Electronic Supplementary Material (ESI) for Physical Chemistry Chemical Physics.

Supporting Information

Positional recurrence maps: a powerful tool to de-correlate static and dynamical disorder in distribution maps from molecular dynamics simulations

A. Piovano,[a*] A. Perrichon,[b] M. Bohem,[a] M Johnson,[a] W. Paulus[b]

[a] Institut Laue-Langevin, 71 avenue des Martyrs 38000 Grenoble, France.

[b] University of Montpellier 2, UMR 5253, ICGM, C2M, CC1504, 5 Place Eugène Bataillon, 34095 Montpellier, France

Table S1 – Most important input parameters used for the magnetic molecular dynamic simulations performed on the $Nd_2NiO_{4.0}$ system.

ALGO = V	ISYM = 0	ISTART = 0
MAXMIX = 40	ISIF = 2	ISMEAR = 0
IBRION = 0	SMASS = 1	SIGMA = 0.05
NSW = 21000	LREAL = .TRUE.	ISPIN = 2
NBLOCK = 1	LCHARG = .TRUE.	ICHARG = 2
POTIM = 1	NELMIN = 4	MAGMOM = 64*0.0 2.0 2.0 2.0 2.0 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 2.0 2.0 2.0 2.0 32*0.0
TEBEG = 300	PREC = ACCURATE	

Table S2 – Most important input parameters used for the molecular dynamic simulations performed on the Nd_2NiO_{4+d} system. The only variable is TEBEG, the simulation temperature.

ALGO = V	POTIM = 2	LCHARG = .FALSE.
MAXMIX = 40	TEBEG = 300	PREC = LOW
IBRION = 0	ISYM = 0	ISTART = 0
NSW = 21000	ISIF = 2	ISMEAR = 0
NELMIN = 4	SMASS = 2	SIGMA = 0.05
NBLOCK = 1	LREAL = .TRUE.	

Position Recurrence Map code. Here after the core of the PRM code is reported, including exhaustive explanations. The code is written in MATLAB language and results easily adaptable to any structure where dynamical behavior can be catch with a 2D projection. Here the data importation is defined for 3 types of atoms, O, Ni, Nd, but can be easily adapted for other compositions and structures. The following script needs as input only the XDATCAR file from VASP molecular dynamics.

```
DISCALIMER: this code is given free and 'as is' to any user it might find
it useful. However it is the results of our research activity, so PLEASE
CITE THIS MANUSCRIPT (A. Piovano, A. Perrichon, M. Bohem, M Johnson, W.
Paulus, Phys Chem Chem Phys (2015) DOI: 10.1039/c5cp06464c.) for any future
manuscript that will include results coming from the use or adaptation of
our code



% Extract atomic positions from XDATCAR (VASP output for molecular
dynamics) and calculate
% and plot positional recurrence maps (PRM), MSD and position histogram
    % cd_sym=0 without further corrections
    % cd_sym=1 added ortho/tetragonal symmetry (direction shall be defined
at line 208/212)
    % cd_plot(1) Positional recurrence maps (2D histogram)
    % cd_plot(2) Histogram of mean square displacements
    % cd_plot(3) Histogram of positions x/y/z
    % cd_space
        % 'real' for real cartesian space (positions), where PRM got static
& dynamical components
        % 'com' for center-of-mass space (displacements), where PRM got
dynamical component only
    % cd_space_grid
        % define a 2D high-symmetry grid (in respect to split positions)
        % not needed if cd_space='com'
        % format [x0,y0,ux,uy], as a node (x0,y0) with vectors (ux,uy)

    %%% V4.2 @ 2015/11/20, A. Perrichon
        % colorbar labels scaled to 1-X counts
        % histogram of positions implemented
    %%% V4.1 @ 2015/11/12, A. Perrichon
        % absolute scaling of the color log scale, figure's things
    %%% V4.0 @ 2015/11/11, A. Perrichon
        % parallelization of things (to optimize comp. time)
        % import data rewritten to handle any number of atom types
        % proper normalization of things
        % cleaning + rename of variables for clarity

%%%%%%%%%%%%%%%%
% Initialisation
%%%%%%%%%%%%%%%%
close all;
clear; clc;

%%
%%%%%%%%%%%%%%%%%%%%
% Command parameters
%%%%%%%%%%%%%%%%%%%%
cd_pathway='dat\XDATCAR_MD410T310'; % pathway to XDATCAR file
```

```
cd_at=num2cell(11:42); % selected equivalent atoms (XDATCAR lines)
cd_supercell=[2,2,1]; % supercell dimension for figure scaling / e.g.
[2,2,1] / e.g. [2*sqrt(2),2*sqrt(2),1]

cd_PRM_direction=3; % 1 for x-proj, 2 for y-proj, 3 for z_proj
cd_PRM_resolution=300; % resolution of the PRM / must be even / default:
300
cd_dist_max=2; % upper limit in AA for MSD/positions histogram
cd_sym=1; % 0/1 % see header

cd_space='real'; % real/com % see header
cd_space_grid=[1/8 1/8 1/4 1/4]; % format [x0,y0,ux,uy] % see header

cd_time_step=2; % time step of the molecular dynamics in fs
cd_time_start=0; % number of steps to remove at the beginning of simulation
%0/1000
cd_time_stop=20000; % [] % step number to stop at

cd_plot=[1,1,1]; % plotting % see header

%%
%%%%%%%%%%%%%%%
% Import data
%%%%%%%%%%%%%%%
disp('Data importation and conversion: started'); tic;
% read number of atoms + number of inequivalent atoms
fid=fopen(cd_pathway);
linenum=7;
line_tmp=textscan(fid,'%s',1,'delimiter','/n','headerlines',linenum-1);
n_to=sum(sscanf(line_tmp{1,1}{1,1},'%f'));
fclose(fid); clear fid linenum line_tmp;
% read cell parameters
fid=fopen(cd_pathway);
linenum=3;
line_tmp=textscan(fid,'%s',3,'delimiter','/n','headerlines',linenum-1);
lat=[sscanf(line_tmp{1,1}{1,1},'%f'),sscanf(line_tmp{1,1}{2,1},'%f'),sscanf
(line_tmp{1,1}{3,1},'%f')];
fclose(fid); clear fid linenum line_tmp;
% import trajectories
fid=fopen(cd_pathway);
S_tmp=textscan(fid,'%f %f %f','headerlines',7);
S=[S_tmp{1,1}(:,1),S_tmp{1,2}(:,1),S_tmp{1,3}(:,1)];
len=length(S);
fclose(fid); clear fid S_tmp ans;
disp('Data importation and conversion: done'); toc; disp(' ');

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extract data for selected atoms
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('Data extraction for selected atoms and time frame: started'); tic;
if isempty(cd_time_stop); cd_time_stop=len/n_to; else end
if cd_time_stop>(len/n_to); cd_time_stop=len/n_to; else end
S=S([1:n_to,n_to*(cd_time_start+1)+1:n_to*(cd_time_stop)],:);
len=length(S);
S=reshape(permute(S,[2 1]),[3,n_to,len/n_to]); S=permute(S,[3 1 2]);
pos=S(:,:,cell2mat(cd_at)-8);
disp('Data extraction for selected atoms and time frame: done'); toc;
disp(' ');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Correct data from periodic boundaries
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('Data correction from periodic boundaries: started'); tic;
% can't be parallelized (step by step domino effect)
for h=1:length(cd_at)
    for v=1:3
        for k=2:len/n_to
            % if displacement from a frame to another is |d|>0.8 then shift
back
            if pos(k,v,h)-pos(k-1,v,h)<-0.8==1
                pos(k,v,h)=pos(k,v,h)+1;
            elseif pos(k,v,h)-pos(k-1,v,h)>0.8==1
                pos(k,v,h)=pos(k,v,h)-1;
            else
            end
        end; clear k;
    end; clear v;
end; clear h;
disp('Data correction from periodic boundaries: done'); toc; disp(' ');

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Prepare data for PRM job
%%%%%%%%%%%%%%%%%%%%%%%%%%%
if strcmp(cd_space,'real')
    disp('Find site label: started'); tic;
    % define grid type corresponding to high-sym. structure: yet only for
2D x-y
    grid_x=[cd_space_grid(1):-
cd_space_grid(3):0,cd_space_grid(1)+cd_space_grid(3):cd_space_grid(3):1];
    grid_y=[cd_space_grid(2):-
cd_space_grid(4):0,cd_space_grid(2)+cd_space_grid(4):cd_space_grid(4):1];
    % find for all selected atoms the closest node on the defined grid
    rea=zeros(1,3,length(cd_at));
    for h=1:length(cd_at)
        clear dl; dl=zeros(length(grid_x),length(grid_y));
        for x=1:length(grid_x)
            for y=1:length(grid_y)
                dt=pos(1,:,h)-[grid_x(x),grid_y(y),pos(1,3,h)];
                dl(x,y)=sqrt(power(dt(1),2)+power(dt(2),2));
            end; clear y;
        end; clear x;
        [~,tm]=min(dl(:)); [dx,dy]=ind2sub(size(dl),tm);
        rea(1,:,h)=[grid_x(dx),grid_y(dy),pos(1,3,h)]-pos(1,:,h);
    end; clear h grid_x grid_y grid_xy tm dx dy dl dt;
    disp('Find site label: done'); toc; disp(' ');
else
    if strcmp(cd_space,'real')==0
        rea=zeros(1,3,length(cd_at));
        for h=1:length(cd_at)
            rea(1,:,h)=[0,0,0];
        end; clear h;
    else
    end
end

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Processing job (positional recurrence maps & things)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
disp('Histogram job for selected atoms: started'); tic;
% shift all selected equivalent atoms to (0.5,0.5,0.5)
shi=zeros(1,3,length(cd_at)); shi(1,:,:)=0.5-pos(1,:,:);
map=pos+repmat(shi,[len/n_to 1 1])+repmat(rea,[len/n_to 1 1]);
clear shi rea;

% set box size for PRM, MSD and positions histogram (PRM resolution)
bs_xsize_min=0; bs_xsize_max=1; bs_ysize_min=0; bs_ysize_max=1;
bs_zsize_min=0; bs_zsize_max=1;
bs_xvec=(bs_xsize_min:(bs_xsize_max-
bs_xsize_min)/cd_PRM_resolution:bs_xsize_max); bs_xvec=bs_xvec(:);
bs_yvec=(bs_ysize_min:(bs_ysize_max-
bs_ysize_min)/cd_PRM_resolution:bs_ysize_max); bs_yvec=bs_yvec(:);
bs_zvec=(bs_zsize_min:(bs_zsize_max-
bs_zsize_min)/cd_PRM_resolution:bs_zsize_max); bs_zvec=bs_zvec(:);
bs_void=zeros(cd_PRM_resolution+1,cd_PRM_resolution+1,cd_PRM_resolution+1);

od_size_min=0;
od_step=sqrt(power((bs_xsize_max-
bs_xsize_min)/cd_PRM_resolution*lat(1,1),2)+...
    power((bs_ysize_max-bs_ysize_min)/cd_PRM_resolution*lat(2,2),2)+...
    power((bs_zsize_max-bs_zsize_min)/cd_PRM_resolution*lat(3,3),2));
od_vec=(od_size_min:od_step:cd_dist_max);
od_void=zeros(1,length(od_vec)); od_msd=zeros(length(cd_at),len/n_to-1);

id_size_min=0; id_step=0.05;
id_vec_x=(id_size_min:id_step:cd_dist_max);
id_vec_y=(id_size_min:id_step:cd_dist_max);
id_vec_z=(id_size_min:id_step:cd_dist_max);
id_x=zeros(1,length(id_vec_x)); id_y=zeros(1,length(id_vec_y));
id_z=zeros(1,length(id_vec_z));

% for each atom, for each frame of MD, locate in which box is the atom
% can't be fully parallelized (cost too much memory)
if cd_plot(1)==1
    for h=1:length(cd_at)
        for k=2:len/n_to
            % PRM
            [~,bs_cd_x]=min(abs(map(k,1,h)-bs_xvec));
            [~,bs_cd_y]=min(abs(map(k,2,h)-bs_yvec));
            [~,bs_cd_z]=min(abs(map(k,3,h)-bs_zvec));

bs_void(bs_cd_x,bs_cd_y,bs_cd_z)=bs_void(bs_cd_x,bs_cd_y,bs_cd_z)+1;
        end; clear k;
    end; clear h bs_cd_x bs_cd_y bs_cd_z;
else
end;
if cd_plot(2)==1
    for h=1:length(cd_at)
        for k=2:len/n_to
            % MSD
            od_msd(h,k)=sqrt(power(((map(k,1,h)-0.5)*lat(1,1)),2)+...
                power(((map(k,2,h)-0.5)*lat(2,2)),2)+...
                power(((map(k,3,h)-0.5)*lat(3,3)),2));
            [~,od_cd]=min(abs(od_msd(h,k)-od_vec));
            od_void(od_cd)=od_void(od_cd)+1;
        end; clear k;
    end; clear h od_cd od_msd;
else
end;
if cd_plot(3)==1
```

```
    for h=1:length(cd_at)
        for k=2:len/n_to
            % Positions
            [~,id_cd_x]=min(abs(((map(k,1,h)-0.5)*lat(1,1))-id_vec_x));
            [~,id_cd_y]=min(abs(((map(k,2,h)-0.5)*lat(2,2))-id_vec_y));
            [~,id_cd_z]=min(abs(((map(k,3,h)-0.5)*lat(3,3))-id_vec_z));
            id_x(id_cd_x)=id_x(id_cd_x)+1;
            id_y(id_cd_y)=id_y(id_cd_y)+1;
            id_z(id_cd_z)=id_z(id_cd_z)+1;
        end; clear k;
    end; clear h id_cd_x id_cd_y id_cd_z;
else
end;

% normalisation, projection
norm=((len/n_to)-1)*length(cd_at); % total count = 1
bs_void=bs_void./norm; bs_plan=sum(bs_void,cd_PRM_direction);
bs_plan=reshape(bs_plan,[cd_PRM_resolution+1,cd_PRM_resolution+1]);
bs_plan=permute(bs_plan,[2 1]);
cd_supercell(cd_PRM_direction)=[]; cd_supercell=fliplr(cd_supercell);
od_void=od_void./norm; id_x=id_x./norm; id_y=id_y./norm; id_z=id_z./norm;
disp('Histogram job for selected atoms: done'); toc; disp(' ');

% symmetry, log scale
if cd_sym==1
    disp('Symmetry expansion: started'); tic;
    % set fwd and bcwd axis, set transpose matrix (i.e. 4 operation)
    sym_fwd=1:1:length(bs_void); sym_bcwd=length(bs_void):-1:1;
bs_plan_rev=bs_plan';
    if cd_PRM_direction==3

bs_sym=(bs_plan(sym_fwd,sym_fwd)+bs_plan(sym_fwd,sym_bcwd)+bs_plan(sym_bcwd
,sym_fwd)+...

bs_plan(sym_bcwd,sym_bcwd)+bs_plan_rev(sym_fwd,sym_fwd)+bs_plan_rev(sym_fwd
,sym_bcwd)+...

bs_plan_rev(sym_bcwd,sym_fwd)+bs_plan_rev(sym_bcwd,sym_bcwd))./8;
    else

bs_sym=(bs_plan(sym_fwd,sym_fwd)+bs_plan(sym_fwd,sym_bcwd)+bs_plan(sym_bcwd
,sym_fwd)+...
            bs_plan(sym_bcwd,sym_bcwd))./4;
    end
    bs_log=log(bs_sym);
    disp('Symmetry expansion: done'); toc; disp(' ');
else
    bs_log=log(bs_plan);
end; clear sym_fwd sym_bcwd bs_plan_rev;

%%
%%%%%%%%%%%%%
% Plot things
%%%%%%%%%%%%%
fig_flag=1;

if cd_plot(1)==1 % positional recurrence maps (2D histogram)
    f_e=figure(fig_flag);
    surf(bs_xvec,bs_yvec,bs_log,'lines','none');
    set(gcf,'color','w'); axis image; box on; grid off;
    xmin=0.5*(1-1/cd_supercell(2)); xmax=0.5*(1+1/cd_supercell(2));
```

```
    ymin=0.5*(1-1/cd_supercell(1)); ymax=0.5*(1+1/cd_supercell(1));
    xlim([xmin xmax]); ylim([ymin ymax]);
    set(gca,'Xtick',[xmin ((xmax-xmin)*0.25+xmin)...
        0.5 ((xmax-xmin)*0.75+xmin) xmax]);
    set(gca,'Ytick',[ymin ((ymax-ymin)*0.25+ymin)...
        0.5 ((ymax-ymin)*0.75+ymin) ymax]);
    set(gca,'XTickLabel',{sprintf('%1.0f',0.5-(abs(0.5-
xmin)*cd_supercell(2))),...
        sprintf('%0.2f',0.5-(abs(0.5-((xmax-
xmin)*0.25+xmin))*cd_supercell(2))),...
        '0.5',sprintf('%0.2f',0.5+(abs(0.5-((xmax-
xmin)*0.75+xmin))*cd_supercell(2))),...
        sprintf('%1.0f',0.5+(abs(0.5-xmax)*cd_supercell(2)))});
    set(gca,'YTickLabel',{sprintf('%1.0f',0.5-(abs(0.5-
ymin)*cd_supercell(1))),...
        sprintf('%0.2f',0.5-(abs(0.5-((ymax-
ymin)*0.25+ymin))*cd_supercell(1))),...
        '0.5',sprintf('%0.2f',0.5+(abs(0.5-((ymax-
ymin)*0.75+ymin))*cd_supercell(1))),...
        sprintf('%1.0f',0.5+(abs(0.5-ymax)*cd_supercell(1)))});
    title(sprintf('Positional Recurrence Maps in %s space', cd_space));
    if cd_PRM_direction==1; xdir='y'; ydir='z'; else end;
    if cd_PRM_direction==2; xdir='x'; ydir='z'; else end;
    if cd_PRM_direction==3; xdir='x'; ydir='y'; else end;
    xlabel(sprintf('%s with respect to 1x1x1 supercell (r.l.u.)',xdir));
    ylabel(sprintf('%s with respect to 1x1x1 supercell (r.l.u.)',ydir));
    caxis([-13.5 -5.8]); % arbitrary, about X=1 count to X=2000 counts by
log(X/norm)

cb_scale=[log(0.1/norm),log(1/norm),log(10/norm),log(100/norm),log(1000/nor
m),log(10000/norm)];
    c=colorbar('YTick',cb_scale);

set(c,'YTickLabel',{sprintf('%1.0f',0.1),sprintf('%1.0f',1),sprintf('%1.0f'
,10),...

sprintf('%1.0f',100),sprintf('%1.0f',1000),sprintf('%1.0f',10000)});
    ylabel(c,sprintf('Count per d^3r = %1.2e angstrom^3',(bs_xsize_max-
bs_xsize_min)/...
        cd_PRM_resolution*lat(1,1)*(bs_ysize_max-
bs_ysize_min)/cd_PRM_resolution*lat(2,2)...
        *(bs_zsize_max-bs_zsize_min)/cd_PRM_resolution*lat(3,3)));
    set(f_e,'Position',[80,15,480,480]);
    view(0,90);
    fig_flag=fig_flag+1;
else
end; clear xmin xmax ymin ymax cb_scale xdir ydir cb_pos c f_e;

if cd_plot(2)==1 % histogram of mean square displacements
    f_g=figure(fig_flag);
    set(f_g,'Position',[80,15,480,480]);
    plot(od_vec,od_void,'-ob');
    title(sprintf('Histogram of MSD in %s space', cd_space));
    xlabel('Distance (angstrom)');
    ylabel(sprintf('Count per dr=%0.3f angstrom',od_step));
    fig_flag=fig_flag+1;
    set(gcf,'color','w');
else
end; clear f_g;

if cd_plot(3)==1 % histogram of x,y,z positions
```

```
    f_f=figure(fig_flag);
    set(f_f,'Position',[80,15,480,480]);
    hold on
        plot(id_vec_x,id_x(:),'-ob');
        plot(id_vec_y,id_y(:),'-og');
        plot(id_vec_z,id_z(:),'-or');
    hold off
    title(sprintf('Histogram of positions in %s space', cd_space));
    xlabel('Distance (angstrom)');
    ylabel(sprintf('Count per dr=%0.3f angstrom',id_step));
    legend('x','y','z');
    fig_flag=fig_flag+1;
    set(gcf,'color','w');
    ylim([0 0.2]);
else
end; clear f_f;
```