The MMJMesh Library

Matthias Baitsch

Table of contents

1 Introduction **3**
1.1 Demo for 1D meshes 3

2 The Mesh interface **10**

3 Plotting **12**
3.1 Plot a 1D mesh 12
3.1.1 On a straight line 12
3.1.2 Vertical 15
3.1.3 On a spiral 16
3.2 Plot a 2D mesh 17
3.2.1 Quad mesh 17
3.2.2 Triangle mesh 20

1 Introduction

The `MMJMesh` library is designed to be a simple yet versatile basis for the implementation of finite element methods or postprocessing tools.

1.1 Demo for 1D meshes

First, we need to load the required modules. Note that `CairoMakie` included by `import` and not by `using` in order to avoid name collissions.

```
import CairoMakie
using MMJMesh.Plots
using MMJMesh.Meshes
using MMJMesh.Utilities
```

In the simplest case, a 1D mesh is defined by parameter bounds and the number of elements.

```
m = makemeshoninterval(0.0, 8.0, 4)
plot(m)
```

Elements of the mesh can be easily processed in a loop:

```
for e  elements(m)
    println(e, " with n = ", nodeIdxs(e), " and l = ", length(e))
end
```

```
Edge[1] with n = [1, 2] and l = 2.0
Edge[2] with n = [2, 3] and l = 2.0
Edge[3] with n = [3, 4] and l = 2.0
Edge[4] with n = [4, 5] and l = 2.0
```

Various functions like `nodeIdxs` and `length` exist to access properties. If you are used to an object-oriented language like Java it might be helpful to understand that `nodeIdxs(e)` in Julia is equivalent to `e.nodeIdxs()` in an OO language.

Node coordinates are retrieved using the `coordinates` method:

```
coordinates(m)
```

```
2×5 Matrix{Float64}:
 0.0  2.0  4.0  6.0  8.0
 0.0  0.0  0.0  0.0  0.0
```

Properties are associated with the mesh using the `data` field of the mesh and a name for the property in the form `:name`

```
m.data[:foo] = 99
m.data[:bar] = sqrt
```

```
sqrt (generic function with 42 methods)
```

and then are ready to be used in a later stage

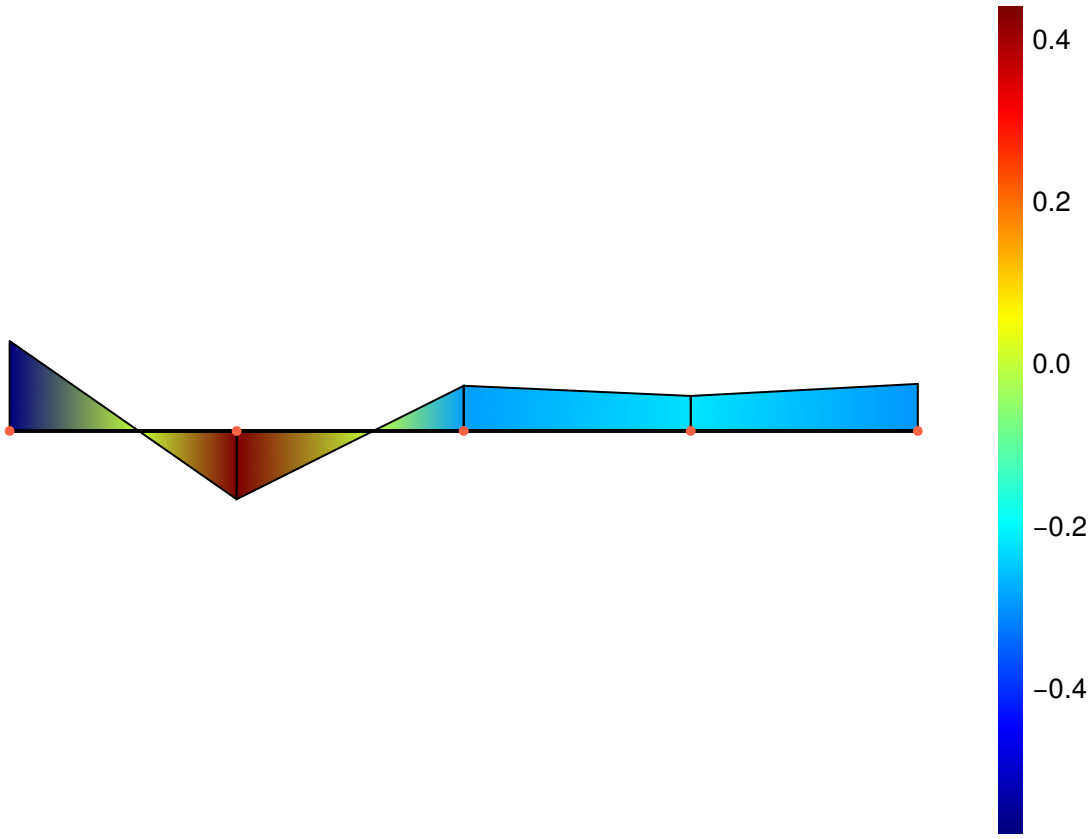
```
f = m.data[:bar]
f(4)
```

```
2.0
```

In Julia, `:name` is called a symbol. In many applications, this is equivalent to the use of strings like `"name"`, however, easier to type.

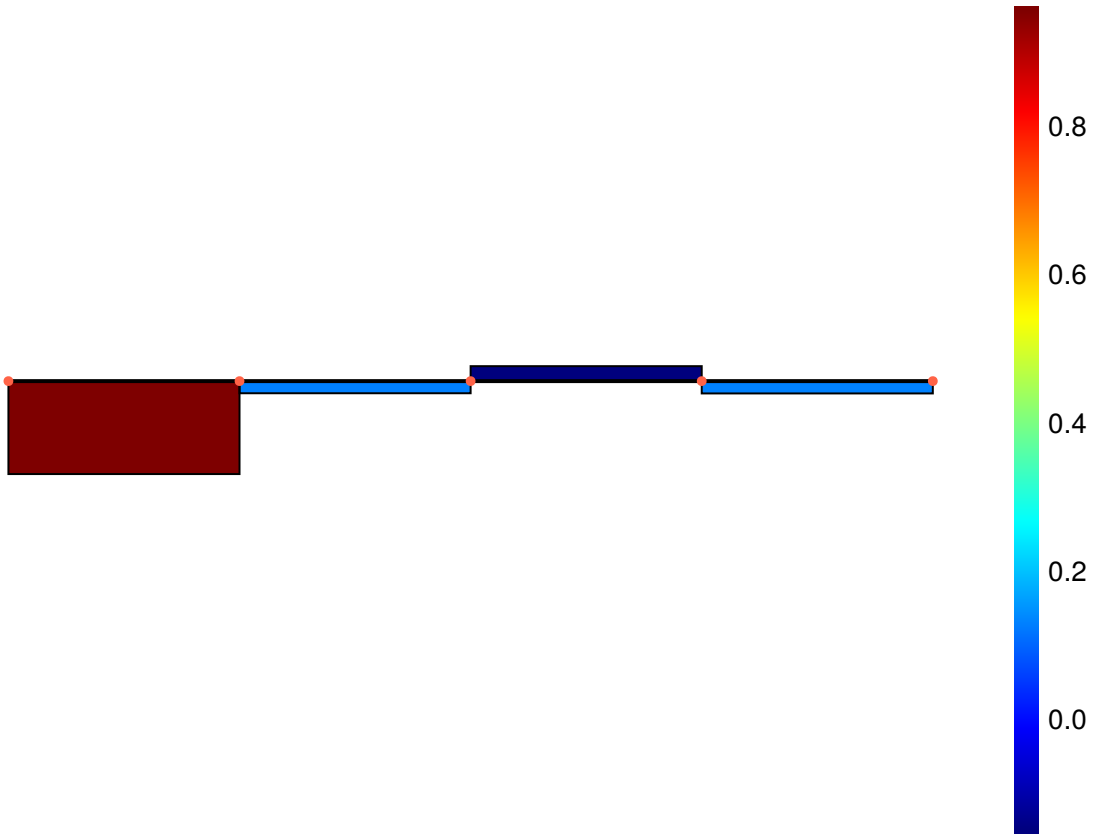
It is easy to plot quantities for nodes

```
plot(m, -1 .+ 2 * rand(nnodes(m)))
```



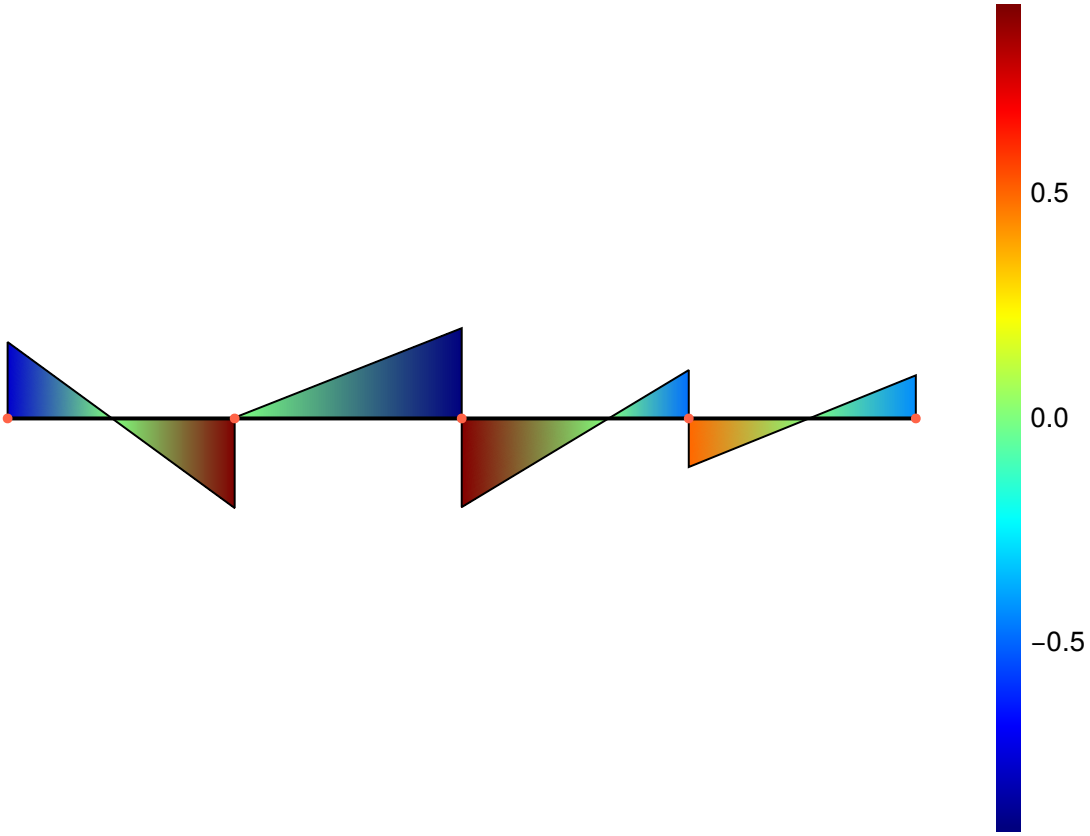
constant on elements

```
plot(m, -1 .+ 2 * rand(nelements(m)))
```



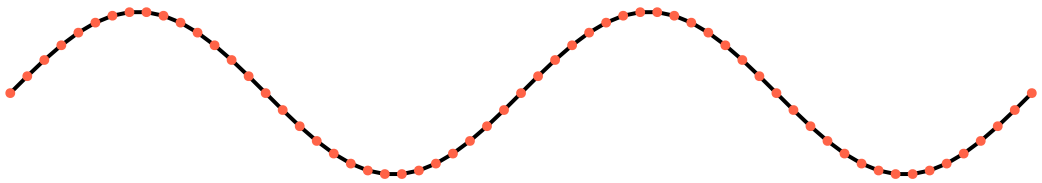
or linear on elements

```
plot(m, -1 .+ 2 * rand(2, nelements(m)))
```



Furthermore, 1D meshes can be created with a parametric function

```
m = makemeshoninterval(0, 4pi, 60, t -> [t; sin(t)])
plot(m)
```



where the last parameter `t -> [t; sin(t)]` (read: `t` is mapped on the vector `(t, sin(t))`) defines the parametric curve

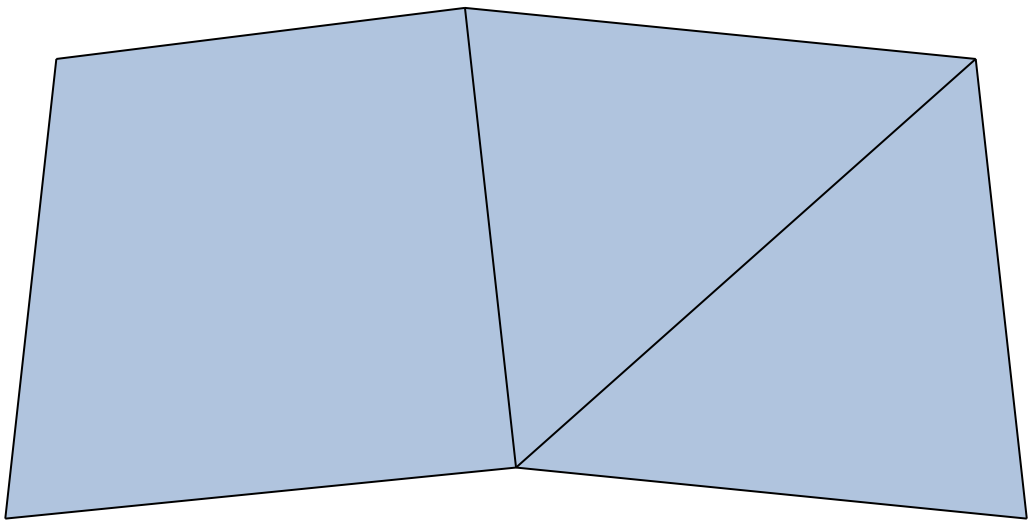
$$\mathbf{u}(t) = \begin{pmatrix} t \\ \sin(t) \end{pmatrix}.$$

2 The Mesh interface

Create mesh

```
coords = [0.0 1.0 2.0 0.1 0.9 1.9; 0.0 0.1 0.0 0.9 1.0 0.9]
elts = [[1, 2, 5, 4], [2, 3, 6], [2, 6, 5]]
m = Mesh(coords, elts, 2)

plot(m)
```



Coordinates of a node

```
coordinates(node(m, 2))
```

```
2-element Vector{Float64}:
 1.0
 0.1
```

Coordinates of a face

```
coordinates(face(m, 3))
```

```
2×3 Matrix{Float64}:
 1.0  1.9  0.9
 0.1  0.9  1.0
```

Process faces and print node and edge indexes

```
for e  elements(m)
    println(e)
    println("- ", nodeIdxs(e))
    println("- ", edgeIdxs(e))
    println("- ", faceIdxs(e))
end
```

```
Face[1]
- [1, 2, 5, 4]
- [1, 2, 3, 4]
- [3]
Face[2]
- [2, 3, 6]
- [5, 6, 7]
- [3]
Face[3]
- [2, 6, 5]
- [7, 8, 2]
- [1, 2]
```

3 Plotting

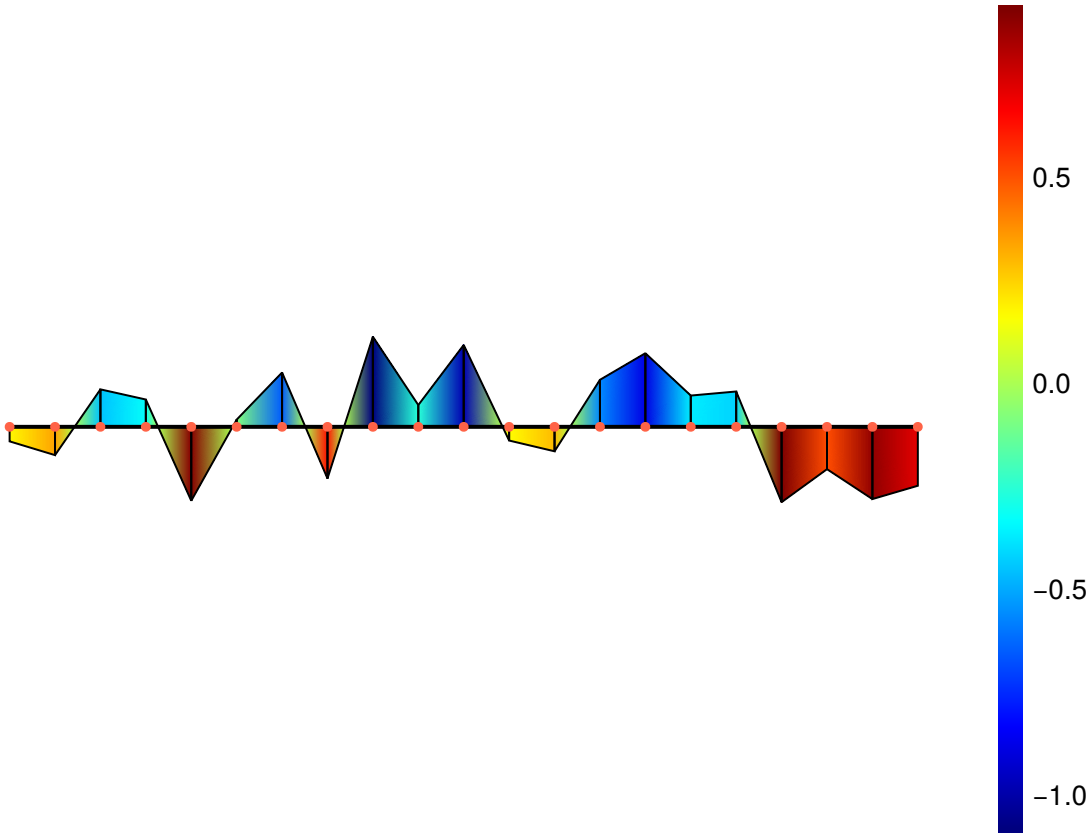
3.1 Plot a 1D mesh

3.1.1 On a straight line

```
m = makemeshoninterval(0, 4, 20)
plot(m)
```

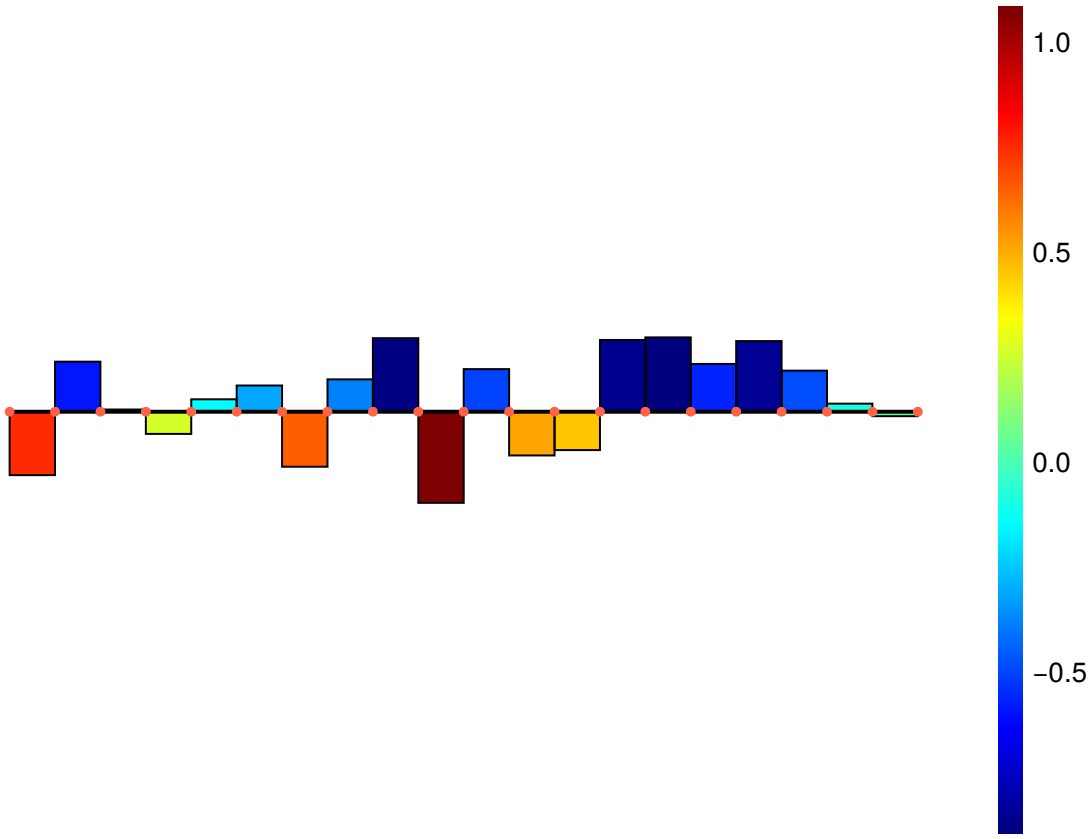
One value per node

```
plot(m, -1.1 .+ 2.2 * rand(nentities(m.topology, 0)))
```



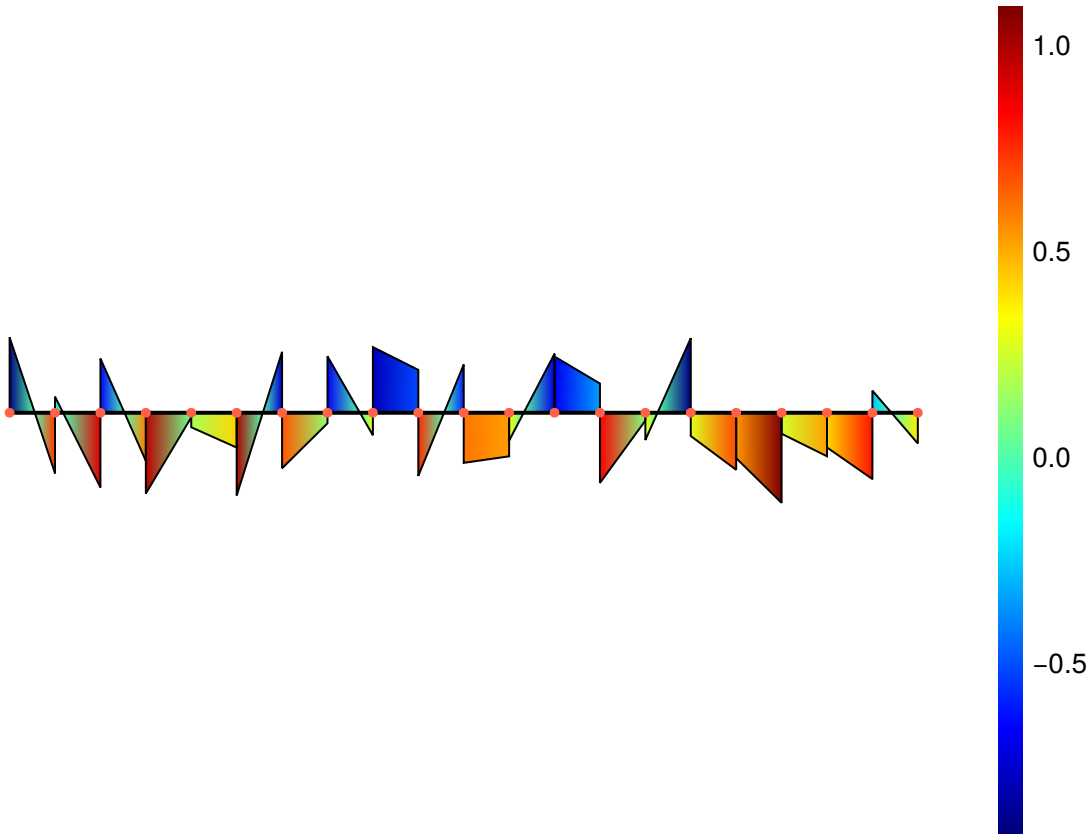
One value per element

```
plot(m, -1.1 .+ 2.2 * rand(nedges(m)))
```



Two values per element

```
plot(m, -1.1 .+ 2.2 * rand(2, nedges(m)))
```



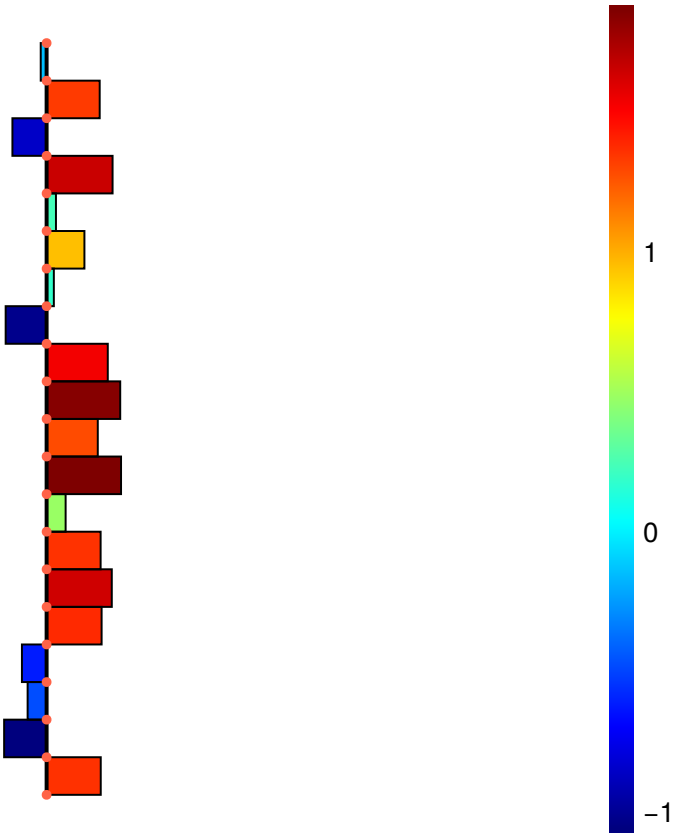
3.1.2 Vertical

```
m = makemeshoninterval( , 3 , 20, t -> [0; t])
```

```
================================================================================================
Mesh{1, 2}
================================================================================================
Topology{1}
Entities
         0: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]
         1: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
Links
    (1, 0): [[1, 2], [2, 3], [3, 4], [4, 5], [5, 6], [6, 7], [7, 8], [8, 9], [9, 10], [10, 1
------------------------------------------------------------------------------------------------
Geometry{2}
```

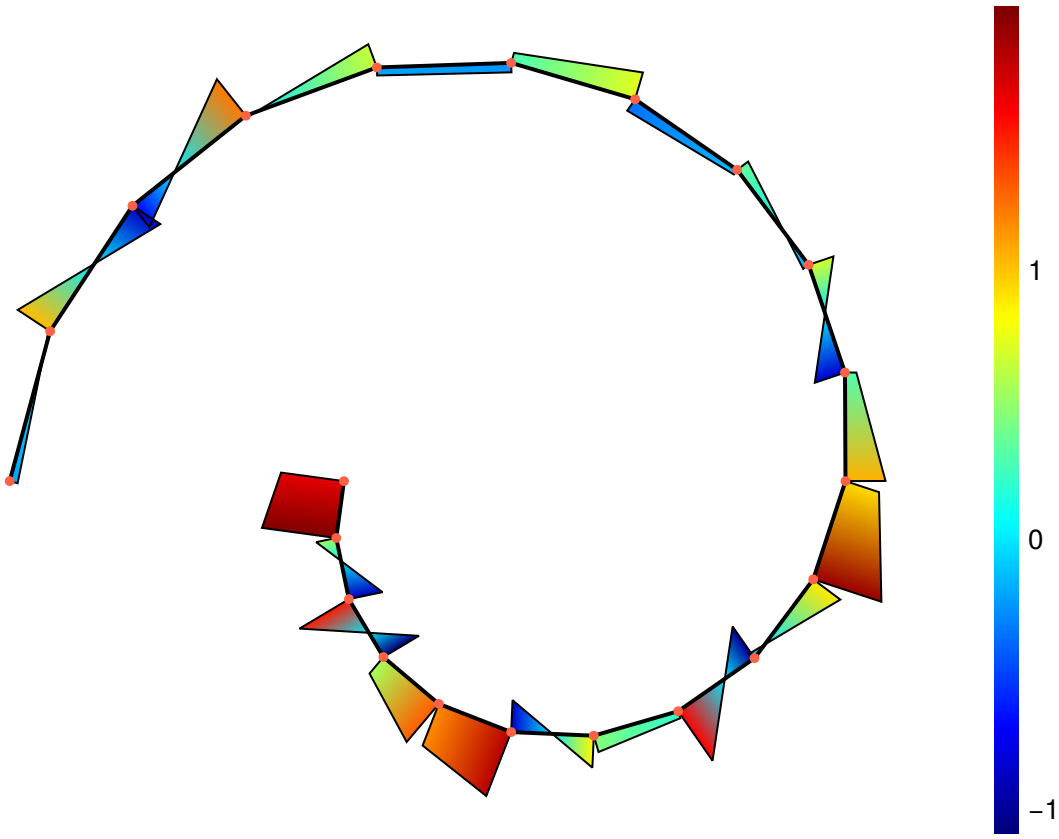
```
21 Points
[0.0 0.0 … 0.0 0.0; 3.141592653589793 3.4557519189487724 … 9.1106186954104 9.42477796076938]
=============================================================================================
```

```
plot(m, -1.1 .+ 3.2 * rand(nedges(m)))
```



3.1.3 On a spiral

```
m = makemeshoninterval( , 3 , 20, t -> t * [cos(t); sin(t)])
plot(m, -1.1 .+ 3.2 * rand(2, nedges(m)))
```



3.2 Plot a 2D mesh

3.2.1 Quad mesh

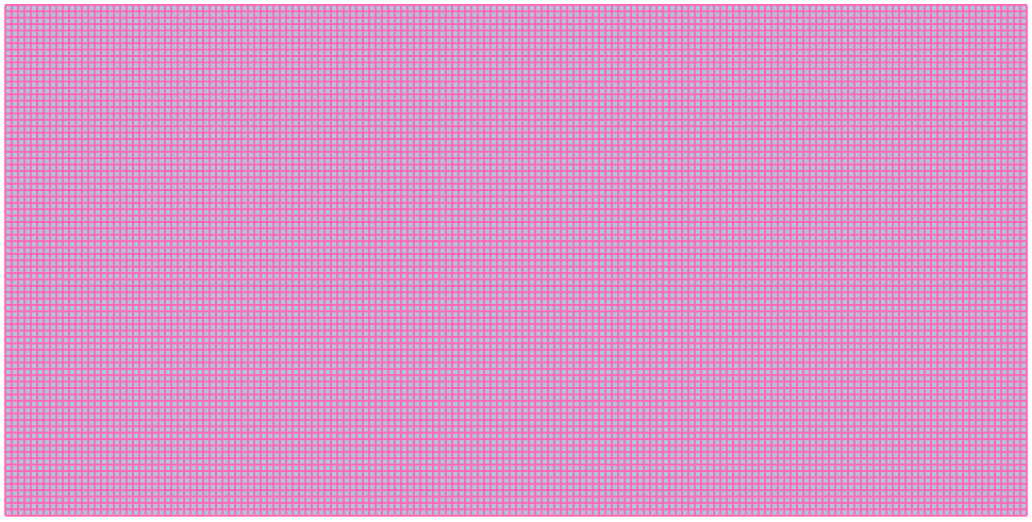
```
a = 80
m = makemeshonrectangle(9.0, 4.5, 2a, a)
println("Number of nodes is Nn = ", (a + 1) * (2a + 1))
print("Links...")
@time l12 = links(m.topology, 1, 2);
```

```
Number of nodes is Nn = 13041
Links...
```

```
  0.071324 seconds (1.10 M allocations: 85.724 MiB, 30.80% gc time)
```

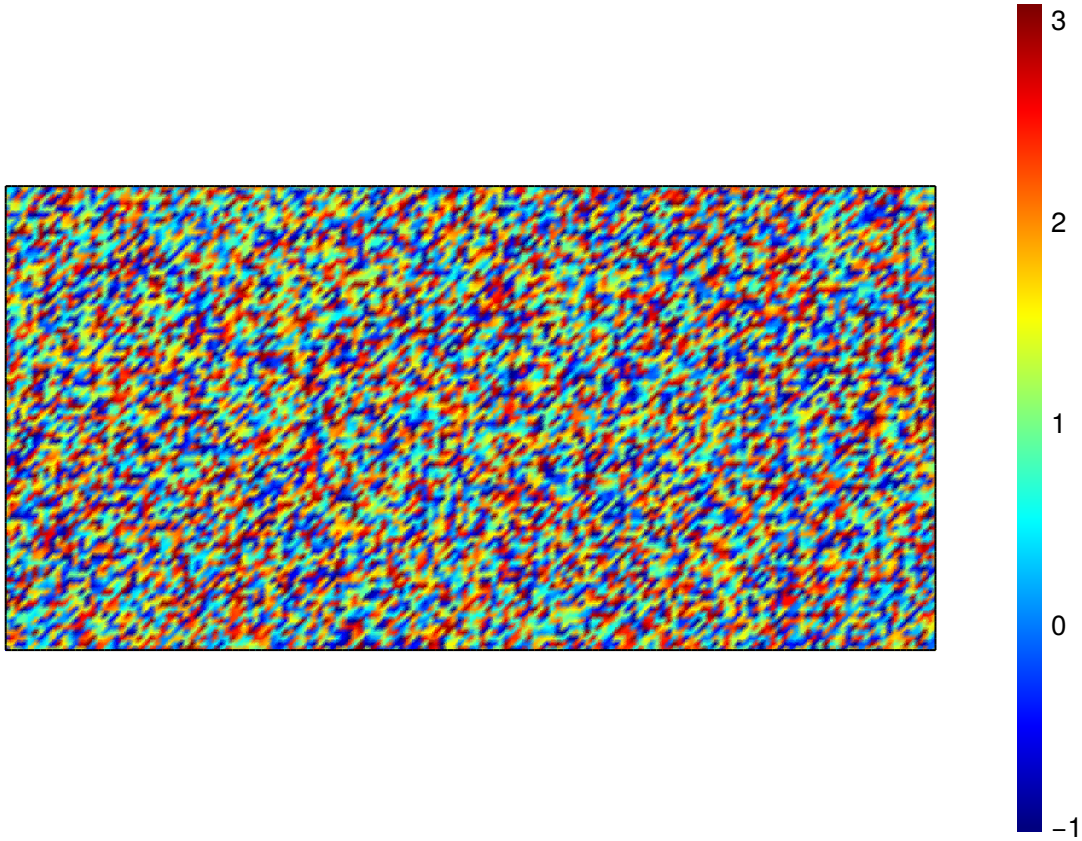
Default color

```
ps = PlotStyle(m)
ps.edges.outlineonly = false
ps.edges.color = :hotpink
plot(m, ps)
```



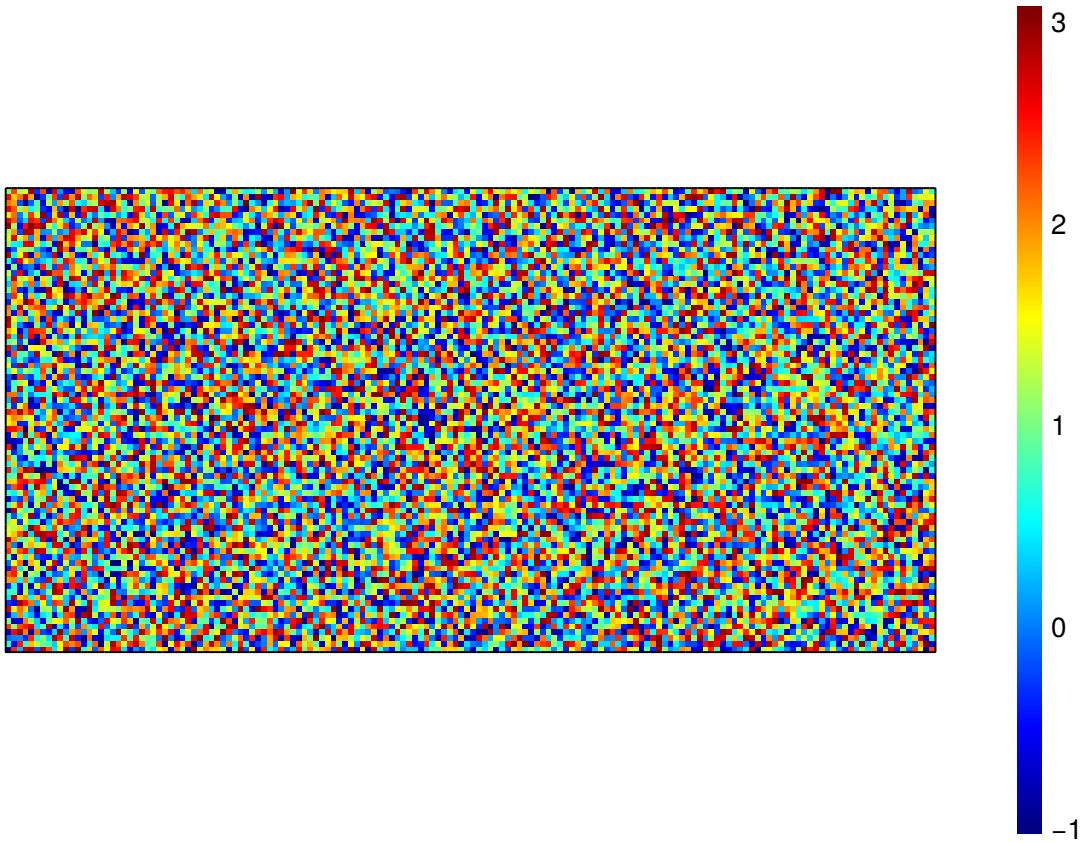
Colors for nodes

```
plot(m, 4.1 * (rand(nentities(m.topology, 0)) .- 0.25))
```



Colors for elements

```
plot(m, 4.1 * (rand(nentities(m.topology, 2)) .- 0.25))
```



3.2.2 Triangle mesh

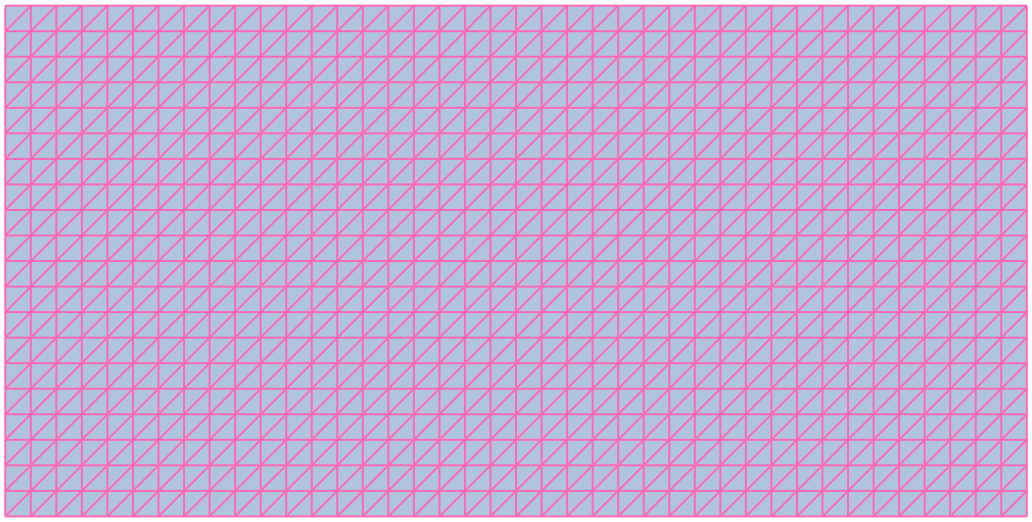
```
a = 20
m = makemeshonrectangle(9.0, 4.5, 2a, a, TRIANGLE)
println("Nn = ", (a + 1) * (2a + 1))
print("Links (1, 2):")
@time l12 = links(m.topology, 1, 2);
```

```
Nn = 861
Links (1, 2):
```

```
  0.004171 seconds (101.71 k allocations: 8.227 MiB)
```

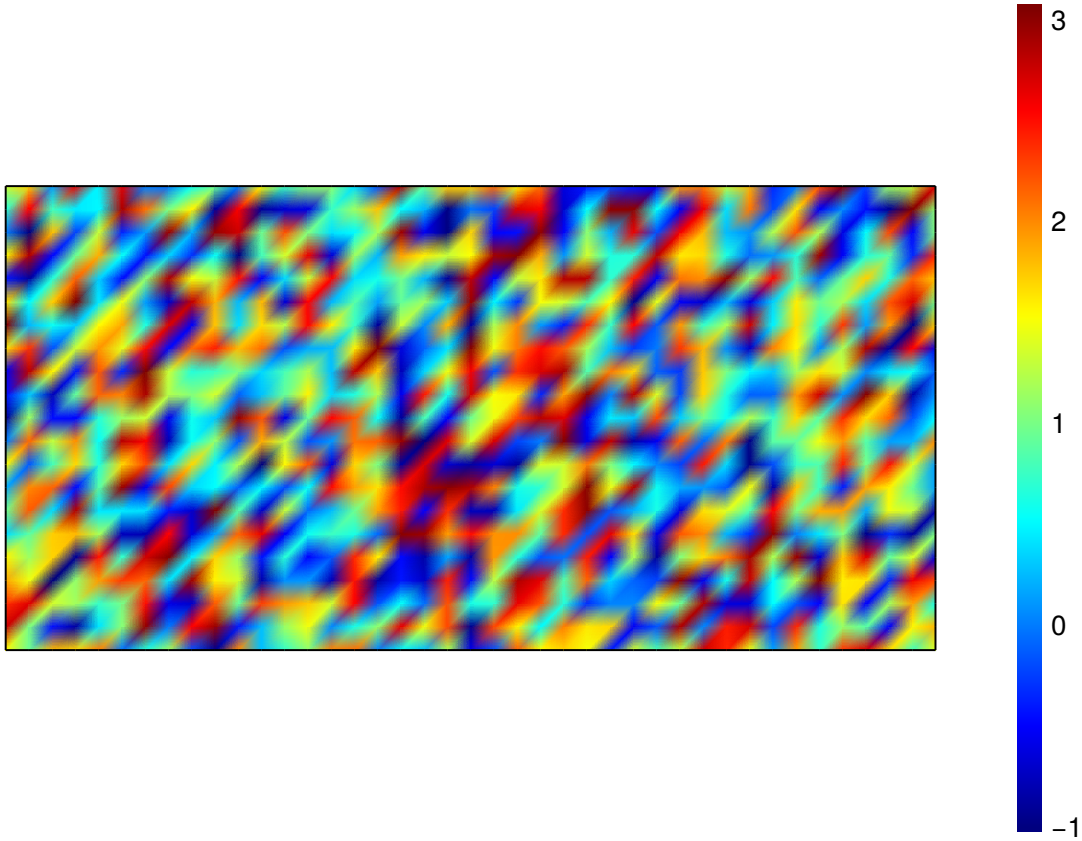
Default color

```
ps = PlotStyle(m)
ps.edges.outlineonly = false
ps.edges.color = :hotpink
plot(m, ps)
```



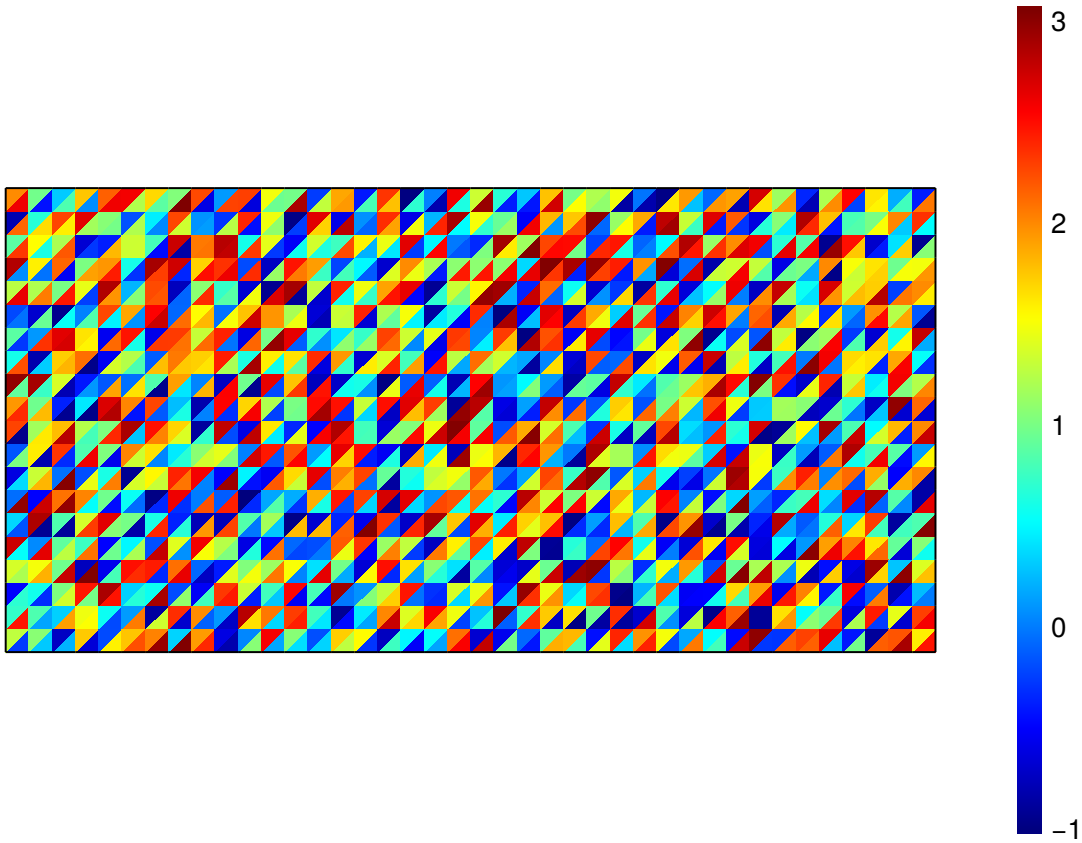
Colors for nodes

```
plot(m, 4.1 * (rand(nentities(m.topology, 0)) .- 0.25))
```



Colors for elements

```
plot(m, 4.1 * (rand(nentities(m.topology, 2)) .- 0.25))
```


3
2
1
0
−1