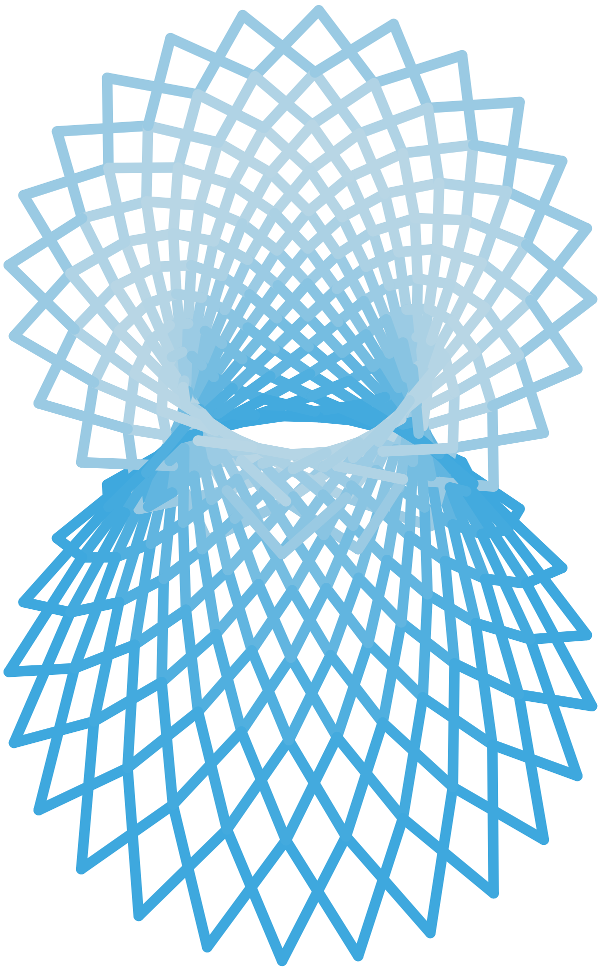


FDMremote User Guide

Version 0.1

Keith J. Lee & Adam Burke

January 12, 2023

Contents

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.

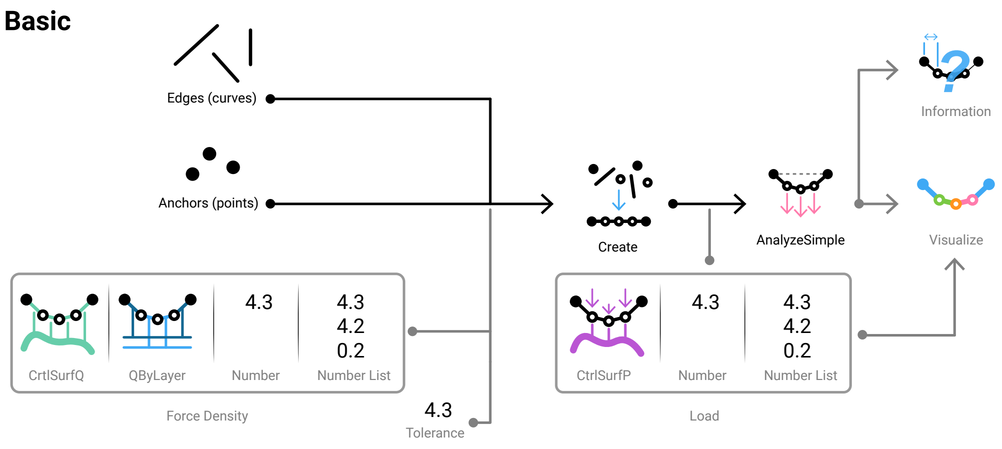
3.1 General notes

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** defines mandatory component interactions, **gray** defines optional interfaces (e.g., a default value is provided).

3.2 Basic workflow

The most basic workflow for the design and analysis requires two components: CreateNetwork (Create) and AnalyzeSimpleNetwork (AnalyzeSimple).



Create takes a collection of points to define anchors (fixed points) of a network and a collection of curves (edges). The position and indices of free nodes are automatically calculated based on the start and end points of the input curves. `Tolerance` defines the search radius at the end points of each curve to determine whether an end point is an anchor, and which end points are shared among edges. You can adjust this if your input geometry is less precise.

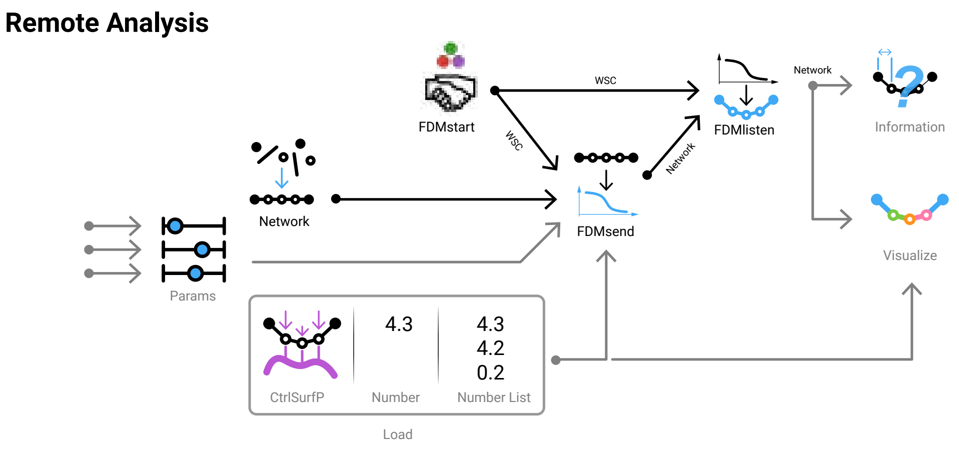
A default force density (q) value of 1 is prescribed to all edges. You can either provide a single numeric override value, or provide a list of values in order of the input edges and of the same length. A more natural control method can be performed using Edge Control Surface (CtrlSurfQ) and QByLayer (LayerQ).

Analyze takes a network object and returns a new network with equilibrium node positions. It takes an optional external load value applied to the free nodes. This can be defined by a single vector for a consistent load, or a list of vectors in order of each free node. A more natural control method can be performed using Point Control Surface (CtrlSurfP).

See `0_simplenetwork.gh` and `1_controlsurfaces.gh` for examples of basic network analysis.

3.3 Remote workflow

In example files 0 and 1, there are sliders defining the density of the initial grid network. Try adjusting the lower bounds to a larger number (say, 15), and observe the decrease in performance and response for these larger networks. Adjusting anchor position sliders is no longer fluid. This is where the *remote* part of FDMremote comes in, for rapid analysis and real-time response for very dense networks. The following diagram shows the general workflow for remote analysis.



AnalyzeSimple is replaced by a series of three components: Remote Start (FDMstart), Remote Send (FDMsend), and Remote Listen (FDMlisten). The output **WSC** in **FDMstart** is the main object that communicates with the Julia server. Ideally, initialize the Julia server first by opening a Julia instance and running the following commands:

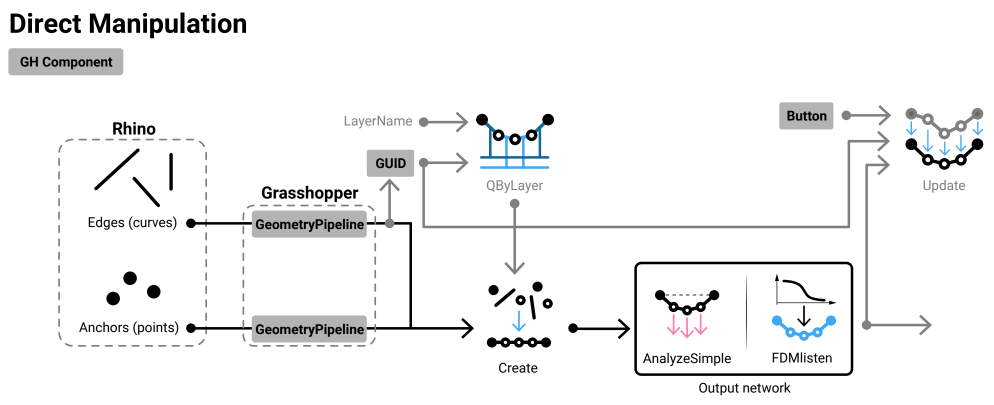
1. `using FDMremote`
2. `FDMsolve!()`

Then, replace **AnalyzeSimple** with the three **FDM** components, and use normally. The first solve will take a while as the Julia server compiles; you should notice a significant performance improvement for subsequent analyses. Try making the density of the grid in Example 1 significantly larger.

See `2_remoteanalysis`.gh for an example.

3.4 Direct Manipulation

Sometimes we want to directly interact with the drawn network: freely move nodes around, organize edges with layers, and export rendered geometry. You can do this by creating a translation layer between Rhino and Grasshopper using the built in **GeometryPipeline** component. Create two separate pipelines: one for edges and one for anchors.

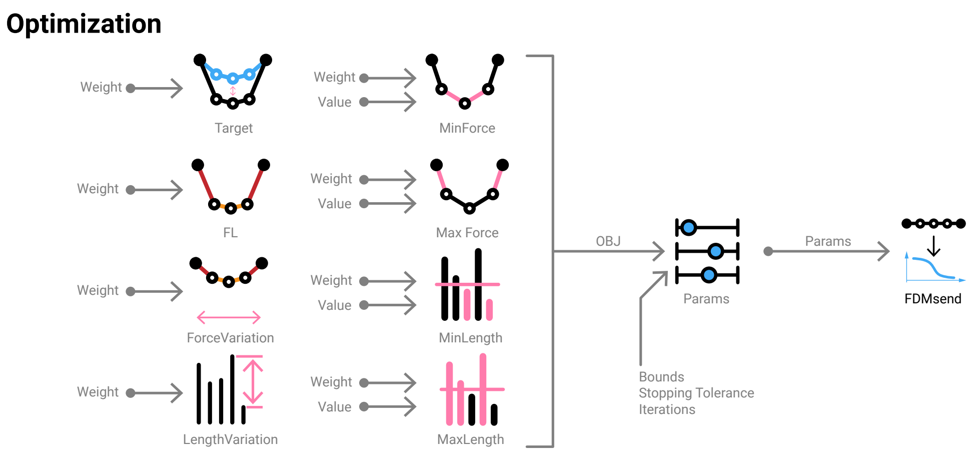


The use of pipelines now allows you to modify the input geometry directly and observe the resulting changes in the FDM analysis. It also enables two useful components: QByLayer (LayerQ) and Update Geometry (Update). These components make use of the GUIDs (Globally Unique Identifier) associated with the drawn curves. Extract the GUIDs through the **GUID** component. **QByLayer** allows for a single force density value to be assigned to all edges in a given layer; **Update** updates the geometry in Rhino to the solved network geometry (either from **AnalyzeSimple** or **FDMlisten**).

See `3_directmanipulation.gh` for an example.

3.5 Optimization parameters

When using remote analysis, the optional input **Params** define the parameters for optimization. The input **OBJ** takes any combination of the (currently) 8 possible objective functions. All objective functions take an associated relative numeric weight; some objective functions also take in a numeric threshold value for minimum/maximum objectives.



The other inputs are self-explanatory: stopping tolerance for optimization, maximum number of iterations, whether or not to send back intermediate results from the server, etc. Couple of notes:

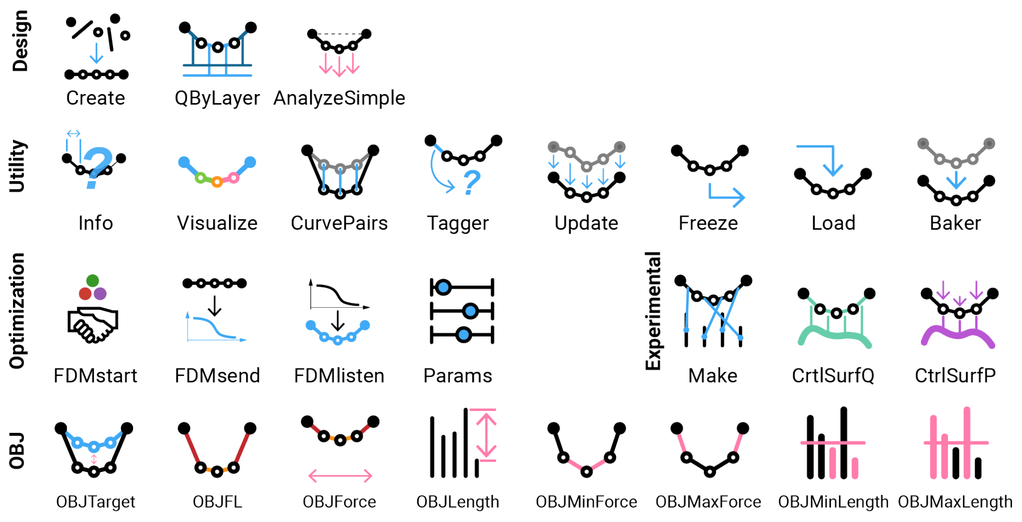
1. For compression networks, make sure you adjust the lower bound **LB** to a negative value. The default values for both **LB** and **UB** are positive.
2. Check the Julia REPL for information of the current problem: optimization provides iteration count information and when the optimizer is finished; if you receive an “INVALID INPUT" message, check the bounds parameters.
3. The first time using a new objective function will have some delay; subsequent uses will be very fast.

See `4_optimization.gh`, `5_advanced.gh`, and `5_advanced.3dm` for examples.

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



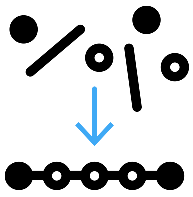
4.1 Design

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.

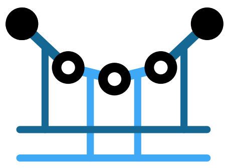


- **Anchors** (A): list of points that define the fixed nodes of the network. A minimum of 3 is required.
- **Force Density** (q): either a single value [*force*/*length*] to apply to all edges or a list of values (of equal length to C) for individual force densities. Default: 1.0
- **Tolerance** (tol): 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.



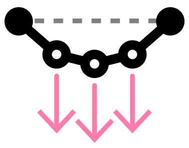
Inputs:

- **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.

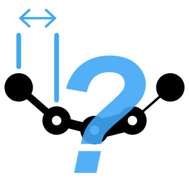
WARNING: the native solver relies on suboptimal linear algebra libraries, and performance will drastically decrease when networks exceed ∼300 elements. Use remote calling for dense networks.

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.



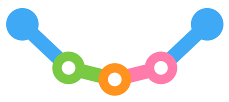
Input: **Network**

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:

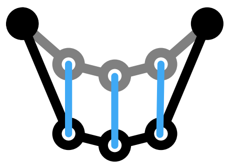
  Force: default value

  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.

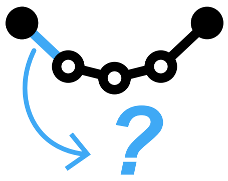


Inputs:

- **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.

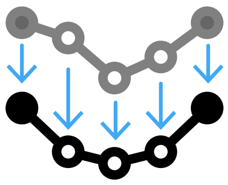


Inputs:

- **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.



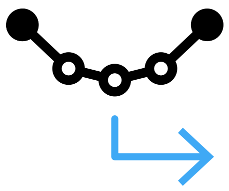
Inputs:

- **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

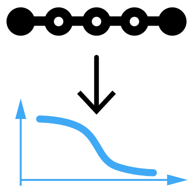
Outputs:

- **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.

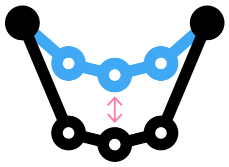
$$OBJ = \|X\vec{Y}Z_0 - X\vec{Y}Z_f\|$$

Inputs:

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

Outputs:

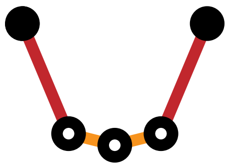
- **Objective** (OBJ): Objective function



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

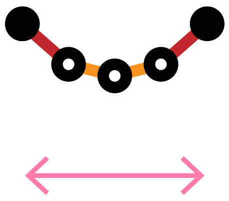
Outputs:

- **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

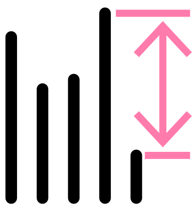
Outputs:

- **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.

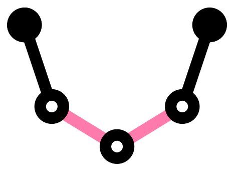
$$\vec{t} = [v - F_i] \ \forall \ i$$
$$OBJ = \sum t_i + |t_i|$$

Inputs:

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

Outputs:

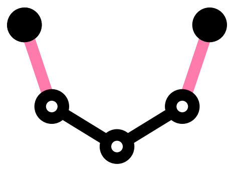
- **Objective** (OBJ): Objective function



4.4.6 MaximumForce (OBJMaxForce)

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

$$
\begin{aligned}
\vec{t} &= [F_i - v] \;\; \forall \, i \\
OBJ &= \sum t_i + |t_i|
\end{aligned}
$$



Inputs:

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

Outputs:

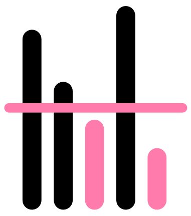
- **Objective** (OBJ): Objective function

4.4.7 MinimumLength (OBJMinLength)

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

$$
\begin{aligned}
\vec{t} &= [v - F_i] \;\; \forall \, i \\
OBJ &= \sum t_i + |t_i|
\end{aligned}
$$

Inputs:



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

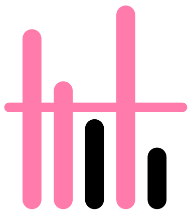
Outputs:

- **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.

$$\vec{t} = [F_i - v] \ \forall \ i$$
$$OBJ = \sum t_i + |t_i|$$



Inputs:

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

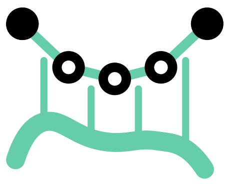
Outputs:

- **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)

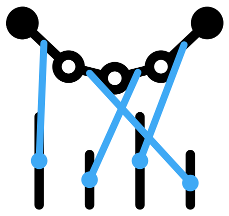
Generates a control NURBs surface that assigns free point-wise force magnitudes in a controllable manner. Normalized control sliders are automatically generated.

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.