9_digital simulation

particle-spring systems

"Science is a tool for ideas [...] and it is not only a means to verify structural strength. Science must lead us to discover the optimized geometry for that particular static (or dynamic) condition".

Sergio Musmeci

Structures that transmit forces through axial compression or tension have an increased capacity to withstand loads with smaller cross sectional areas. Traditional form-finding strategies for axially loaded structures include: complex physical models, hanging chain networks, stretched fabrics, soap films etc.. These techniques are difficult and time consuming. As a result, few designers investigated these form-finding potentials. Traditional form-finding techniques can now be digitally found using *particle-spring systems* that simulated the physical behavior of deformable bodies. Originally developed for character animation and cloth simulation, *particle-spring systems* have emerged as a powerful technique for form-finding. Whereas traditional techniques were difficult to apply, digital simulations allow designers to investigate form, in real time, by updating forces, supports and physical properties.

A particle-spring system is a **discretization** of a continuous model into a finite number of masses, called particles, connected by perfectly elastic springs. The main components of a *particle-spring system* are:

- **Particles**: each particle in the system is a lumped mass, that changes position and velocity as the simulation evolves.
- **Springs:** a spring is an elastic linear connection between two particles that behaves according to the Hooke's law: *a spring has an initial resting length and a stiffness value (k).*
- **Forces:** weights and external loads are simulated by vectors that are applied exclusively to particles.
- **Anchor Points:** particles that do not change position during the simulation.

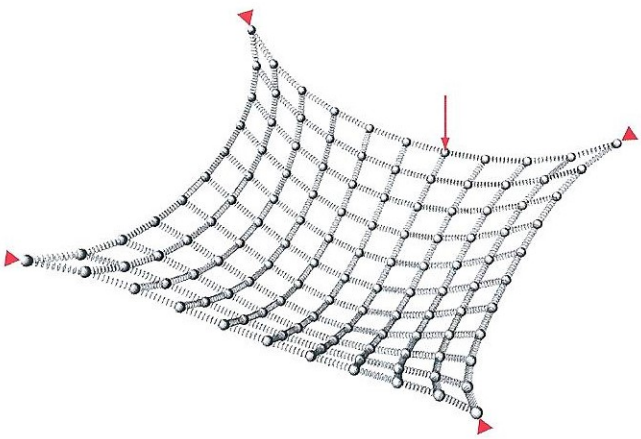


FIGURE 9.1
A particle-spring system that simulates a square membrane anchored at its corners. Force vectors are applied to the particles.

Once the simulation has started, the particles move from their initial position until they reach an equilibrium state which is dependent on the initial geometry, the force vectors, and the springs' defined properties. In accordance with Hooke's law, the lower the *stiffness* or *k value* the greater the spring elongation. Since particles in this system behave like spherical hinges without the capacity to resist moment forces, equilibrium solutions carry defined loads exclusively through **axial forces**. This is the ideal condition for form-finding strategies.

9.1 Kangaroo plug-in

Particle-spring systems (PSS) iterative calculations approach an equilibrium state were the sum of all forces is zero. The iterative calculations are performed by mathematical **solvers**.

Solvers operate iteratively within a main *engine,* meaning every subsequent iteration narrows the position and velocity of particles from the previous step towards an equilibrium solution. This process, similar to a key-frame animation, creates the illusion of movement when frames are calculated in a continuous sequence. Several particle-spring software packages have been developed recently including: CADenary[25] written by Axel Kilian, Dan Chak and Megan Galbraith in 2002. Most particle-spring software packages are standalone and do not fully integrated into a CAD or other modeling software. Furthermore, many of these standalone products are difficult to master. In contrast, **Kangaroo** a physics based particle-spring system engine developed by Daniel Piker[26] (development team: Robert Cervellione, Giulio Piacentino, Daniel Piker), is easy-to-use, designer-oriented, and is integrated as a plug-in for Grasshopper.

FIGURE 9.2
The Kangaroo toolbar.

Kangaroo enables designers to interact with form through *particle-spring system* simulations in real time. There are two interaction methods:

- **Direct Interaction**: includes the manipulation of anchor points, forces and spring properties;
- **Parametric or Associative Interaction**: anchor points, forces and spring properties can be parametrically linked to other parts of the 3D model. For example, the anchor points can be defined as the end points of a set of lines whose position is defined by another part of the algorithm.

NOTE 25
http://designexplorer.net/newscreens/cadenarytool/cadenarytool.html
NOTA 26
Daniel Piker is a researcher at the frontier of the use of computation in the design and realization of complex forms and structures. After studying architecture at the AA, he worked as part of the Advanced Geometry Unit at Arup, and later the Specialist Modelling Group at Foster+Partners. He has taught numerous studios and workshops and presented his work at conferences around the world, and consults and collaborates with a wide range of practices and researchers.

9.2 Kangaroo workflow

The workflow of Kangaroo relies on the same set of rules and operations for low nodal models, such as single digital chains, as high nodal models such as multi-supported membranes. The Kangaroo workflow is illustrated below:

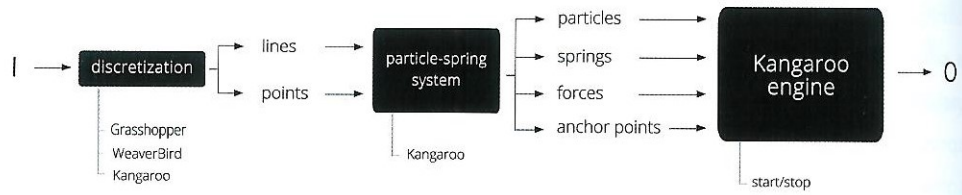


FIGURE 9.3
The Kangaroo workflow.

- **Discretization**: a deformable body, e.g. a fabric membrane or a flexible cable, is created by *discretizing* NURBS-geometries and subsequently processing the resulting geometry with a *particle-spring system*. Kangaroo requires that NURBS-curves are converted to lines and NURBS-surfaces are converted to meshes (i.e. points and lines). Kangaroo cannot process NURBS-surfaces and NURBS-curves. The main components used for discretization are hosted within the *Extract* panel of Weaverbird. We can also find useful components within Grasshopper standard tabs or Kangaroo tab.

- **Particle-spring system**: after a geometry has been discretized, lines are converted into springs and points into particles using specific components hosted within the Kangaroo toolbar. Vectors representing forces are applied to particles, and the anchor points are assigned.

- **Kangaroo Engine:** particles, springs, forces and anchor points are connected to the *Kangaroo Engine* in their respective slots. Toggling between True and False statements, using the component *Boolean Toggle* (Params > Input), starts and stops the simulation. While the simulation is running, particles move until an equilibrium state is reached. For this reason the engine's output can be considered as *dynamic*.

9.3 Cable simulation

In the first example, the behavior of a flexible elastic cable – suspended between two ends points and subjected to loads imposed by self weight – is simulated. The first step in the definition is to set a horizontal line drawn in Rhino using the *Curve* container component.

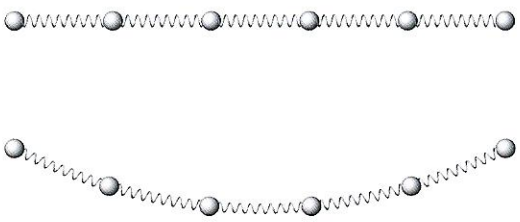
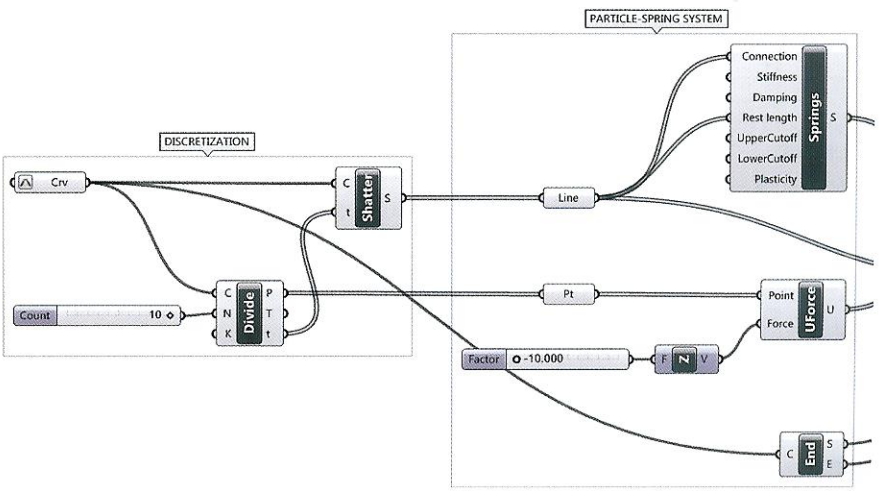


FIGURE 9.4
Representation of a particle-spring system that simulates a flexible cable.

- **Discretization**: the initial geometry is discretized by splitting the set line at division points – calculated by the component *Divide Curve* – using the component *Shatter* (Curve > Division). The N-input of *Divide Curve* sets the number of divisions: the greater the number of division points the greater the final deformation. The *Divide Curve* points are the particles in the system where force is applied or restraints are set. In the following example, N is set to 10.



- **Particle-spring system**: the component *Shatter* outputs a series of unique lines that are converted into springs using the component *Springs From Line* (Kangaroo > Forces), generating the system's springs.

  The output (S) of the *Shatter* component stored in the container component *Line*[27] is connected to the Connection-input and to the Rest Length-input of the *Springs From Line* component. The output (P) of *Divide Curve* is connected to the Point-input of the component *Unary Force* (Kangaroo > Forces) which applies a force-vector to every input point or particle. In this case, a force vector acting in the negative Z direction with a magnitude of 10 is set. Force vectors can be set in any desired direction. Lastly, the anchor points are defined as the end points of the initial curve.

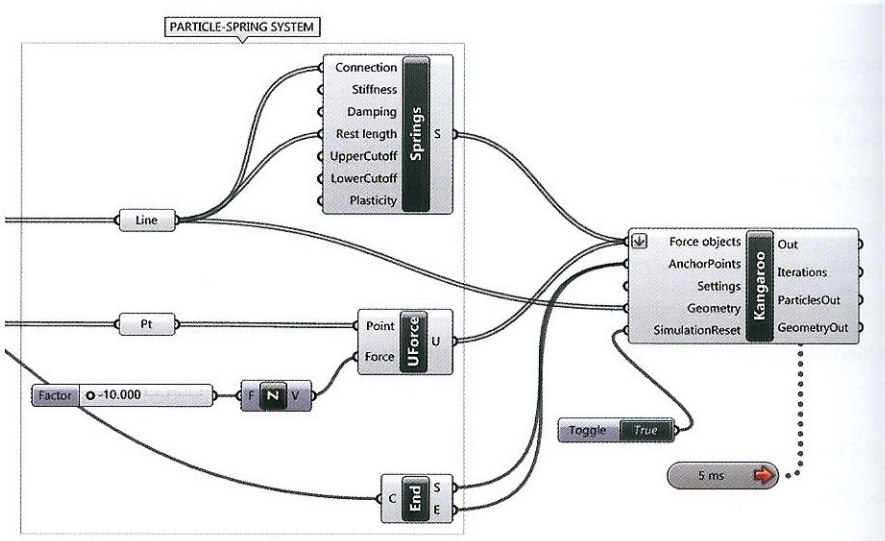


FIGURE 9.5
Particle-spring settings.

- **Kangaroo Engine**: the *KangarooPhysics* engine (Kangaroo > Kangaroo) collects the outputs of *Springs From Line* and *Unary Force* in the Force objects slot, **which must be set to "flatten"**. The AnchorPoints input is satisfied by the end points of the initial curve. By default the simulation influences only the particles, which change velocity while the

NOTE 27
The *Line* container is useful to spot any inputs different from line-geometries (*Springs From Line* can process just line-geometries). If other types of geometries, e.g. curves, come in the *Line* container, it will turn red.

simulation is attempting to reach equilibrium. To visualize the simulation of the cable seeking equilibrium, the output of the *Shatter* component is connected to the Geometry -input of the *KangarooPhysics* component.

To start or stop the simulation a *Boolean Toggle* and a *Timer* (Params > Util) are connected to the *KangarooPhysics* component; which starts (False) and stops (True) the simulation and defines the number of frames per second respectively. Alternatively, if the component is double clicked a contextual panel will appear with a set of buttons: *stop*, *play*, *pause*.

In the example, the *Timer* is set to 5 milliseconds (right-click > Interval > 5). The simulation can be started by double-clicking the toggle and changing the Boolean statement to False. Once the simulation is started the particles move in the direction of the force vectors which are restrained by the elastic-linear springs properties. The cable's geometry changes several times until it reaches an equilibrium state.

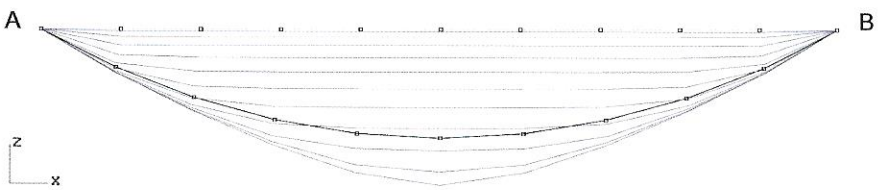


FIGURE 9.6

The frontal view shows the sequence of positions taken by a cable while the Kangaroo simulation is running. The cable bounces several times until it reaches an equilibrium state (dark polyline).

If changes are made to the initial geometry or to the discretization process the simulation is required to be reset and then started again, to visualize the influence of the changes. Other parameters, such as the *Unary Force* direction and magnitude, and the location of the anchor points can be changed during the simulation.

The position of the constraints can be changed manually by setting the anchor points from Rhino instead of relying on the *End Points* component. Once the simulation has started the position of the anchor points can be cchanged "manually": the simulation will react to the change and, once again, seek equilibrium.

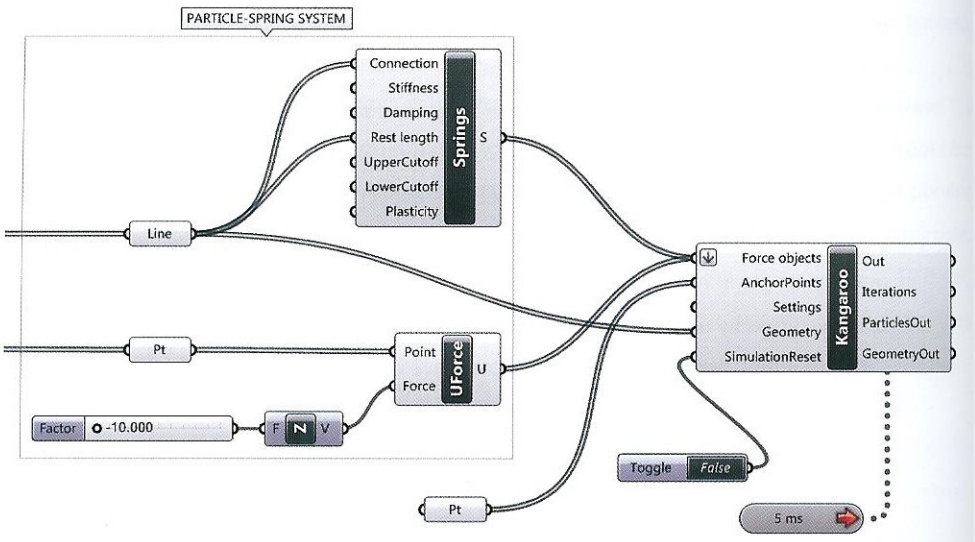


FIGURE 9.7
The anchor points set from Rhino are collected in Grasshopper by the container component *Point*.

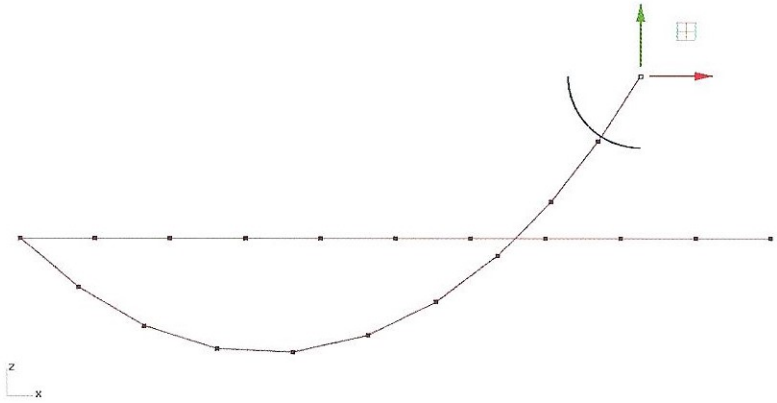


FIGURE 9.8
We can manually move the anchor points set from Rhino: the cable will react as if it were in the physical world.

Manual interaction with the model can lead to errors if the points are not returned to their original position before stopping and starting a new simulation. If the simulation is run without returning the anchor points to their original position the physics engine will not recognize how to restrain the model.

To add additional or new anchor points, points can be set from Rhino or calculated within Grasshopper. Anchor points must be always positioned at particles. For instance, an anchor point drawn in the middle of a spring will have no influence.

FIGURE 9.9
A cable simulation with three anchor points.

9.3.1 Strategy: continuity

Kangaroo's inputs are defined as points, lines or meshes, Kangaroo's output is similarly defined as the same kind of geometries. An elastic cable can be simulated by connecting the ParticlesOut-output to the V-input of the component *NURBS Curve*, defining an interpolated curve through the points. This strategy can be useful to achieve continuity; however, it can lead to physically incorrect results since the Nurbs-curve acts rigidly around point B, which is impossible since particles act as spherical hinges, without moment capacity.

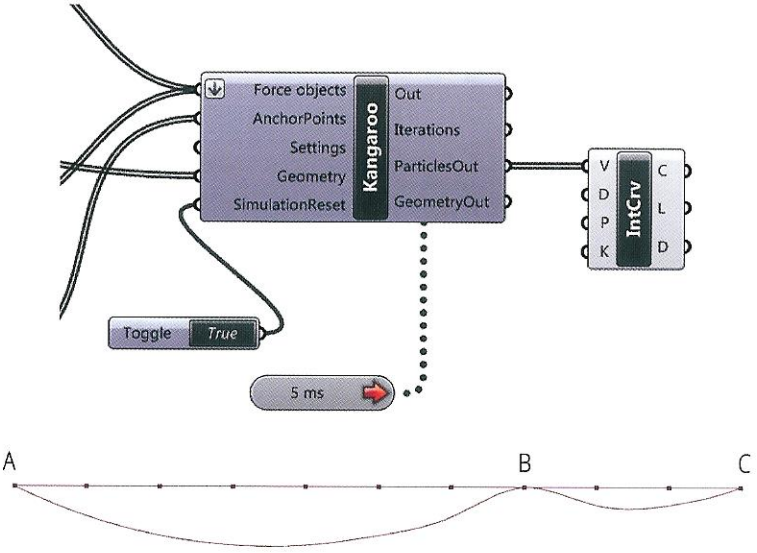


FIGURE 9.10
How to get continuity using an *Interpolate Curve* component connected to Particles Out.

9.4 Elastic behavior: Hooke's law

When the cable reaches an equilibrium state induced by influence of the external forces, and resisted by the elasticity of the springs, the length of each segment increases. For example, the cable illustrated below has a start length of 10 units. The curve is discretized into 5 parts each 2 units in length. Six identical unary forces with a magnitude -20 units in the *Unit Z* direction, are applied to the particles; yielding a final curve with an overall length of 10.46 units, hence a deformation of 0.46 units. This deformation distance is not split evenly among the segments, instead the springs closer to the reactions elongate further. This is because, they bear the weight of the other springs.

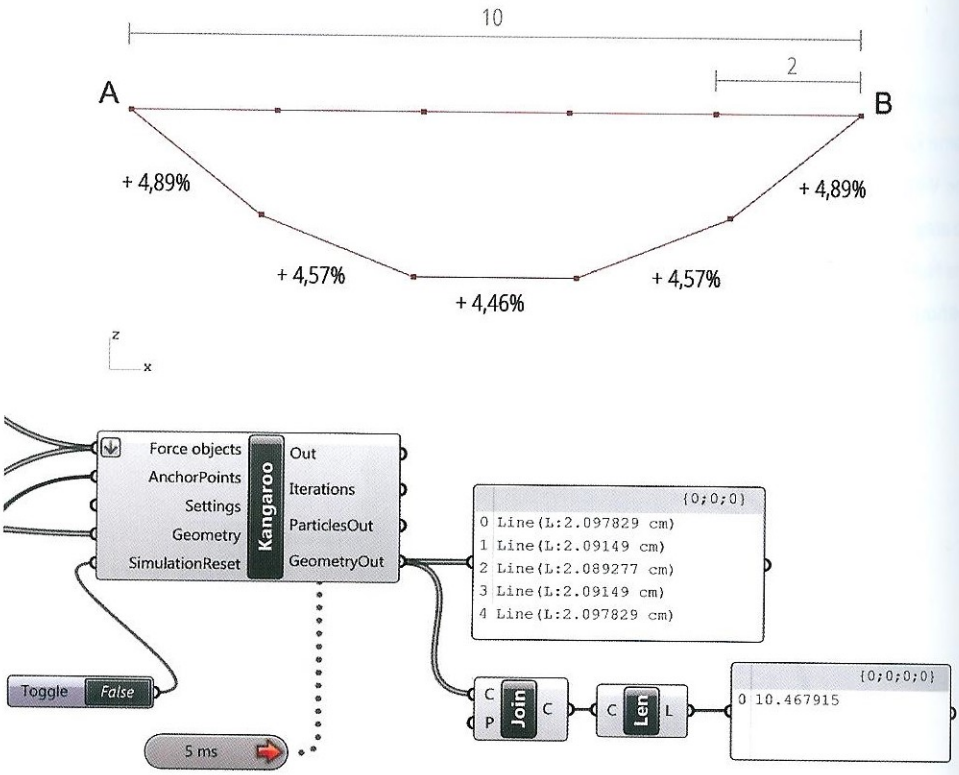


FIGURE 9.11
The deformation causes an uneven extension of the springs with a higher extension of anchored springs.

Kangaroo's elastic behavior follows Hooke's law which states: *displacements or size of the deformation of a body (treated as a spring) is directly proportional to the deforming force or load.* Under these conditions

the body returns to its original shape and size when the loads are removed. Mathematically, Hooke's law is formulated by the expression:

$$F = k \cdot X \quad [1]$$

where:

- **F:** is the applied force, commonly expressed in Newtons (N);
- **k:** is a positive constant called the *Stiffness*. The value of *k* depends on material and cross sectional geometric properties of the elastic body. The constant *k* is commonly expressed in N/cm;
- **X** is the *change in length* or deformation of the body (spring), commonly expressed in cm.

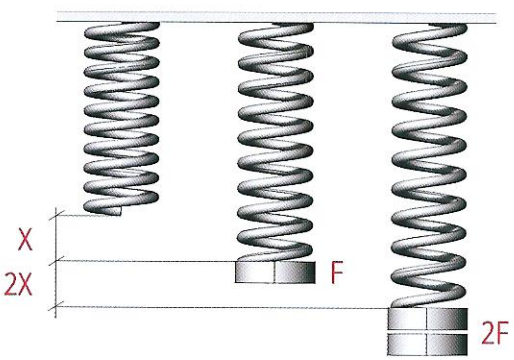


FIGURE 9.12

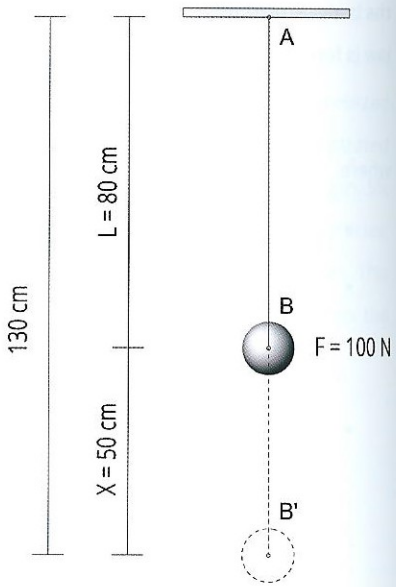
Hooke's law: the springs extension is proportional to the force.

Hooke's law is embedded in Kangaroo's *Springs From Line* component.

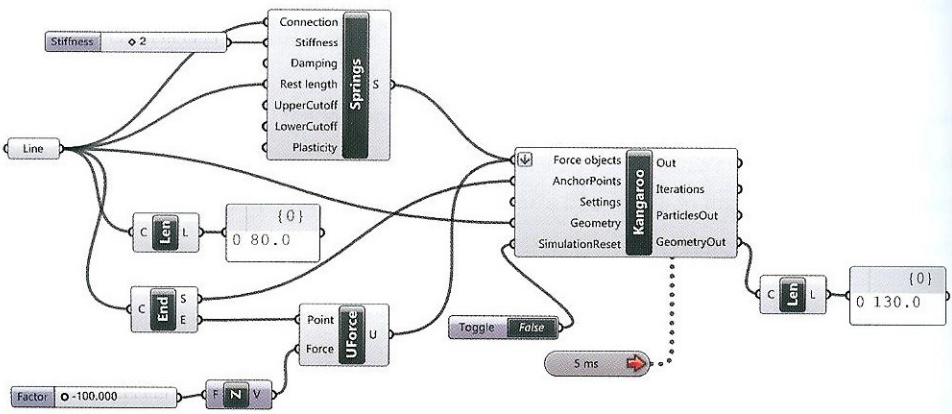
To illustrate how the component *Springs From Line* operates a 100 N weight is applied to an anchored cable with an initial length of 80 cm and a *stiffness* of 2 N/cm. In accordance with Hooke's law, the change in length of the cable is:

$$X = F / k \rightarrow X = 100 / 2 = 50 \text{ cm}$$

FIGURE 9.13
Deformation of a suspended cable according to the Hooke's law.



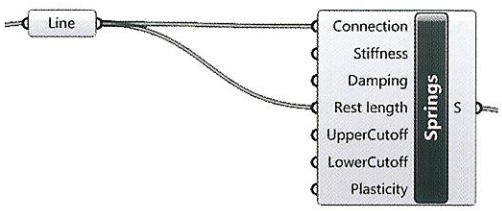
If the load is removed, the cable will return to its start length (80 cm). The length a cable reaches when loads are removed is called the ***Rest Length***, which not always coincides with the start length (as explained later). The time the cable takes to reach an equilibrium state depends on the *Stiffness* as well as the *Damping Constant* which is related to friction caused by drag. The greater the *Damping Constant* the lower the deformation velocity. The *Damping Constant* does not affect the change in lenght but only the deformation velocity.



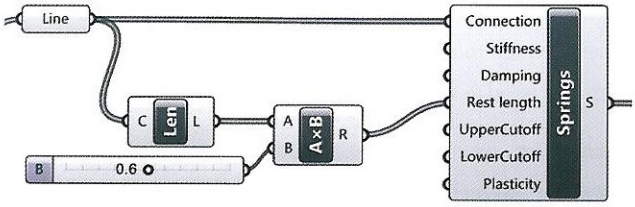
The relative algorithm in Grasshopper considers the cable as a single spring; the calculated results of the spring's deformation match Hooke's law. The *Springs From Line* component embeds the physical characteristics discussed in the previous example, as well as other important parameters including:

- **Connection**: springs are linear elastic connections. The Connection-input requires *lines,* any other geometry will yield null results. Every line's initial length is called the *Start Length*.
- **Stiffness:** according to the Hooke's law, the Stiffness-input, sets the springs' stiffness or k value. The higher the k value, the lower the deformation. Stiffness is determined by material properties as well as the area of spring's cross-section.
- **Damping:** the Damping-input influences the deformation velocity, with no influence on the change in length. By default Damping is set to 10.
- **Rest Length:** the length that a spring endeavors to reach once the loads are removed. The Rest length-input is essential to simulate behaviors of different materials. Three cases can be distinguished:

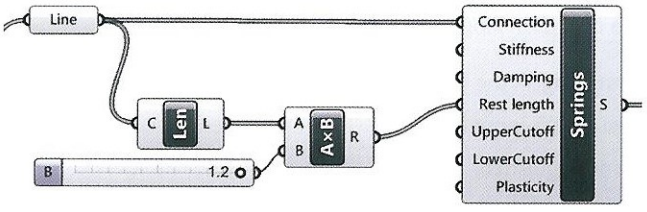
    1. *Rest Length* = *Start Length*. This condition mimics perfectly elastic behavior and is achieved by connecting the springs (Lines) to the Rest length-input. All simulations demonstrated so far have followed this condition.



    2. *Rest Length* < *Start Length*. This condition imitates the effect of pre-tensioning the springs which **shortens** their length. *Rest Lengths* which are less than the *Start Length* simulates tensile materials that attempt to minimize their length or area. This condition is accomplished in Kangaroo by using the component *Length* to measure the initial springs' length then multiplying it by a **Rest Length Factor** ranging between 0 and 1.



3. *Rest Length* > *Start Length*. This condition replicates the relaxation or lengthening of springs. This condition is accomplished in Kangaroo by using the component *Length* to measure the initial springs' length then multiplying it by a *Rest Length Factor* ranging between 1 and N.



- **Upper/Lower cutoff**: sets limits for the springs to operate, below or above respectively. By default the *Upper/Lower cutoffs* are set to 0.
- **Plasticity:** the maximum elastic deformation, as compared to the rest length.

9.5 Catenary simulation

A catenary curve is formed by a perfectly flexible, uniformly dense, and inextensible cable suspended from two end points. The equation of a catenary curve is:

$$y = a \cdot \cosh(x/a) \quad [2]$$

The distance from the x-axis to the point on the curve with a tangent line slope equal to 0 is expressed as variable *a*.

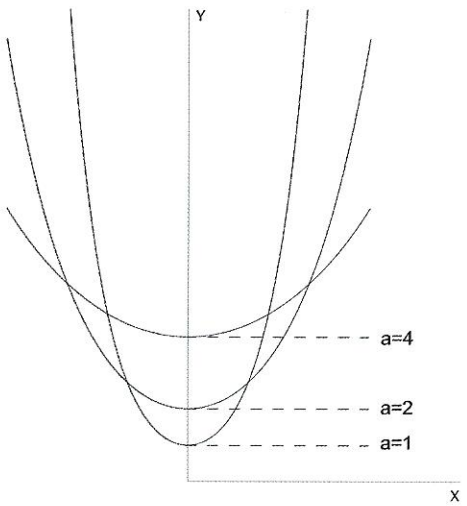


FIGURE 9.14
Catenary graph for different values of *a*.

The curve can be calculated by the *Evaluate (Fx)* component, as showed below, with *a* set to 2.

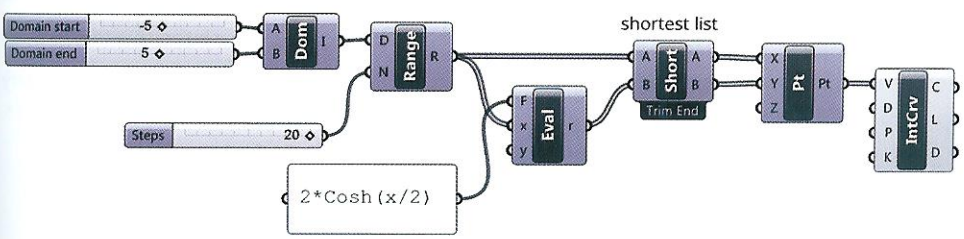
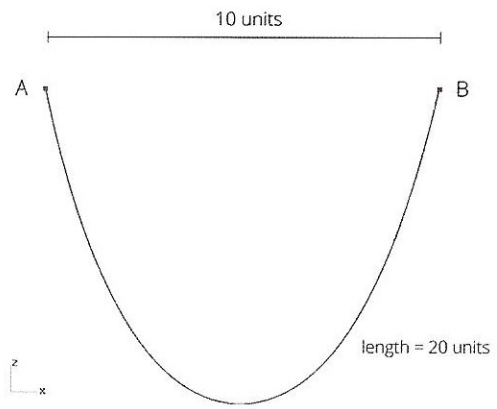


FIGURE 9.15
Algorithm based on the catenary equation.

A catenary curve can also be drawn by the component *Catenary* (Curve > Spline), which embeds the equation of a catenary curve [2]. The *Catenary* component requires as inputs: start (A) and end (B) points of the catenary curve, the length (L) of the curve, and the gravity direction (G).



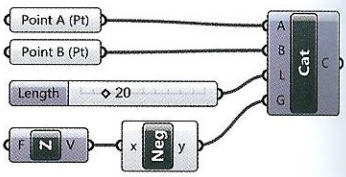


FIGURE 9.16
The catenary algorithm and generated curve.

Catenary curves can also be simulated using *particle-spring systems*. The catenary definition is: "a *curve formed by a perfectly flexible, uniformly dense, and inextensible cable suspended from its endpoints*". Therefore, the curve must comply with four conditions:

1. To be suspended by its end points;
2. To be perfectly flexible;
3. To be uniformly dense;
4. To be inextensible.

These conditions are not entirely met by the deformable linear-curve discussed previously (9.3). In fact, even if a high *stiffness* value is set, the springs will not be inextensible, so the fourth condition is not met. As a result the simulation will generate a curve that is slightly different from a catenary curve, as illustrated in the following image.

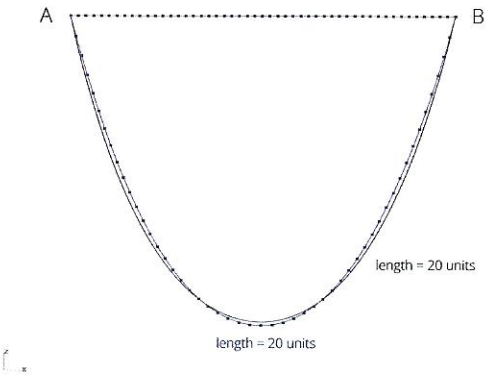


FIGURE 9.17
A line with applied gravity loads at the particles, will output a curve that is slightly different from an catenary curve.

To more closely approximate a catenary curve an arc can be used as the starting geometry, described using the component *Arc 3Pt*, through a set of three points. Then the measured arc-length is set equal to the length of the desired catenary curve.

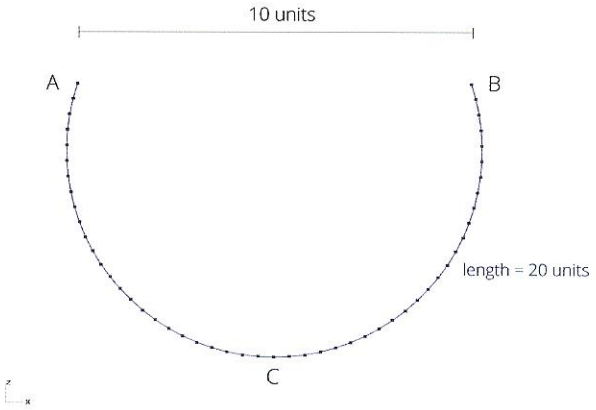
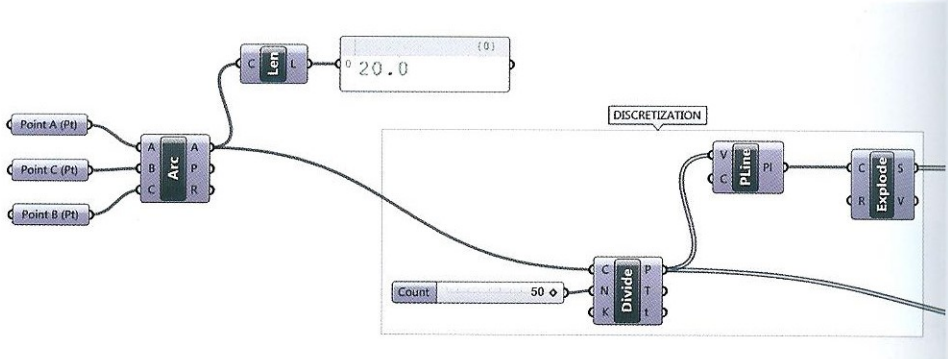
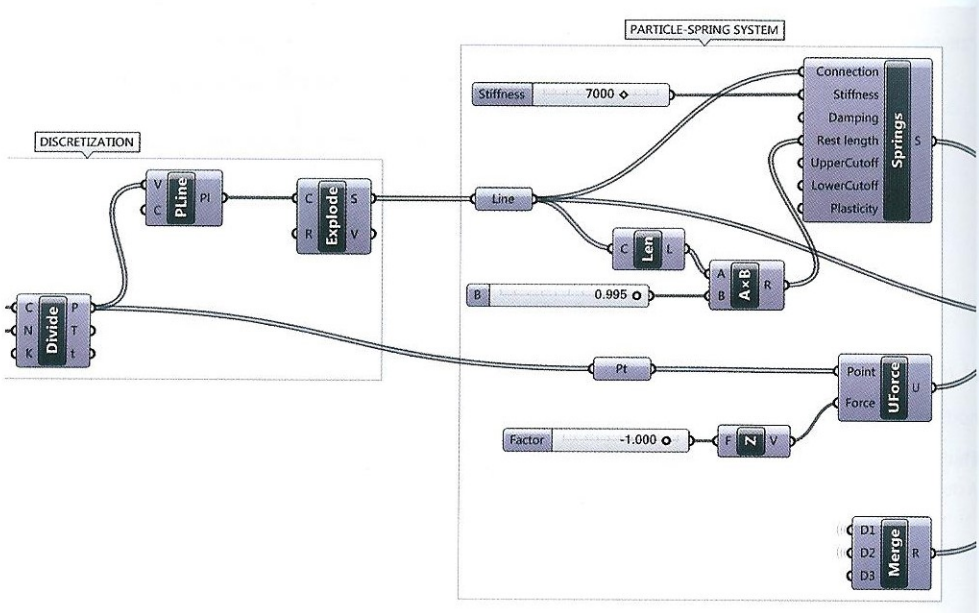


FIGURE 9.18
A close approximation of a catenary curve can be obtained starting from an arc.

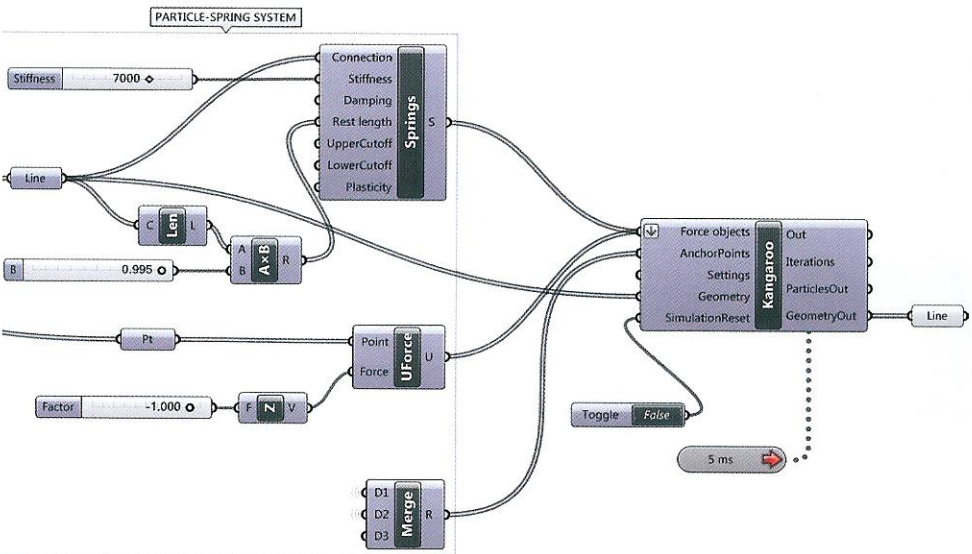
- **Initial geometry**: is an arc with a length 20 units drawn through three points: A, B and C.



- **Discretization**: the arc is then divided using the component *Divide Curve* with the N-input set to 50, yielding 51 equidistant points. A polyline is created through the resulting points and exploded into segments by the component *Shatter* in order to return 50 lines. The lines are then converted into springs, each initially measuring 20/50 = 0.4 units.



- **Particle-spring system**: the output of the *Explode* component is connected to the Connection-input of the *Springs From Line* component, after passing through a *Line* container component. To achieve *inextensibility* of the cable: the rest length is set such that the *rest length* < *start length* using a multiplier factor of 0.995, and a Stiffness-input set to 7000 units. Gravity loads are applied to each particle using the component *Unary Force* (Forces > Kangaroo) in the Z direction with a magnitude of -1, the resulting vectors are connected to the Force objects-input of the component *Kangaroo*. Set points A and B, are combined into a single list using the *Merge* component and connected to the AnchorPoints-input of the component *Kangaroo*.



When the simulation is initiated the segmented arc moves in the negative Z direction taking the form of a catenary cable. The calculated polyline complies with the four conditions of the catenary definition. The segments change minimally in length from their start length (0.4 units) after the simulation.

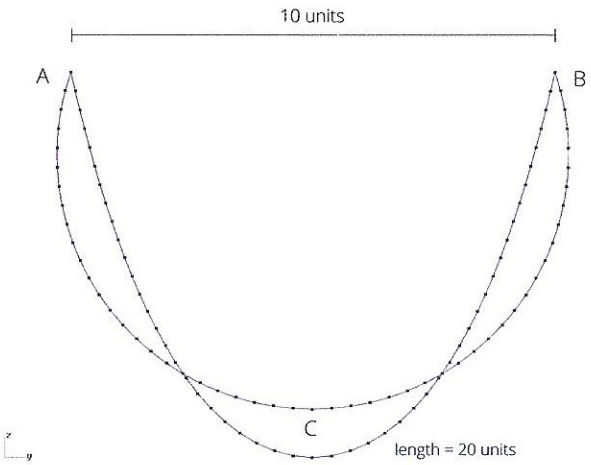


FIGURE 9.19

The segmented arc simulating a cable moves in the negative Z direction taking the form of a catenary cable.

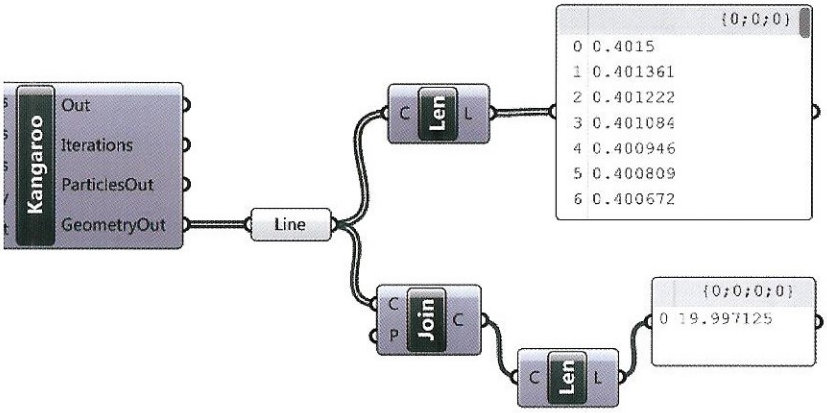


FIGURE 9.20

The segments minimal change in length from their start length (0.4 units) after the simulation.

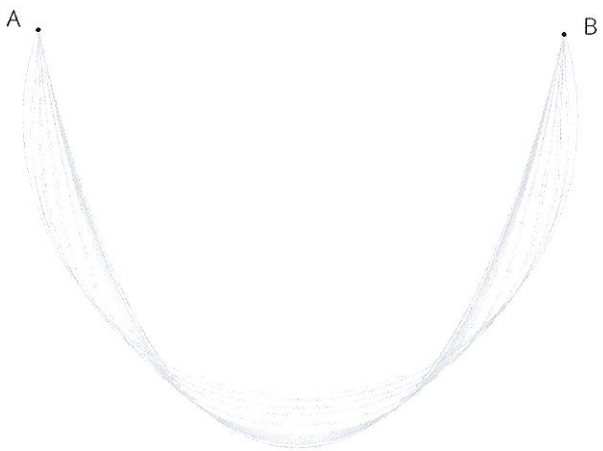


FIGURE 9.21
The image shows the sequence of positions taken by the arc-shape cable, from the initial configuration to the final catenary form.

So far the *Unary Force* value (representing self weight) has been set arbitrarily. More accurately, the *Unary Force* should be set by dividing the cable's total weight by the number of particles. If the catenary has a weight of 10N and is divided by 51 particles each unary force vector will have a magnitude of 10/51 = 0.196N.

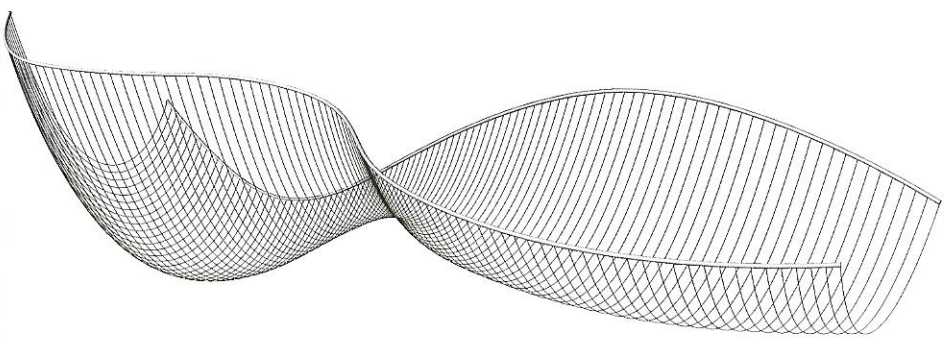


FIGURE 9.22
The image shows a set of catenaries whose start-points lie on two freeform frames.

9.6 Membrane simulation

To simulate membranes or other sheet materials such as fabrics, a grid of springs is defined. Grids can be established using numerous strategies; the most commonly used technique is to convert a NURBS surface to a mesh, then extract the edges and vertices that will become the springs and particles of a particle-spring system.

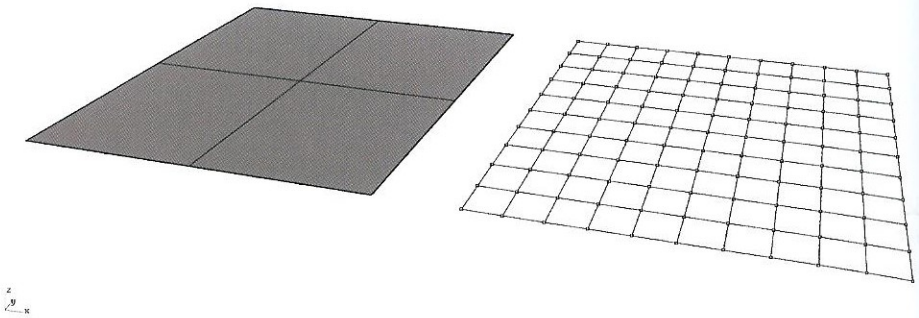
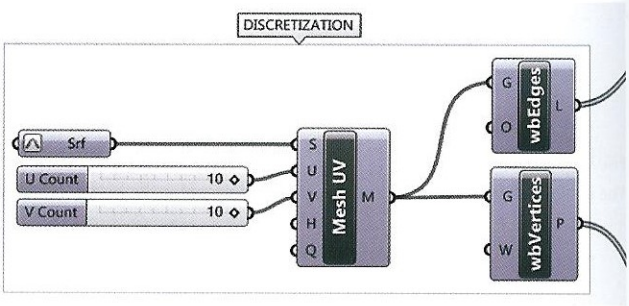
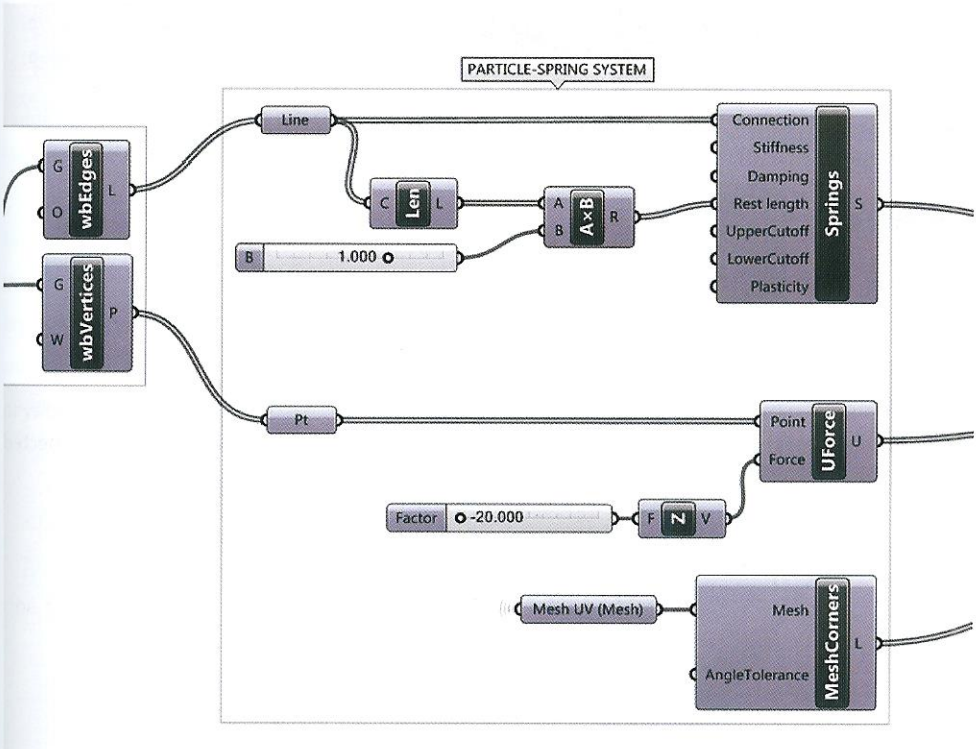


FIGURE 9.23
A grid of springs can be obtained by extracting vertices and edges from a mesh.

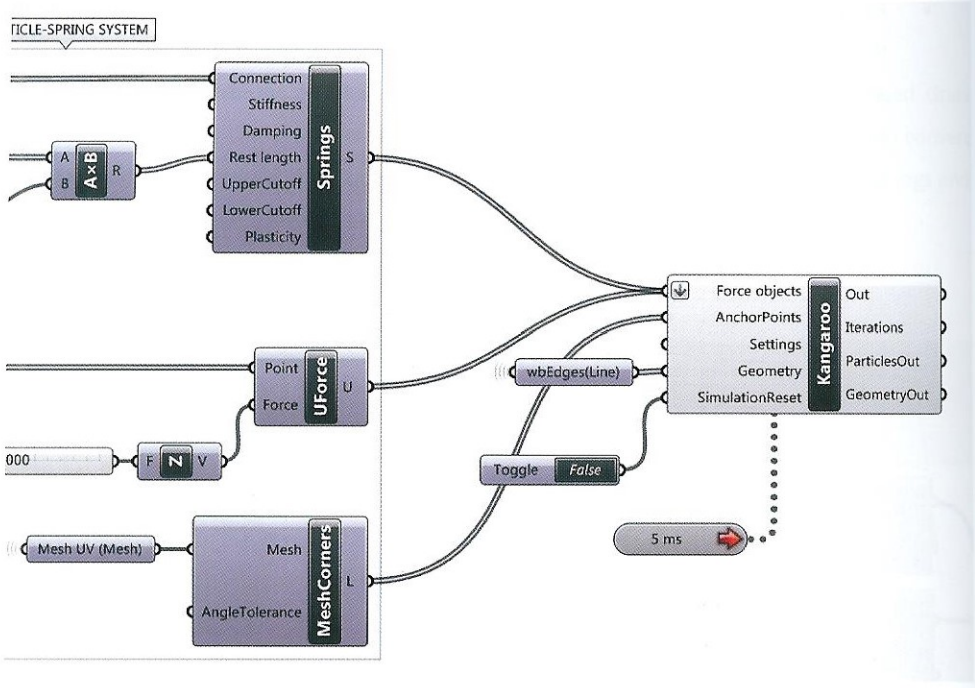
- **Discretization**: The behavior of a rectangular flat membrane anchored at four corners points and subjected to gravity loads can be simulated using particle-spring systems. To construct the discretized model: first, a set NURBS surface is converted into a mesh using the component *Mesh Surface* (Mesh > Util), then the components *wbEdges* (Weaverbird > Extract) and *wbVertices* (Weaverbird > Extract) are used to extract the mesh edges and vertices respectively.



- **Particle-Spring system**: after the model has been discretized, the edges are connected to the Connection-input. The Rest length-input is defined as a multiple of the start length using a *slider*, enabling the springs to be conditionally varied. The output (P) of the component *wbVertices* is connected to the input (point) of the component *Unary Force;* a force vector with a magnitude of -20 acting in the Z direction is applied to each particle. Next, the four corner-points are extracted using the component *MeshCorners* (Kangaroo > Utility) and connected to the AnchorPoints-input of the Kangaroo component.



Lastly the output (springs) of the component *Springs From Line* and the output (U) of the component *Unary Force* are connected to the Force objects-input of Kangaroo, in flatten mode. The *wbEdges* component output (L) is connected to the Geometry-input of the Kangaroo component to define the geometry for the simulation to output. Double-clicking the Boolean toggle switching from True to False initiates the membrane simulation. The membrane, anchored at its corners, is deformed by the *Unary Force* vectors and resisted by the springs stiffness value.



To visualize the mesh faces instead of edges the *Mesh Surface* component output (M) is connected to the Geometry-input of Kangaroo.

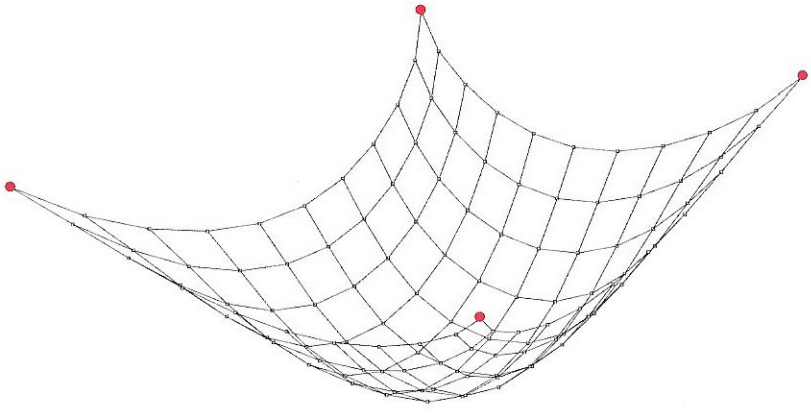


FIGURE 9.24
The membrane, anchored at its corners, is deformed by the *Unary Force* vectors and resisted by the springs stiffness value.

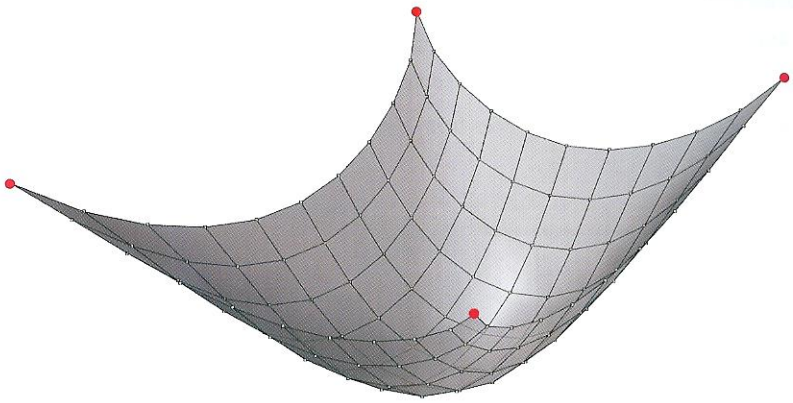


FIGURE 9.25
Mesh faces can be visualized instead of edges by connecting the *Mesh Surface* output to the Kangaroo engine.

Similar to cables, membranes anchor points can be set from Rhino then adjusted with respect to their XYZ position after the simulation is started. Manual interaction with the model can lead to errors if the points are not returned to their original position before stopping and starting a new simulation. If the simulation is run without returning the anchor points to their original position the physics engine will not recognize how restrain the model

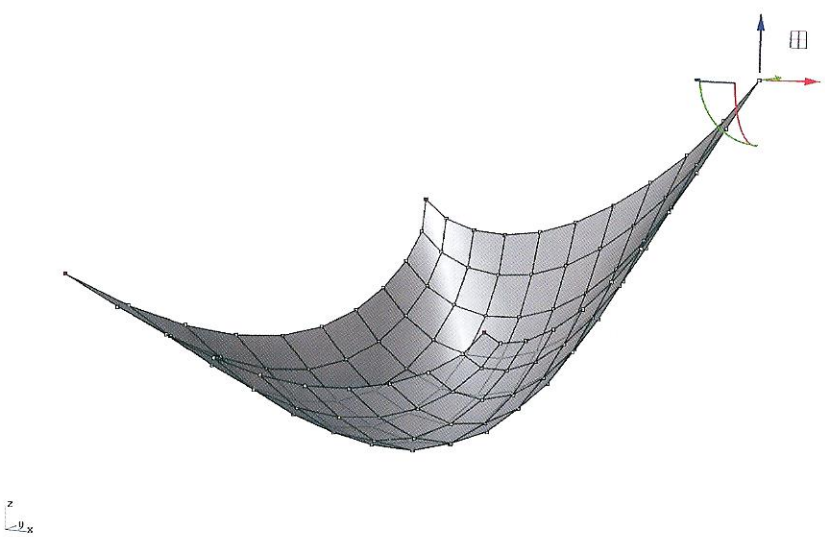


FIGURE 9.26
Membranes anchor points can be manually manipulated in Rhino.

Of course a membrane can be "multi-anchored" provided that anchor points are positioned on particles. Membranes can also be anchored along their edges, by setting the anchor points as the meshes naked vertices. The component *Naked Vertices* (Kangaroo > Utility) extracts the naked edge vertices, i.e. vertices not bordered by faces from a given mesh.

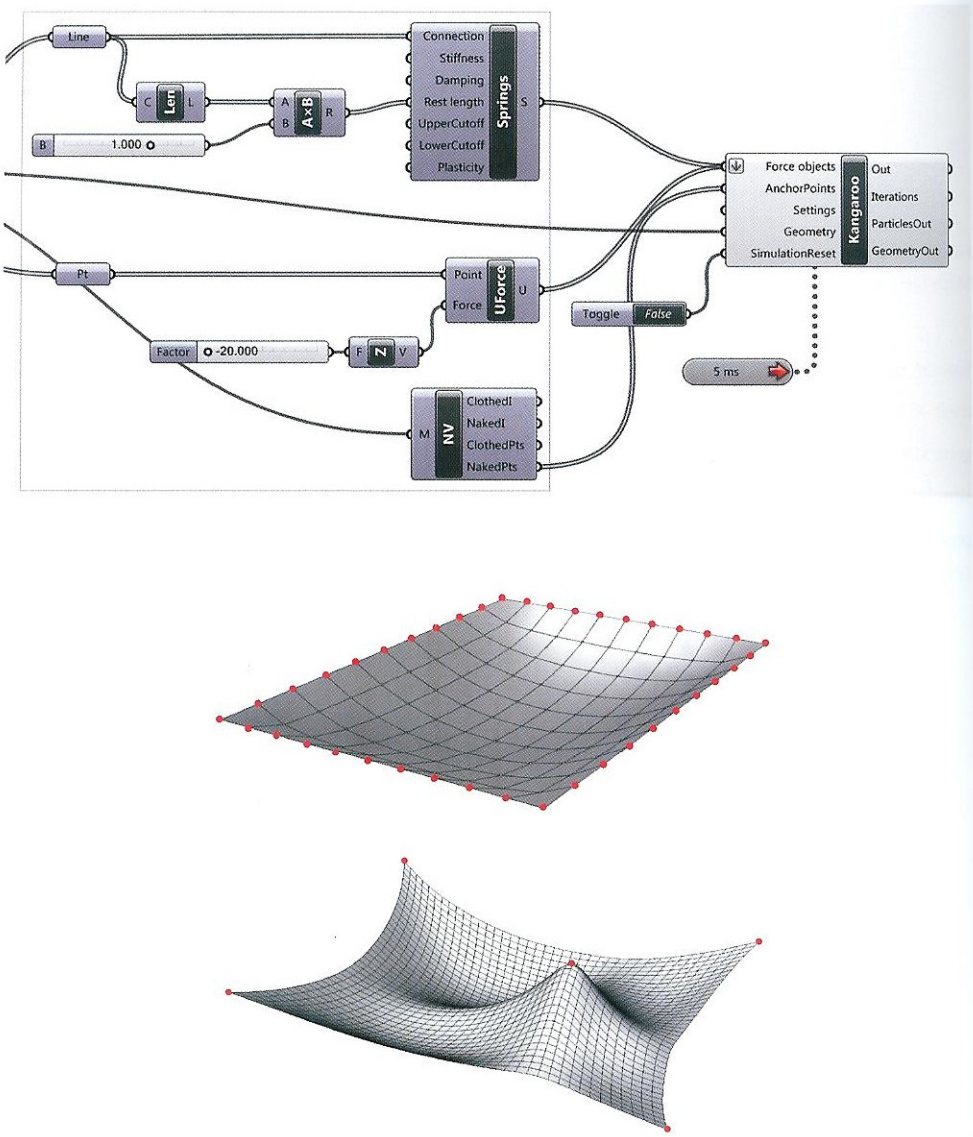


FIGURE 9.27
Top: membrane with naked vertices as anchor points. Bottom: membrane with corner and internal anchor points.

The membrane's area can be minimized by setting the rest length factor to 0. With a rest length set to 0 the membrane behaves like a tensioned-film simulating a soap film.

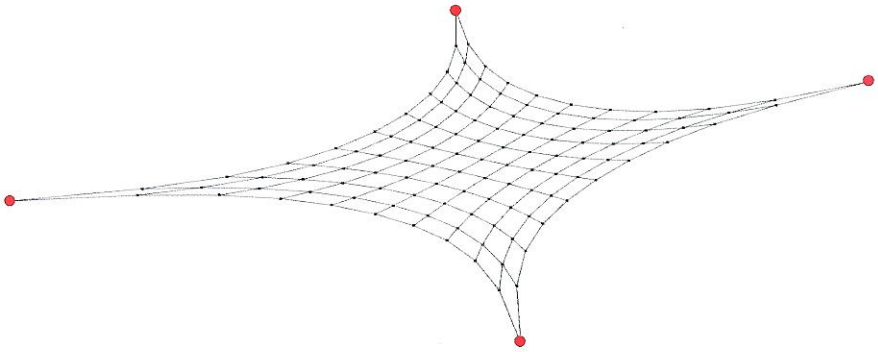


FIGURE 9.28
A membrane with Rest Length Factor equal to 0.

The membrane so far studied behaves like a **cable net**. To simulate a more rigid membrane, **diagonals can be added which prevent the mesh-faces from deforming into diamond shapes**. To computationally mimic the behavior of sheet materials two *Springs from Line* components are used to separate the mesh-edges and the diagonals.

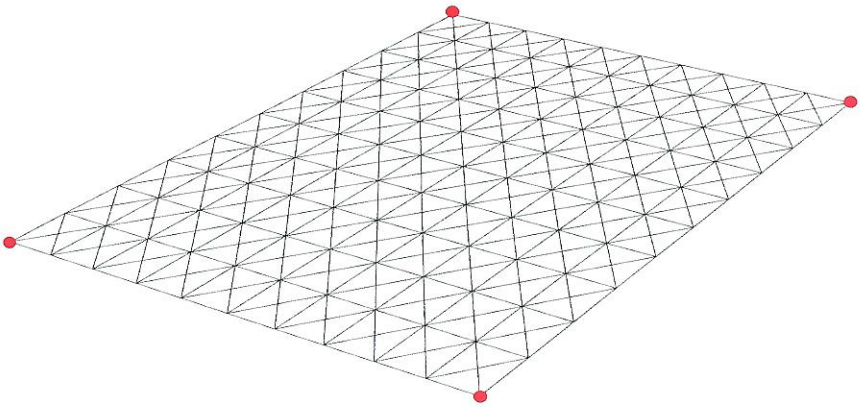


FIGURE 9.29
Diagonals prevent the mesh-faces from deforming into diamond shapes.

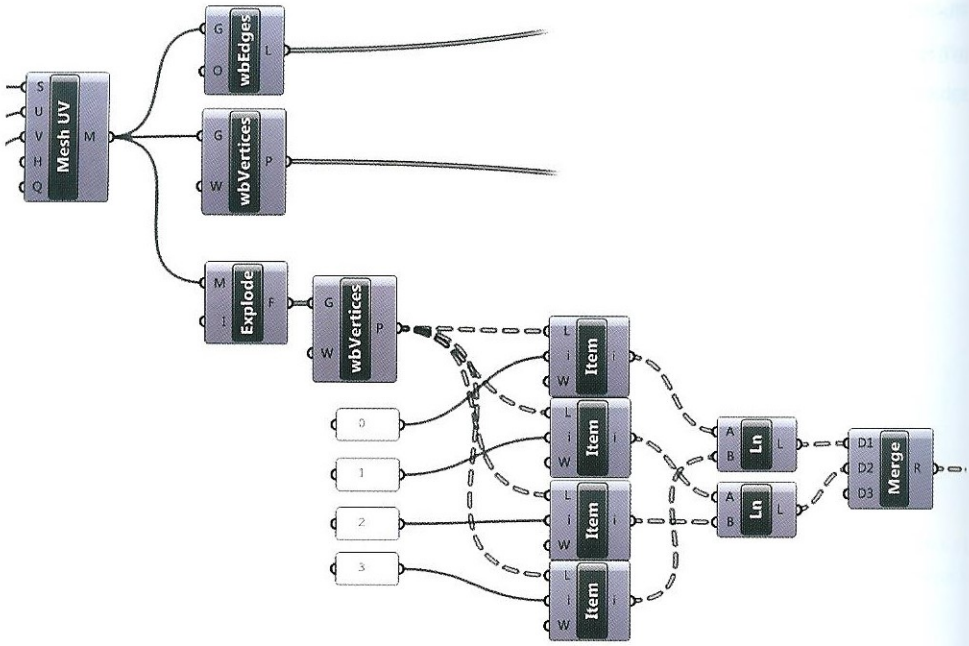
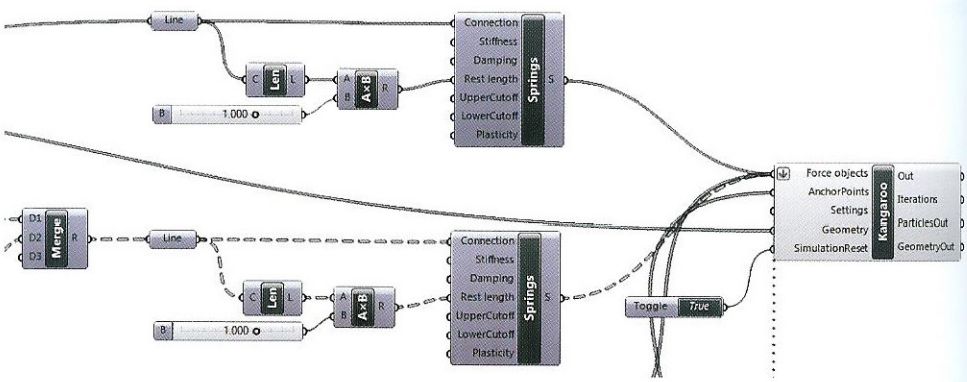
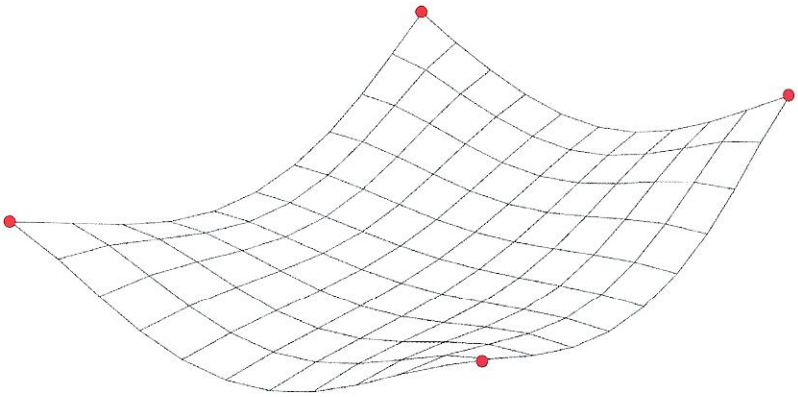


FIGURE 9.30

Two *Springs from Line* components are used to separate the mesh-edges and the diagonals.



Once the simulation is started the more rigid configuration no longer behaves like a cable-net, but instead like a sheet material. If the *Rest Length Factor* is reduced the membrane will form origami-like creases.



Additionally, different diagonals configurations can be tested which yield varying results. For example, if one diagonal is set per quad (image A), asymmetrical behavior can be achieved by setting different *Rest Length* values for the edges. The component *WarpWeft* (Kangaroo > Mesh) can also be used to separate the edges of a mesh according to *warp* and *weft* directions (C).

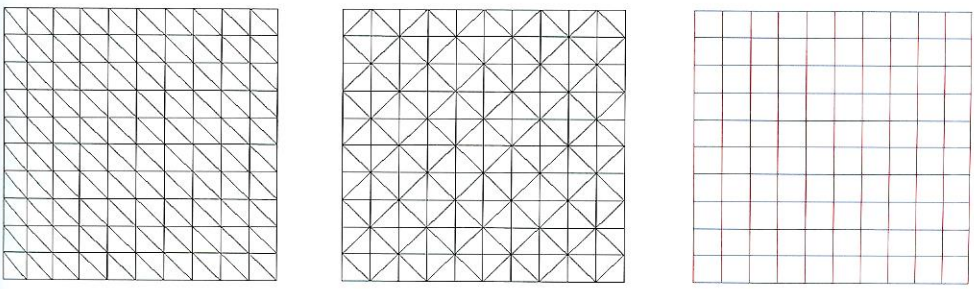


FIGURE 9.31
Different mesh configurations can be set to simulate specific behaviors.

9.6.1 Practical Exercise: multi-supported membrane

The Bach Chamber Music Hall by Zaha Hadid Architects, composed of a continuous ribbon of stretched fabric that intertwines to envelop the performers and the audience, is the focus of the following practical exercise.

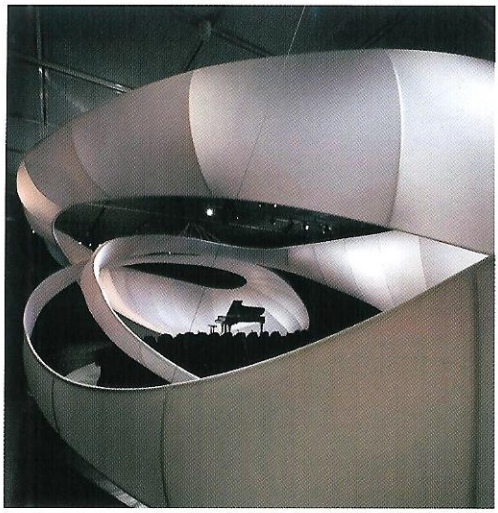


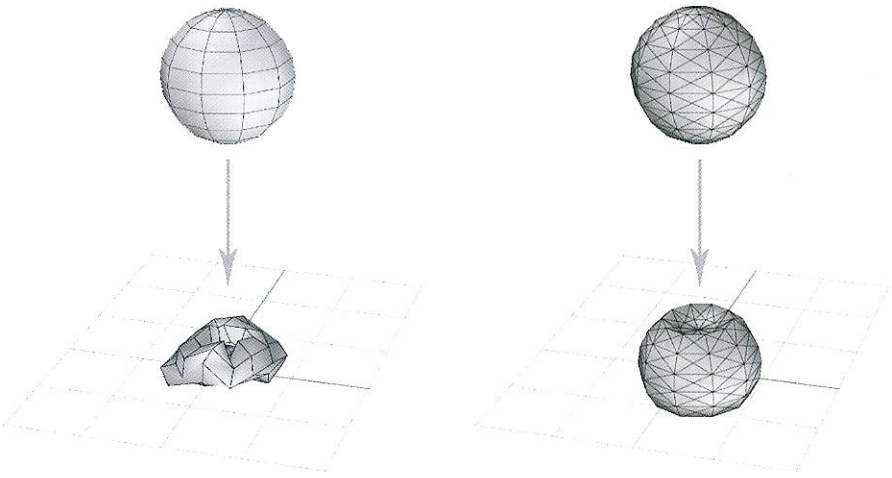
FIGURE 9.32
The *Bach Chamber Music Hall* by Zaha Hadid Architects. Manchester, UK (2009). Image courtesy of Zaha Hadid Architects. Image copyright by Luke Hayes.

Instead of studying the entire ribbon, a section is extracted and studied in a simplified model. Specifically, the model investigates the behavior of a membrane stretched on a truss support-structure composed of 2 horizontal free-form beams and 11 vertical variable profile arc-shape beams.

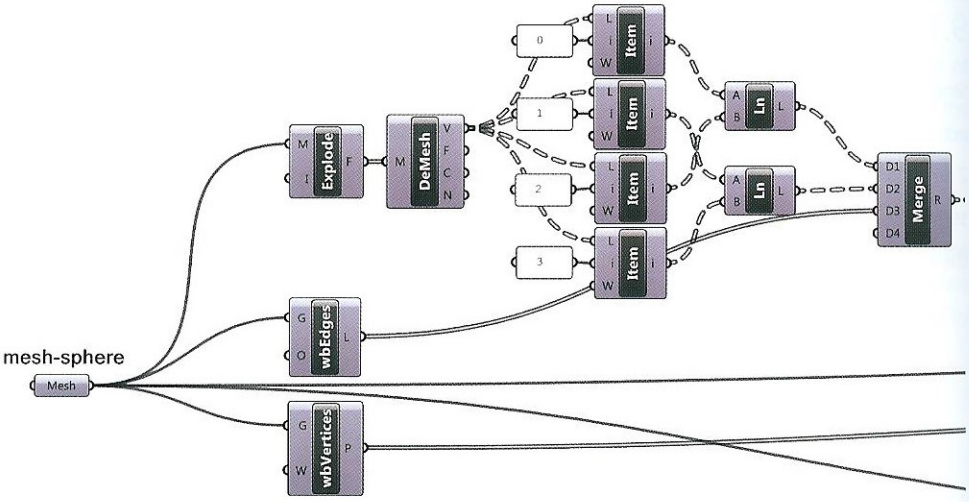
The complete exercise can be found using the following QR code. 4

9.7 Shell behavior

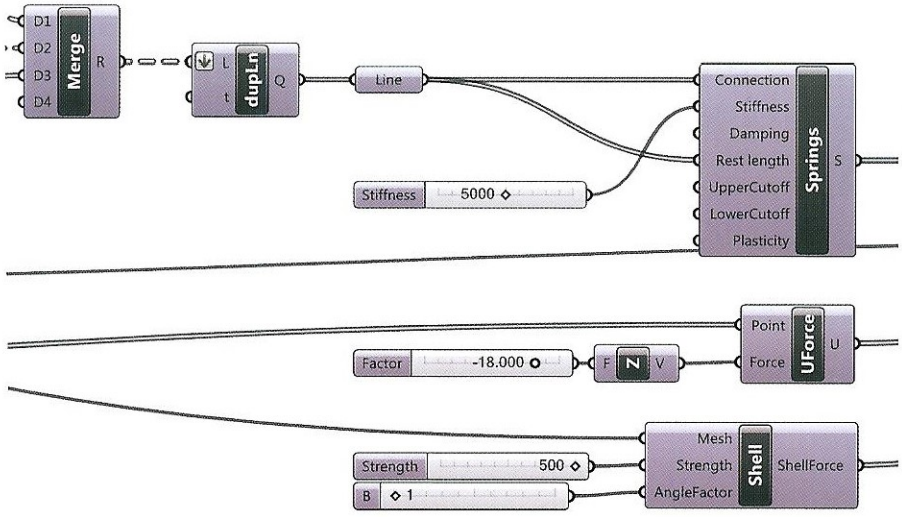
Since particles behave like spherical hinges, without moment capacity, a discretized model cannot act rigidly without additional restraint. For example, a simulated mesh-sphere moving in the negative Z direction under the action of gravity will crumple (even with reinforcing mesh diagonals and a high stiffness value) when the sphere reaches the XY plane or the simulated floor. To prevent crumpling the component *Shell* (Kangaroo > Utility) can be utilized.

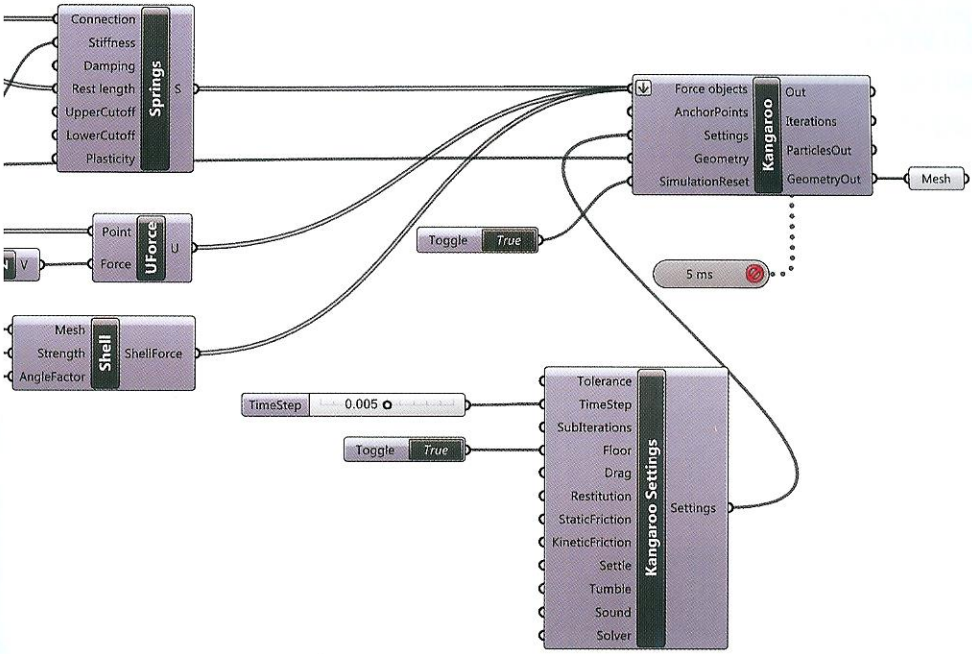


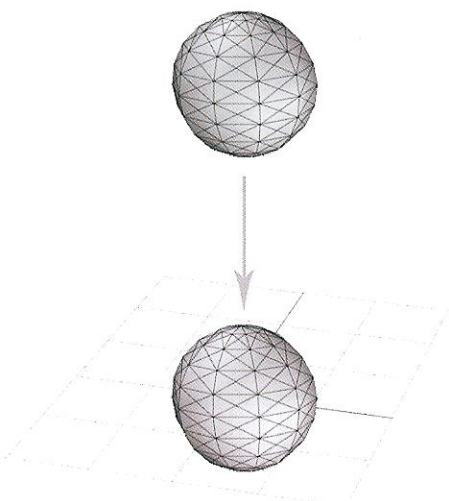
- **Discretization**: The mesh edges and vertices are extracted from a mesh sphere set from Rhino. The points (or particles) are connected to the *Unary Force* component, and the lines (edges and diagonals) are converted into springs. The diagonals are defined by connecting the opposite vertices of each face using the *Mesh Explode* and *Deconstruct Mesh* components.



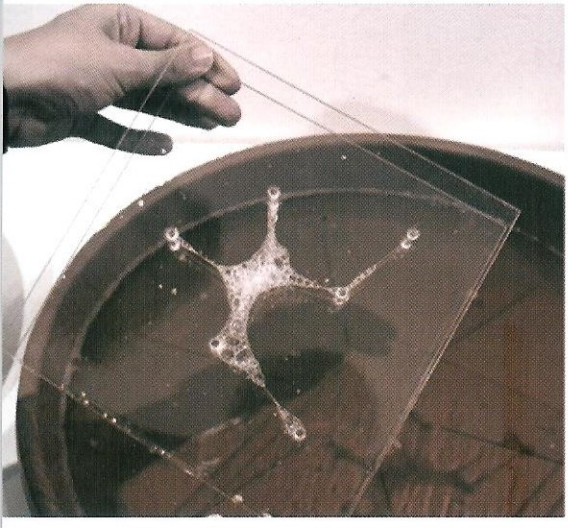
- **Particle-spring system**: The edges and diagonals are merged into a single list using the *Merge* component. Since the mesh-sphere has triangular faces around the poles, the edges and diagonals overlap. This condition would impact the simulation. The component *removeDuplicateLines* (Kangaroo > Utility) removes coincident lines within a tolerance. The output (Q) of the *removeDuplicateLines* component is connected to the inputs Connection-input and Rest length-input of the *Springs* component, with the stiffness set to 5000 units. Lastly, the *Shell* component returns a shell force from the input mesh with an input strength set to 500 and an input angle factor set to 1.

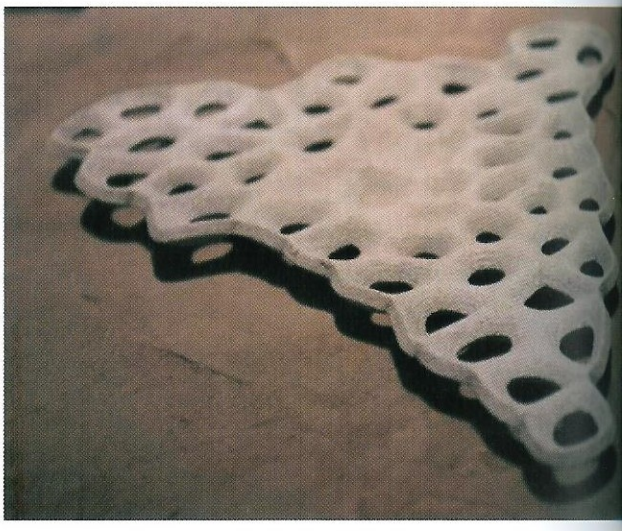






The *Shell* component is useful when simulating materials with bending resistance, such as steel or rubber sheets.





'Synopsis' by Faisal Al Barazi, Michela Falcone, Myrto Grigori and Vittorio Paris, one of the projects of the AA Rome Visiting School "Form As Unknown (X)".

Form as Unknown

Computational Methodology and Material Form Generation in the AA Rome Visiting School Workshops.

Lawrence Friesen, Lorenzo Vianello

"In nature, shape is cheaper than material."

(Lightness. Beukers, Van Hinte)

Material is computational. It is composed of effects and relations, contained in behavior. Each assembly of material system employs a series of characteristics of affections that relate one to another through a system of events to behave in a particular way. The built world around us is composed of relational systems that materially respond to the environment from which architects and designers seek to rationalize and interpret in form. Evolving over centuries, the organization and structure of our built environment is explicit in the urbanized development.

Cities that have resulted in the slow migration and flow of people in response to changes in technology and economies of social organization are generated as a system of relational material events and experiences. The rationalization of these experiences constructs an ordered and predictable form that is recognizable as the built environment; it is generated from a system of economies, and needs which determine the form of the built environment.

> *We live in a world populated by structures - a complex mixture of geological, biological, social, and linguistic constructions that are nothing but accumulations of materials shaped and hardened by history (Manual De Landa. A Thousand Years of Non Linear History. 1997 Zone Books).*

In architecture we are concerned about material that defines space and informs experience. The form of any structure is the result and negotiation of material constraints and the forces that act upon it to produce form. The form making properties are part of the component of behavior embedded in material. Behavior, material and architecture have the expressed relation to predict form, structure and space.

As part of the **AA Rome Visiting School Workshops - Form as Unknown (X) and Form as (Dynamic) Unknown** - the exploration of material through behavior affects the way we see the development of form and structures. Behavior generating form from the relationship of material performance as exemplified in the work most famously of Antoni Gaudì, Frei Otto, Sergio Musmeci and many others, is revealed through the interpretation of material to become a related system of derivation. Working through the experimentation with materials and utilizing the observations and discussions from the famous trio, the Rome workshops sought ways to use this form making methodology to experiment with the translation of a material events to active agents in parametric terms. The material properties themselves are an indication of form making potential, but are indicative in form making and require a translation in scale and properties. The event in form generating becomes a material guide to the understanding a method of interpretation that is material based. The material experimentation indicates a specific effect of shape making that is translatable and predictable based on interpretation. Form generation indicates the clues that can be computed to reveal a way of seeing or understanding the form potential, but through a force generation rather than a purist geometric topology.

Material as Information

Form finding as the method of developing the rational concept of form and structure is predicated on the understanding of material as information. Otto's research in the performance of fabric and soap film materials in the context of the development of the methodology of light weight structures were crafted from observable qualities in natural structures and material behavior. Understanding the natural material organization has bred a new understanding of material for the development of new architectural form. The structures that have been built in response to that exploration are many. The knowledge of these structures has influenced the methods of construction, the use of materials and the way we fabricate. With the more recent advent of parametric design strategies, originally with MAYA and GC and now with Grasshopper, designers have been responsive to the understanding of a new language of form. Although initially providing carefully ordered interpretation of geometries that are artfully formed, they often were void of information that would reflect the making or indeed the environment in which it was placed. They appeared as unbuildable objects that had no relation materially or dynamically to the forces that would engage material in form and structure is often left to the engineer to "work out" the solution independent of form.

The wealth of algorithmic tools that have directly interfaced with the software, in particular Grasshopper for its open sourced approach to development, is now playing a larger role in the development of architecture that is informed from the environment and by building strategies,

there is now a greater potential of interpreting the building information and environmental data in a series of code driven formalizing strategies. If we consider the vastness of data that is collected today from the sources of research and raw data from the number of sensors that register behavior, like mobile phones, personal devices, card data readers and other devices, there is an enormous wealth of data that may influence the way we see and understand cities in which we live, and provide a system of information based support to affect the way make architecture informed by data and responsive to change.

Responsive Design Strategy

Generative design methods and scripting strategies provide architecture the capacity of dealing with large amounts of data and information affecting the composition of urban design and architecture. It also holds a tremendous methodology for the understanding of responsive and mapping capability that informs design to work toward a responsive design strategy.

In traditional forms of design points of analysis are fixed in the development of a design strategy. The information and decisions that make up the process are viewed as a snapshot in time or as an aggregation of moments to a fixed picture while still in the information generation stage of design development. Changes or fluctuations in time are flattened or discarded, subtleties and variation are rationalized, in favor of the generic form.

An algorithmic approach leaves open the fixed point to be variable and responsive, creating a higher order responsive network. As referenced in Autopoiesis of Architecture (Schumacher)

> *The development of computation and scripting techniques was established through a process of development to enrich formal information in the metric diagram. The metric diagram is constrained, and defined by limits without variable quantities. Bringing an order of variability to the metric diagram transforms to a parametric diagram (Schumacher).*

Luigi Moretti first discusses the concept of the parametric diagram in architecture when he adopts the mathematical term Parametric into an architectural lexicon in the 1940's and explores the relational systems that are prevalent in the form of stadia built up of parameters affecting form. Forms become defined by the "forces" acting on them. These forces are observable quantities that help to define a relational system where he alludes to generation of information to transform a formal system. This was of course a highly mathematical approach that was accomplished before the use of computers and software to aid in the data management to control variables, but the form generated is unique and descriptive of the system that produced it.

What happens when we design through a process that utilizes information to affect a system to

become generative? A system that is reflective of the material forces that define form, generates a computational methodology to engage the otherwise 'happy accident' and gives over the form making to the computation of events in space. This methodology employs systems to control the condition and context of an event but allows form to be driven from the event. This performative approach to material form-making, driven from events and forces, offers a further dimension in the direction of computational design strategies. Responsive components and building interfaces may relate the environmental data to reflect the potential of structures changing qualities or transforming uses, allowing for a greater study of effectiveness utilization of spaces to enrich urban spaces.

Limits of Material in Computation

The idea that our world is linked materially in a system of relations is well discussed in Manuel de Landa's book One Thousand Years of Non-Linear History. In computation the effects that drive the forming of architecture is the understanding of data to inform human scale, movement and experience and emotional responses to define an architecture of experience.

The potential of architectural space to be derived through computation techniques through the arrangement and organization of material, offers an opportunity for new language of space to arise. Often though new forms continue to emerge, but are all too close to the original typology from rejecting the influence of material (De Landa). The material computation drives the principles of making and understanding the responses and the structural diagram. Material computation has long been used in engineering and mathematics as models of behavior. Hooke[1] in 1675, used the hanging chain model to decipher the formula for the curvature of a catenary and was used on the unique triple dome structure of St Paul's Cathedral, London and of course Antoni Gaudì with his lengths of strings and lead weights, famously set out the agenda for understanding the way a form will perform in space. Their interpretation of form then translated to the material form. That translation clearly defined by the interaction of material in space. Architecture was once dependent on the understanding of performative design through material to define form.

1. Robert Hooke's hanging chain. Robert Hooke (1635-1703) described the relationship between a hanging chain, which forms a catenary in tension under its own weight, and an arch, which stands in compression.

Form as Unknown / AA Rome Visiting School Workshops
Computation and Making

The premise of the research of the AA Rome Visiting School Workshops was the investigation of computational form through material and performance. Basing the program on Musmeci's definition of form as unknown was part of a larger discussion that outlines that actual form as part of a development through behavior. Indeed computational design strategies are applied not just to structures, but to the order and organization of architecture through the understanding of material and behavior. In the study example of the Basento River Viaduct, in Potenza, the design was envisaged as the result of a stressed skin stretching between support points and the bridge deck. The undulating surface creates a unique spatial quality in the bridge that could be seen as a secondary route. The material exploration that developed this concept was of an elastic surface stretching between points to the degree of force to realize this form. Each of the anchors and fix points represents the force through the surface to find the equilibrium in the form. The final form is translated to a compressive system when it is constructed in concrete. The geometry however is similar and follows the material computation of the material stretched in the model. In this computation approach the form is the unknown of an equation where only the conditions and behavior are given.

Material/Immaterial

Computational material informs all aspects of the development of material formation from environment to behavior. Suspending the idea that a specific material has a particular presence to investigate the relative essence of material in the area of affecting geometry. Deleuze wrote of Spinoza in describing the unique formation of the material world as composed of a single substance delineated by essences and affections (Deleuze on Spinoza). Through this simplicity it is conceived as one substance in many forms. Forms are realized through forces and interactions and ruptures through the substance. This represents an analogy of material in computation. Materials respond to forces and are characteristic by their nature 'pushing' back or resistance to those forces. In these investigations it is conceiving of material not as a specific composition, but by its behavior. A behavior of responses that can be decoded or represented computationally in a model. The decoding informs a process of computationally assembling a strategy in code that follows, mimics, or resembles the material responses.

Computation/Parametric Strategies (Unmaking, Making, Synthesis)

The workshop were set out as an investigation of a material phenomena where the particular responses are recorded and analyzed into the specific event in the material property. The process of analysis is formulated around three categories of investigation:

1) Unmaking or decoding the material phenomena,
2) Making assembling a code of responses which mimic the force lead responses
3) Synthesis, the design of parameters shaping a design response.

Unmaking - investigations of behavior exploring the nature of material in form. In order to understand the system of logic that records the force as a form generator. Unmaking refers to the decoding of a material phenomena, a way of transcribing the observable changes of form from forces acting through the material. In the case of Musmeci and the material based experiments relating to the bridge, the decoding is related to the understanding of the minimal surface geometry and the catenary surfaces which mimic the form of the stretched material. The reference material describes a specific geometry through an event. The event in this case is the forces that pull the material into a state of equilibrium negotiating the forces through the surface geometry. The decoding of the geometric response becomes the tool to investigate responses in a developable model geometry that is translatable.

Making - developing a code that mimics the forces and begins to identify specific form generating systems that will affect a design strategy. Making refers to the computation or parametric strategy in design for understanding the assessment and analysis, and making a proposition based on the observable quality to guide or inform the design process. Investigating the information available to make a proposal defining the constraints of the form generating strategy. In this sense it is interesting to draw a comparison to the mimetic strategies through observation that informed Da Vinci's research in physical phenomena, guiding the design of machines and devices, learned from the environment, observation of forces and defining forms. The design of machines were based on the observation and decoding of the affecting elements. The design proposal utilizes the coding strategy interpreted from the observable elements of the dynamic of form to manipulate form and derive design solutions.

Synthesis - is a translation of the observed to form the parametric and performative components of the design strategy. In shaping the design through information, forces become active elements informing design decisions in a synthesis of form. The design agenda becomes dynamic and responsive to data forces and the potential of shaping of a new architecture. This system is an approach that may employ the same tools that are traditionally the hold of engineers to verify a static design, to one that is used to inform. A system that gives architects and form makers the

power to make architecture that performs and is responsive and is founded on the dynamics of form making. Dynamic analysis informs design strategy where the understanding of forces works interactively to bring the form making potential to designers to inform the making of architecture, and generate a more responsive, efficient, and intelligent building culture.

A Responsive Agenda for Architecture/Urban Form

Recent trends in Architecture have moved away from the understanding of form and performance and the connection to material in design and making. The preoccupation of architects to reference and image design strategy has pushed the study of architecture to the realm of ideological false hood that is removed from material and materiality. An image is not form and is removed from the material sources that inform the design decision. Materiality has to be more than a reference for a mapping code to produce renders in hyper reality. Architects must move beyond the image making and engaging with materiality and the informing of material from our data rich world if it is to be relevant in the world that is reshaping at the rate it is currently. Architecture must respond to the development of technology that informs, and design must respond and be dynamically engaged with the information that shapes our world. In a world of increased urbanization at unprecedented rates, cities will increase at levels unlike ever before. Over the next half century, the world will add approximately one new city of a million every 5 days (Fragkias).

Dynamic responsive technique informing design where forces are informing the shape of our built environment and playing a more active role in the shaping of cities and urban form, offers the potential to design a more appropriate level of urbanization that is informed in design from behavioral and environmental data making changes to form through generated responses. Material changes and perception of spaces and development of new form that is responsive to dynamic forces and social forces that shape our material cities and environment. Not simply buildings but entire cities giving a design agenda that architects must employ a responsive technique to design methods to inform the design and to build responsibly to meet the challenges of the changing natural and urban environment. Through the investigation and analysis of behavior, a performative agenda of real time informing material strategies and responsive environments, architects may gain a skill that is perceptibly lost. As technology in parallel industry grows and techniques that are employed in manufacturing that engage robotics and responsive materials, optimization of form, to create change while responding to changing environmental data, the building design industry must learn from these approaches to use techniques that inform the unknown. The investigation of material and materiality is key to the forming of a behavioral responsive architecture as form making and building.

References:

A. Beukers, E. Van Hinte, 1998, *Lightness - The Inevitable Renaissance of Minimum Energy Structures*, 010 publishers, Rotterdam.

M. De Landa, 1997, *1000 Years of Nonlinear History*, Zone Books, New York.

P. Schumacher, 2010, *The Autopoiesis of Architecture*, vol. 1, John Wiley & Sons Ltd., London.

G. Deleuze, 1990, *Expressionism in Philosophy: Spinoza, trans*, Martin Joughin, Zone Books.

M. Fragkias, 2012, *The Rise and Rise of Urban Expansion*, IGBP Global Change Magazine Issue 78, March 2012.

Web references:

P. Block, M. DeJong, J. Ochsendorf, As Hangs the Flexible Line: Equilibrium of Masonry Arches, http://web.mit.edu/masonry/papers/block_dejong_ochs_NNJ.pdf
Building Technology Program MIT, Cambridge MA 02139 USA.

Lawrence Friesen studied Architecture and Environmental design at Dalhousie University, Canada, and worked at a number of architectural practices before beginning the Design Geometry modeling group at Buro Happold Engineers in London. In the past 15 years he has been involved in a number of complex projects whose innovative realization entailed digital method of building and fabrication. At the GenGeo, architectural structures are designed, optimized structurally and realized through a digital design process to engage with simulation software. Forms are translated into a material design process that is true to the intent of the design but have a spatial and material that enhances the design realization.
Formerly he was a studio technical tutor at the AA graduate masters program Design Research Lab (DRL) and technical tutor to AA intermediate Unit 10. Currently teaching at the AA Visiting Schools in Rome, Italy, and is Associate Lecturer Unit lead at Oxford Brookes University, he is also a practice principle for GenGeo consulting for architects and designers and is a design lead for design projects.

Lorenzo Vianello graduated in architecture in Italy in 2005. From 2005 to 2009 and from 2011 to 2013 he collaborated with several firms, including OMA, Studio Fuksas, UNStudio and Foster+Partners. From 2009 to 2011 he deepened his research, attending the Design Research Laboratory program at the Architectural Association. Here he was guided by tutor Patrik Schumacher, partner at Zaha Hadid Architects, during the development of his final thesis. The theme of the thesis was a proto-tower, system of algorithms that generates projects of high-rises with optimized properties for different environments and functional programs. Lorenzo currently is based in London and works as a Design Tutor at the Oxford Brookes University, as an AA Rome Visiting School program co-director at the Architectural Association and as a freelance architect.

The Protohouse project was developed by Softkill Design in the Architectural Association School's Design Research Lab within the 'behavioral matter' studio of Robert Stuart-Smith. Image courtesy of Softkill Design. http://www.softkilldesign.com/.