```
function nonrevNNI2DE(theta,plotRun,calcIndicators,n)
%% Non-reversible 2D Nearest-Neighbor Ising model
% WARNING: This implementation only works for some special cases

% --------------------------------------------------------------------
%% Initialise variables

%Check number of input arguments
if nargin < 1
    theta = 0.01;
    plotRun = 0;
    calcIndicators = 0;
    n = 5000;
end

if nargin < 2
    plotRun = 0;
    calcIndicators = 0;
    n = 5000;
end

if nargin < 3
    calcIndicators = 0;
    n = 5000;
end

if nargin < 4
    n = 5000;
end

%Number of Samples
%n = 10000;

%Number of Spins (quadratic/even number)
%N = 64;
N = 225;
%N = 900;
%N = 2500;
NN = sqrt(N);

%Temperatur and Inverse Temperatur
T = 2/log(1 + sqrt(2));
beta = 1/T;

%Exchange Energy
J = 1;

%Theta
%theta = 0.001;

%Magnetization
M = 0;
iniE = 9;

%Prepare 'good' starting checkerboard
while abs(iniE) > 8

%Prepare checkerboard with magnetization
if M == 0
   C = ones(1,N);
```

```
   B = randperm(numel(C));
   C(B(1:ceil(N/2))) = -1;
   A = reshape(C,NN,NN);
else
   m = N/2 - sign(M)*M/2;
   C = sign(M)*ones(1,N);
   B = randperm(numel(C));
   C(B(1:m)) = -sign(M);
   A = reshape(C,NN,NN);
end

%Energy level depending on checkerboard
posmovM = zeros(5,N+1);    %(-8,-4,0,4,8) possible moves, remember moves
deltaE = [8 4 0 4 8];      %abs
iniE = 0;

for i = 1:NN
    for j = 1:NN

        %Determine possible move
        dE = readE(A,i,j);

        %Remember possible move
        if  dE== -8
            posmovM(1,1) = posmovM(1,1) + 1;
            r = 2*posmovM(1,1);
            posmovM(1,r)     = i;
            posmovM(1,r + 1) = j;
            iniE = iniE - 4;
        elseif dE == -4
            posmovM(2,1) = posmovM(2,1) + 1;
            r = 2*posmovM(2,1);
            posmovM(2,r)     = i;
            posmovM(2,r + 1) = j;
            iniE = iniE - 2;
        elseif dE == 0
            posmovM(3,1) = posmovM(3,1) + 1;
            r = 2*posmovM(3,1);
            posmovM(3,r)     = i;
            posmovM(3,r + 1) = j;
        elseif dE == 4
            posmovM(4,1) = posmovM(4,1) + 1;
            r = 2*posmovM(4,1);
            posmovM(4,r)     = i;
            posmovM(4,r + 1) = j;
            iniE = iniE + 2;
        elseif dE == 8
            posmovM(5,1) = posmovM(5,1) + 1;
            r = 2*posmovM(5,1);
            posmovM(5,r)     = i;
            posmovM(5,r + 1) = j;
            iniE = iniE + 4;
        else
            error('Error while initialising Checkerboard!');
        end

    end
end

end %while
```

```
%Samples (#,value/dim,stepsize/direction)
samples = zeros(n,3);
samples(1,1) = iniE;

%Prepare direction and stepsize
dir = randsrc;
samples(1,3) = dir;
if dir == 1
    step = randi(3) + 2;
    samples(1,2) = step;
else
    step = randi(3);
    samples(1,2) = step;
end

%Accepted Samples in Step B
accepted = 1;

%Prepare figure
x = -N:4:N;
close all;
figure;
set(gcf, 'Position', get(0,'Screensize'));

% ------------------------------------------------------------------------
%% Calculate Samples and Draw Checkerboard

l = 2;
while l < n

    %Step B ----------------------------------------
    %Read last sample -> propose new sample
    E = samples(l-1,1);
    step = samples(l-1,2);
    absdeltaE = deltaE(step);
    dir = samples(l-1,3);
    proposal = E + dir*absdeltaE;

    %Update checkerboard ----------------------------
    %Check if move is possible
    if posmovM(step,1) == 0
       acc = 0;
    %Randomly flip spin and update posmov
    else
       acc = 1;
       posmov = posmovM;
       %Possible moves before
       before = posmov(step,1);

       %Pick random coordinate
       rr = 2*randi(posmov(step,1));
       i = posmov(step,rr);
       j = posmov(step,rr + 1);

       %Delete from posmov and add to counter side
       if step == 1
          posmov(1,1) = posmov(1,1) - 1;
          posmov(1,rr:end) = [posmov(1,rr+2:end) 0 0];
          posmov(5,1) = posmov(5,1) + 1;
```

```
   r2 = 2*posmov(5,1);
   posmov(5,r2)     = i;
   posmov(5,r2 + 1) = j;
elseif step == 2
   posmov(2,1) = posmov(2,1) - 1;
   posmov(2,rr:end) = [posmov(2,rr+2:end) 0 0];
   posmov(4,1) = posmov(4,1) + 1;
   r2 = 2*posmov(4,1);
   posmov(4,r2)     = i;
   posmov(4,r2 + 1) = j;
elseif step == 3
   %Nothing to do
elseif step == 4
   posmov(4,1) = posmov(4,1) - 1;
   posmov(4,rr:end) = [posmov(4,rr+2:end) 0 0];
   posmov(2,1) = posmov(2,1) + 1;
   r2 = 2*posmov(2,1);
   posmov(2,r2)     = i;
   posmov(2,r2 + 1) = j;
elseif step == 5
   posmov(5,1) = posmov(5,1) - 1;
   posmov(5,rr:end) = [posmov(5,rr+2:end) 0 0];
   posmov(1,1) = posmov(1,1) + 1;
   r2 = 2*posmov(1,1);
   posmov(1,r2)     = i;
   posmov(1,r2 + 1) = j;
end

%Look at sorrounding spins
up    = mod(i-2,NN) + 1;
down  = mod(i,NN)   + 1;
left  = mod(j-2,NN) + 1;
right = mod(j,NN)   + 1;

dd = [up j down j i left i right];

for ll = 1:2:7

    %Read surrounding energies
    E1 = readE(A,dd(ll),dd(ll+1));
    A(i,j) = -sign(A(i,j));
    E2 = readE(A,dd(ll),dd(ll+1));

    %Add and remove from posmov
    if E1 == -8
       posmov(1,1) = posmov(1,1) - 1;
       r = searchIndex(posmov(1,2:end),dd(ll),dd(ll+1)) + 1;
       posmov(1,r:end) = [posmov(1,r+2:end) 0 0];
       posmov(2,1) = posmov(2,1) + 1;
       r2 = 2*posmov(2,1);
       posmov(2,r2)     = dd(ll);
       posmov(2,r2 + 1) = dd(ll+1);
    elseif E1 == -4
       posmov(2,1) = posmov(2,1) - 1;
       r = searchIndex(posmov(2,2:end),dd(ll),dd(ll+1)) + 1;
       posmov(2,r:end) = [posmov(2,r+2:end) 0 0];
         if E2 == -8
            posmov(1,1) = posmov(1,1) + 1;
            r2 = 2*posmov(1,1);
            posmov(1,r2)     = dd(ll);
```

```
            posmov(1,r2 + 1) = dd(ll+1);
         elseif E2 == 0
            posmov(3,1) = posmov(3,1) + 1;
            r2 = 2*posmov(3,1);
            posmov(3,r2)     = dd(ll);
            posmov(3,r2 + 1) = dd(ll+1);
         else
            error('err3');
         end
      elseif E1 == 0 && E2 ~= 0
          posmov(3,1) = posmov(3,1) - 1;
          r = searchIndex(posmov(3,2:end),dd(ll),dd(ll+1)) + 1;
          posmov(3,r:end) = [posmov(3,r+2:end) 0 0];
           if E2 == 4
            posmov(4,1) = posmov(4,1) + 1;
            r2 = 2*posmov(4,1);
            posmov(4,r2)     = dd(ll);
            posmov(4,r2 + 1) = dd(ll+1);
         elseif E2 == -4
            posmov(2,1) = posmov(2,1) + 1;
            r2 = 2*posmov(2,1);
            posmov(2,r2)     = dd(ll);
            posmov(2,r2 + 1) = dd(ll+1);
         else
            error('err3');
         end
      elseif E1 == 4
         posmov(4,1) = posmov(4,1) - 1;
         r = searchIndex(posmov(4,2:end),dd(ll),dd(ll+1)) + 1;
         posmov(4,r:end) = [posmov(4,r+2:end) 0 0];
           if E2 == 0
            posmov(3,1) = posmov(3,1) + 1;
            r2 = 2*posmov(3,1);
            posmov(3,r2)     = dd(ll);
            posmov(3,r2 + 1) = dd(ll+1);
         elseif E2 == 8
            posmov(5,1) = posmov(5,1) + 1;
            r2 = 2*posmov(5,1);
            posmov(5,r2)     = dd(ll);
            posmov(5,r2 + 1) = dd(ll+1);
         else
            error('err3');
         end
      elseif E1 == 8
         posmov(5,1) = posmov(5,1) - 1;
         r = searchIndex(posmov(5,2:end),dd(ll),dd(ll+1)) + 1;
         posmov(5,r:end) = [posmov(5,r+2:end) 0 0];
         posmov(4,1) = posmov(4,1) + 1;
         r2 = 2*posmov(4,1);
         posmov(4,r2)     = dd(ll);
         posmov(4,r2 + 1) = dd(ll+1);
      end

      A(i,j) = -sign(A(i,j));
   end %for

   %Possible moves after flip
   if step == 1; stepback = 5;
   elseif step == 2; stepback = 4;
   elseif step == 3; stepback = 3;
```

```
   elseif step == 4; stepback = 2;
   elseif step == 5; stepback = 1;
   end

   after = posmov(stepback,1);

end %update moves
%End update checker board

%New sample check
samples(l,3) = -samples(l-1,3);
if acc == 1
%Acceptance propability
factor = before/after;
acceptance = factor*exp(beta*J*deltaE(step));

if rand < acceptance
     samples(l,1) = proposal;
     accepted = accepted + 1;
     %Final flip
     A(i,j) = -sign(A(i,j));
     posmovM = posmov;
else
     samples(l,1) = samples(l-1,1);
end
%end step B

    l = l+1;
end %if

%STEP C ----------------------------------------
if rand < 1 - theta
    %Keep direction
    samples(l-1,3) = -samples(l-1,3);
end
%end step b

%STEP A ----------------------------------------
%Pick new stepsize
dir = samples(l-1,3);
if dir == 1
step = randi(3) + 2;
samples(l-1,2) = step;
else
step = randi(3);
samples(l-1,2) = step;
end
%end step A

%Plot ----------------------------------------
if plotRun

%Clear current figure
clf;

%Plot distribution
subplot(2,1,1);
title('SampleHistogramm');
xlabel('Energy Level');
ylabel('Probability');
```

```
    grid off;
    xlim([-2*N 2*N]);
    hold('on');
    y = samples(1:l,1);
    a = histc(y,x);
    zz = sum(a);
    bar(x,a/zz,'b','EdgeColor',[0 0 0.6])

    %Plot checkerboard
    subplot(2,1,2);
    title('Checkerboard');
    hold('on');
    colormap(copper);
    imagesc(A);
    axis image;

    %Pause
    tic; while toc < 0.00001; end
    drawnow;

%     %Wait for button press - presentation
%     if l == 100 || l == 1000
%         waitforbuttonpress;
%     end

    end %if plotRun

end %sampling

% ------------------------------------------------------------------------
%% Output after Calculation

%Read samples for plots
y = samples(:,1);

%Plot sample histogram and movement of samples
if ~plotRun

    %Plot samples
    subplot(2,1,1);
    title('Sample Histogramm');
    grid off;
    xlim([-2*N 2*N]);
    xlabel('Energy Level');
    ylabel('Probability');
    hold('on');
    a = histc(y,x);
    zz = sum(a);
    bar(x,a/zz,'b','EdgeColor',[0 0 0.6]);
    hold('on');
    fitt = fitdist(y,'kernel');
    plot(x,4*pdf(fitt,x),'g','LineWidth',3);

    %Movement of energy level
    subplot(2,1,2);
    title('Movement');
    ylabel('Energy Level');
    xlabel('Samples');
    hold('on');
    y = y(1:end);
```

```
    plot(1:length(y),y,'r','LineWidth',1);

end %plot

%Display indicators
if calcIndicators

   samples = y;
   L = 100;
   m = mean(samples);
   v = cov(samples);
   autocorrs = zeros(L,1);

   %Autocorrelation
   for i = 1:L
       autosum = 0;
       for j = 1:n-i
           autosum = autosum + (samples(j,:)-m)/(2*v)*(samples(j+i,:)-m)';
       end
       autocorrs(i) = autosum/(n-i-1);
    end

   INEFFICIENCY = 1+2*sum(autocorrs)
   ACCEPTANCERATE = accepted/n

end %indicators

function dE = readE(A,i,j)
%% Read current energy level of matrix A

    [NN,~] = size(A);

    %Cycle representation
    up    = mod(i-2,NN) + 1;
    down  = mod(i,NN)   + 1;
    left  = mod(j-2,NN) + 1;
    right = mod(j,NN)   + 1;

    %Determine possible move
    dE = -2*A(i,j)*(A(up,j) + A(down,j) + A(i,right) + A(i,left));

end %read energy

function r = searchIndex(x,i,j)
%% Search for position of wanted index

    le = length(x);
    for k = 1:2:le-1
        if x(k) == i
            if x(k+1) == j
                r = k;
                return;
            end
        end
    end
    r = 0;
end %search index

end %main
```