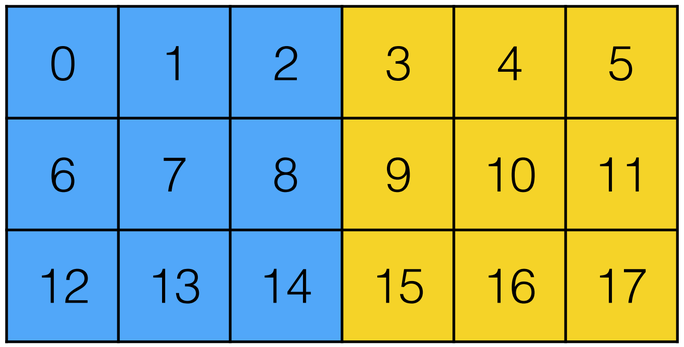
High Performance Computing for Science and Engineering

Exercise 9: Sparse Linear Algebra with MPI

Solution

Mapping grid to vector



global indices

vector

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

GlbToRank()

0
0
0
1
1
1
0
0
0
1
1
1
0
0
0
1
1
1

GlbToLoc()

0
1
2
0
1
2
3
4
5
3
4
5
6
7
8
6
7
8

rank 0

rank 1

```
grid size:

NX = 6

NY = 3


number of blocks:

NBX = 2

NBY = 1
```

Matrix-vector product **b=Au**

- Mul() is not aware of the grid-vector mapping,
  relies only on GlbToRank() and GlbToLoc()
- traverse rows of **A**, the product requires some elements of **u**
  for local elements: compute the product,
  for remote elements: collect indices of **u**, indices of **b** and coefficients of **A**
- send/recv indices of **u**
- send/recv values of **u**
- append to product **b**

Communication

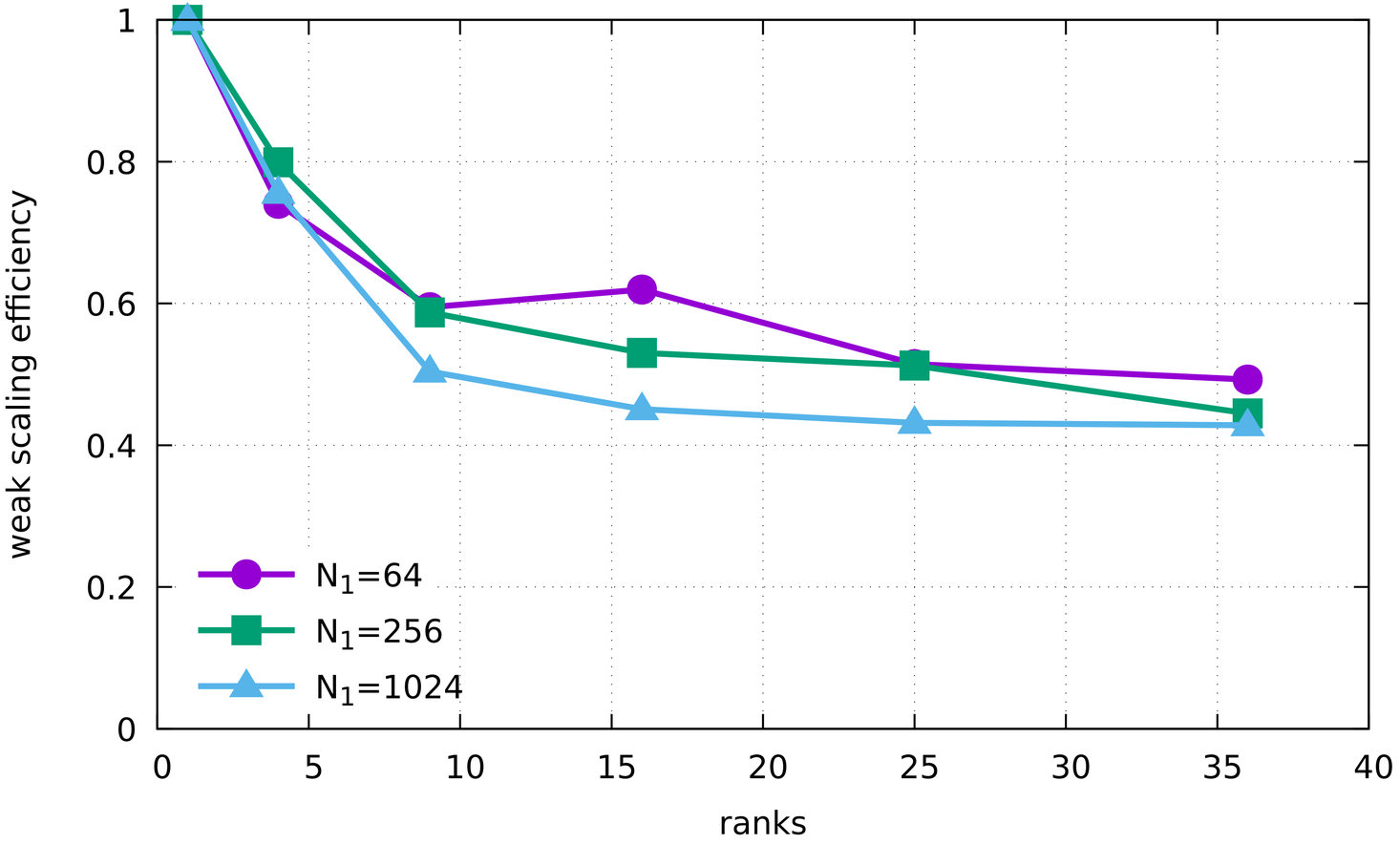
- each processor acts as a server:
  receives **requests** (indices of **u**) and sends **responses** (values of **u**)

- how many requests to receive?
  each processor **r** knows where to send the requests,
  array: **m**[**re**]=1 if sending to rank **re**, otherwise **m**[**re**]=0
  do MPI_Allreduce on **m**, then **m**[**r**] is how many to receive

- how to know the message size?
  use MPI_Probe and MPI_Get_count

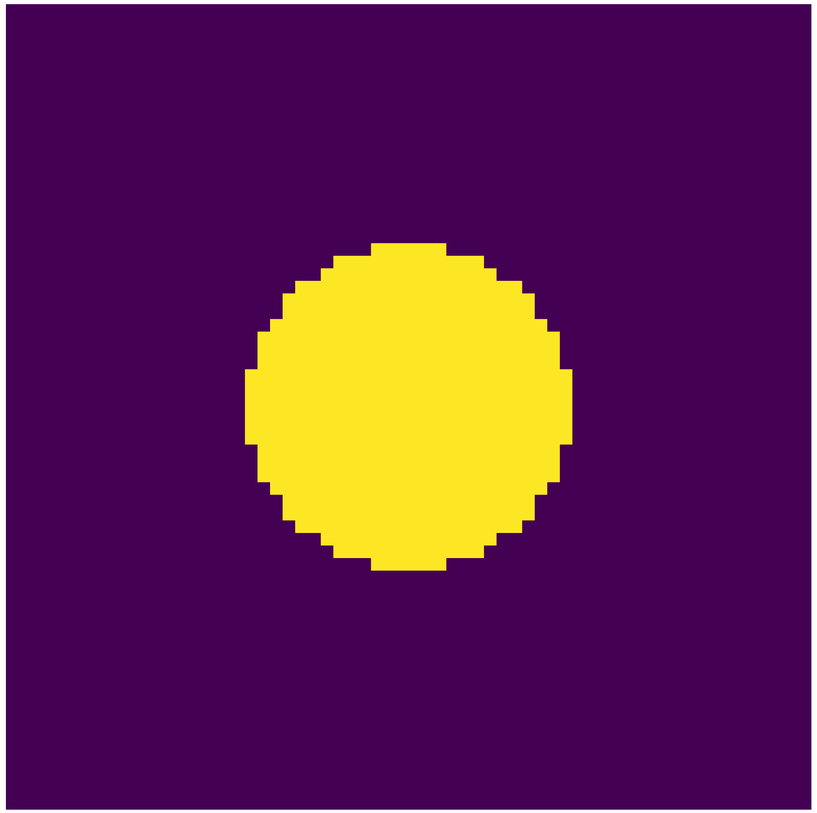
- avoid deadlocks by having **send** or **recv** non-blocking

Weak scaling

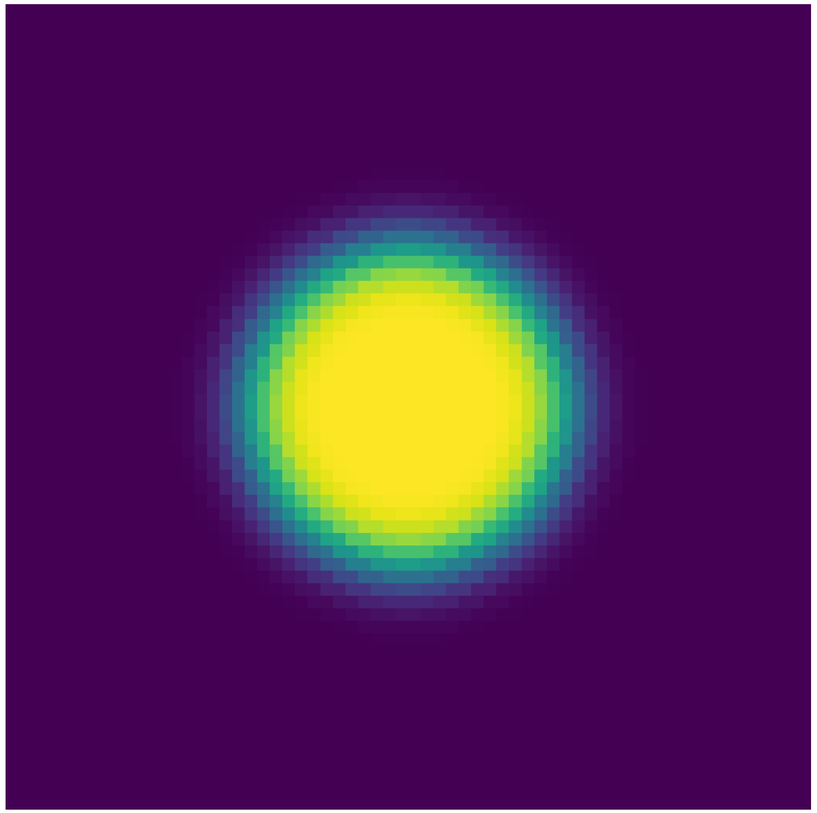


- constant number of cells per rank
- sufficiently many timesteps (runtime about 10s)
- exclude I/O

Diffusion equation



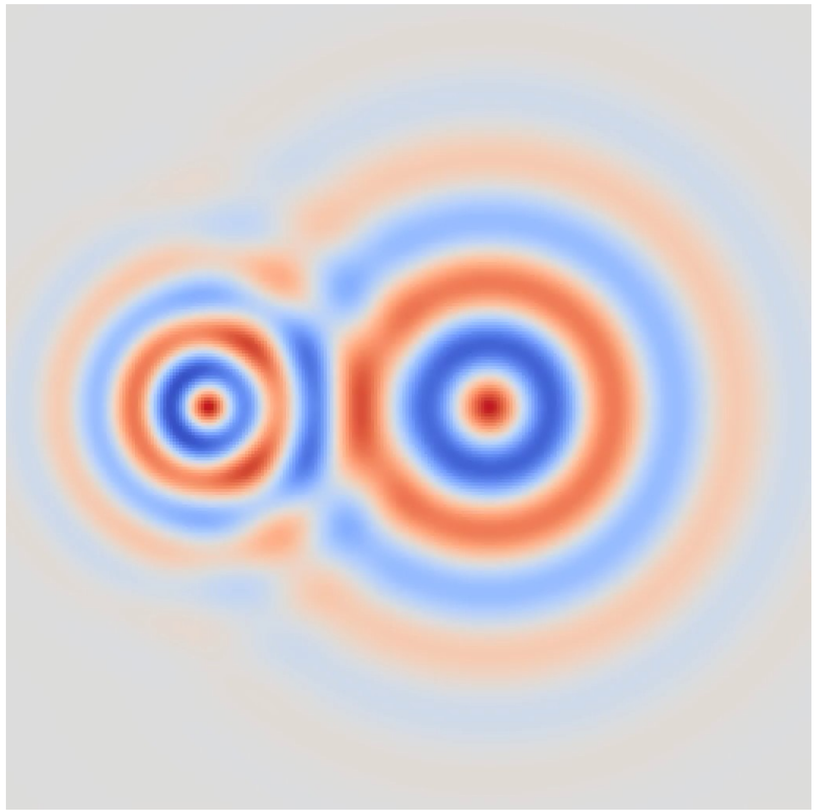
initial



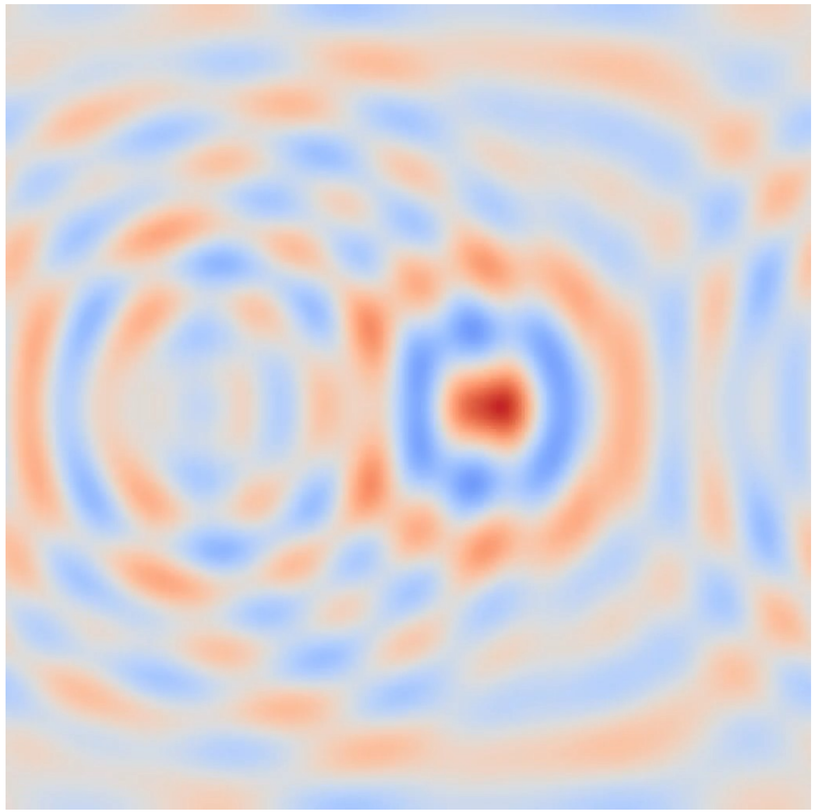
advanced

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

Wave equation

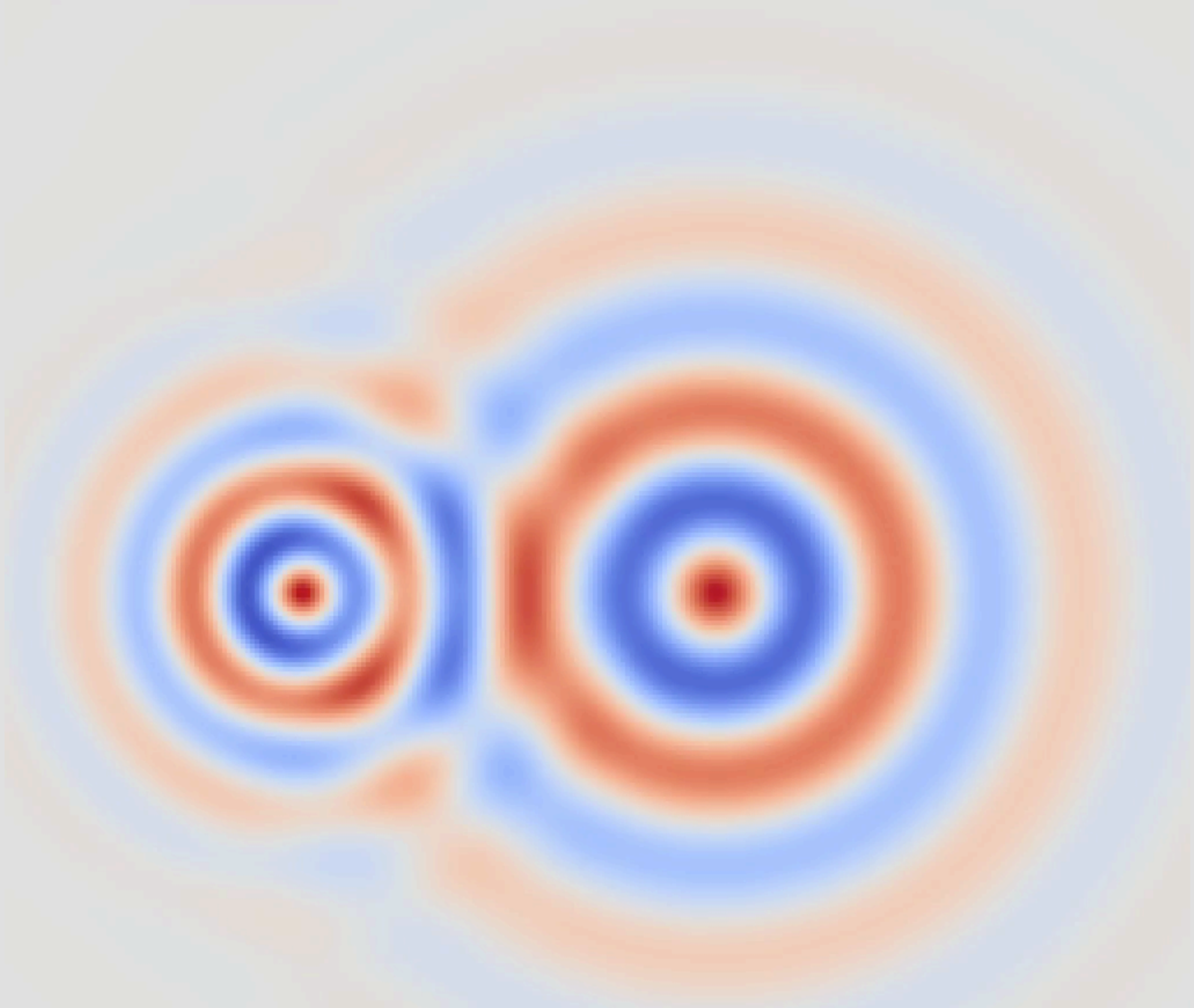


initial

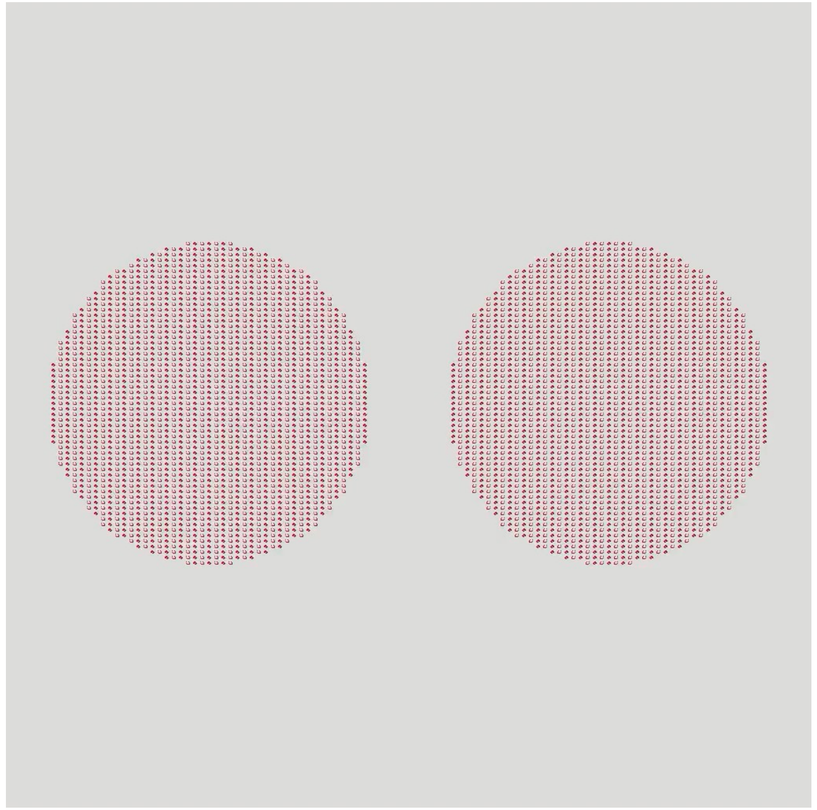


advanced

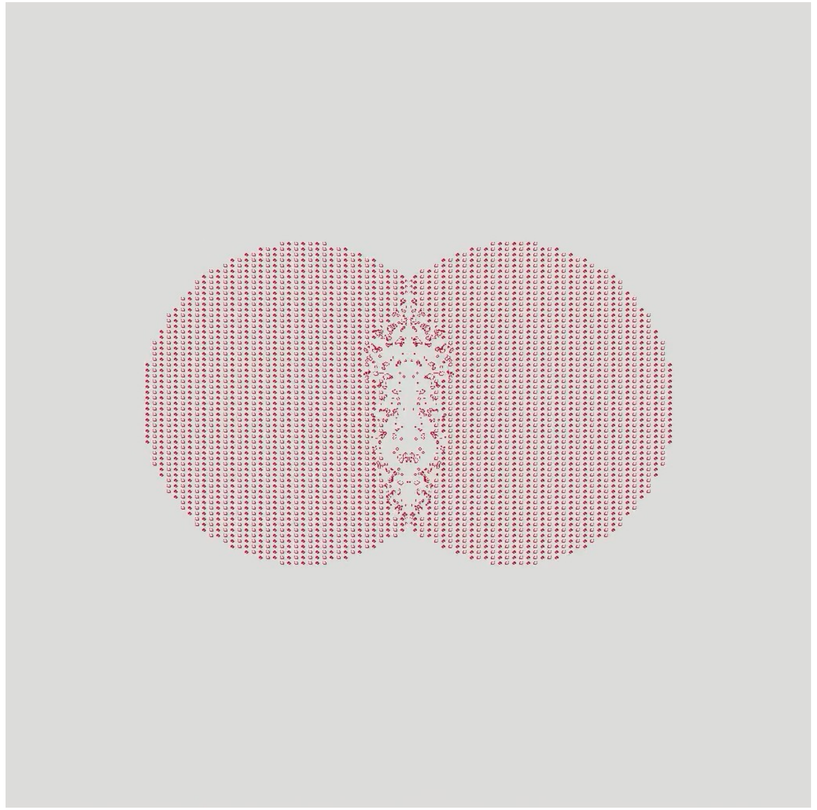
$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$



Game of Life



initial

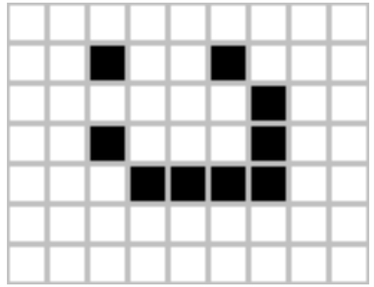


advanced

time stepping rule

old	sum [3x3]	new
0	3	1
0	else	0
1	2 or 3	1
1	else	0

lightweight spaceship



wikipedia