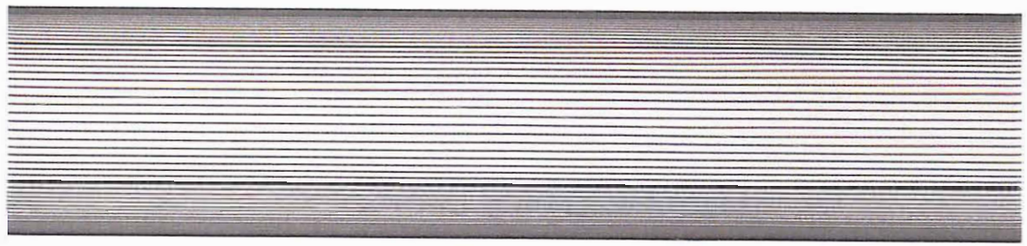


The upper components 'Explode' and 'Length' are the ones from the beginning, the ones that read the length of the polyline's segments. The 'Division' component controls the extrusion dividing the length by 3 in this example.

The lower components 'Vector length', 'Remap numbers' and 'Multiplication' create a number between 0.5 and 1 that represents a percentage according to the distance between layers and modifies the E values. Before remapping the 'Vector Length', we need to remove the first value as it works with the points themselves and 'Explode' works with segments between points. This means that in the lower row we will always have one more value than in the upper one. To remove the first value and so that the lists match at 'Multiplication' we can use a previously explained component, 'Cull Index' with a 'zero' value at 'I'. The component 'Bounds' is very useful when used in combination with 'Remap Numbers' as it produces the domain of the lengths. This means that the shortest length will get the smallest coefficient (0.5), the largest one the biggest (1) and the rest of lengths a proportional coefficient.

With this simple definition we can work on models as the previous ones or as following, without risk of overflow.

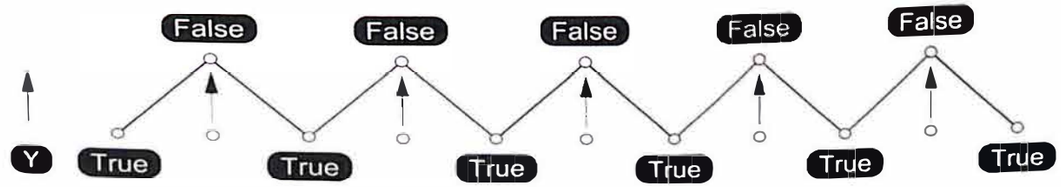
In the images below, the layers' gradient and a 3D printed part in white PLA showing variable layer heights.



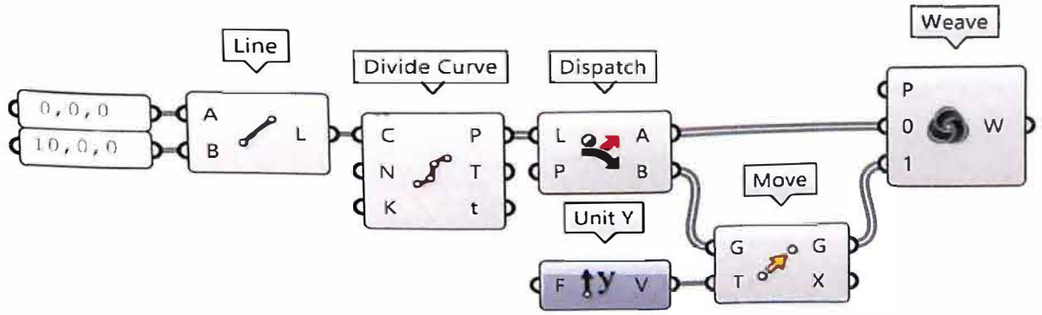
3D printed part in porcelain

Waves

Weaving patterns have a lot of design possibilities. This can be done in a mathematical way, or with attractors such as points, curves, vectors or images. But before going into design possibilities, let's understand the basic tools necessary for their creation, first, with a single line, and then with a surface like a cylinder.



1. The basic strategy is based on selecting items, one 'yes' (True) one 'no' (False) from a list of points and moving the 'false' ones in a certain direction. For the previous example, use 'Line'. Connect one 'Panel' to Start Point (e.g. with the values (0,0,0)) and one more 'Panel' to End Point (e.g. (10,0,0)). Connect 'Divide Curve' to 'Line' to get the list of points.

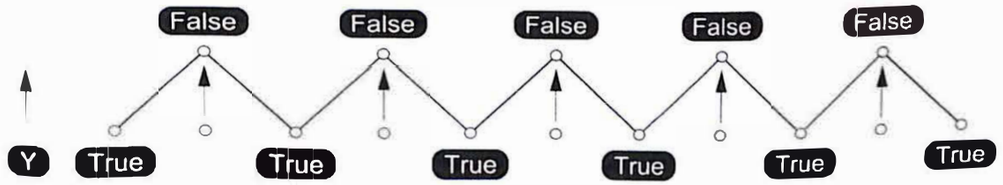


2. To separate the true and false points, use the function 'Dispatch', with 'List A' being the true values and 'List B' the false values. This is down to the 'Dispatch pattern'. By default, the pattern is TRUE-FALSE, so the first point will be TRUE, the second FALSE, the third TRUE and so on, till the end of the list of points.

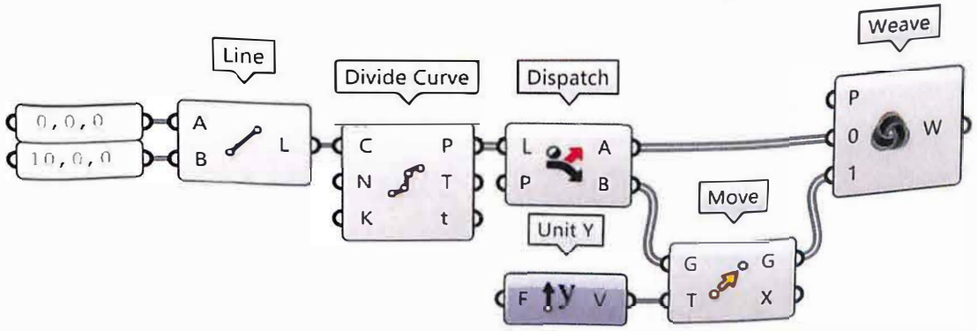
3D printed part in porcelain

Waves

Weaving patterns have a lot of design possibilities. This can be done in a mathematical way, or with attractors such as points, curves, vectors or images. But before going into design possibilities, let's understand the basic tools necessary for their creation, first, with a single line, and then with a surface like a cylinder.



1. The basic strategy is based on selecting items, one 'yes' (True) one 'no' (False) from a list of points and moving the 'false' ones in a certain direction. For the previous example, use 'Line'. Connect one 'Panel' to Start Point (e.g. with the values (0,0,0)) and one more 'Panel' to End Point (e.g. (10,0,0)). Connect 'Divide Curve' to 'Line' to get the list of points.



2. To separate the true and false points, use the function 'Dispatch', with 'List A' being the true values and 'List B' the false values. This is down to the 'Dispatch pattern'. By default, the pattern is TRUE-FALSE, so the first point will be TRUE, the second FALSE, the third TRUE and so on, till the end of the list of points.

3. Use 'Move' to separate the false points. This movement could be done in Y direction. For the development of curved surfaces, this movement will be done in the normal direction to the surface at that point.
4. Now use 'Weave' to intertwine the two list of points as they were before the motion. Connect 'Stream 0' of 'Weave' to 'List A' of 'Dispatch' and 'Stream 1' of 'Weave' to the output of 'Move'.
5. Optionally use 'Polyline' to create a curve that displays the path for the wave after 'Weave'.

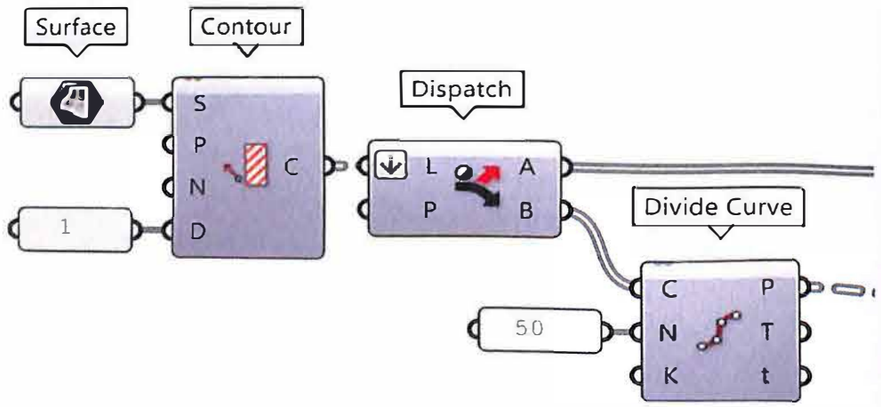
With the basic tool understood, we can now work on a small planar surface aligned to the XZ plane with the following parameters:

Length: 50mm. in X direction

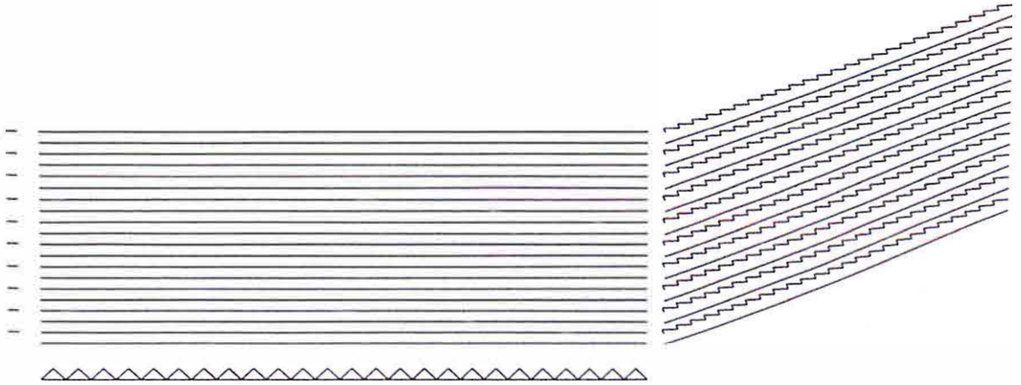
Height: 20mm. in Z direction

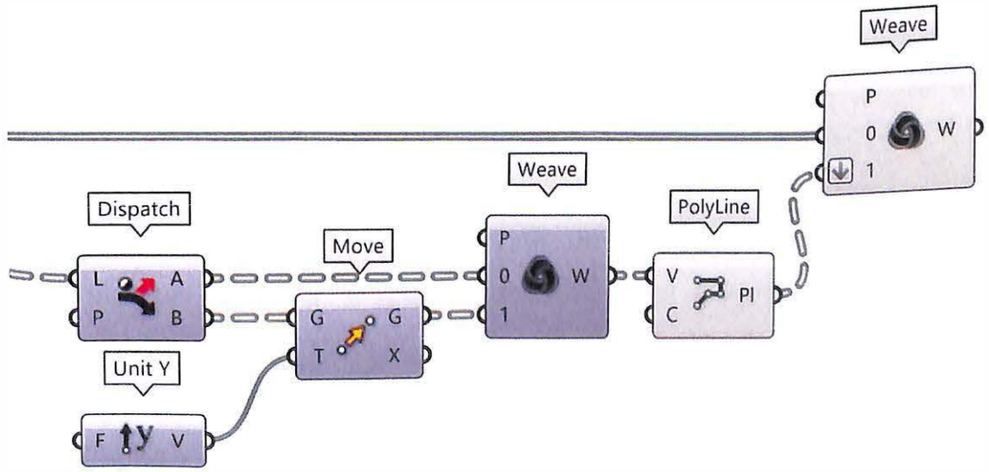
Layer height: 1mm.

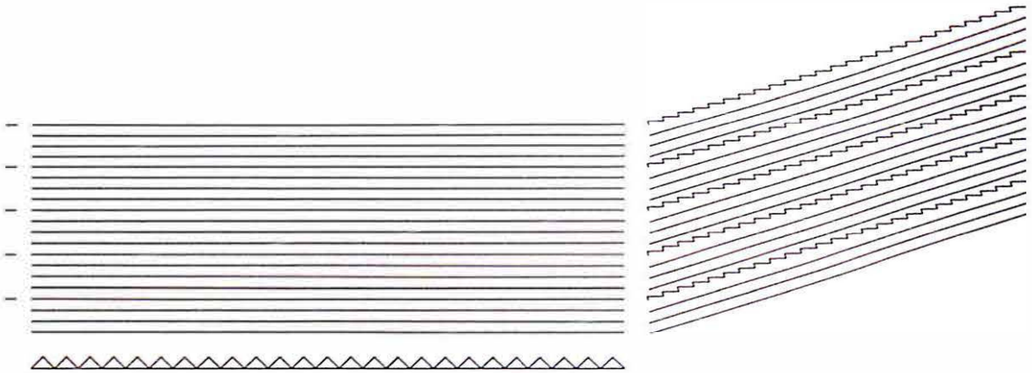
As explained in previous examples, the tool used to section the surface is 'Contour'. We could design waves in all the sections. In this case it is as simple as applying the previous definition to the curves. That would create an effect similar to an undulating surface. What could happen is that we would prefer to change the wave from one layer to the next or alternate waves and straight lines. In this case, before working on the points of the curves, it is necessary to 'Dispatch' the list of contours:



1. After 'Contour' it is necessary to 'Flatten' the tree of data as it outputs every curve into a single list.
2. Use 'Dispatch' to split the list of contours, one 'true' one 'false'. By default, 'Dispatch' uses a TRUE-FALSE pattern. If another rhythm is desired, other Booleans could be added inside the 'Dispatch pattern' or in a 'Panel'.
3. 'List A' of curves, will correspond with the odd list of numbers. 'List B' will correspond with the even ones.
4. 'List A' remains as it is.
5. 'List B' suffers the 'waving grammar'.
6. To join both lists again in the previous order use one more 'Weave' component:







The pattern at the first 'Dispatch' can be modified. A pattern like TRUE TRUE TRUE FALSE will create three straight lines and a wavy one. To maintain the order in the last component ('Weave') it is necessary to add the equivalent and opposite pattern 0 0 0 1

In Grasshopper®, a TRUE or FALSE parameter is named as Boolean. TRUE is equivalent to 1 or 'yes' and FALSE to 0 or 'no'. So, words or numbers can be used in an equivalent way.

Based on this premise, if the user prefers to generate a panel for 'Weave' by using the panel that feeds 'Dispatch' as an input, we can do this in a simple way by using an 'Expression' component. Inside the component is written:

If(x=0, 1, 0)

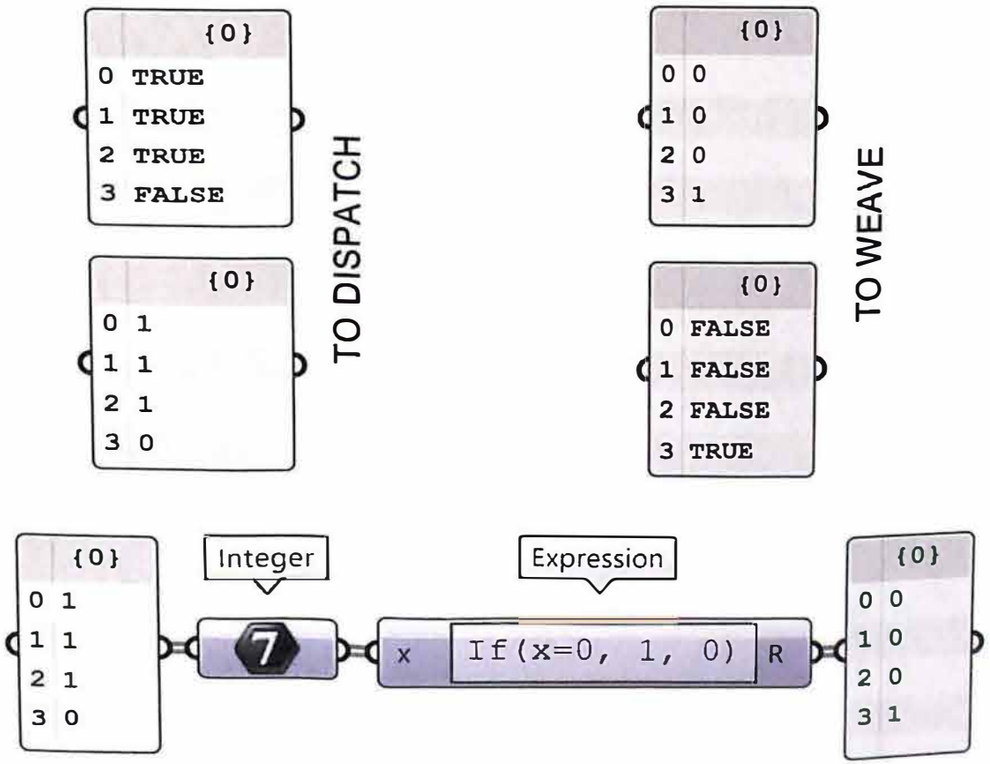
Which means: If x=0, then x will be 1. If not, then x will be 0.

The 'Integer' parameter is necessary to transform the panel into a list of numbers, as it is a parameter that can only store integer numbers.

It is very common the use of conditional sentences in Grasshopper® with =, <, >, etc.

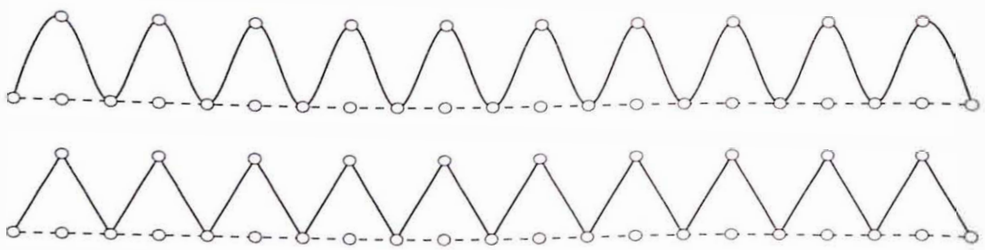
Thanks to that expression, the list of values from the first panel will be transformed into the second:

1 1 1 0 → 0 0 0 1

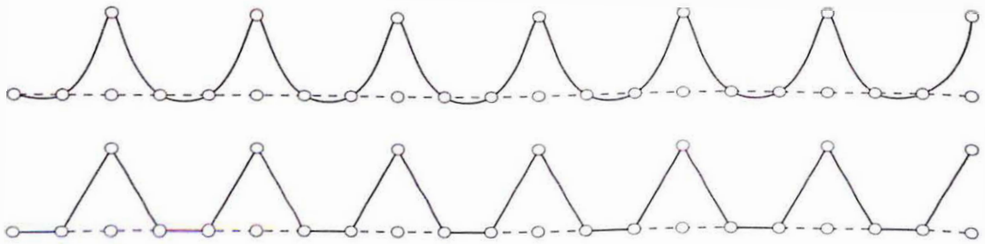


Similar logics could be used also to define the rhythm for the wave.

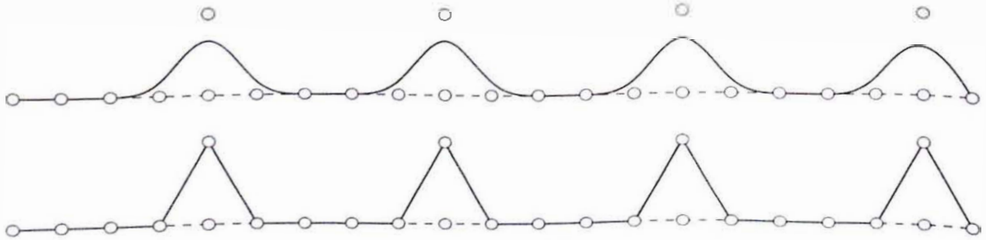
Examples in the next pages. With 'Nurbs', 'Polyline' or 'Interpolate' curves.



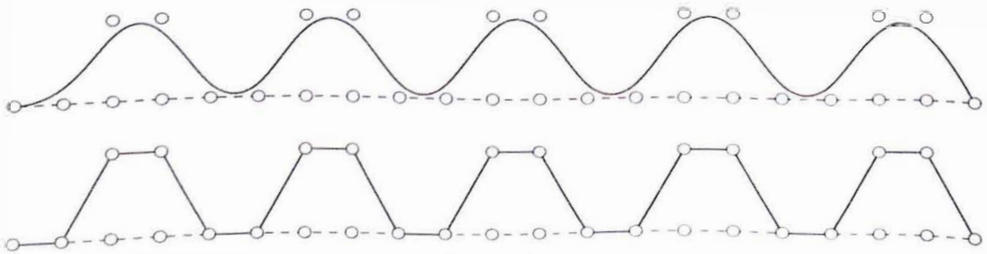
1-0



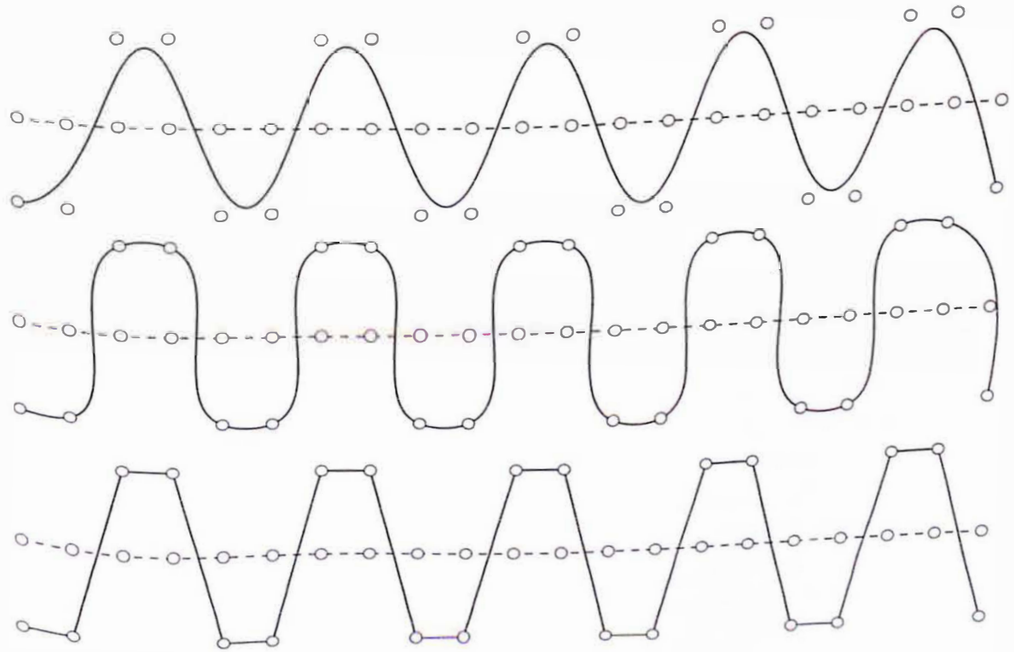
1-1-0



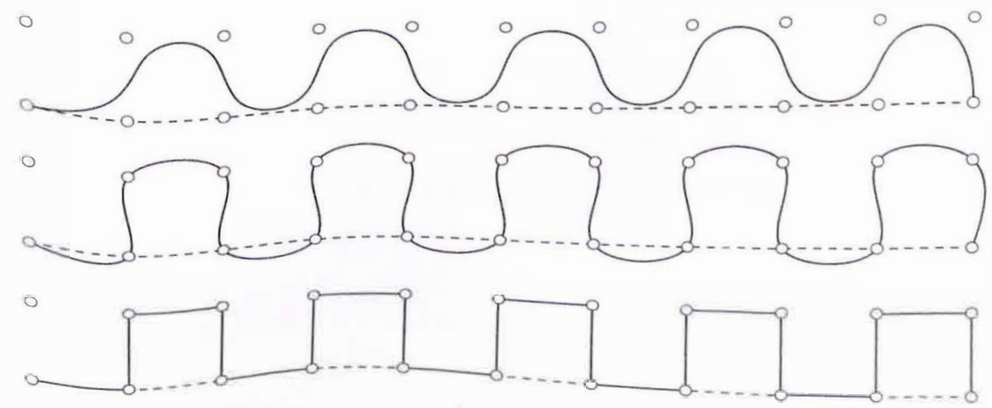
1-1-1-1-0



1-1-0-0



1-1-0-0



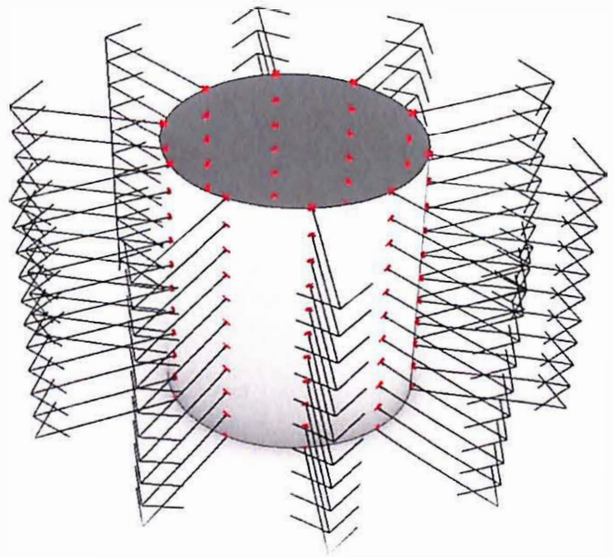
1-1-0-0

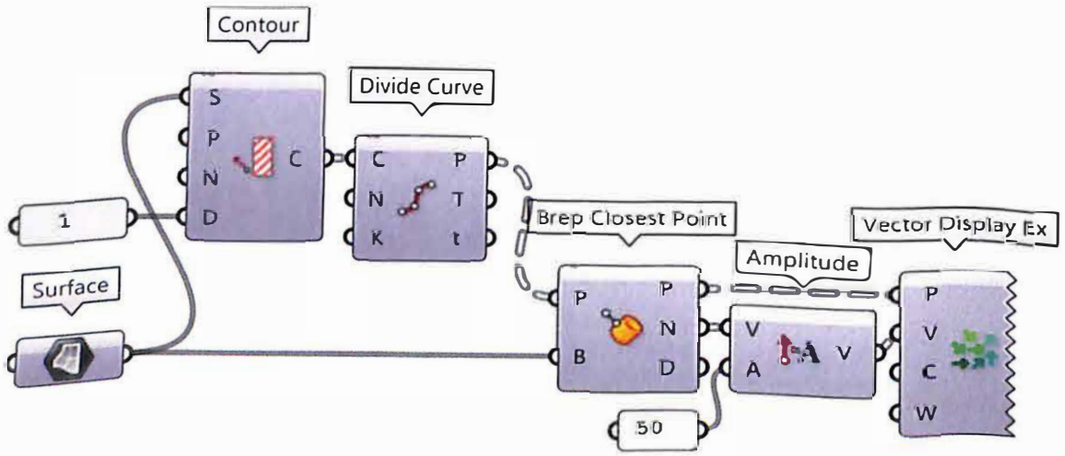
The explanation of these two patterns requires a bit more of detail but a skilled Grasshopper® user can make them easily.

Working with waves on a curved surface needs to get control over two tasks, the **normal vectors** for the motion of the points and to **fit the wave's pattern at the surface's seam.**

Normal vectors:

As usual, there are several tools and paths that allow us to get the normal vectors of a surface at certain points. In the last updates of Grasshopper® , a simple, powerful and useful tool has been added: 'Brep Closest Point'. Get the normal vectors just by feeding its inputs with a Brep (surface, extrusion or polysurface) and cartesian points (X,Y,Z).

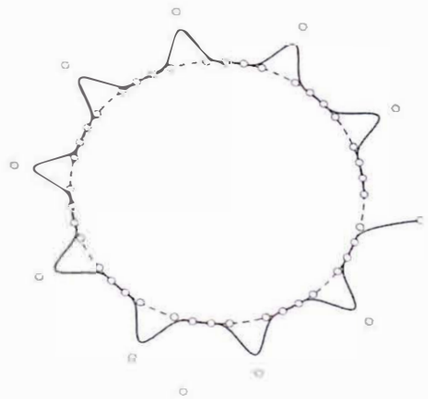


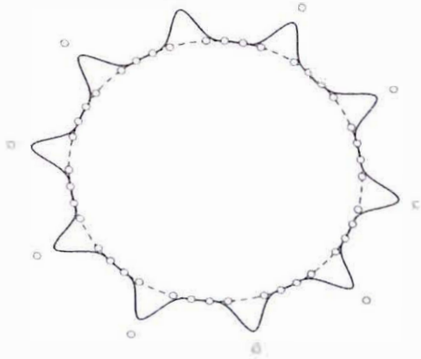


In this example 'Amplitude' controls the size of the vector. Use "Vector Display Ex" to show the vectors as the typical arrows.

The wave's pattern must fit at the seam:

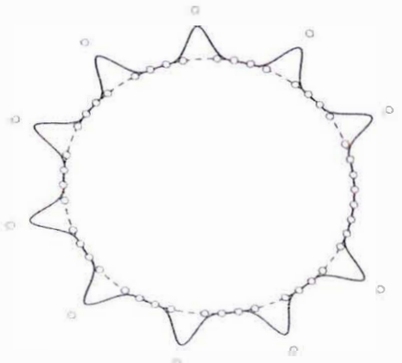
The seam is one of the most critical points for a 3d printed object. If it is not under control, it will most likely be displayed in an undesired or unexpected way. When drawing waves, the seam represents the first and the last point, so the continuity of the wave depends on that exact point.





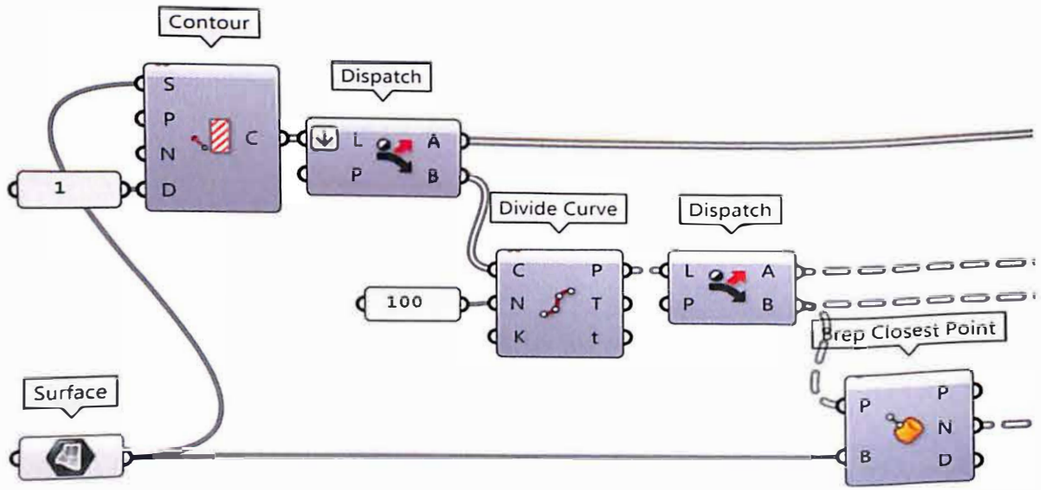
Pattern: 1 1 1 1 0

In the first drawing the curve is open. We must set the input 'Closed' to TRUE if it is a 'Polyline' component. If it is a 'Nurbs Curve', as in this example, then we would set the 'Periodic' input to TRUE.



The pattern for the design of the wave has to be proportional to the number of divisions of the curve. In this example the pattern is 1 1 1 1 0, so 5 booleans are necessary to create one wave. That also means that the number of divisions has to be proportional to 5. In the first two drawings it was 50, whereas in this one it is 54. That means that there are 4 points left off an unfinished wave.

Let's put all this together to have a base before going into attractors.



As a summary:

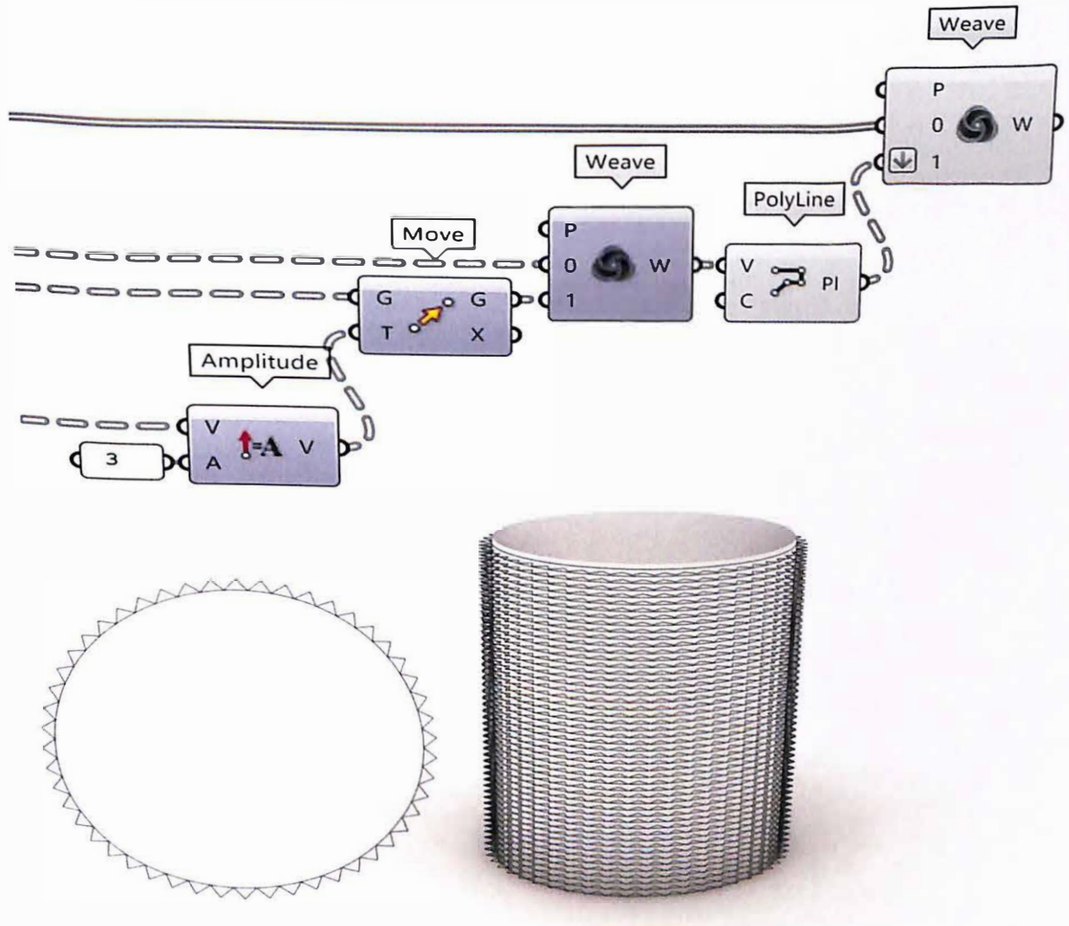
A surface is sectioned by 'Contour' at a certain layer height (1 mm.).

The sections are split in two lists due to 'Dispatch'. 'List A' remains as it is. Every curve at 'List B' is divided in 100 points. The odd ones (50 of them) go to 'List A' and the even ones (50 left) go to 'List B' as explained before. 'List A' remains as it is. The points at 'List B' are moved in the normal direction to the surface with vectors of 3 mm. length.

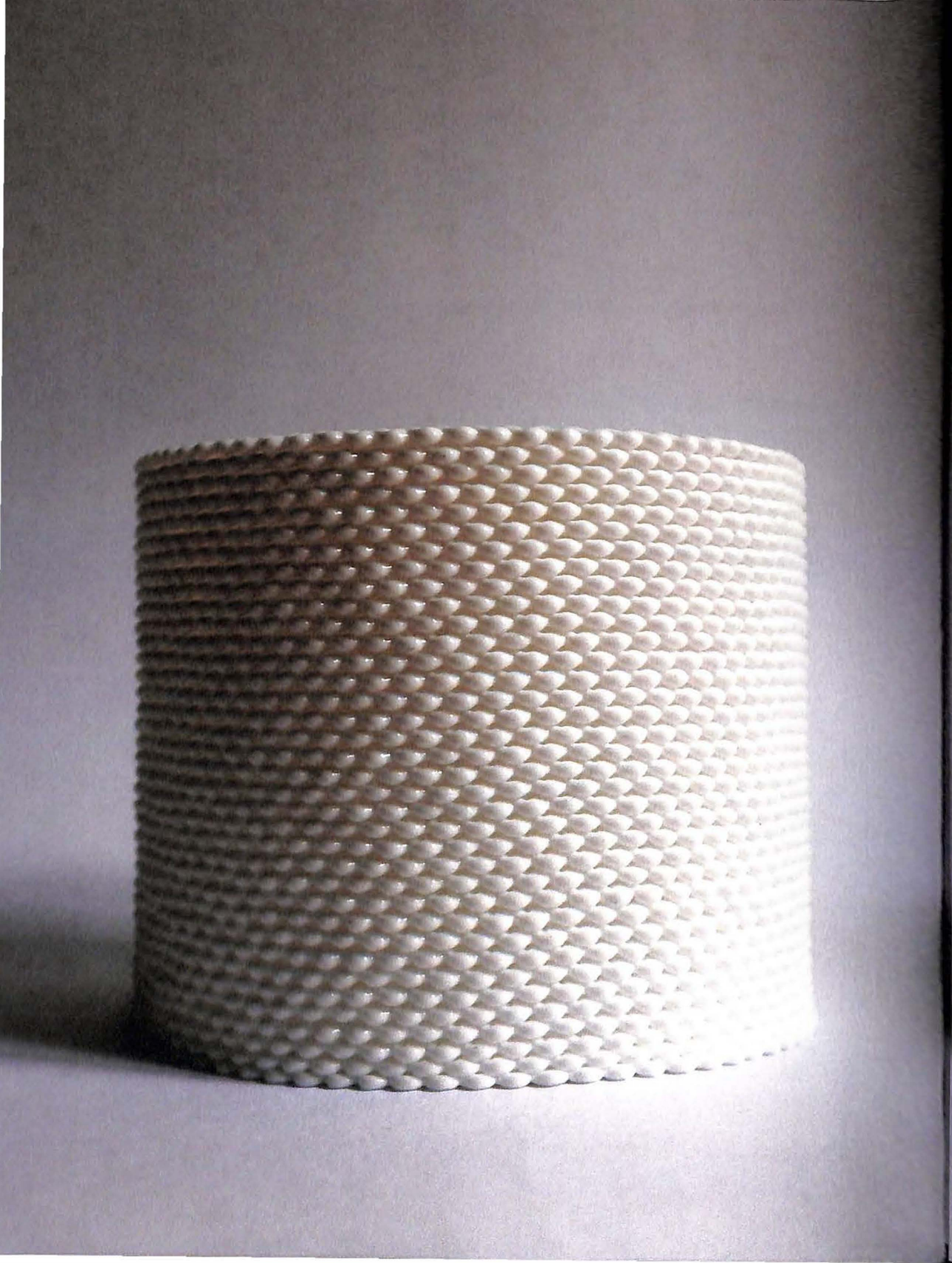
The first 'Weave' component reconstitutes the points list in the original structure. 'Polyline' displays the wave. This can also be used to check that everything has been edited as expected.

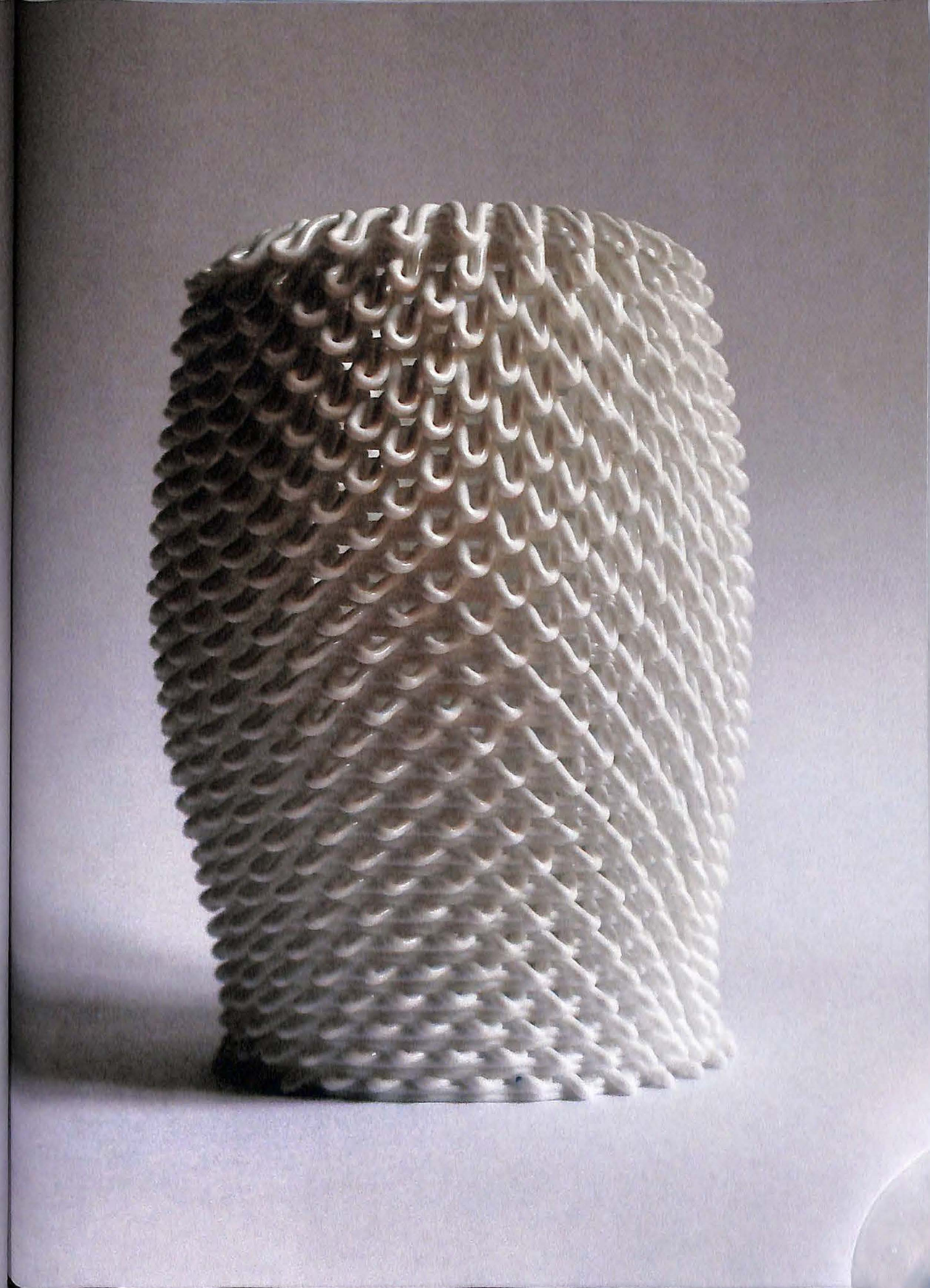
The second 'Weave' component reconstitutes the curves' list in the original structure: one section, one waving polyline.

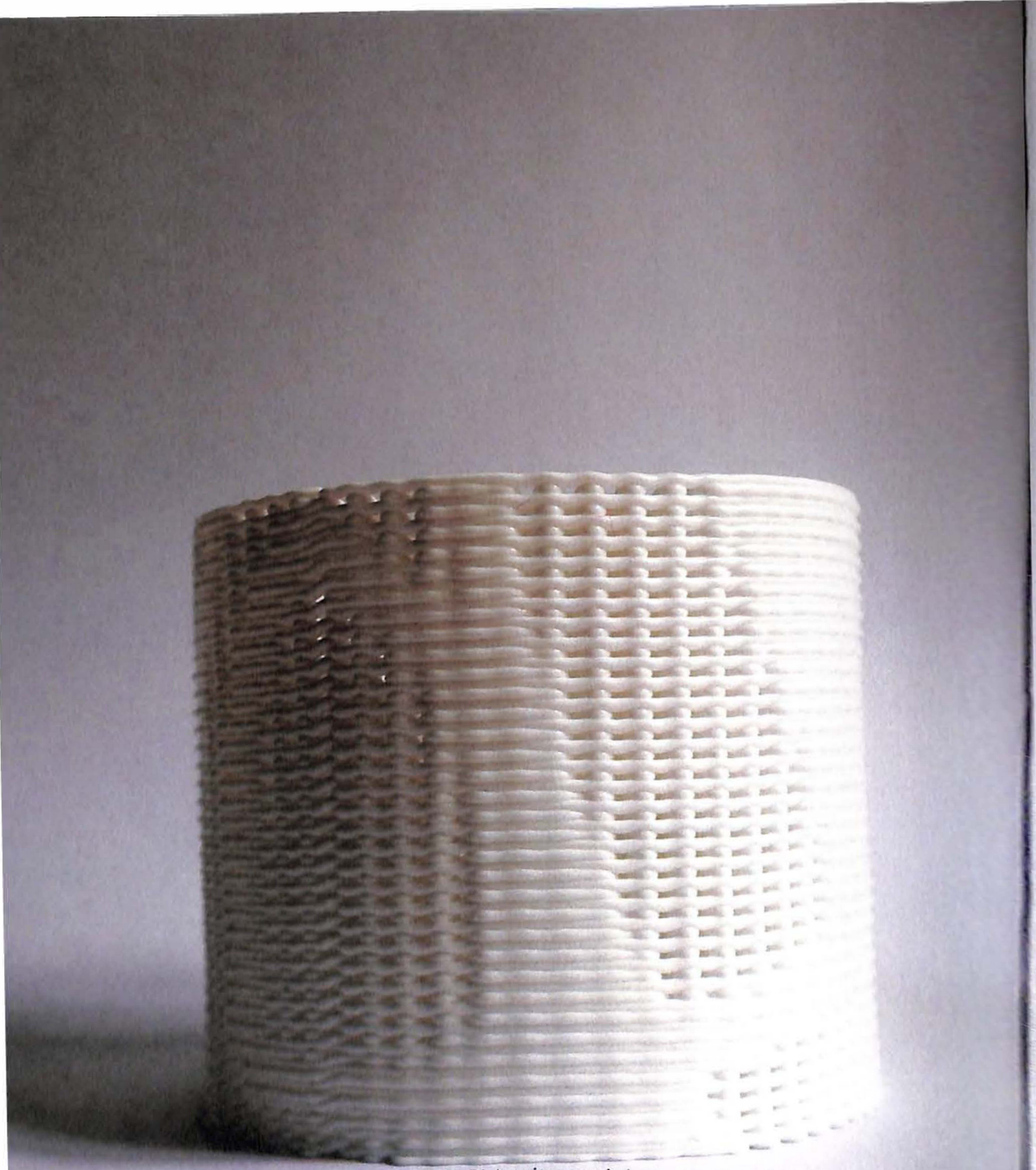
After this, use the strategy at chapter *Brep -> Curves -> Polyline -> Points -> X,Y,Z* to make one continuous polyline if desired.



Some examples 3D printed with white PLA using several waving patterns:







3D printed part in white PLA using a picture as attractor

Attractors

An attractor is an entity that modifies an object or a system.

The most common entities used as attractors in Grasshopper® are points, curves, vectors and images, but some others such as 3D objects, any kind of Surface, Brep or even just data from a spreadsheet could be used.

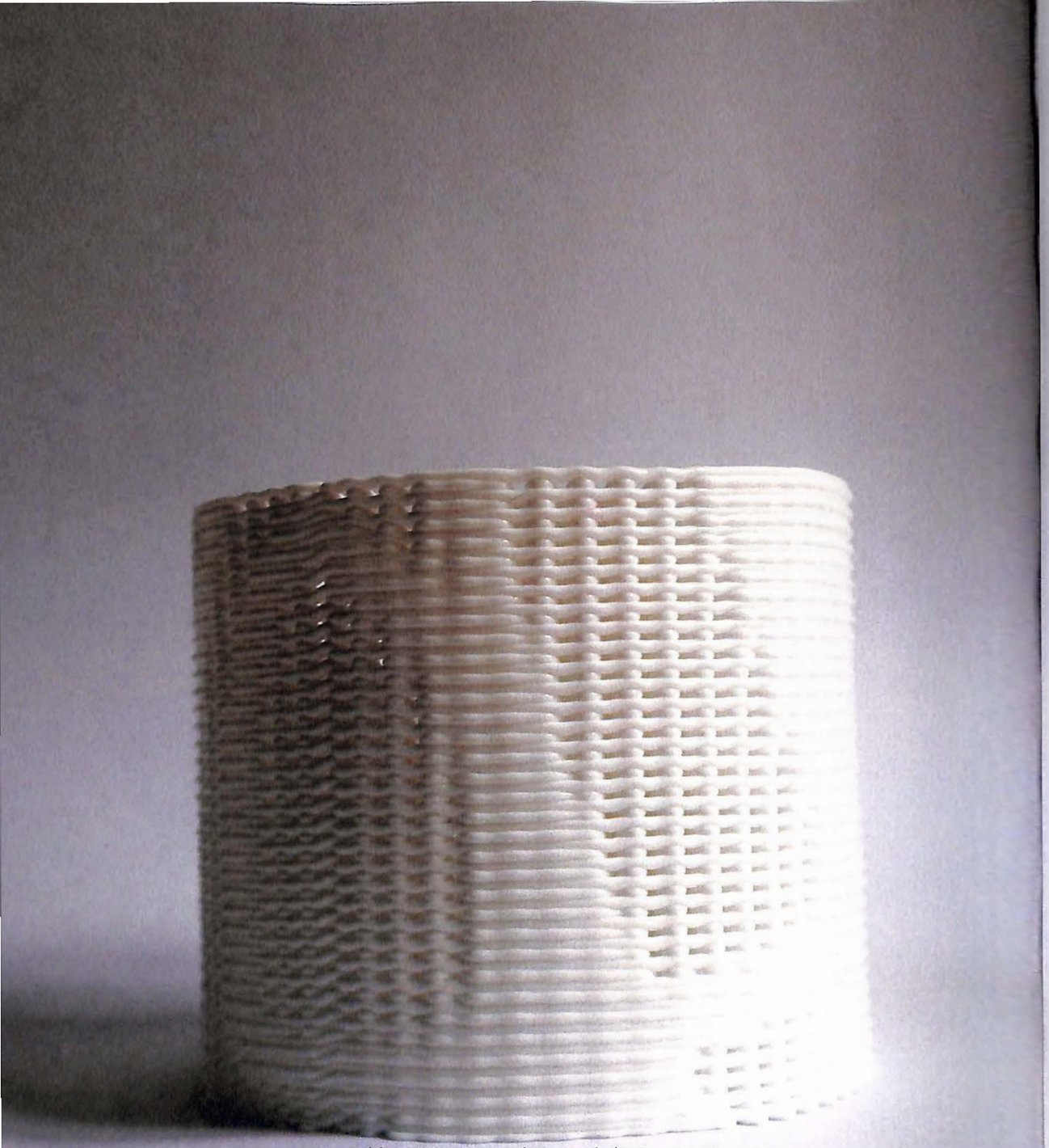
The name attractor does not necessarily mean that it must 'attract' objects. It could also 'repel' them by changing the values or the direction of the vectors.

Independently from the type of attractor used, it must be transformed into a set of numerical values. The most common values extracted from a point or a curve are their distances. From vectors, angles. From images, the amount of black and white present.

When working with waves, the attractors can easily modify the height of the peaks creating interesting results in a simple way.

As samples, we can test the results of different shapes as single vertical surfaces, cylinders and vases.

To display the effect of the attractors we will modify a simple weaving pattern as in the one at previous chapter (1-0) consisting of a contour and a wave. The wave is the first and simpler pattern too (1-0).



3D printed part in white PLA using a picture as attractor

Attractors

An attractor is an entity that modifies an object or a system.

The most common entities used as attractors in Grasshopper® are points, curves, vectors and images, but some others such as 3D objects, any kind of Surface, Brep or even just data from a spreadsheet could be used.

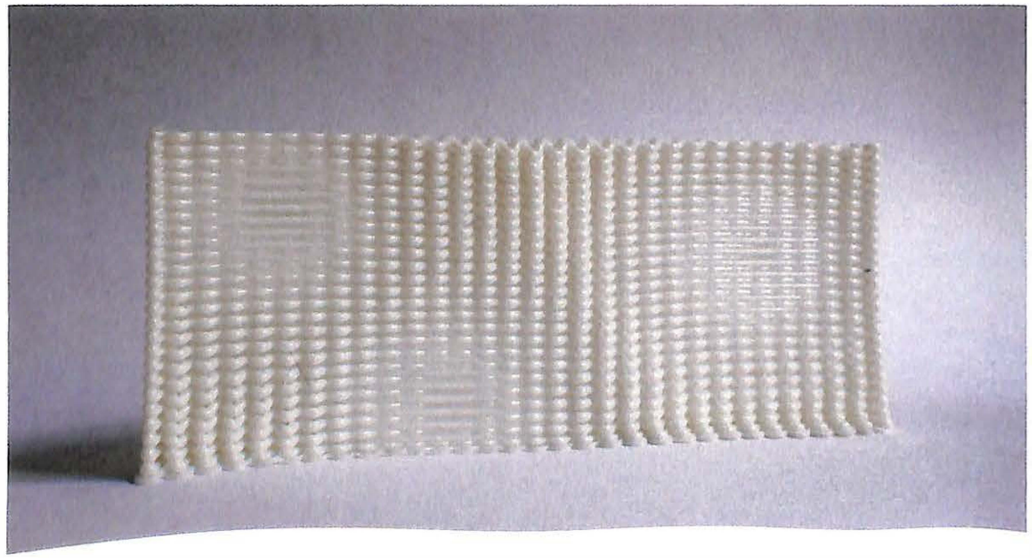
The name attractor does not necessarily mean that it must 'attract' objects. It could also 'repel' them by changing the values or the direction of the vectors.

Independently from the type of attractor used, it must be transformed into a set of numerical values. The most common values extracted from a point or a curve are their distances. From vectors, angles. From images, the amount of black and white present.

When working with waves, the attractors can easily modify the height of the peaks creating interesting results in a simple way.

As samples, we can test the results of different shapes as single vertical surfaces, cylinders and vases.

To display the effect of the attractors we will modify a simple weaving pattern as in the one at previous chapter (1-0) consisting of a contour and a wave. The wave is the first and simpler pattern too (1-0).



3D printed part in white PLA with three attractor points.

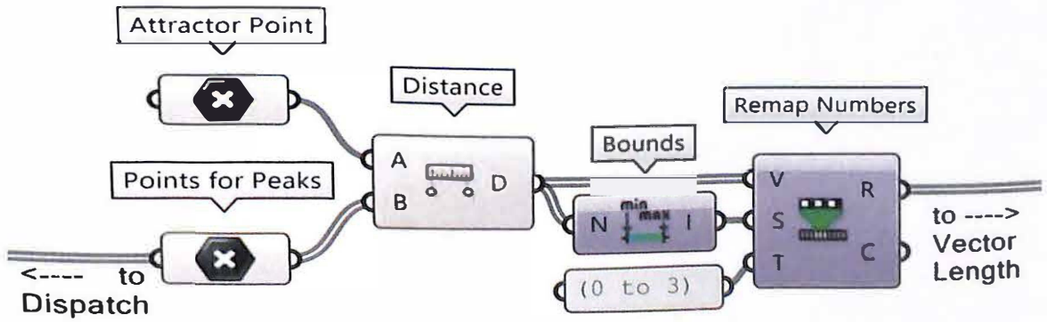
Point

When the attractor is a point.

It is important to highlight that the object being modified or altered by the attractor could be anything. In this example we will change the length of the vectors used for the creation of the waves. This means that we only need to 'work' on the points that make the peaks of the waves. There are many examples of attractors on the Internet with circles, meshes, and so on.

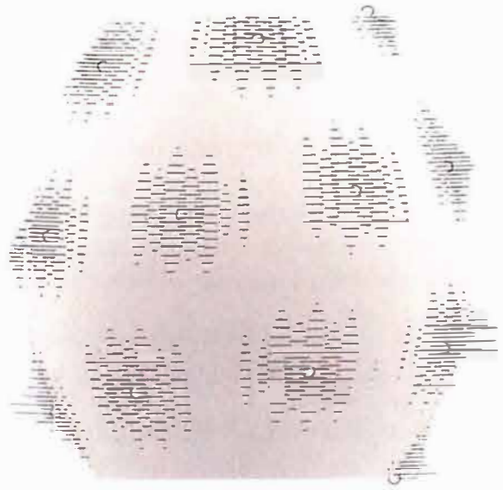
To modify them we will measure the distance from an 'attractor' point to the rest of points responsible for the waves' peaks.

Create a point that will be the 'attractor point'. It could be done in Grasshopper® or referenced from Rhinoceros®. Use 'Distance' and feed it with the attractor point and the peaks' points. The distances could be huge for our waves. To resize them, there is 'Division' but this is still a bit out of control. It is much better to use 'Remap Numbers'. 'Remap numbers' was explained at chapter *Flow (E)*, and in combination with 'Bounds' it remaps or recalculates the distances in this example into a new target domain (e.g. 0 to 3). Remember the units for numbers in Grasshopper® are the units of the Rhinoceros® file. If we are in millimetres, 0 to 3 stand for millimetres.



Notice that the tag 'Attractor Point' was manually modified. It is a simple 'Point' component.

When the attractor is not a single point but a bunch of points.

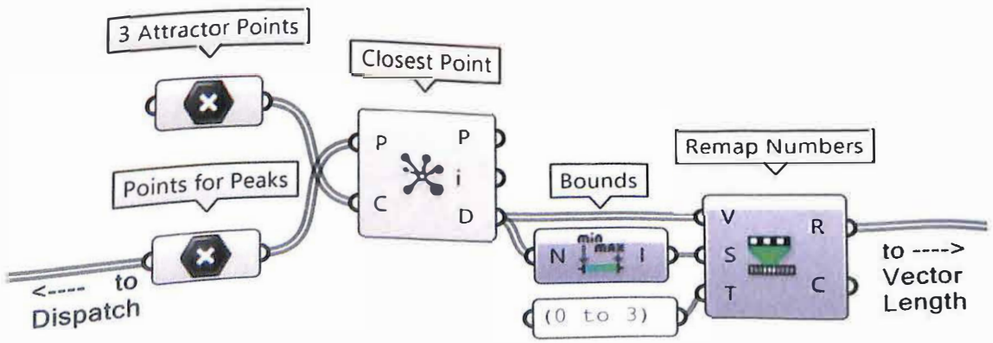


In this case adding several points to the 'Attractor Point' component will not work as expected, as we will be comparing two lists with more than one point in each list. Imagine we have 3 points as attractors and 100 points for the peaks. The first attractor will be compared with the first peak, the second with the second, and the third attractor with the ninety-eight peaks left. So those ninety-eight points will only perform according to the position of the third of the attractors. To make the bunch of peaks react to all of the three attractors we need to 'graft tree' the list of peaks, so that we get lists of 3 different distances – one to each of the attractors - and then use the mean of those three to obtain one single value. This could follow different paths and could be a bit tricky. However, Grasshopper® has a component to deal with several points at the same time: 'Closest Point'. This evaluates the numerical mean for the distances of several attractor points. It has two inputs:

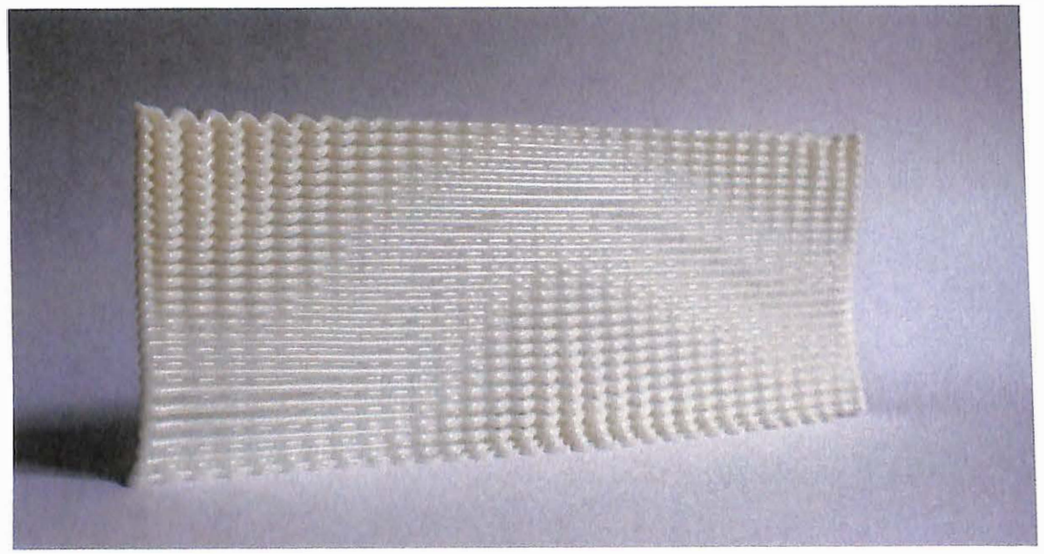
'Point (P)' (Points to search from) must be fed with the peaks' points.

'Cloud (C)' (Cloud of points to search) must be fed with our attractor points.

This component solves the problem of working with several attractor points as the pictures above and on the right.



3D printed part in porcelain using points as attractors

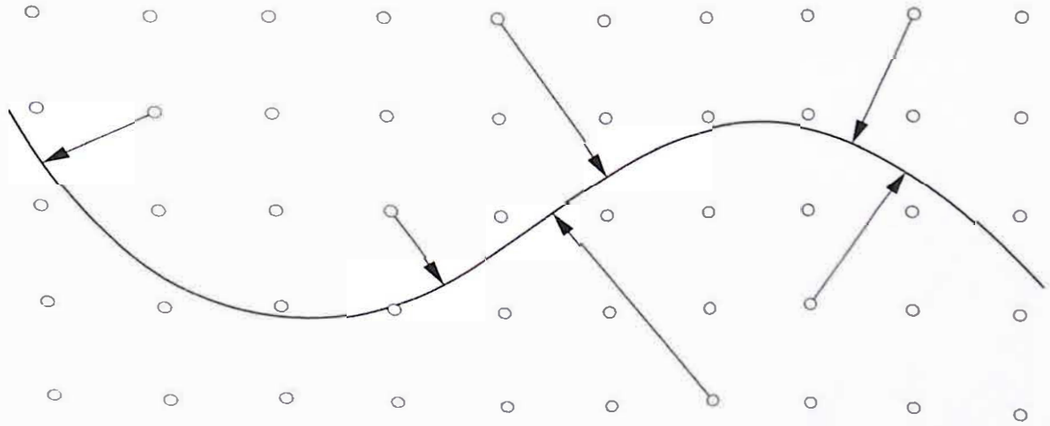


3D printed part in white PLA with one attractor curve.

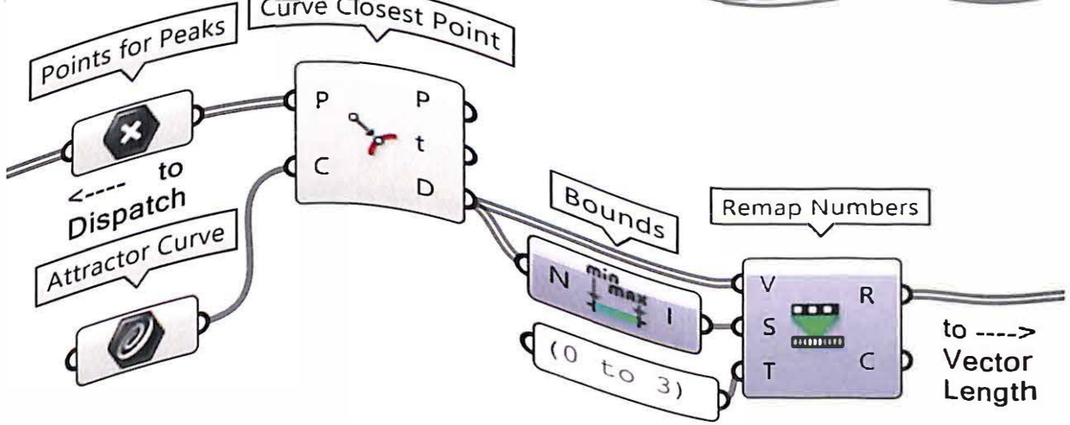
Curve

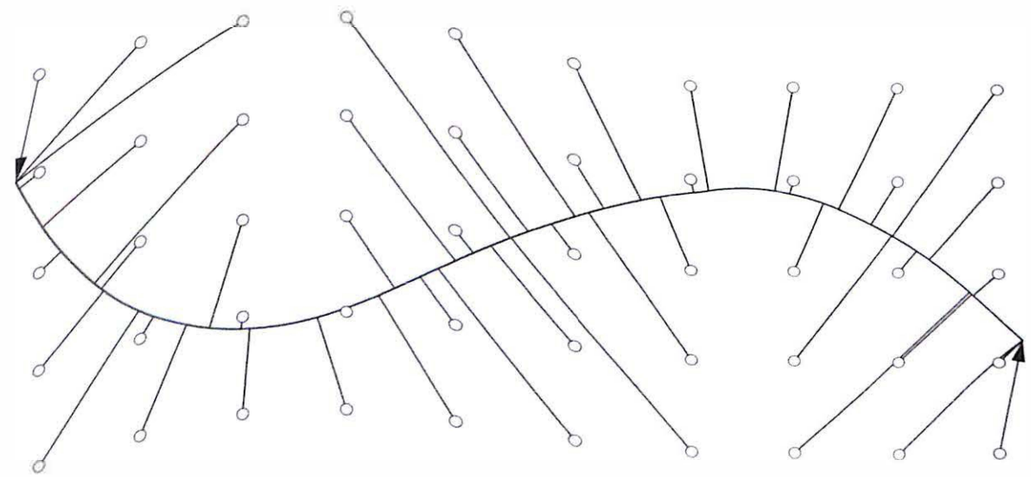
When the attractor is a Curve.

The main difference between the distance from a peak to an attractor point and a peak to a curve, is that for the former there is only one possible value, but for the latter there are infinite values. So, we have to identify just one distance from a point to a curve from the infinite possibilities. It could be the shortest, the largest, the mean... but it is very common to work with the shortest as it is usually also the perpendicular to the curve, as we can see in the figure:

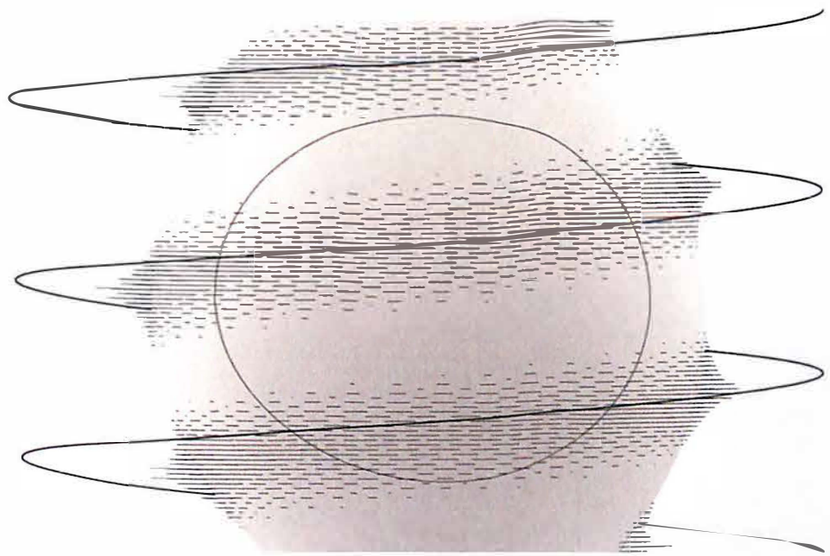


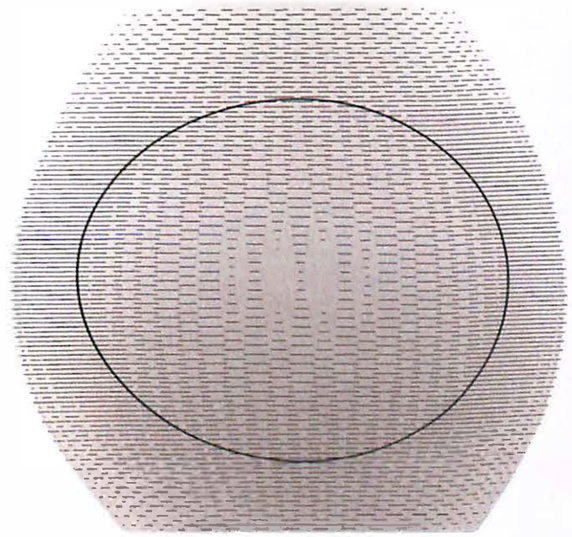
The 'problem' comes with the top-left and bottom-right points as they don't have a perpendicular direction to the curve to measure the shortest distance from. There is a component in Grasshopper® that solves this issue: it finds the closest point on the curve from another point. If it is perpendicular, it is clear that this is the shortest distance. If not, it will find the closest point. In both cases, those are the closest points to the curve, and so, the closest distances:





As in the previous example the distances must be remapped to suit our design.





Examples in the next pages. 3D printed parts in porcelain.