EDEN: Exploring Disks Embedded in N-body simulations

Science/Documentation Paper: https://arxiv.org/abs/2408.01487

# Contents



# 1 Overview

Exploring Disks Embedded in N-body (EDEN) simulations is a set of dark-matter-only cosmological zoom-in re-simulations with embedded disk potentials of Milky Way (MW)-mass hosts ($M_{\rm vir} \sim 10^{12} M_\odot$) from the Symphony Simulation compilation (https://web.stanford.edu/group/gfc/gfcsims/build/html/symphony_overview.html). The goal of this project is to capture the range of additional tidal stripping effects on the subhalo population of MW-mass hosts given the range of baryonic disk masses and assembly histories allowed by observational constraints. To achieve such a goal, we update the successful and widely tested cosmological simulation code Gadget2 https://wwwmpa.mpa-garching.mpg.de/galform/gadget with the ability to apply additional gravitational acceleration from a time-evolving analytic disk potential that mimics the baryonic galaxy disk. The benefits of such an approach are three-fold:

- Averting the smooth-particle-hydrodynamics solver of the code makes computation much faster and yields more generic results without being subject to a specific choice of subgrid physics.
- Evaluating accelerations from an analytic gravitational potential is fast and only adds $\mathcal{O}(n)$ time complexity to the N-body gravity tree solver.
- The growth history of the disk can be easily tweaked without changing any subgrid physics, flexible for theoretical experiments that varies the galaxy–halo connection.

In EDEN, we assume that the MW-mass host's disk grows its mass following the predictions of the semi-empirical galaxy–halo connection model UniverseMachine (UM, https://arxiv.org/abs/1806.07893), which takes into account the correlation between galaxy star formation and halo assembly histories conditioned on halo mass (secondary halo bias). For more background and simulation results, please refer to the scientific publication of EDEN (https://arxiv.org/abs/2408.01487). EDEN simulation data can be accessed as part of the Symphony Simulation data release (https://web.stanford.edu/group/gfc/gfcsims/build/html/zoom_ins.html). In the following, we introduce the detailed changes that we have made to Gadget2 to create EDEN and provide instructions for how to set up EDEN for re-simulating any generic cosmological simulation.

## 2 New stuff in EDEN compared to Gadget2

The EDEN code repository can be accessed at https://github.com/Bulk826R/EDEN. The `main` branch contains the default version of EDEN where we use UM-based stellar mass growth histories. This branch is used for all main-text results, including the nine high-mass subsamples, in the EDEN scientific paper.

The curious user might also want to check out the `Gas_Guo23` branch, where we increase the total mass of the baryonic disk by the universal gas fractions given by the semi-empirical gas–halo connection model NeutralUniverseMachine (NeuUM, https://arxiv.org/abs/2307.07078). This branch is used for results in Appendix F of the EDEN scientific paper for investigating the impact of the gas disk, which showed a less prominent impact than the mass of the stellar disk.

In the following, we summarize all detailed code changes made, with specific changes relevant to either branches (the main difference is their disk mass implementation). Please note that all major code blocks added in EDEN on top of the Gadget2 source code have the delimiter '`//RW `' to highlight changes added by Richie Wang.

## 2.1 New variables introduced

We introduce six new global variables at the end of ALLVARS.H and ALLVARC.C. These six parameters include:

- `id0`: The particle ID of the sink particle which acts as the disk potential center. The sink particle is chosen to be the low-resolution (type 4) particle that is closest to the Rockstar-defined halo center of the host halo at $z = 3$ when we initialize the disk.
- `cenx`: The $x$-coordinate of the sink particle, in units of comoving Mpc/$h$.
- `ceny`: The $x$-coordinate of the sink particle, in units of comoving Mpc/$h$.
- `cenz`: The $x$-coordinate of the sink particle, in units of comoving Mpc/$h$.
- `Mdisk`: The total baryonic mass of the disk potential, in units of $10^{10} M_{\odot}/h$.
- `Fdisk`: The stellar mass fraction of the total baryonic disk mass, specified by the NeuUM model. Defaulted to 1 in the `main` branch with all disk mass coming from the UM-predicted stellar mass; loads redshift-dependent NeuUM values from a table.

## 2.2 Initialization of the disk potential

The disk potential is initialized from the $z = 3$ snapshot of the dark-matter-only (DMO) SymphonyMilkyWay simulation for each host halo. The disk potential is always centered on the 'sink' particle, which is assigned a fixed mass of $1.5 \times 10^8 M_{\odot}$ since $z = 3$ and is moved to the Rockstar-defined host halo center at that snapshot. Since the sink particle is way more massive than the high-resolution particles near the host halo center ($2.8 \times 10^5 M_{\odot}/h$), it can be kept near the host halo center by dynamical friction and acts as a stable anchor for the center of the disk potential. We add the files `disk_mass.c` and `disk_mass.h` for loading UM and NeuUM model values of stellar masses and stellar-to-gas fractions:

- `double Ms_UM(double z)`: Function that loads in the pre-saved UM predicted stellar mass history at the corresponding redshift (input 'z') of the snapshots for each host.

- `double Fs_Guo(double z)`: Function that loads in the pre-saved NeuUM predicted stellar mass fraction at the corresponding redshift (input 'z') of the snapshots for each host. This function is only called in the 'Guo_Gas23' branch.

The main code changes for applying these loaded values to the disk potential initialization are added in `init.c` and :

- Starting from line 62 to 103, the position, velocity, and particle ID of the sink particle are set. The position of the disk potential, Mdisk (calling `Ms_UM`), and Fdisk (calling `Fs_Guo`) are also set. The parameters `HALO_CEN_X`, `HALO_CEN_Y`, `HALO_CEN_Z`, `HALO_CEN_VX`, `HALO_CEN_VY`, `HALO_CEN_VZ`, `SINK_ID` are global variables set in the header file `halo_props.h`.

The header file `halo_props.h` is an important file that stores all the necessary information to initialize the disk potential and sink particle, and it is specific for every halo. One needs to retrieve all the relevant information from the DMO run and replace it every time that EDEN is run for a different host and re-compile the Gadget2 binary. The `halo_props.h` file for each EDEN host is stored in the path `./props/`. More specifically, for each halo's `halo_props.h`:

- `START_A`: The scale factor of the $z = 3$ snapshot of the DMO SymphonyMilkyWay run of each host, which we use as the initial condition of EDEN.

- `FORCE_A`: To avoid sudden kick in of the disk potential and shock the central part of the host halo, we gradually (linear in scale factor) turn on the full impact of the disk potential from 0% at `START_A` to 100% at $FORCE\_A$.

- `SINK_ID`: Particle ID of the sink particle.

- `HALO_CEN_X`: $x$-position of the Rockstar-defined host halo center in the DMO run, in units of comoving Mpc/$h$.

- `HALO_CEN_Y`: $y$-position of the Rockstar-defined host halo center in the DMO run, in units of comoving Mpc/$h$.

- HALO_CEN_Z: $z$-position of the Rockstar-defined host halo center in the DMO run, in units of comoving Mpc/$h$.
- HALO_VX: $x$-physical peculiar velocity of the Rockstar-defined host halo in the DMO run, in units of physical $km/s$.
- HALO_VY: $y$-physical peculiar velocity of the Rockstar-defined host halo in the DMO run, in units of physical $km/s$.
- HALO_VZ: $z$-physical peculiar velocity of the Rockstar-defined host halo in the DMO run, in units of physical $km/s$.
- HALO_JX: $x$-angular momentum of the Rockstar-defined host halo in the DMO run, in units of phsyical $(M_\odot/h) \times (Mpc/h) \times km/s$.
- HALO_JY: $y$-angular momentum of the Rockstar-defined host halo in the DMO run, in units of phsyical $(M_\odot/h) \times (Mpc/h) \times km/s$.
- HALO_JZ: $z$-angular momentum of the Rockstar-defined host halo in the DMO run, in units of phsyical $(M_\odot/h) \times (Mpc/h) \times km/s$.
- HALO_ID: The unique number that we assign to each EDEN host. The user could specific their individual halo numberings.
- HALO_NORM: The disk mass normalization factor with respect to the UM-predicted stellar mass. This parameter is defaulted to 1.0 and it is set to 2.5 for the nine high disk mass subsamples.
- TOT_SNAP: Total number of snapshots of the MW-host galaxy's UM-predicted stellar mass growth history (the _Mstar.txt and _Z.txt files in ./um_sfh/). Note that this number could be smaller than the total number of Rockstar snapshots given the specific formation time of the MW host galaxy.

## 2.3 Force calculation of the disk potential

The implementation of the additional gravitational acceleration from the disk potential in one time step is added in gravtree.c. The disk potential model function is implemented in the new function starting from line 14:

- `double AccRz(double R, double z, int type, double msz, double fstar)`

The variables $R$ and $z$ are the cylindrical coordinates of each particle with respect to the sink particle; the variable `type` specifies the $z$ (`type == 0`) and $R$ (`type == 1`) components of the negative acceleration on the particle; `msz` is the loaded UM-predicted stellar mass that defines the total disk mass (already has `HALO_NORM` folded in from `init.c` or `predict.c`); `fstar` is the loaded stellar-to-gas fraction from NeuUM, which is only used in the `Gas_Guo23` branch (see variable definitions for `Mdsz,Mdgz,MBz`).

Since we are considering the disk potential-sink particle combination as one entity, there are three major steps that relate to updating the additional acceleration correctly in addition to the existing Gadget-2 gravity tree calculation ($N$-body forces):

- The gravitational forces from all other $N-1$ particles in the simulation on the sink particle is naturally taken care of by the gravity tree code. However, in this case, the disk-sink combination does not just have a mass of $M_{\rm Sink}$, but instead $M_{\rm Disk}+M_{\rm Sink}$. So we scale this $N-1$-body gravity tree acceleration by $M_{\rm Sink}/(M_{\rm Disk}+M_{\rm Sink})$.

- We call function `AccRz` and apply the disk acceleration onto each particle. We also sum up the back-reaction force from each particle onto the disk-sink combination. The smooth turn-on of the disk potential from `START_A` to `FORCE_A` is also implemented.

- We apply the back reaction from the step above (combines forces together from all subprocesses/particle domains) on the disk-sink combination.

The above changes are implemented from line 498 to 582 in `gravtree.c`. We re-emphasize that applying the back reaction from all other particles onto the disk potential is the most crucial step such that the disk-sink approach maintains to be dynamically self-consistent throughout the re-simulation.

The logic to update the disk potential parameters and advance to the next time step is added in `predict.c`. We update the disk center before updating the position of the sink particle, given the new accelerations calculated in `gravtree.c`. We update the disk mass and stellar fraction by calling `Ms_UM` and `Fs_Guo` from `disk_mass.h`. These changes are reflected in lines 49 to 92.

Finally, since we assign a fiducial 'Type6' particle type for the sink particle during force evaluation, we adjust it back to 'Type4' during each snapshot

output for consistent I/O behavior. We make changes to `io.c` between lines 39 to 45 and 130 to 136 for this purpose.

# 3 Example steps for running one EDEN simulation

Here we provide a step-by-step guidance for running EDEN on your desired DMO simulation.

## 3.1 Pre-requisites

There are three pre-requisites for your DMO simulation data products before starting any EDEN re-simulation:

- The DMO simulation must be ran with Gadget2 and have at least advanced to your desired starting redshift (`START_A`) for EDEN (e.g., $z = 3$ in our case).
- The DMO simulation must have a halo finder (e.g., Rockstar) run on it such that you could retrieve all necessary information of the host halo at `START_A`.
- The DMO simulation must have a galaxy–halo connection model (e.g., UM) run on it such that you could retrieve the stellar mass history of the host.

## 3.2 Setting up EDEN

Once you are ready with the DMO data products, you can begin setting up EDEN:

- Overwrite all variables in `halo_props.h` with your halo's info at the starting snapshot.
- Copy the `_Mstar.txt` and `_Z.txt` files for your halo to `./um_sfh/`. Please note that the file names should match `HaloXXX8_Mstar.txt` and `HaloXXX8_Z.txt` where `XXX` is the `HALO_ID` that you chose in `halo_props.h`.

- Modify `GSL_INCL, GSL_LIB, FFTW_INCL, FFTW_LIB` file paths in `Makefile`. Run `make clean && make` to generate the executable binary `Gadget2disk`.
- Make a directory for your simulation output path e.g., `/path/to/HaloXXX`.
- Copy the simulation binary `Gadget2disk` to `/path/to/HaloXXX`.
- Make a directory `/path/to/HaloXXX/ic_disk`. Copy the snapshot in your DMO simulation that you want to use as the initial condition for EDEN into `/path/to/HaloXXX/ic_disk`.
- Make a directory `/path/to/HaloXXX/output_disk` that stores the snapshot outputs of EDEN.
- Copy `parameters.tex` from the EDEN code repository to `/path/to/HaloXXX`. Modify `InitCondFile` to the appropriate file prefix for your simulation. Modify this condition again if you ever need to restart the simulation. Of course, replace the actual snapshot file in `ic_disk` before doing so.
- Copy `submit_gadget.sh` from the EDEN code repository to `/path/to/HaloXXX`. Modify everything to suit your system's specifications.

## 3.3 Running EDEN

Submit the shell script `submit_gadget.sh` to your compute cluster. If everything goes well, you should expect your EDEN outputs to show up in the directory `/path/to/HaloXXX/output_disk`.

## 3.4 Changing the stellar mass history model

If the user wants to experiment with different stellar mass or fraction models other than UM and NeuUM, one could easily do so by modifying the `Ms_UM` and `Fs_Guo` functions in `disk_mass.c`. To verify whether your new stellar mass history and stellar fraction models behave in the way you would expect, we have provided a testing script `test_um.c`, which can be compiled and run using the following command:

- `gcc disk_mass.c test_um.c -o test_um -lgsl -lgslcblas -lm && ./test_um`