Simulating Active Matter with subMIT

Sunghan Ro

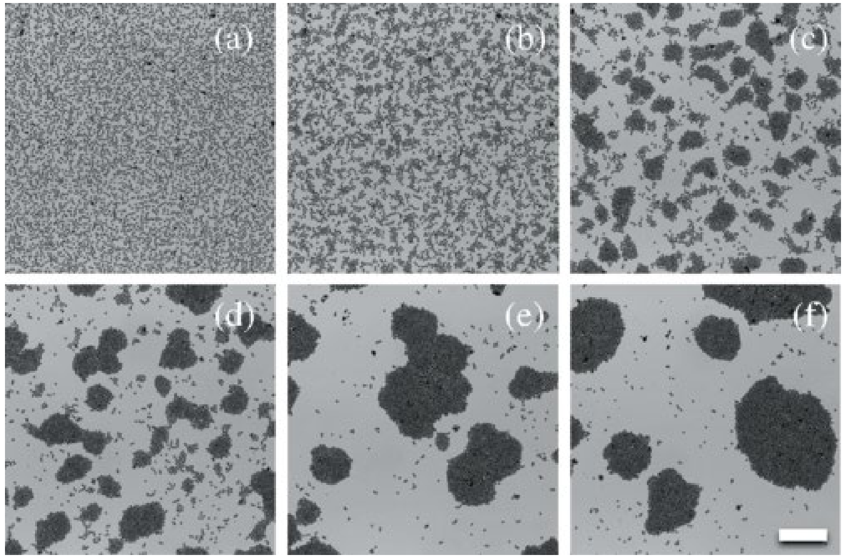
Workshop on Basic Computing Service in the Physics Department

Jan. 6, 2023

Massachusetts
Institute of
Technology

Active matter

Motility-induced phase separation



Van der Linden, PRL, (2019)

Flocking of birds

Modeling active matter

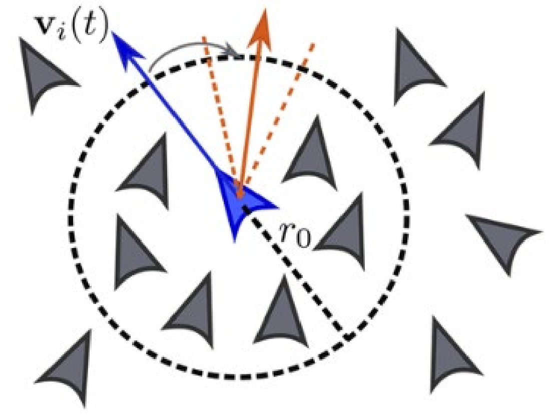


Image by Martin Zumaya

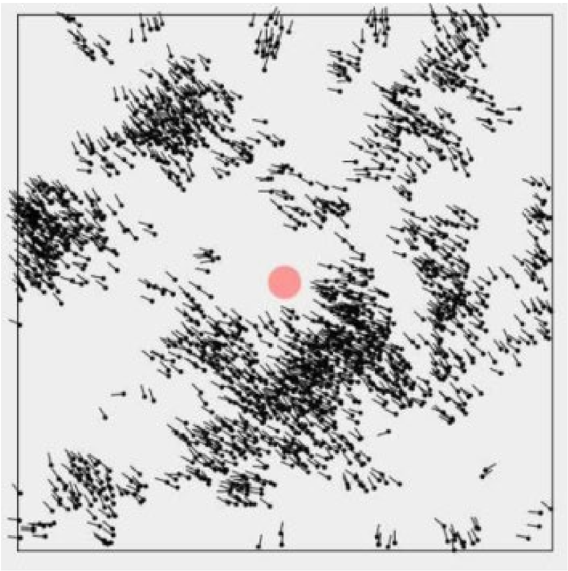
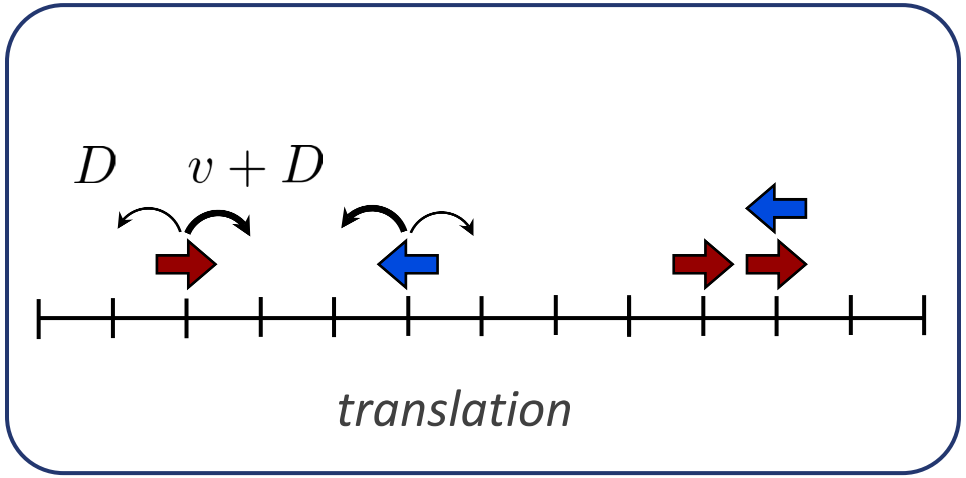
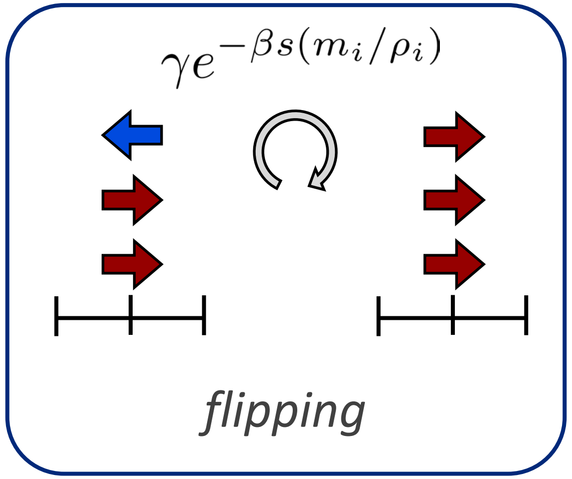


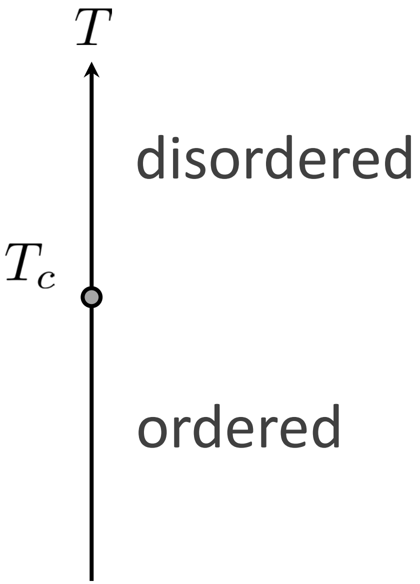
Image by Rogelio A. Hernandez-Lopez

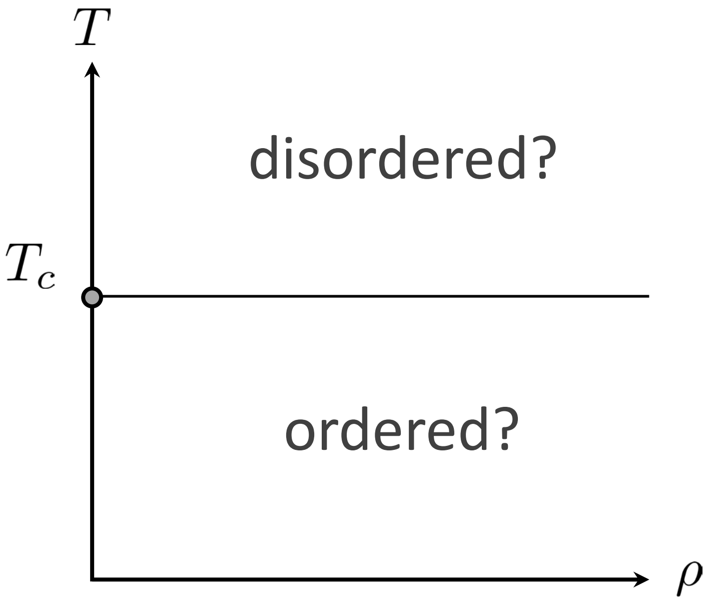
Active Ising Model

A. P. Solon, J. Tailleur, PRL, (2013)

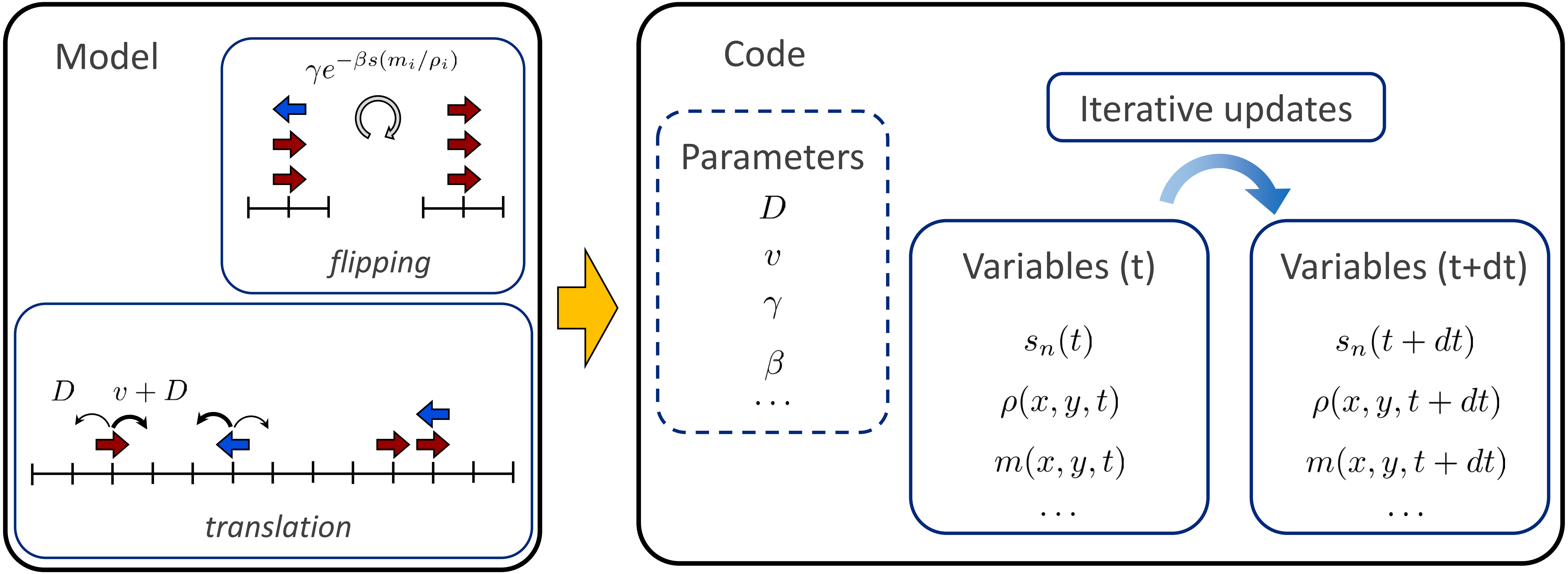








Model to code



Model

$\gamma e^{-\beta s(m_i/\rho_i)}$

flipping

D $v + D$

translation

Code

Iterative updates

Parameters

D

v

γ

β

$\dots$

Variables (t)

$s_n(t)$

$\rho(x, y, t)$

$m(x, y, t)$

$\dots$

Variables (t+dt)

$s_n(t + dt)$

$\rho(x, y, t + dt)$

$m(x, y, t + dt)$

$\dots$

Examples

Parameters

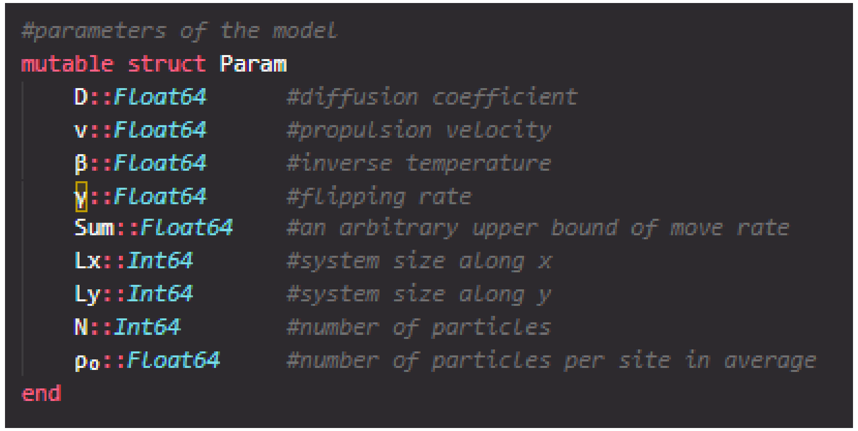
D

v

γ

β

$\dots$



```julia
#parameters of the model
mutable struct Param
    D::Float64      #diffusion coefficient
    v::Float64      #propulsion velocity
    β::Float64      #inverse temperature
    γ::Float64      #flipping rate
    Sum::Float64    #an arbitrary upper bound of move rate
    Lx::Int64       #system size along x
    Ly::Int64       #system size along y
    N::Int64        #number of particles
    ρ₀::Float64     #number of particles per site in average
end
```

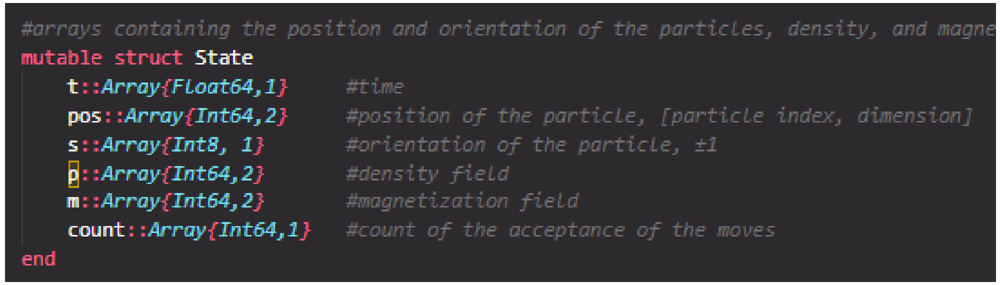
Variables (t)

$s_n(t)$

$\rho(x, y, t)$

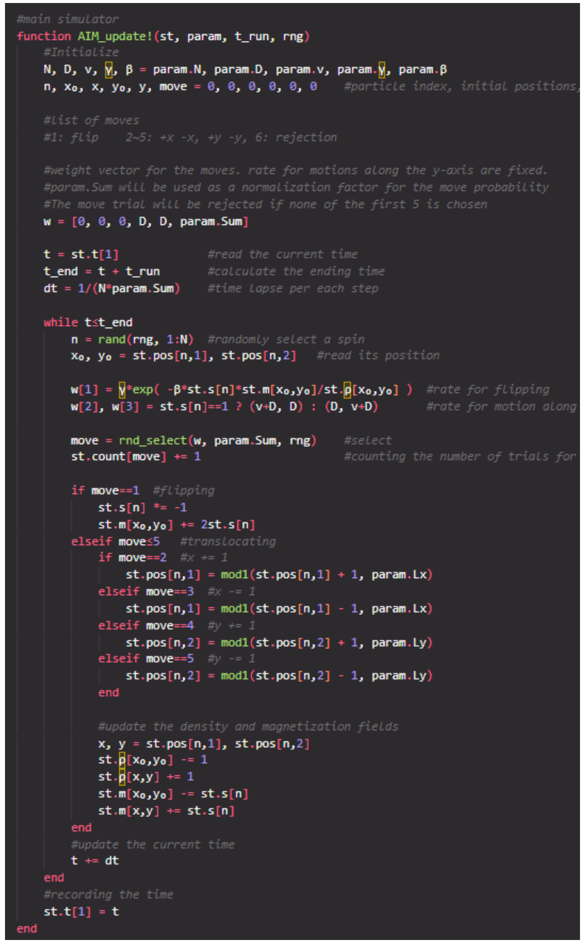
$m(x, y, t)$

$\dots$



```julia
#arrays containing the position and orientation of the particles, density, and magne
mutable struct State
    t::Array{Float64,1}     #time
    pos::Array{Int64,2}     #position of the particle, [particle index, dimension]
    s::Array{Int8, 1}       #orientation of the particle, ±1
    ρ::Array{Int64,2}       #density field
    m::Array{Int64,2}       #magnetization field
    count::Array{Int64,1}   #count of the acceptance of the moves
end
```

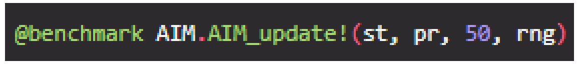
Iterative updates



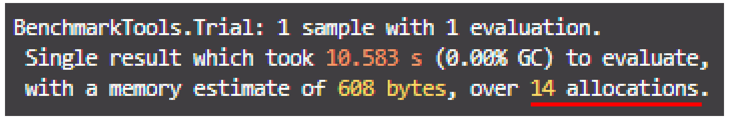
```julia
#main simulator
function AIM_update!(st, param, t_run, rng)
    #Initialize
    N, D, v, γ, β = param.N, param.D, param.v, param.γ, param.β
    n, x₀, x, y₀, y, move = 0, 0, 0, 0, 0, 0    #particle index, initial positions,

    #List of moves
    #1: flip    2~5: +x -x, +y -y, 6: rejection

    #weight vector for the moves. rate for motions along the y-axis are fixed.
    #param.Sum will be used as a normalization factor for the move probability
    #The move trial will be rejected if none of the first 5 is chosen
    w = [0, 0, 0, D, D, param.Sum]

    t = st.t[1]             #read the current time
    t_end = t + t_run       #calculate the ending time
    dt = 1/(N*param.Sum)    #time lapse per each step

    while t≤t_end
        n = rand(rng, 1:N)  #randomly select a spin
        x₀, y₀ = st.pos[n,1], st.pos[n,2]   #read its position

        w[1] = γ*exp( -β*st.s[n]*st.m[x₀,y₀]/st.ρ[x₀,y₀] )  #rate for flipping
        w[2], w[3] = st.s[n]==1 ? (v+D, D) : (D, v+D)       #rate for motion along

        move = rnd_select(w, param.Sum, rng)    #select
        st.count[move] += 1                     #counting the number of trials for

        if move==1  #flipping
            st.s[n] *= -1
            st.m[x₀,y₀] += 2st.s[n]
        elseif move≤5   #translocating
            if move==2  #x += 1
                st.pos[n,1] = mod1(st.pos[n,1] + 1, param.Lx)
            elseif move==3  #x -= 1
                st.pos[n,1] = mod1(st.pos[n,1] - 1, param.Lx)
            elseif move==4  #y += 1
                st.pos[n,2] = mod1(st.pos[n,2] + 1, param.Ly)
            elseif move==5  #y -= 1
                st.pos[n,2] = mod1(st.pos[n,2] - 1, param.Ly)
            end

            #update the density and magnetization fields
            x, y = st.pos[n,1], st.pos[n,2]
            st.ρ[x₀,y₀] -= 1
            st.ρ[x,y] += 1
            st.m[x₀,y₀] -= st.s[n]
            st.m[x,y] += st.s[n]
        end
        #update the current time
        t += dt
    end
    #recording the time
    st.t[1] = t
end
```

Performance analysis

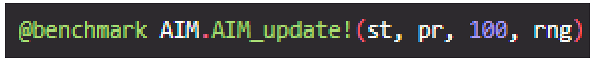
Benchmarking



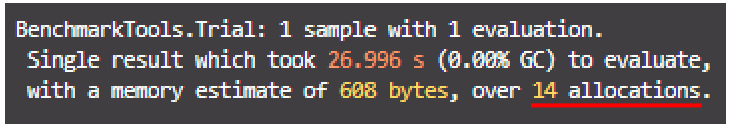
```
@benchmark AIM.AIM_update!(st, pr, 50, rng)
```



```
BenchmarkTools.Trial: 1 sample with 1 evaluation.
 Single result which took 10.583 s (0.00% GC) to evaluate,
 with a memory estimate of 608 bytes, over 14 allocations.
```

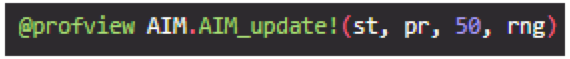


```
@benchmark AIM.AIM_update!(st, pr, 100, rng)
```

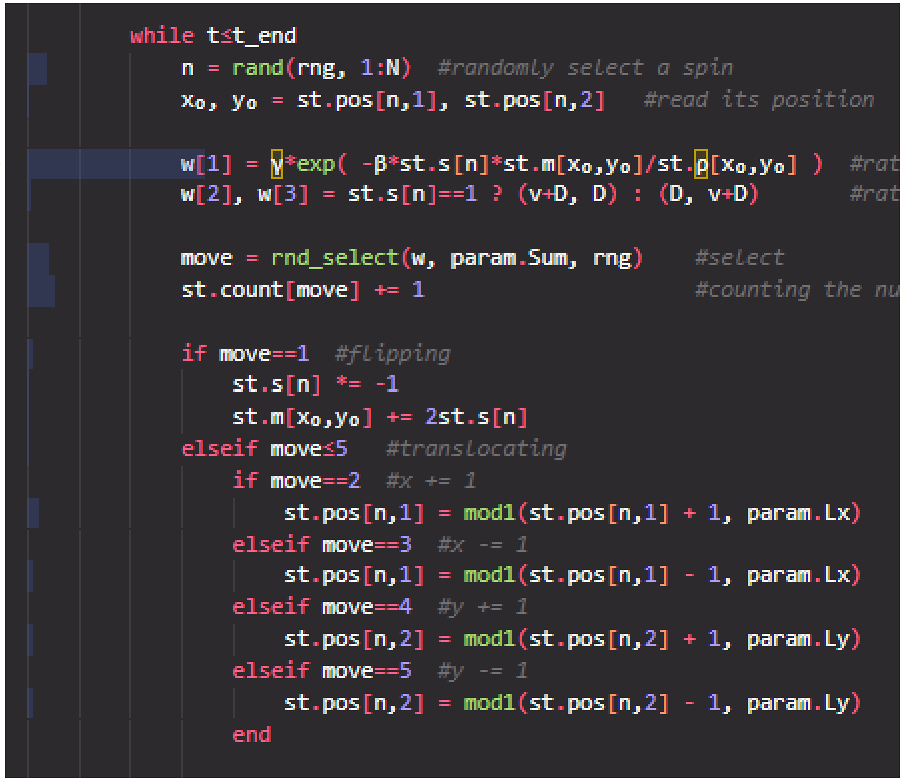


```
BenchmarkTools.Trial: 1 sample with 1 evaluation.
 Single result which took 26.996 s (0.00% GC) to evaluate,
 with a memory estimate of 608 bytes, over 14 allocations.
```

Profiling

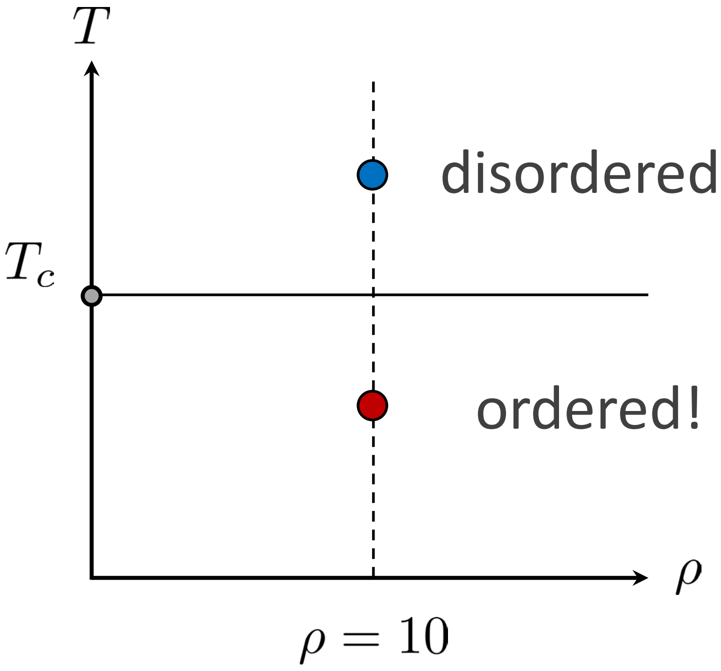


```
@profview AIM.AIM_update!(st, pr, 50, rng)
```



```
while t≤t_end
    n = rand(rng, 1:N)  #randomly select a spin
    x₀, y₀ = st.pos[n,1], st.pos[n,2]   #read its position

    w[1] = γ*exp( -β*st.s[n]*st.m[x₀,y₀]/st.ρ[x₀,y₀] )  #rat
    w[2], w[3] = st.s[n]==1 ? (v+D, D) : (D, v+D)       #rat

    move = rnd_select(w, param.Sum, rng)      #select
    st.count[move] += 1                       #counting the nu

    if move==1  #flipping
        st.s[n] *= -1
        st.m[x₀,y₀] += 2st.s[n]
    elseif move≤5   #translocating
        if move==2  #x += 1
            st.pos[n,1] = mod1(st.pos[n,1] + 1, param.Lx)
        elseif move==3  #x -= 1
            st.pos[n,1] = mod1(st.pos[n,1] - 1, param.Lx)
        elseif move==4  #y += 1
            st.pos[n,2] = mod1(st.pos[n,2] + 1, param.Ly)
        elseif move==5  #y -= 1
            st.pos[n,2] = mod1(st.pos[n,2] - 1, param.Ly)
        end
```

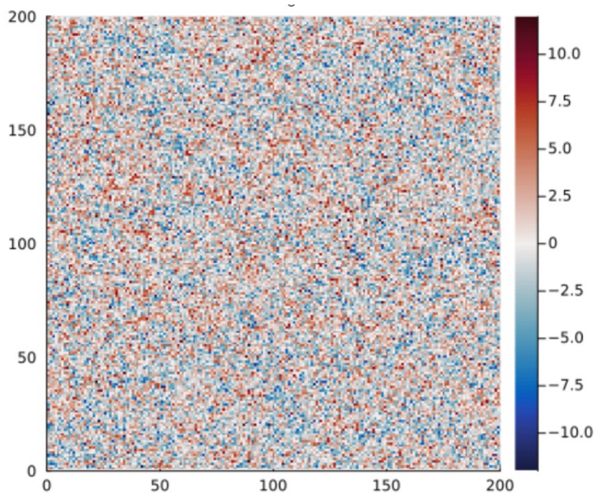
Visualizations



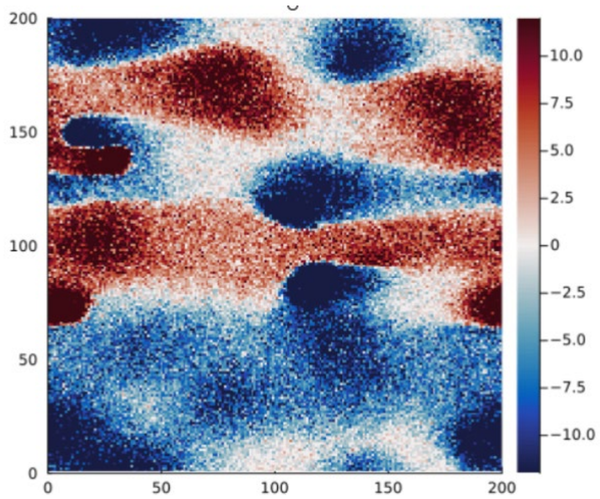
disordered!

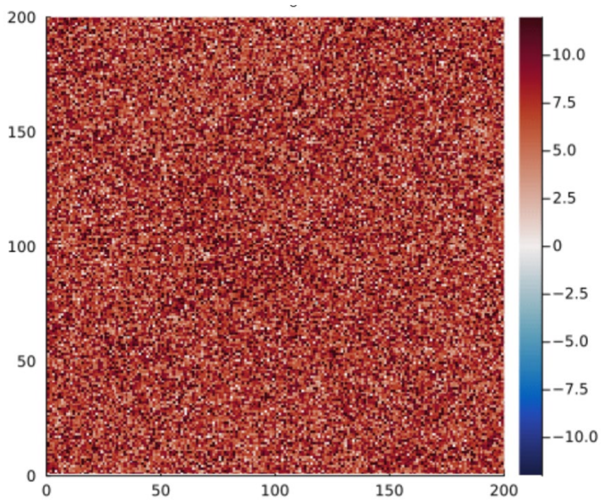
ordered!

●



●

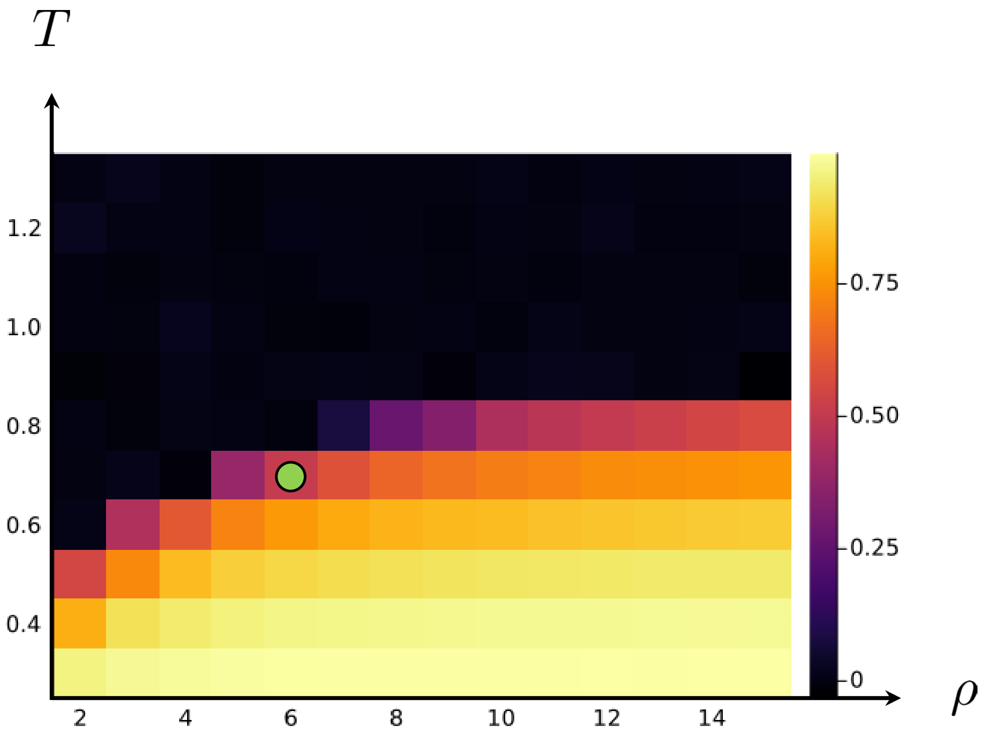


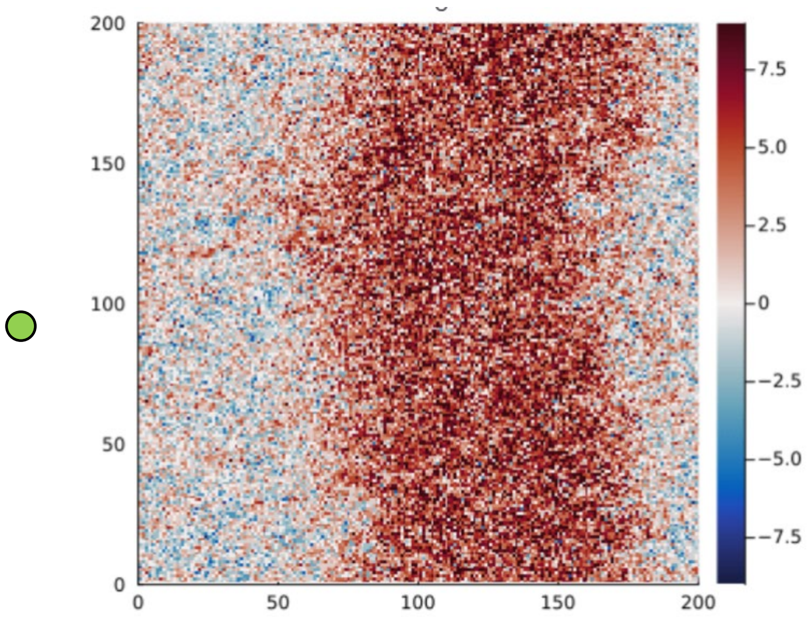


Parameter scanning using submit cluster

Submitted to subMIT via Slurm, script based on *submit-examples/Julia/Slurm*

Phase separation between the disordered and ordered phase in the low-density regime.





The code will be submitted to submit-examples

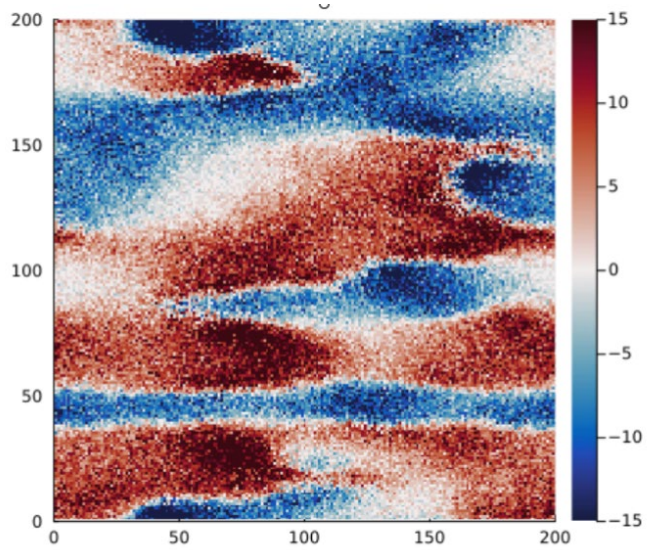
Questions?

sunghan@mit.edu

References

A. P. Solon and J. Tailleur, Phys. Rev. Lett. **111**, 078101 (2013)

A. P. Solon and J. Tailleur, Phys. Rev. E **92**, 042119 (2015)



200
150
100
50
0
0
50
100
150
200
15
10
5
0
−5
−10
−15