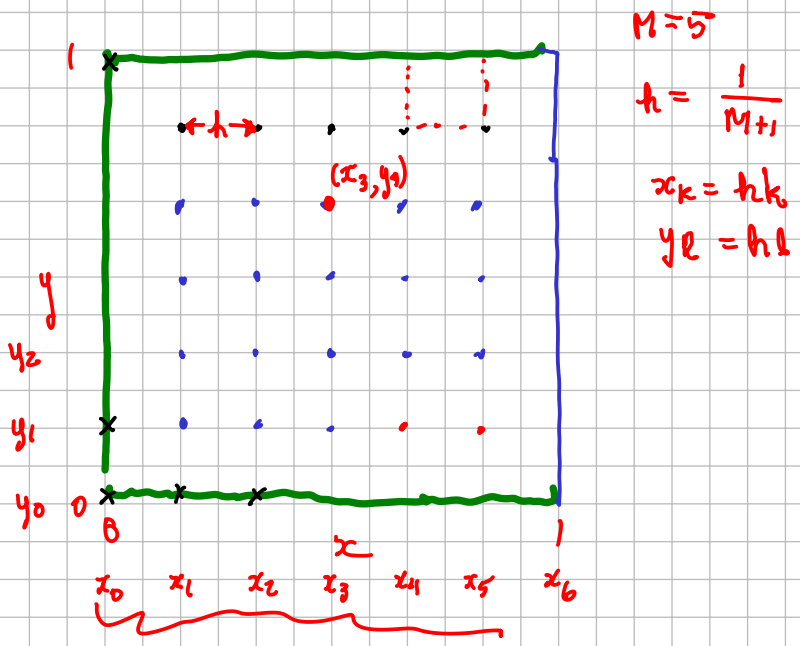



```
M=10
h=1/(M+1)

exact=zeros(M+2,M+2)
for k=0:M+1
    for l=0:M+1
        exact[k+1,l+1]=uexact(k*h,l*h)
    end
end
```

x y

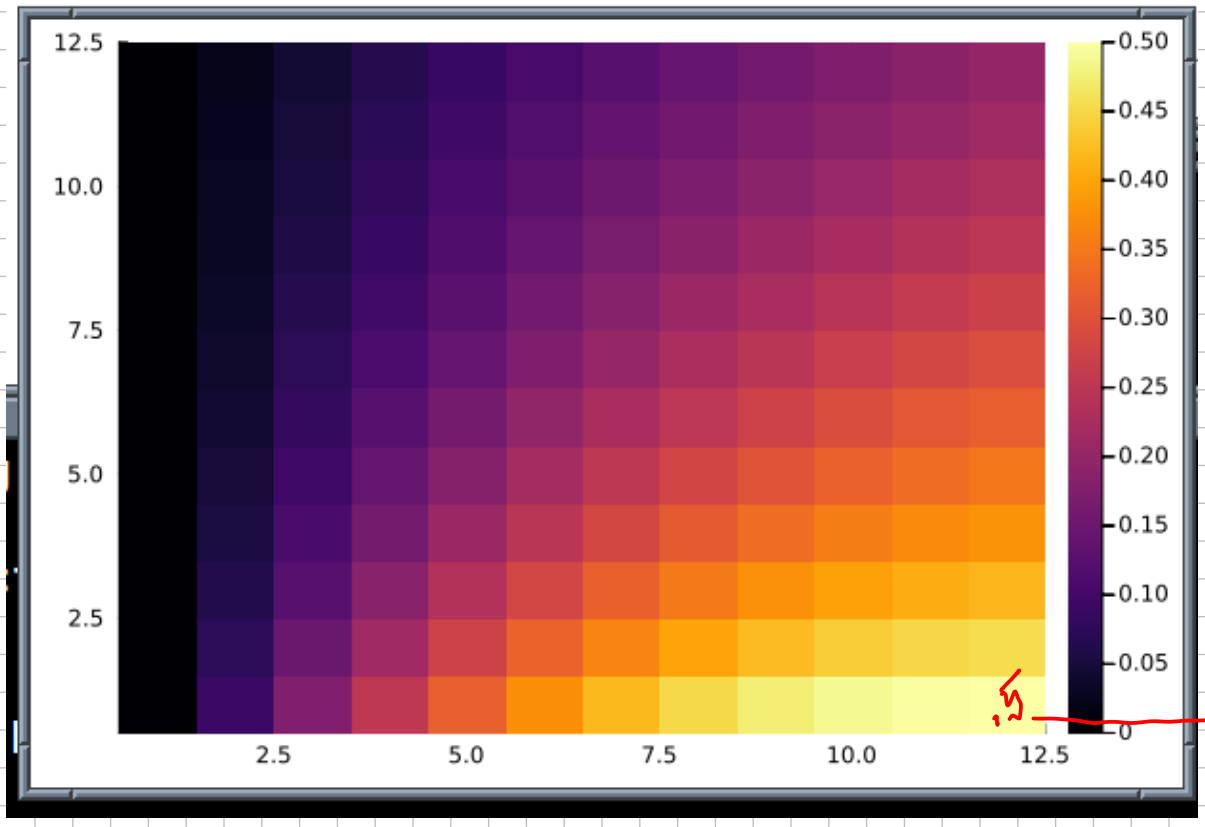
```
julia> exact
12×12 Matrix{Float64}:
 0.0  0.0901639  0.176      0.253846   …  0.497738  0.5
 0.0  0.0758621  0.148649   0.215686      0.45082   0.456604
 0.0  0.0647059  0.127168   0.185393      0.408922  0.417241
 0.0  0.0558376  0.11       0.160976      0.371622  0.381703
 0.0  0.0486726  0.0960699  0.141026      0.338462  0.349711
 0.0  0.0428016  0.0846154  0.124528   …  0.308989  0.320955
 0.0  0.037931   0.0750853  0.110738      0.282776  0.295122
 0.0  0.0338462  0.0670732  0.0990991     0.259434  0.27191
 0.0  0.0303867  0.060274   0.0891892     0.238612  0.251037
 0.0  0.0274314  0.0544554  0.0806846     0.22      0.232246
 0.0  0.0248869  0.0494382  0.0733333  …  0.203327  0.215302
 0.0  0.0226804  0.045082   0.0669371     0.188356  0.2
```

x-axis

x

y axis

```
ulia> heatmap(exact)
```

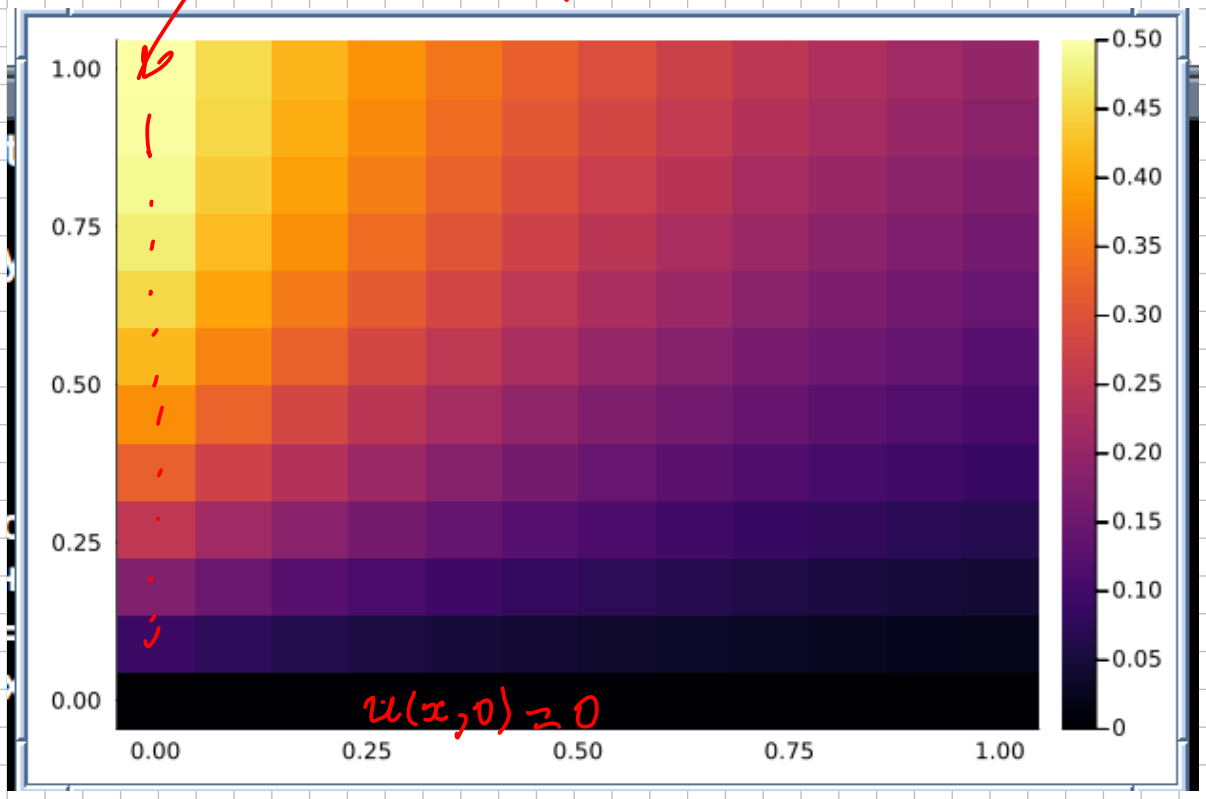


x axis

y axis

```
heatmap(xs,xs,exact')
```

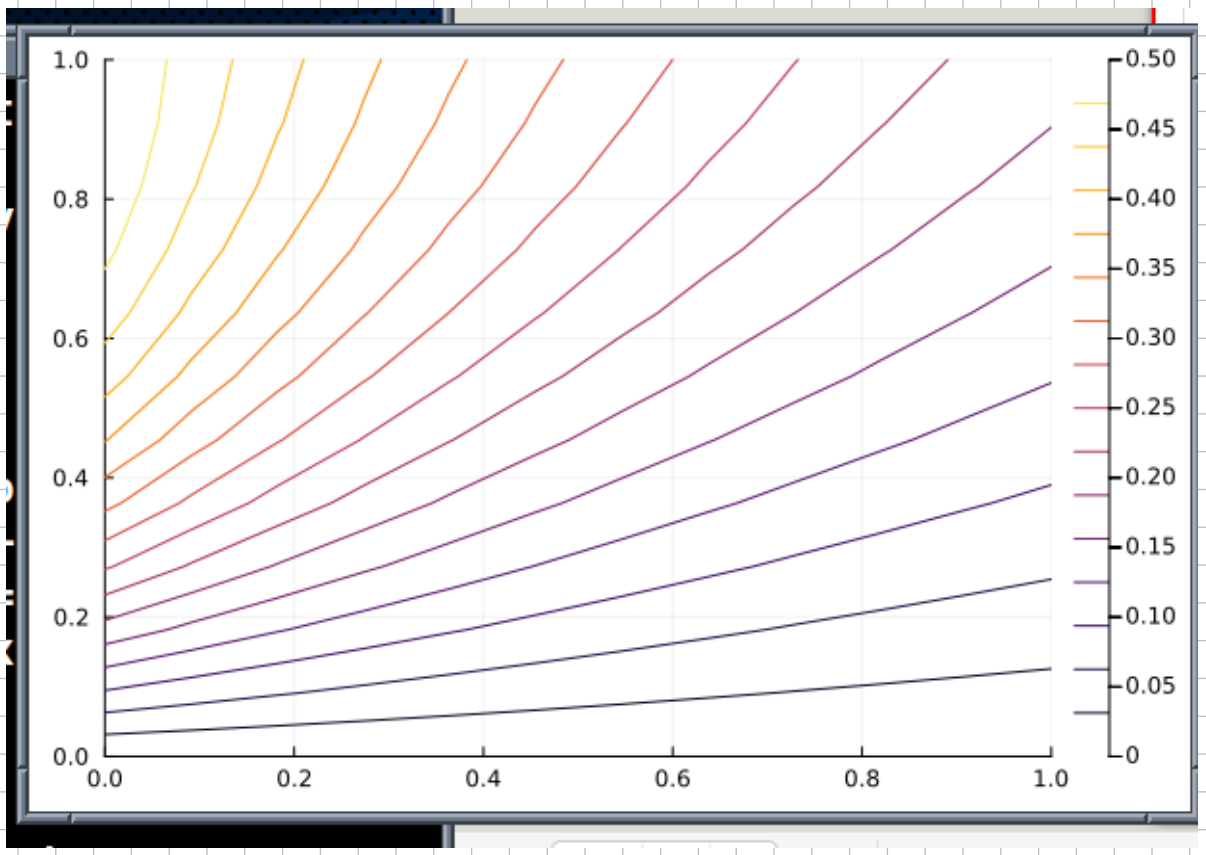
$u(0,y) = y/1+y^2$



y axis

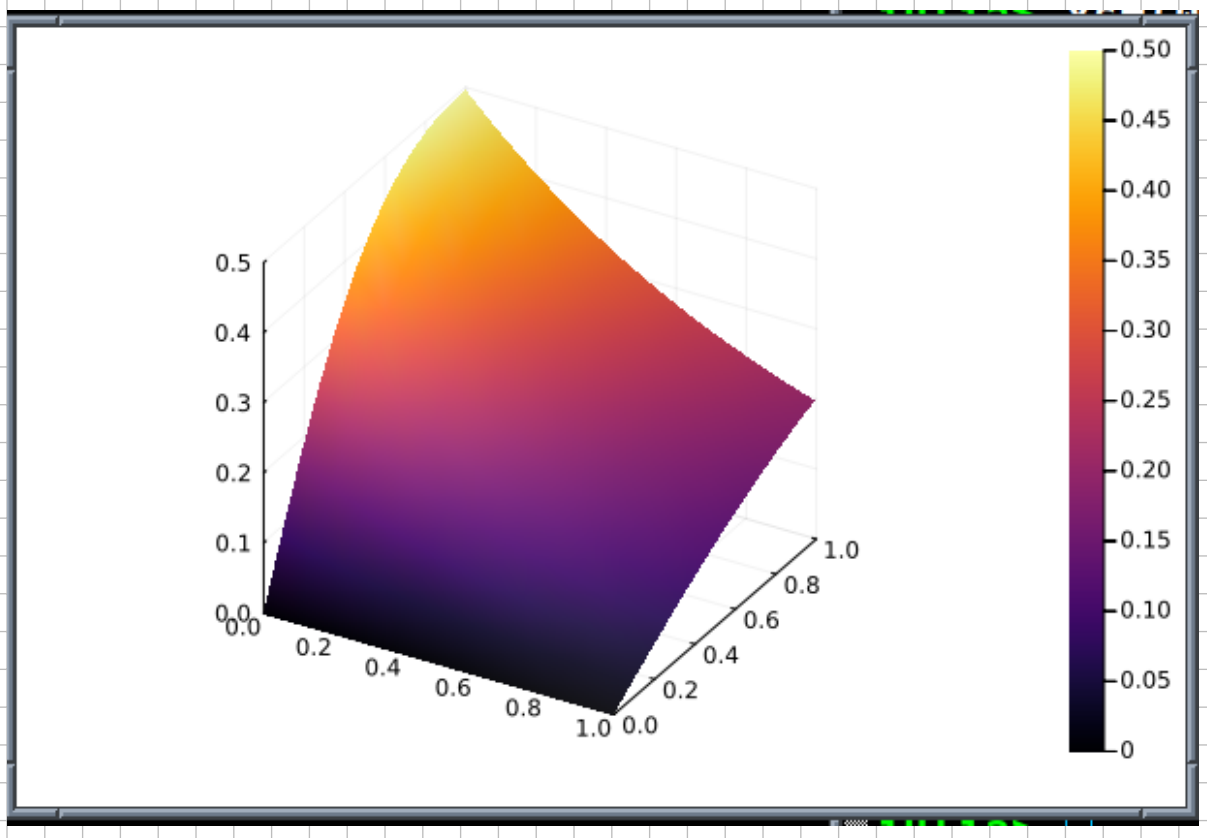
x axis

```
contour(xs,xs,exact')
```



Same information on contours..

```
julia> surface(xs,xs,exact')
```

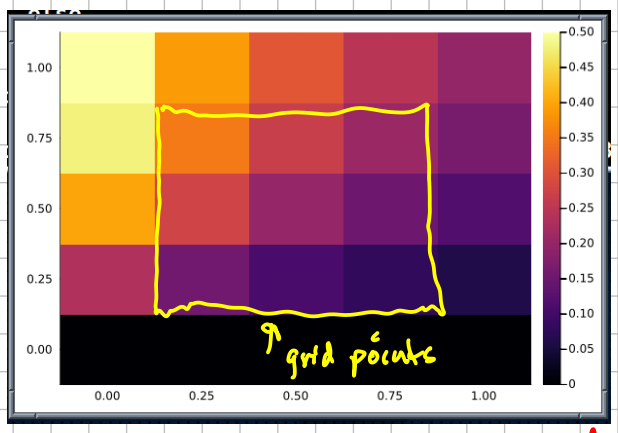


```
function klton(k,l)
    return l+(k-1)*M
end
```

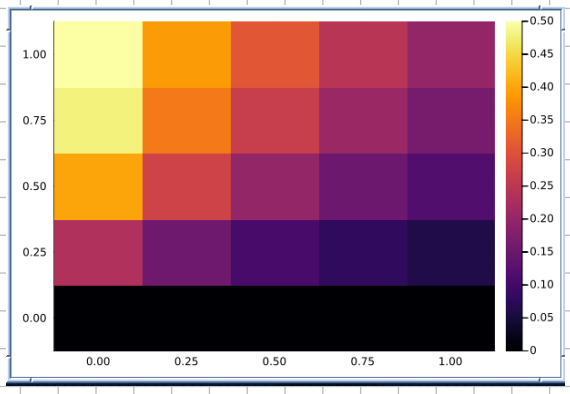
Mapping between the grid points and how they are stored...

$u(x_k, y_l) \approx u[l+(k-1)*M]$

approximation M=3

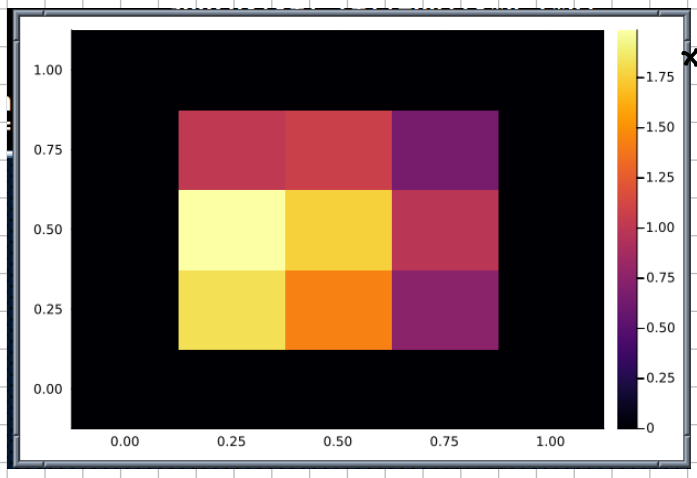


Exact solution



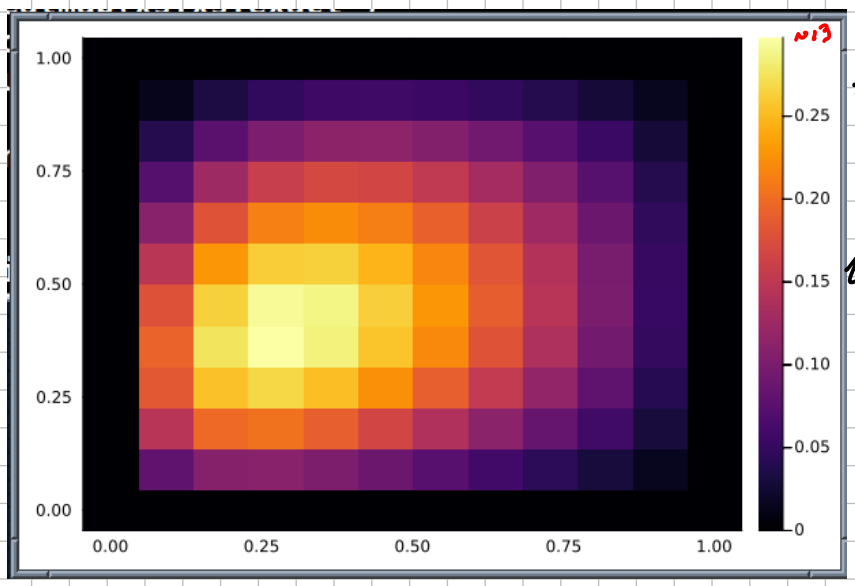
looks same

```
julia> heatmap(xs,xs,1000*(approx-exact)')
```



If you increase grid size does it converge?

M = 10



$\times 10^{-3}$

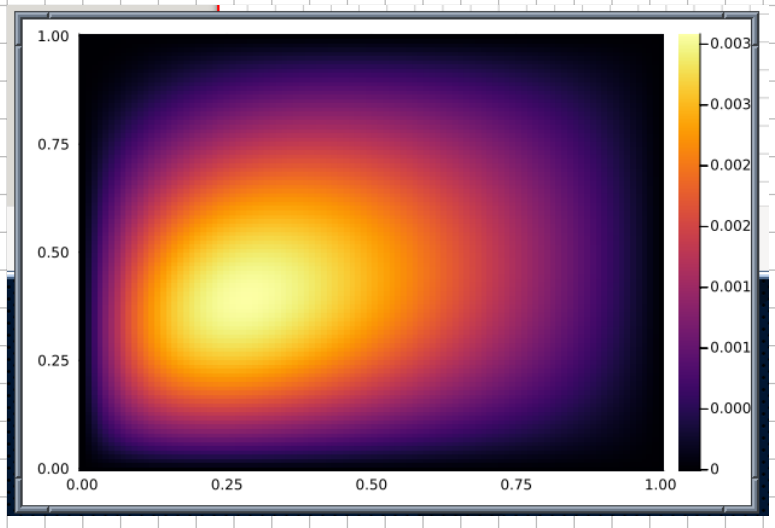
← error here?

```
julia> 3/10
0.3

julia> (3/10)^2
0.09

julia> 1.75*0.09
0.1575
```

M = 100



← 100 times smaller than error before...

That's what one expects for $O(h^2)$ method...