

Lattice gas with the HPP rule

General function returning the state of the gas after time T

```
In[12]:= latticeGasAtTime[initialState_, T_] :=
(*
input:
  initialState =
  the 2-dimensional array of lists containing states in the lattice sites:
  {north, west, south, east, wall}
  where wall = 5 if the site is a part of the boundary or of the wall,
otherwise wall = 0,
  (number 5 is used for convenience,
plotting in grey scales gives then black for walls),
  and north etc = 1 if there is a particle incoming to the site
  in the corresponding direction,
  T = the time up to which to evolve the gas,
output:
  latticeEvolution = the 3-
  dimensional array of evolution of the number of incoming particles for all sites,
  for walls there will be 5,
  finalState = the 2-
  dimensional array of lists containing states in the lattice sites after T iterations
*)
Module[{m, n, dirs, revDirs, latticeState, latticeEvolution, oldConf, newConf},
{m, n} = Dimensions[initialState, 2];
dirs = {{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
revDirs = {3, 4, 1, 2}; (* which directions are opposite *)
(* auxiliary array for evolution of states, *)
latticeState = ConstantArray[0, {2, m, n}];
latticeState[[1]] = initialState;
latticeState[[2]] = initialState;
oldConf = 1;
newConf = 2;
latticeEvolution = ConstantArray[0, {T + 1, m, n}];
latticeEvolution[[1]] = Total[initialState, {3}];
Do[
  (* first change old configuration to deal with collisions *)
  Do[
    If[latticeState[[oldConf, i, j]] == {1, 0, 1, 0, 0},
      latticeState[[oldConf, i, j]] = {0, 1, 0, 1, 0},
      If[latticeState[[oldConf, i, j]] == {0, 1, 0, 1, 0},
        latticeState[[oldConf, i, j]] = {1, 0, 1, 0, 0}]
    ],
    {i, 1, m}, {j, 1, n}
  ];
,
```

```

(* move all particles in the current directions,
or bounce from the walls *)
Do[
  If[ (* there is no wall, check from where particles are coming *)
    latticeState[[oldConf, i, j, 5]] == 0,
    Do[
      If[ (* if there is a wall at the site from which the particle could come *)
        latticeState[[oldConf, i + dirs[[d, 1]], j + dirs[[d, 2]], 5]] == 5,
        (* then if there is a particle incoming to
         a current site from the opposite direction it bounces back*)
        (* If[j==10 &&latticeState[[oldConf,i,j,revDirs[[d]]]]==1,Print["Ted ",it]];*)
        latticeState[[newConf, i, j, d]] = latticeState[[oldConf, i, j, revDirs[[d]]]],
        (* otherwise there could be a particle coming from that site *)
        latticeState[[newConf, i, j, d]] =
          latticeState[[oldConf, i + dirs[[d, 1]], j + dirs[[d, 2]], d]]
      ],
      {d, 1, 4}
    ];
    {i, 1, m}, {j, 1, n}
  ];
  (* finally count particles for each site *)
  latticeEvolution[[it + 1]] = Total[latticeState[[newConf]], {3}];
  {oldConf, newConf} = {newConf, oldConf},
  {it, 1, T}
];
Return[{latticeEvolution, latticeState[[oldConf]]}]
];

```

Gas in the divided box with a slit

```

In[13]:= {m, n} = {50, 100};
(* initial state is given as a two-dimensional array of lists
   containing states in the lattice sites:
   {north, west, south, east, wall}
   where wall = 5 if the site is a part of the boundary or of the wall,
   and north etc = 1 if there is a particle incoming to the site
   in the corresponding direction
*)
iniState = ConstantArray[{0, 0, 0, 0, 5}, {m, n}]; (* wall everywhere *)
(* remove walls from and put particles randomly to
   inner sites of the lattice with probability p *)
p = 0.5;
MyRandSeq[prob_, num_] := Delete[Table[If[RandomReal[] <= prob, 1, 0], num], 0];
Do[
  iniState[[i, j]] = {MyRandSeq[p, 4], 0},
  {i, 2, m - 1}, {j, 2, 49}
];
Do[
  iniState[[i, j]] = {0, 0, 0, 0, 0},
  {i, 2, m - 1}, {j, 51, n - 1}
];
Do[
  iniState[[23 + i, 50]] = {0, 0, 0, 0, 0},
  {i, 1, 5}
];
T = 300;
{lattEvol, lastState} = latticeGasAtTime[iniState, T];
Manipulate[
  ArrayPlot[lattEvol[[it]],
  {it, 1, T + 1, 1}]
]

```



Lattice gas model is reversible

```
In[23]:= iniState2 = lastState;
Do[
  If[iniState2[[i, j]] == {1, 0, 1, 0, 0},
    iniState2[[i, j]] = {0, 1, 0, 1, 0},
    If[iniState2[[i, j]] == {0, 1, 0, 1, 0},
      iniState2[[i, j]] = {1, 0, 1, 0, 0}]
  ];
  {iniState2[[i, j, 1]], iniState2[[i, j, 2]], iniState2[[i, j, 3]], iniState2[[i, j, 4]]} =
    {iniState2[[i, j, 3]], iniState2[[i, j, 4]], iniState2[[i, j, 1]], iniState2[[i, j, 2}}},
  {i, 1, m}, {j, 1, n}
];
{lattEvolBack, firstState} = latticeGasAtTime[iniState2, T];
Manipulate[ArrayPlot[lattEvolBack[[it]]], {it, 1, T+1, 1}]
```



Even very small perturbation destroys reversibility

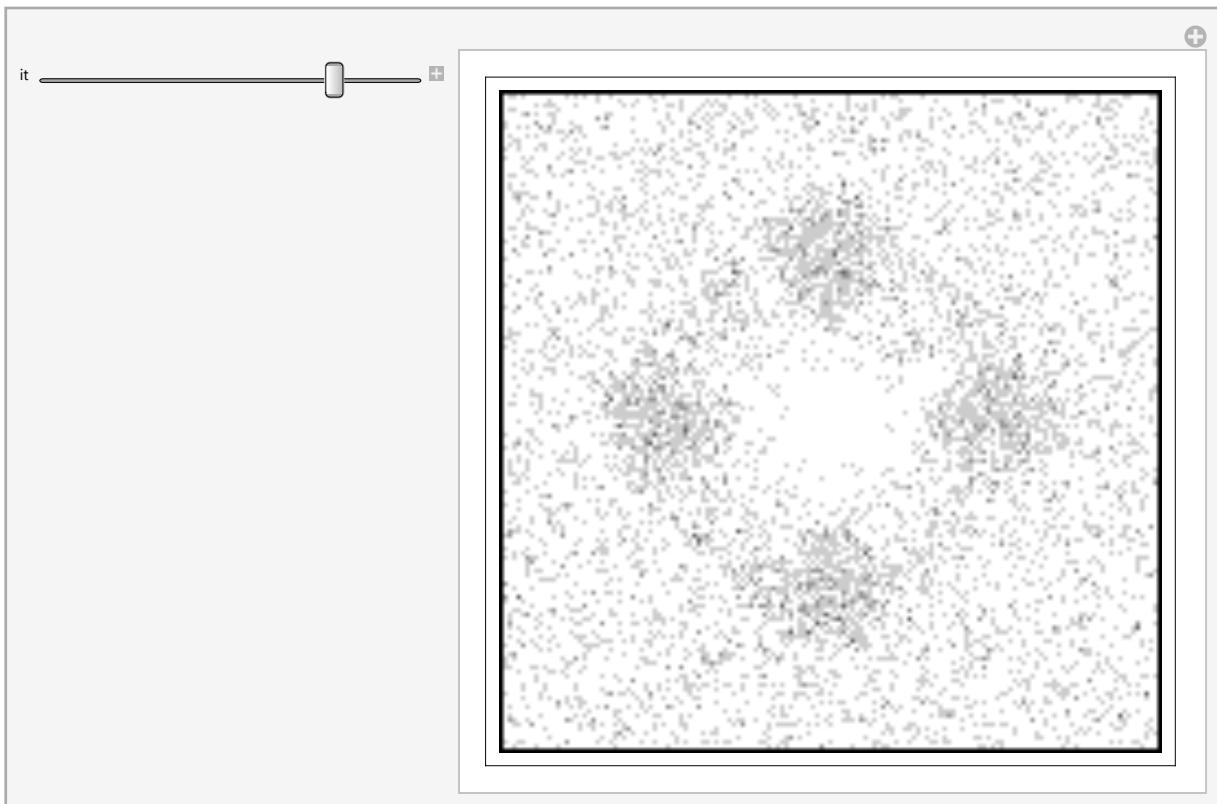
```
In[27]:= iniState2[[20, 60]]
Out[27]= {0, 0, 0, 0, 0}
```

```
In[29]:= iniState3 = iniState2;
iniState3[[20, 60]] = {0, 0, 1, 0, 0};
{lattEvolBackPert, firstStatePert} = latticeGasAtTime[iniState3, T];
Manipulate[ArrayPlot[lattEvolBackPert[[it]]], {it, 1, T + 1, 1}]
```



Waves are not isotropic in the square lattice

```
In[53]:= {m, n} = {151, 151};
middle = Floor[{(m + 1) / 2, (n + 1) / 2}];
iniState4 = ConstantArray[{0, 0, 0, 0, 5}, {m, n}]; (* wall everywhere *)
s = 15; (* size of the initial cluster of molecules in the middle *)
(* remove walls from and put particles randomly to
   inner sites of the lattice with probability p *)
p1 = 0.05;
p2 = 0.6;
MyRandSeq[prob_, num_] := Delete[Table[If[RandomReal[] <= prob, 1, 0], num], 0];
Do[
  iniState4[[i, j]] = {MyRandSeq[p1, 4], 0};
  (* increase the density in the circle in the middle *)
  If[Norm[middle - {i, j}] < s, iniState4[[i, j]] = {MyRandSeq[p2, 4], 0}],
  {i, 2, m - 1}, {j, 2, n - 1}]
];
T = 50;
{lattEvolWave, lastStateWave} = latticeGasAtTime[iniState4, T];
Manipulate[ArrayPlot[lattEvolWave[[it]]], {it, 1, T + 1, 1}]
```



No randomness in the dynamics - symmetric initial

conditions produce symmetric results

```
In[44]:= {m, n} = {51, 51};  
{m2, n2} = Floor[{(m + 1) / 2, (n + 1) / 2}];  
iniState5 = ConstantArray[{0, 0, 0, 0, 5}, {m, n}]; (* wall everywhere *)  
iniState5[[2 ;; m - 1, 2 ;; n - 1]] = {0, 0, 0, 0, 0};  
s = 5;  
iniState5[[m2 - s ;; m2 + s, n2 - s ;; n2 + s]] = {1, 1, 1, 1, 0};  
T = 1000;  
{lattEvolSymm, lastStateSymm} = latticeGasAtTime[iniState5, T];  
Manipulate[ArrayPlot[lattEvolSymm[[it]]], {it, 1, T + 1, 1}]
```

