

# **FDMremote User Guide**

Version 0.1

Keith J. Lee & Adam Burke

January 11, 2023

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Installing FDMremote_GH	3
2.2	Installing FDMremote.jl	3
<b>3</b>	<b>Design Workflow</b>	<b>5</b>
3.1	General notes	5
3.2	A simple network	5
3.3	Control of Q and P	5
3.4	Remote Analysis	5
3.5	Direct Manipulation	5
3.6	Optimization	5
3.7	Advanced	5
<b>4</b>	<b>Component Overview</b>	<b>6</b>
4.1	Design	6
4.1.1	CreateNetwork (Create)	6
4.1.2	QByLayer (LayerQ)	7
4.1.3	AnalyzeSimpleNetwork (AnalyzeSimple)	8
4.2	Utility	8
4.2.1	NetworkInformation (Info)	8
4.2.2	Visualize Network (Visualize)	9
4.2.3	Curve to Curve (CurvePairs)	10
4.2.4	Element Tagger (Tagger)	11
4.2.5	Update Geometry (Update)	11
4.2.6	Freeze Network (Freeze)	12
4.2.7	Load Network (Load)	13
4.2.8	Baker (Baker)	13
4.3	Optimization	14
4.3.1	Remote Start (FDMstart)	14
4.3.2	Remote Send (FDMsend)	15
4.3.3	Remote Listen (FDMlisten)	15
4.3.4	Optimization Parameters (Params)	16
4.4	Objective Functions	17
4.4.1	Target (OBJTarget)	17
4.4.2	StructuralPerformance (OBJFL)	18
4.4.3	ForceVariation (OBJForce)	18
4.4.4	LengthVariation (OBJLength)	19
4.4.5	MinimumForce (OBJMinForce)	19

4.4.6	MaximumForce (OBJMaxForce)	20
4.4.7	MinimumLength (OBJMinLength)	20
4.4.8	MaximumLength (OBJMaxLength)	21
4.5	Experimental	22
4.5.1	Edge Control Surface (CtrlSurfQ)	22
4.5.2	Point Control Surface (CtrlSurfP)	23
4.5.3	Make Helper (Make)	24

# 1 Introduction

This is a brief guide to using [FDMremote](#), a twin package consisting of a high-performing Julia solver back-end and a front-facing set of native Grasshopper components for the design and analysis of Tension or Compression networks using the [Force Density Method \(FDM\)](#). The Julia-Grasshopper connection is made possible through modification of the WebSocket components in the [Bengesht](#) plugin by Behrooz Tahanzadeh.

This plugin was developed through the Digital Structures research group at the Massachusetts Institute of Technology by Keith J. Lee and Adam Burke.

## 2 Installation

### 2.1 Installing FDMremote\_GH

*FDMremote has only been tested on Windows computers + Rhino 7.*

1. Download the Grasshopper plugin [here](#)
2. As always, right click > Properties > Unblock the .zip file before extraction.
3. Extract the .zip file into your Grasshopper components folder. You can find this folder by opening Grasshopper and: File > Special Folders > Components Folder.
4. Open Grasshopper and verify that you have a "FDMremote" tab on the top ribbon.

You can now start designing and analyzing simple networks!

### 2.2 Installing FDMremote.jl

To achieve much faster analysis, design more complex networks, and perform optimization, you must install FDMremote.jl.

1. Download Julia [here](#). For convenience, add Julia to your PATH when prompted.
2. Open Julia (Windows + Julia to search for it in your start menu). This terminal looking thing is the Julia REPL (Read-Evaluate-Print Loop). It's a way of interfacing with the Julia engine.

3. Enter Package Mode by typing `]` in the Julia REPL.
4. Add FDMremote.jl by typing:  
add `https://github.com/keithjlee/FDMremote` and pressing enter.

You've now unlocked the full potential of FDMremote! See the next section to get started.

## 3 Design Workflow

Refer to [Component Overview](#) for detailed information on each component.

### 3.1 General notes

1. The global XY plane is the ground plane of the network: many utility components will not work as expected if assumed otherwise.
2. Network edges are topological only: curved edges will be redefined as straight lines between start and end nodes
3. Turn on object snapping to ensure that curves are connected to each other, and nodes are not floating in space. If importing geometry, try adjusting the Tolerance input when creating a network if errors occur.
4. FDMremote is **unit agnostic**. It is up to you to ensure that values of force density and external loads are consistent with the geometric units used on the Rhino canvas.
5. Many utility components (Bake, Maker, CtrlSurfQ, CtrlSurfP) are based on bounding boxes aligned to the global axes. They will work best when your drawn network is somewhat aligned with these axes. I.e., try and make sure the primary span of your input network is aligned with either the global X or Y axis. If one span is significantly longer than the other, align this span with the global X axis.

The examples below can be downloaded [here](#). A general architecture diagram is provided in each section: **black** arrows and borders define mandatory component interactions, **gray** arrows define optional (but useful) interactions.

### 3.2 A simple network

### 3.3 Control of Q and P

### 3.4 Remote Analysis

### 3.5 Direct Manipulation

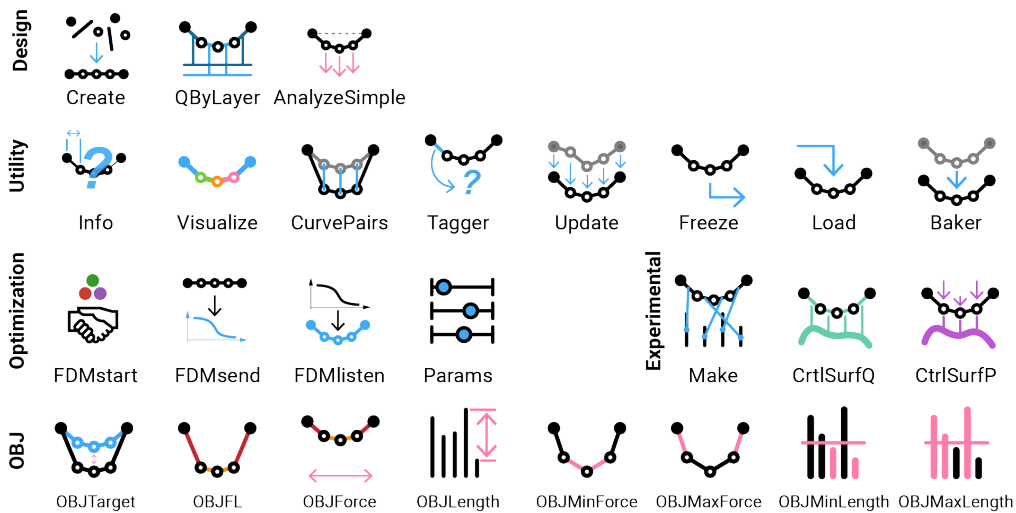
### 3.6 Optimization

### 3.7 Advanced

## 4 Component Overview

Components are split into 5 main categories:

1. **Design:** Network generation and simple analysis
2. **Utility:** Visualization, information, and I/O
3. **Optimization:** Remote connection to Julia server
4. **OBJ:** Composable objective functions for optimization
5. **Experimental:** Utility components in Beta



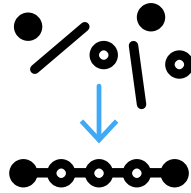
### 4.1 Design

These components form the core functionality of FDMremote, and is used to defined a Network object.

#### 4.1.1 CreateNetwork (Create)

This component generates a Network object. Inputs:

- **Curves (C):** list of curves that define the edges of the network. Only the start and end points are considered: curved lines will be converted into straight edges.

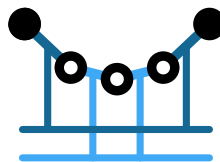


- **Anchors** (A): list of points that define the fixed nodes of the network. A minimum of 3 is required.
- **Force Density** (q): either a single value [ $force/length$ ] to apply to all edges or a list of values (of equal length to C) for individual force densities. Default: 1.0
- **Tolerance** (tol): snapping tolerance when calculating topology of network. This is the search sphere radius at the ends of each edge to determine which edges have shared nodes, and which nodes are defined as anchors. Default: 0.1

Output: a **Network** object.

#### 4.1.2 QByLayer (LayerQ)

Assign specific force densities for edges in specific layers.



Inputs:

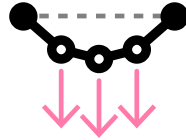
- **EdgeGUIDs** (GUIDs): GUIDs of all input network edges.
- **LayerNames** (Layers): Names of layer(s) of edge groups. Default: "Default"
- **ForceDensities** (Q): Force density to assign to edges in layer. **length of Layers must equal length of Q**. Default: 1.0
- **DefaultQ** (Qdefault): Force density to assign to unspecified edges. Default: 1.0

Output: a list of force densities (Q) in order of input edges.



### 4.1.3 AnalyzeSimpleNetwork (AnalyzeSimple)

This component performs a native analysis of an FDM network.



Inputs:

- **Network** (Network): network object to be analyzed
- **Load** (P): either a single vector value to apply to all free nodes, or a list of vectors (of equal length to all free nodes) for individual application.  
Default: [0,0,0]

Output: a new **Network** object of the solved equilibrium positions.

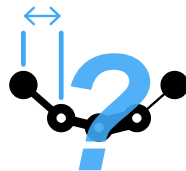
**WARNING:** the native solver relies on suboptimal linear algebra libraries, and performance will drastically decrease when networks exceed ~300 elements. Use remote calling for dense networks.

## 4.2 Utility

These components allow for the visualization, querying, export/import of networks, etc.

### 4.2.1 NetworkInformation (Info)

Pretty much everything you need to know about a network.



Input: **Network**

Outputs:

- **Nodes** (N): all nodes in network (fixed/free)
- **Edges** (E): all edges in network
- **Edge Lengths** (L): lengths of edges
- **Start Index** (iStart): indices of starting node in (N) for each edge in (E)
- **End Index** (iEnd): indices of ending node in (N) for each edge in (E)
- **Anchors** (A): all anchor points in network
- **Force Densities** (q): force densities of all edges
- **Number Edges** (Ne): total number of edges
- **Number Nodes** (Nn): total number of nodes
- **Free Indices** (iN): indices of nodes in (N) that are free points
- **Fixed Indices** (iF): indices of nodes in (N) that are anchor points
- **Member forces** (Force): internal force of all edges in network ( $q \times L$ ).  
**WARNING:** this is a naive calculation that assumes the input network has already been analyzed, i.e. the edges are at a stressed state.
- **Reaction Forces** (Reactions): forces acting at anchors. Same warning as above.

#### 4.2.2 Visualize Network (Visualize)

Visualize a network.



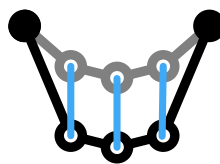
Inputs:

- **Show** (Show): visualize the network. Default: true
- **Network** (Network): network to visualize
- **Loads** (P): either a single vector or list of vectors applied to network. All loads are normalized to the largest load before visualization. Default: [0,0,0]
- **Load Scale** (Pscale): length of (P) arrows after normalization. Default: 1.0

- **Minimum Colour** (Cmin): minimum color for graded edges. Default: pink
- **Neutral Colour** (Cmed): neutral color for graded edges. Default: light gray
- **Maximum Colour** (Cmax): maximum color for graded edges. Default: blue
- **Property** (Property): edge property to colour. Right click to access options:
  - Force: default value
  - None: no gradation; all edges will be coloured by (Cmax)
  - Q: coloured by edge force densities
- **Line Thickness** (Thickness): thickness of edge lines. Default: 2
- **Load Colour** (CLoad): colour of applied loads. Default: red
- **Show Loads** (Load): display loads. Default: true
- **Reaction Colour** (Creaction): colour of reaction forces. Default: green
- **Show Reactions** (Reaction): display reactions. Default: false

#### 4.2.3 Curve to Curve (CurvePairs)

Sometimes the solved network looks nothing like the initially drawn input network. This component helps visualize which edge has gone where.

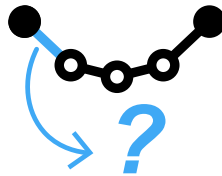


Inputs:

- **Initial Network** (Network1): the initial network
- **Solved Network** (Network2): the solved network
- **Line Colour** (Colour): colour of connecting lines. Default: gray
- **Line Weight** (Weight): thickness of lines. Default: 2
- **Show** (Show): show lines. Default: true

#### 4.2.4 Element Tagger (Tagger)

This displays element-wise information (force density, length, force) as text tags centered at each edge.

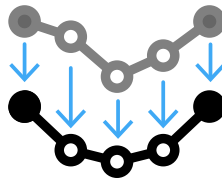


Inputs:

- **Network** (Network): network to tag
- **Show** (Show): show tags. Default: true
- **Tag size** (Size): size of tags. Default: 2
- **Tag colour** (Colour): colour of tags. Default: black

#### 4.2.5 Update Geometry (Update)

Update the drawn curves to reflect a new equilibrium position. This allows for iterative solving of networks. **NOTE:** this only works if input curves were drawn in Rhino.



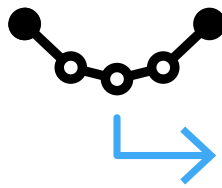
Inputs:

- **Update Curves** (Update): update Rhino curves to new positions. Attach a boolean button here. Default: false. **WARNING:** depending on when this component is triggered, it may be a destructive process. IE ctrl+Z at your own risk. Use the FreezeNetwork component to provide a safeguard.
- **Target Network** (Network): the target network to base curve updates.

- **Edge IDs** (GUIDs): the GUIDs of the input curves to update. These can be extracted using the **GUID** component in Grasshopper. This component works by matching the GUIDs of drawn geometry (in Rhino) with the GUIDs of the network edges.

#### 4.2.6 Freeze Network (Freeze)

Store a hard copy of the current network.



Inputs:

- **Freeze** (Freeze): freeze the network. Attach a boolean button here.
- **Network** (Network): network to freeze
- **Loads** (P): load vectors to freeze
- **Save Directory** (Dir): folder to save data. Must be written with double backslashes.
- **File Name** (Name): name of file to save. Must end in .json. Default: `network.json`
- **Save Offset** (Offset): offset of geometry (curves and points) that is frozen. The default offset is 1.5 times the overall bounding box width in the positive X direction.

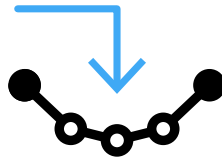
Outputs:

- **Edges** (E): frozen edges
- **Anchors** (A): frozen anchors
- **Force Densities** (q): frozen force densities
- **Network** (Network): frozen network

The output reflects the input network at the last time the **Freeze** button was triggered. Input values can be deleted and the component will retain outputs. In worst-case scenarios, bake the geometry or load the resulting file to restart.

#### 4.2.7 Load Network (Load)

Load a saved network .json.



Inputs:

- **Folder** (Dir): folder containing file. Must be written with double backslashes.
- **File Name** (Name): name of file to load. Must end in .json

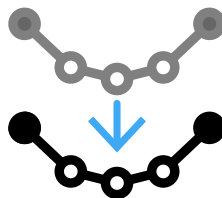
Outputs:

- **Curves** (Curves): curves that define network
- **Anchors** (Anchors): points that define anchors
- **Force Densities** (q): force density values
- **Loads** (P): vectors that define loads applied to saved network

Outputs can directly be connected to a new **Create Network** component.

#### 4.2.8 Baker (Baker)

Bake the geometry of a network, with edges optionally coloured with respect to a given network property. Use this for exporting final geometries for editing/visualization.



Inputs:

- **Network** (Network): Network to bake.

- **BakeLayer** (Layer): Layer to bake geometry. Default: "Layer 05"
- For parameters below, see [Visualize Network \(Visualize\)](#):
  - Cmin
  - Cmed
  - Cmax
  - Property
- **Bake** (Bake): Bake the geometry with above parameters. Default: false
- **BakeOffset** (Offset): Additional offset of baked geometry from reference network; default is set to 1.5x the overall bounding box width in the negative X direction. Default: [0,0,0]

Outputs:

- **Curves** (Curves): Curves of network (*not offset*)
- **Anchors** (Anchors): Anchors of network (*not offset*)

## 4.3 Optimization

These components form the core of FDMremote, and requires installation of FDMremote.jl.

### 4.3.1 Remote Start (FDMstart)

Initializes the connection to the Julia server and generates a WebSocket object.



Inputs:

- **Reset** (Rst): Reset the connection to the server. In most cases, this input should not be necessary. Connect a Boolean button. Default: false
- **Host** (Host): Host address of server. Default: 127.0.0.1
- **Port** (Port): Port address of server. Default: 2000

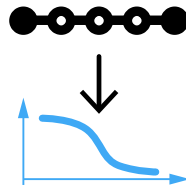
Outputs:

- **WebSocket object** (WSC): WebSocket object that communicates with Julia server

It is recommended to start the Julia server by running `FDMsolve!()` before using this component.

#### 4.3.2 Remote Send (FDMsend)

Send network information to the Julia server.



Inputs:

- **WebSocket object** (WSC): WebSocket object; attach the output of **FDMstart**
- **Network** (Network): Network to analyze
- **Optimization Parameters** (Params): Parameters for analysis/optimization. Default: null parameter for a single solve
- **Load** (P): Applied loads to analyze. Default: [0,0,0]
- **Close** (Close): Send a 'CLOSE' message to sever connection. Default: false

Outputs:

- **Network** (Network): a copy of the input network
- **Message** (Msg): A copy of the raw message sent to server

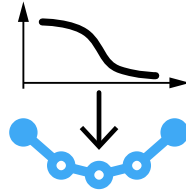
#### 4.3.3 Remote Listen (FDMlisten)

Receive and parse messages sent back by server.

Inputs:

- **WebSocket object** (WSC): WebSocket object; attach the output of **FDMstart**





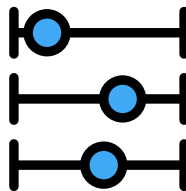
- **Auto Update** (Upd): Automatically update output values for each new message. Default: true
- **Network** (Network): Copy of initial network sent to server. Connect to output of **FDMsend**

Outputs:

- **Data** (Data): Raw message received from server
- **Network Status** (Sts): Status of current client-server connection
- **Optimization Finished** (Finished): Status of optimization/solving routine
- **Force densities** (q): Solved force density values
- **Loss** ( $f(q)$ ): Current objective function value (for optimization)
- **Iteration** (Iter): Current iteration of optimization routine
- **Loss Trace** ( $f(q(t))$ ): History of objective function value to date
- **Solve Network** (Network): Solved network from optimization

#### 4.3.4 Optimization Parameters (Params)

Parameters for optimization.



Inputs:

- **Objective Functions** (Obj): One or more objective functions. Default: Null Objective

- **Lower Bound (LB)**: Lower bound of q values for optimization. Either a single value or list of values. Default: 0.1
- **Upper Bound (UB)**: Upper bound of q values for optimization. Either a single value or list of values. Default: 100
- **Absolute Tolerance (AbsTol)**: Absolute stopping tolerance. Default: 1e-3
- **Relative Tolerance (RelTol)**: Relative stopping tolerance (currently not in use).
- **Maximum Iterations (MaxIter)**: Maximum iterations during optimization. Default: 400
- **Update Frequency (Frequency)**: Frequency intermediate messages sent back to client. Default: 20
- **Show Iterations (ShowIter)**: Send back intermediate messages (primarily for visualization). Default: true

Outputs:

- **Optimization Parameters (Params)**: consolidated parameters. Connect to **FDMsend**

## 4.4 Objective Functions

A suite of composable objective functions for optimization. The decision variables are (currently) all edge force densities. All objective functions have a numeric weight that can be adjusted to control relative importance. These weights are *always* relative.

### 4.4.1 Target (OBJTarget)

labelOBJTarget Match the target shape. Equivalent to minimizing the distance a node travels from its drawn position to the equilibrium position.

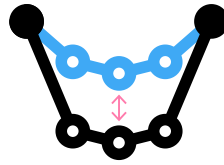
$$OBJ = \|\vec{XYZ}_0 - \vec{XYZ}_f\|$$

Inputs:

- **Weight (W)**: Objective weight. Default: 1.0

Outputs:

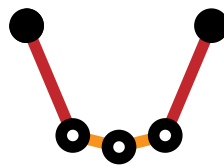
- **Objective (OBJ)**: Objective function



#### 4.4.2 StructuralPerformance (OBJFL)

Proxy for structural performance.

$$OBJ = \sum_1^{N_e} |F_i| L_i$$



Inputs:

- **Weight** (W): Objective weight. Default: 1.0

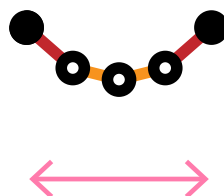
Outputs:

- **Objective** (OBJ): Objective function

#### 4.4.3 ForceVariation (OBJForce)

Minimize the difference between the largest and smallest forces.

$$OBJ = \max(F) - \min(F)$$



Inputs:

- **Weight** (W): Objective weight. Default: 1.0

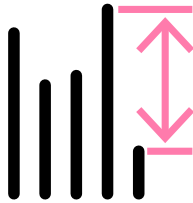
Outputs:

- **Objective** (OBJ): Objective function

#### 4.4.4 LengthVariation (OBJLength)

Minimize the difference between the largest and smallest stressed lengths.

$$OBJ = \max(L) - \min(L)$$



Inputs:

- **Weight** (W): Objective weight. Default: 1.0

Outputs:

- **Objective** (OBJ): Objective function

#### 4.4.5 MinimumForce (OBJMinForce)

Penalize all internal forces less than a given value, proportional to the distance away from threshold.

$$\vec{t} = [v - F_i] \quad \forall i$$

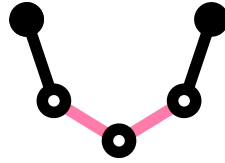
$$OBJ = \sum t_i + |t_i|$$

Inputs:

- **Weight** (W): Objective weight. Default: 1.0
- **Minimum Force** (Force): Threshold force. Default: 1.0

Outputs:

- **Objective** (OBJ): Objective function

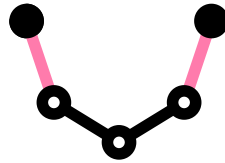


#### 4.4.6 MaximumForce (OBJMaxForce)

Penalize all internal forces less than a given value, proportional to the distance away from threshold.

$$\vec{t} = [F_i - v] \forall i$$

$$OBJ = \sum t_i + |t_i|$$



Inputs:

- **Weight (W)**: Objective weight. Default: 1.0
- **Maximum Force (Force)**: Threshold force. Default: 10000.0

Outputs:

- **Objective (OBJ)**: Objective function

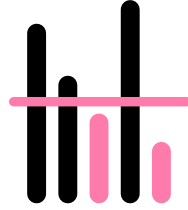
#### 4.4.7 MinimumLength (OBJMinLength)

Penalize all stressed lengths less than a given value, proportional to the distance away from threshold.

$$\vec{t} = [v - F_i] \forall i$$

$$OBJ = \sum t_i + |t_i|$$

Inputs:



- **Weight (W)**: Objective weight. Default: 1.0
- **Minimum Length (Length)**: Threshold length. Default: 1.0

Outputs:

- **Objective (OBJ)**: Objective function

#### 4.4.8 MaximumLength (OBJMaxLength)

Penalize all stressed lengths less than a given value, proportional to the distance away from threshold.

$$\vec{t} = [F_i - v] \quad \forall i$$

$$OBJ = \sum t_i + |t_i|$$



Inputs:

- **Weight (W)**: Objective weight. Default: 1.0
- **Maximum Length (Length)**: Threshold length. Default: 1000

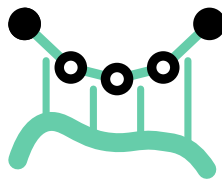
Outputs:

- **Objective (OBJ)**: Objective function

## 4.5 Experimental

### 4.5.1 Edge Control Surface (CtrlSurfQ)

Generates a control NURBs surface that assigns element-wise force density values in a controllable manner. Normalized control sliders are automatically generated.



Inputs:

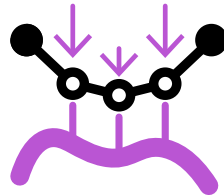
- **Generate** (Generate): Generate the control surface and sliders. Default: false
- **Curve** (Curve): Network edges
- **Ucount** (nU): Control point density in direction 1. Default = 3
- **Vcount** (nV): Control point density in direction 2. Default = 3
- **Surface Offset** (Offset): Offset of displayed control surface. Default: [0, 0, -300]
- **Maximum Value** (Max): Value associated with highest point on control surface. Default: 100
- **Minimum Value** (Min): Value associated with highest point on control surface. Default: 0.1
- **Control Value** (Value): Z-heights of surface control points. **NOTE: these will be auto-generated by Generate**
- **Show Surface** (Show): Show the control surface. Default: true
- **Text Scale** (TextScale): Scale of text tags on control points. Default: 20

Outputs:

- **Surface** (Surface): The control surface
- **Values** (Vals): The values of the control surface in order of network edges. Input into the **CreateNetwork** component for q values.

#### 4.5.2 Point Control Surface (CtrlSurfP)

Generates a control NURBs surface that assigns free point-wise force magnitudes in a controllable manner. Normalized control sliders are automatically generated.



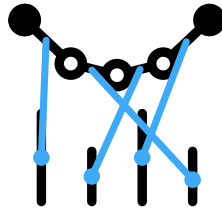
Inputs:

- **Generate** (Generate): Generate the control surface and sliders. Default: false
- **Network** (Network): Network to assign forces to
- **Ucount** (nU): Control point density in direction 1. Default = 3
- **Vcount** (nV): Control point density in direction 2. Default = 3
- **Surface Offset** (Offset): Offset of displayed control surface. Default: [0, 0, -300]
- **Maximum Value** (Max): Value associated with highest point on control surface. Default: 100
- **Minimum Value** (Min): Value associated with highest point on control surface. Default: 0.1
- **Control Value** (Value): Z-heights of surface control points. **NOTE: these will be auto-generated by Generate**
- **Show Surface** (Show): Show the control surface. Default: true
- **Text Scale** (TextScale): Scale of text tags on control points. Default: 20

Outputs:

- **Surface** (Surface): The control surface
- **Values** (Vals): The values of the control surface in order of free network nodes. Use these values to scale vectors to input into analysis.





### 4.5.3 Make Helper (Make)

A utility component that aids the materialization of networks.

Inputs:

- **Show** (Show): Show visualizations. Default: true
- **Network** (Network): Network to analyze.
- **Material Stiffness** (E): Young's modulus of edge. Default: 100
- **Area** (A): Cross sectional area of edge. Default: 10
- **Show Nodes** (Nodes): Show nodes as enlarged points. Default: false
- **Node Radius** (Radius): Radius of node points. Default: 5
- **Colour Gradients** (F0, F1, N0, N1): Colour gradients to distinguish between fixed and free nodes.
- **Edge Thickness** (Thickness): Thickness of edges. Default: 5
- **Straight Spacing** (Spacing): Spacing between edges in aligned row visualization. Default: 10
- **Show Straight** (Straight): Show aligned edges. Default: true
- **Show Projection** (Projection): Show 2D projection. Default: true
- **Show Pairs** (Pairs): Show edge-edge pairs between network and visualizations. Default: false
- **PairColour** (Cpair): Colour of edge-edge pair lines. Default: gray
- **Text Size** (TextSize): Size of text tags. Default: 3
- **Show Text** (Text): Show text tags. Default: true
- **Text Colour** (Ctext): Colour of text tags. Default: black
- **Straight Offset** (StraightOffset): Offset of aligned edge visualization.
- **Projection Offset** (ProjOffset): Offset of 2D projection visualization.
- **Text Offset** (TextOffset): Offset of text tags in Z direction.
- **Pair Indexer** (Indexer): extracts element-wise assembly information given an integer index.