Laplace's equation

Contents

1 How to solve a maze without AI

> See these LinkedIn posts to see some comments and discussions:
>
> - https://www.linkedin.com/feed/update/urn:li:activity:6831291311832760320/
> - https://www.linkedin.com/feed/update/urn:li:activity:6973982270852325376/
>
> Other people's maze-related posts:
>
> - https://www.linkedin.com/feed/update/urn:li:activity:6972370982489509888/
> - https://www.linkedin.com/feed/update/urn:li:activity:6972949021711630336/
> - https://www.linkedin.com/feed/update/urn:li:activity:6973522069703516160/
> - https://www.linkedin.com/feed/update/urn:li:activity:6973921855275458560/
> - https://www.linkedin.com/feed/update/urn:li:activity:6974663157952745472/
> - https://www.linkedin.com/feed/update/urn:li:activity:6974979951049519104/
> - https://www.linkedin.com/feed/update/urn:li:activity:6982049404568449024/

Say you are Homer Simpson and you want to solve a maze drawn in a restaurant's placemat, one where both the start and end are known beforehand. In order to avoid falling into the alligator's mouth, you can exploit the ellipticity of the Laplacian operator to solve any maze (even a hand-drawn one) without needing any fancy AI or ML algorithm. Just FeenoX and a bunch of standard open source tools to convert a bitmapped picture of the maze into an unstructured mesh.

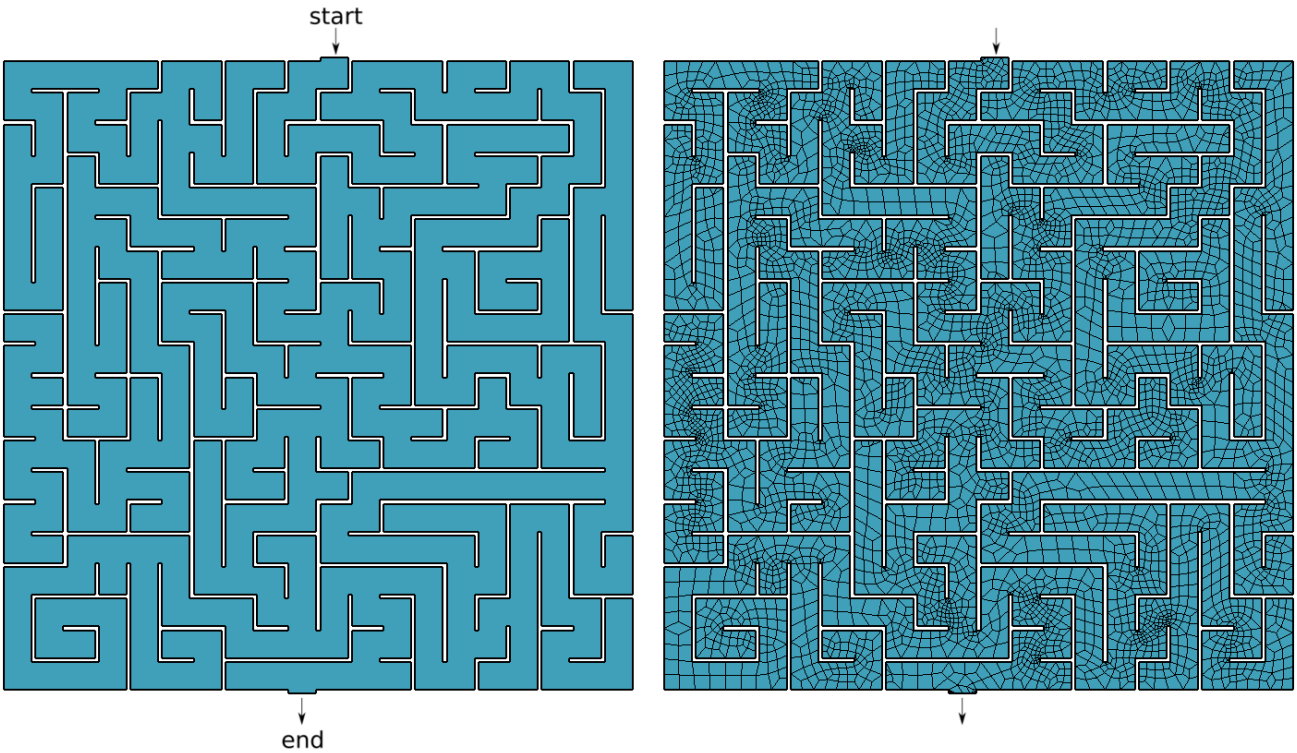

Figure 1: Bitmapped maze from https://www.mazegenerator.net (left) and 2D mesh (right)

```
PROBLEM laplace 2D  # pretty self-descriptive, isn't it?
READ_MESH maze.msh

# boundary conditions (default is homogeneous Neumann)
BC start  phi=0
BC end    phi=1

SOLVE_PROBLEM

# write the norm of gradient as a scalar field
# and the gradient as a 2d vector into a .msh file
WRITE_MESH maze-solved.msh \
    sqrt(dphidx(x,y)^2+dphidy(x,y)^2) \
    VECTOR dphidx dphidy 0
```

```
$ gmsh -2 maze.geo
[...]
$ feenox maze.fee
$
```

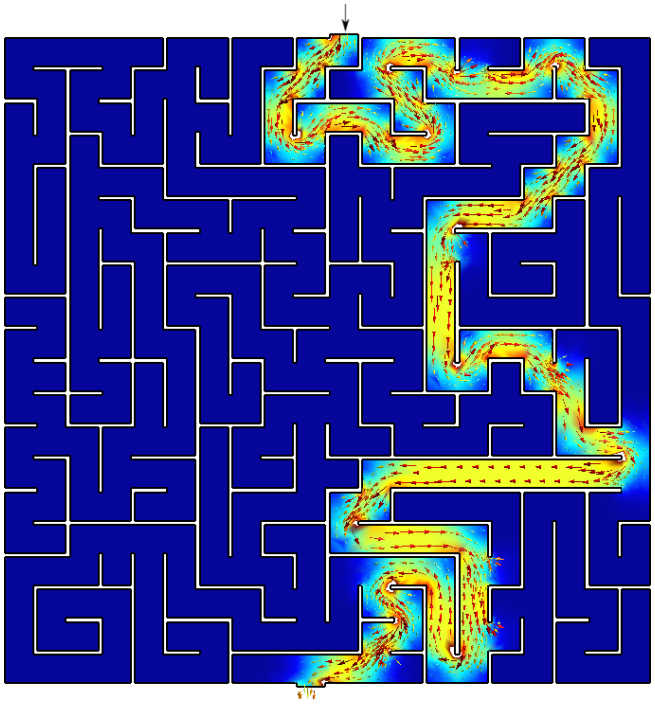

Figure 2: Solution to the maze found by FeenoX (and drawn by Gmsh)

1.1 Transient top-down

Instead of solving a steady-state en exploiting the ellipticity of Laplace's operator, let us see what happens if we solve a transient instead.

```
PROBLEM laplace 2D
READ_MESH maze.msh

phi_0(x,y) = 0              # inital condition
end_time = 100              # some end time where we know we reached the steady-state
alpha = 1e-6                # factor of the time derivative to make it advance faster
BC start   phi=if(t<1,t,1)  # a ramp from zero to avoid discontinuities with the initial condition
BC end     phi=0            # homogeneous BC at the end (so we move from top to bottom)

SOLVE_PROBLEM
PRINT t

WRITE_MESH maze-tran-td.msh phi    sqrt(dphidx(x,y)^2+dphidy(x,y)^2) VECTOR -dphidx(x,y) -dphidy(x,y) 0
```

```
$ feenox maze-tran-td.fee
0
0.00433078
0.00949491
0.0170774
0.0268599
[...]
55.8631
64.0819
74.5784
87.2892
100
$ gmsh maze-tran-td-anim.geo
# all frames dumped, now run
ffmpeg -y -framerate 20 -f image2 -i maze-tran-td-%03d.png maze-tran-td.mp4
ffmpeg -y -framerate 20 -f image2 -i maze-tran-td-%03d.png maze-tran-td.gif
$ ffmpeg -y -framerate 20 -f image2 -i maze-tran-td-%03d.png maze-tran-td.mp4
[...]
$ ffmpeg -y -framerate 20 -f image2 -i maze-tran-td-%03d.png maze-tran-td.gif
[...]
```

1.2 Transient bottom-up

Now let us see what happens if we travel the maze from the exit up to the inlet. It looks like the solver tries a few different paths that lead nowhere until the actual solution is found.

```
PROBLEM laplace 2D
READ_MESH maze.msh

phi_0(x,y) = 0
end_time = 100
alpha = 1e-6
BC end     phi=if(t<1,t,1)
BC start   phi=0
```

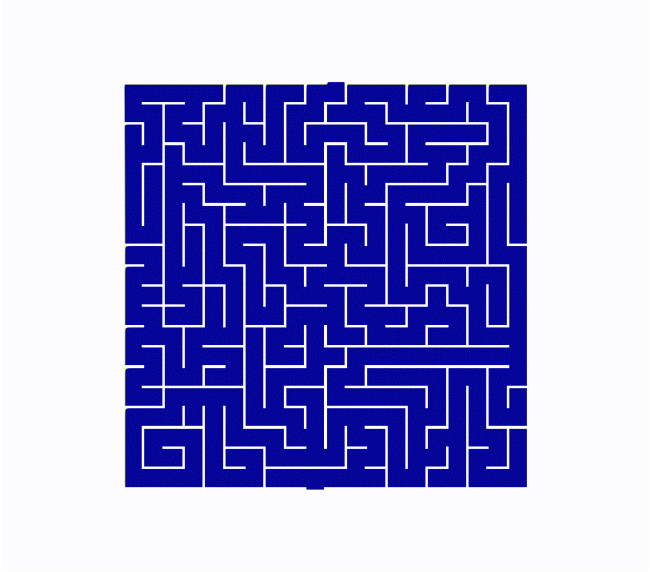

Figure 3: Transient top-bottom solution to the maze found by FeenoX (and drawn by Gmsh)

```
SOLVE_PROBLEM
PRINT t

WRITE_MESH maze-tran-bu.msh phi    sqrt(dphidx(x,y)^2+dphidy(x,y)^2) VECTOR -dphidx(x,y) -dphidy(x,y) 0
```

```
$ feenox maze-tran-bu.fee
0
0.00402961
0.00954806
0.0180156
0.0285787
[...]
65.3715
72.6894
81.8234
90.9117
100
$ gmsh maze-tran-bu-anim.geo
# all frames dumped, now run
ffmpeg -y -framerate 20 -f image2 -i maze-tran-bu-%03d.png maze-tran-bu.mp4
ffmpeg -y -framerate 20 -f image2 -i maze-tran-bu-%03d.png maze-tran-bu.gif
$ ffmpeg -y -framerate 20 -f image2 -i maze-tran-bu-%03d.png maze-tran-bu.mp4
[...]
$ ffmpeg -y -framerate 20 -f image2 -i maze-tran-bu-%03d.png maze-tran-bu.gif
[...]
```

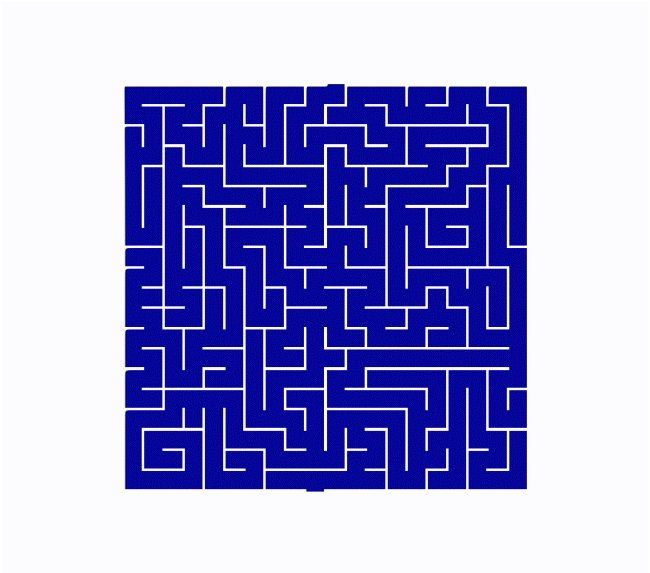

Figure 4: Transient bottom-up solution. The first attempt does not seem to be good.