Manual of pre-built

(Dated: March 31, 2018)

Manual of pre-built

I. RUN SIMULATION

The number of the lattice are 512×512.

Before run simulation, one have to set the exchange strength, magnetic momentum and boundary condition, using the button shown in Fig. 1.

The exchange strength is a lua script generate J for every lattice index. It is introduced in Sec. II.

The magnetic momentum can be either a lua script or a image file introduced in Sec. III.

The boundary condition is a image file introduced in Sec. IV.

Then set the parameters as shown in Fig. 2.

The simulation can be paused at any time, when paused, all the parameters can be changed. The simulation can also been stopped automatically. It is shown in Fig. 3. When $step = stop\ at\ step$, it will stop, and when $step \bmod save\ every\ step = 0$, the data is saved automatically. Using the configure in Fig. 3, the simulation will not stop automatically, and the state of the magnetic momentum will be saved when steps are $0, 30000, 60000, \ldots$. The output files are introduced in Sec. V.

The configuration of above will generate results shown in Fig. 4.

II. EXCHANGE STRENGTH

The exchange strength is a lua script with a function 'GetJValueByLatticeIndex', and return a table with the function. For example, a constant exchange strength $J = 2$ can be written as

```lua
-- Exchange Strength is constant
function GetJValueByLatticeIndex(x, y)
    return 2.0
end

-- Need to register the function
return {
```

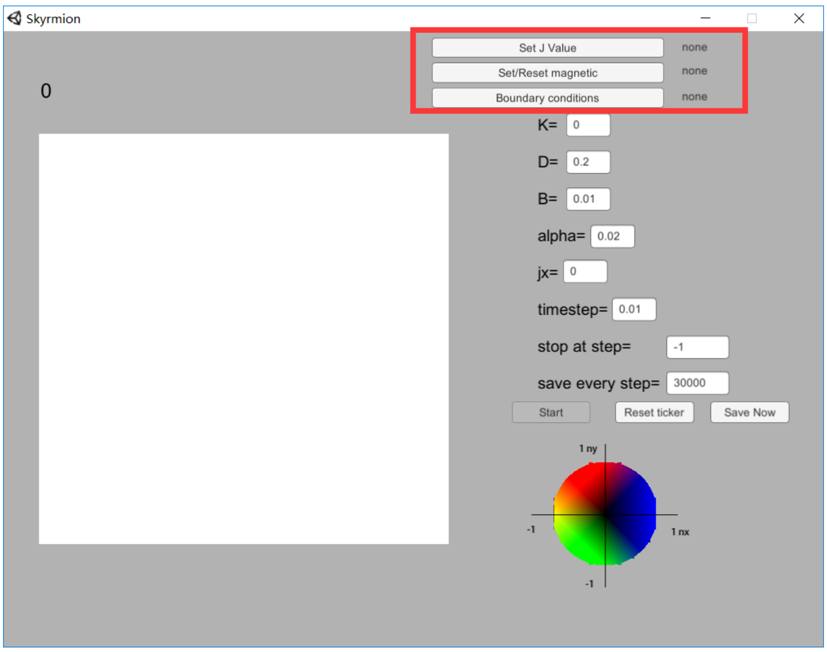


FIG. 1: Set the exchange strength, magnetic momentum and boundary condition.

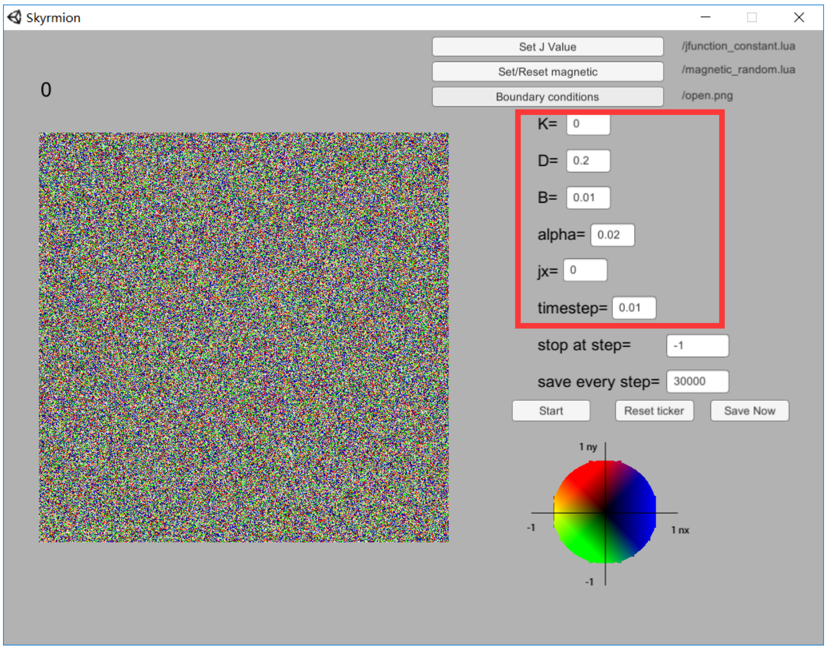


FIG. 2: Set other parameters.

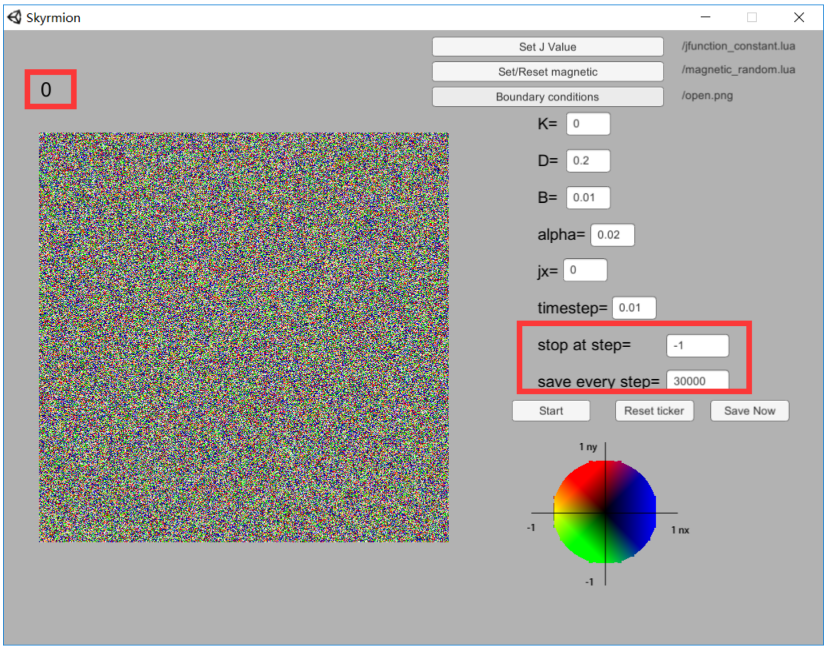
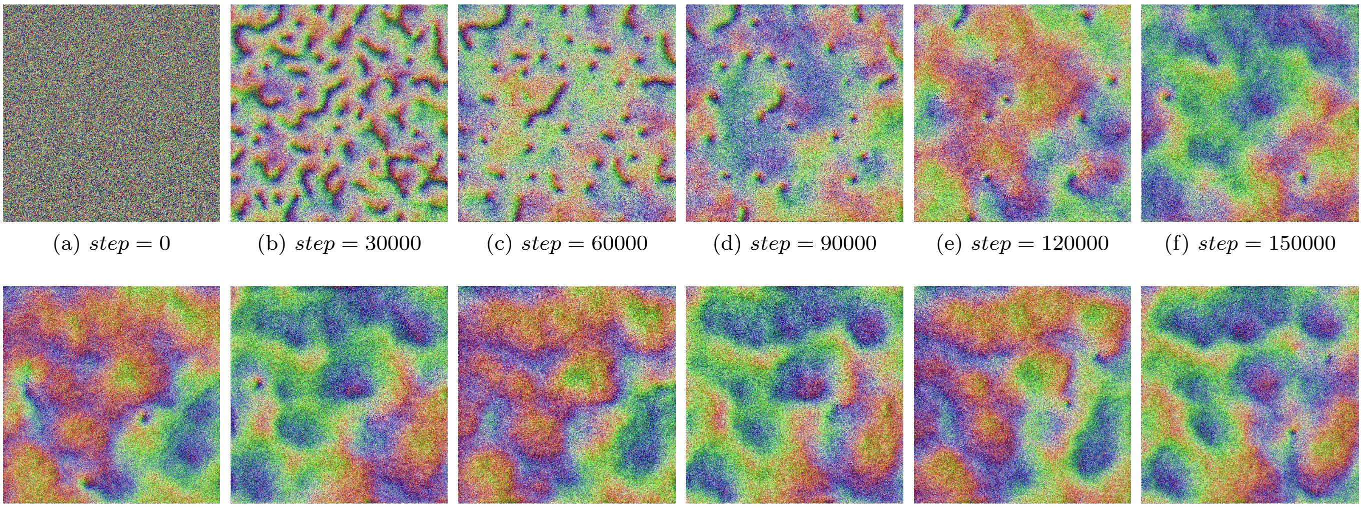


FIG. 3: Set stop steps. The number in left-up corner is the number of step.

```
    GetJValueByLatticeIndex = GetJValueByLatticeIndex,
}
```

while a pin with $J = 1 + \exp\left(-0.001\rho^2\right)$ at lattice index $(255, 255)$ can be written as

```
-- Exchange Strength is pin
function GetJValueByLatticeIndex(x, y)
    local j0 = 1
    local j1 = 1
    local j2 = 0.001
    local rho = (x - 255) * (x - 255) + (y - 255) * (y - 255)
```



(a) $step = 0$ (b) $step = 30000$ (c) $step = 60000$ (d) $step = 90000$ (e) $step = 120000$ (f) $step = 150000$

(g) $step = 180000$ (h) $step = 210000$ (i) $step = 240000$ (j) $step = 270000$ (k) $step = 300000$ (l) $step = 330000$

FIG. 4: Results

```

    return j0 + j1 * math.exp(-1.0 * j2 * rho)
end

-- Need to register the function
return {
    GetJValueByLatticeIndex = GetJValueByLatticeIndex,
}
```

The lua files are put in 'LuaScript' sub-folder in the folder of the executable file.

III. MAGNETIC MOMENTUM

The magnetic momentums can be initialized or reset using a lua script, with a function 'GetMagneticByLatticeIndex', and return the table with the function.

For example, random magnetic momentums can be written as

```
-- Initial the magnetic by random
function GetMagneticByLatticeIndex(x, y)
    local nx = math.random() * 2.0 - 1.0
    local ny = math.random() * 2.0 - 1.0
    local nz = math.random() * 2.0 - 1.0
    local length_inv = 1.0 / math.sqrt(nx * nx + ny * ny + nz * nz)
    length_inv = math.max(length_inv, 0.000000001)
    return nx * length_inv, ny * length_inv, nz * length_inv
end

-- Need to register the function
return {
    GetMagneticByLatticeIndex = GetMagneticByLatticeIndex,
}
```

The magnetic momentums pointing up can be written as

```
-- Initial the magnetic by point up
function GetMagneticByLatticeIndex(x, y)
    return 0, 0, 1
end

-- Need to register the function
```

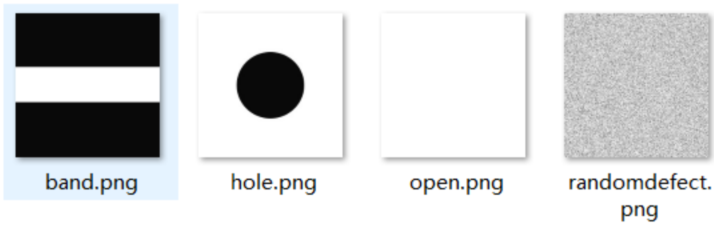


FIG. 5: Boundary conditions.

```
return {
    GetMagneticByLatticeIndex = GetMagneticByLatticeIndex,
}
```

The magnetic momentums of a skyrmion at $100, 255$ can be written as

```
-- Initial the magnetic by random
function GetMagneticByLatticeIndex(x, y)
    -- skyrmion at position 100, 255
    local rho = (x - 100) * (x - 100) + (y - 255) * (y- 255)
    -- radius 20
    local theta = math.pi * math.exp(- rho / (20 * 20))
    local phi = math.atan2(x - 100, y - 255)

    local nx = math.cos(phi) * math.sin(theta)
    local ny = math.sin(phi) * math.sin(theta)
    local nz = math.cos(theta)

    return nx, ny, nz
end

-- Need to register the function
return {
    GetMagneticByLatticeIndex = GetMagneticByLatticeIndex,
}
```

The magnetic momentum can also been initialized or reset using a 512×512 image, with color $R = n_x \times 0.5 + 0.5$, $G = n_x \times 0.5 + 0.5$, $B = n_x \times 0.5 + 0.5$. This is also the file format of 'raw image' of the output introduced in Sec. V. So one can load, for example 'month-day-hour-munite_step_raw.png' and set or reset the magnetic momentum with the result simulated before.

IV. BOUNDARY CONDITION

The boundary condition are 512×512 image files. If the red of the color of the image file is less then 0.5, the lattice is considered as not exist. For example, if the color at the $5, 5$ pixel is black, the magnetic momentum at lattice index $5, 5$ is always $\mathbf{n} = (0, 0, 0)$.

There are 4 common boundary condition files in 'BoundConditions' sub-folder in the folder of the executable file, they are shown in Fig. 5.

V. OUTPUT

The output files are put in 'Output' sub-folder in the folder of the executable file.

Whenever the save button is pressed or the autosave is triggered, there will be 4 files generated, with the name 'month-day-hour-minute_step_show.png', 'month-day-hour-minute_step_raw.png', 'month-day-hour-minute_step_pic.png' and 'month-day-hour-minute_step_prof.txt', as shown in Fig. 6.

The 'show.png' is the image shown in the program.

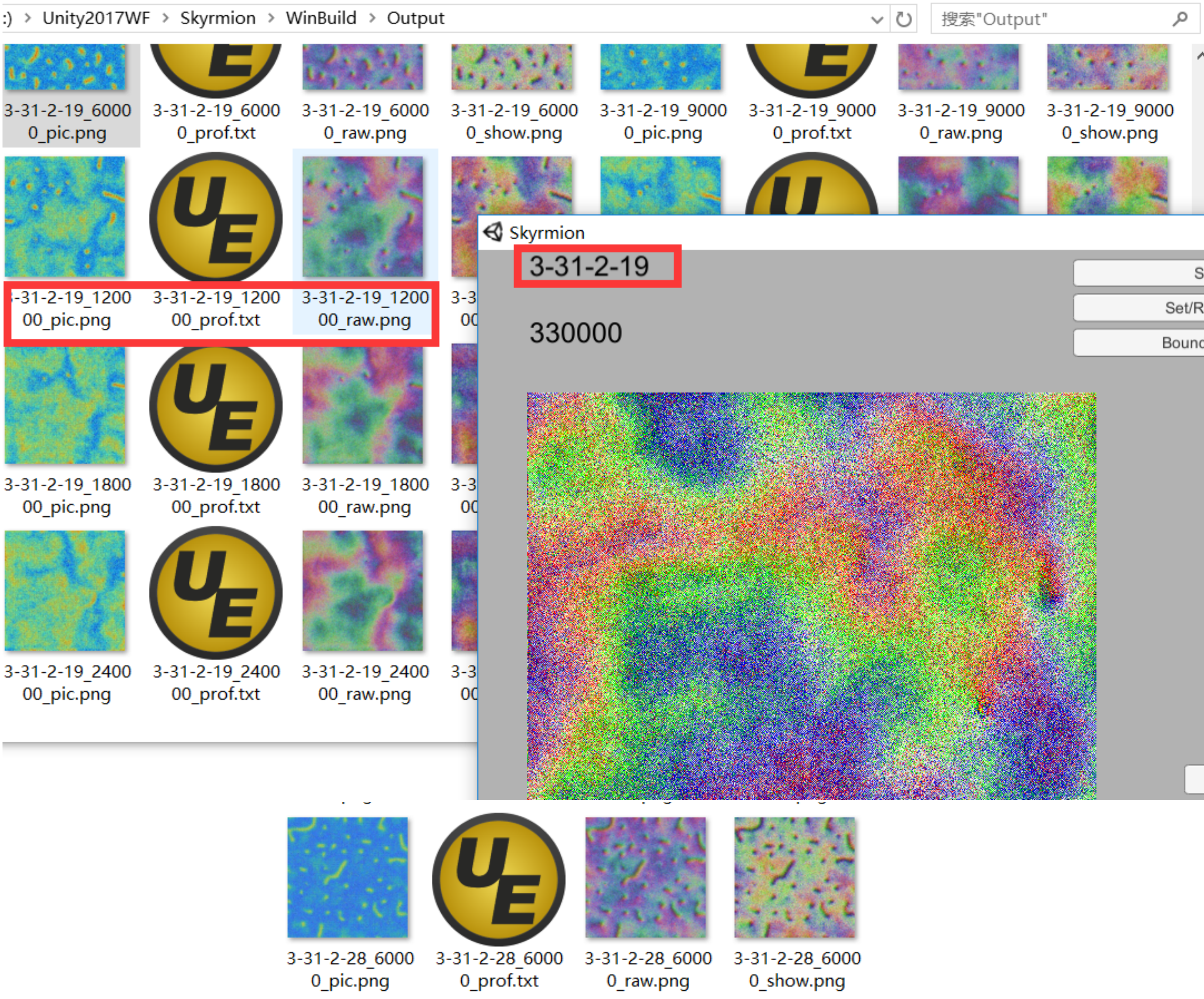


FIG. 6: Output files.

The 'raw.png' is a image with the RGB color set as $r = 0.5 \times n_x + 0.5$, $g = 0.5 \times n_y + 0.5$ and $b = 0.5 \times n_z + 0.5$. Which can be used as input in for example matlab.

For example, the code

```
imds = imageDatastore({'3-30-13-55_90000_raw.png'});
img = readimage(imds,1);

nz=zeros(512,512);
newc=uint8(zeros(506,506,3));
for i=1:512
    for j=1:512
        nz(i,j)=2.0*(double(img(i,j,3))/255.0) - 1.0;
        if i > 3 && j > 3 && i < 510 && j < 510
            newc(i - 3, j - 3, 1) = img(i, j, 1);
            newc(i - 3, j - 3, 2) = img(i, j, 2);
            newc(i - 3, j - 3, 3) = img(i, j, 3);
        end
    end
```

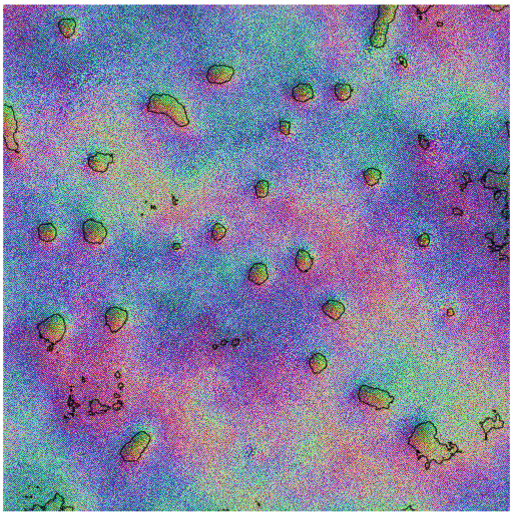


FIG. 7: Possible edge (NOTE: NOT Accurate!).

```
end

%apply blur to remove small points
intImage = integralImage(nz);
avgH = integralKernel([1 1 7 7], 1/49);
nzblured = integralFilter(intImage, avgH);

%cos(pi*exp(-0.25))
x = 1:1:506;
y = 1:1:506;
[X,Y] = meshgrid(x,y);
Z = nzblured;
v = [cos(pi*exp(-0.7)),cos(pi*exp(-0.7))];

figure
imshow(newc)
hold on;
contour(X,Y,Z,v,'linecolor','k')
```

will detect possible edge of a skyrmion, and the result is shown in Fig. 7.

The content of the 'prof.txt‘ is the configuration of the simulation, for example

```
start time=03-31-2018 02:28:46
step=120000
j value=/jfunction_constant.lua
initla magnetic=/magnetic_random.lua
boundary condition=/open.png
K=0
D=0.2
B=0.01
Gilbert alpha=0.02
Electric current jx=0
time step=0.01
```

The 'pic.png‘ is a illustration of the magnetic momentums. Part of it is shown in Fig.8

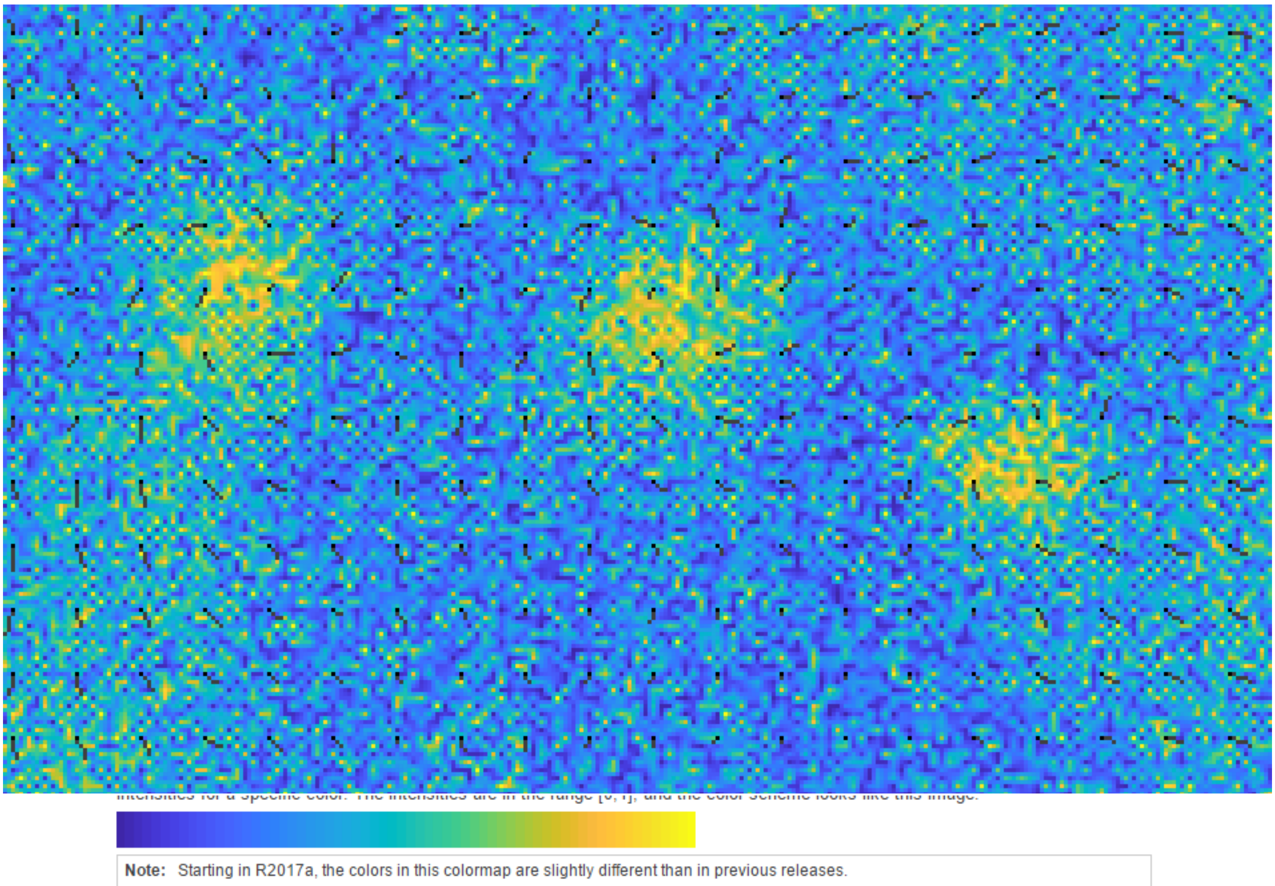


FIG. 8: Illustration of the magnetic momentums. The color shows n_z. The color map is 'parula' in Matlab. The small black lines show (n_x, n_y).