Read Me

In this data repository, the data for the work "*Coherent and incoherent structures in fuzzy dark matter halos*", published on MNRAS in 2023 (see arxiv:2211.02565 for preprint), are provided. To support the generality of our discussion, we do not only just provide the data to reproduce our figures but also append the data to generate figures, like Fig. 3 (a) and (b), Fig. 4 and Fig. 7, for all of our merger simulations, which are 10 independent halos for each total mass, namely, the 30 points in this work are all accessible. In addition, the plotting commands are briefly addressed/appended for essential reproductions.

Below, the **mat files** are marked in bold, and the relevant figures are marked with underlines. MATLAB *commands* are mentioned in italic format. The "ini" variable labels the simulation with a different initial configuration. Please note that such configurations are independent of different M (total mass).

The listed mat files are given in the data repository with details addressed in this document for this work:

mat file name	Brief description	page
spe-M=100-ini=9-t=*.mat	t=0, 1.5 and 8 For t=8, there are further post processed profiles. See the following description for more detail.	2
PO_M=100-ini=9.mat	The first 5 largest PO modes in fully 3D array form	2
correlation_func_avg-t=4.5-9.3625.mat and **correlation_func_avg-t=8-8.5.mat**	The spatial correlation function for the profiles averaged different time span for M=100, ini=9 data set.	3
M=*energy_evolution.mat	the total energy and its decomposition into total kinetic, quantum pressure, compressible, incompressible and gravitational energies	3
M=*_collection.mat	The spatially radial profiles discussed in the content are provided in detail.	3
PowerSpectrum_M=*.mat	The momentum spectra discussed in the content are provided in detail.	5
M=100-ini=9-zeta.mat	The zeta values as function of time in Fig. 6 (b) are provided.	7

▪ spe-M=100-ini=9-t=*.mat series

The wavefunctions for Fig. 2 (a)-(i) and (ii) are given in "**spe-M=100-ini=9-t=0.mat**" and "**spe-M=100-ini=9-t=1.5.mat**" with the variables listed below

- x,y and z: the spatial grids in 1D array format.
- ut: the wavefunction in the fully 3D array form.

Additional variables are provided in **spe-M=100-ini=9-t=8.mat**,

- ix_recen, iy_recen, iz_recen: the position index to recenter the spatial profile for the origin being the peak density location.
- Vsq_incom: the absolute square value of the incompressible velocity field in the fully 3D array form.

By using volume rendering along 3 different directions (x, y and z axes) and plotting the iso density the plots in Fig. 2 (a) can be regenerated.

In "**spe-M=100-ini=9-t=8.mat**", the full wavefunction, ut, and the absolute square value of the incompressible velocity field, Vsq_incom, are given together with the spatial coordinate, x, y and z, and the re-centred grid index which uses the peak density at the centre of the coordinate. By using the *slice*" and "*isosurface* in MATLAB, one can straightforwardly obtain Fig. 2 (a)-(iii), Fig. 6 (a) and Fig. 7 (a).

The volume rendering image, Fig. 1, contains both the information from "**spe-M=100-ini=9-t=8.mat**" and these two different volume rendered images are blended by the Photoshop-like "screen" effect via the mathematic post-process, $1-(1-A)\times(1-B)$, where A and B are the RGB values of volume rendered density and the absolute square of the incompressible velocity field. More detail can be referred to https://photoblogstop.com/photoshop/photoshop-blend-modes-explained. The colour schemes for the density and the absolute square of the incompressible velocity field can be approached by "*parula*" and "*hot*" respectively (Very marginal fine-tuning of the colorbars are used in Fig. 1, so this fine detail is not given to simplify the document here.).

▪ PO_M=100-ini=9.mat

The first 3 Penrose-Onsager condensate modes are provided. Similar to the structure in **spe-M=100-ini=9-t*.mat** series, this file contains

- ew: the first 5 PO condensate number (divided by rho0=0.1)
- den_cen_avg: the time-averaged density in a fully 3D array form.
- uPO: the largest PO condensate mode in a fully 3D array form.
- uPO2 and uPO3: the 2nd and 3rd PO condensate modes in a fully 3D array form.
- x,y and z: the spatial grids in 1D array format.

The slice profile in Fig. 3 (c) can be generated by

```
figure,
imagesc(x,y,log(abs(A(:,:,length(z)/2+1).^2))/log(10));
axis equal
caxis([-2 4.2816])
```

for A being uPO, uPO2 and uPO3.

▪ correlation_func_avg-t=* series

correlation_func_avg-t=4.5-9.3625.mat and **correlation_func_avg-t=8-8.5.mat** contain

- x,y and z: the spatial grids in 1D array format.
- g1: the time averaged first-order correlation function in fully 3D array form.
- g2: the time averaged second-order correlation function in fully 3D array form.
- t_avg: the times considered in the time average.

Fig. 3 (d) illustrates their sliced profiles across the origin with the following commands:

```
figure,
subplot(1,2,1)
imagesc(x,y,real(g1(:,:,length(z)/2+1)));
axis square

subplot(1,2,2)
imagesc(x,y,g2(:,:,length(z)/2+1));
axis square
```

▪ M=*energy_evolution.mat

In "**M=*energy_evolution.mat**", the total energy and its decomposition into total kinetic, quantum pressure, compressible, incompressible and gravitational energies are described. Each of the variables is packed in cell form. Using *A{ini}* to read out the interest variable/evolution for A=A(t) being:

- t_start: The only 1D array in these files provides the tentative "virialized" time of the system.
- Eg: Gravitational energy.
- Eke: Total kinetic energy containing quantum pressures and classical kinetic energies.
- Eke_com: Compressible part of the classical kinetic energy.
- Eke_incom: Incompressible part of the classical kinetic energy.
- Eke_qf: Quantum pressure energy.
- Etot: Total energy, which is roughly a constant, is the summation of Eke and Eg.
- t_all: the time spans of the simulation. Each cell contains an array of time.

The plotting command is straightforward by using *plot(t,A)* for the interested quantity A.

▪ M=*_collection.mat series

In the series of "**M=*_collection.mat**" with the details listed below, the data set for regenerating Fig. 2 (c) and (d), Fig. 3 (for the configuration ini=9 for M=100) and Fig. 5 (a). The variable listed below are included in this file set:

Fixed parameters/variables:

- rho0: the spatially averaged density value while the density profiles below are all scaled.
- r: the radial coordinate

Time-averaged results (10 realizations of each variable, each row corresponds to a different simulation from a different initial configuration.)

- den_avg_r: radial profile of density, the origin of the axes is re-centred to the position of peak density.

- dden_avg_r: the corresponding standard deviation.
- fch: the cored-halo fitting profile extracted from den_avg_r.
- rc: the core-radius of the cored-halo fit. There are three columns. The first column is the most confidential value of the fit and the second and third are the 95% confidential lower and upper bounds of the fit. Each row of the array corresponds to the results of an ini-th simulation.
- rt: An 1D array records the transition radius of the cored-halo fit for ini-th simulation.
- denPO_r: radial profile of the Penrose-Onsager mode.
- ddenPO_r: the standard deviation of the Penrose-Onsager mode.
- conc_r: concentration parameter/phase-space density, equivalent to *rho0*den_avg_r/Vabs_sq_r*.
- dconc_r: the standard deviation of the phase-space density. The fluctuation in Vabs_vsq is not considered.
- G1_r: radial profile of the time-averaged first-order correlation function
- dG1_r: the standard deviation of the azimuthal average of the first-order correlation function
- G2_r: radial profile of the time-averaged second-order correlation function
- dG2_r: the standard deviation of the azimuthal average of the second-order correlation function
- Vabs_avg_r: radial profile of the absolute square of the incompressible velocity field.
- dVabs_avg_r: the standard deviation of the azimuthal average of the absolute square of the incompressible velocity field.
- ke_r: Radial profile of the total kinetic energy, containing quantum pressure, compressible and incompressible energy.
- dke_r: The standard deviation of the total kinetic energy density, as a function of r.
- ke_com_r: Radial profile of the compressible kinetic energy density.
- dke_com_r: The standard deviation of the compressible kinetic energy density, as a function of r.
- ke_incom_r: Radial profile of the incompressible kinetic energy density.
- dke_incom_r: The standard deviation of the incompressible kinetic energy density, as a function of r.
- ke_qf_r: Radial profile of the quantum pressure energy density.
- dke_qf_r: The standard deviation of the quantum pressure energy density, as a function of r.

By using

plot(r,A) or *errorbar(r,A,dA)*

for specific variables, one can obtain the radial profile of A (with errorbar if dA is available).

These variables are cell arrays and each cell

- t_all: the time spans of the simulation. Each cell contains an array of time.
- den_rt_all: each cell contains the radial density profiles ρ/ρ_0 at a given time in the way *den_rt_all{ini}*=den(t,r) (2D array)for ini=1-10 labelling the different initial configurations.
- Fin_Func: The core-halo fitting function extracted for the dynamical radial density profile. A cell in Fit_Func has two other cells, describing the fitting functions for the core and halo parts. One can use it together with rc_dync, rt_dync and rh_dync (listed below) to replot the core-halo fitting r_core = r(r<rt)
- rc_dync: the core radius of the cored-halo fit for dynamical radial density profile at time t. The same as the rc, there are three columns for *rc_dync{ini}* correspond to the most

confidential and the 95% lower and upper bounds. The vertical index denotes the time index with one-to-one relation to *t_all{ini}*.

- rt_dync: the transition radius of the core-halo fit for the dynamical radial density profile. Each *rt_dync{ini}* is an 1D array for the transition radius as a function of time.
- rh_dync: The halo length scale of the core-halo fit for the dynamical radial density profile. The three-column structure is the same as the rc_dync.

Combining the above variables, one can replot the dynamical density profile and the corresponding core-halo fit as a function of r, e.g. Fig. 5 (a), via

```
ini = 9

t_target = [6 7]

den_rt = den_rt_all{ini};
rc_t = rc_dync{ini};
rh_t = rh_dync{ini};
rt_t = rt_dync{ini};
core_halo_func = Fit_Func{ini};

figure,
for tt = 1 : length(t_target)
    it_target = find(t_all{ini}==t_target(tt));
    r_core = rc_t(it_target,1);
    r_tran = rt_t(it_target);
    r_halo = rh_t(it_target,1);
    eval(['fc = ' core_halo_func{it_target,1}]);
    eval(['fh = ' core_halo_func{it_target,2}]);
    core_fit = fc(r,r_core);
    halo_fit = fh(r,r_halo);
    core_fit(r>r_tran) = 0;
    halo_fit(r<=r_tran) = 0;
    ch_fit = core_fit + halo_fit;
    subplot(1,length(t_target),tt);
    plot(r,den_rt(it_target,:),'-k','linewidth',2);
    hold on
    plot(r,ch_fit,'--m','linewidth',1);
    set(gca,'Xscale','log','Yscale','log');
    title(['t=' num2str(t_all{ini}(it_target))])
    xlabel('r/l_{ref}')
    ylabel('\rho/\rho_0')
end
```

▪ PowerSpectrum_M=*.mat series

In "**PowerSpectrum_M=*.mat**", the spectra in the radial momentum are provided

- kr: The array of radial momentum.
- t_start: an array specifies the tentative ‘virialized’ time for a given simulation, said, *t_start(ini)* for ini-th initial configuration.
- t_all: the time spans of the simulation. Each cell contains an array of time.

- Delta_krt: The dynamical profile of the power spectrum of the full density field as a function of radial momentum, kr. The data here is in *cell* format for different simulations. Using *Delta_krt{ini}* gives the power spectrum as a function of time, t, and kr in the form of Delta(t,kr) (2D array).
- Delta_sc_krt : The dynamical profile of the power spectrum of the over-density field, Eq. (40), as a function of radial momentum, kr. The data here is in *cell* format for different simulations. Using *Delta_sc_krt{ini}* will give the power spectrum as a function of time, t, and kr in the form of Delta_sc(t,kr) (2D array).
- varepsilon_incom_krt: The dynamical profile of vortex-/incompressible energy spectrum at a different time for all 10 simulations for specific M. The data here is in *cell* format for different simulations. Using *varepsilon_incom_krt {ini}* gives the power spectrum as a function of time, t, and kr in the form of varepsilon_incom_krt (t,kr) (2D array).
- k_peak_Delta_sc: The time-averaged peak moment of the over-density power spectrum, Delta_sc.
- dk_peak_Delta_sc: The standard deviation of the peak location of the full power spectrum through time, after *t_start(ini)* for a given initial configuration.
- k_peak_incom_avg: The time-averaged peak moment of the vortex-/incompressible-energy spectrum, varepsilon_incom_krt
- dk_peak_incom_avg: The standard deviation of the peak location of the vortex-/incompressible-energy spectrum through time, after *t_start(ini*) for a given initial configuration.

The combination of "**M=100_collection.mat**" and "**PowerSpectrum_M=100.mat**" for ini=9 can generate Fig. 5. Similar to the plotting for radial density profiles,

plot(kr,A) or *errorbar(kr,A,dA)*

for specific variables, one can obtain the momentum spectrum A (with errorbar if dA is available).

To generate figures like Fig. 7 (b) and (c) one can use

```
ini = 4, % number of configuration: 1 - 10

dV = (10/288)^3; % uni volume in the simulation to cover the Fourier factor

t = t_all{ini};
it_avg = find(t==t_start(ini));

f = mean(varepsilon_incom_krt{ini}(it_avg:end,:),1)*dV^2;
df = std(varepsilon_incom_krt{ini}(it_avg:end,:),0,1)*dV^2;
figure(72)
errorbar(kr,f,df)
set(gca,'Xscale','log','Yscale','log')
xlabel('$k_r (l_{ref}^{-1})$','interpreter','latex')
ylabel('$\tilde{\epsilon}_\mathrm{ke}^i$','interpreter','latex')

f = mean(Delta_sc_krt{ini}(it_avg:end,:),1)*dV^2;
figure(73)
plot(kr,f);
set(gca,'Xscale','log','Yscale','lin')
ylabel('$\Delta^2_{\delta}(k_r)$','interpreter','latex')
```

```
xlabel('$k_r (l_{ref}^{-1})$','interpreter','latex')
```

For the configuration number ("ini" in the plotting scripts) are ini=4 for M=50, ini=9 for M=100 and ini=3 for M=150 respectively in the main content. For Fig. 7 (d), one can refer to

```
figure(75);
hold on
errorbar(k_peak_Delta_sc,k_peak_incom_avg,dk_peak_incom_avg,dk_peak_incom
_avg,dk_peak_Delta_sc,dk_peak_Delta_sc,'o','linewidth',2);
```

or

```
for jj = 1 : size(Delta_krt,2)
    t = t_all{jj};
    it_avg = find(t==t_start(jj));
    Delta = Delta_krt{jj};
    Delta_sc = Delta_sc_krt{jj};
    KE_incom = varepsilon_incom_krt{jj};
    kmax_Delta_t = [];
    kmax_Delta_sc_t= [];
    kmax_KE_incom_t = [];
    for tt = 1 : length(t)
        [v_Delta_max,iDelta_max] = max(Delta(tt,:));
        [v_Delta_sc_max,iDelta_sc_max] = max(Delta_sc(tt,:));
        [v_KE_incom_max,iKE_incom_max] = max(KE_incom(tt,:));
        kmax_Delta_t(tt) = kr(iDelta_max);
        kmax_Delta_sc_t(tt) = kr(iDelta_sc_max);
        kmax_KE_incom_t(tt) = kr(iKE_incom_max);
    end
    kmax_Delta_avg(jj) = mean(kmax_Delta_t(it_avg:end));
    dkmax_Delta_avg(jj) = std(kmax_Delta_t(it_avg:end));
    kmax_Delta_sc_avg(jj) = mean(kmax_Delta_sc_t(it_avg:end));
    dkmax_Delta_sc_avg(jj) = std(kmax_Delta_sc_t(it_avg:end));
    kmax_KE_incom_avg(jj) = mean(kmax_KE_incom_t(it_avg:end));
    dkmax_KE_incom_avg(jj) = std(kmax_KE_incom_t(it_avg:end));
end

figure(74)
errorbar(kmax_Delta_sc_avg,kmax_KE_incom_avg,dkmax_KE_incom_avg,dkmax
_KE_incom_avg,dkmax_Delta_sc_avg,dkmax_Delta_sc_avg,'o','linewidth',2);
    hold on
```

and repeat the commands for different values of M.

- **M=100-ini=9-zeta.mat**

The zeta value, Eq. (36), is given in "**M=100-ini=9-zeta.mat**" containing:

- t: the time sequence in 1D array form for t>=4\tau_ref.
- Vsq_incom_rCt: the 2D array for the time evolution for zeta value at different x for each column.
- rc_ratio: the value of x/rc for rc being the time-averaged one

Fig. 6 (b) can be essentially reproduced by the following MATLAB commands,

```
tt = find(t==4.5);
rc_index = [1 4 10 33]
```

```
Colors = (colormap(parula(length(rc_index )+0)));
Legend = {};
jj = 1;
rc = 0.11;
for kk = fliplr(rc_index)
        kk
        if kk == 2
                LW = 2
        elseif kk == 1
                LW = 3
        else
                LW = 2;
        end
        Legend{jj} = ['x=' num2str(rc_ratio(kk)) 'r_c'];
        figure(32);
        plot(t(1:size(Vsq_incom_rCt,1)),Vsq_incom_rCt(:,kk),'linewidth',2,'color',C
        olors(jj,:));
        hold on
        jj = jj + 1;
end
legend(Legend)
```