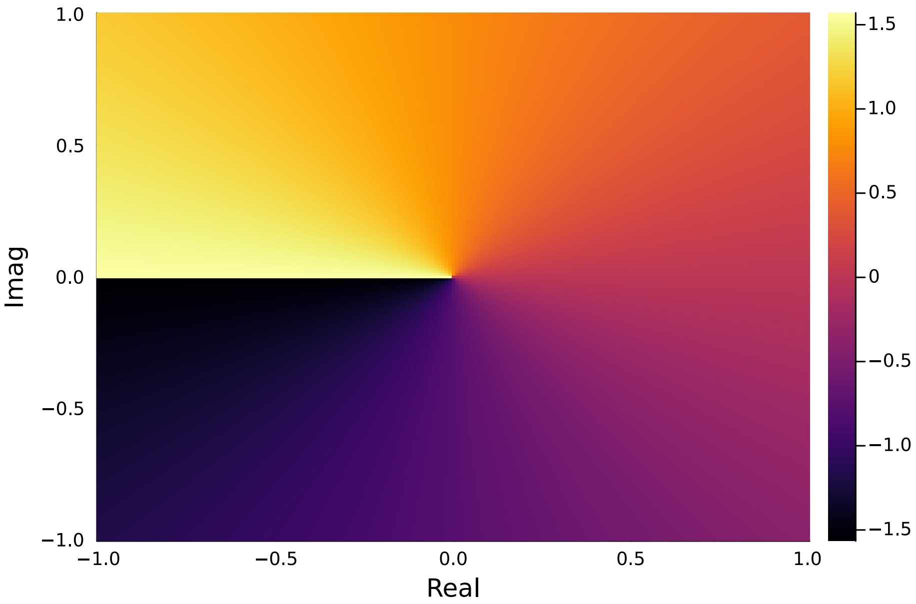
In [1]:
```julia
using Plots
```

Check location of branch cut of standard square root:

In [116…
```julia
δ = 0.01
t = -1:δ:1
z = t' .+ im*(t);
```

In [117…
```julia
heatmap(t,t,angle.(sqrt.(z)),xlabel="Real",ylabel="Imag")
```

Out[117…



In [118…
```julia
sqrt(-1+0.001*im)
```

Out[118…
```
0.0004999999375000273 + 1.000000124999961im
```

In [119…
```julia
sqrt(-1-0.001*im)
```

Out[119…
```
0.0004999999375000273 - 1.000000124999961im
```

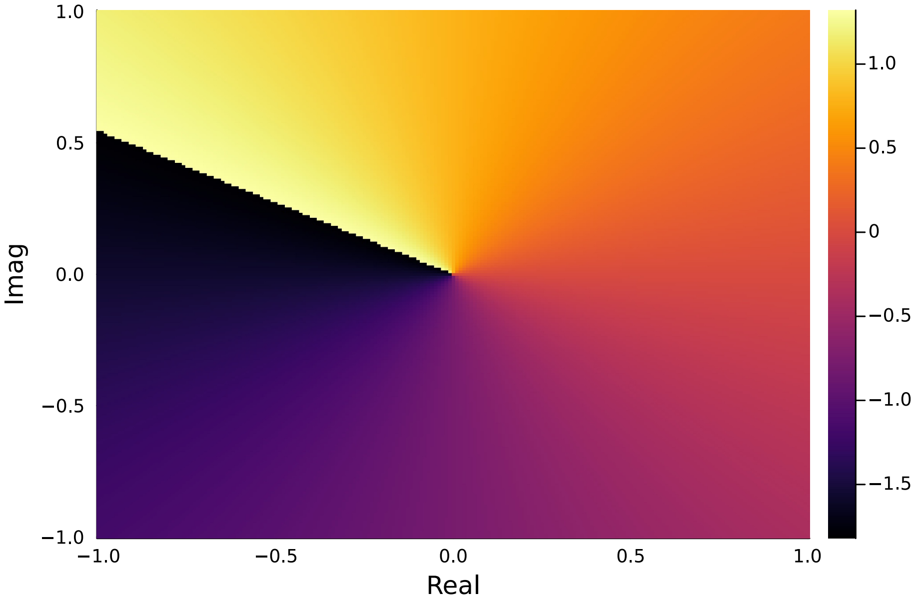
Now make our own. The logic here is that if $\arg z \in [\theta, \pi]$, then we want to be on a different branch to the standard square root function `sqrt` (which has its branch cut along $\arg z = \pi$). We can switch branches simply by multiplying by -1. Otherwise, our function should match the standard square root function `sqrt` .

In [120…
```julia
function sqrt_modified(z, θ)
   if (θ <= angle(z))
        return -sqrt(z) #switch branch
    else
        return sqrt(z) #dont switch branch
    end
end
```

Out[120…
```
sqrt_modified (generic function with 1 method)
```

In [121…
```julia
Z = sqrt_modified.(z,π-0.5)
B = sqrt_modify.(z, π-0.5)
heatmap(t,t,angle.(sqrt_modified.(z,π-0.5)),xlabel="Real",ylabel="Imag")
```

Out[121…



In [ ]:

Processing math: 100%