nTopology

Field-Driven Design

George Allen, nTop Fellow

1 Introduction

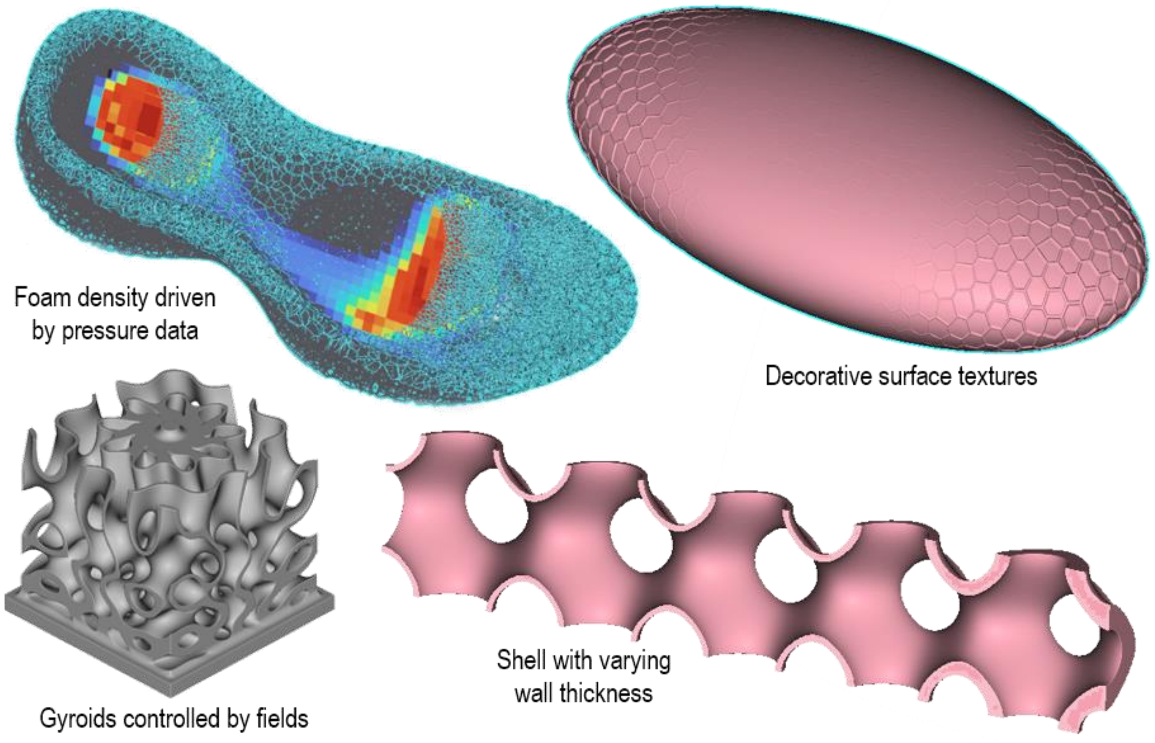
There are three basic technology pillars at the core of nTop Platform that make it different from most other engineering design software available today:

- **Fast unbreakable geometry**: A new approach to engineering geometry (implicit modeling) that delivers unmatched speed, scalability, and reliability.
- **Field-driven design**: A design methodology that lets you drive design variations using formulas, distances, test or simulation results, or other data.
- **Remixable workflows**: Configurable, automated, reusable, and shareable design processes that capture engineering knowledge.

This document covers the second of these pillars: field-driven design. We will discuss:

- The general concepts of fields and field-driven design (section 2)
- How to create and use fields in nTop Platform (section 3)
- Some example applications of field-driven design (section 4)

Simply put, fields give you a way to specify spatial variations of design features. This enables you to control complex geometry in ways that would otherwise be impractical. Several applications of field-driven design are described in more detail in section 4, but, as a preview, here a few examples of designs that are easy with fields and extremely difficult without them. As you can see, these all involve some sort of spatial variation of feature sizes or shapes:


Foam density driven by pressure data
Decorative surface textures
Gyroids controlled by fields
Shell with varying wall thickness

2 Field Concepts

This section discusses field definitions and concepts in general terms, independently of any particular software system. Later, in section 3, we'll see how these concepts are applied to enable field-driven design in nTop Platform specifically.

2.1 Definition: What is a Field?

A **field** is a rule that associates a value to each point of 3D space. Usually, these values are numbers (scalars), in which case we have a **scalar field**. If you're mathematically inclined, you'll immediately see that a scalar field is just a real-valued function defined on $\mathbb{R}^3$ (or some subset). So, at each point $P = (x, y, z)$ in 3D space, a scalar field F gives us a numerical value $F(x, y, z)$.

The values given by a field function are not always scalars. Sometimes they're vectors, instead, in which case we have a **vector field**. nTop Platform supports vector fields, but we won't concern ourselves with them very much in this document. So, for now, the term "field" by itself just means a scalar field.

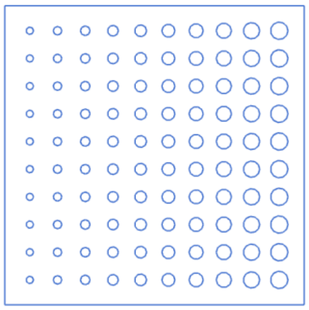
Weather reports provide some nice concrete examples of fields. For example, the temperature and humidity at various points in the atmosphere can both be regarded as scalar fields, and wind velocity is a vector field.

In much of this document, we'll be using 2D examples, for clarity. In the 2D scenario, a field F gives us a value $F(x, y)$ at each point (x, y) in a 2D plane. Two-dimensional fields are easier to explain and visualize than 3D ones, but the basic concepts are the same in either case.

This might all seem very theoretical and abstract, and entirely unrelated to the work that you do. But much of the power of fields lies in their generality, and you'll start to see their relevance and value within the next few pages.

2.2 A Very Simple Example

Imagine a square plate, with dimensions slightly larger than $1''$ by $1''$, into which we're going to cut a 10×10 array of holes. We want the diameters of the holes to gradually increase from left to right, starting at $0.03''$ and ending at $0.07''$.

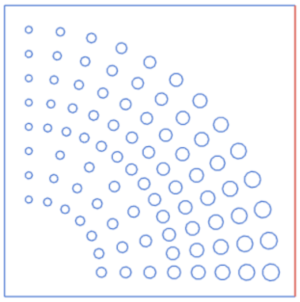


To achieve this, we can stipulate that the hole centered at the point (x, y) should have a diameter given by the formula

$$F(x, y) = 0.03 + 0.04x$$

Clearly $F(x, y) = 0.03$ when $x = 0$, and $F(x, y) = 0.07$ when $x = 1$, so this formula gives us what we want. The function F is a 2D field: it gives us the value $0.03 + 0.04x$ at any given point (x, y).

Note that none of the above depends on the holes being in a regular array. Here's an example where the holes are distributed in a less regular way, but we're still using the same field function F to control their diameters:



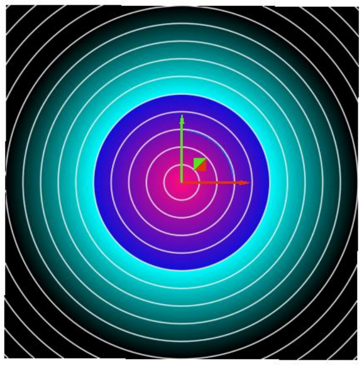
The field we're using here is defined by a simple formula, which makes the explanations easier. Later, in section 3, we'll see how these kinds of formulas can be used in nTop Platform, and we'll also see that there are other ways to define fields that are often more convenient.

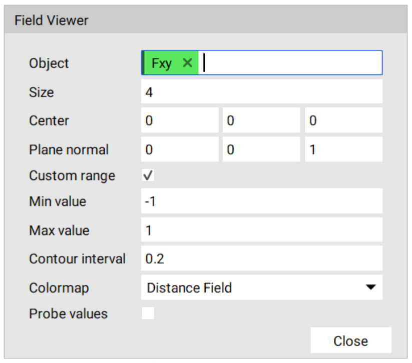
2.3 Visualizing Fields in nTop Platform

Though most of this section is supposed to be independent of nTop Platform, we're going to make an exception by discussing the nTop *Field Viewer*. As its name suggests, the Field Viewer helps you to visualize fields. As an example of the use of the viewer, let's look at the 2D field:

$$F(x, y) = \sqrt{x^2 + y^2} - 1$$

The term $\sqrt{x^2 + y^2}$ is just the distance to the origin, of course, so we would expect a radially symmetric pattern. When we examine this field in the nTop Field Viewer, we get:





Field Viewer

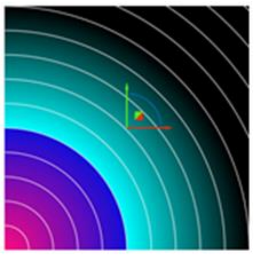
Object	Fxy		
Size	4		
Center	0	0	0
Plane normal	0	0	1
Custom range	✓		
Min value	-1		
Max value	1		
Contour interval	0.2		
Colormap	Distance Field		
Probe values			

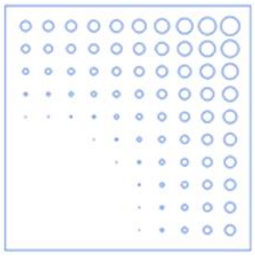
Cyan and black colors indicate positive values of the field, and blue and red indicate negative ones. Often what's most important is the hard border between cyan and blue, which shows where the field changes sign from negative (blue) to positive (cyan). We'll be using images from the Field Viewer to help with explanations in the rest of this document.

2.4 Fields are Gradients for Geometry

If you've ever used a drawing program like Illustrator or PowerPoint, then you're probably familiar with the concept of gradients, which you can use to specify variation of color within an object. In nTop Platform, fields give you precise control over spatial variations of shape, just as gradients allow you to control variation of color. So, you might find it helpful to think of fields as "gradients for geometry."

In the pictures below, the field we defined in the previous section is used to control hole diameters, and a radial gradient is used to control grey level, which should make the field-to-gradient analogy more clear:

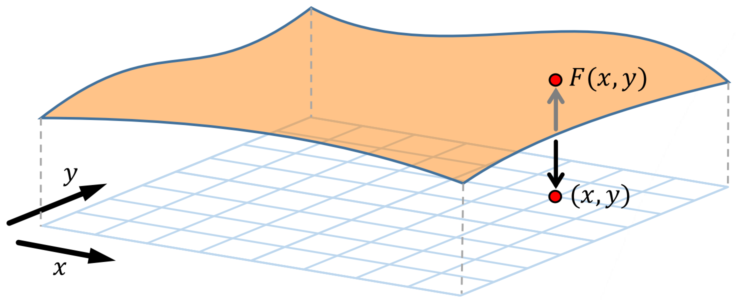

Field viewer


Geometry

Grey-scale gradient

2.5 Another Way to Visualize Fields

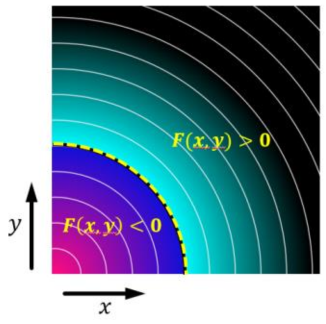
In addition to the "color gradient" analogy mentioned above, there's another way to visualize a 2D field. At any given point (x, y) in a horizontal plane, a 2D field F gives us a value $F(x, y)$ that we can interpret as an elevation above that plane. So, the field describes a landscape $z = F(x, y)$ with the usual slopes, hills, and valleys. The contour curves that you see in the Field Viewer are the contours that you'd see on a map of this landscape.

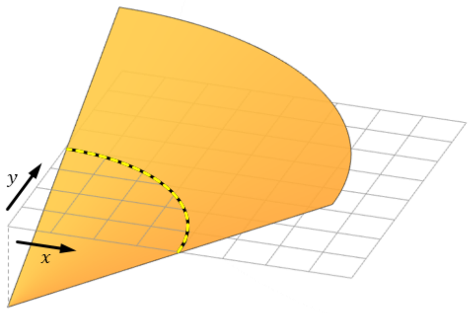


As a concrete example, let's look again at the field we defined in section 2.3 above:

$$F(x, y) = \sqrt{x^2 + y^2} - 1$$

Here are the Field Viewer and "landscape" visualizations of this field:

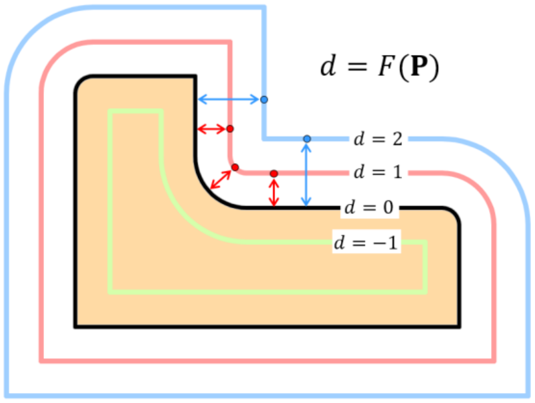




The landscape in this case is just a portion of a cone. In both views, you can see that the boundary between positive and negative field values is the circle $x^2 + y^2 = 1$ (the yellow dashed circle).

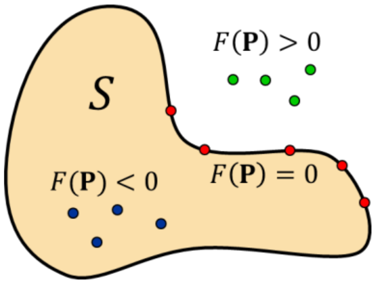
2.6 Fields, Bodies and Signed Distances

If we have a solid body, S, we can define a field F by using the **signed distance function (SDF)** of the body. For any given point $\mathbf{P}$, the field value $F(\mathbf{P})$ is defined to be the distance from the point $\mathbf{P}$ to the boundary of S, positive if $\mathbf{P}$ is outside$S$, and negative inside, as shown below.

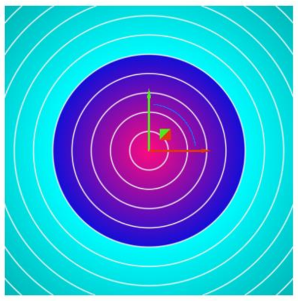


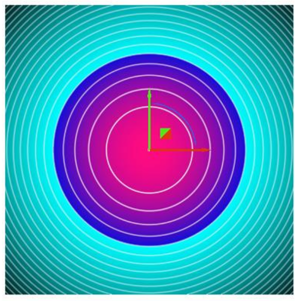
In fact, as we will see in section 3.2, using a signed distance function is one of the most common ways to define a field in nTop Platform.

Conversely, if we have a field F, then the set of points $\mathbf{P}$ where F is negative is an object called an implicit body, S. In mathematical jargon $S = \{\mathbf{P} \in \mathbb{R}^3 : F(\mathbf{P}) < 0\}$. Implicit bodies and implicit modeling are the foundation of nTop Platform, and are discussed at length in our modeling white paper, but the basic idea is shown in this picture:



Typically, there will be several different fields that produce the same solid body. For example, the two fields shown below are obviously different, but they both define the same spherical shape:

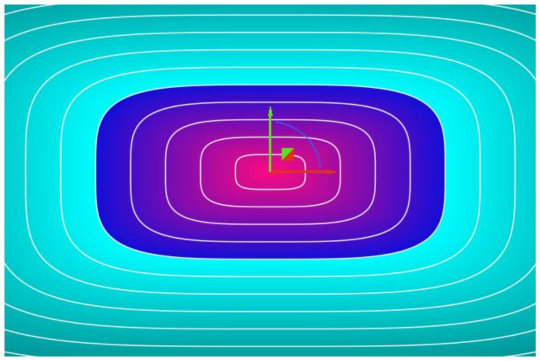

$\sqrt{x^2 + y^2 + z^2} - 1$


$x^2 + y^2 + z^2 - 1$

When using a field to represent an implicit solid in nTop, it's always best to use one that's an SDF, or else you might get some unexpected results in subsequent modeling operations like offsetting and filleting. You can identify a field that's an SDF because its contour lines in the Field Viewer are parallel and equally spaced. So, of the two shown above, the field on the left is a better choice for representing a spherical shape in nTop,

But, in many cases, we 're using a field to control variation, not to represent a solid object, and then we are free to use any field we want — we don't have to restrict ourselves to SDFs.

Another example is the field $F(x, y) = x^4 + 16y^4 - 1$, which looks like this when displayed in the Field Viewer:



As you can see, the contour lines are neither parallel nor equally spaced, so this field is certainly not an SDF. So it might not be prudent to define a solid body from this field, but it's perfectly fine to use it to control design variations.

2.7 Fields Give You Control Over Complexity

The field-driven design approach is valuable because it lets you build just the right degree of flexibility into your designs. You can use it to make highly complex designs, with many parameters, or simple easily-controlled ones with few parameters. To understand what this means, let's go back to our first simple example of a plate with an array of 100 holes. The simplest design would be one in which the holes all have the same diameter, say d_0. We could use a constant field $F(x, y) = d_0$ to achieve that, but using a fixed number instead of a field would work just as well. This design is pretty boring and inflexible, but at least it's easy to control — we can modify the design merely by adjusting the value of a single number, d_0.

At the other extreme, we might construct a model in which all 100 holes are independent, and each has its own individual diameter. This gives an extremely flexible design, but it's difficult to control because modifying 100 individual diameters takes a lot of time and effort.

In the original example, we used the field $F(x, y) = 0.03 + 0.04x$ to control the diameters of holes. For reasons that will become clear in a moment, let's write this in the form $F(x, y) = 0.03(1 - x) + 0.07x$ instead. As we noted before, $F(x, y) = 0.03$ when $x = 0$, and $F(x, y) = 0.07$ when $x = 1$. But there's nothing magic about the numbers 0.03 and 0.07, and clearly we could define a somewhat more general field $F(x, y) = d_0(1 - x) + d_1 x$, which would give us $F(x, y) = d_0$ when $x = 0$, and $F(x, y) = d_1$ when $x = 1$. By doing this, we have introduced two parameters, d_0 and d_1, that control the design. So, by using this field, we have achieved a compromise: our design is more flexible than the one with a single parameter, but much easier to control than the one with 100 parameters.

Let's take this a little further: suppose we control the hole diameters using the field

$$F(x, y) = (1 - x)(1 - y)d_{00} + x(1 - y)d_{10} + (1 - x)yd_{01} + xyd_{11}$$

It's easy to confirm that

$$F(0,0) = d_{00} \quad ; \quad F(0,1) = d_{01} \quad ; \quad F(1,0) = d_{10} \quad ; \quad F(1,1) = d_{11}$$

So, the four numbers $d_{00}, d_{01}, d_{10}, d_{11}$ control the hole diameters at the four corners of the plate, with simple bilinear interpolation in the interior. Now we have a design with four parameters — more flexible than two, but more easily controlled than 100. In fact, by suitable choices of field functions, we can introduce as many parameters as we like (well, up to 100, anyway). So, using fields lets us make an intelligent trade-off between too few parameters and too many. We can get just the flexibility we want, and no more.

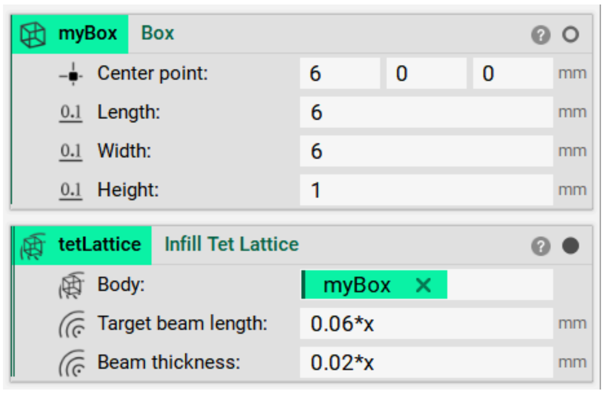
The following table summarizes the preceding discussion:

Field Function	**Params**	**Flexibility/Simplicity**
$F(x,y) = d_0$ (constant)	1	Simple, but inflexible.
$F(x,y) = d_0(1-x) + d_1x$	2	Good compromise
$F(x,y) = (1-x)(1-y)d_{00} + x(1-y)d_{10} + (1-x)yd_{01} + xyd_{11}$	4	Good compromise
None (individual diameters)	100	Too complex

Even this very simple example illustrates an interesting point: fields often correspond to design intentions that are simple to express in words, but surprisingly difficult to realize in many CAD systems. It's very easy to say "diameters increase from d_0 at the left to d_1 at the right", but most CAD systems don't give you an easy way to construct a hole array with this property. In the worst case, you might have to achieve this variation by defining all the hole diameters one at a time. In a 10 × 10 array, this might be barely tolerable, but a 50 × 50 array would keep you busy for quite a while. Fields let you introduce the parameterizations you need to control complex designs conveniently.

3 Fields in nTop Platform

In nTop Platform, fields are used extensively to control spatial variations of design variables. In fact, whenever you see the symbol to the left of a parameter in an nTop block, it means you can input a scalar field, rather than a fixed number. So, in the image below, the length, width and height of myBox are fixed numbers, but the length and thickness of the beams of tetLattice are being controlled by simple fields.

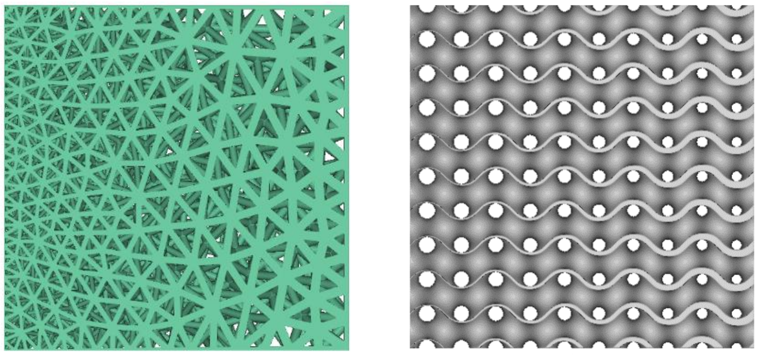

myBox Box

Center point:	6	0	0	mm
Length:	6			mm
Width:	6			mm
Height:	1			mm

tetLattice Infill Tet Lattice

Body:	myBox	
Target beam length:	0.06*x	mm
Beam thickness:	0.02*x	mm

The image on the left below shows the tet lattice defined by the values above, and the one on the right shows a thin-walled gyroid lattice whose wall thickness is controlled by a similar field:



In both lattices, thicknesses gradually increase as we move from left to right, as we would expect from the fields we used to control them.

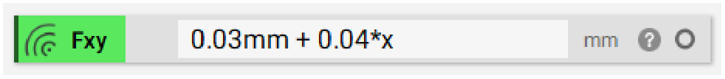
A wide range of parameters in nTop Platform can be controlled using fields, including:

- Beam thickness in lattices
- Fillet radii in lattices
- Wall thicknesses in offsetting and shelling operations
- Radii of fillets created during boolean operations
- Material properties like density, Young's modulus, and thermal conductivity
- Scales used to warp lattice cells

In the examples so far, we have defined our fields using simple formulas, to emphasize the generality of the concept and make things easy to explain. However, other methods of defining fields are available that are often easier to use, as described in the following sections.

3.1 Using Formulas

In section 2, we presented several examples of field functions defined by simple algebraic formulas. You can define fields this way in nTop Platform, but it's not a very common approach. Also, if you want to type formulas in nTop, the syntax needs to be somewhat different from our earlier examples. We can use our first simple example to illustrate. In section 2.2 we defined a field using the formula $F(x, y) = 0.03 + 0.04x$, but in nTop Platform, you'd actually have to type something like **`0.03mm + 0.04*x`** to conform to nTop's rules about units and dimensionality

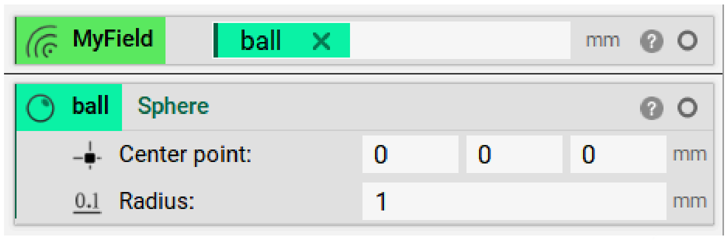


Fxy 0.03mm + 0.04*x mm

For more information, please see this video.

3.2 Using Distance Functions

In nTop Platform, we have the option of defining a field as the signed distance function of an implicit body, as described in section 2.6. So, when you define a scalar field, the input you provide can either be a formula that you type, or an existing implicit body:



MyField ball mm

ball Sphere

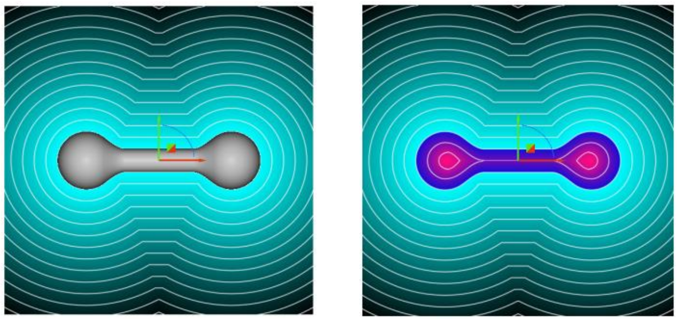
Center point: 0 0 0 mm

Radius: 1 mm

You can get exactly the same field either by using the implicit body directly, or by writing out the formula for its signed distance function. So, for example, the field defined by the unit sphere (with radius 1 and center at the origin) is exactly the same as the field defined by the formula $F(x, y, z) = \sqrt{x^2 + y^2 + z^2} - 1$. The following table gives a few examples:

Geometric Object	**Field (Signed Distance Function)**
Unit sphere. Radius = 1, center = (0,0,0)	$F(x, y, z) = \sqrt{x^2 + y^2 + z^2} - 1$
Infinite cylinder, along z-axis, centered at (0,0,0), radius = 1.	$F(x, y, z) = \sqrt{x^2 + y^2} - 1$
The xy-plane, i.e. the plane $z = 0$	$F(x, y, z) \ = z$

It's often very difficult to write down a formula for the signed distance function of an object. In these cases, the geometry-based approach is easier. For example, here is the field defined by a dumb-bell shaped object whose signed distance function would be difficult to construct:

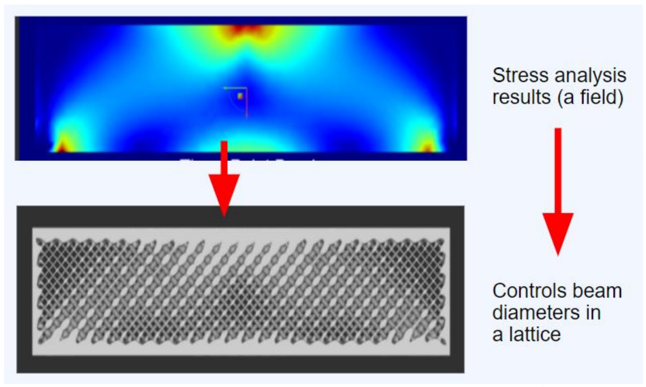


On the other hand, as we saw in the previous section there are useful fields that are not signed distance functions, so you sometimes have to use other methods.

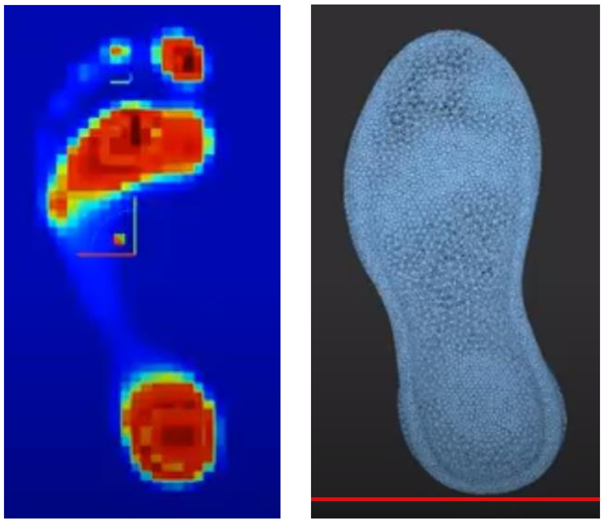
3.3 Interpolating Point Data

Sometimes we only know the value of a field at certain specific points in space, and we need to interpolate to get values at other points. nTop Platform provides functions for performing this interpolation. You begin by creating an nTop *Point Map* from the known data points. This is just a list of points along with another list of corresponding values. Then you use the *Field from Point Map* block to interpolate the values in the *Point Map* and obtain a scalar field. This capability is very significant because it allows you to drive your designs using tables of data from outside nTop Platform. First, you create a CSV file of (x, y, z, s) values, where s is the field value at the point(x, y, z). Then you use this CSV file to create a *Point Map*, and interpolate as outlined above.

A common example is a CSV dataset that is the result of a simulation, or data from physical testing, or any other source. You use the CSV data to create a field, as outlined above, and then use this field to control geometry, in the usual way. For example, if you have results from a stress analysis, you might use these to control the diameter of beams in a lattice, or the thickness of walls in a shell: in areas of higher stress, you'd specify larger lattice beams or thicker walls.

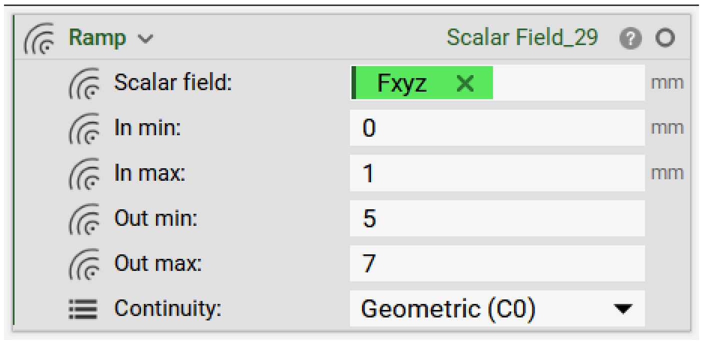

Stress analysis results (a field)
Controls beam diameters in a lattice

Similarly, the example below shows how you can use pressure measurements to control the density of foam in the insole of an athletic shoe:



3.4 Using the Ramp Block

The nTop *Ramp* block gives you a way to rescale existing fields to create new ones.


Ramp — Scalar Field_29
Scalar field: Fxyz (mm)
In min: 0 (mm)
In max: 1 (mm)
Out min: 5
Out max: 7
Continuity: Geometric (C0)

Let's say we have an existing field $F(x, y, z)$, and we enter values into the *Ramp* definition as follows:

$$\begin{array}{lll} \text{In min } = a_0 & ; & \text{Out min } = b_0 \\ \text{In max } = a_1 & ; & \text{Out max } = b_1 \end{array}$$

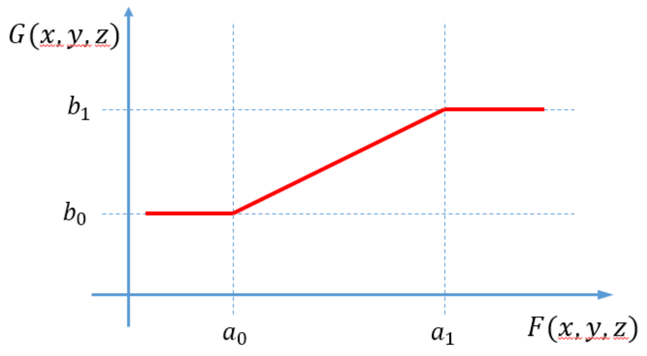
Then the result is a new field G where

$$G(x, y, z) \;=\; b_0 \;\text{ if }\; F(x, y, z) < a_0$$

$$G(x, y, z) \;=\; b_1 \;\text{ if }\; F(x, y, z) > a_1$$

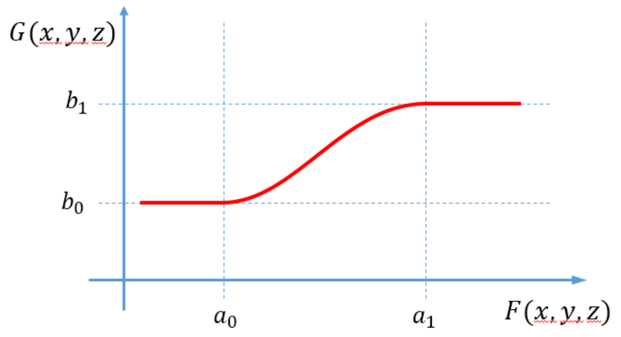
$$G(x, y, z) = \frac{a_1 - F(x, y, z)}{a_1 - a_0} b_0 + \frac{F(x, y, z) - a_0}{a_1 - a_0} b_0 \quad \text{if} \quad a_0 \; < \; F(x, y, z) \; < \; a_1$$

If you like pictures better than formulas, you can understand the important properties of the ramp function from the following graph:

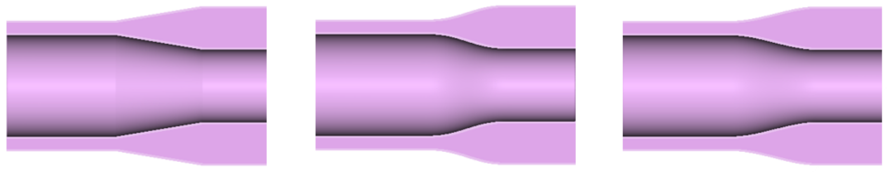


The net effect is that the ramp linearly rescales the values of the original field F, giving us a new field G that varies between b_0 and b_1 as F varies between a_0 and a_1.

Since we chose the *Geometric (C0)* option for continuity in the *Ramp* block, the corresponding graph is a straight line. If we choose *Tangent (C1)*, instead, then we get a smoother curve with a more complicated equation that we won't worry about here:



The following picture shows an example of how we can use the three different continuity options to control the wall thickness in a pipe:


Geometric (C0) Tangent (C1) Curvature (C2)

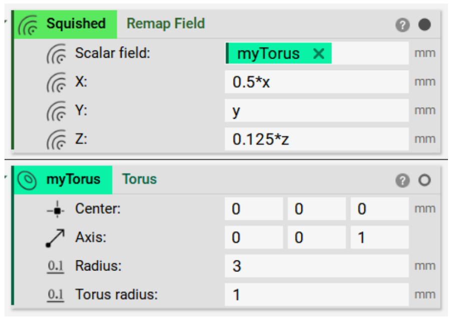
The *Ramp* block is very widely used in field-driven design. Typically, the field obtained from a distance function or interpolation is not directly usable, so you'll first have to rescale it with a *Ramp* block to suit your needs.

3.5 Using the Remap Block

If we have four fields F, F_x, F_y, F_z, then we can use an nTop *Remap* block to define a new field

$$G(x, y, z) = F\left(F_x(x, y, z), F_y(x, y, z), F_z(x, y, z)\right)$$

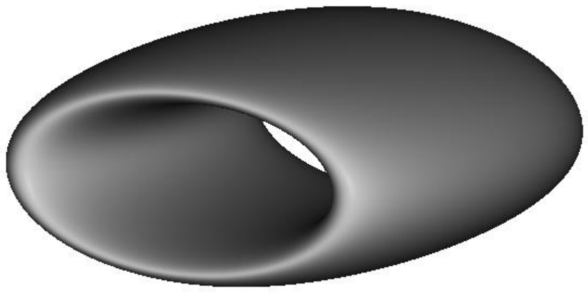
If we set $F_x(x, y, z) = ax$, $F_y(x, y, z) = by$, and $F_z(x, y, z) = cz$, we get the field $G(x, y, z) = F(ax, by, cz)$, which represents a non-uniform scaling of the original field F. For example, the field $G(x, y, z) = F(x, y, z/2)$ simply stretches the field F by a factor of 2 in the z-direction. Non-uniform scaling provides an easy way to make ellipse-like shapes of various kinds. Here's what you get if you scale a torus in this way:


Squished Remap Field

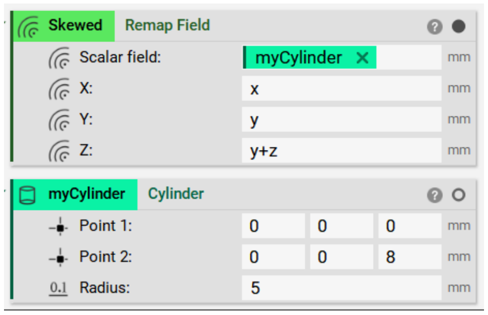
Scalar field:	myTorus	mm
X:	0.5*x	mm
Y:	y	mm
Z:	0.125*z	mm

myTorus Torus

Center:	0	0	0	mm
Axis:	0	0	1	
Radius:	3			mm
Torus radius:	1			mm



Similarly, the field $G(x, y, z) = F(x, y, y + z)$ skews the field F by 45 degrees. Here's what you get if you apply this skewing to a cylindrical shape:



Skewed Remap Field

Scalar field: myCylinder mm

X: x mm

Y: y mm

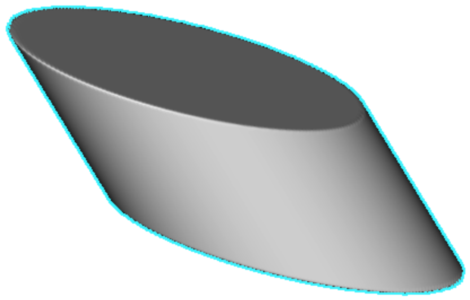
Z: y+z mm

myCylinder Cylinder

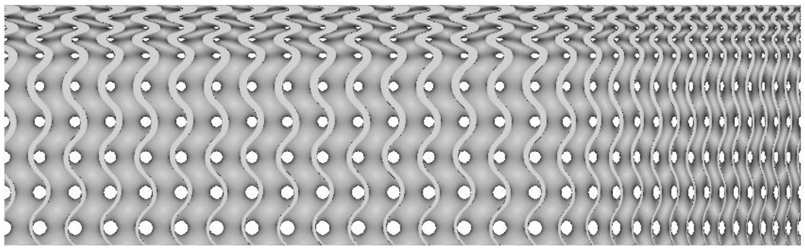
Point 1: 0 0 0 mm

Point 2: 0 0 8 mm

Radius: 5 mm



Finally, the following design shows the use of *Remap* to vary the cell sizes in two directions in a gyroid lattice. For a full description of this example, please see this video.

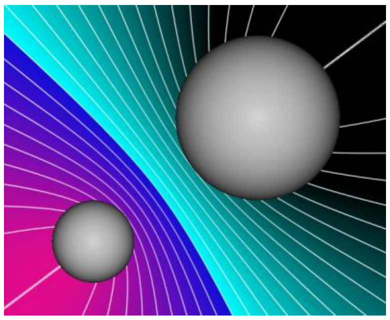


Some more advanced uses of the *Remap* block are described in this video.

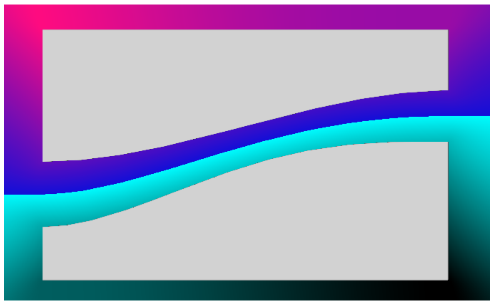
3.6 Using Field Arithmetic

The values of scalar fields are just numbers, so you can combine them using the usual arithmetic operations and functions. So, if F and G are two fields, we can define new fields like $F + G$, $F - G$, $\cos(F)$, $F^2 + G^2$ and so on.

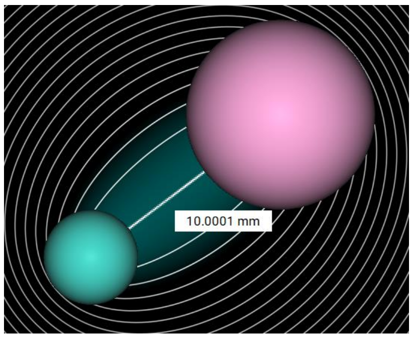
The field $F - G$ is especially interesting. If F and G are the signed distance fields of two objects, then $F(x, y, z) - G(x, y, z) = 0$ means that $F(x, y, z) = G(x, y, z)$, so the point (x, y, z) is equidistant from the two objects. In the picture below, the zero contour between the blue and cyan areas gives us the points that are equidistant from the two spheres.



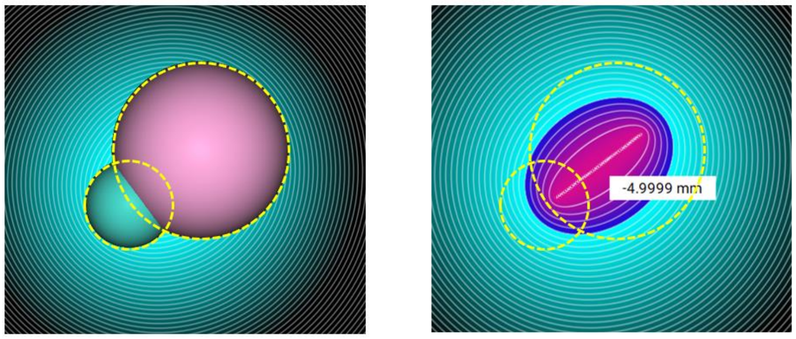
There are many situations where a “mid-surface” like this is useful. For example, you might use it to shape a dividing wall that runs down the middle of a channel, or to simplify a thin-walled design prior to simulation.



The field $F + G$ is also very useful, because it measures clearance or penetration depth. If the corresponding solid bodies are disjoint, then $F(x, y, z) + G(x, y, z)$ is the sum of the distances from the point (x, y, z) to the two bodies. So, the minimum value of this field is the distance between the two bodies. This minimum occurs everywhere along the shortest straight line connecting the two bodies:



If the two bodies are intersecting, $F + G$ gives the depth of penetration:



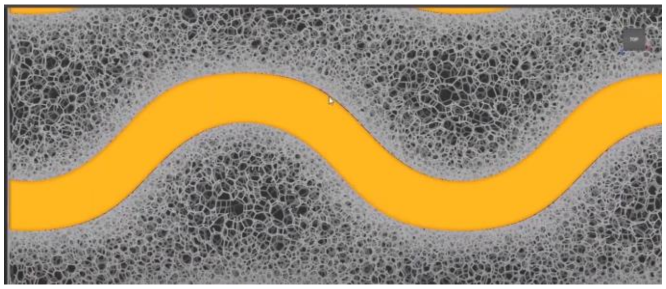
Some creative uses of field arithmetic are described in this video and this one.

4 Applications of Field-Driven Design

As we've explained in the earlier sections, fields are a very general concept, and they have many uses. This section provides a few examples of how you can use the principles of field-driven design in practice.

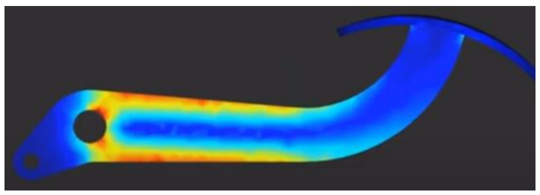
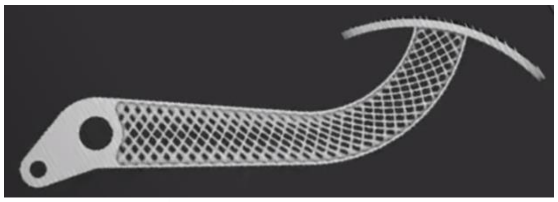
4.1 Field-Driven Foam Density

In the following example, a lattice has been used to create a cushioning foam. We defined a field using distance from the wavy curve, and rescaled it with a Ramp block. The density of the foam is controlled by this field, so we get higher density (i.e. smaller cells) near the wavy curve, and areas of lower density further away from it.



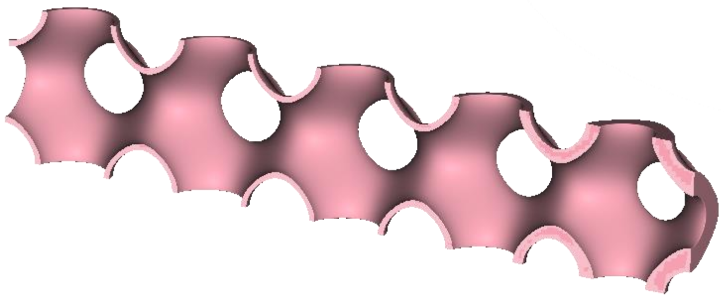
4.2 Lattice Beam Diameters Driven by Simulation Results

Here, we have performed a stress analysis on a brake pedal design. This could be done either inside nTop Platform or in an external analysis program. We created a field from the Von Mises stress results, and used this field to control the diameters of beams in a lattice, placing thicker beams in regions of higher stress.

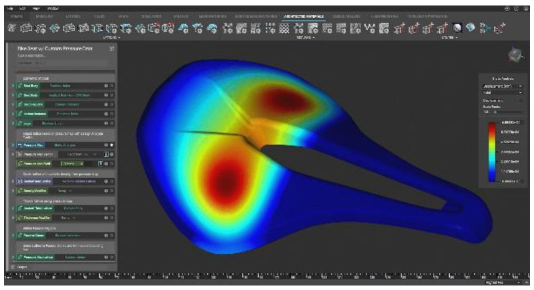
4.3 Varying Wall Thickness in a Shell

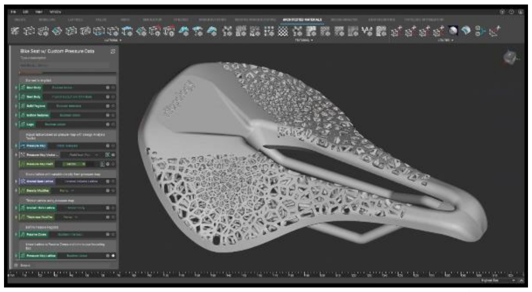
You can use a field to vary the thickness of a thin-walled shell. The field might be constructed using results from a structural or thermal analysis, or defined some other way. The shell with varying wall thickness is likely to be much lighter than one with constant thickness, of course.



4.4 Lattice Beams Driven by Test Data

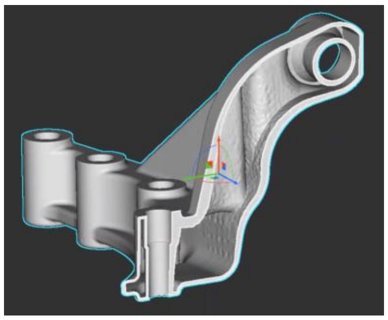
The pressure was measured at various points on a bicycle saddle. The measurements were imported into nTop Platform and used to create a field that controls the density of a lattice. The lattice cells are smaller in areas of higher pressure.

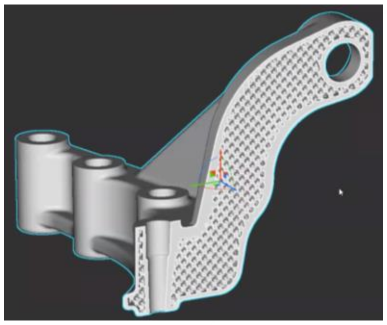




4.5 Variations Driven by Stress Analysis Results

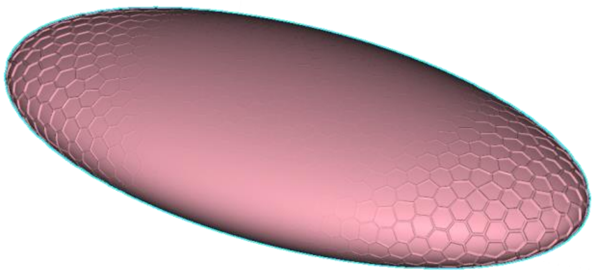
In this design, we have reduced the weight of this part by using stress analysis results to modulate both the outer wall thickness and the lattice beam diameters. For the full story, please see this video.





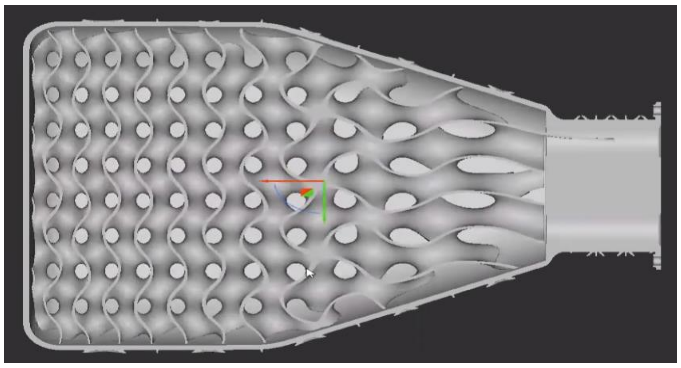
4.6 Gradient Surface Textures

In the designs shown below, lattices were used to create surface textures. You can use a field to control the thickness of the beams in a lattice. Thicker beams create a bolder texture, while very thin ones cause the texture to fade away. Textures like these are often applied to consumer products to improve their appearance or prevent them slipping out of our hands. This video explains further.



4.7 A Field-Driven Pressure Vessel Design

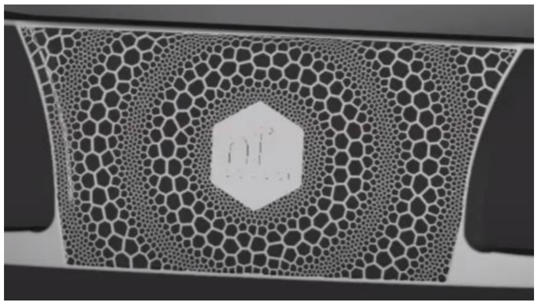
Here, we are using two different fields to vary the wall thickness and cell size in a gyroid-based pressure vessel. A cross-section view is shown below. For details, see this video.



4.8 Field-Driven Perforation Patterns

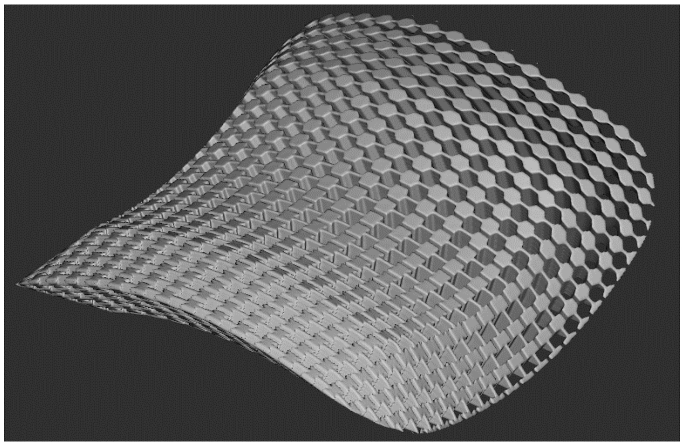
Fields can be used to control any repeating design feature. Here, fields were used to control the perforation pattern on a cell-phone case (left) and a grill (right). For more about perforation patterns, see this video and this one.

Field controlling
decorative
hole pattern
nTopology



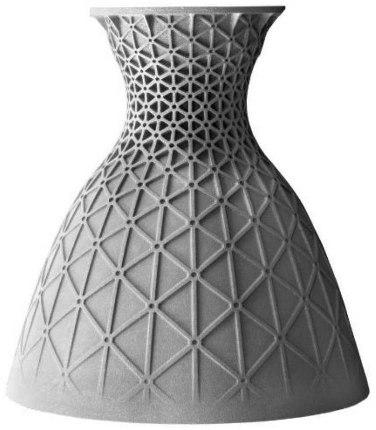
4.9 Advanced Manufacturing

Advanced manufacturing processes can create materials with anisotropic materials properties, and shapes that conform to specified geometry. Field driven design enables geometric and functional information to control the direction of the material properties by driving the build and process directions with fields.



4.10 Field-Driven Design of Isogrids

In this example, fields are used to control the spacing and height of ribs in an isogrid structure on the outside of a rocket nozzle. Isogrids provide reduced weight with excellent stiffness.



5 Summary

Fields are a central and pervasive concept in nTop Platform. There are many ways you can construct fields, and many parameters that you can control by using them. nTop Platform is unique in this respect.

Fields give you a convenient way to manage complex geometry. They provide a “gradients for geometry” capability that lets you control variations in size and shape from one point in space to another, giving you unprecedented design freedom and flexibility. Quite often, a field allows you to express a design intent that is simple to state in words, but surprisingly difficult to realize in traditional CAD systems.

Field-driven design allows you to create designs that are easy to imagine, and easy to describe, but would be very difficult to create by any other means. In this way, field-driven design forms an essential element of nTopology’s mission: to empower engineering teams to design transformative products

6 What Next?

Field-driven design is often fun, which makes it a popular topic among our Application Engineers. So, if you want to learn more about the subject you will find quite a few relevant blog posts and “nTop Live” videos on our web site. The list below provides links to most of the ones that were available when this document was written, but new ones appear every week. If you search the internet for “field-driven design”, you’ll find lots of material authored by our folks. The documents below are arranged roughly in order of increasing erudition: the first few should be fairly easy to understand, but the later ones might require some effort, for some people.

Andrew Reitz’s blog post on fields and implicits

Chris Cho video using fields to control a hole pattern

John Graham & Jonathan Harris video introducing fields

Sam Kratky using fields to design decorative textures and grills

[Andrew Reitz using fields to apply textures to consumer products]

[Evan Pilz using simulation data to control shell thickness and lattice beams]

[Gabrielle Thelen using fields to control gyroid-based shapes]

[The nTopology modeling technology white paper]

[A tutorial video by our customer Matt Shomper from Tangible Solutions]

[And another good one from Matt Shomper]

[Deeper insights into fields, from our CTO, Blake Courter]

[Advanced field-driven design techniques for heat exchangers]

If you decide that field-driven design looks useful to you, then you can [request a demo or a trial copy] of the nTop Platform software to investigate further. The files used in the nTop Live videos are all available, which will give you a good place to start your explorations.