MOCAT-SSEM Multi Mass Species Workbook

Description

This demo is intended to provide a reference example of a more complicated model. It includes multi-mass-binned species, empirical launch rates, and indicator variables.

How to use it:

This MATLAB Workbook is intended to be a straightforward, clear and interactive way to build familiarity with the MOCAT-SSEM model framework. The code is explained in comments in as much detail as possible for ease of use and clarity. **This workbook is intended to be a simple introduction and does not describe all functionality of the MOCAT-SSEM framework.**

Each model is built with a top-level simulation object. The simulation object has a scen_properties object, which sets global parameters, and a set of species with their own species_properties objects that define parameters for that species. The simulation object will automatically handle equation compilation and numerical integration of the model based on the provided parameters.

The recommended workflow beings with defining the scen_properties object and species before assembling and running the model.

To begin, you set the properties of the scenario you want to run such as the altitudes to be considered and atmospheric model to use, in **Define Scenario Properties**. Simply change the values that the variables are set to.

You can set the properties of the species, such as the mass of derelict objects and launch rate of satellites in **Define Species Properties**.

The sections **Collision Modeling** and **Define Initial population** require no user input but are necessary code. The first calculates the population gain/loss between species as they collide. The second, takes in the scenario properties you set earlier and a list of satellites from the file ('initialized.mat') and creates an array (x0) of of the number of species in each shell for the model at the start of the simulation.

You can select which, if any, visualizations of results you want to see in the final section, **Make and Run Simulation**. The plot for the evolution of total species numbers in each shell is already selected in my_sim.total_species_evol_vis();. To see the other visualizations, simply uncomment them (remove the % at the start of the line). You can also export numerical results for additional analysis as you desire.

Once you've finished all the inputs you'd like, simply run this workbook (F5) and it will begin.

Set up paths and clear workspace to run file.

This is necessary so that the model can find the necessary classes and functions. Clearing your workspace is not strictly necessary, but using an existing workspace may result in unintended interactions.

```
clear
clc

% The demo workbooks are in a subfolder of the main directory. We need to
% add the path recursively for the main directory.
```

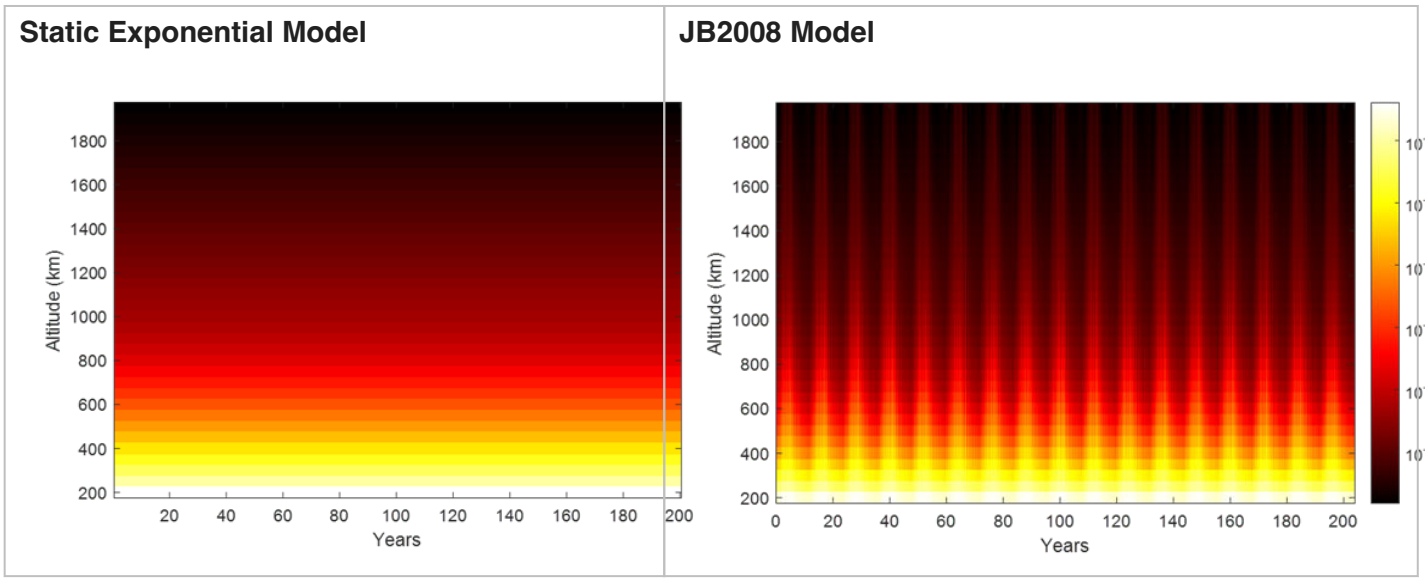
```
fileName = matlab.desktop.editor.getActiveFilename;
[folder, ~, ~] = fileparts(fileName);
folder = fullfile(folder, '..');
addpath(fileparts(folder));
addpath(genpath(folder));
cd(folder);
```

Define Scenario Properties

Set simulation parameters such as start time, duration, and time steps:

```
scenario_properties = MOCATSSEM_Scen_Prop_Cons( ...
    start_date = datetime(2022,1,3,0,0,0), ... # Starting date (year,
month, day, hour, minute, second)
    simulation_duration = 100.0, ... % Years of simulation to run
    steps = 200, ... % Number of steps to run in simulation
    min_altitude = 200, ... % Maximum altitude shell in km
    max_altitude = 1400, ... % Minimum altitude shell in km
    shells = 40, ... % Number of altitude shells
    ... % We create this many equally spaced shells between the minimum and
maximum altitudes
    delta = 10, ... %ratio of the density of disabling to lethal debris
    ... % this term, delta, considers the possibility that disabling
collisions can generate new derelicts
    integrator = @ode15s); % MATLAB integrator used to run simulation
    % we typically use ode15s because often, models with many shells become
stiff diff eqs.
```

You can choose between a static exponential or JB2008-based atmospheric density models. The static exponential model is not time dependent, and will run faster than JB2008. The JB2008 model is time/space-weather dependent and more accurate, but runs slower. You can see spikes in the JB2008 graph as the density changes over time.



Set atmospheric density model:

```
atm_density_choice = "Static Exponential"
```

```
atm_density_choice =
"Static Exponential"
```

```
switch atm_density_choice
    case "JB2008"
        scenario_properties.density_filepath =
fullfile(".","Drag","Atmosphere Model", "JB2008", "Precomputed",
"dens_highvar_2000.mat");
        scenario_properties.dens_model = @JB2008_dens_func;
        scenario_properties.time_dep_density = true;
    case "Static Exponential"
        scenario_properties.dens_model = @static_exp_dens_func;
        scenario_properties.time_dep_density = false;
end
```

Processes and saves the scenario properties you just set to be used in the simulation:

```
scen_properties = scen_properties_class(scenario_properties);
```

To access and edit ODE options, e.g.:

- scen_properties.options.AbsTol = 1e-2;
- scen_properties.options.RelTol = 1e-2;

```
% scen_properties.options
```

Define Species Properties

Note that null launch functions are used for each species. This is because the null launch function will be later replaced with an empirical launch function prior to model build. However, object properties need to be defined for the method that will bin the future launch traffic into the chosen model species.

Unlike the MOCAT-3 and MOCAT-4 workbooks, this version also reuses the species_properties structure between species, changing parameters as necessary rather than clearing it entirely. This is okay as long as you are careful and understand what you are doing.

Choose your launch function to determine the rate of new active satellites (species Su are inserted into orbit).

```
launch_func_choice = "Null"
```

```
launch_func_choice =
"Null"
```

```
switch launch_func_choice
    case "Null"
        launch_func_sat = @launch_func_null;
    case "Constant"
        launch_func_sat = @launch_func_constant;
        lambda_constant = num2cell(500 * rand(scen_properties.N_shell,1));
```

```
end
```

Define drag function:

```
my_drag_func = @drag_func;
```

species_properties is a struct object that takes in your inputs for the properties of a species. Once all the properties of a certain species have been set, they are saved and processed for the model by the species function. Then we clear species_properties at the end of each section so we can use it again for the next species (and we dont get properties left over from the last species).

```
species_properties = struct;
```

List of Species Properties

- *sym_name* - Symbolic name for species
- *Cd* - Coefficient of drag, unitless
- *mass* - Mass of species [kg]
- *radius* - Radius of species [m]
- *A* - Area of species in [m^2]
- *amr* - Area/mass ratio in [m^2/kg]
- *beta* - Ballistic coefficient
- *slotted* - Whether species is slotted or not [bool]
- *drag_effected* - Whether object decreases altitude due to drag or not [bool]

Only for unslotted satellites (Su)

- *lambda_constant* - Determines launch rate if launch function is Constant
- *maneuverable* - Whether species is maneuverable or not [bool]
- *deltat* - Lifetime of spacecraft in years
- *Pm* - Post-mission disposal efficacy, if post-mission disposable is successful, the satellite is removed from the simulation. If it fails, the satellite transitions to a derelict [Float [0,1]]
- *alpha* - Efficacy of Collision Avoidance vs. inactive
- alpha_active = 0.01; % Efficacy of Collision Avoidance vs. other active
- alpha = 2e-3; % Efficacy of Collision Avoidance vs. inactive
- *alpha_active* - Efficacy of Collision Avoidance vs. other active
- *RBflag* - No Rocket Bodies yet.

Only for slotted satellites (S)

- *slotting_effectiveness* - proportion of conjunctions between pairs of slotted spacecraft that are mitigated by the slotting system

Define species properties for unslotted satellites

```
% Su
species_properties.sym_name = "Su";
```

```
species_properties.drag_effected = false; % bool, does object decrease
altitude due to drag
species_properties.active = true; % bool, use alpha_active for collisions
vs. trackable
species_properties.maneuverable = true; % bool, use alpha_active for
collisions vs. trackable
species_properties.trackable = true; % bool, others can avoid this with
alpha
species_properties.RBflag = 0; % No Rocket Bodies yet.
species_properties.Cd = 2.2; % unitless
species_properties.mass = [260 473]; % kg
species_properties.radius = [0.728045069 2.077681285]; % m
species_properties.A = [1.6652 13.5615];
species_properties.amr = species_properties.A./species_properties.mass;  %
m^2/kg
species_properties.beta = species_properties.Cd*species_properties.amr; %
ballistic coefficient

% Orbit Properties
species_properties.slotted = false; % bool
species_properties.slotting_effectiveness  = 1.0; % Float [0,1],

% Capabilities
species_properties.drag_effected = false; % bool, does object decrease
altitude due to drag
species_properties.active = true; % bool, use alpha_active for collisions
vs. trackable
species_properties.maneuverable = true; % bool, use alpha_active for
collisions vs. trackable
species_properties.trackable = true; % bool, others can avoid this with
alpha.
species_properties.deltat = [8 8]; % lifetime of spacecraft in years
species_properties.Pm = .65; % Post-mission disposal efficacy Float [0,1]
species_properties.alpha = 1e-5; % Efficacy of Collision Avoidance vs.
inactive
species_properties.alpha_active = 1e-5; % Efficacy of Collision Avoidance
vs. other active
species_properties.RBflag = 0; % No Rocket Bodies yet.
if launch_func_choice == "Constant"
    species_properties.lambda_constant = lambda_constant;
end
```

Create species instance with a launch function, post-mission disposable function, drag function, species properties which are specific to satellites, and scenario properties which are specific to the simulation. If post-mission disposable is successful (based on probabily Pm), the satellite is removed from the simulation. If it fails, the satellite transitions to a derelict.

```
Su_species = multi_property_species(launch_func_sat, @pmd_func_sat,
my_drag_func, species_properties, scen_properties).species_list;
```

```
for i = 1:length(Su_species)
    Su_masses(i) = Su_species(i).species_properties.mass;
    Su_radii(i) = Su_species(i).species_properties.radius;
end
```

Define species properties for slotted satellites

```
% S
species_properties.sym_name = "S";
species_properties.drag_effected = false; % bool, does object decrease
altitude due to drag
species_properties.active = true; % bool, use alpha_active for collisions
vs. trackable
species_properties.maneuverable = true; % bool, use alpha_active for
collisions vs. trackable
species_properties.trackable = true; % bool, others can avoid this with
alpha
species_properties.RBflag = 0; % No Rocket Bodies yet.
species_properties.Cd = 2.2; % unitless
species_properties.mass = [1250 750 148]; % kg
species_properties.radius = [4 2 0.5]; % m
species_properties.A = [pi*species_properties.radius.^2];

% For this amr and others, we use the first mass so multi_property_species
will
% scale it automatically for us using the scaled radii.
species_properties.amr = species_properties.A./species_properties.mass;  %
m^2/kg
species_properties.beta = species_properties.Cd*species_properties.amr; %
ballistic coefficient

% Orbit Properties
species_properties.slotted = true; % bool
species_properties.slotting_effectiveness  = 1.0; % Float [0,1],

% Capabilities
species_properties.drag_effected = false; % bool, does object decrease
altitude due to drag
species_properties.active = true; % bool, use alpha_active for collisions
vs. trackable
species_properties.maneuverable = true; % bool, use alpha_active for
collisions vs. trackable
species_properties.trackable = true; % bool, others can avoid this with
alpha.
species_properties.deltat = [8]; % lifetime of spacecraft in years
species_properties.Pm = .90; % Post-mission disposal efficacy Float [0,1]
species_properties.alpha = 1e-5; % Efficacy of Collision Avoidance vs.
inactive
species_properties.alpha_active = 1e-5; % Efficacy of Collision Avoidance
vs. other active
```

```
species_properties.RBflag = 0; % No Rocket Bodies yet.
if launch_func_choice == "Constant"
    species_properties.lambda_constant = lambda_constant;
end
S_species = multi_property_species(launch_func_sat, @pmd_func_sat,
my_drag_func, species_properties, scen_properties).species_list;

for i = 1:length(S_species)
    S_masses(i) = S_species(i).species_properties.mass;
    S_radii(i) = S_species(i).species_properties.radius;
end
```

Define species properties for Sns 3U cubesat with no station_keeping

```
% Sns
species_properties.sym_name = "Sns";
species_properties.Cd = 2.2; % unitless
species_properties.deltat = 3;
species_properties.mass = 6; % kg
species_properties.radius = 0.105550206; % m
species_properties.A = 0.035; % m^2
species_properties.amr = species_properties.A./species_properties.mass;  %
m^2/kg
species_properties.beta = species_properties.Cd*species_properties.amr; %
ballistic coefficient
species_properties.slotted = false; % bool
species_properties.active = true; % bool, use alpha_active for collisions
vs. trackable
species_properties.maneuverable = false; % bool, use alpha_active for
collisions vs. trackable
species_properties.Pm = 0; % Post-mission disposal efficacy Float [0,1]

Sns_species = species(@launch_func_null, @pmd_func_sat, my_drag_func,
species_properties, scen_properties);

for i = 1:length(Sns_species)
    Sns_masses(i) = Sns_species(i).species_properties.mass;
    Sns_radii(i) = Sns_species(i).species_properties.radius;
end
```

Define species properties for debris.

```
% N
% Assuming al spheres of 1 cm, 10cm diameter, 15 kg
species_properties.sym_name = "N";
species_properties.Cd = 2.2; % unitless
species_properties.deltat = NaN;
species_properties.mass =   [0.00141372    0.5670    S_masses Su_masses
Sns_masses];
```

```
species_properties.radius = [0.01          0.1321    S_radii  Su_radii
Sns_radii]; % m
species_properties.A = [pi*species_properties.radius.^2];
species_properties.amr = species_properties.A./species_properties.mass;  %
m^2/kg
species_properties.beta = species_properties.Cd*species_properties.amr; %
ballistic coefficient
species_properties.drag_effected = true; % bool, does object decrease
altitude due to drag
species_properties.maneuverable = false; % bool, use alpha_active for
collisions vs. trackable
species_properties.active = false;
species_properties.slotted = false; % bool
species_properties.slotting_effectiveness  = 1.0; % Float [0,1]
species_properties.Pm = 0; % Post-mission disposal efficacy Float [0,1]
species_properties.alpha = 0; % Efficacy of Collision Avoidance vs. inactive
species_properties.alpha_active = 0; % Efficacy of Collision Avoidance vs.
other active

N_species = multi_property_species(@launch_func_null, @pmd_func_derelict,
my_drag_func, species_properties, scen_properties).species_list;
```

Define species properties for rocket bodies.

```
species_properties.sym_name = "B";
species_properties.RBflag = 1;
species_properties.Cd = 2.2; % unitless
species_properties.mass = [1783.94]; % kg
species_properties.radius = [2.687936011]; % m
species_properties.A = [22.6980069221863]; % m^2
species_properties.amr = species_properties.A/species_properties.mass;  %
m^2/kg
species_properties.beta = species_properties.Cd*species_properties.amr; %
ballistic coefficient
species_properties.drag_effected = true; % bool, does object decrease
altitude due to drag
species_properties.maneuverable = false; % bool, use alpha_active for
collisions vs. trackable
species_properties.active = false;

B_species = species(@launch_func_null, @pmd_func_none, my_drag_func,
species_properties, scen_properties);
```

We now chose to pair the various satellites to matching debris species for derelicts.

```
debris_species = pair_actives_to_debris(scen_properties, [S_species,
Su_species, Sns_species], N_species);
```

```
Pairing the following active species to debris classes for PMD modeling...
    "S_148kg"    "S_750kg"    "S_1250kg"    "Su_260kg"    "Su_473kg"    "Sns"
```

```
Matched species S_148kg to debris species N_148kg.
Matched species S_750kg to debris species N_750kg.
Matched species S_1250kg to debris species N_1250kg.
Matched species Su_260kg to debris species N_260kg.
Matched species Su_473kg to debris species N_473kg.
Matched species Sns to debris species N_6kg.
    Name: N_0p0014137kg
    Name: N_0p567kg
    Name: N_6kg
    pmd_linked_species: Sns
    Name: N_148kg
    pmd_linked_species: S_148kg
    Name: N_260kg
    pmd_linked_species: Su_260kg
    Name: N_473kg
    pmd_linked_species: Su_473kg
    Name: N_750kg
    pmd_linked_species: S_750kg
    Name: N_1250kg
    pmd_linked_species: S_1250kg
```

Next, we add the species we just defined to the scenario properties.

```
species_list = [S_species, Su_species, N_species, Sns_species, B_species];
scen_properties.species_cell = struct('S', S_species, 'Su', Su_species,
'N', N_species, 'Sns', Sns_species, "B", B_species);
scen_properties.species = species_list;
```

Collision Modeling

Define the collision pair possibilities using mass-binned EVOLVE fragmentation model information. (NOTE: When you have lots of species pairs, like in this case, it will be slow.)

```
scen_properties = make_collision_pairs_SBM(scen_properties);
```

```
Creating species pairs...
Done
```

Define Initial Population

Starting Pop and FLM and displaying starting population

NOTE: compiling the launch and initial population can take a while. After you do it once, you can simply reuse the information if you have not changed your species properties.

```
compile_launch_and_initial_pop = true;
if compile_launch_and_initial_pop
    disp("Compiling initial population and launch information.")

    % Large Constellation Model, you can view/change at:
    % MOCAT-SSEM\Launch\Launch
Traffic\MC_Launch_Traffic\megaconstellationLaunches.xlsx
    % Assumes ongoing replenishment after end of deployment phase
    constellationFile = fullfile(".","Launch","Launch
Traffic","megaconstellationLaunches.xlsx");
```

```
    constellationSheet = "Constellations";

    % Looped Historical Launches (to simulate baseline traffic)
    randomizedRepeatedLaunchFile = "launch_repeatlaunch_2018to2022.mat";

    % Initial Traffic (based on TLEs, DISCOS property data, and
    % interpolation laws for missing fields.  See MOCAT-MC documentation
    % for more information).
    initialPopFile = fullfile(".","Start Population","x0.mat");

    scenarioYrs = 200;
    save_table = true;
    save_dir = fullfile(folder, "Launch", "Launch Traffic");
    recompute_FLC = true;
    objects =
make_MC_population_ASEM_format(constellationFile,constellationSheet,
initialPopFile, randomizedRepeatedLaunchFile, scenarioYrs,
scen_properties.start_date, save_table, save_dir, recompute_FLC);
end
```

```
Compiling initial population and launch information.
Loading starting population...
x0: start: 1 end: 19084
Loading Repeated Randomized Launch...
Regenerating launches for 03-Jan-2022 to 24-Dec-2026.
Regenerating launches for 24-Dec-2026 to 14-Dec-2031.
Regenerating launches for 14-Dec-2031 to 03-Dec-2036.
Regenerating launches for 03-Dec-2036 00:00:00 to 23-Nov-2041 00:00:00.
Regenerating launches for 23-Nov-2041 to 13-Nov-2046.
Regenerating launches for 13-Nov-2046 to 03-Nov-2051.
Regenerating launches for 03-Nov-2051 00:00:00 to 23-Oct-2056 00:00:00.
Regenerating launches for 23-Oct-2056 to 13-Oct-2061.
Regenerating launches for 13-Oct-2061 to 03-Oct-2066.
Regenerating launches for 03-Oct-2066 to 23-Sep-2071.
Regenerating launches for 23-Sep-2071 to 12-Sep-2076.
Regenerating launches for 12-Sep-2076 to 02-Sep-2081.
Regenerating launches for 02-Sep-2081 00:00:00 to 23-Aug-2086 00:00:00.
Regenerating launches for 23-Aug-2086 to 13-Aug-2091.
Regenerating launches for 13-Aug-2091 to 02-Aug-2096.
Regenerating launches for 02-Aug-2096 to 24-Jul-2101.
Regenerating launches for 24-Jul-2101 to 14-Jul-2106.
Regenerating launches for 14-Jul-2106 00:00:00 to 04-Jul-2111 00:00:00.
Regenerating launches for 04-Jul-2111 to 23-Jun-2116.
Regenerating launches for 23-Jun-2116 to 13-Jun-2121.
Regenerating launches for 13-Jun-2121 to 03-Jun-2126.
Regenerating launches for 03-Jun-2126 to 24-May-2131.
Regenerating launches for 24-May-2131 to 13-May-2136.
Regenerating launches for 13-May-2136 to 03-May-2141.
Regenerating launches for 03-May-2141 00:00:00 to 23-Apr-2146 00:00:00.
Regenerating launches for 23-Apr-2146 to 13-Apr-2151.
Regenerating launches for 13-Apr-2151 to 02-Apr-2156.
Regenerating launches for 02-Apr-2156 00:00:00 to 23-Mar-2161 00:00:00.
Regenerating launches for 23-Mar-2161 to 13-Mar-2166.
Regenerating launches for 13-Mar-2166 to 03-Mar-2171.
Regenerating launches for 03-Mar-2171 to 21-Feb-2176.
Regenerating launches for 21-Feb-2176 to 10-Feb-2181.
Regenerating launches for 10-Feb-2181 to 31-Jan-2186.
Regenerating launches for 31-Jan-2186 to 21-Jan-2191.
Regenerating launches for 21-Jan-2191 00:00:00 to 11-Jan-2196 00:00:00.
```

```
Regenerating launches for 11-Jan-2196 to 01-Jan-2201.
Regenerating launches for 01-Jan-2201 to 22-Dec-2205.
Regenerating launches for 22-Dec-2205 to 12-Dec-2210.
Regenerating launches for 12-Dec-2210 to 02-Dec-2215.
Regenerating launches for 02-Dec-2215 to 21-Nov-2220.
Regenerating launches for 21-Nov-2220 to 11-Nov-2225.
Regenerating launches for 11-Nov-2225 00:00:00 to 01-Nov-2230 00:00:00.
rrlf: start: 55000 end: 152807
Loading future constellation population...
Warning: Column headers from the file were modified to make them valid MATLAB identifiers
before creating variable names for the table. The original column headers are saved in the
VariableDescriptions property.
Set 'VariableNamingRule' to 'preserve' to use the original column headers as table variable names.
         2023: Number of constellations ramping: 16          maintaining: 0
         2030: Number of constellations ramping: 33          maintaining: 14
         2037: Number of constellations ramping: 20          maintaining: 34
         2043: Number of constellations ramping: 20          maintaining: 34
         2050: Number of constellations ramping: 7          maintaining: 47
         2057: Number of constellations ramping: 0          maintaining: 54
         2064: Number of constellations ramping: 0          maintaining: 54
         2071: Number of constellations ramping: 0          maintaining: 54
         2078: Number of constellations ramping: 0          maintaining: 54
         2085: Number of constellations ramping: 0          maintaining: 54
         2091: Number of constellations ramping: 0          maintaining: 54
         2098: Number of constellations ramping: 0          maintaining: 54
         2105: Number of constellations ramping: 0          maintaining: 54
         2112: Number of constellations ramping: 0          maintaining: 54
         2119: Number of constellations ramping: 0          maintaining: 54
         2126: Number of constellations ramping: 0          maintaining: 54
         2133: Number of constellations ramping: 0          maintaining: 54
         2139: Number of constellations ramping: 0          maintaining: 54
         2146: Number of constellations ramping: 0          maintaining: 54
         2153: Number of constellations ramping: 0          maintaining: 54
         2160: Number of constellations ramping: 0          maintaining: 54
         2167: Number of constellations ramping: 0          maintaining: 54
         2174: Number of constellations ramping: 0          maintaining: 54
         2181: Number of constellations ramping: 0          maintaining: 54
         2187: Number of constellations ramping: 0          maintaining: 54
         2194: Number of constellations ramping: 0          maintaining: 54
         2201: Number of constellations ramping: 0          maintaining: 54
         2208: Number of constellations ramping: 0          maintaining: 54
         2215: Number of constellations ramping: 0          maintaining: 54
fl_LLC: start: 152808 end: 2178503
Saving file.
```

```
%T =
readtable("initial_population_20230101_launch_every_year_randomtime_megacons
tellationLaunches_forexport2.csv", 'TextType','string');
%T = T(1:9377036,:);
%writetable(T, fullfile(save_dir,
"initial_population_20230101_launch_every_year_randomtime.csv"))

bin_launch_data = true;
if bin_launch_data == true
    [~, pop_name, ~] = fileparts(initialPopFile);
    [~, repeat_name, ~] = fileparts(randomizedRepeatedLaunchFile);
    [~, cons_name, ~] = fileparts(constellationFile);
    cons_name = cons_name + "_" + constellationSheet;
    name = pop_name + "_" + repeat_name + "_" + cons_name;
```

```
    object_file = fullfile(save_dir, name + ".csv");
    %object_file = fullfile(save_dir,
"initial_population_20230101_launch_every_year_randomtime.csv");
    [x0,FLM_steps] = ADEPT_Traffic_Model(object_file, scen_properties);
    save(fullfile(save_dir, "ADEPT_Traffic_Model.mat"), "x0", "FLM_steps")
else
    load(fullfile(save_dir,"ADEPT_Traffic_Model.mat"))
end
```

```
                         Value    Count   Percent
  Non-station-keeping Satellite    36985      1.73%
                    Rocket Body     7187      0.34%
                         Debris    35957      1.68%
      Station-keeping Satellite    36757      1.72%
          Coordinated Satellite    2025696     94.54%
Assigning objects to corresponding mass classes.
    "Coordinated Satellite"    "S"

    "Debris"    "N"

    "Non-station-keeping Satellite"    "Sns"

    "Rocket Body"    "B"

    "Station-keeping Satellite"    "Su"

Assigning objects to corresponding altitude bins.
Assembling future launch information. This may take a while...
Start Time: 12-Jul-2026 20:24:43 Stop Time: 12-Jan-2027 09:20:48
Start Time: 22-Jul-2031 05:45:31 Stop Time: 21-Jan-2032 18:41:36
Start Time: 30-Jul-2036 15:06:19 Stop Time: 30-Jan-2037 04:02:24
Start Time: 09-Aug-2041 00:27:08 Stop Time: 08-Feb-2042 13:23:12
Start Time: 18-Aug-2046 09:47:56 Stop Time: 17-Feb-2047 22:44:01
Start Time: 27-Aug-2051 19:08:44 Stop Time: 27-Feb-2052 08:04:49
Start Time: 05-Sep-2056 04:29:32 Stop Time: 07-Mar-2057 17:25:37
Start Time: 14-Sep-2061 13:50:21 Stop Time: 17-Mar-2062 02:46:25
Start Time: 23-Sep-2066 23:11:09 Stop Time: 26-Mar-2067 12:07:14
Start Time: 03-Oct-2071 08:31:57 Stop Time: 03-Apr-2072 21:28:02
Start Time: 11-Oct-2076 17:52:45 Stop Time: 13-Apr-2077 06:48:50
Start Time: 21-Oct-2081 03:13:34 Stop Time: 22-Apr-2082 16:09:38
Start Time: 30-Oct-2086 12:34:22 Stop Time: 02-May-2087 01:30:27
Start Time: 08-Nov-2091 21:55:10 Stop Time: 10-May-2092 10:51:15
Start Time: 17-Nov-2096 07:15:58 Stop Time: 19-May-2097 20:12:03
Start Time: 27-Nov-2101 16:36:47 Stop Time: 30-May-2102 05:32:51
Start Time: 07-Dec-2106 01:57:35 Stop Time: 08-Jun-2107 14:53:40
Start Time: 16-Dec-2111 11:18:23 Stop Time: 17-Jun-2112 00:14:28
Start Time: 24-Dec-2116 20:39:11 Stop Time: 26-Jun-2117 09:35:16
```

```
disp("Done loading initial population")
```

```
Done loading initial population
```

```
% Building launch functions for each species based on FLM
disp("Building launch functions for each species based on Future Launch
Model...")
```

```
Building launch functions for each species based on Future Launch Model...
```

```
FLM_to_launch_functions(FLM_steps, scen_properties);

disp("Done assigning future launch traffic model.")
```

```
Done assigning future launch traffic model.
```

Make Indicators Variables

Indicator variables are quantities of interest that do not correspond directly to the quantity of a species in a shell. They can be differential equations that need to be integrated alongside species, functions of system states, or even derivatives of system states. This section demonstrates some indicator variables related to potential orbital capacity limitations.

First we make an empty array of the indicator variable class.

```
indicator_var_list = indicator_var_class.empty();
```

Orbital Volume

This looks at estimated numbers of slotted spacecraft that fit in LEO subject to certain assumptions.

```
intrinsic_ind = make_intrinsic_cap_indicator(scen_properties, "distance",
...
                                                'sep_dist', 60, ...
                                                'inc', 40, 'shell_sep',
5, ...
                                                'graph', false);
```

Operational CA

These correspond to estimates for maneuvers, based on the number of mitigated conjunctions simulated by the kinectic theory of gases-based collision estimation model. This is not particularly accurate, but can be a quantity of interest.

This computes on a per spacecraft basis per each species:

```
ca_man_struct = make_ca_counter(scen_properties, "maneuverable",
"trackable", "per_species", true, "per_spacecraft", true);
```

This looks at overall numbers:

```
ca_man_struct_agg = make_ca_counter(scen_properties, "maneuverable",
"trackable", "per_species", false, "per_spacecraft", false);
```

This indicator looks at active spacecraft lost to collisions in each shell:

```
percentage = true;
per_species = false;
short_term_ind = make_active_loss_per_shell_counter(scen_properties,
percentage, per_species);
```

This looks at overall collisions per species:

```
all_col_indicators = make_collisions_per_species(scen_properties);
```

Once you assemble your list of indicator variables, they need to be added to the scen_properties object. There is a separate variable for indicator variables that need to be integrated.

```
indicator_var_list = [intrinsic_ind; ca_man_struct; ca_man_struct_agg;
short_term_ind;all_col_indicators];
scen_properties.indicator_var_list = indicator_var_list;
```

Build Model

The runtime depends on your computer and the number of species in the model. This model should take about 10 minutes or less to run.

`simulation_class` is responsible for building and running the model. The `build_model()` function compiles the equations. It adds together all of the terms (regarding launch, collisions, post-mission disposal, and atmospheric drag).

Building the model is only required once in order to run the model if the user doesn't want to change any input parameters.

```
x0run = reshape(table2array(x0), [], 1);% Pad initial conditions with the
indicator variables
my_sim = simulation_class(scen_properties.species, scen_properties);
my_sim.build_model();
```

```
Building model...
ans =
  simulation_class with properties:

       species_list: [1×15 species]
    scen_properties: [1×1 scen_properties_class]
          equations: [40×15 sym]
            results: []
           xdot_eqs: []
           xdot_fun: []
            var_col: []
    drag_term_upper: []
      drag_term_cur: []
Now processing S_148kg
S_148kg
Now processing S_750kg
S_750kg
Now processing S_1250kg
S_1250kg
Now processing Su_260kg
Su_260kg
Now processing Su_473kg
Su_473kg
Now processing N_0p0014137kg
N_0p0014137kg
Now processing N_0p567kg
```

```
N_0p567kg
Now processing N_6kg
N_6kg
Now processing N_148kg
N_148kg
Now processing N_260kg
N_260kg
Now processing N_473kg
N_473kg
Now processing N_750kg
N_750kg
Now processing N_1250kg
N_1250kg
Now processing Sns
Sns
Now processing B
B
Done building model.
```

Run Simulation

`run_model()` is where the simulation model is actually run. It takes a list of initial conditions (x0).

The function also accepts many optional parameters:

- progressBar - Displays percentage of simulation run in another window - bool
- disp_times - Displays time steps in console - bool
- start_date - Start date of simulation - MATLAB datetime object
- simulation_duration - Years of simulation to run - double
- N_step - Number of simulations steps to run - int

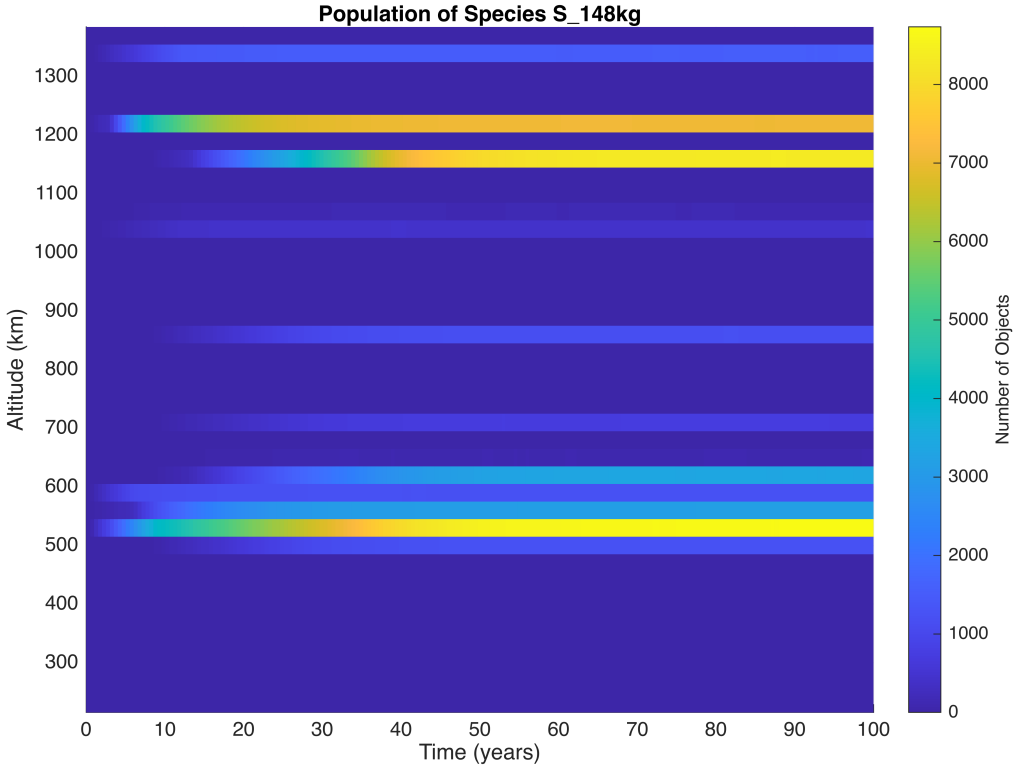
```
my_sim.run_model(x0run, 'progressBar', true, 'disp_times', false)
```

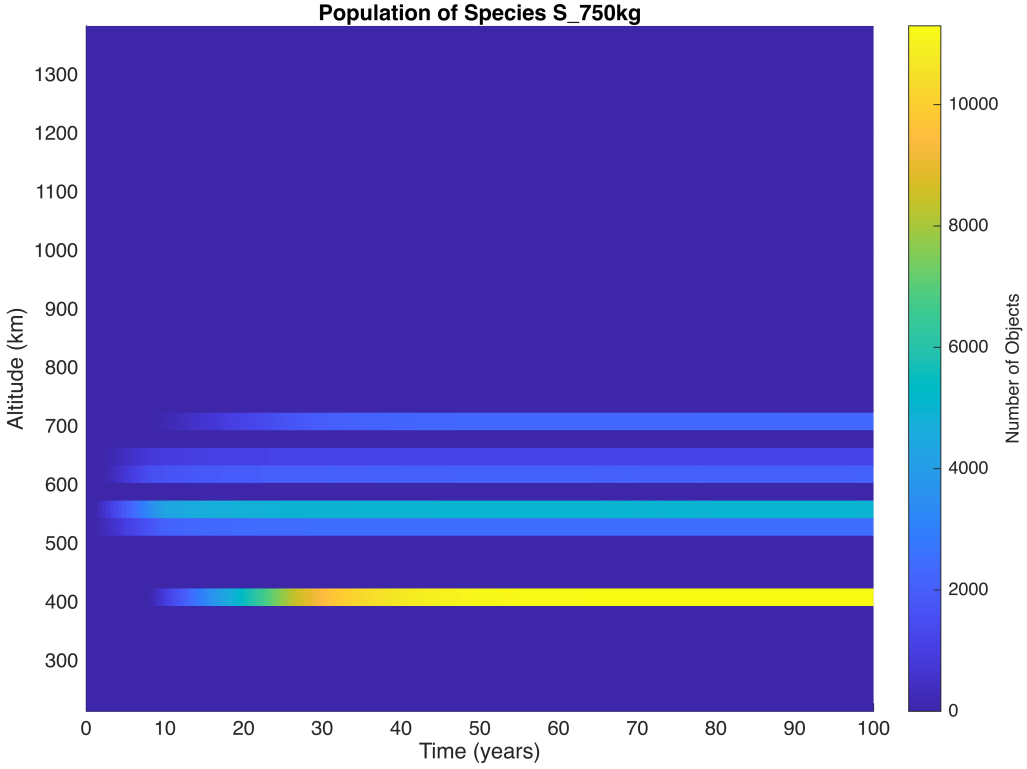
Plots

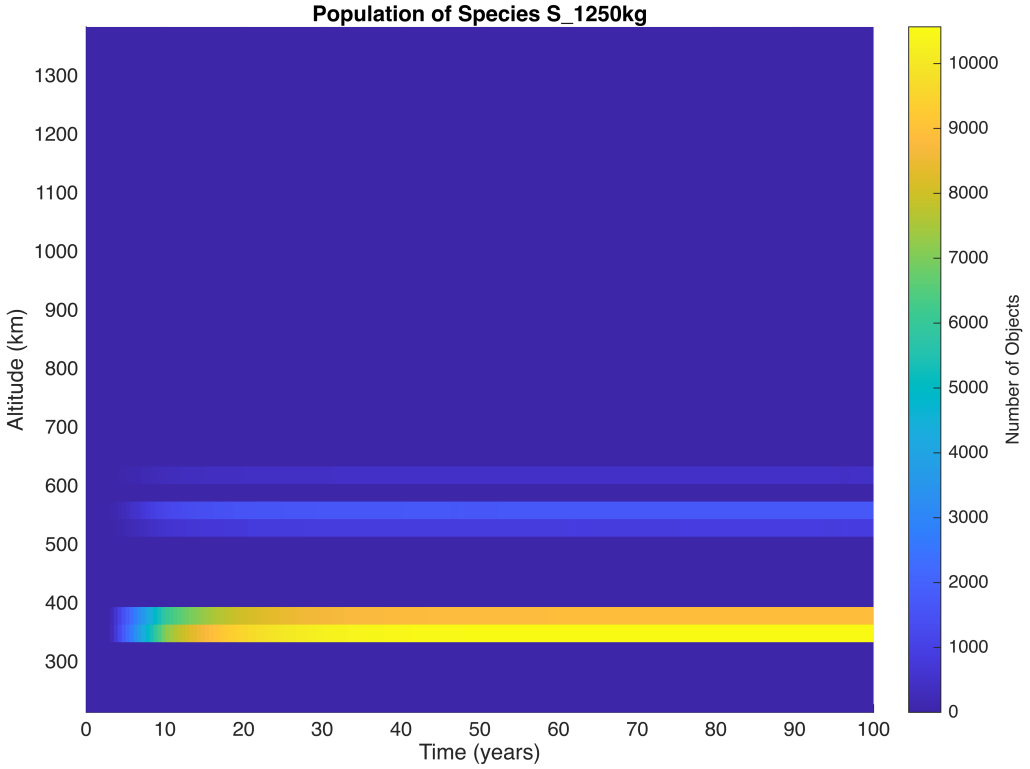
Produces a Figure with subplots for each species population evolution over time

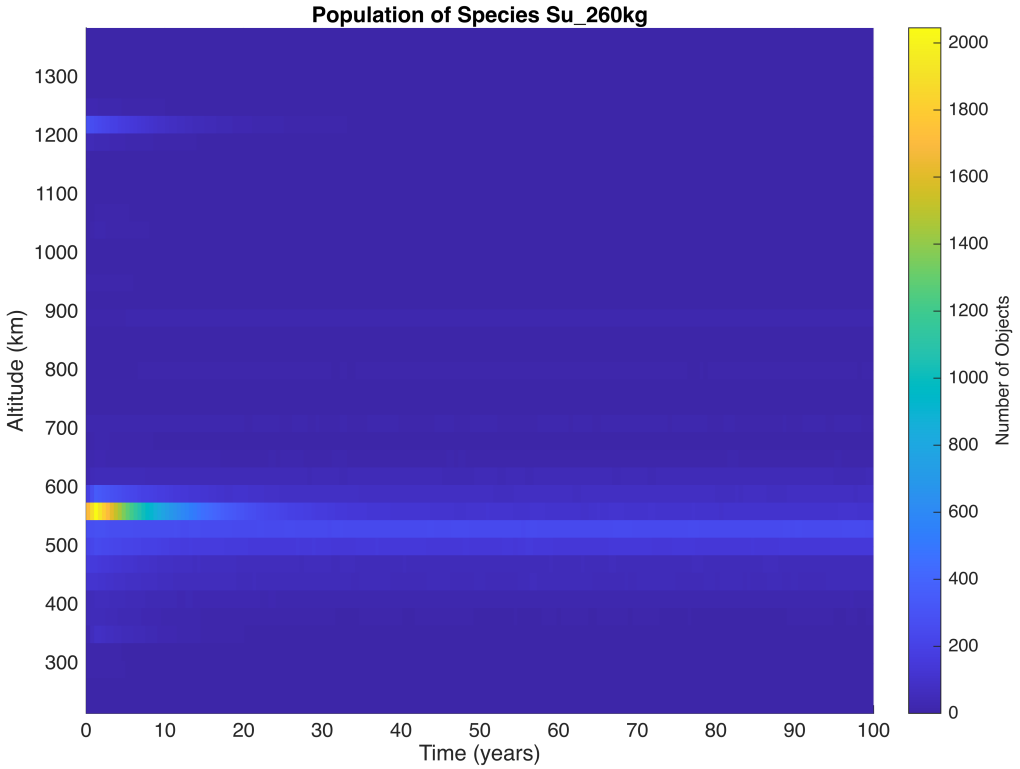
```
my_sim.total_species_evol_vis();
```

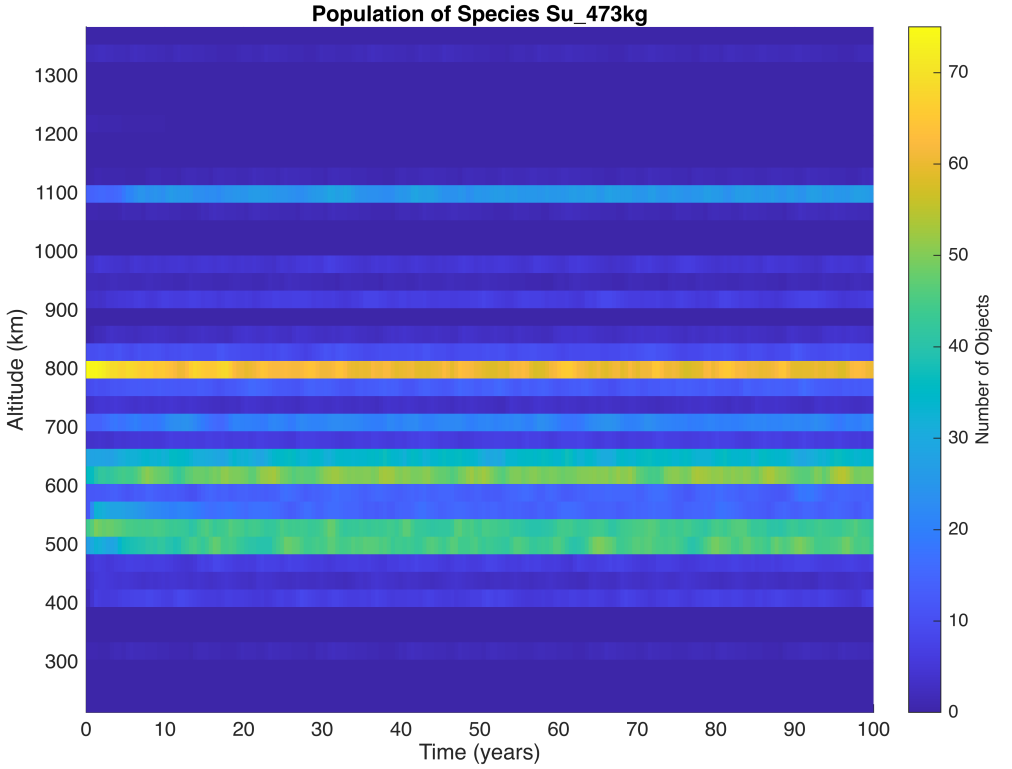
```
Producing visuals for the evolution of each species.
```

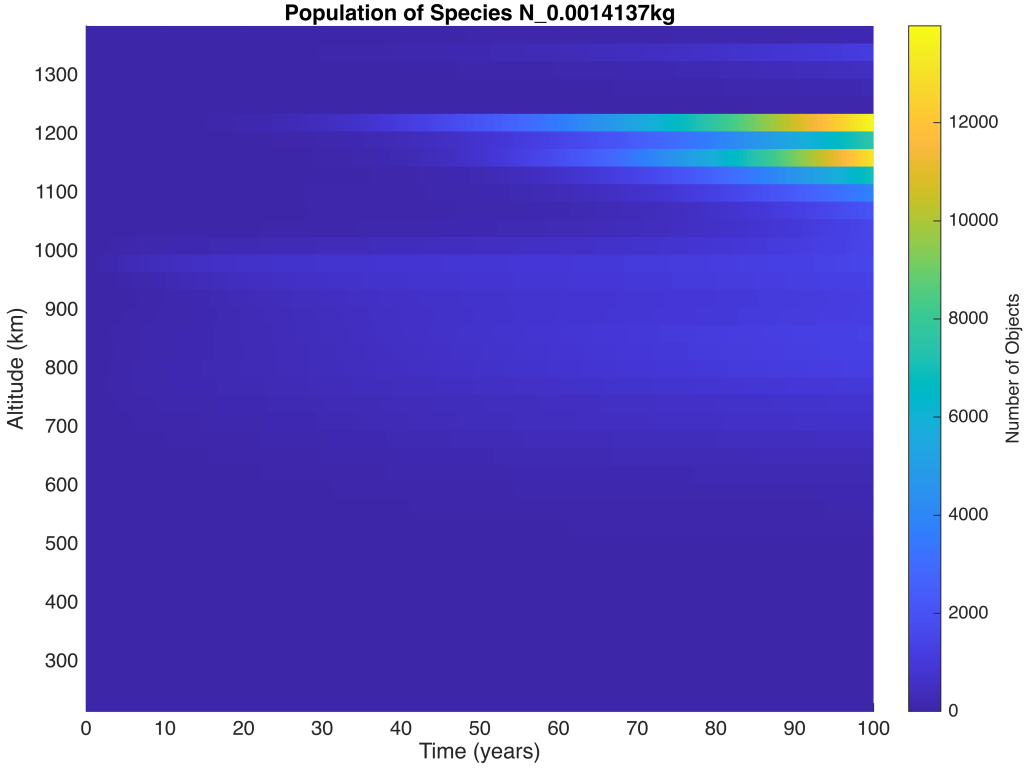

Population of Species S_148kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
8000
7000
6000
5000
4000
3000
2000
1000
0
Number of Objects

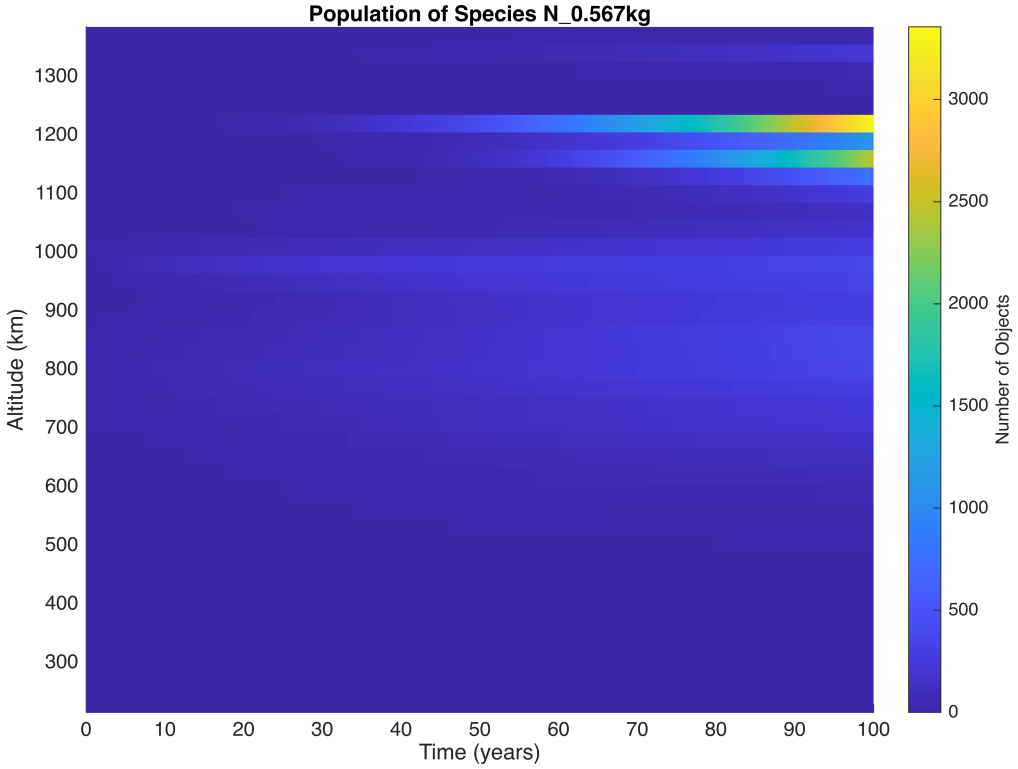

Population of Species S_750kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
10000
8000
6000
4000
2000
0
Number of Objects

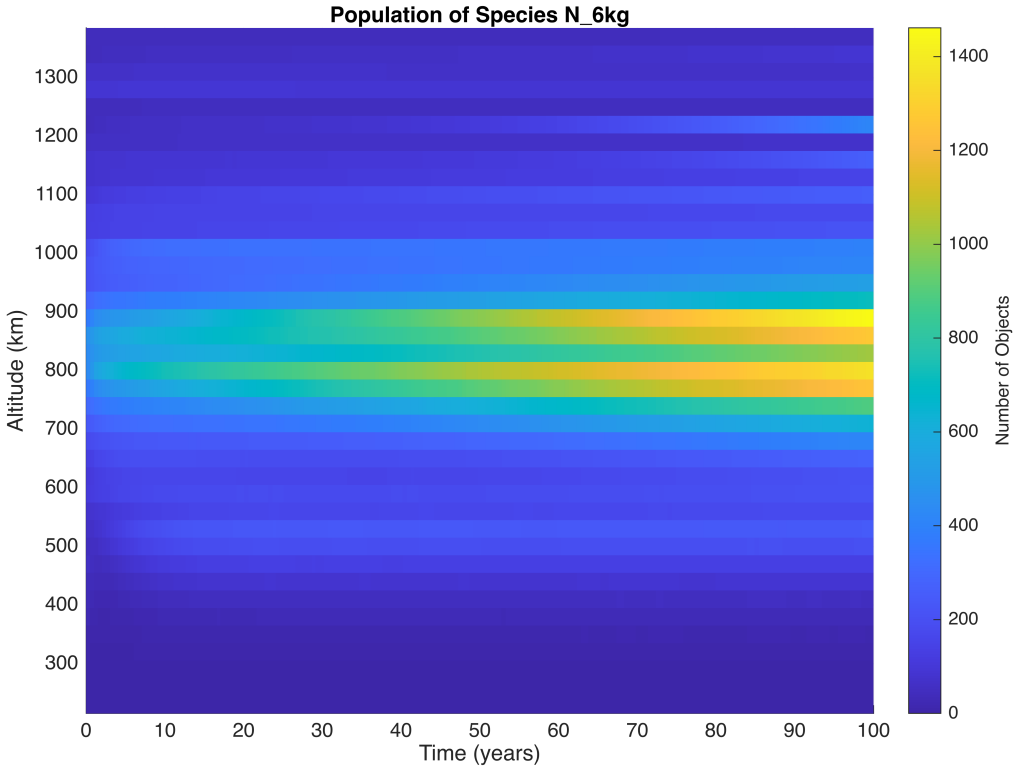

Population of Species S_1250kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
Number of Objects
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0

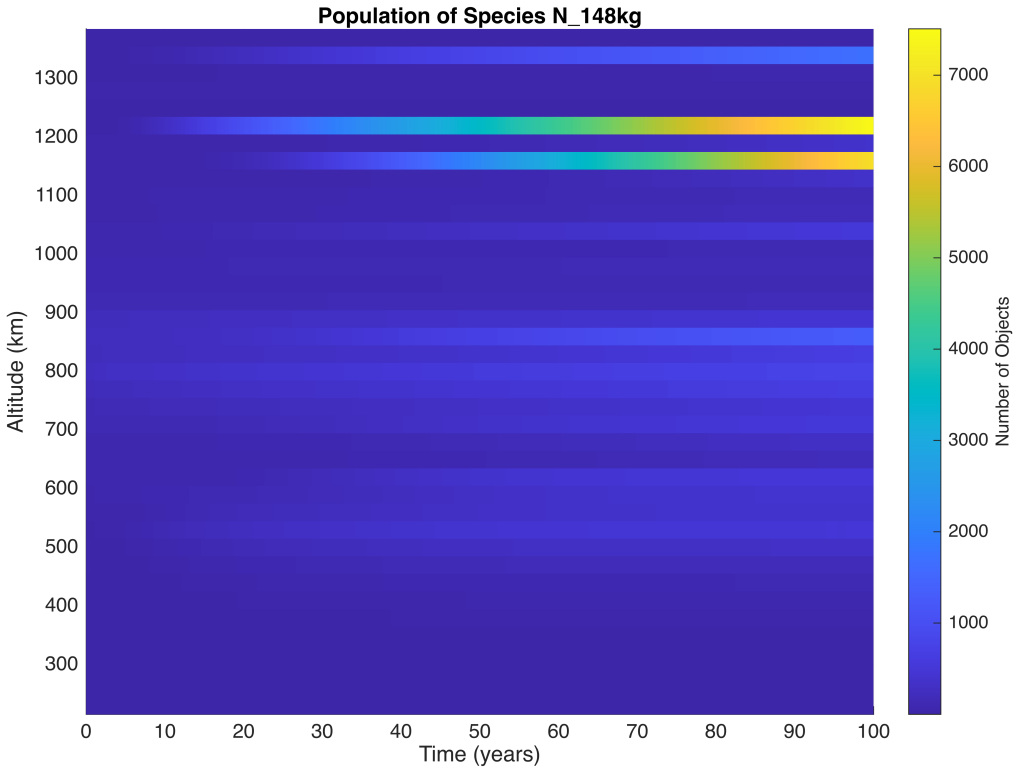

Population of Species Su_260kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
Number of Objects
2000
1800
1600
1400
1200
1000
800
600
400
200
0

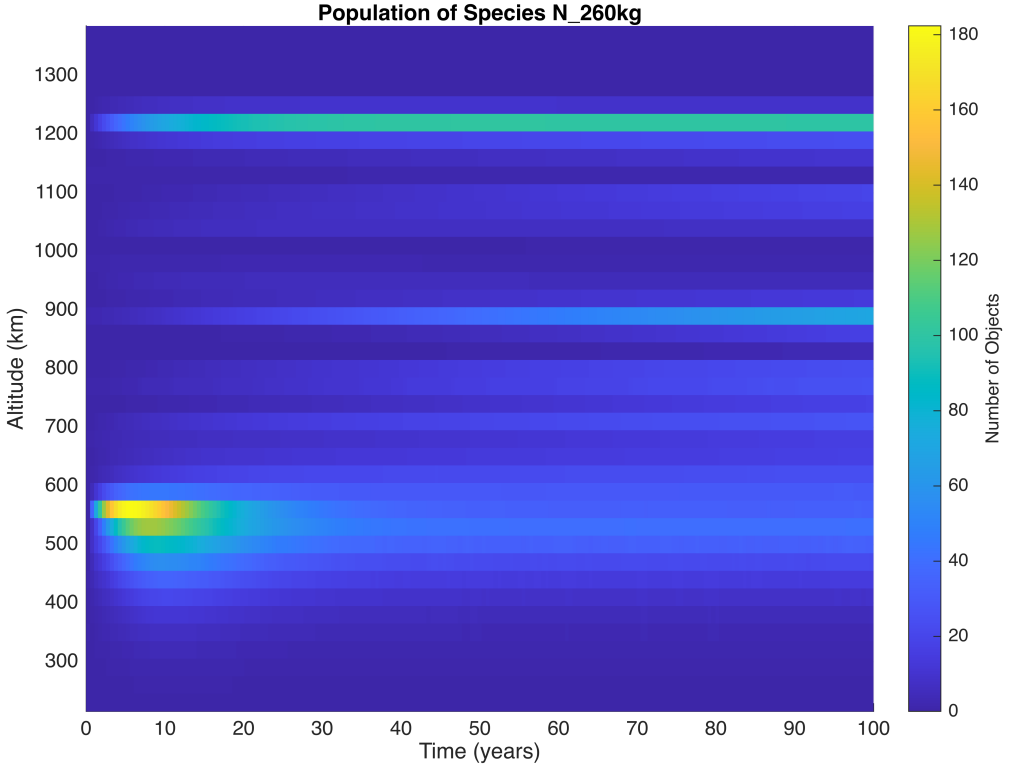

Population of Species Su_473kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
70
60
50
40
30
20
10
0
Number of Objects

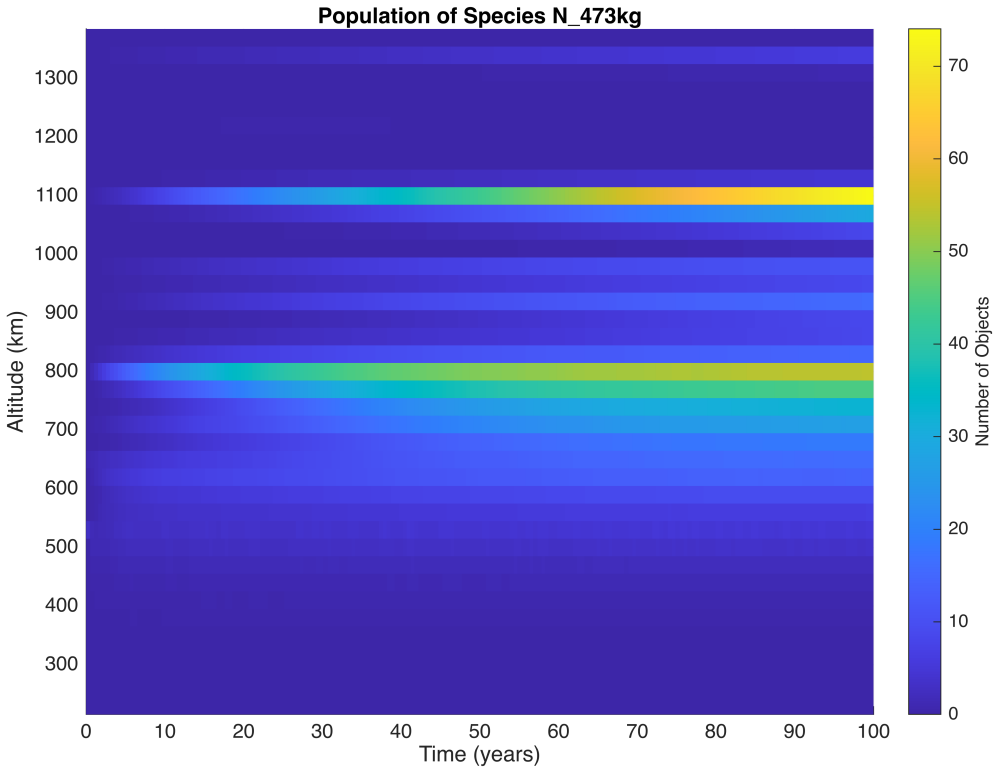

Population of Species N_0.0014137kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
12000
10000
8000
6000
4000
2000
0
Number of Objects

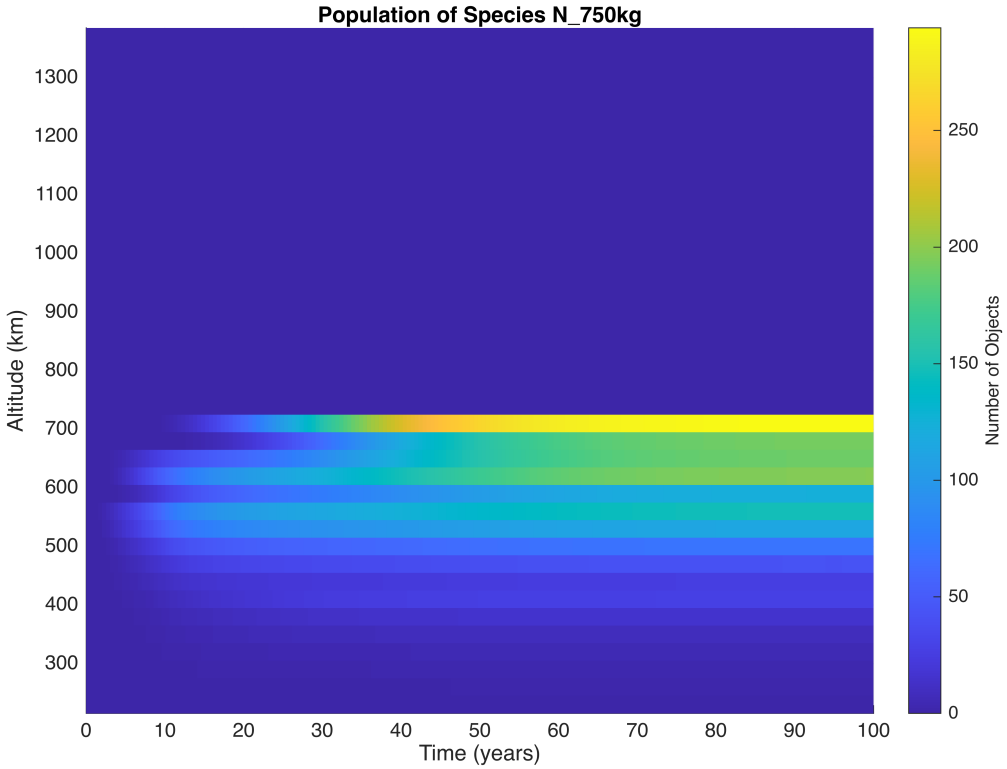

Population of Species N_0.567kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
3000
2500
2000
1500
1000
500
0
Number of Objects

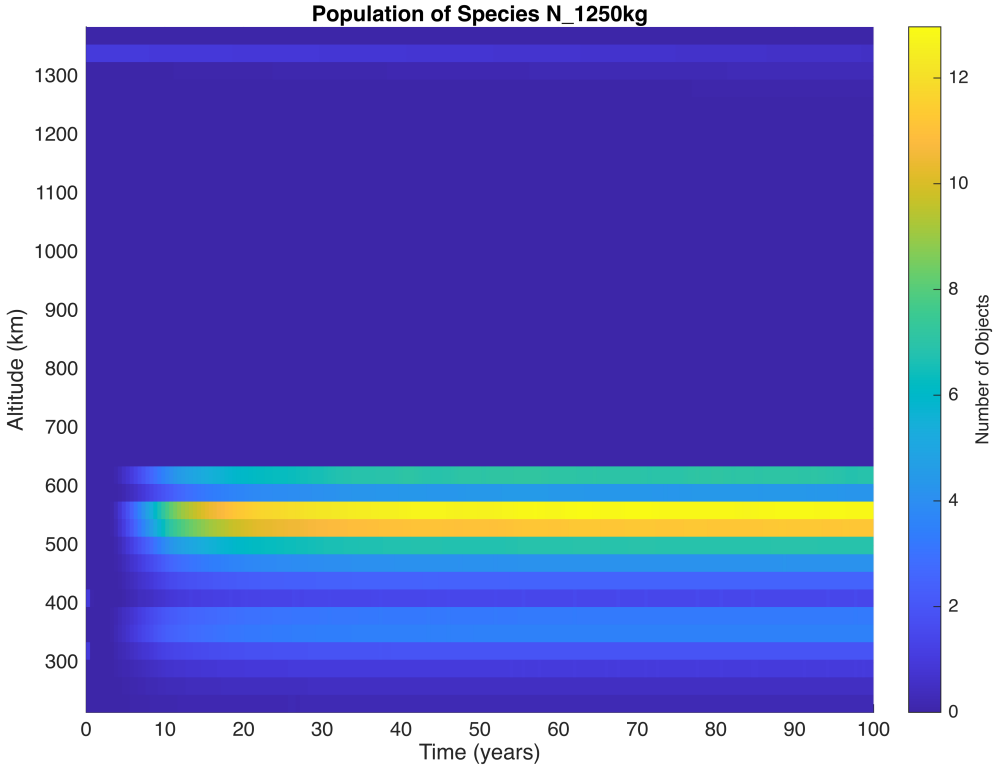

Population of Species N_6kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
1400
1200
1000
800
600
400
200
0
Number of Objects

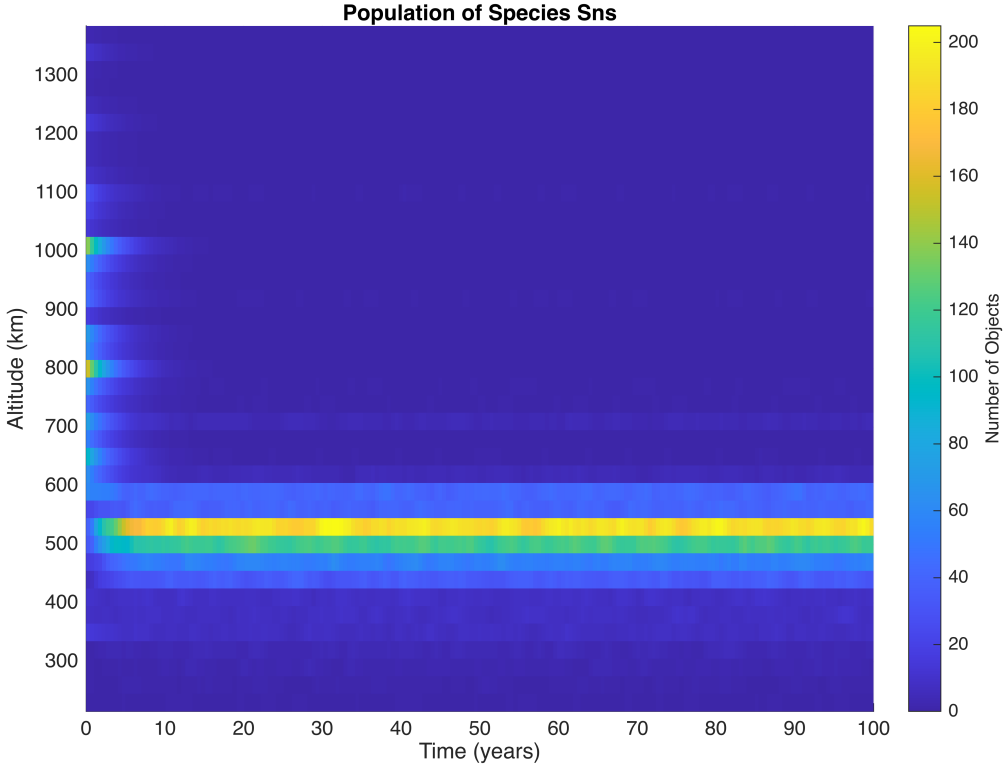

Population of Species N_148kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
7000
6000
5000
4000
3000
2000
1000
Number of Objects

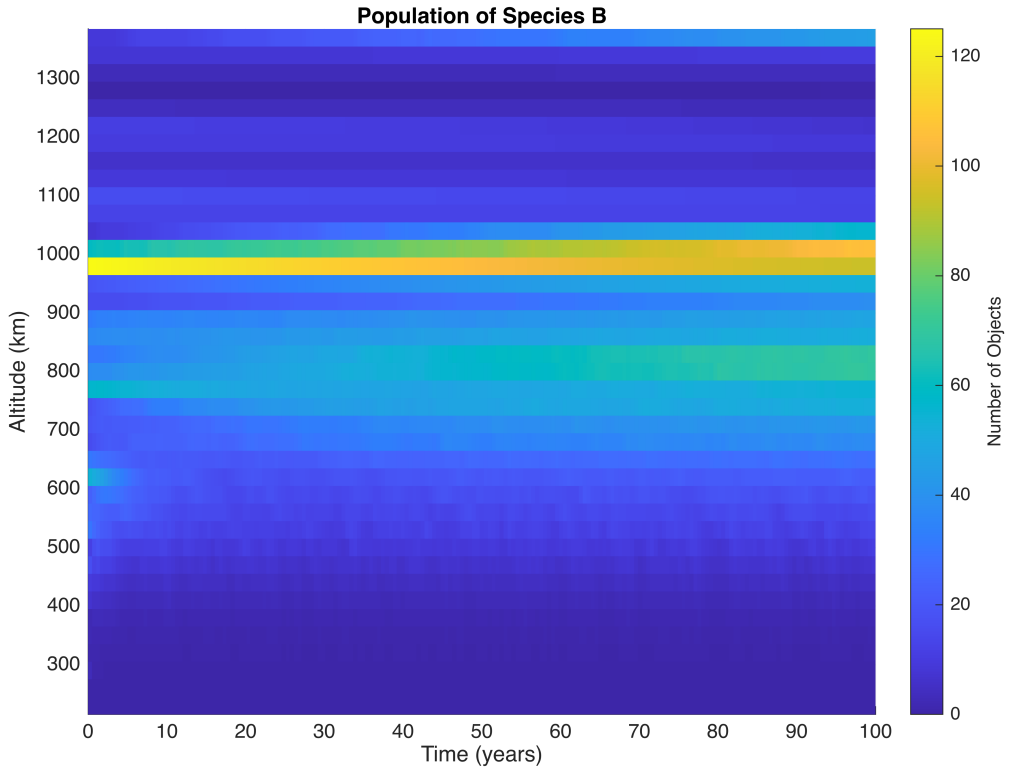

Population of Species N_260kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
180
160
140
120
100
80
60
40
20
0
Number of Objects


Population of Species N_473kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
Number of Objects
70
60
50
40
30
20
10
0


Population of Species N_750kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
Number of Objects
250
200
150
100
50
0


Population of Species N_1250kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
Number of Objects
12
10
8
6
4
2
0

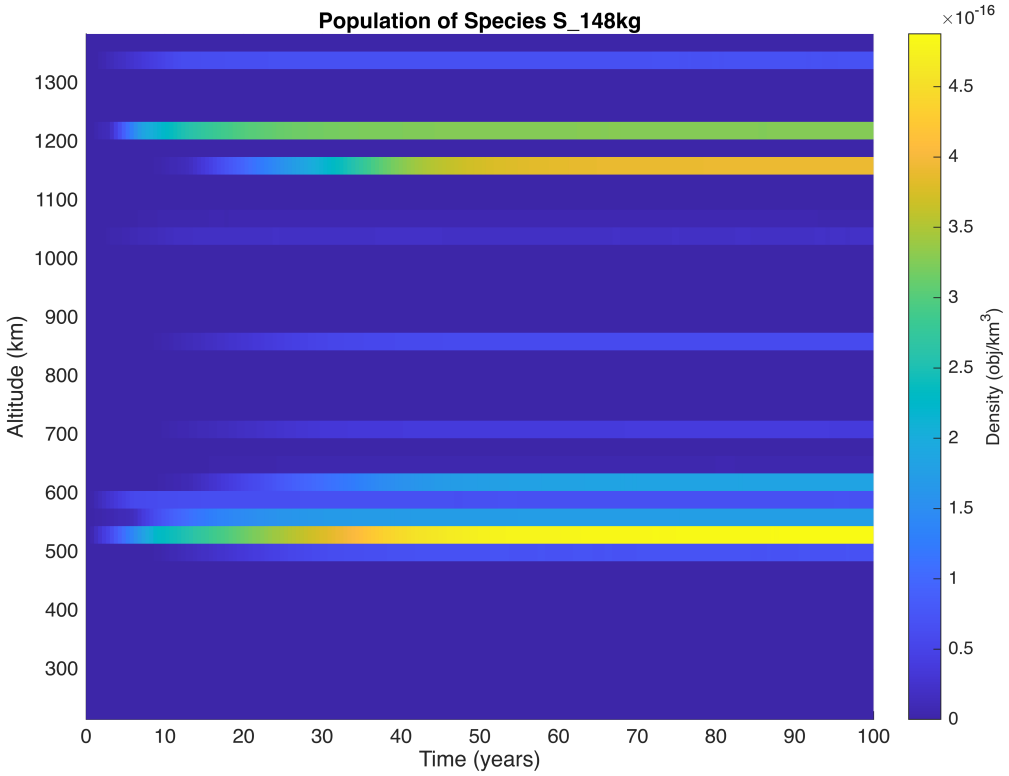

Population of Species Sns
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
Number of Objects
200
180
160
140
120
100
80
60
40
20
0

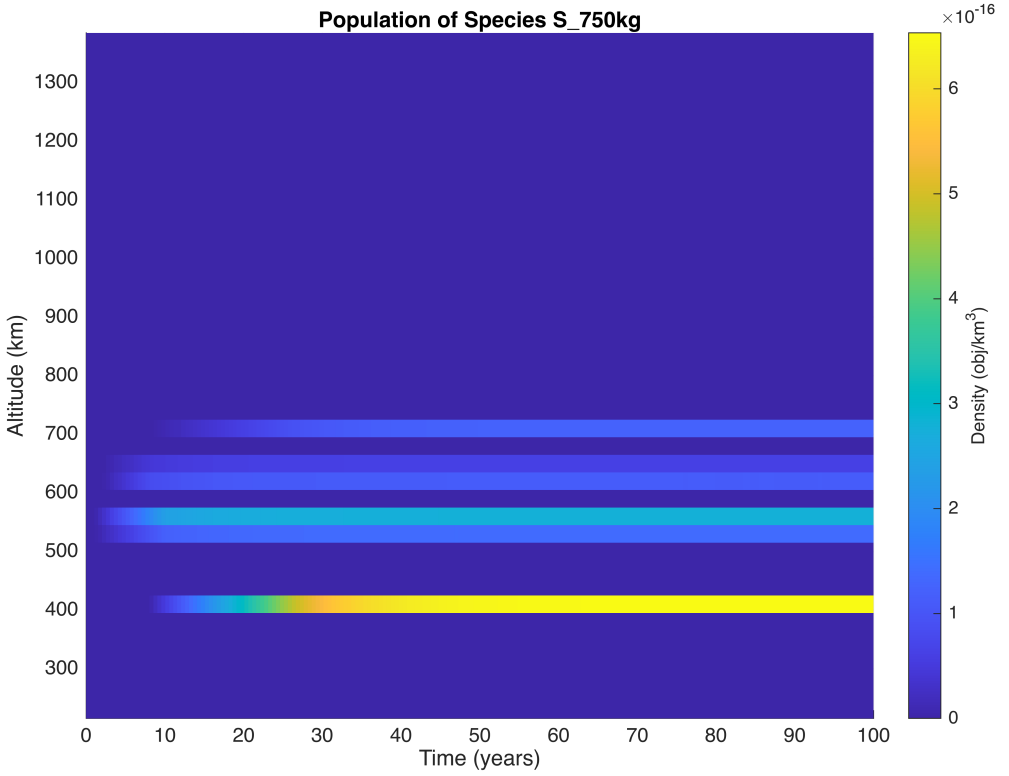


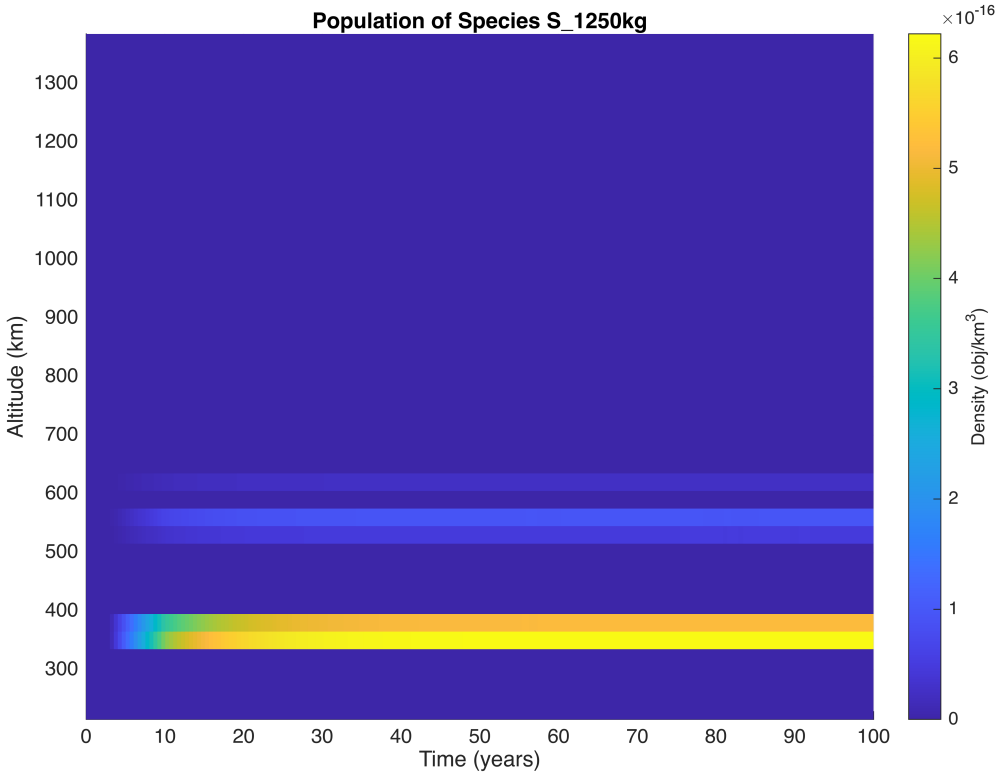
Produces a Figure with subplots for each species population density evolution over time

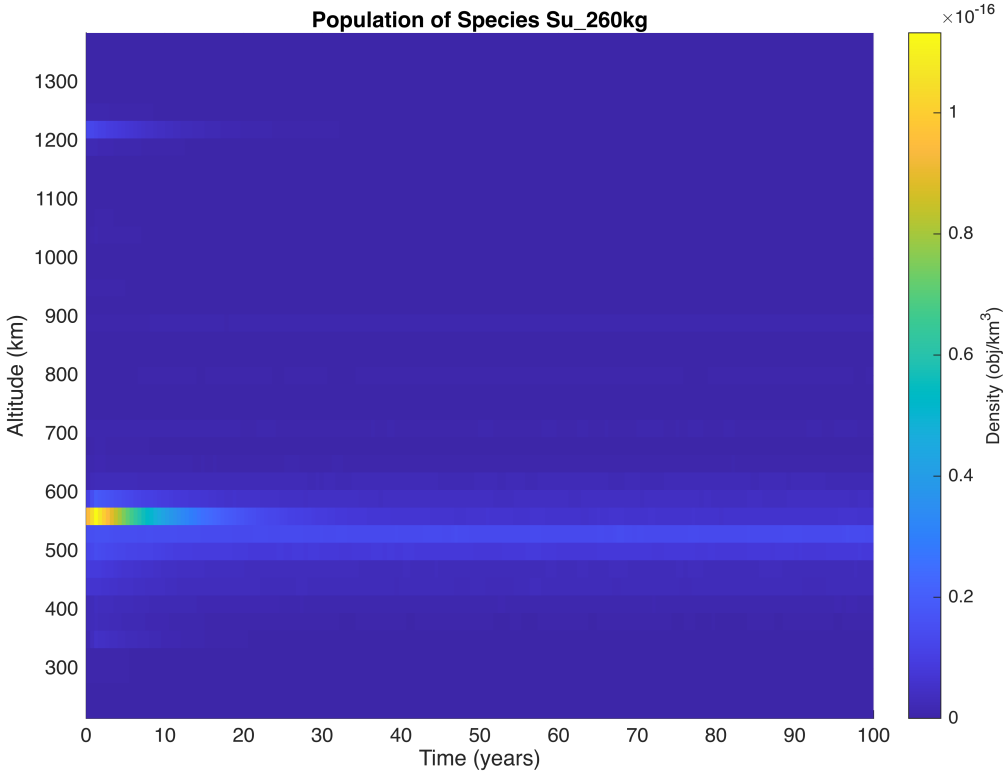
```
my_sim.density_species_evol_vis();
```

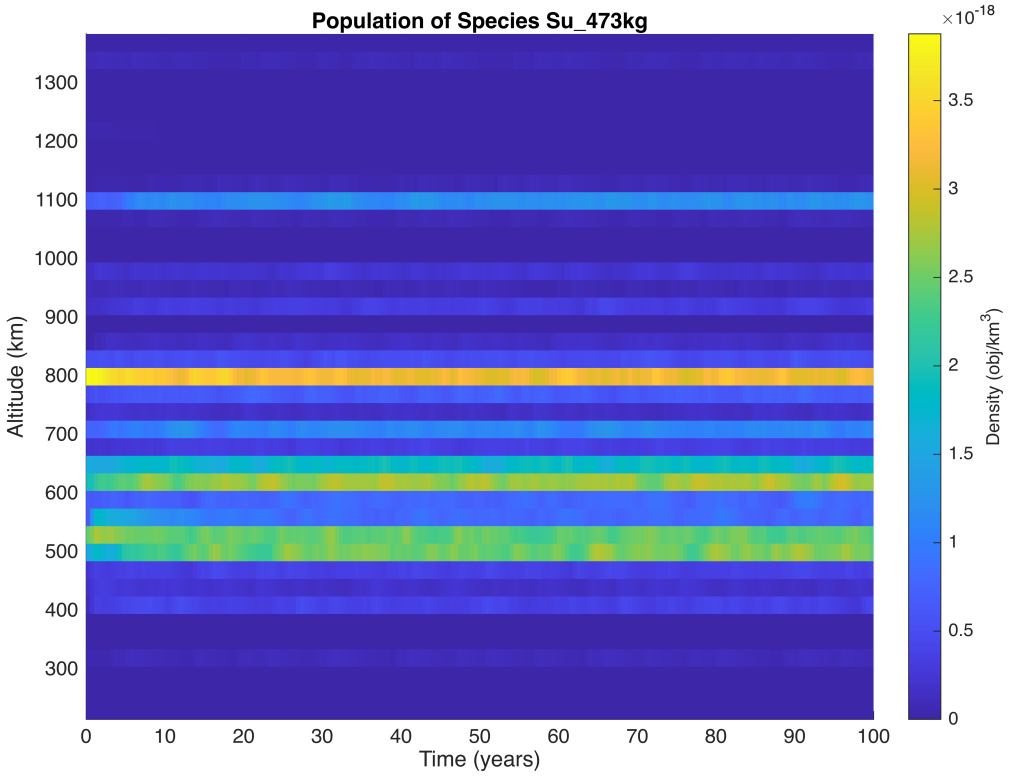
```
Producing visuals for the evolution of each species' density.
```

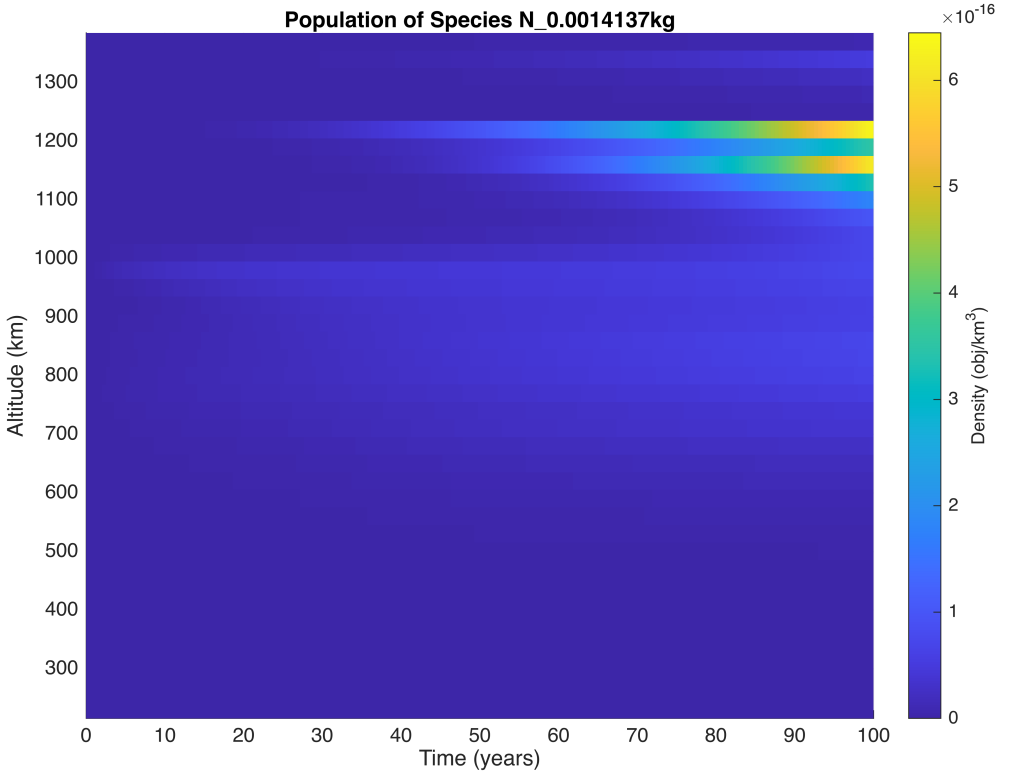

Population of Species S_148kg
Altitude (km)
Time (years)
Density (obj/km^3)
×10^-16

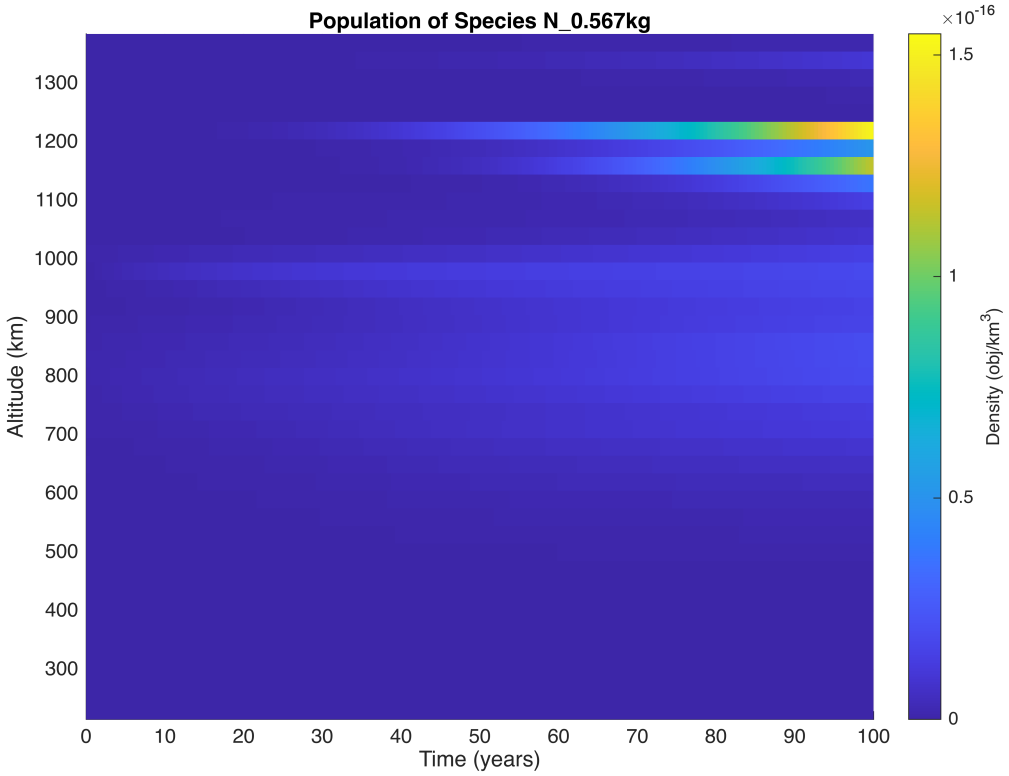

Population of Species S_750kg
Altitude (km)
Time (years)
Density (obj/km^3)
×10^-16

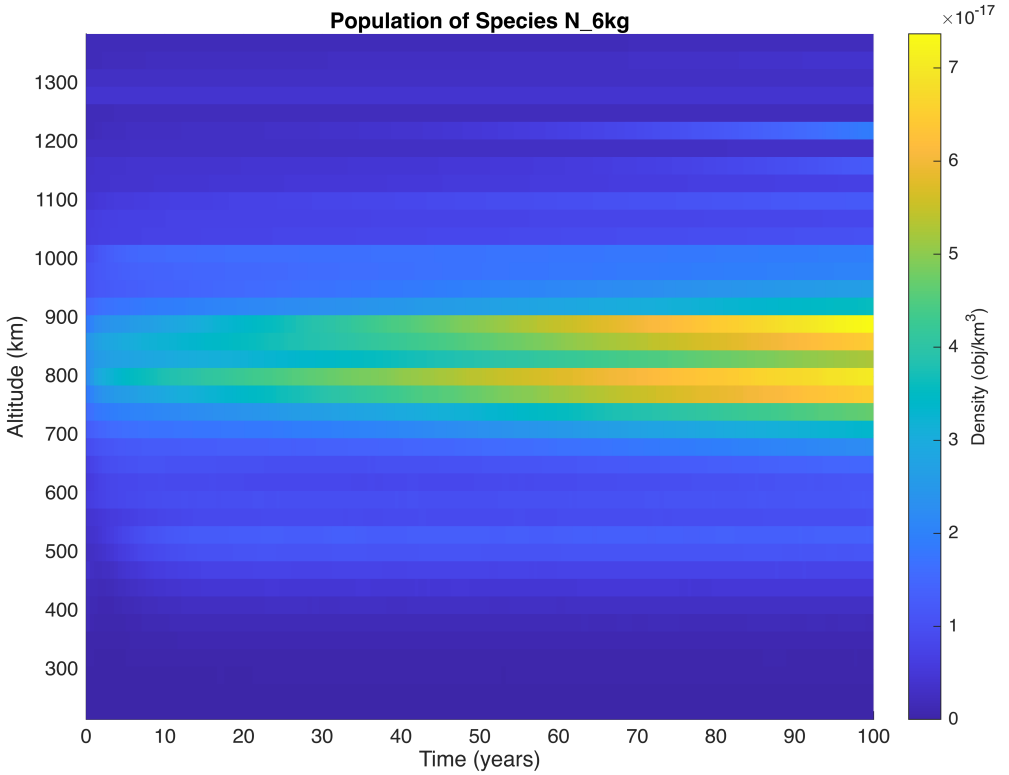

Population of Species S_1250kg
Altitude (km)
Time (years)
Density (obj/km^3)
×10^-16

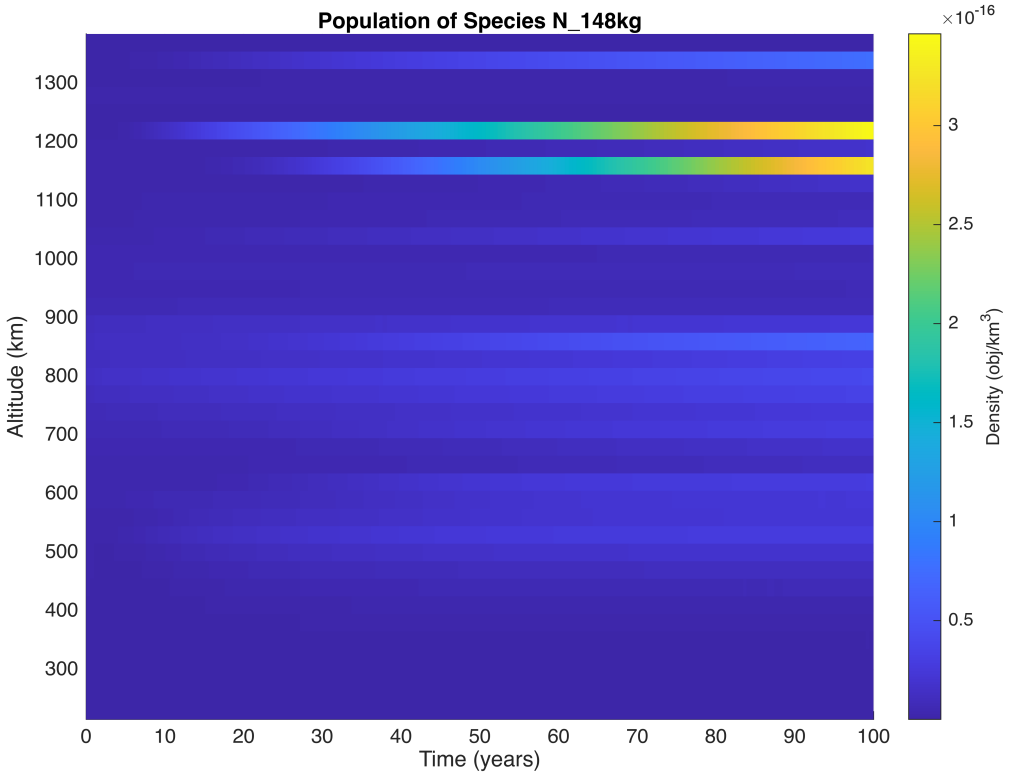

Population of Species Su_260kg
Altitude (km)
Time (years)
Density (obj/km^3)
×10^-16

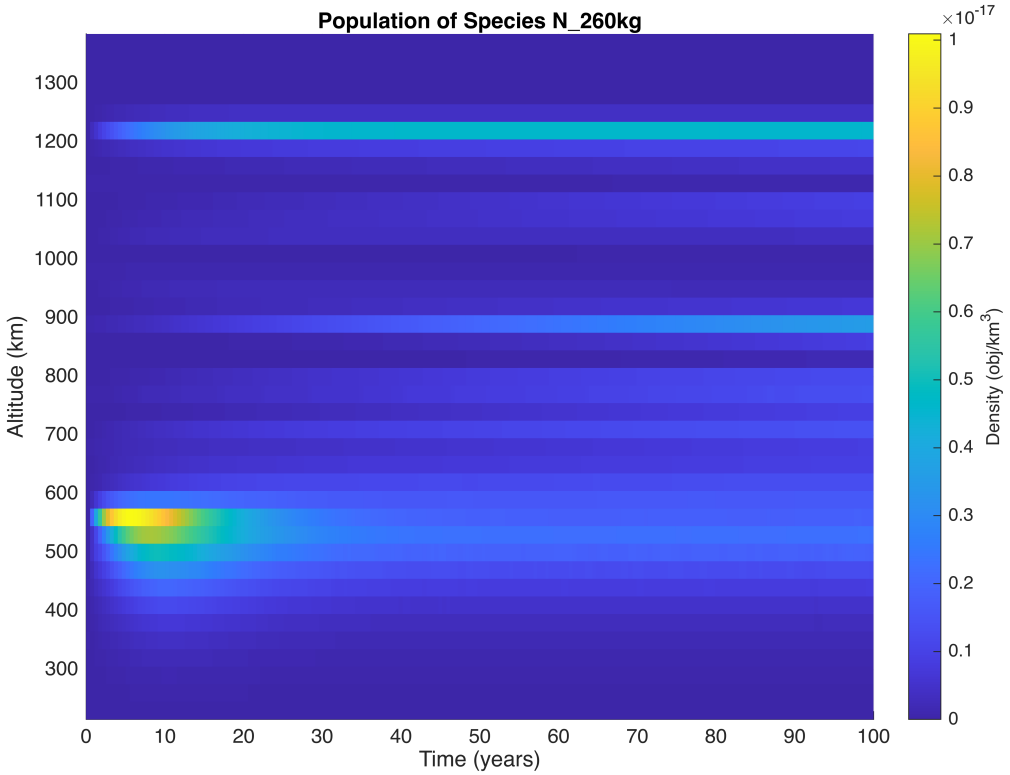

Population of Species Su_473kg
Altitude (km)
Time (years)
Density (obj/km^3)
$\times 10^{-18}$

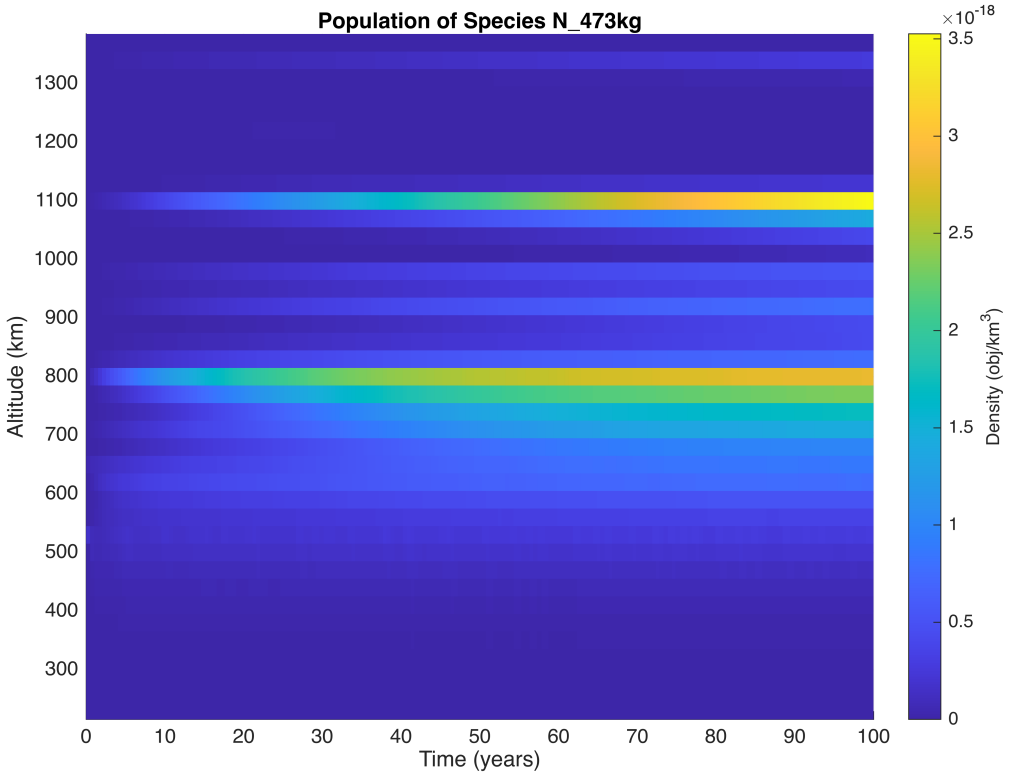

Population of Species N_0.0014137kg
Altitude (km)
Time (years)
Density (obj/km^3)
$\times 10^{-16}$

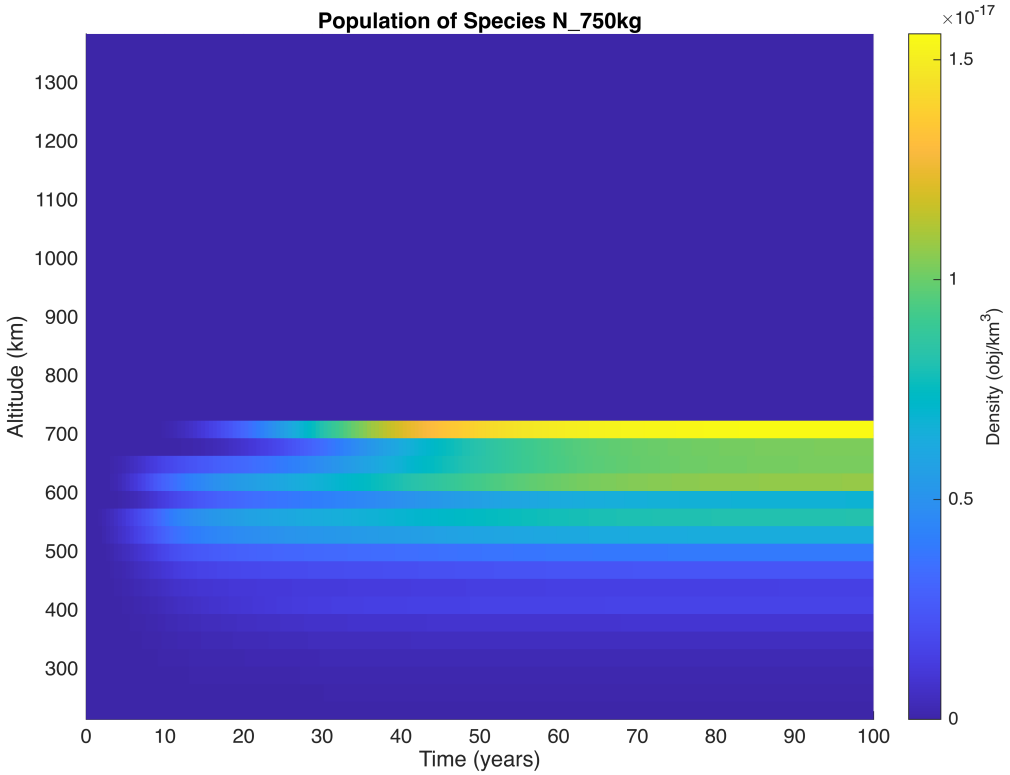

Population of Species N_0.567kg
Altitude (km)
Time (years)
Density (obj/km^3)
×10^-16
1.5
1
0.5
0
300
400
500
600
700
800
900
1000
1100
1200
1300
0
10
20
30
40
50
60
70
80
90
100

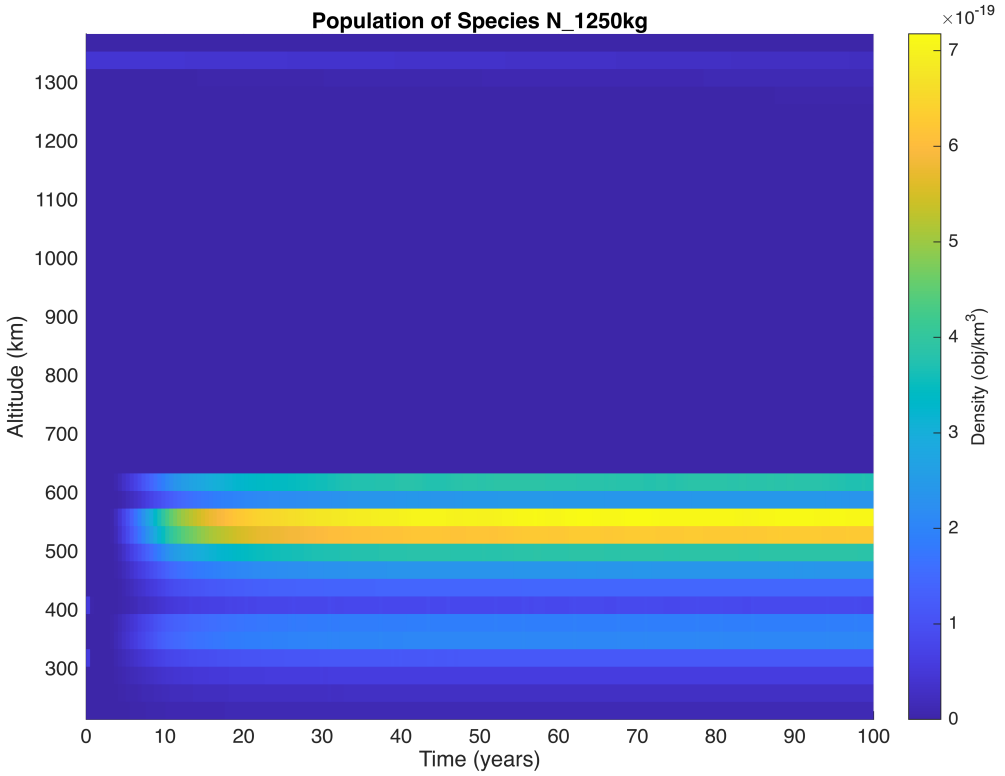

Population of Species N_6kg
Altitude (km)
Time (years)
Density (obj/km^3)
×10^-17
7
6
5
4
3
2
1
0
300
400
500
600
700
800
900
1000
1100
1200
1300
0
10
20
30
40
50
60
70
80
90
100

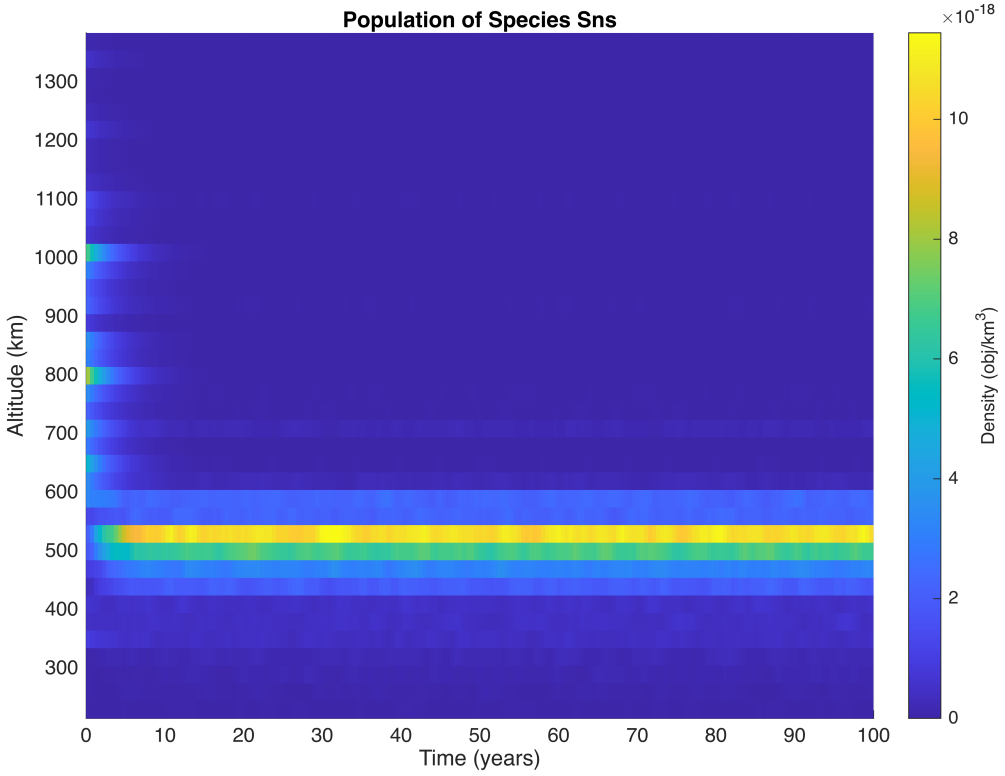

Population of Species N_148kg
×10^-16
Altitude (km)
Time (years)
Density (obj/km^3)
300
400
500
600
700
800
900
1000
1100
1200
1300
0
10
20
30
40
50
60
70
80
90
100
0.5
1
1.5
2
2.5
3

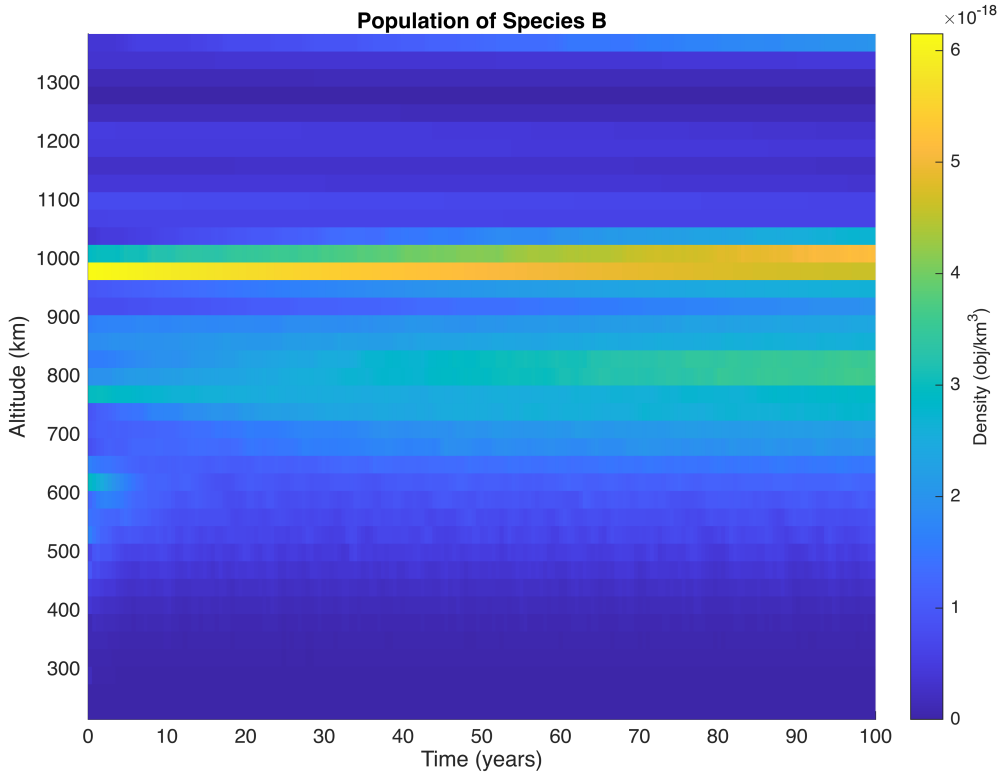

Population of Species N_260kg
×10^-17
Altitude (km)
Time (years)
Density (obj/km^3)
300
400
500
600
700
800
900
1000
1100
1200
1300
0
10
20
30
40
50
60
70
80
90
100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Population of Species N_473kg
Altitude (km)
Time (years)
Density (obj/km^3)
×10^-18
3.5
3
2.5
2
1.5
1
0.5
0
300
400
500
600
700
800
900
1000
1100
1200
1300
0
10
20
30
40
50
60
70
80
90
100


Population of Species N_750kg
Altitude (km)
Time (years)
Density (obj/km^3)
×10^-17
1.5
1
0.5
0
300
400
500
600
700
800
900
1000
1100
1200
1300
0
10
20
30
40
50
60
70
80
90
100


Population of Species N_1250kg
Altitude (km)
Time (years)
Density (obj/km3)
×10-19


Population of Species Sns
Altitude (km)
Time (years)
Density (obj/km3)
×10-18



Produces a Figure for the evolution of the total population of each species over time across all shells

```
% my_sim.pop_vs_time_vis();
```

Graphs the rate of change for the total inactive population added across all altitudes at each time.

percentage = (bool, defaults to false) that expresses rate of change in percentage terms.

```
% my_sim.total_deb_pop_time_deriv('percentage', true)
```

Produces a Figure with subplots for numerical derivative of each inactive species population evolution over time constraint is the value used in the color scale.

percentage is bool (default false) for whether to use percentage or absolute.

color_scale_crop_percentiles is pair (default [0, 100]). Color scale will be set based on the inclusive range from the lower value to the higher value.

```
% my_sim.total_deb_species_deriv_evolv_vis();
```

Produces a Figure for the evolution of the total population of all objects over time across all shells

```
% my_sim.tot_vs_time_vis();
```

Produces a Figure per species for the evolution of the total population of each species over time across all shells

```
% my_sim.species_time_vis();
```

Displays total spatial density in each shell over time

```
% my_sim.density_vs_time();
```

Computes and displays Number_Time_Product

```
% my_sim.num_time_prod(x0);
```

Produces a Figure comparing the total initial number of objects and the total final number of objects

```
% my_sim.init_vs_final_vis();
```

Produces a Figure comparing the initial population of each species alongside a plot of the final population of each species

```
% my_sim.init_and_final_vis();
```

Produces a Figure showing number of debris across altitudes by debris type

```
% my_sim.final_debris_vis();
```

Computes and displays cumulative Criticality of Space Index - CSI

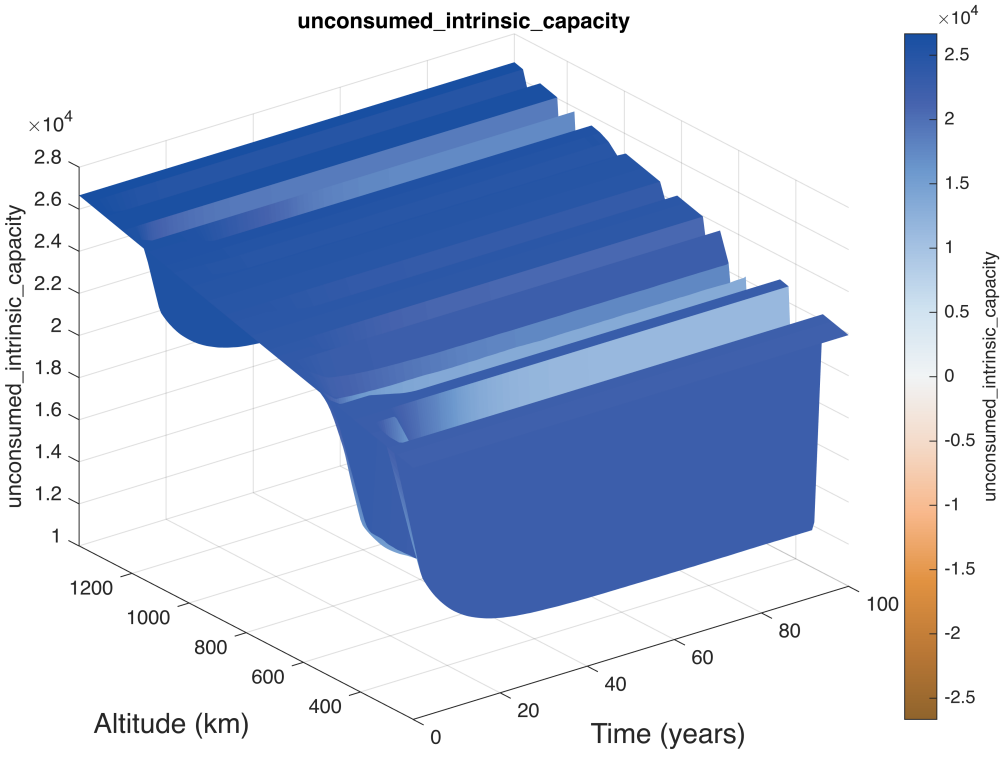
```
% my_sim.cum_CSI()
```

Indicator variables visualization

Intrinsic Capacity

```
my_sim.per_shell_indicator("unconsumed_intrinsic_capacity", "constraint",
0.0, 'constraint_type', "lower")
```

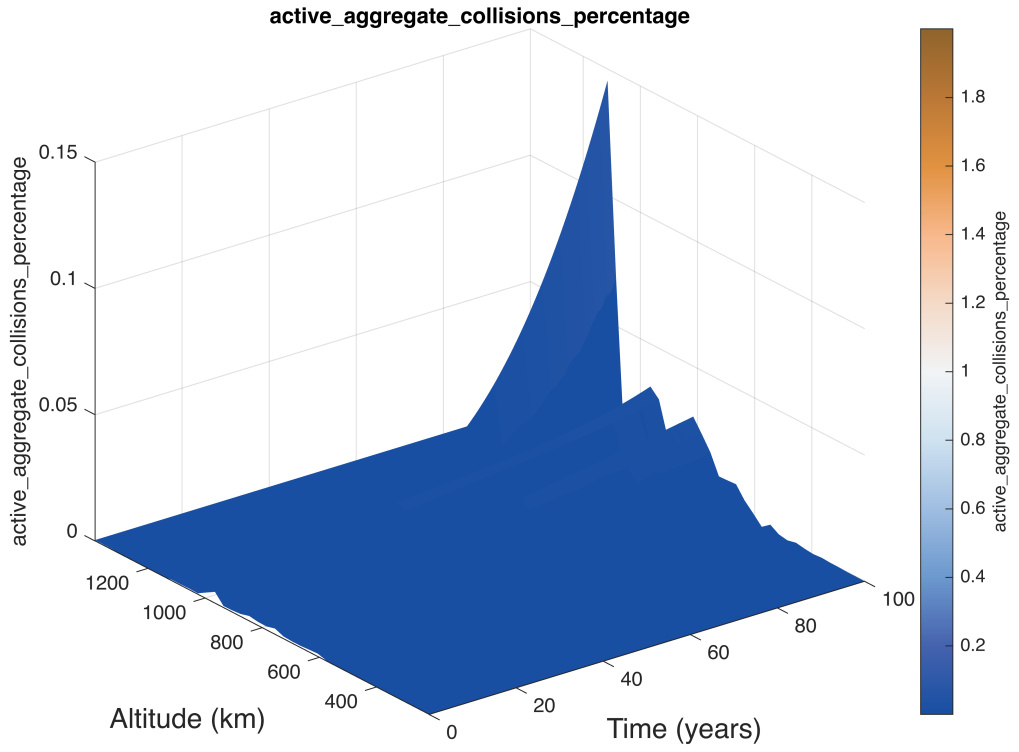
```
Producing visuals for the evolution of indicator.
```



Operational CA

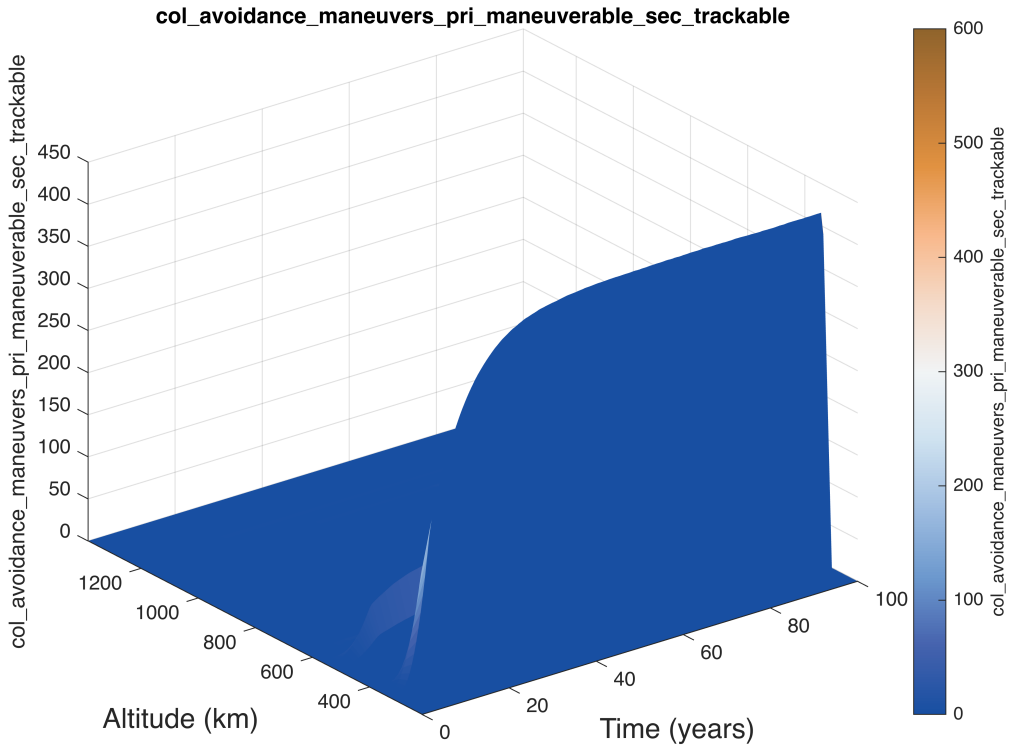
```
my_sim.per_shell_indicator("active_aggregate_collisions_percentage",
"constraint", 1.0)
```

```
Producing visuals for the evolution of indicator.
```



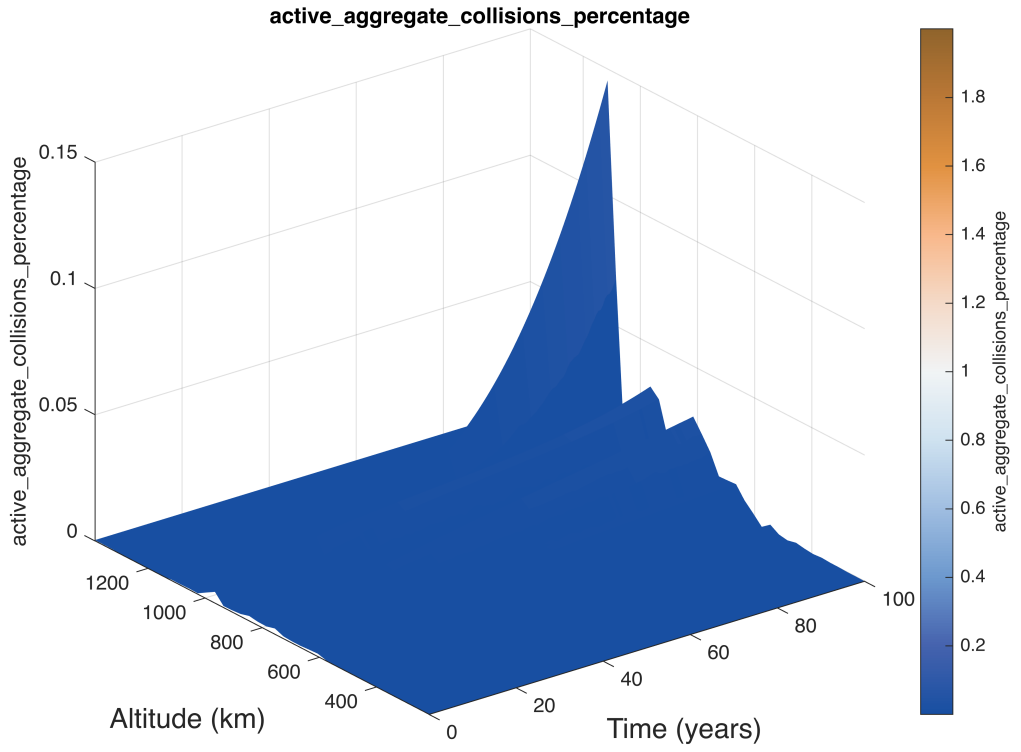

```
my_sim.per_shell_indicator("col_avoidance_maneuvers_pri_maneuverable_sec_tra
ckable", "constraint", 300)
```

```
Producing visuals for the evolution of indicator.
```

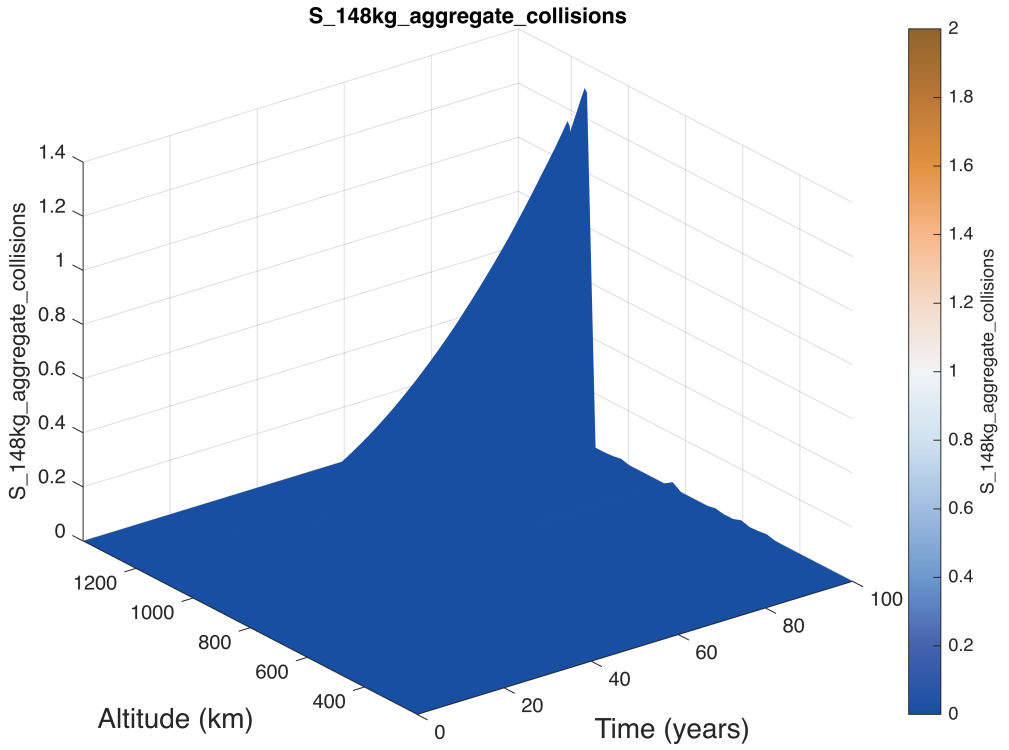



```
for ii = 1:length(scen_properties.indicator_var_list)
    if contains(scen_properties.indicator_var_list(ii).name,
"aggregate_collisions")
        disp(scen_properties.indicator_var_list(ii).name)

my_sim.per_shell_indicator(scen_properties.indicator_var_list(ii).name,
"constraint", 1.0, 'constraint_type', "upper")
    end
end
```

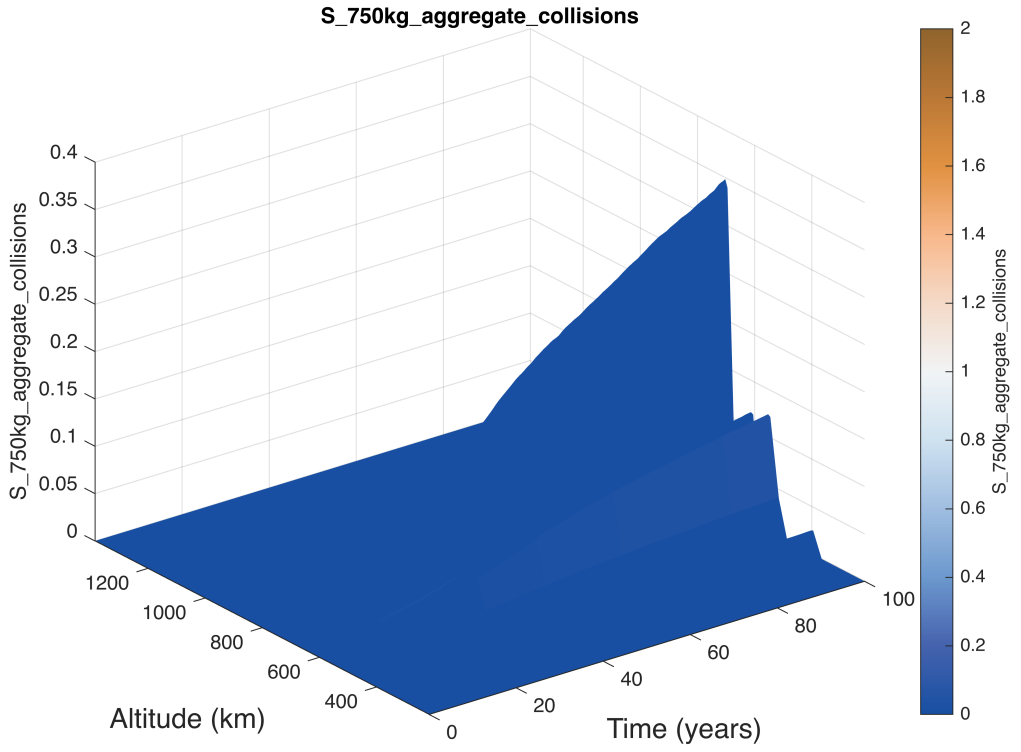
```
active_aggregate_collisions_percentage
Producing visuals for the evolution of indicator.
```



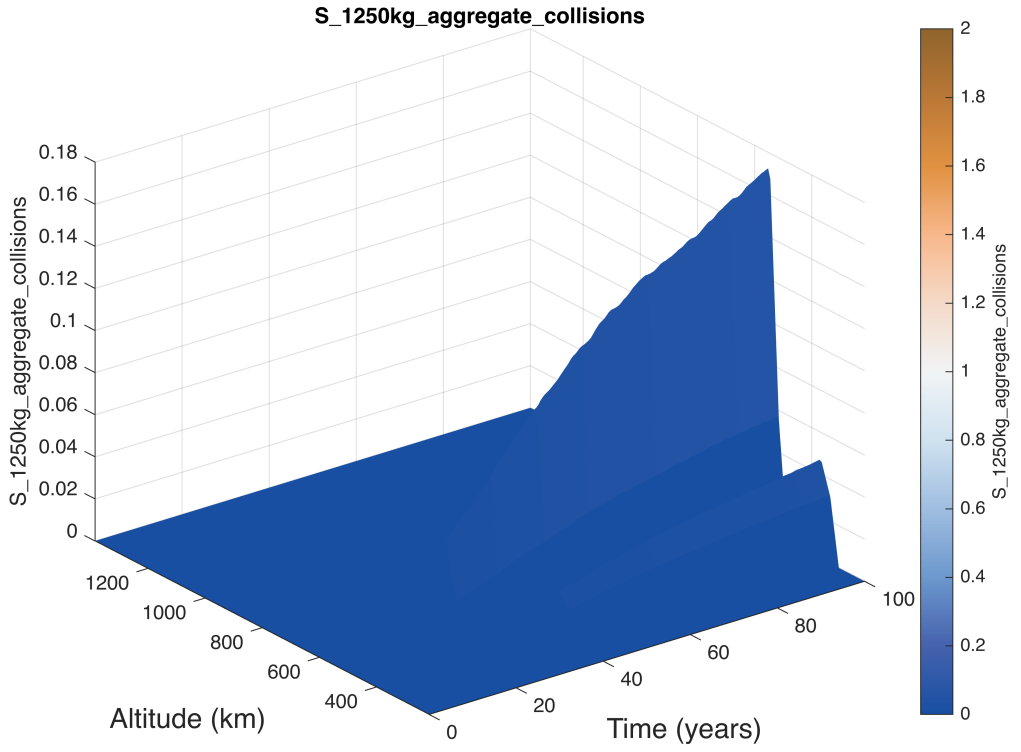

```
S_148kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



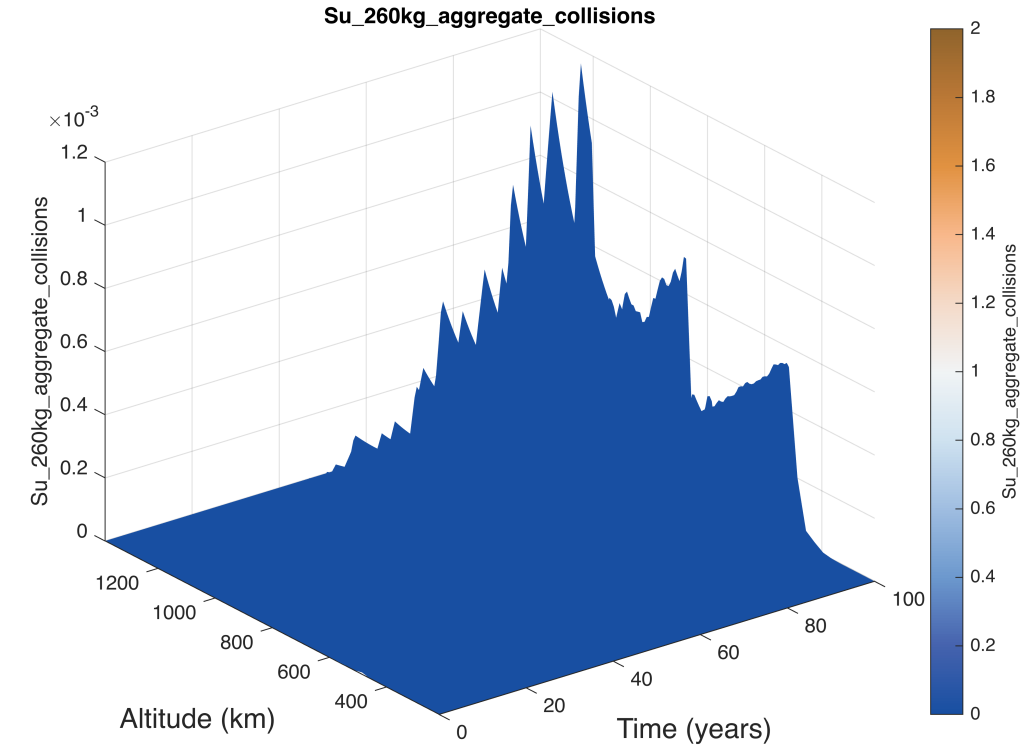

```
S_750kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



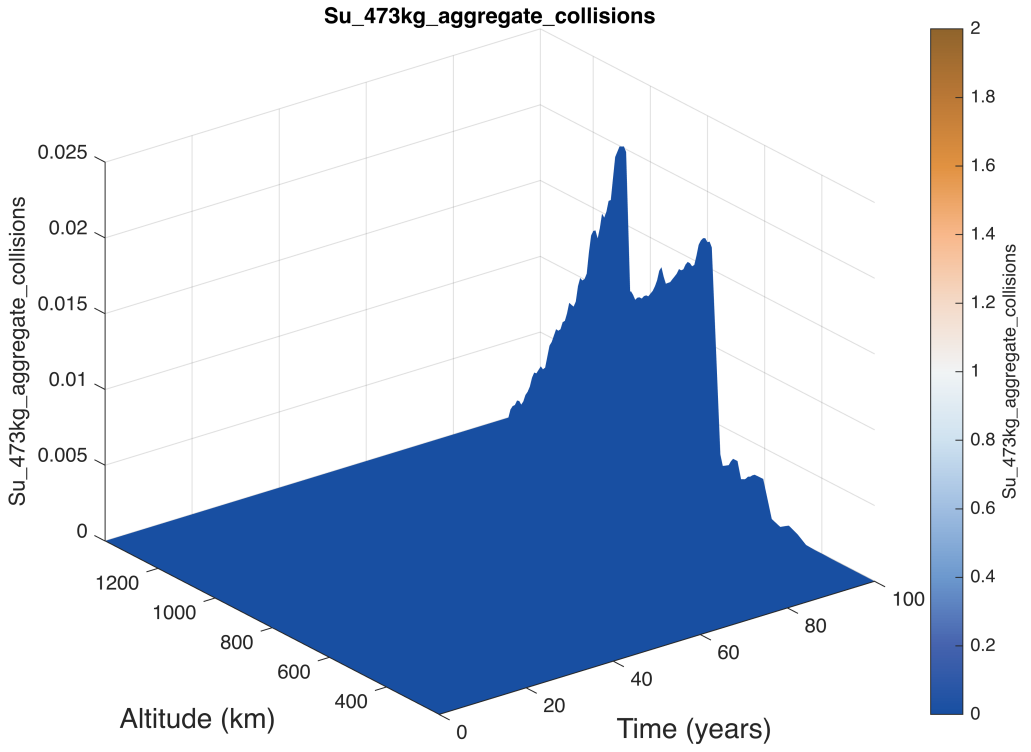

```
S_1250kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



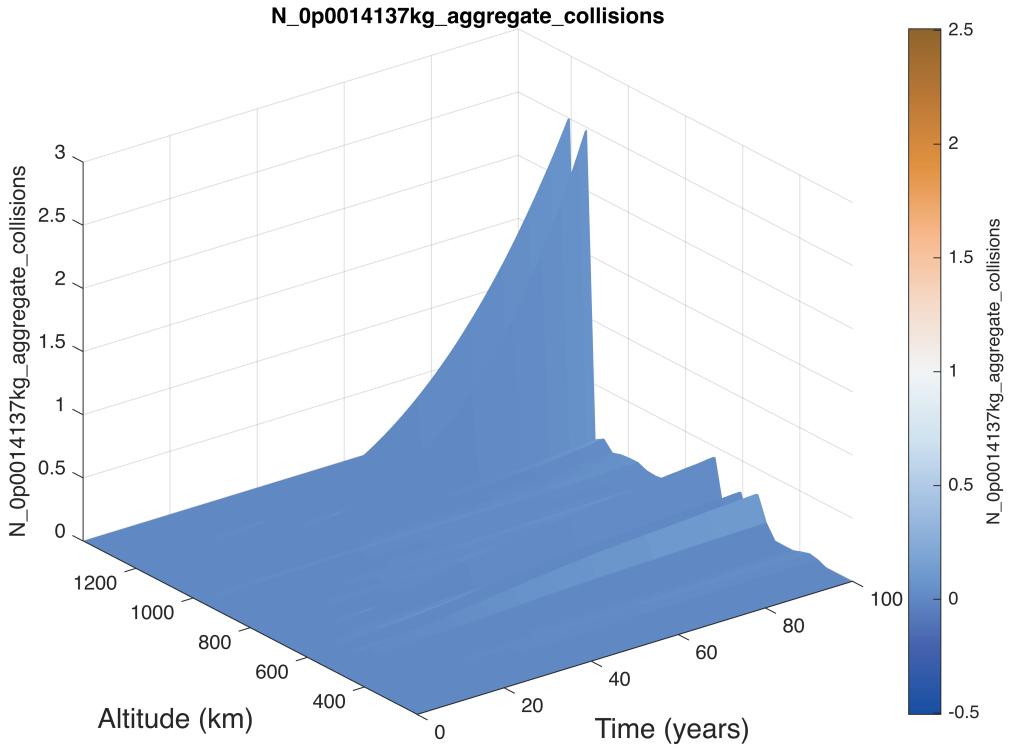

```
Su_260kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



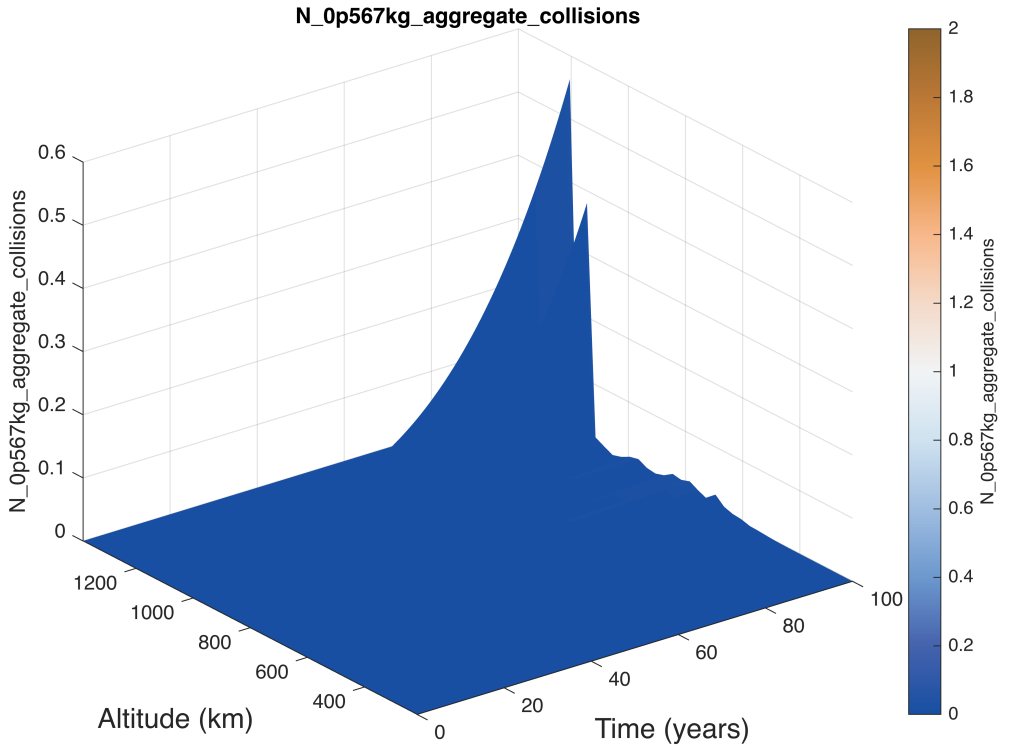

```
Su_473kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



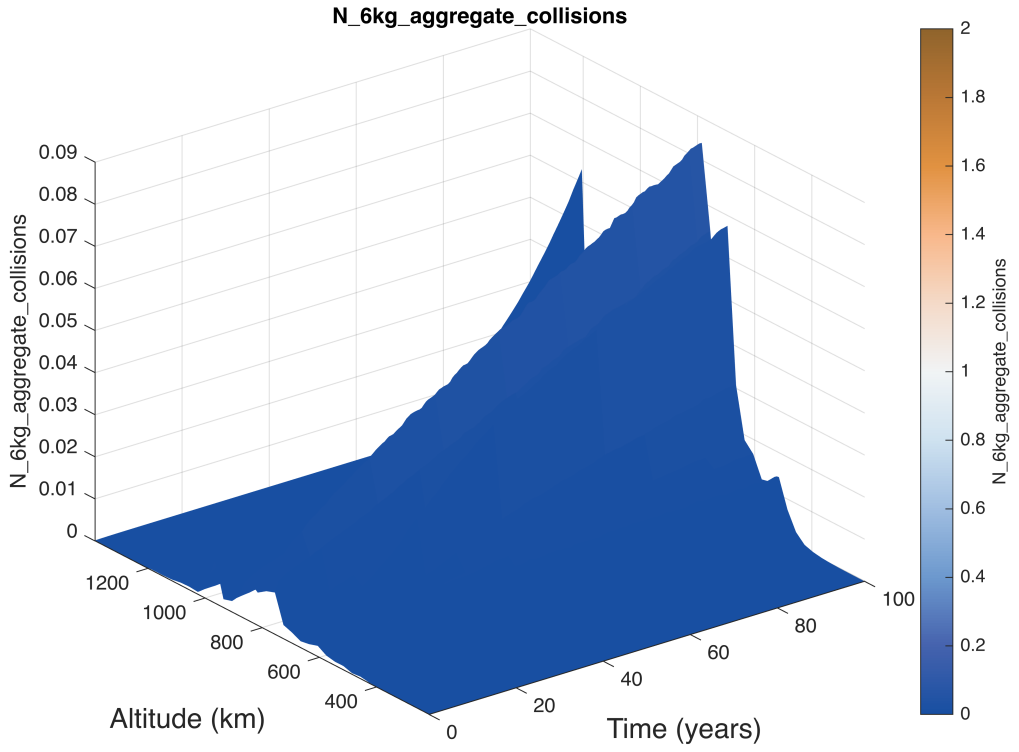

```
N_0p0014137kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



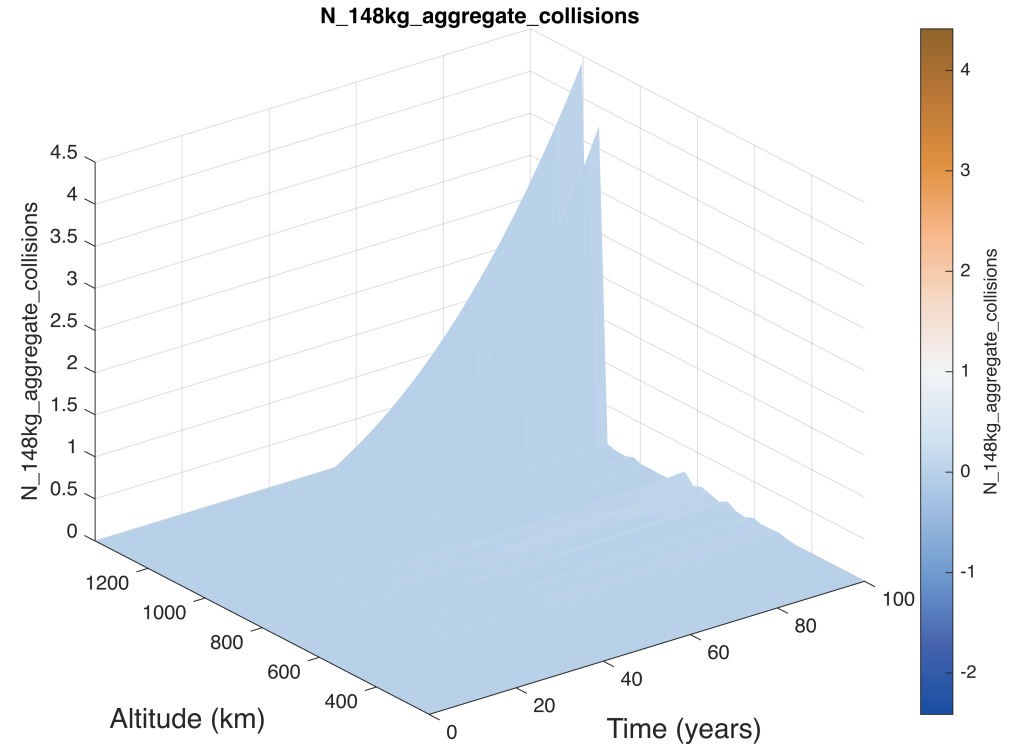
N_0p567kg_aggregate_collisions
Producing visuals for the evolution of indicator.



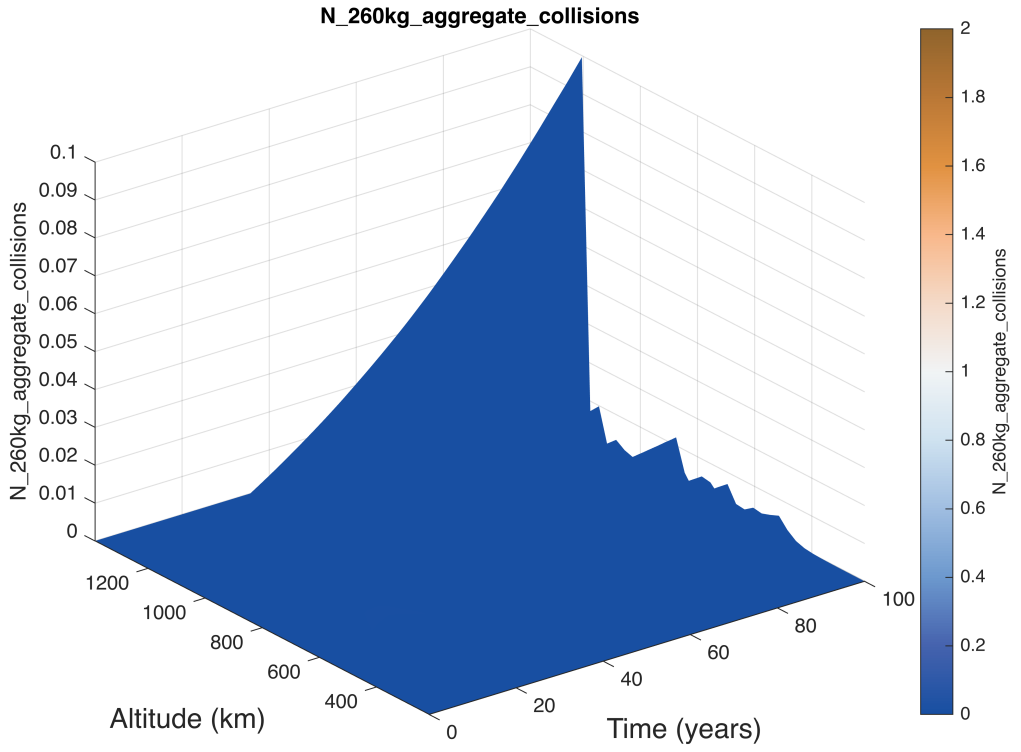
N_6kg_aggregate_collisions
Producing visuals for the evolution of indicator.



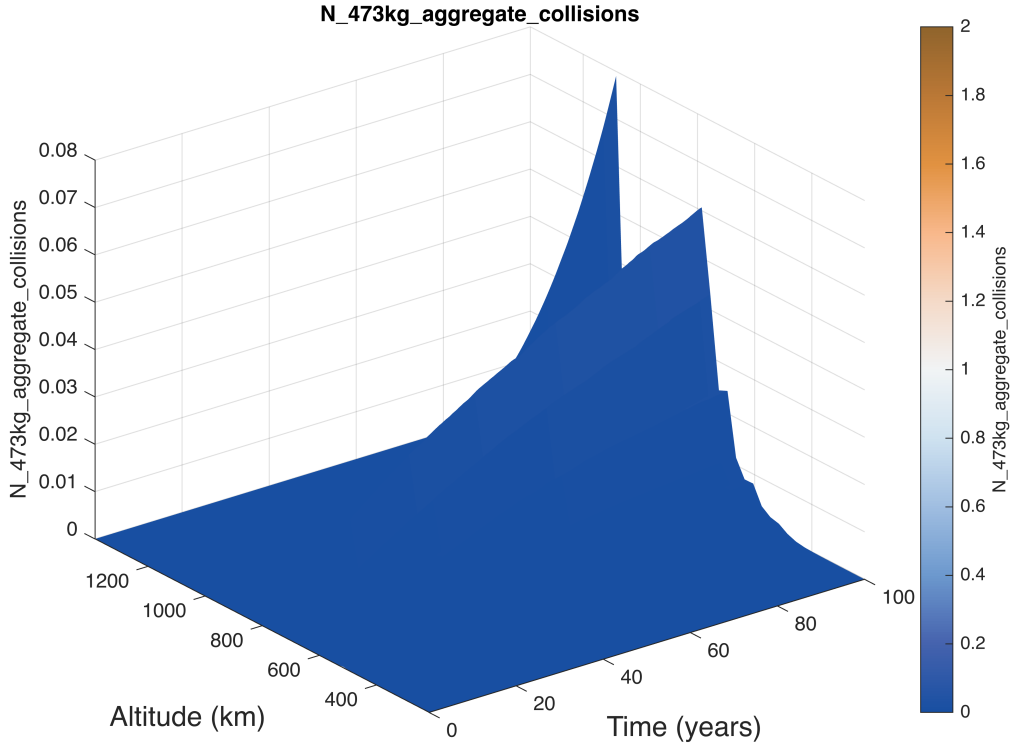

```
N_148kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



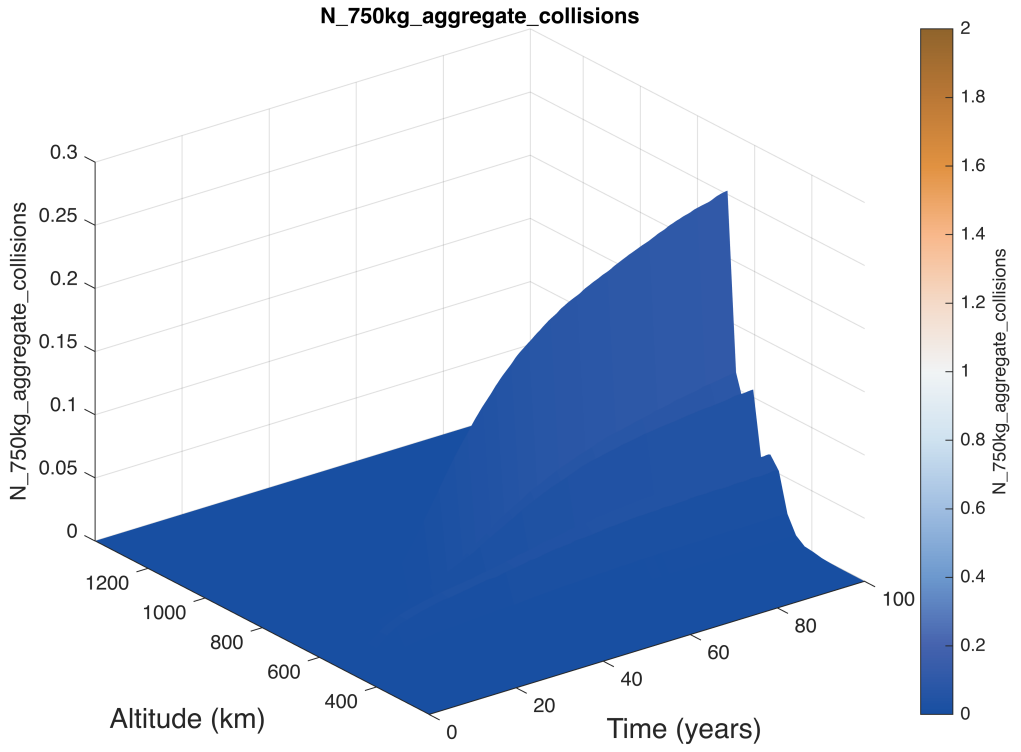

```
N_260kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



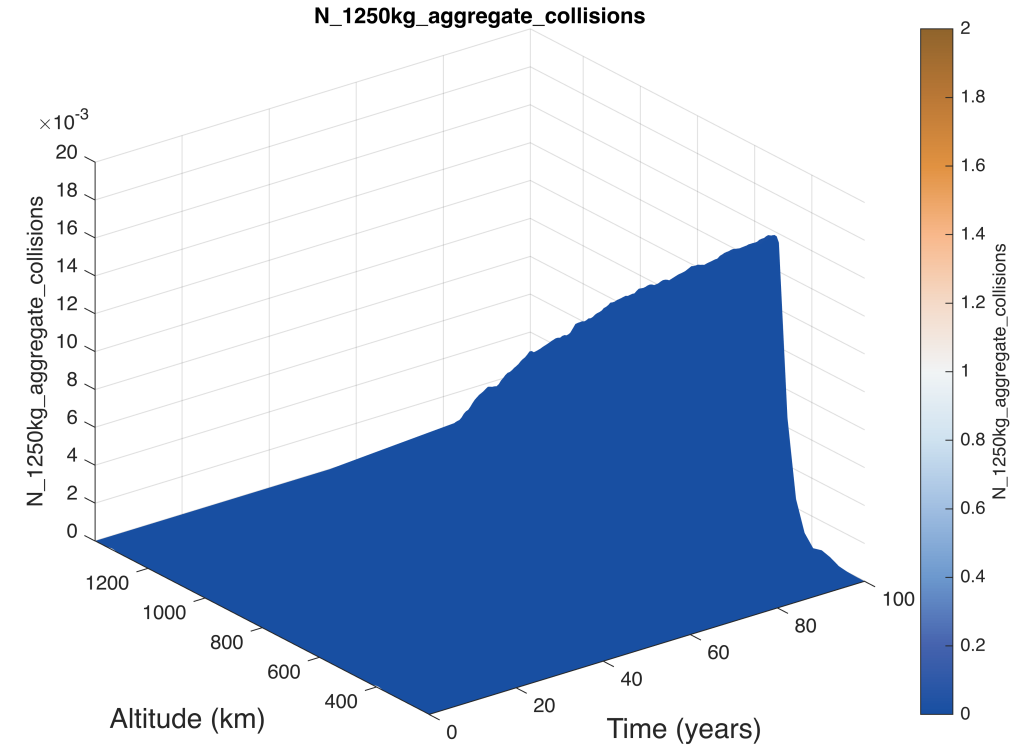
N_473kg_aggregate_collisions
Producing visuals for the evolution of indicator.



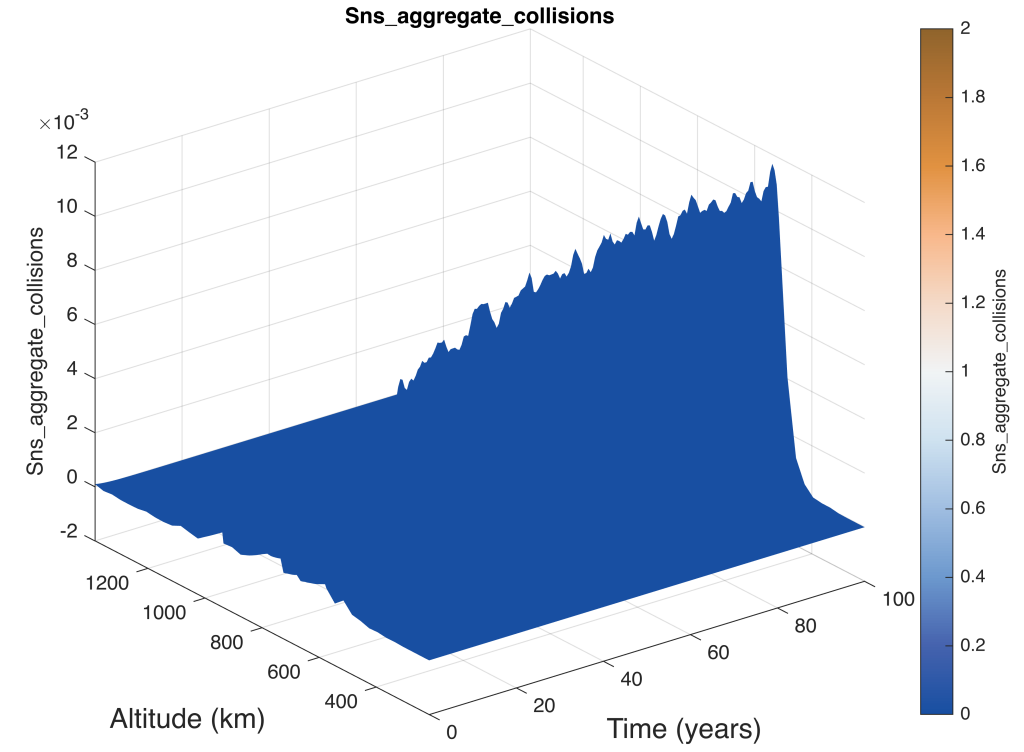
N_750kg_aggregate_collisions
Producing visuals for the evolution of indicator.



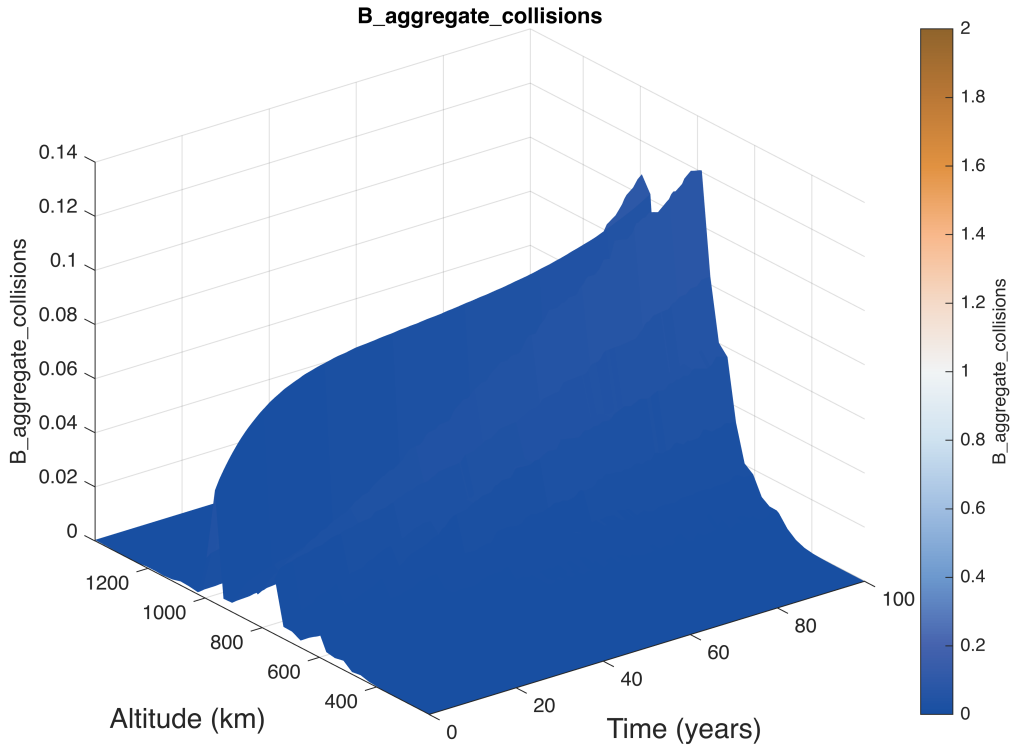
N_1250kg_aggregate_collisions
Producing visuals for the evolution of indicator.



Sns_aggregate_collisions
Producing visuals for the evolution of indicator.

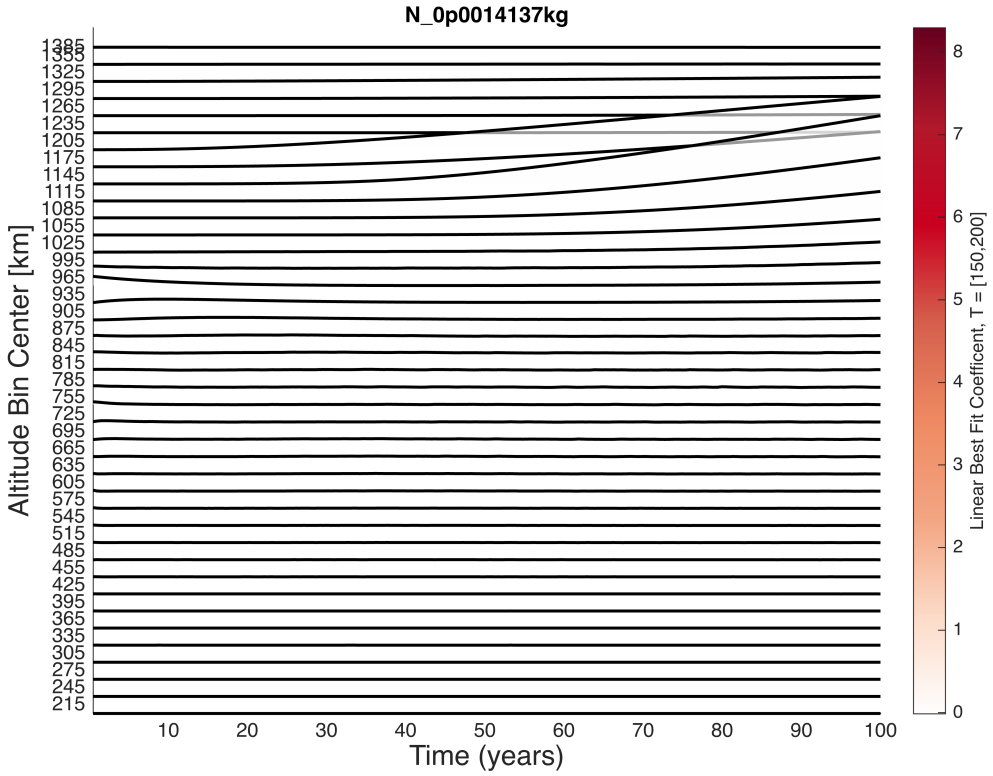


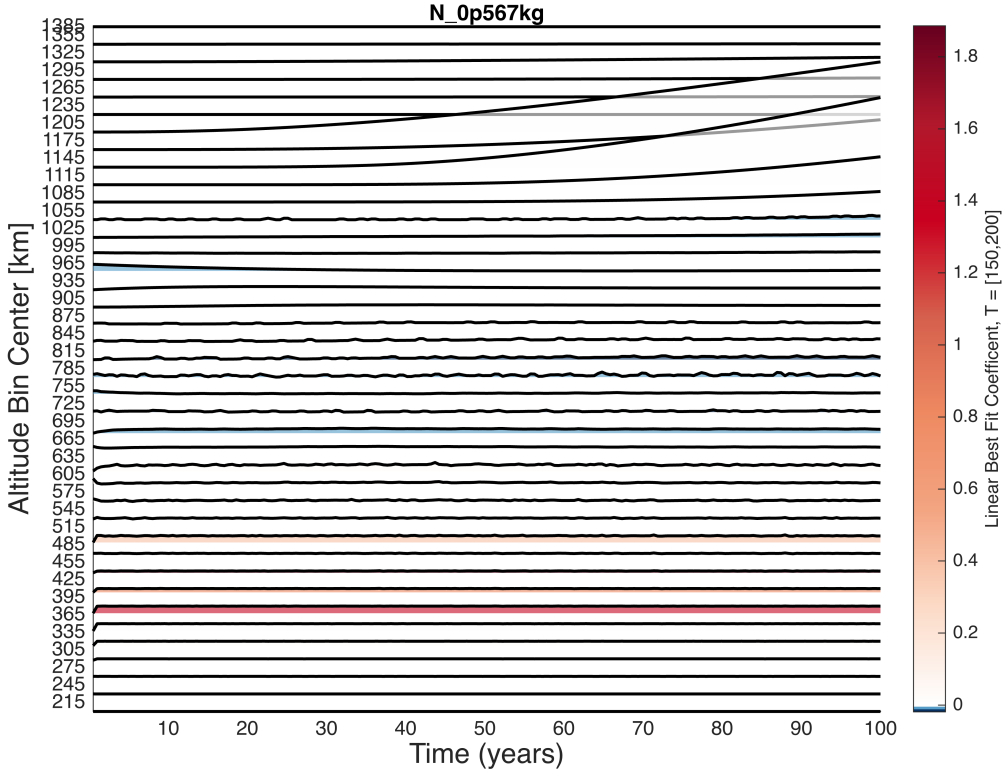

```
B_aggregate_collisions
Producing visuals for the evolution of indicator.
```

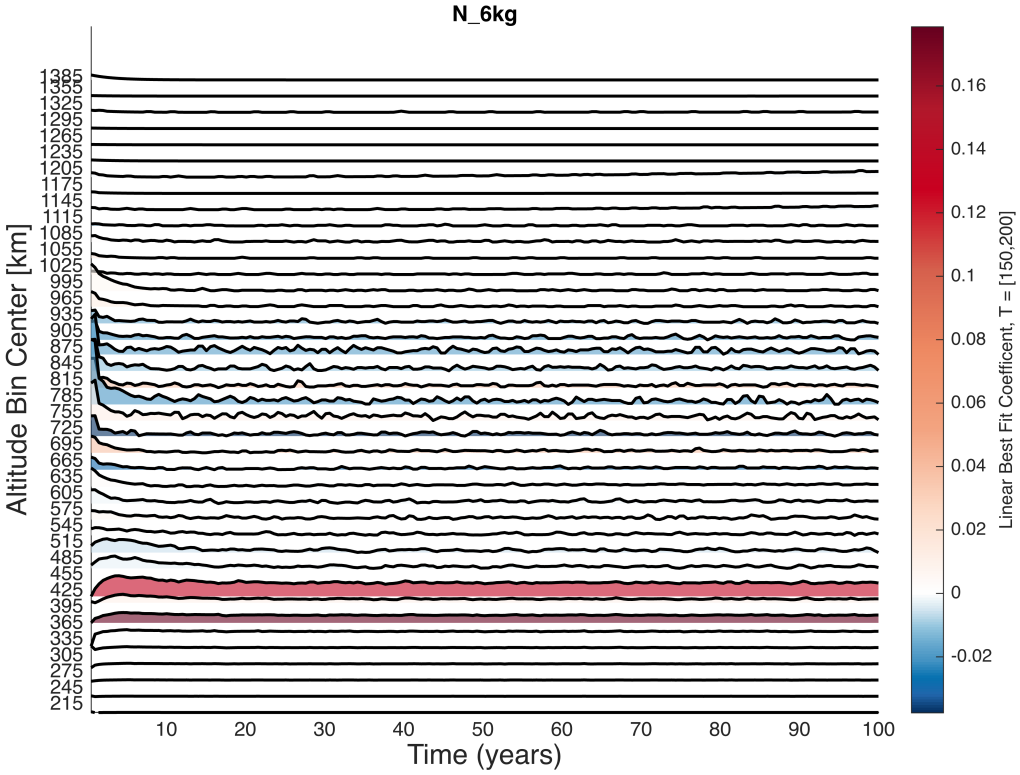


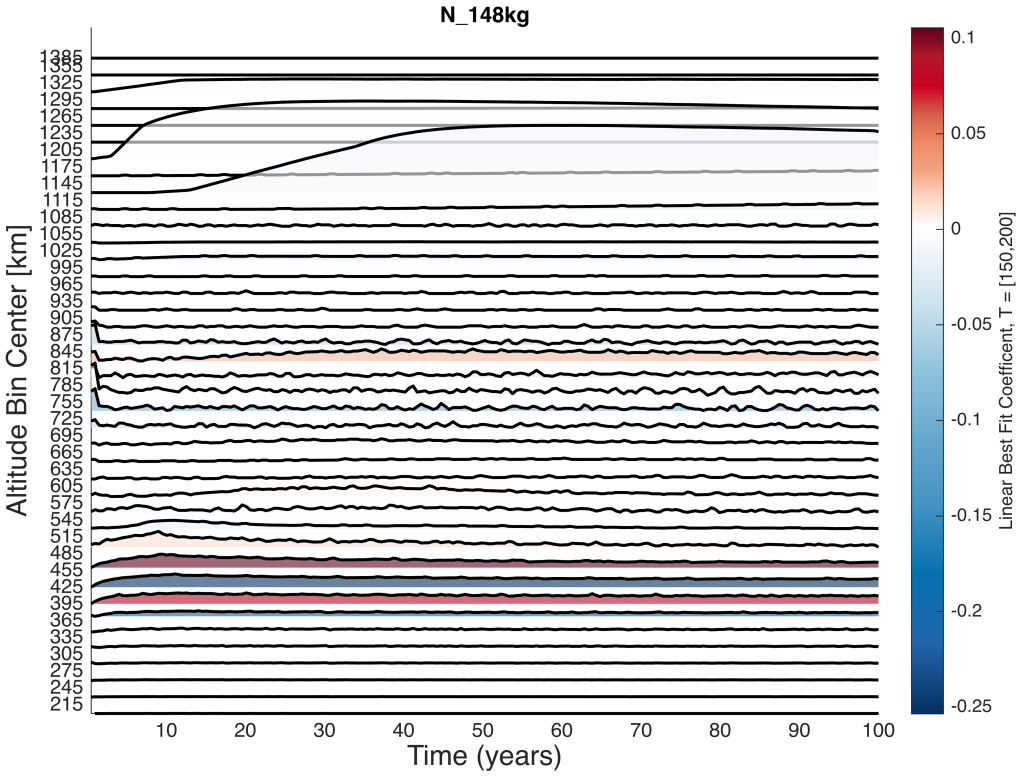
Long-Term Sustainability

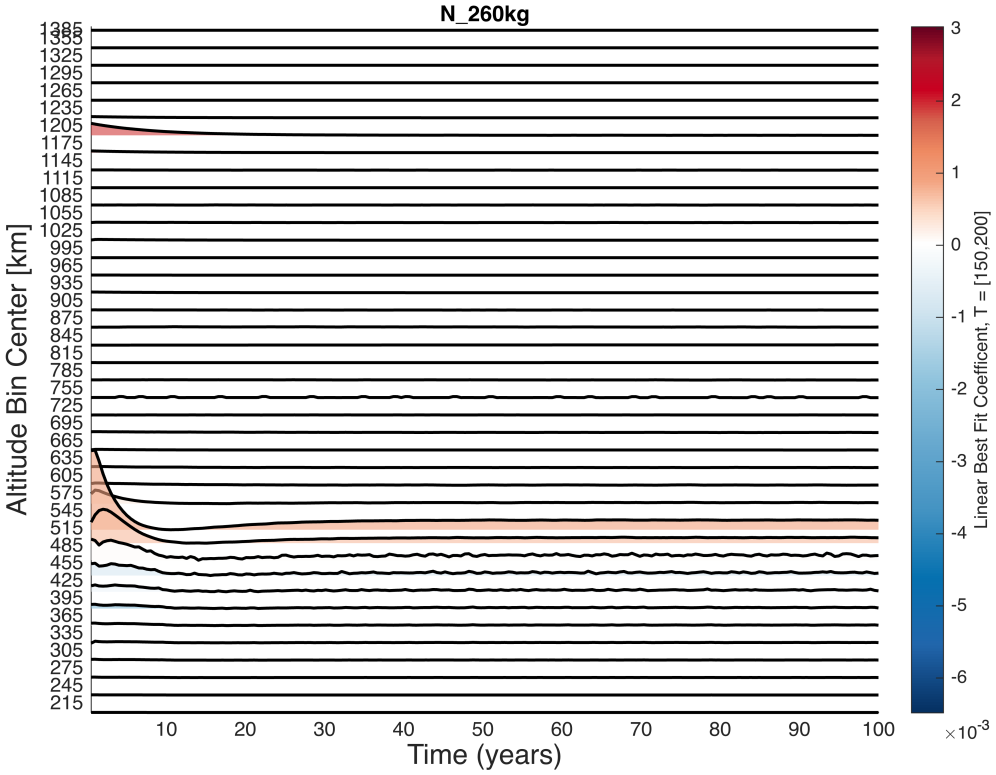
```
derivative_joy_plot(my_sim, N_species)
```

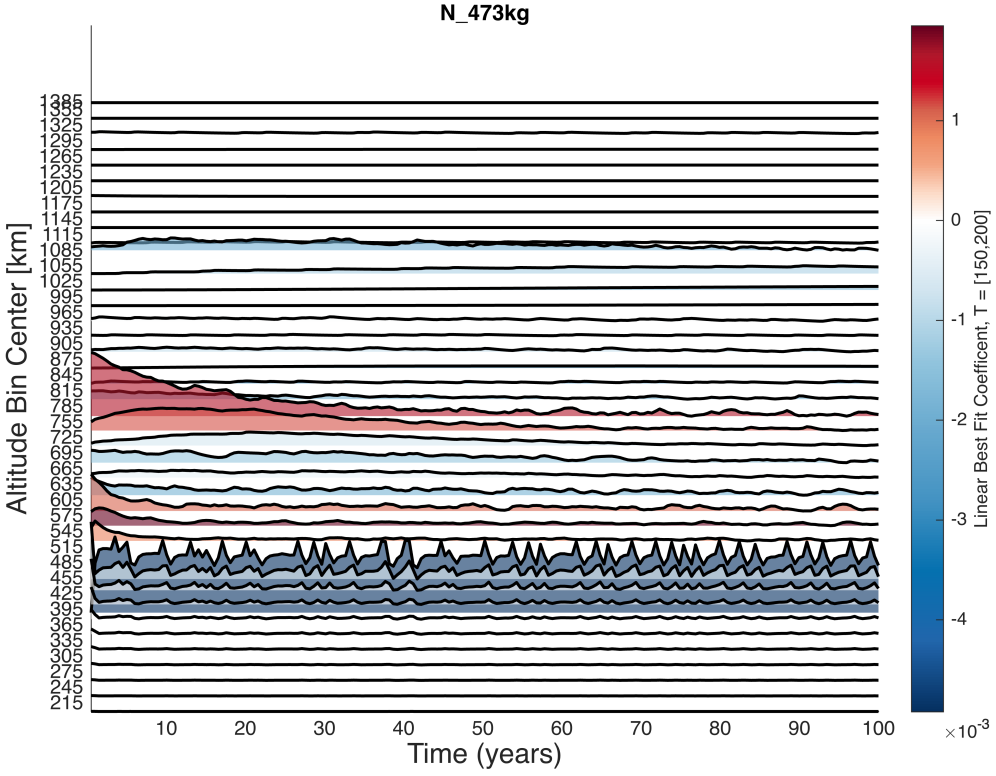


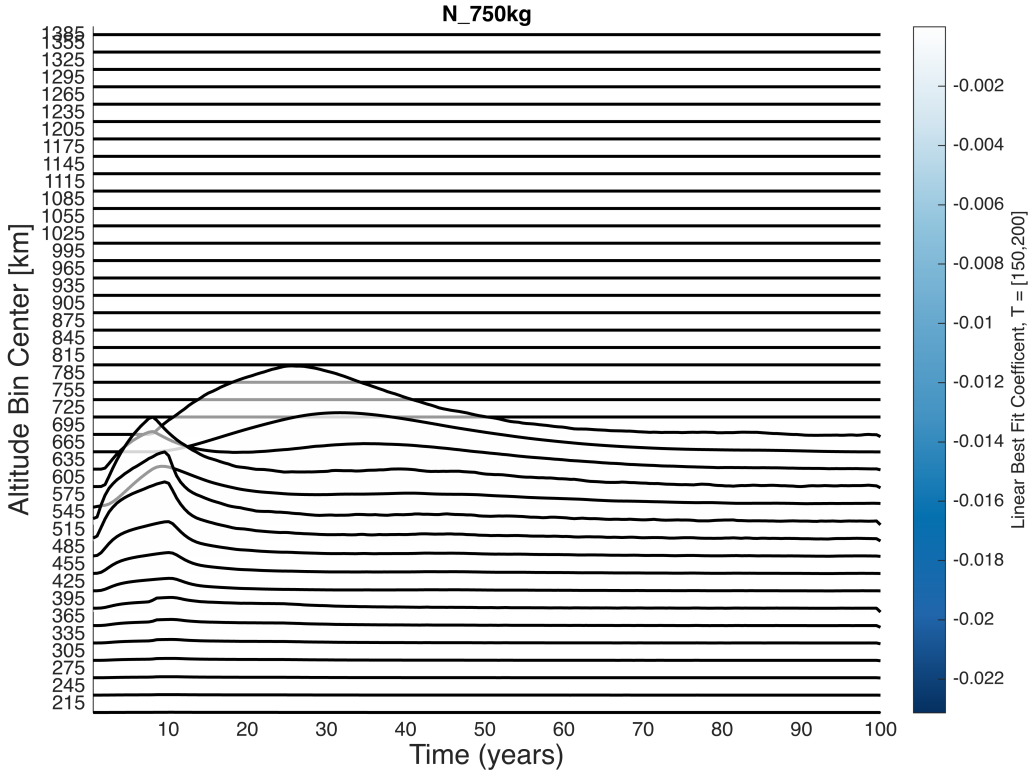


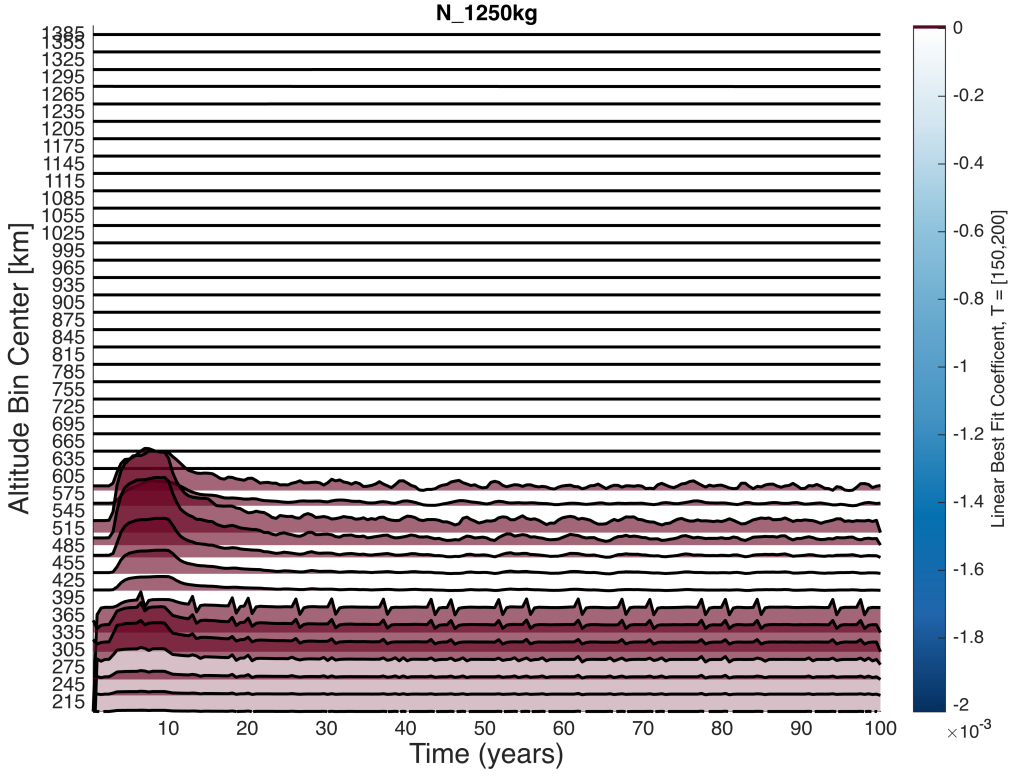

N_6kg
Altitude Bin Center [km]
1385
1355
1325
1295
1265
1235
1205
1175
1145
1115
1085
1055
1025
995
965
935
905
875
845
815
785
755
725
695
665
635
605
575
545
515
485
455
425
395
365
335
305
275
245
215
10
20
30
40
50
60
70
80
90
100
Time (years)
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
Linear Best Fit Coefficent, T = [150,200]


N_148kg
Altitude Bin Center [km]
1385
1355
1325
1295
1265
1235
1205
1175
1145
1115
1085
1055
1025
995
965
935
905
875
845
815
785
755
725
695
665
635
605
575
545
515
485
455
425
395
365
335
305
275
245
215
10
20
30
40
50
60
70
80
90
100
Time (years)
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
Linear Best Fit Coefficent, T = [150,200]


N_260kg
Altitude Bin Center [km]
Time (years)
Linear Best Fit Coefficent, T = [150,200]
×10^-3


N_473kg
Altitude Bin Center [km]
Time (years)
Linear Best Fit Coefficent, T = [150,200]
×10^-3

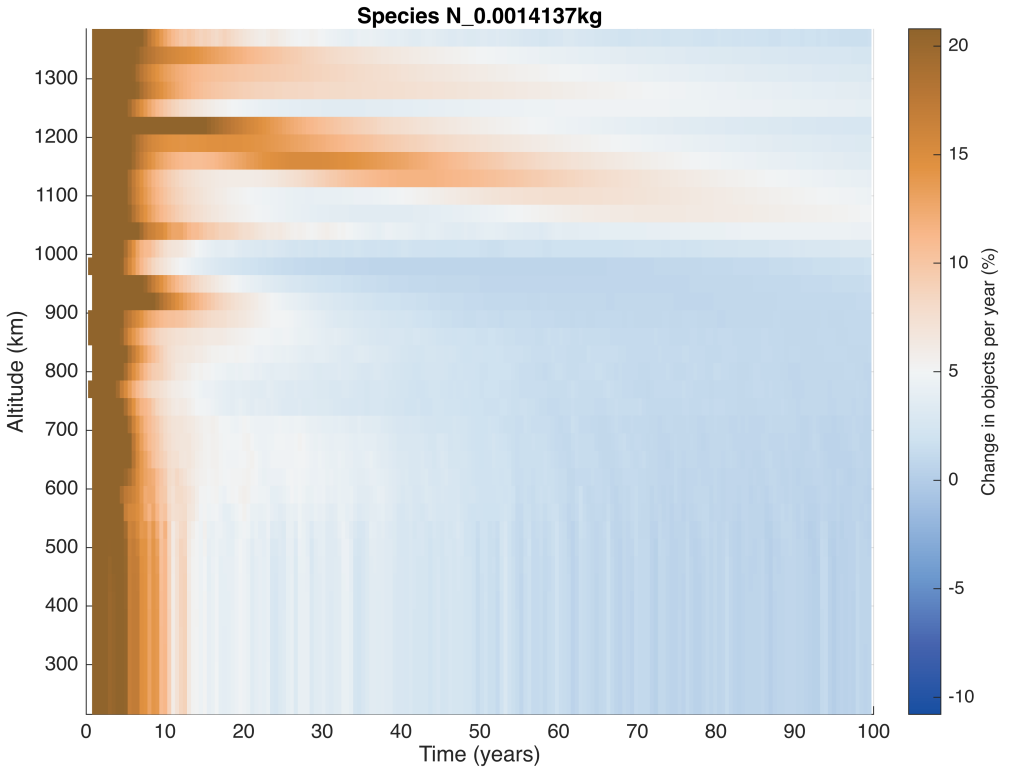


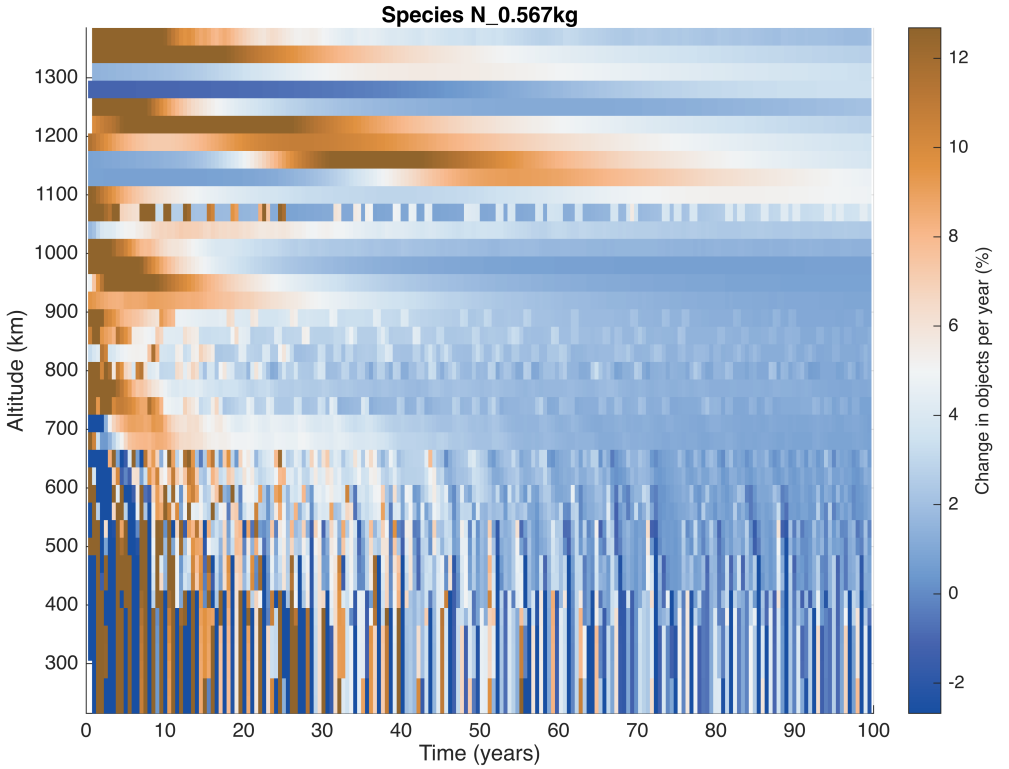


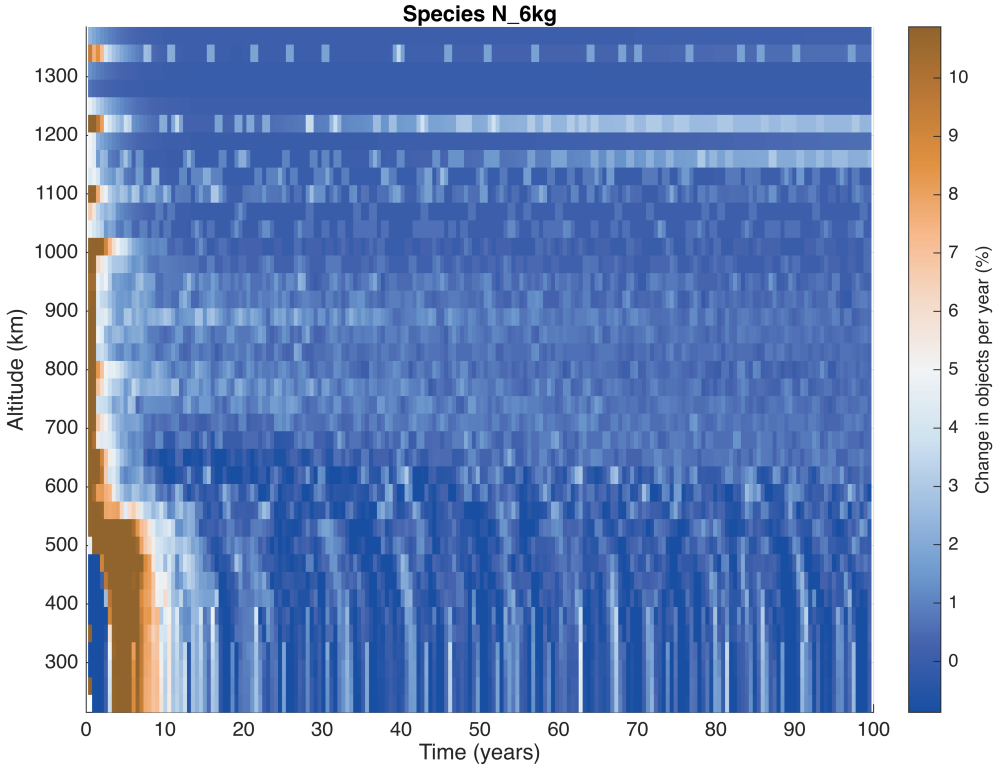

```
ans = 1×13
     0     0     0     0     0    50    51    52    53    54    55    56    57
```

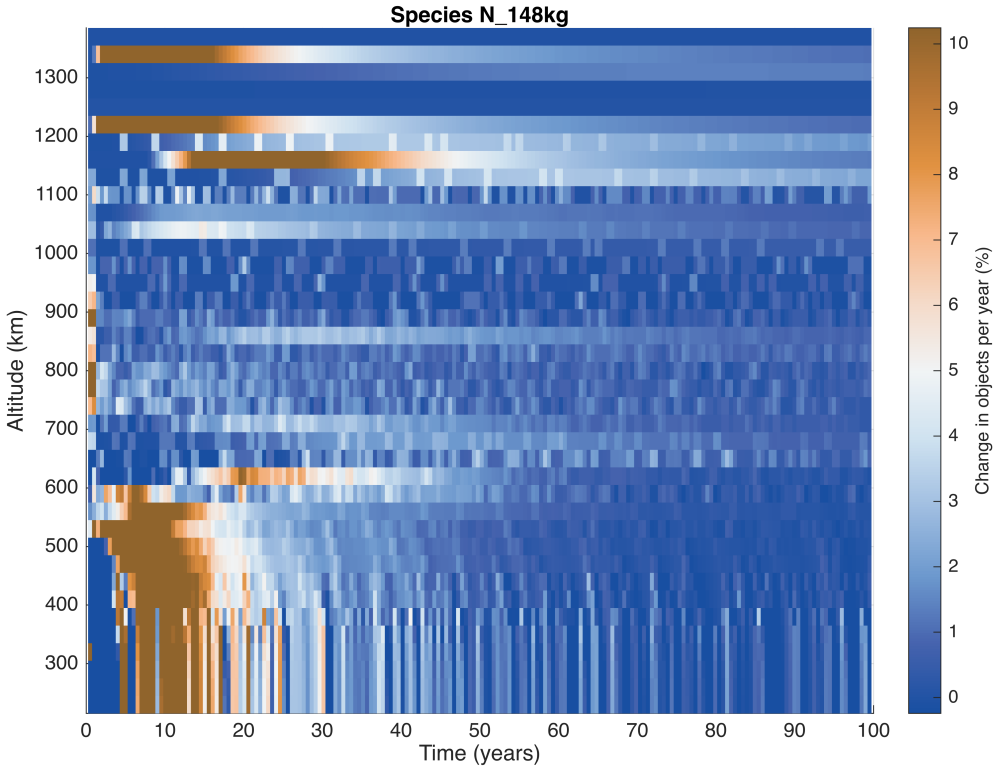
```
my_sim.total_deb_species_deriv_evolv_vis("percentage", true,
"color_scale_crop_percentiles", [5, 95], "constraint", 5)
```

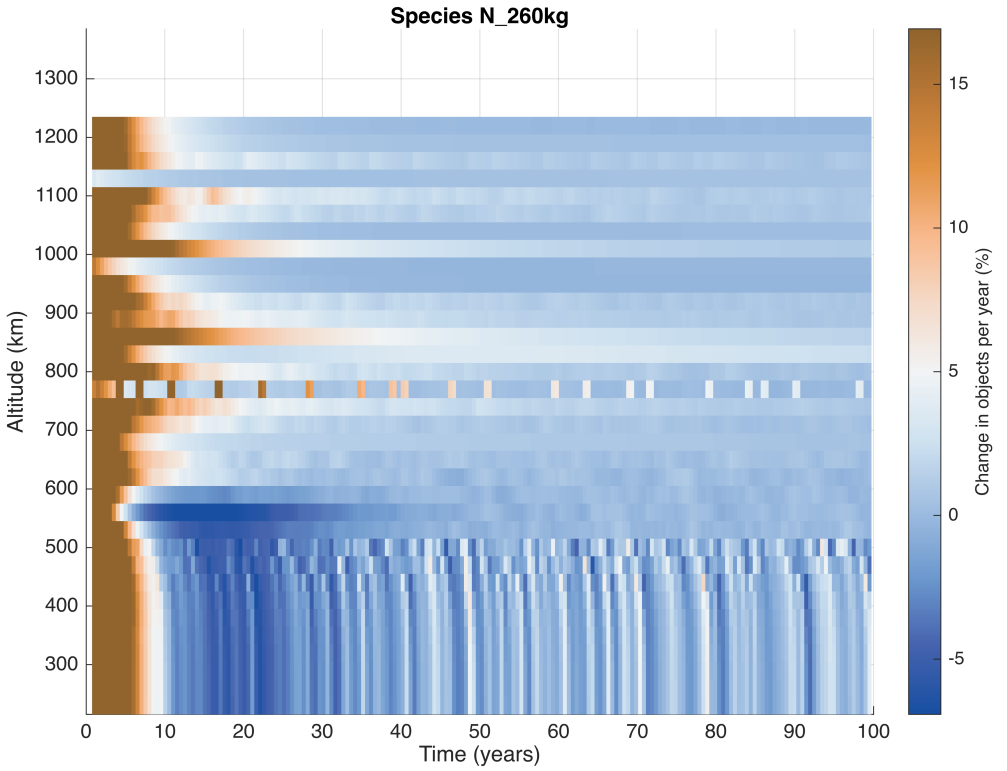
```
Producing visuals for the rate change of each debris species.
```

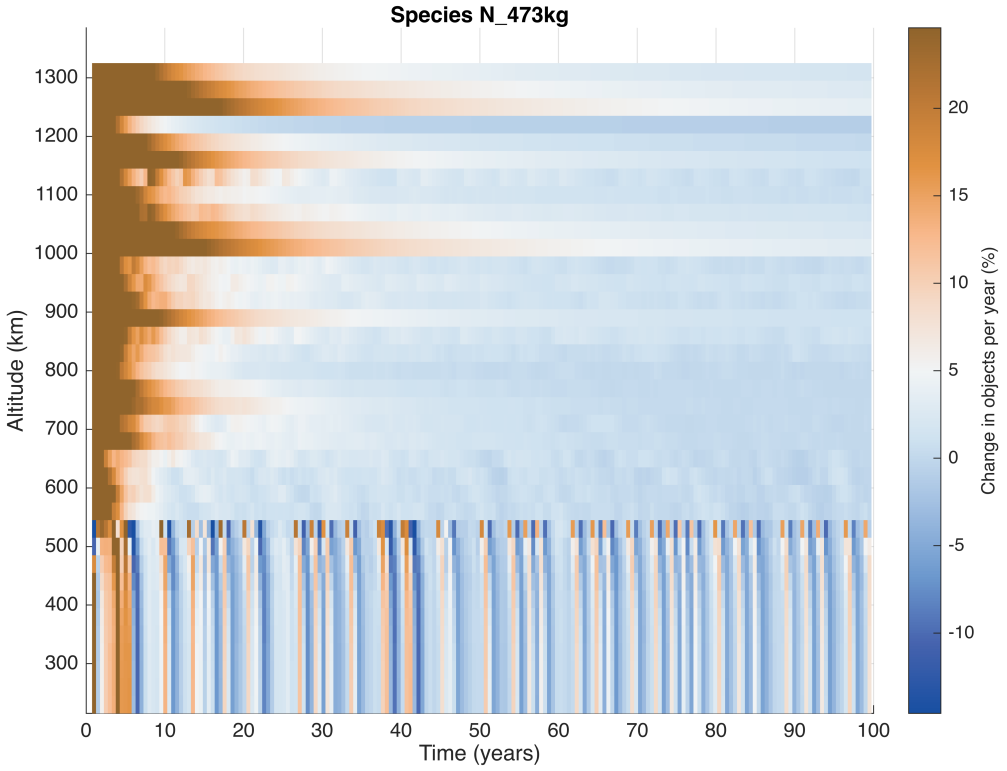


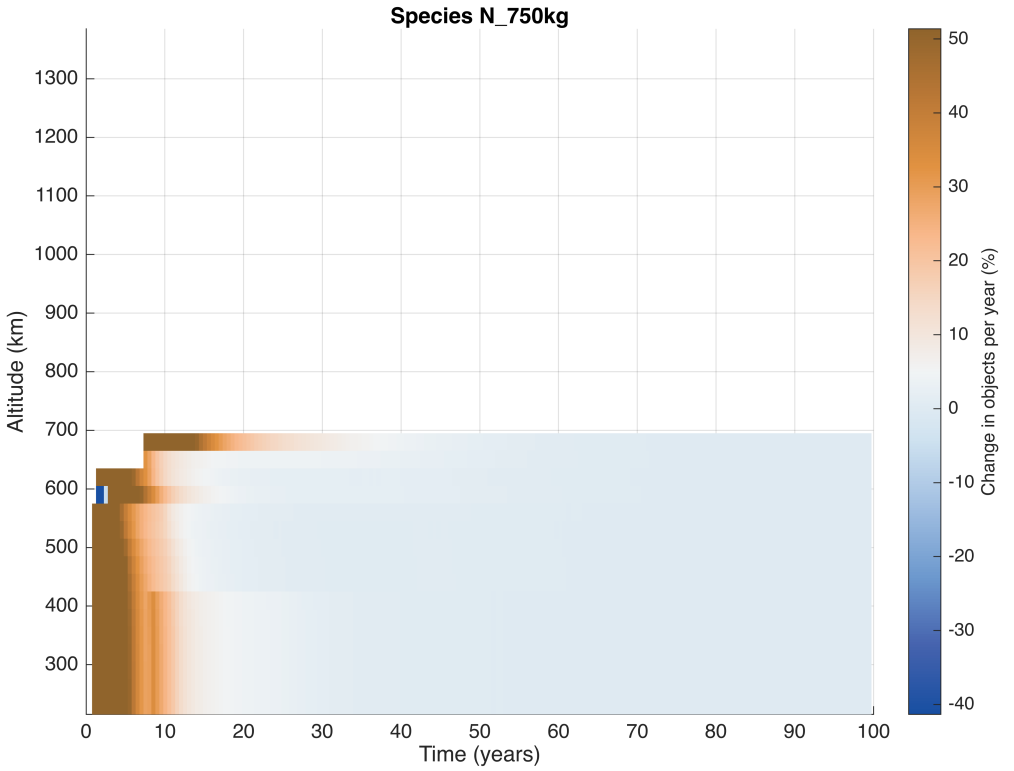


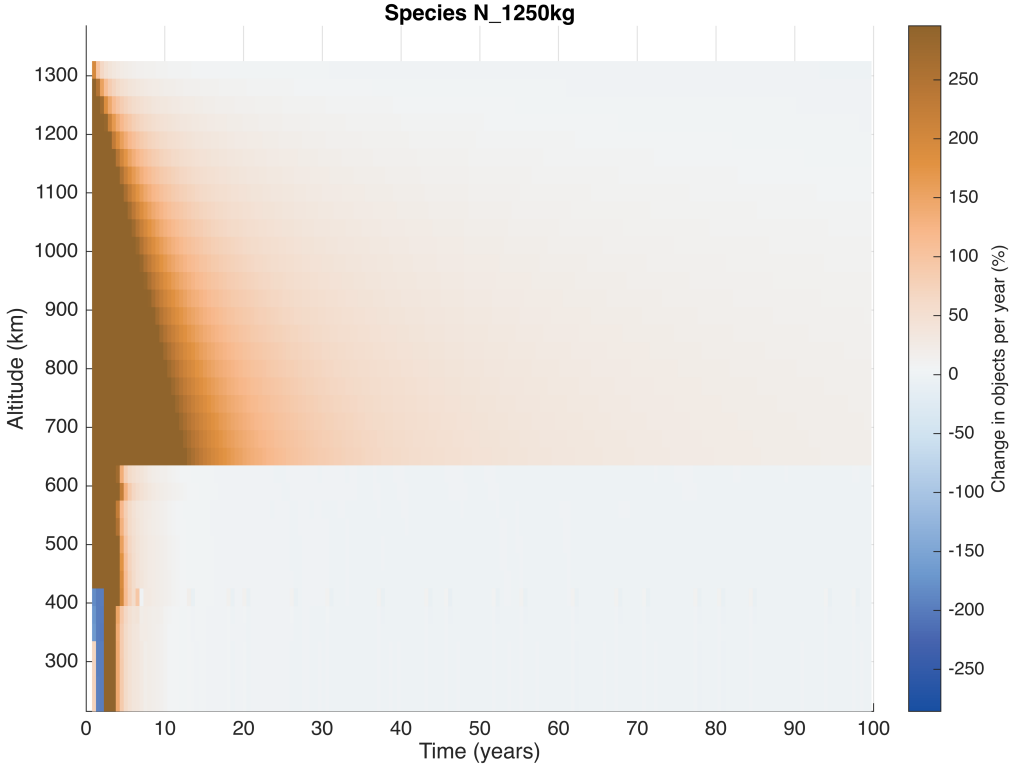

Species N_6kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
Change in objects per year (%)
10
9
8
7
6
5
4
3
2
1
0

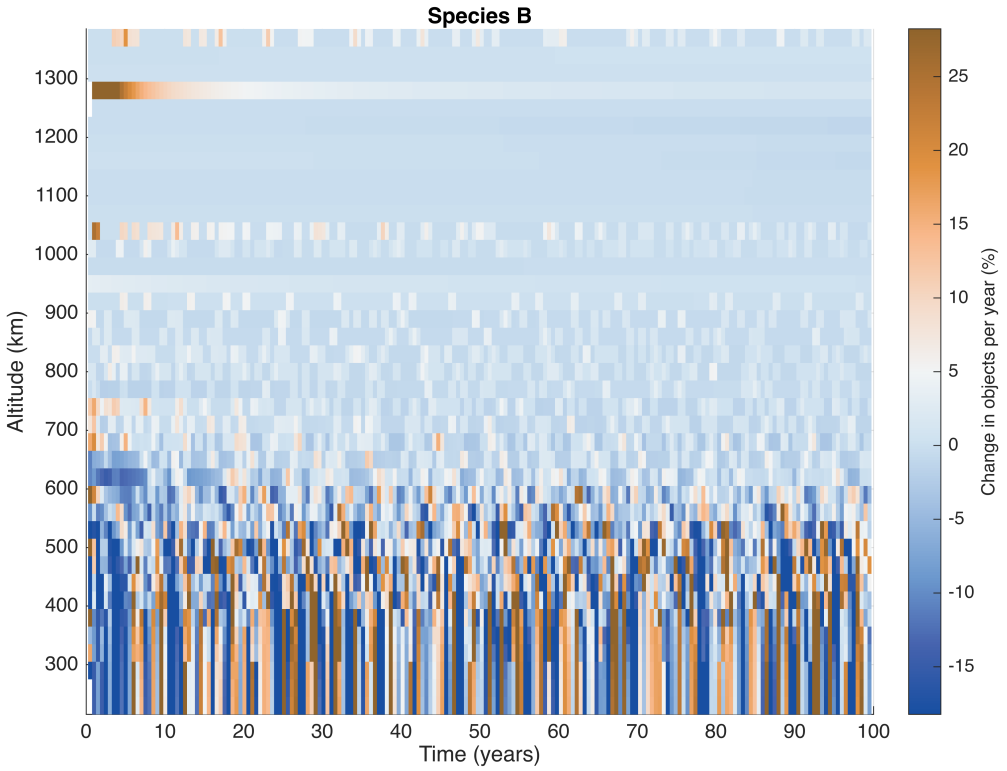

Species N_148kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
Change in objects per year (%)
10
9
8
7
6
5
4
3
2
1
0


Species N_260kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
15
10
5
0
-5
Change in objects per year (%)


Species N_473kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
20
15
10
5
0
-5
-10
Change in objects per year (%)


Species N_750kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
50
40
30
20
10
0
-10
-20
-30
-40
Change in objects per year (%)


Species N_1250kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
250
200
150
100
50
0
-50
-100
-150
-200
-250
Change in objects per year (%)



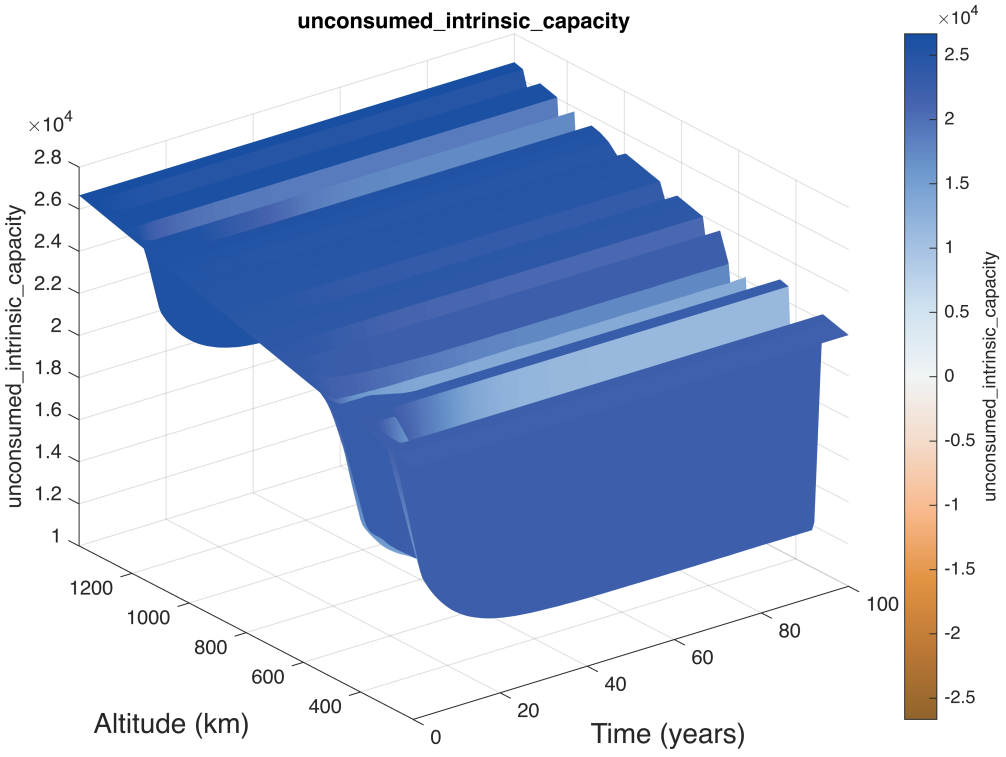

```
%my_sim.total_deb_pop_time_deriv2("constraint", 5, "percentage", true)
```

Show Indicators

Intrinsic Capacity

```
my_sim.per_shell_indicator("unconsumed_intrinsic_capacity", "constraint",
0.0, 'constraint_type', "lower")
```

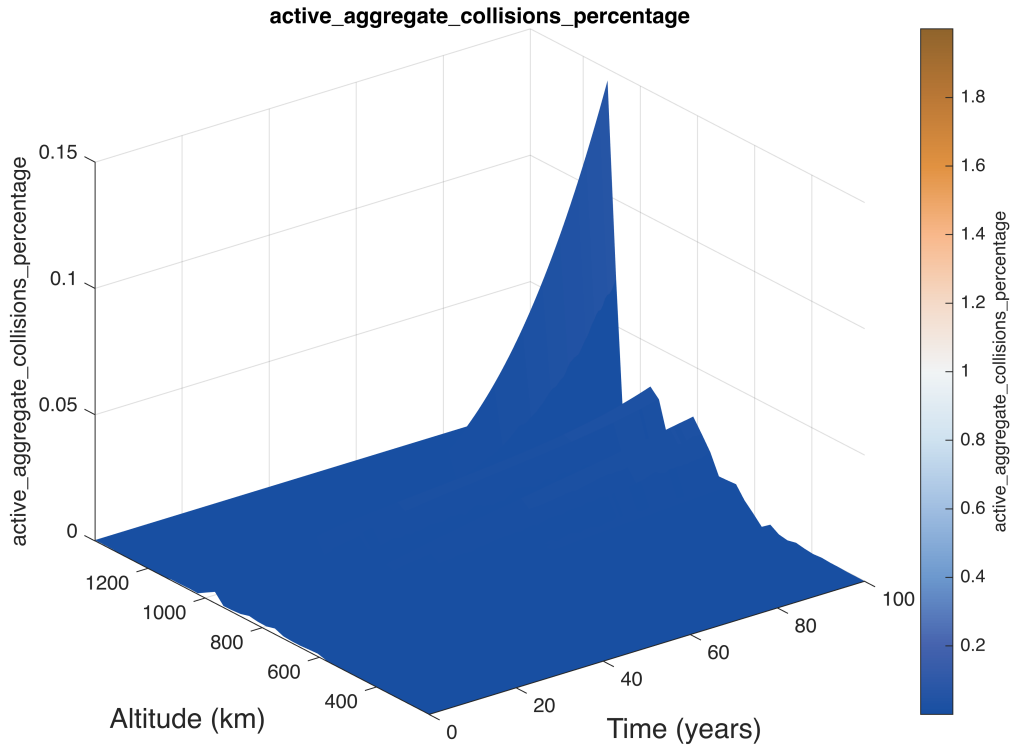
```
Producing visuals for the evolution of indicator.
```



Operational CA

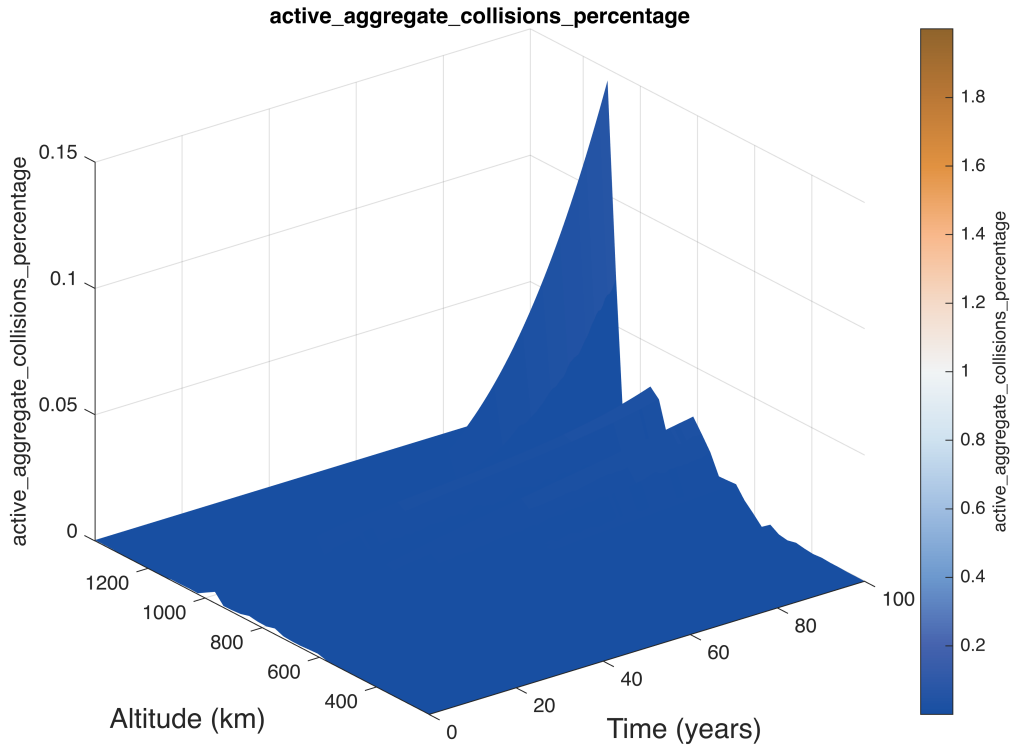
```
my_sim.per_shell_indicator("active_aggregate_collisions_percentage",
"constraint", 1.0)
```

```
Producing visuals for the evolution of indicator.
```

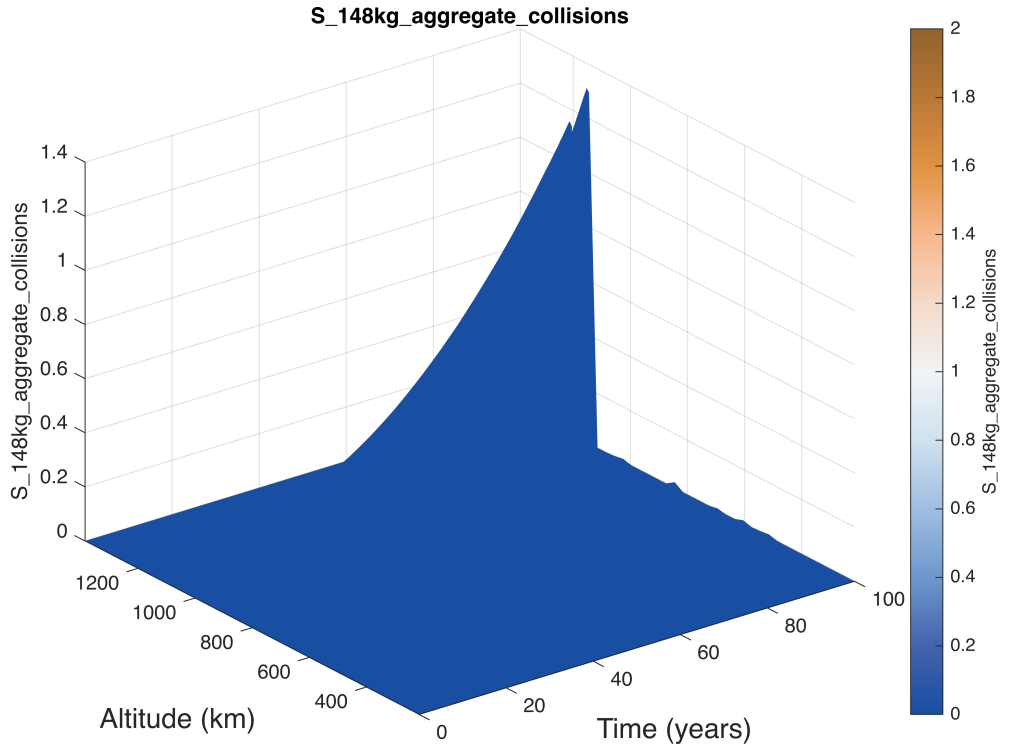



```
for ii = 1:length(scen_properties.indicator_var_list)
    if contains(scen_properties.indicator_var_list(ii).name,
"aggregate_collisions")
        disp(scen_properties.indicator_var_list(ii).name)

my_sim.per_shell_indicator(scen_properties.indicator_var_list(ii).name,
"constraint", 1.0, 'constraint_type', "upper")
    end
end
```

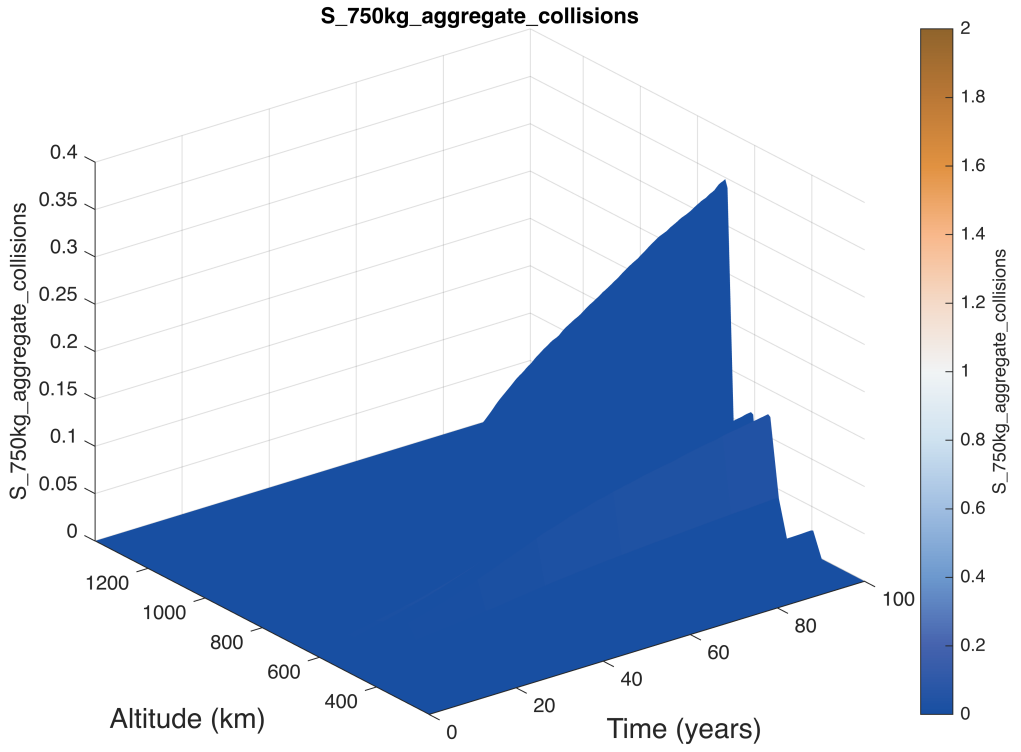
```
active_aggregate_collisions_percentage
Producing visuals for the evolution of indicator.
```



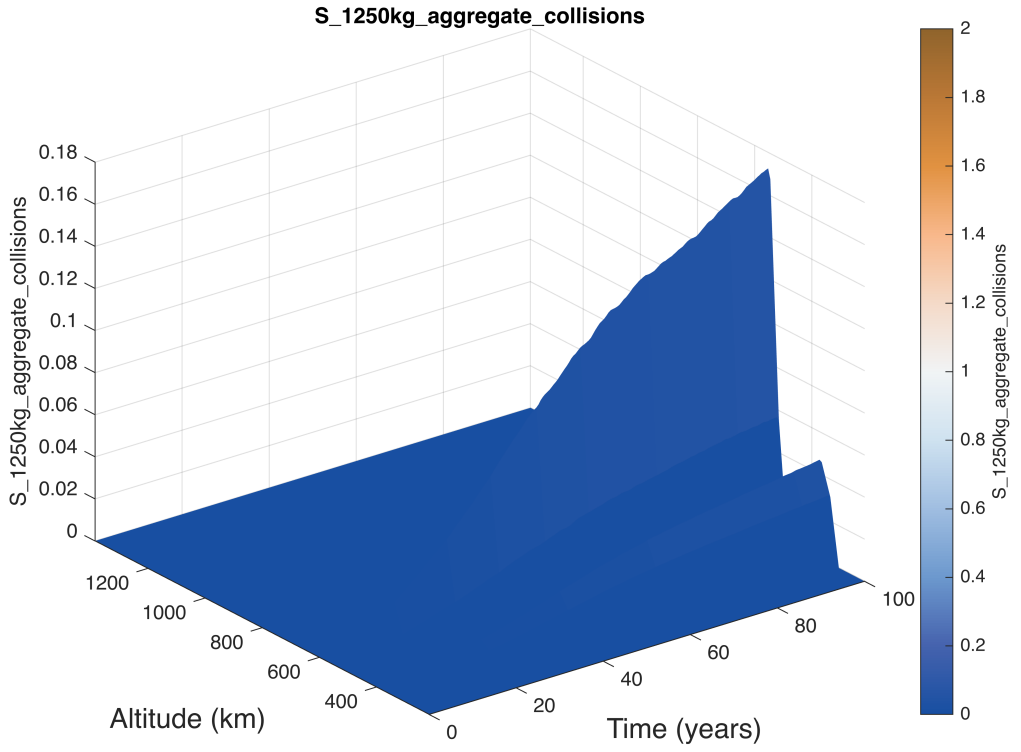

```
S_148kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



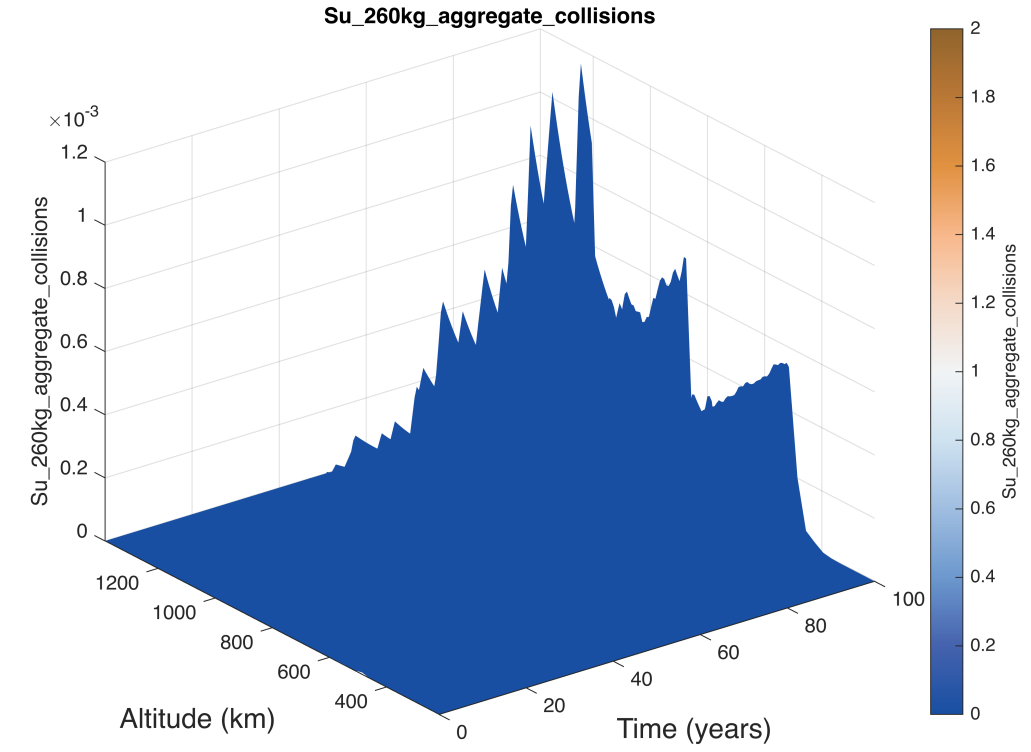

```
S_750kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



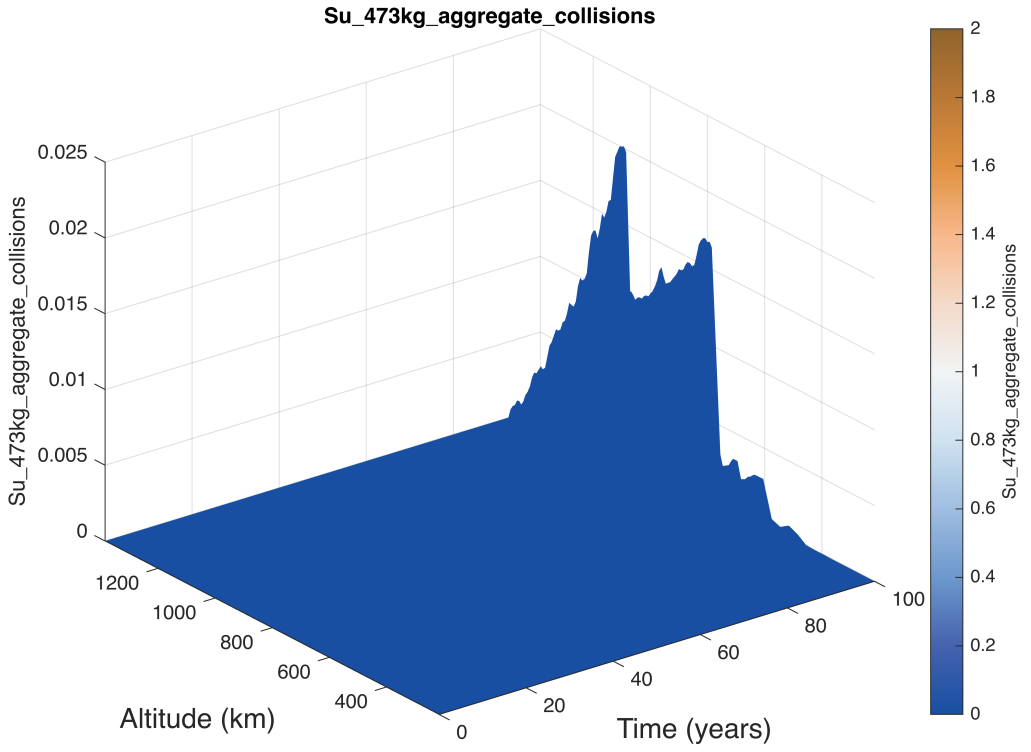
S_1250kg_aggregate_collisions
Producing visuals for the evolution of indicator.



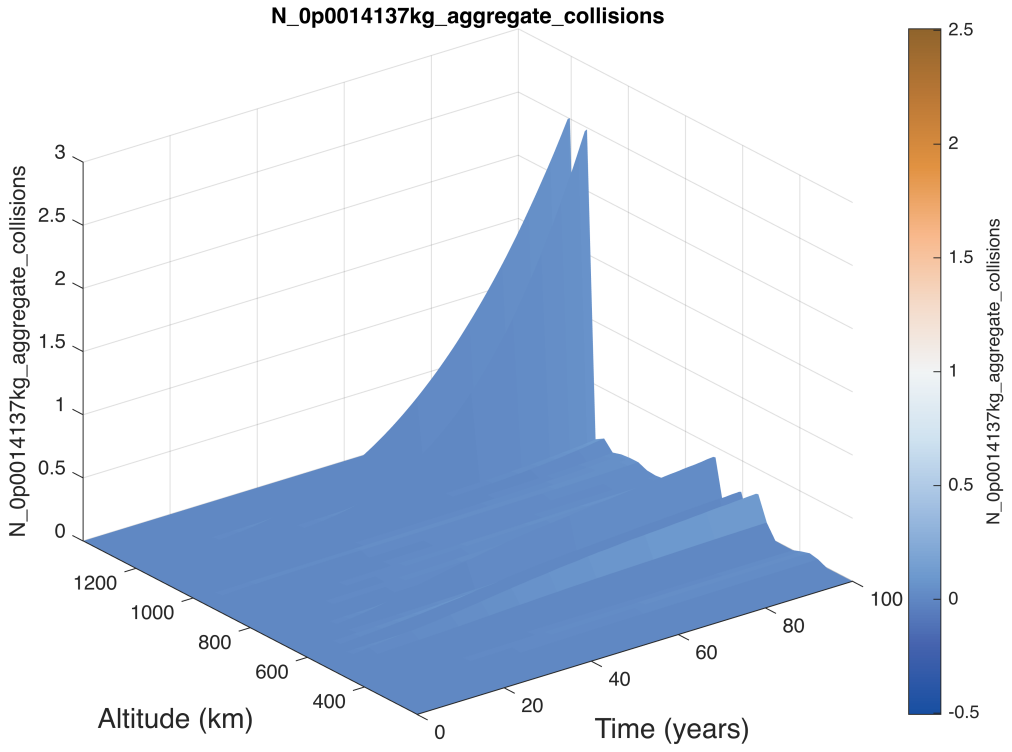
Su_260kg_aggregate_collisions
Producing visuals for the evolution of indicator.



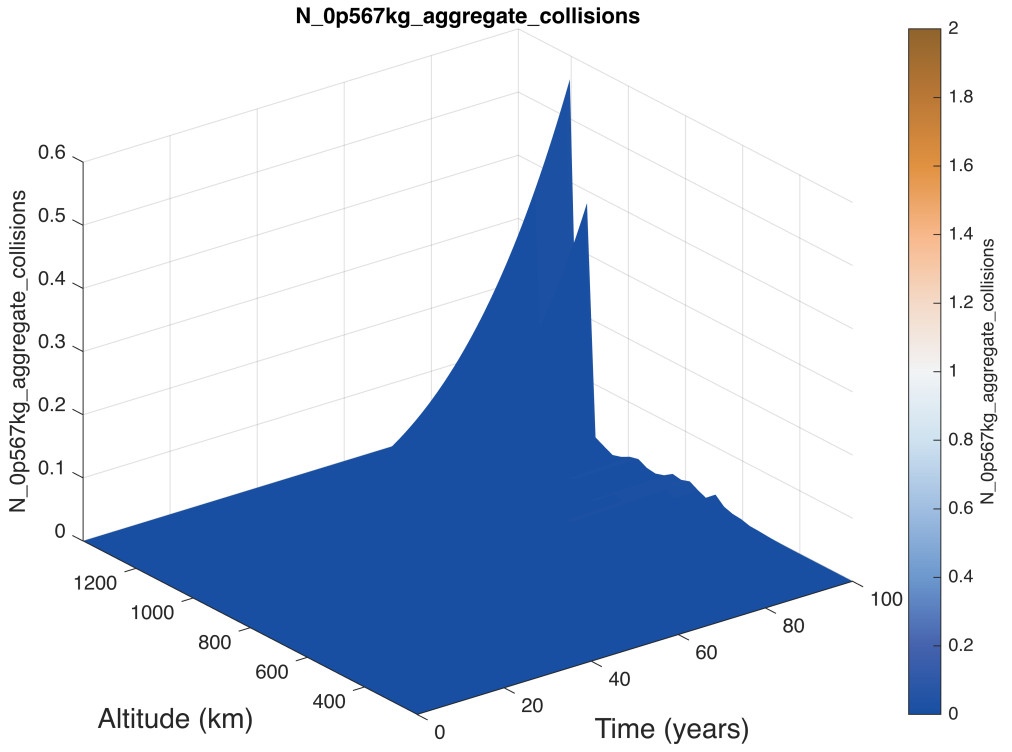

```
Su_473kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



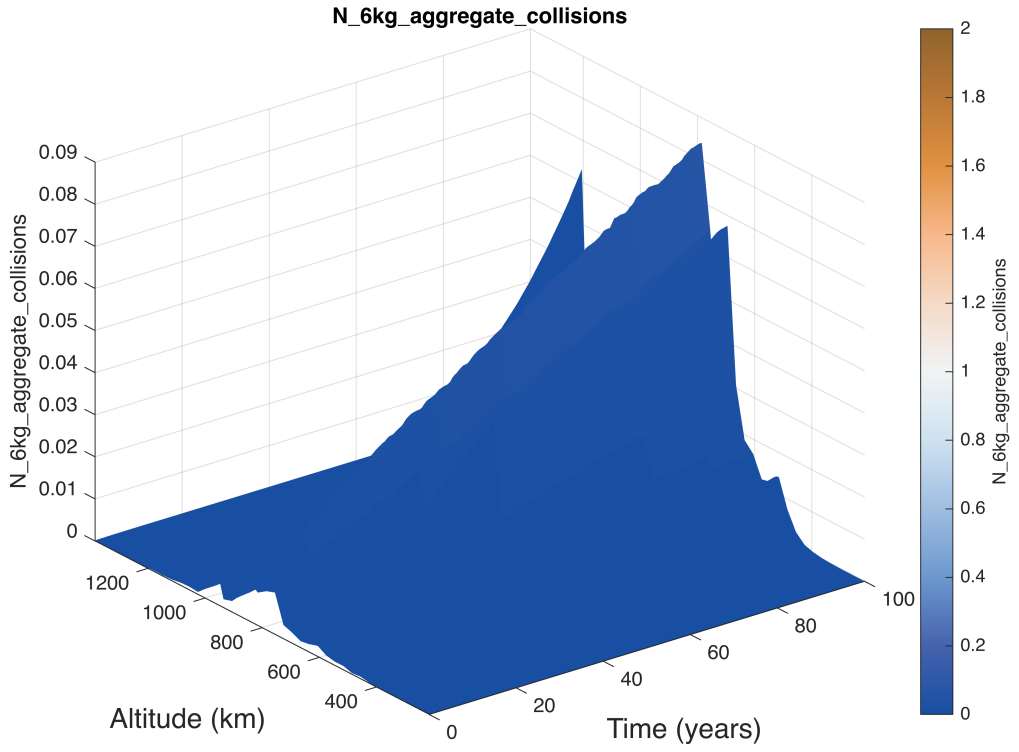

```
N_0p0014137kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



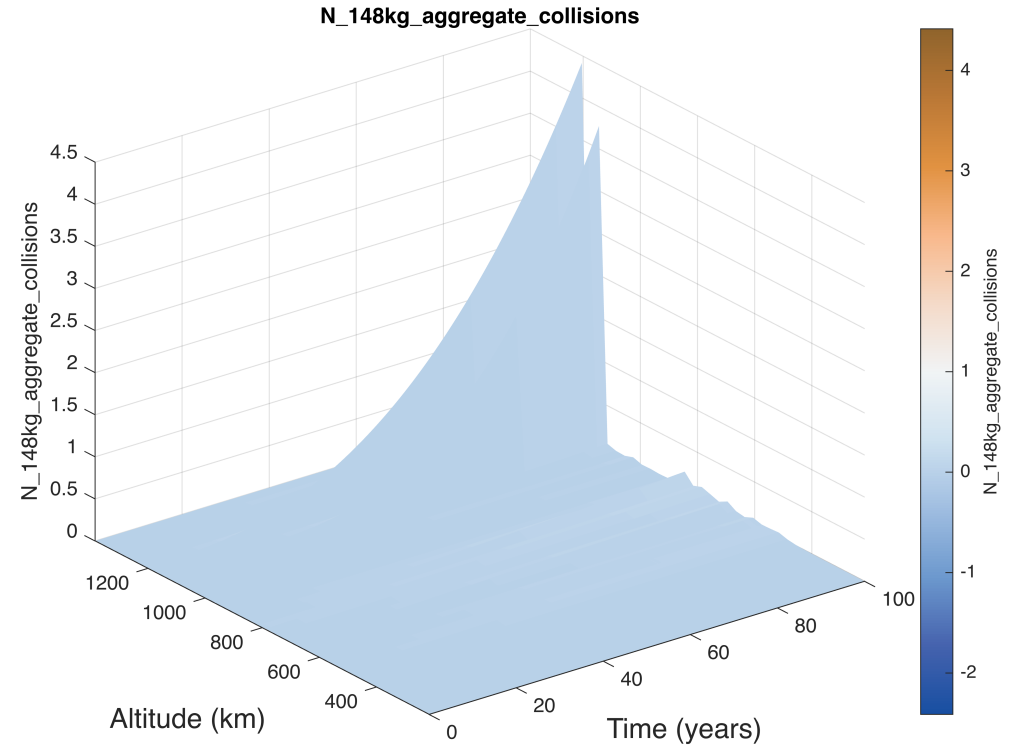
N_0p567kg_aggregate_collisions
Producing visuals for the evolution of indicator.



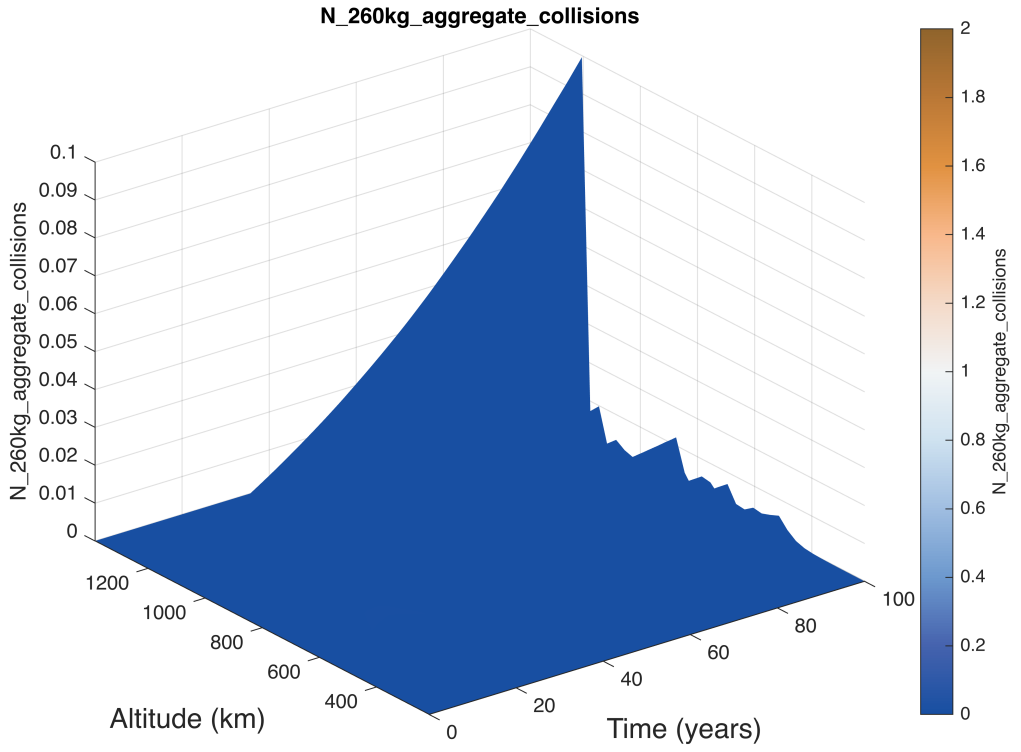
N_6kg_aggregate_collisions
Producing visuals for the evolution of indicator.



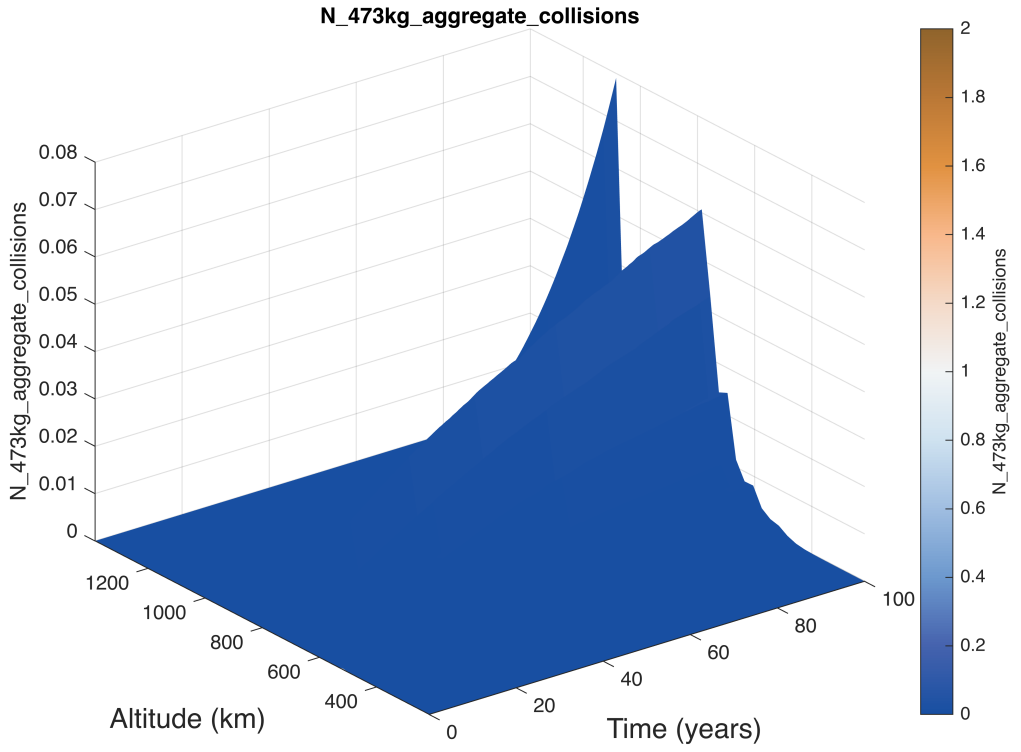

```
N_148kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



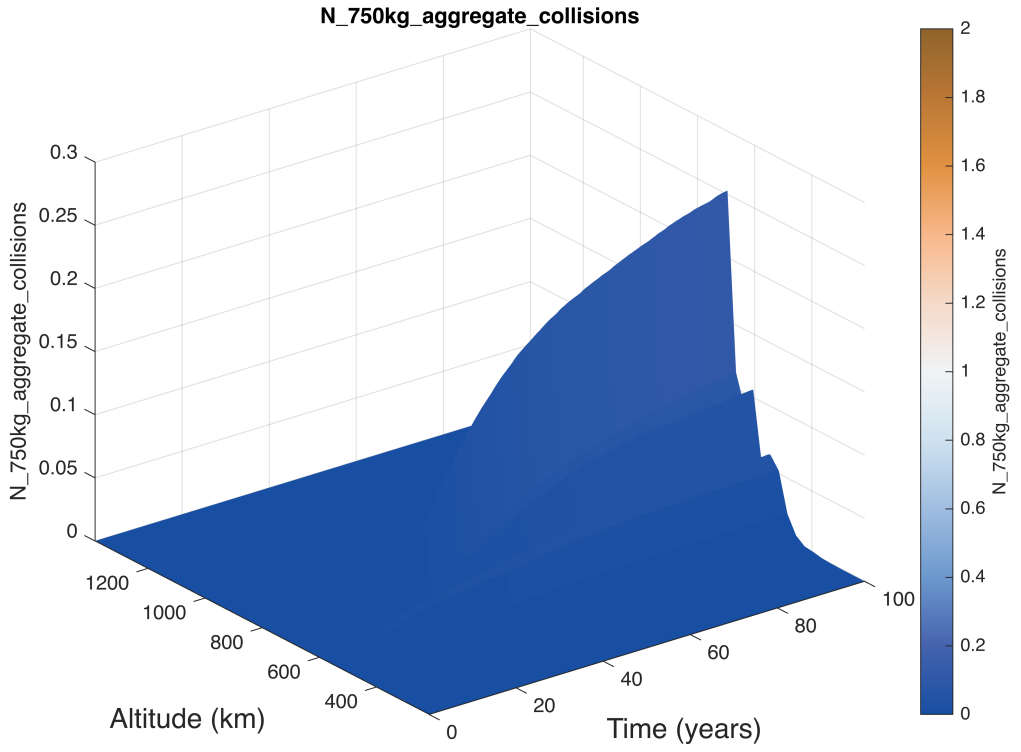

```
N_260kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



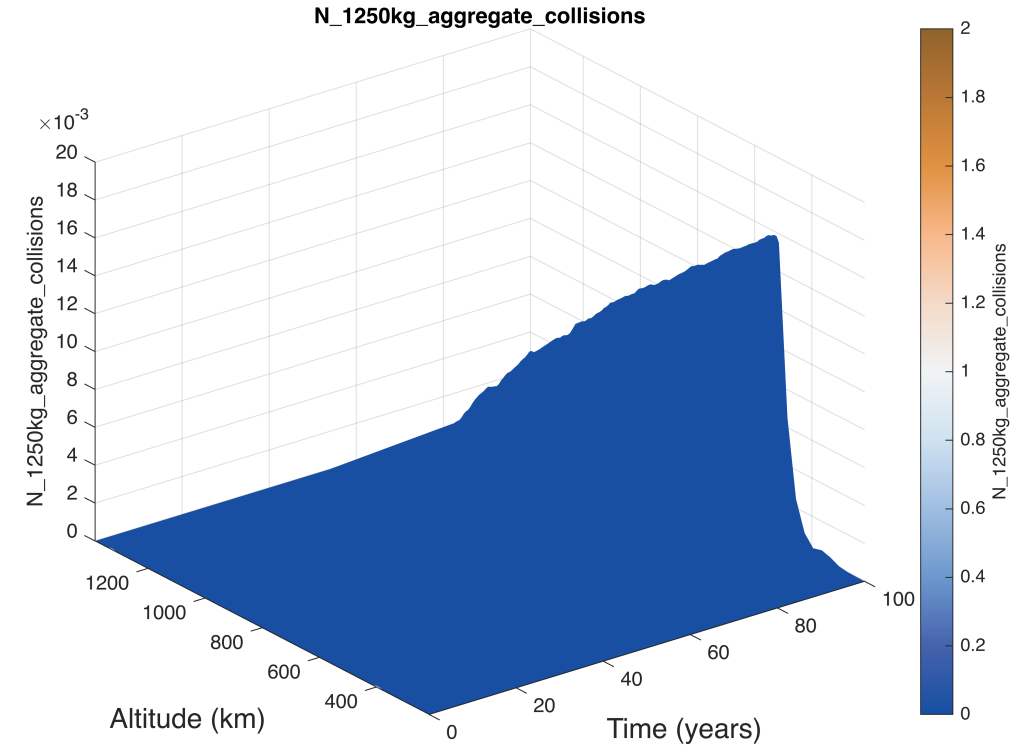

```
N_473kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



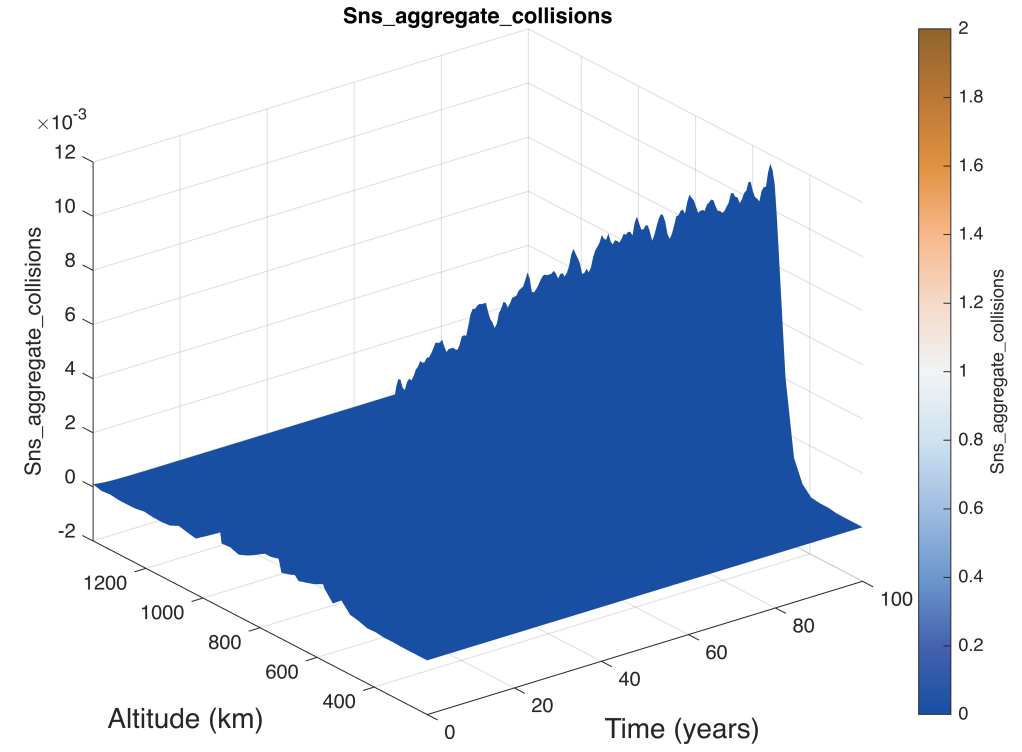

```
N_750kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



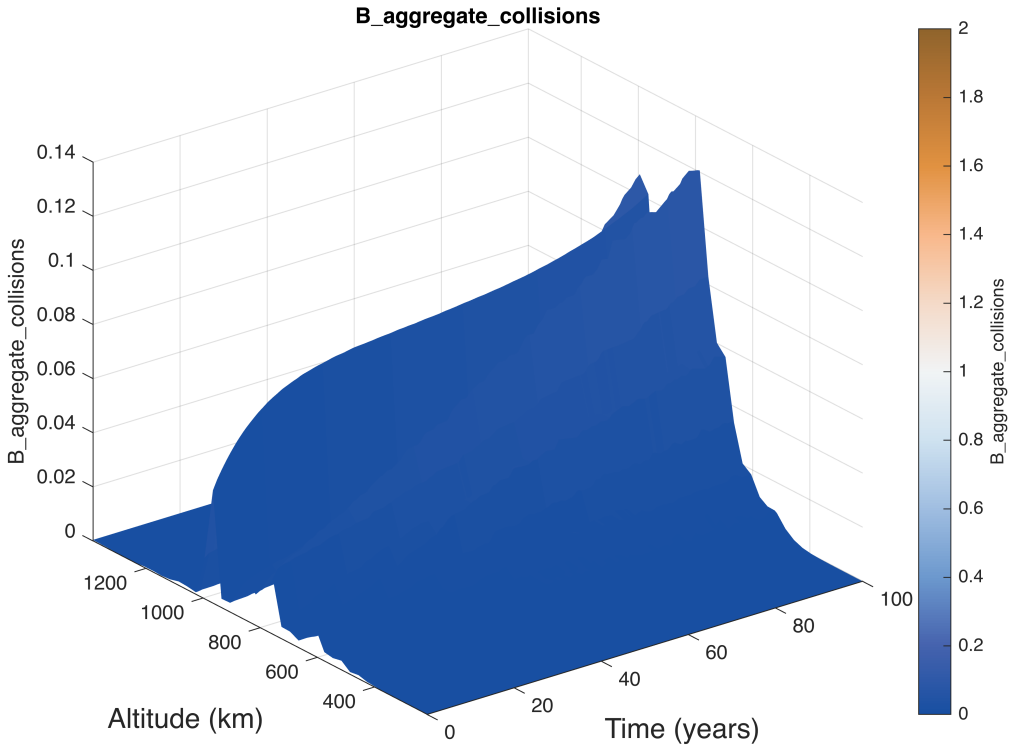
N_1250kg_aggregate_collisions
Producing visuals for the evolution of indicator.



Sns_aggregate_collisions
Producing visuals for the evolution of indicator.

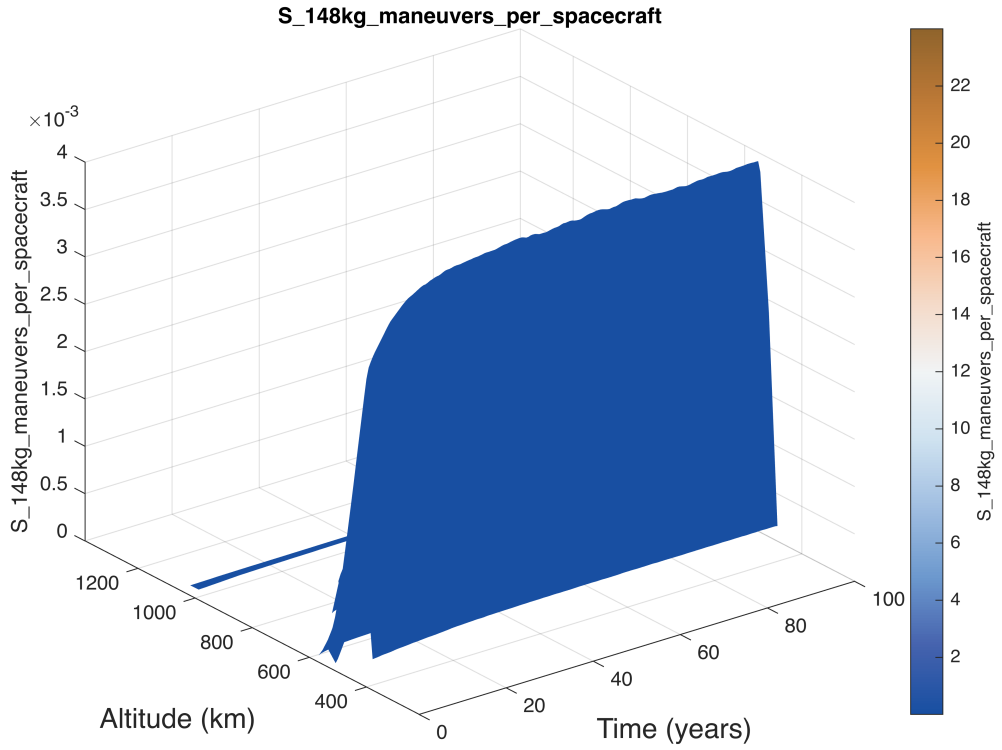



```
B_aggregate_collisions
Producing visuals for the evolution of indicator.
```

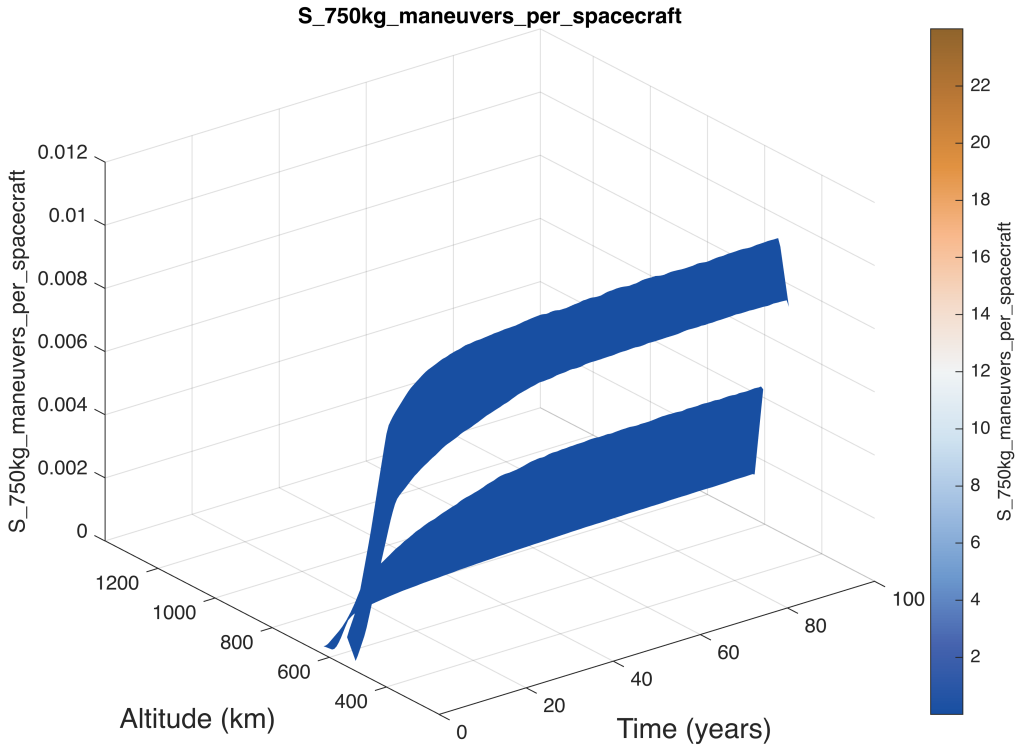


```
for ii = 1:length(scen_properties.indicator_var_list)
    if contains(scen_properties.indicator_var_list(ii).name,
"maneuvers_per_spacecraft") &&
contains(scen_properties.indicator_var_list(ii).name, "S")
        disp(scen_properties.indicator_var_list(ii).name)

my_sim.per_shell_indicator(scen_properties.indicator_var_list(ii).name,
'constraint', 12,'constraint_type', 'upper')
    end
end
```

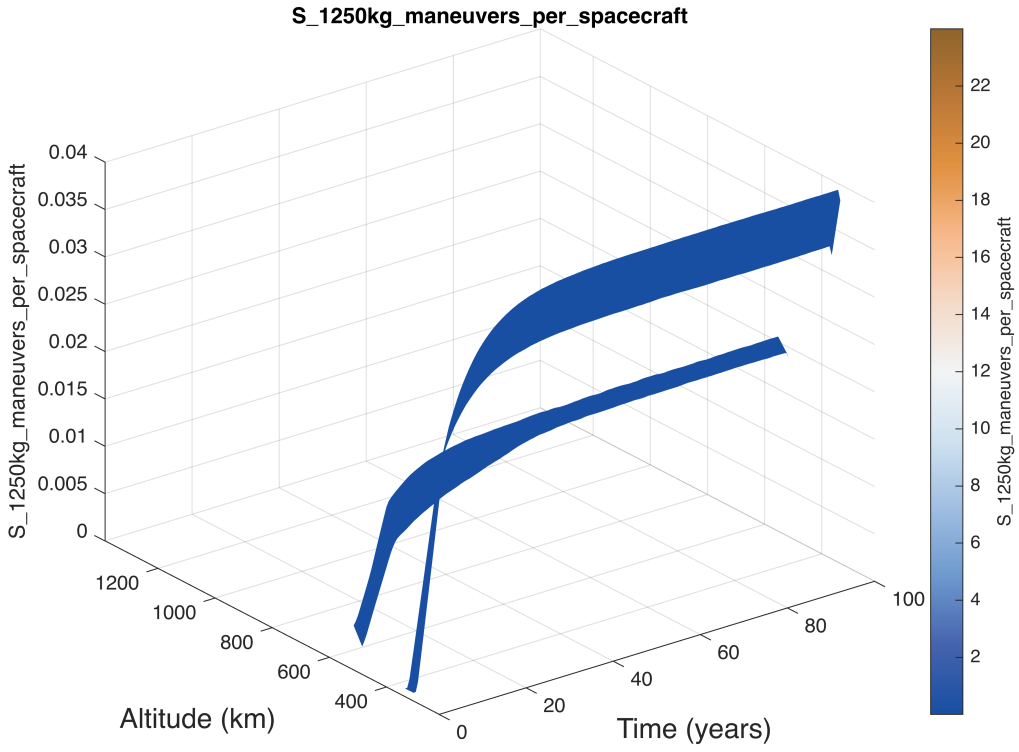
```
S_148kg_maneuvers_per_spacecraft
Producing visuals for the evolution of indicator.
```



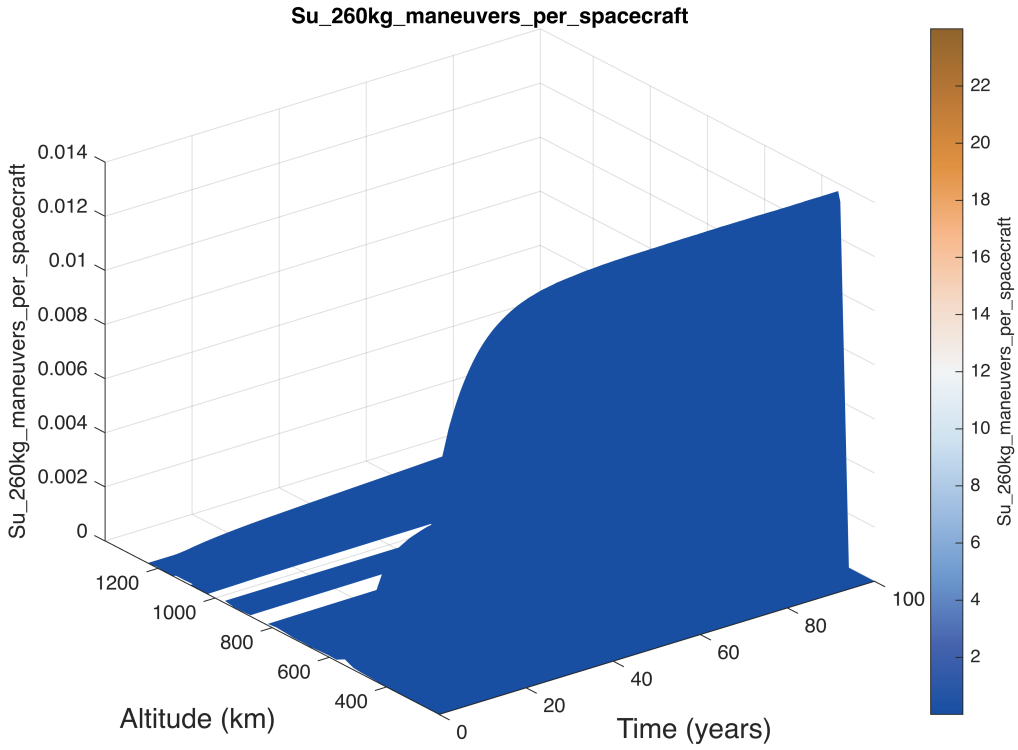

```
S_750kg_maneuvers_per_spacecraft
Producing visuals for the evolution of indicator.
```



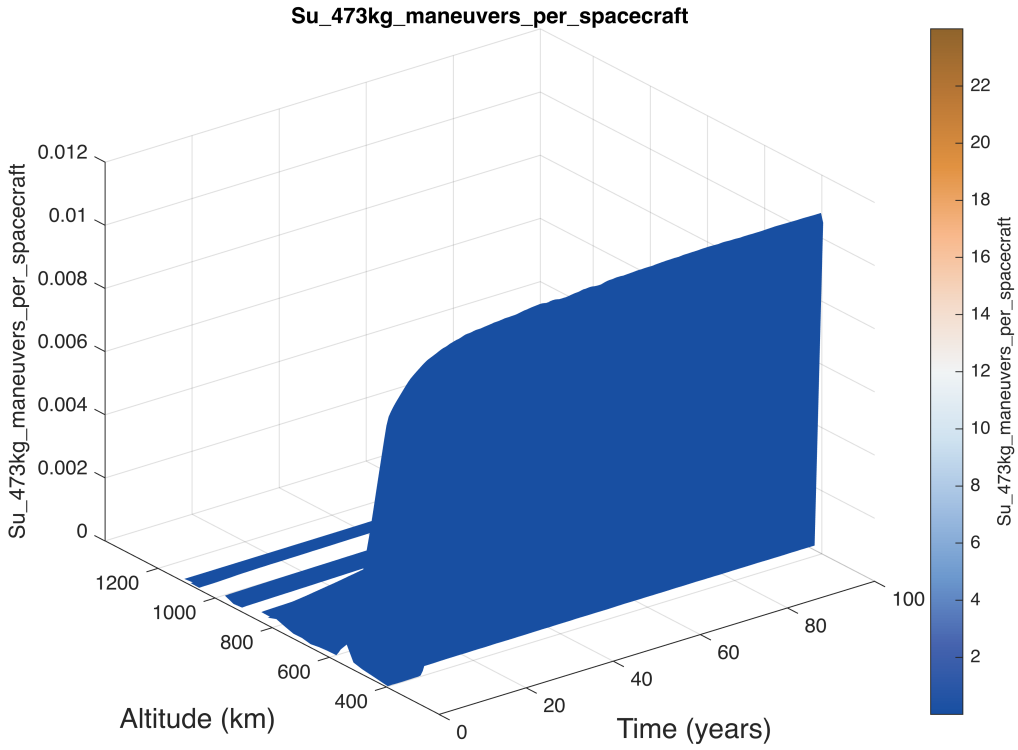
S_1250kg_maneuvers_per_spacecraft
Producing visuals for the evolution of indicator.



Su_260kg_maneuvers_per_spacecraft
Producing visuals for the evolution of indicator.

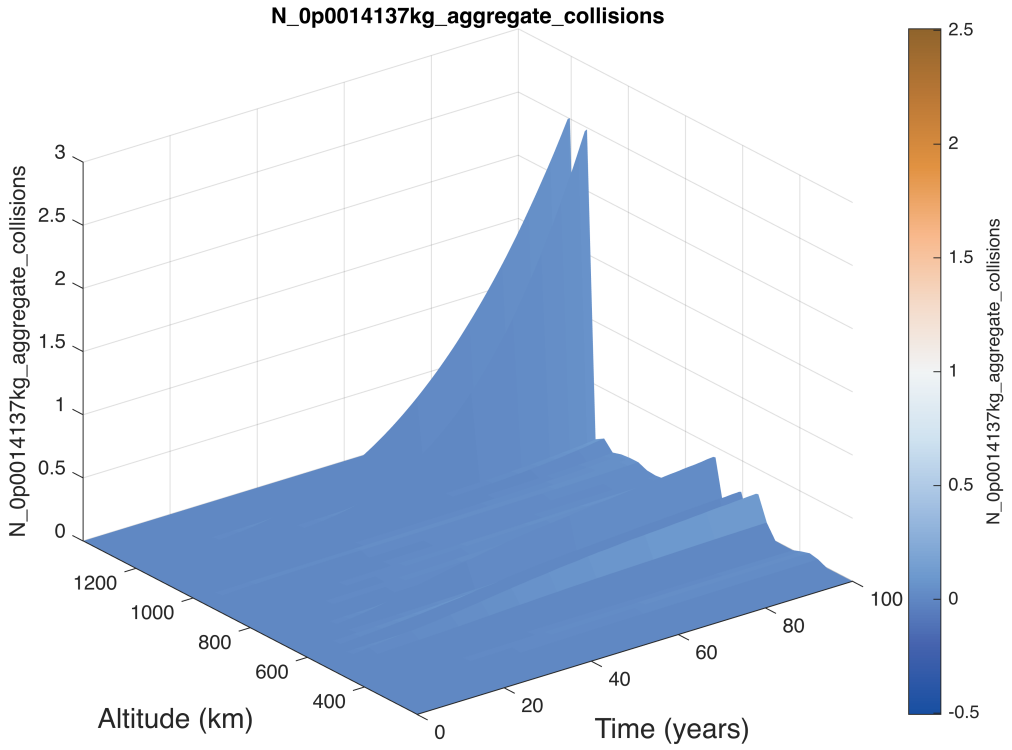



```
Su_473kg_maneuvers_per_spacecraft
Producing visuals for the evolution of indicator.
```



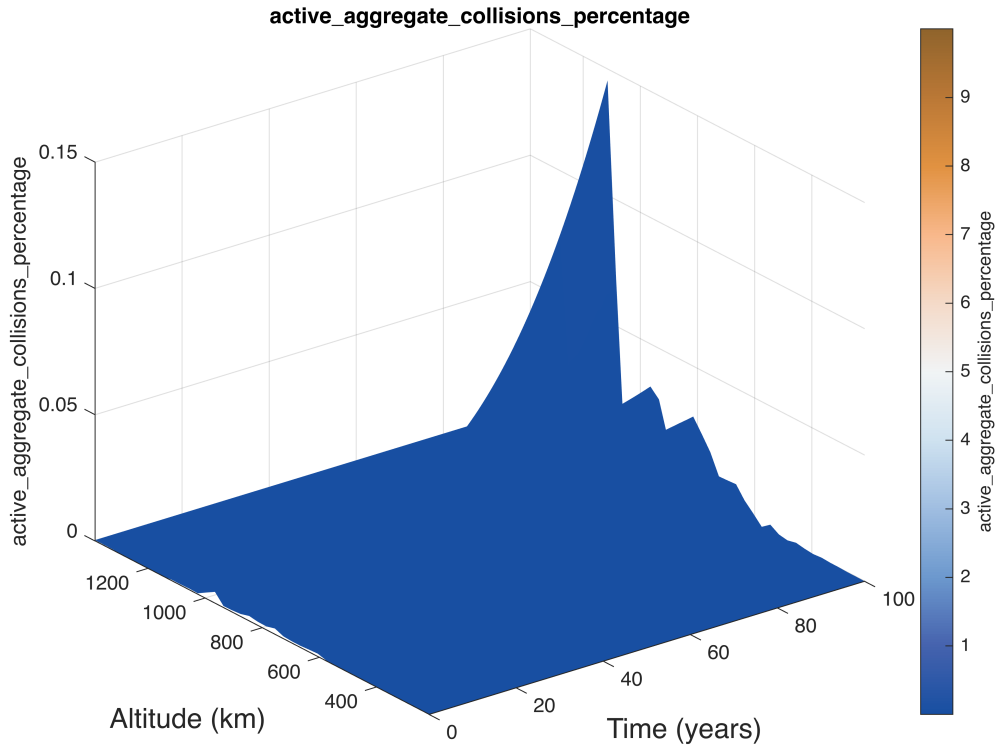
```
for ii = 1:length(scen_properties.indicator_var_list)
    if contains(scen_properties.indicator_var_list(ii).name, "collisions")
&& contains(scen_properties.indicator_var_list(ii).name, "N_0p0014137kg")
        disp(scen_properties.indicator_var_list(ii).name)

my_sim.per_shell_indicator(scen_properties.indicator_var_list(ii).name,
'constraint', 1,'constraint_type', 'upper')
    end
end
```

```
N_0p0014137kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```




```
my_sim.per_shell_indicator("active_aggregate_collisions_percentage",
"constraint", 5.0, 'constraint_type', "upper")
```

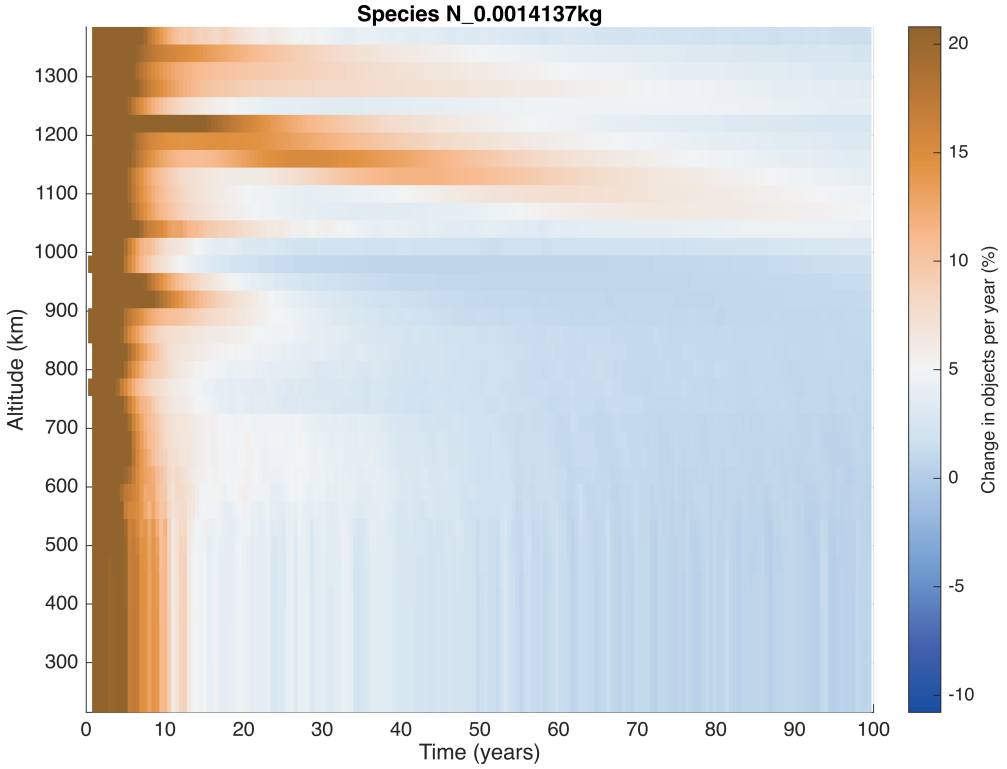
```
Producing visuals for the evolution of indicator.
```

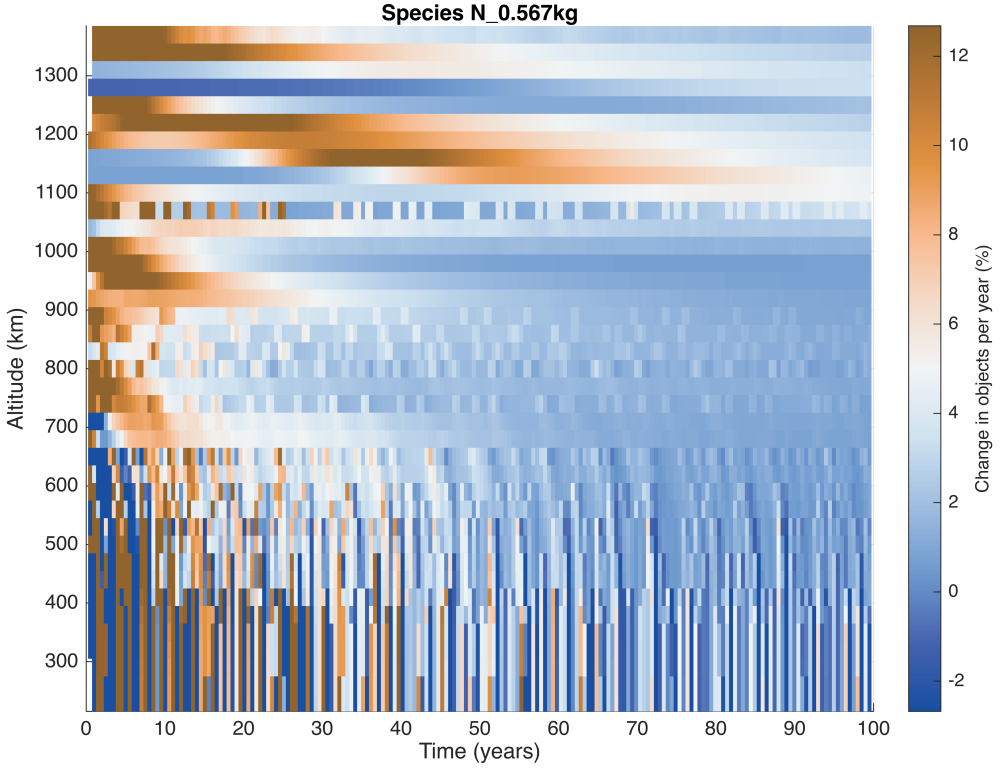


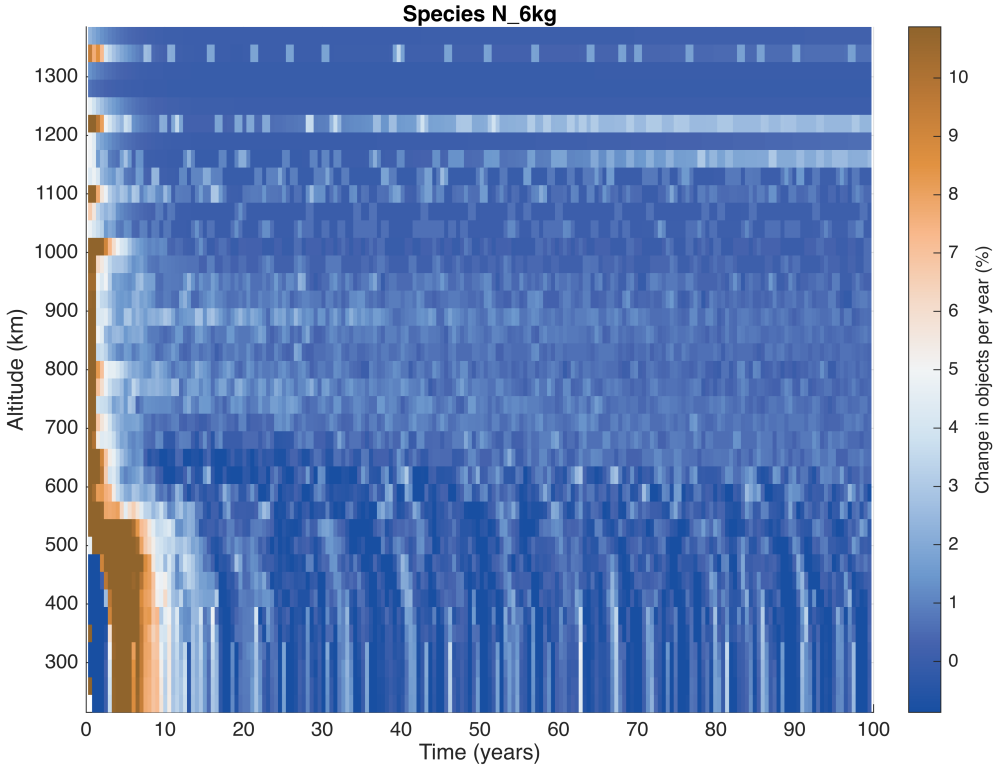
Long-Term Sustainability

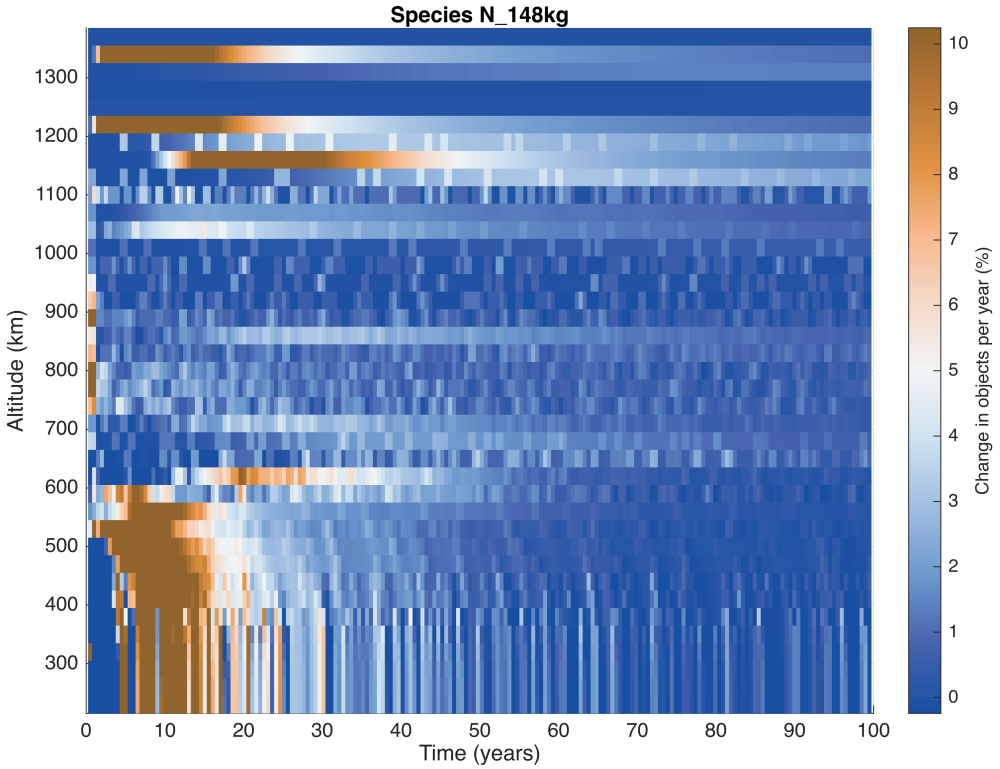
```
my_sim.total_deb_species_deriv_evolv_vis("percentage", true,
"color_scale_crop_percentiles", [5, 95], "constraint", 5)
```

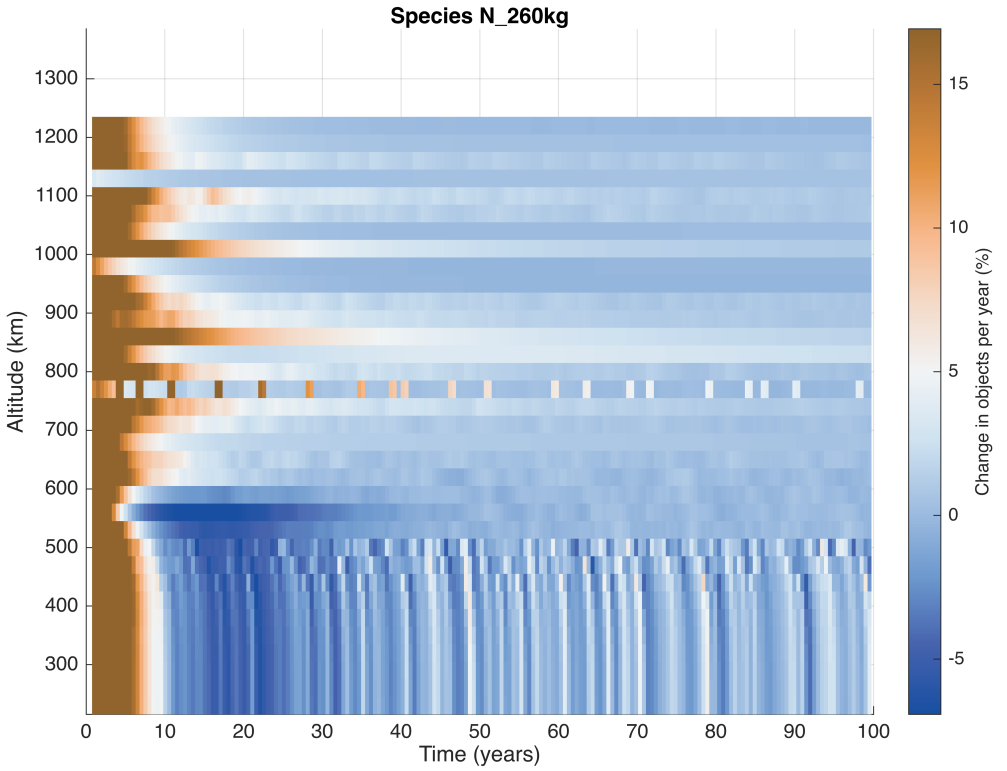
```
Producing visuals for the rate change of each debris species.
```

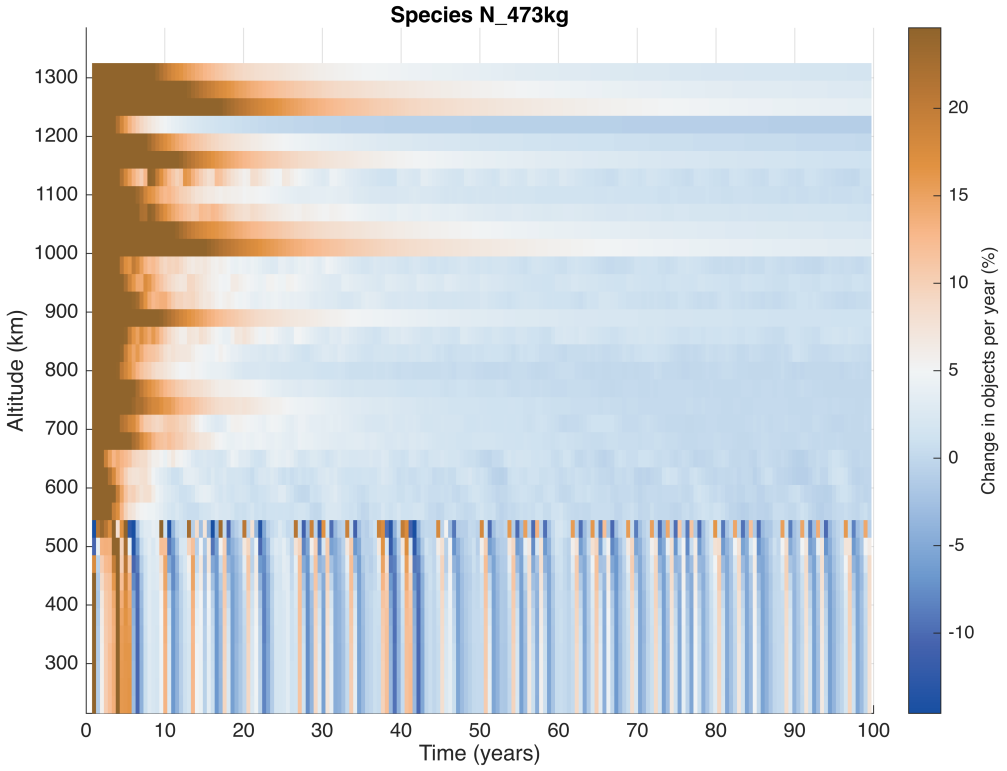

Species N_0.0014137kg
Altitude (km)
Time (years)
Change in objects per year (%)

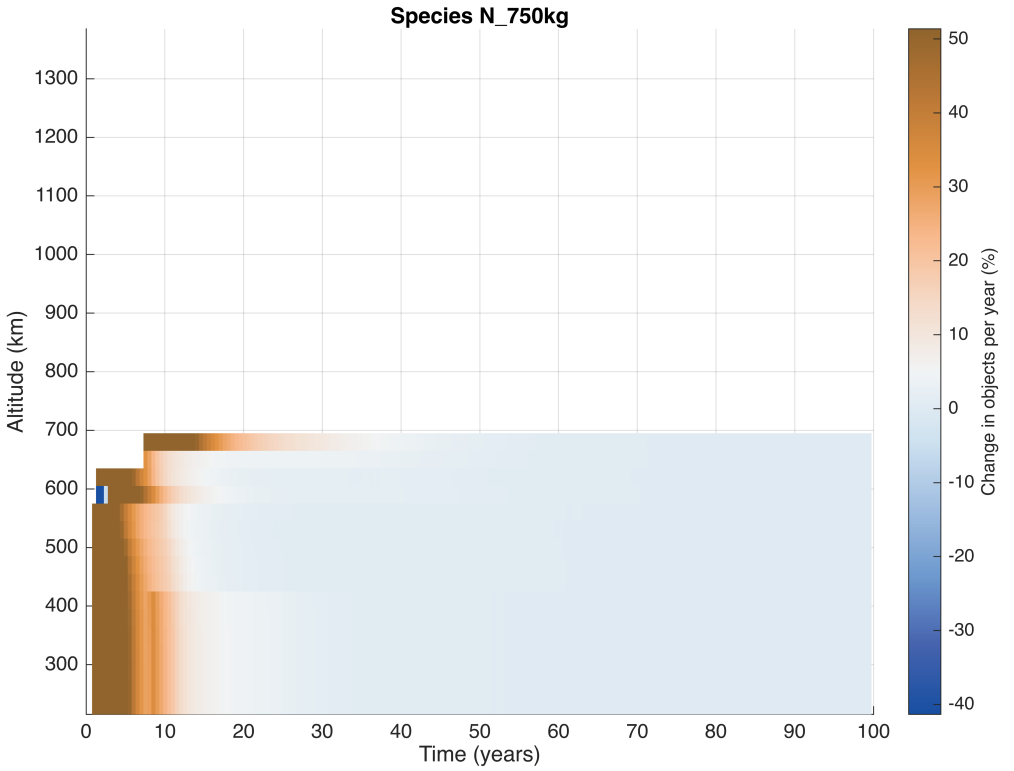

Species N_0.567kg
Altitude (km)
Time (years)
Change in objects per year (%)

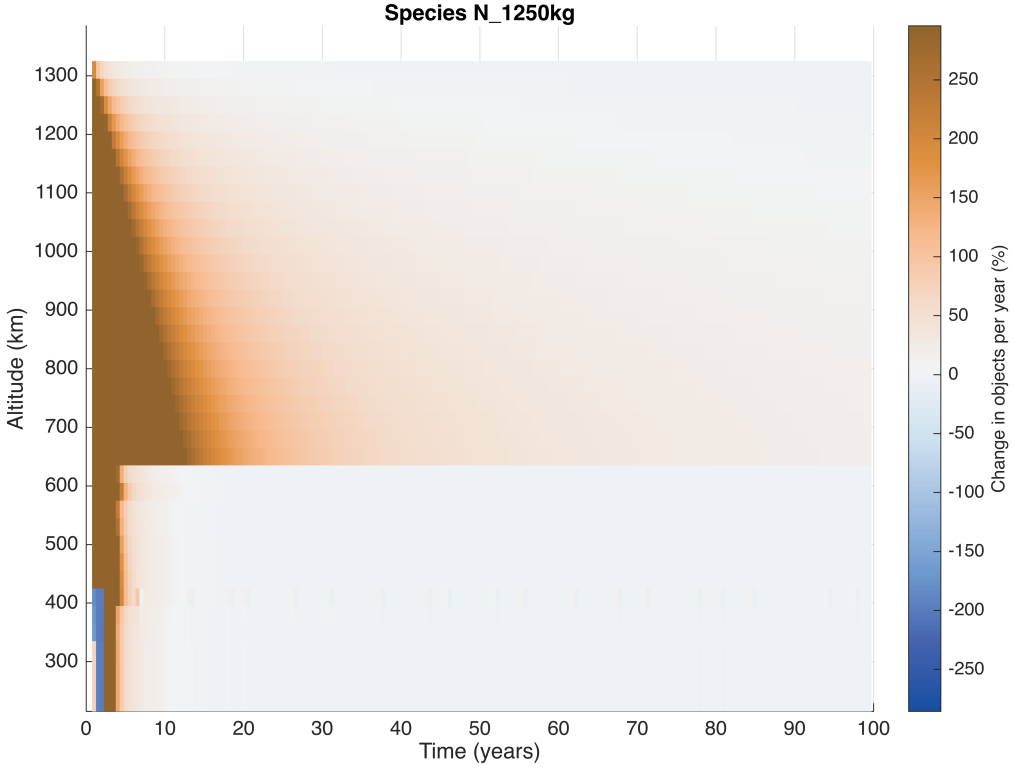

Species N_6kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
Change in objects per year (%)
10
9
8
7
6
5
4
3
2
1
0

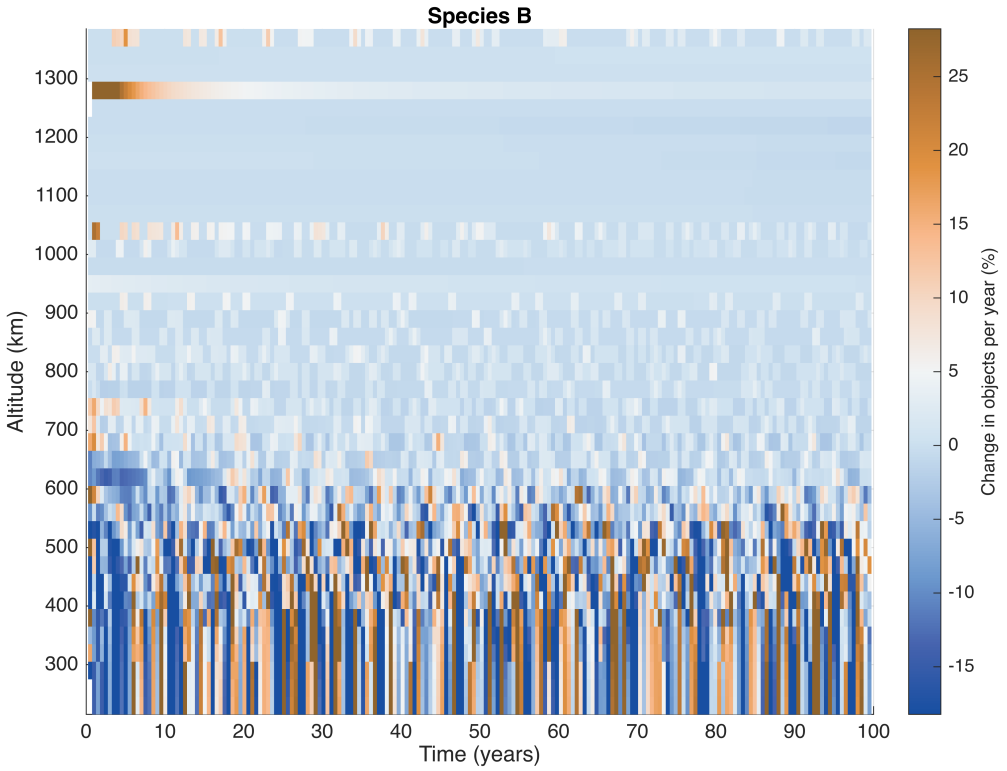

Species N_148kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
Change in objects per year (%)
10
9
8
7
6
5
4
3
2
1
0


Species N_260kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
15
10
5
0
-5
Change in objects per year (%)


Species N_473kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
20
15
10
5
0
-5
-10
Change in objects per year (%)

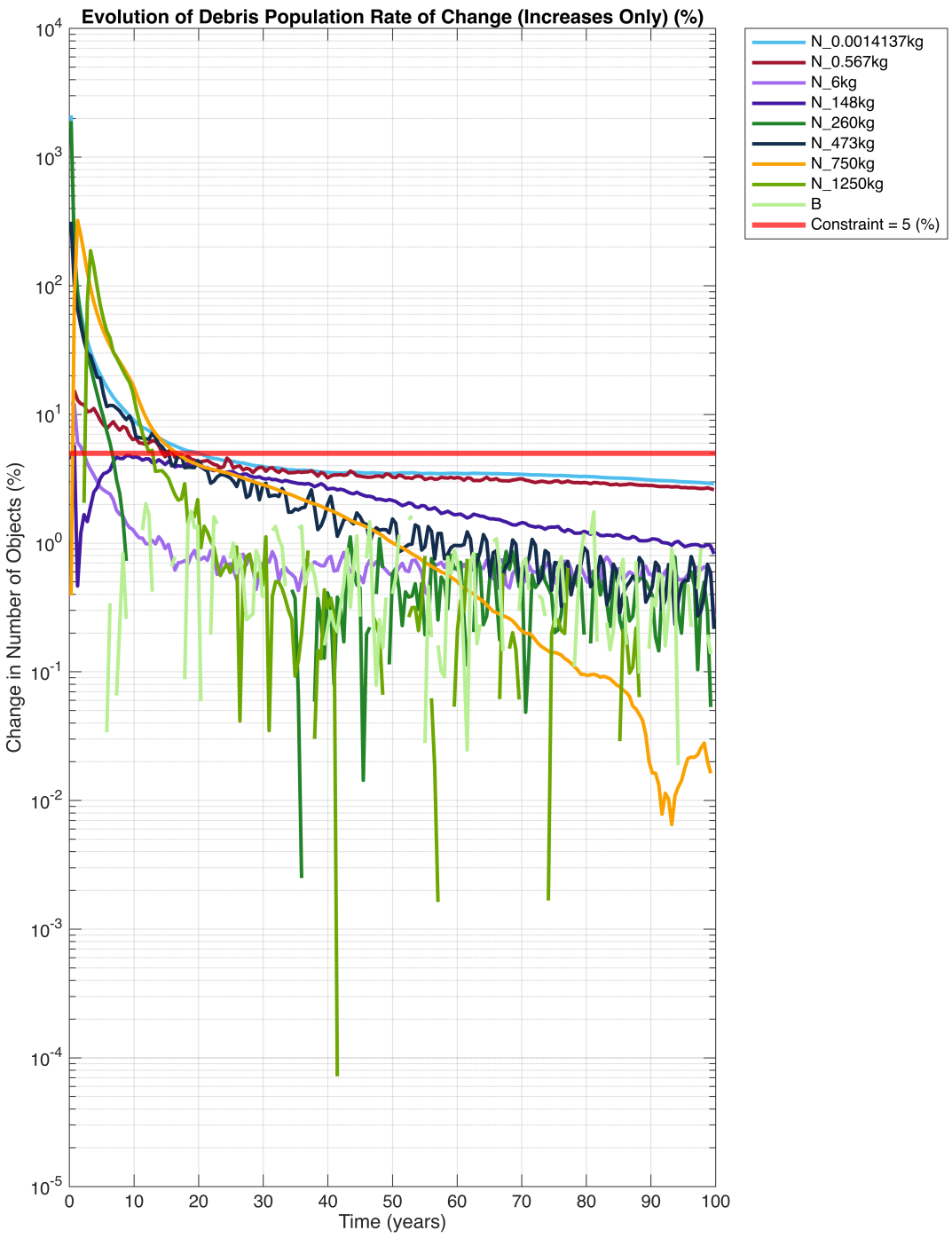

Species N_750kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
50
40
30
20
10
0
-10
-20
-30
-40
Change in objects per year (%)


Species N_1250kg
Altitude (km)
1300
1200
1100
1000
900
800
700
600
500
400
300
0
10
20
30
40
50
60
70
80
90
100
Time (years)
250
200
150
100
50
0
-50
-100
-150
-200
-250
Change in objects per year (%)




```
my_sim.total_deb_pop_time_deriv("constraint", 5, "percentage", true)
```

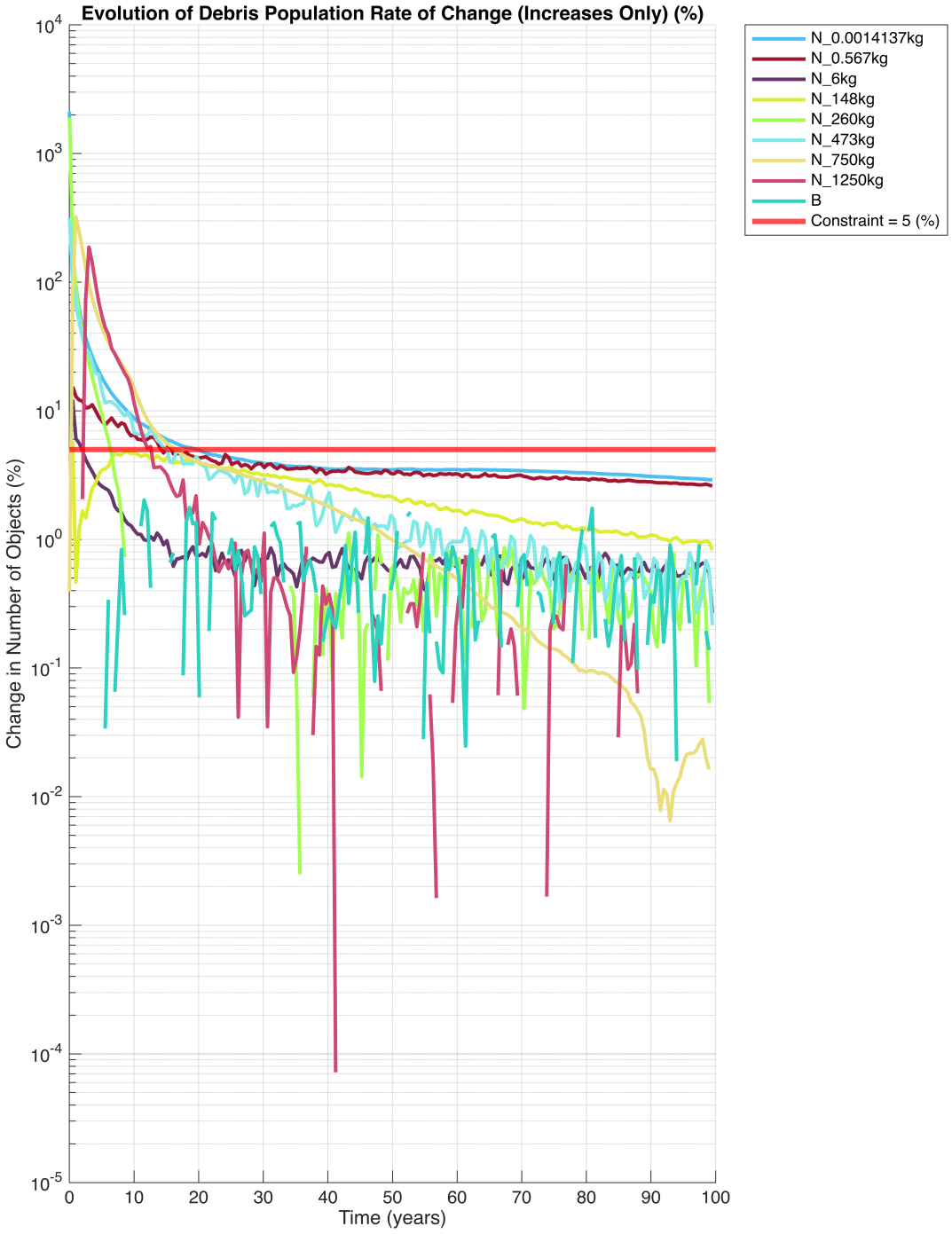
```
Producing visual of the change in inactive populations across all altitudes vs time.
N_0p0014137kg
N_0p567kg
N_6kg
N_148kg
N_260kg
N_473kg
N_750kg
N_1250kg
B
Warning: Negative data ignored
```




```
my_sim.total_deb_pop_time_deriv2("constraint", 5, "percentage", true)
```

```
Producing visual of the change in inactive populations across all altitudes vs time.
N_0p0014137kg
N_0p567kg
N_6kg
```

```
N_148kg
N_260kg
N_473kg
N_750kg
N_1250kg
B
Warning: Negative data ignored
```

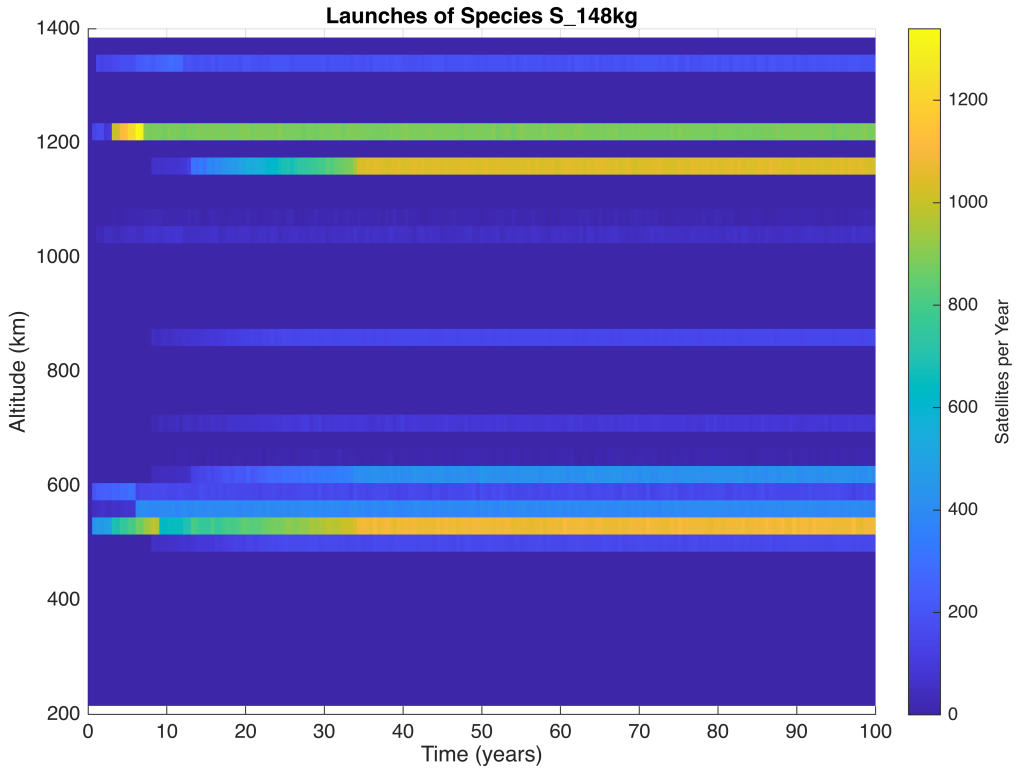


Visualize Launch Rates

```
cust_launch_funcs_species = [S_species Su_species Sns_species B_species];
alts = my_sim.scen_properties.HMid;
for i = 1:length(cust_launch_funcs_species) % For Species.
    cust_species = cust_launch_funcs_species(i);
    cust_name = cust_species.species_properties.sym_name;

    launch_data = zeros(numel(my_sim.scen_properties.N_shell),
numel(my_sim.results.T));
    launch_funcs = cust_species.species_properties.lambda_funs;

    disp("Species " + cust_name + " " + string(i) + " / " +
string(numel(cust_launch_funcs_species)))
    for j = 1:numel(alts) % For each Altitude
        disp("   Shell " + string(j) + " / " + string(numel(alts)))
        cust_species.species_properties.lambda_funs{j}(1, my_sim.results.T);
        alt_time_func = @(t) cust_species.species_properties.lambda_funs{j}
(1, t);

        if j == 1
            HandleVisibility = "on";
        else
            HandleVisibility = "off";
        end

        launch_data(j,:) =  alt_time_func(my_sim.results.T);
    end
    figure()
    hold on
    [X, Y] = meshgrid(my_sim.results.T, alts);
    surf(X, Y, launch_data,'edgecolor','none');
    xlabel('Time (years)');
    ylabel('Altitude (km)');
    zlabel('Launches Sats/Year');
    title('Launches of Species
'+strrep(cust_species.species_properties.sym_name,'p','.'), 'Interpreter',
'none');
    %legend('Location', 'southoutside');
    c = colorbar();
    c.Label.String = "Satellites per Year";
    grid on;
    hold off;
end
```

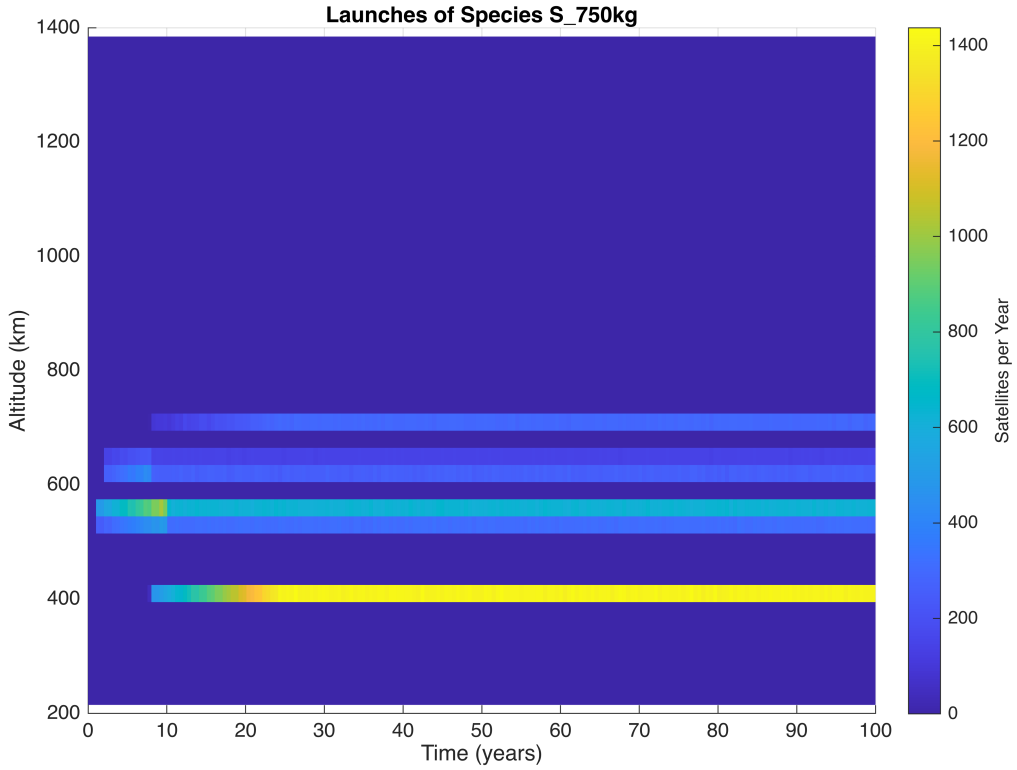
```
Species S_148kg 1 / 7
   Shell 1 / 40
   Shell 2 / 40
   Shell 3 / 40
   Shell 4 / 40
```

```
Shell 5 / 40
Shell 6 / 40
Shell 7 / 40
Shell 8 / 40
Shell 9 / 40
Shell 10 / 40
Shell 11 / 40
Shell 12 / 40
Shell 13 / 40
Shell 14 / 40
Shell 15 / 40
Shell 16 / 40
Shell 17 / 40
Shell 18 / 40
Shell 19 / 40
Shell 20 / 40
Shell 21 / 40
Shell 22 / 40
Shell 23 / 40
Shell 24 / 40
Shell 25 / 40
Shell 26 / 40
Shell 27 / 40
Shell 28 / 40
Shell 29 / 40
Shell 30 / 40
Shell 31 / 40
Shell 32 / 40
Shell 33 / 40
Shell 34 / 40
Shell 35 / 40
Shell 36 / 40
Shell 37 / 40
Shell 38 / 40
Shell 39 / 40
Shell 40 / 40
```



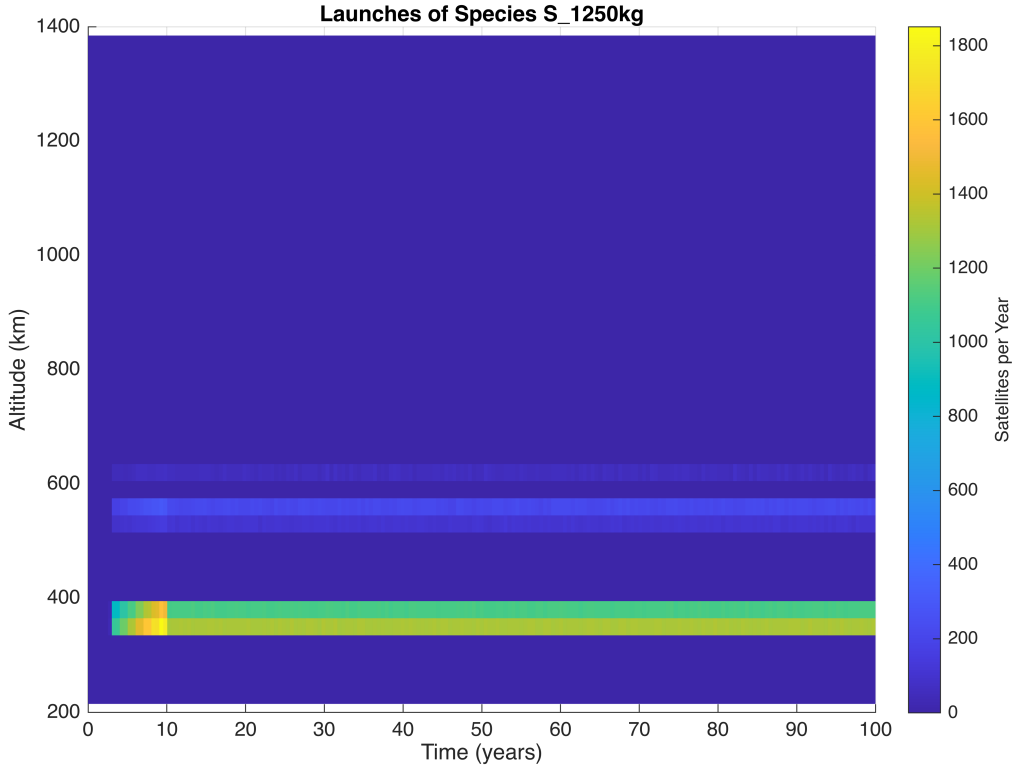

```
Species S_750kg 2 / 7
    Shell 1 / 40
    Shell 2 / 40
    Shell 3 / 40
    Shell 4 / 40
    Shell 5 / 40
    Shell 6 / 40
    Shell 7 / 40
    Shell 8 / 40
    Shell 9 / 40
    Shell 10 / 40
    Shell 11 / 40
    Shell 12 / 40
    Shell 13 / 40
    Shell 14 / 40
    Shell 15 / 40
    Shell 16 / 40
    Shell 17 / 40
    Shell 18 / 40
    Shell 19 / 40
    Shell 20 / 40
    Shell 21 / 40
    Shell 22 / 40
    Shell 23 / 40
    Shell 24 / 40
    Shell 25 / 40
    Shell 26 / 40
    Shell 27 / 40
    Shell 28 / 40
    Shell 29 / 40
    Shell 30 / 40
    Shell 31 / 40
    Shell 32 / 40
    Shell 33 / 40
```

```
   Shell 34 / 40
   Shell 35 / 40
   Shell 36 / 40
   Shell 37 / 40
   Shell 38 / 40
   Shell 39 / 40
   Shell 40 / 40
```



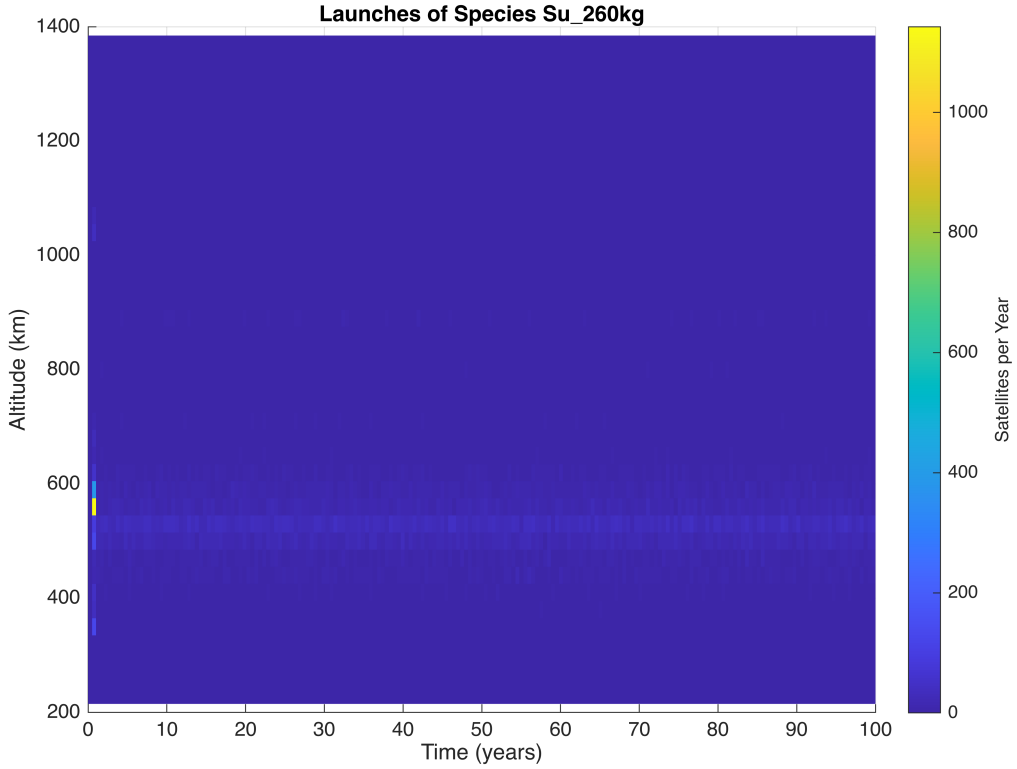

```
Species S_1250kg 3 / 7
   Shell 1 / 40
   Shell 2 / 40
   Shell 3 / 40
   Shell 4 / 40
   Shell 5 / 40
   Shell 6 / 40
   Shell 7 / 40
   Shell 8 / 40
   Shell 9 / 40
   Shell 10 / 40
   Shell 11 / 40
   Shell 12 / 40
   Shell 13 / 40
   Shell 14 / 40
   Shell 15 / 40
   Shell 16 / 40
   Shell 17 / 40
   Shell 18 / 40
   Shell 19 / 40
   Shell 20 / 40
   Shell 21 / 40
   Shell 22 / 40
   Shell 23 / 40
   Shell 24 / 40
   Shell 25 / 40
   Shell 26 / 40
```

```
   Shell 27 / 40
   Shell 28 / 40
   Shell 29 / 40
   Shell 30 / 40
   Shell 31 / 40
   Shell 32 / 40
   Shell 33 / 40
   Shell 34 / 40
   Shell 35 / 40
   Shell 36 / 40
   Shell 37 / 40
   Shell 38 / 40
   Shell 39 / 40
   Shell 40 / 40
```



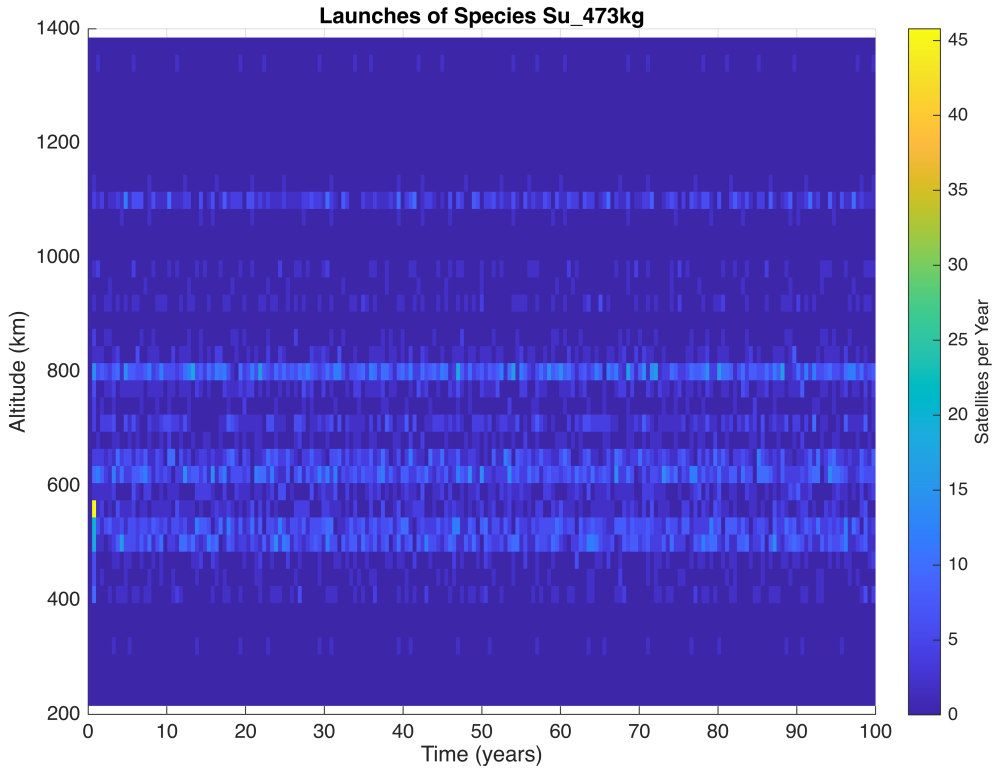

```
Species Su_260kg 4 / 7
   Shell 1 / 40
   Shell 2 / 40
   Shell 3 / 40
   Shell 4 / 40
   Shell 5 / 40
   Shell 6 / 40
   Shell 7 / 40
   Shell 8 / 40
   Shell 9 / 40
   Shell 10 / 40
   Shell 11 / 40
   Shell 12 / 40
   Shell 13 / 40
   Shell 14 / 40
   Shell 15 / 40
   Shell 16 / 40
   Shell 17 / 40
   Shell 18 / 40
   Shell 19 / 40
```

```
   Shell 20 / 40
   Shell 21 / 40
   Shell 22 / 40
   Shell 23 / 40
   Shell 24 / 40
   Shell 25 / 40
   Shell 26 / 40
   Shell 27 / 40
   Shell 28 / 40
   Shell 29 / 40
   Shell 30 / 40
   Shell 31 / 40
   Shell 32 / 40
   Shell 33 / 40
   Shell 34 / 40
   Shell 35 / 40
   Shell 36 / 40
   Shell 37 / 40
   Shell 38 / 40
   Shell 39 / 40
   Shell 40 / 40
```



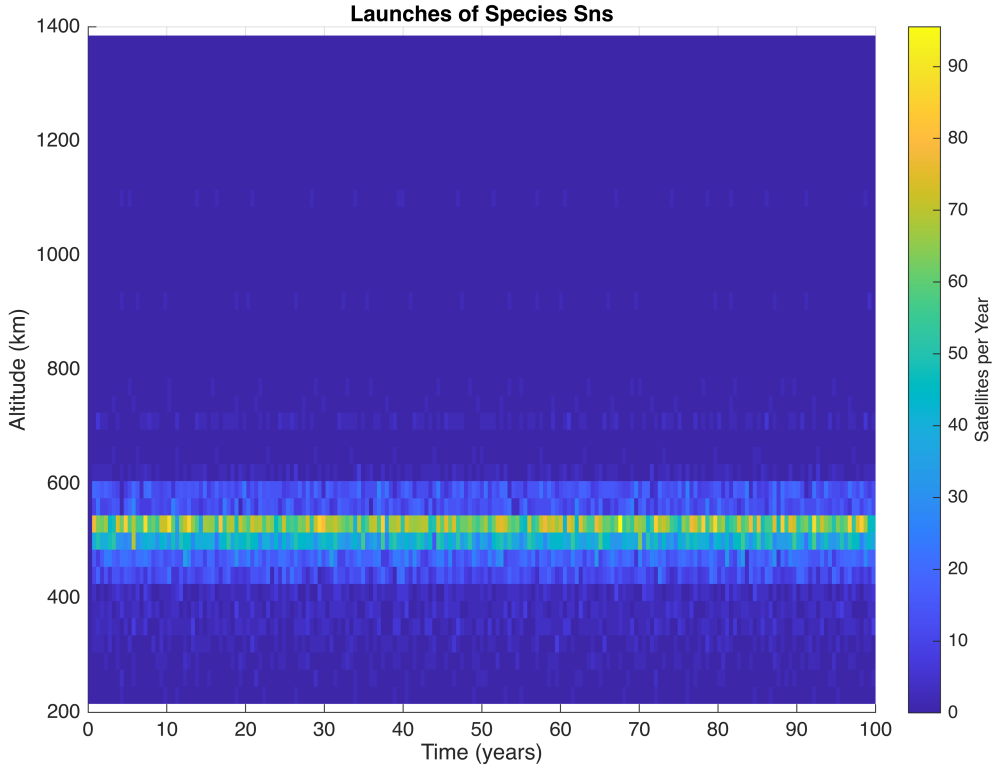

```
Species Su_473kg 5 / 7
   Shell 1 / 40
   Shell 2 / 40
   Shell 3 / 40
   Shell 4 / 40
   Shell 5 / 40
   Shell 6 / 40
   Shell 7 / 40
   Shell 8 / 40
   Shell 9 / 40
   Shell 10 / 40
   Shell 11 / 40
   Shell 12 / 40
```

```
   Shell 13 / 40
   Shell 14 / 40
   Shell 15 / 40
   Shell 16 / 40
   Shell 17 / 40
   Shell 18 / 40
   Shell 19 / 40
   Shell 20 / 40
   Shell 21 / 40
   Shell 22 / 40
   Shell 23 / 40
   Shell 24 / 40
   Shell 25 / 40
   Shell 26 / 40
   Shell 27 / 40
   Shell 28 / 40
   Shell 29 / 40
   Shell 30 / 40
   Shell 31 / 40
   Shell 32 / 40
   Shell 33 / 40
   Shell 34 / 40
   Shell 35 / 40
   Shell 36 / 40
   Shell 37 / 40
   Shell 38 / 40
   Shell 39 / 40
   Shell 40 / 40
```



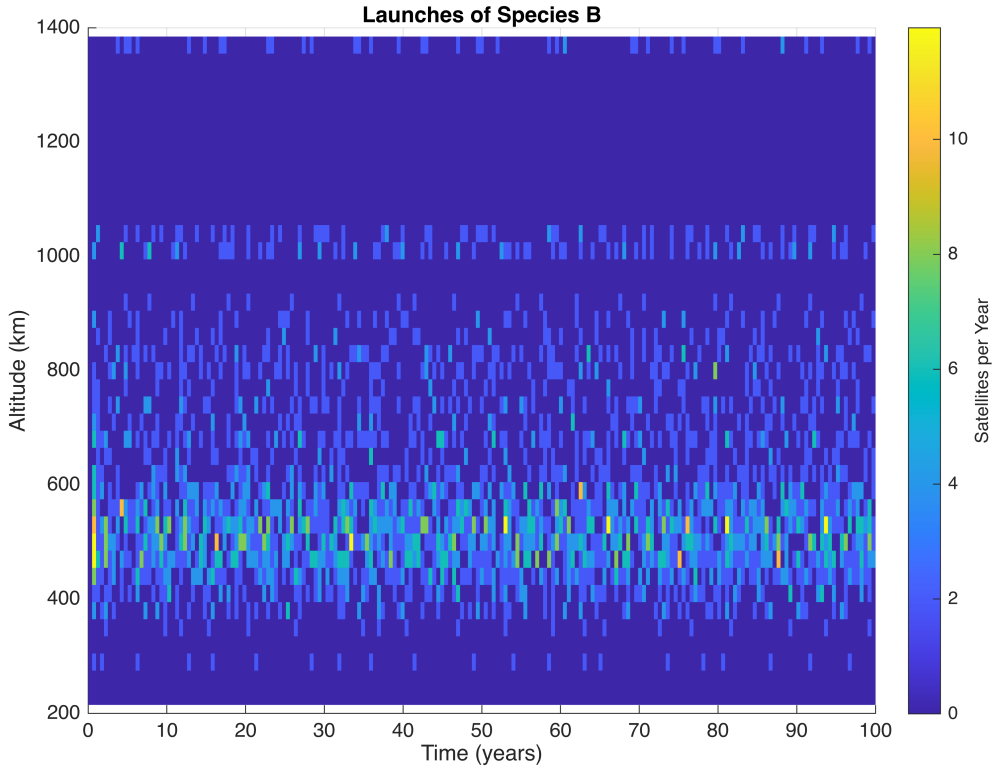

```
Species Sns 6 / 7
   Shell 1 / 40
   Shell 2 / 40
   Shell 3 / 40
   Shell 4 / 40
   Shell 5 / 40
```

```
Shell 6 / 40
Shell 7 / 40
Shell 8 / 40
Shell 9 / 40
Shell 10 / 40
Shell 11 / 40
Shell 12 / 40
Shell 13 / 40
Shell 14 / 40
Shell 15 / 40
Shell 16 / 40
Shell 17 / 40
Shell 18 / 40
Shell 19 / 40
Shell 20 / 40
Shell 21 / 40
Shell 22 / 40
Shell 23 / 40
Shell 24 / 40
Shell 25 / 40
Shell 26 / 40
Shell 27 / 40
Shell 28 / 40
Shell 29 / 40
Shell 30 / 40
Shell 31 / 40
Shell 32 / 40
Shell 33 / 40
Shell 34 / 40
Shell 35 / 40
Shell 36 / 40
Shell 37 / 40
Shell 38 / 40
Shell 39 / 40
Shell 40 / 40
```



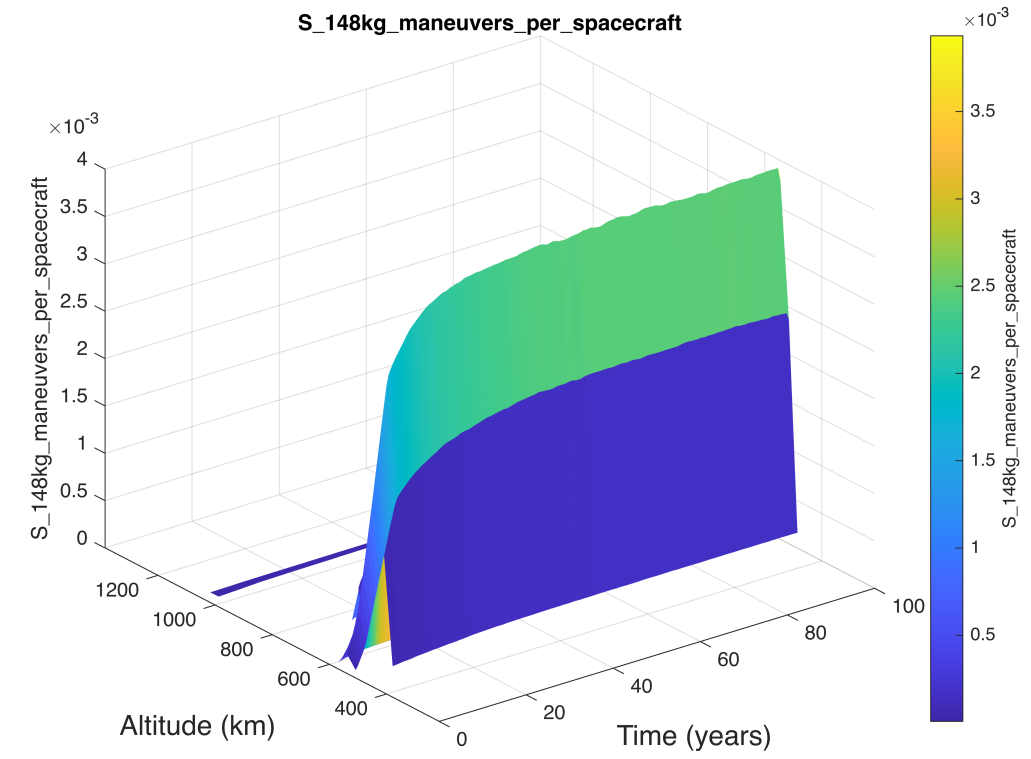

```
Species B 7 / 7
   Shell 1 / 40
   Shell 2 / 40
   Shell 3 / 40
   Shell 4 / 40
   Shell 5 / 40
   Shell 6 / 40
   Shell 7 / 40
   Shell 8 / 40
   Shell 9 / 40
   Shell 10 / 40
   Shell 11 / 40
   Shell 12 / 40
   Shell 13 / 40
   Shell 14 / 40
   Shell 15 / 40
   Shell 16 / 40
   Shell 17 / 40
   Shell 18 / 40
   Shell 19 / 40
   Shell 20 / 40
   Shell 21 / 40
   Shell 22 / 40
   Shell 23 / 40
   Shell 24 / 40
   Shell 25 / 40
   Shell 26 / 40
   Shell 27 / 40
   Shell 28 / 40
   Shell 29 / 40
   Shell 30 / 40
   Shell 31 / 40
   Shell 32 / 40
   Shell 33 / 40
```

```
Shell 34 / 40
Shell 35 / 40
Shell 36 / 40
Shell 37 / 40
Shell 38 / 40
Shell 39 / 40
Shell 40 / 40
```

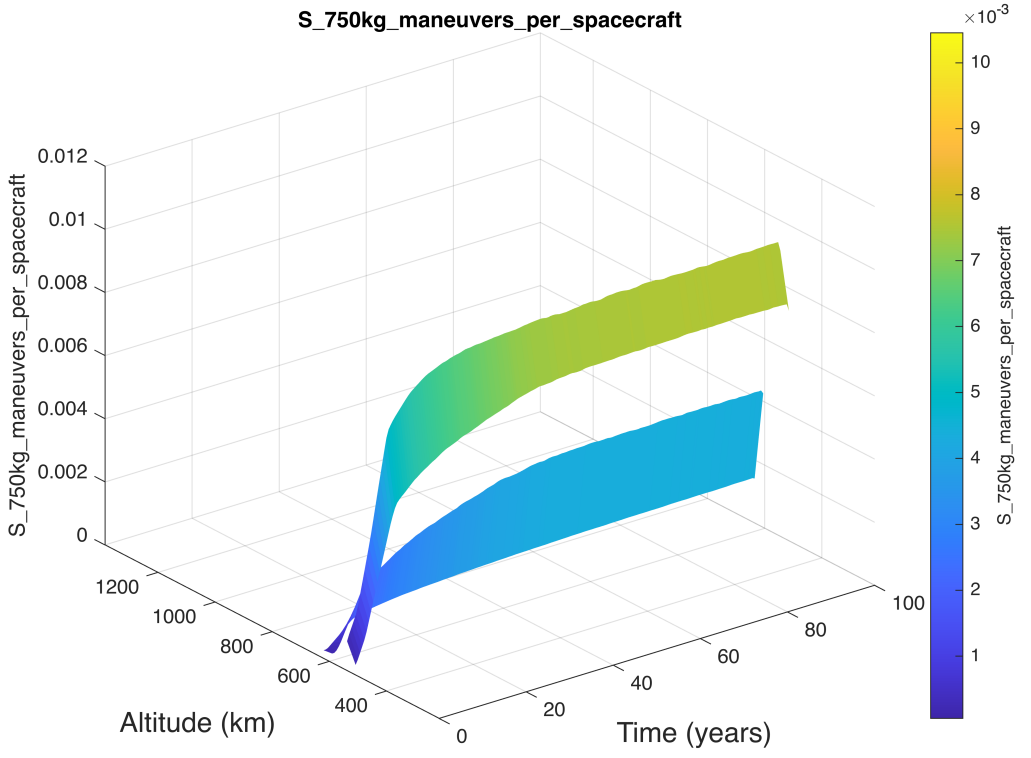



```
for ii = 1:length(scen_properties.indicator_var_list)
    if contains(scen_properties.indicator_var_list(ii).name, "maneuvers")
        disp(scen_properties.indicator_var_list(ii).name)

my_sim.per_shell_indicator(scen_properties.indicator_var_list(ii).name,
'constraint_type', 'upper')
    end
end
```

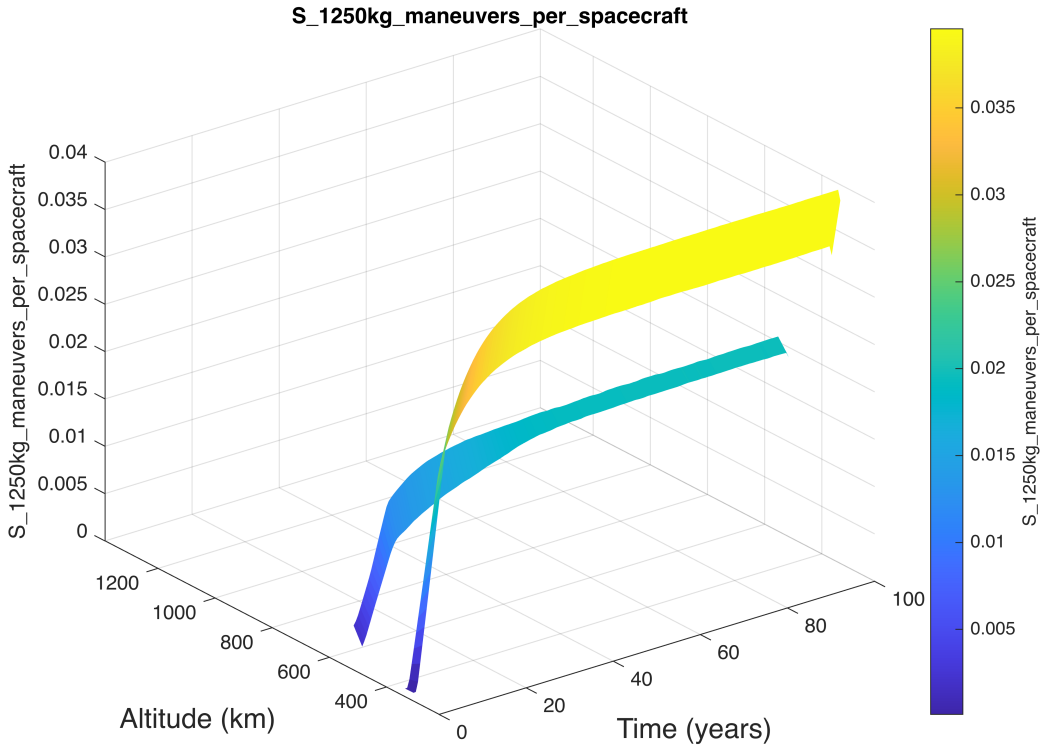
```
S_148kg_maneuvers_per_spacecraft
Producing visuals for the evolution of indicator.
```



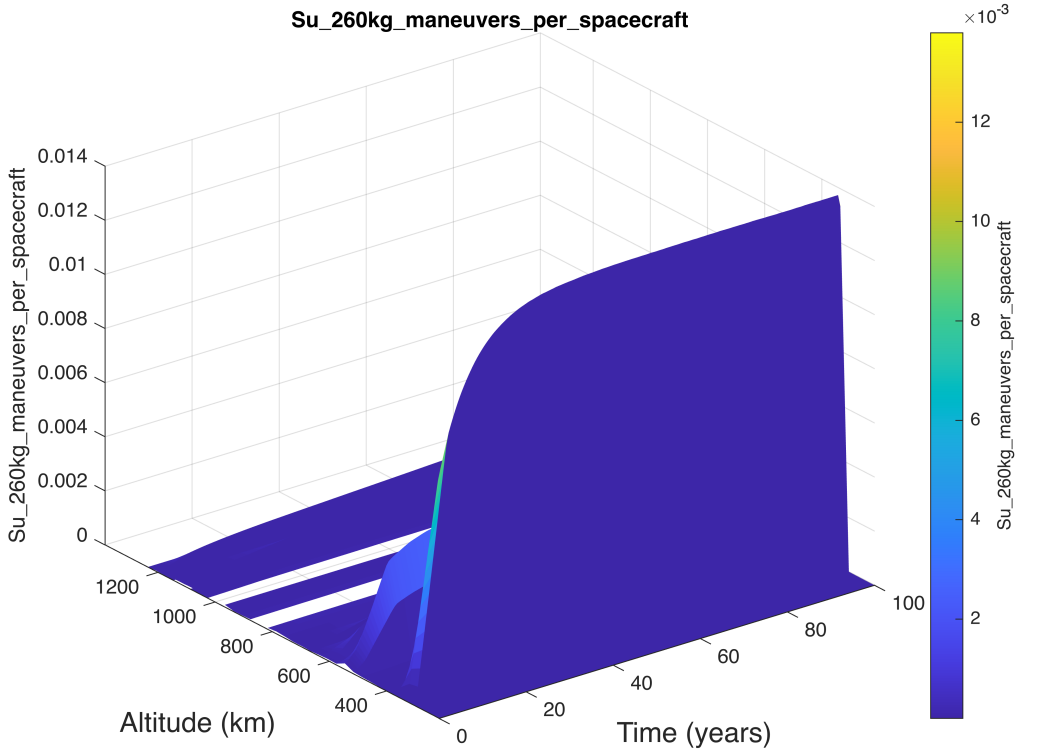
S_750kg_maneuvers_per_spacecraft
Producing visuals for the evolution of indicator.



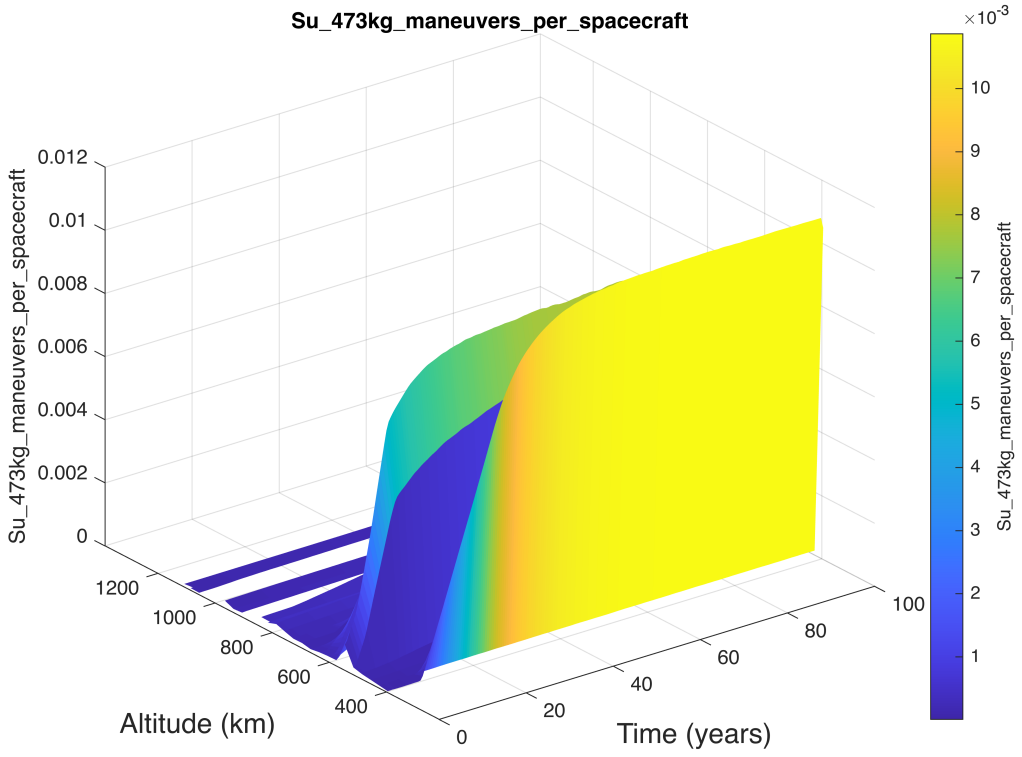
S_1250kg_maneuvers_per_spacecraft
Producing visuals for the evolution of indicator.



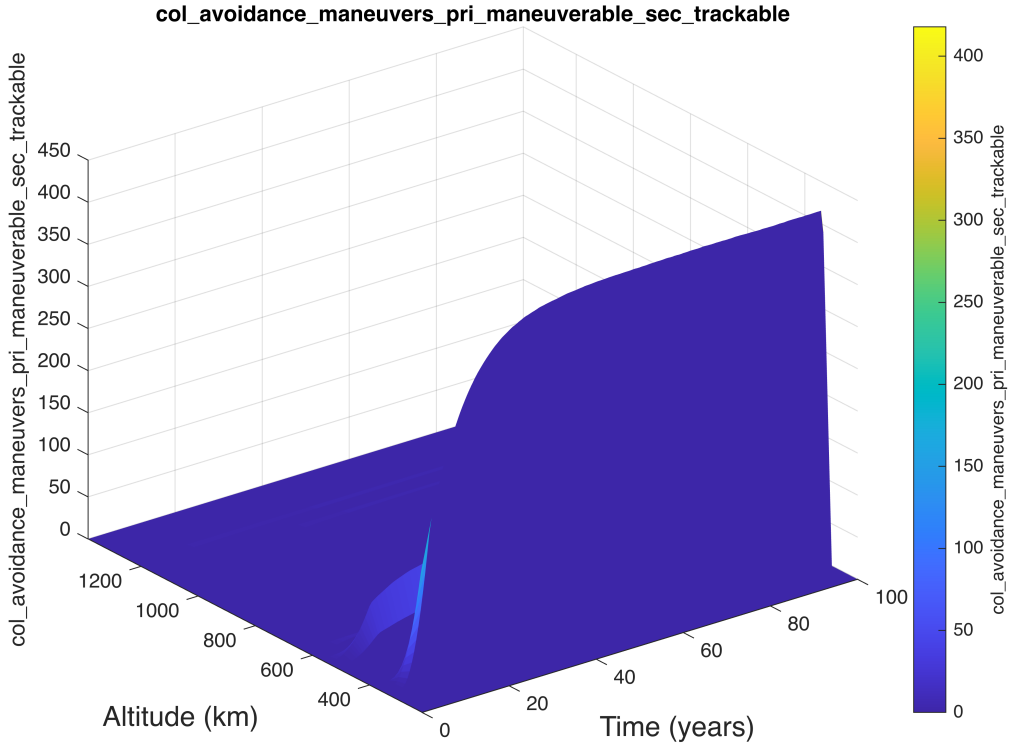

```
Su_260kg_maneuvers_per_spacecraft
Producing visuals for the evolution of indicator.
```




```
Su_473kg_maneuvers_per_spacecraft
Producing visuals for the evolution of indicator.
```

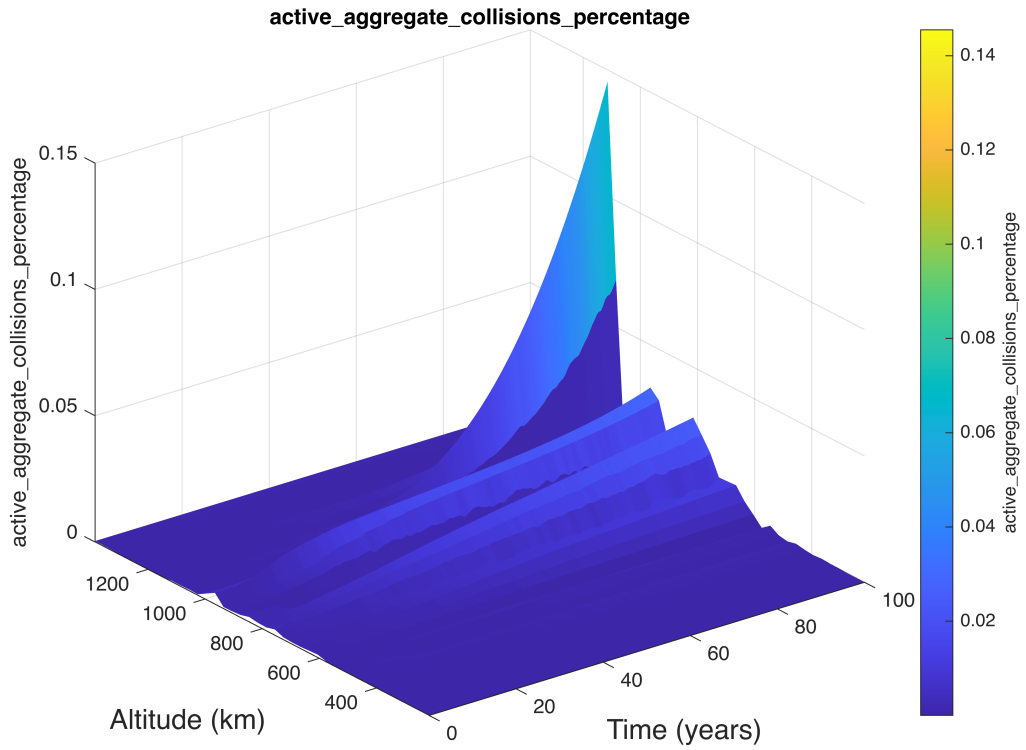



```
col_avoidance_maneuvers_pri_maneuverable_sec_trackable
Producing visuals for the evolution of indicator.
```

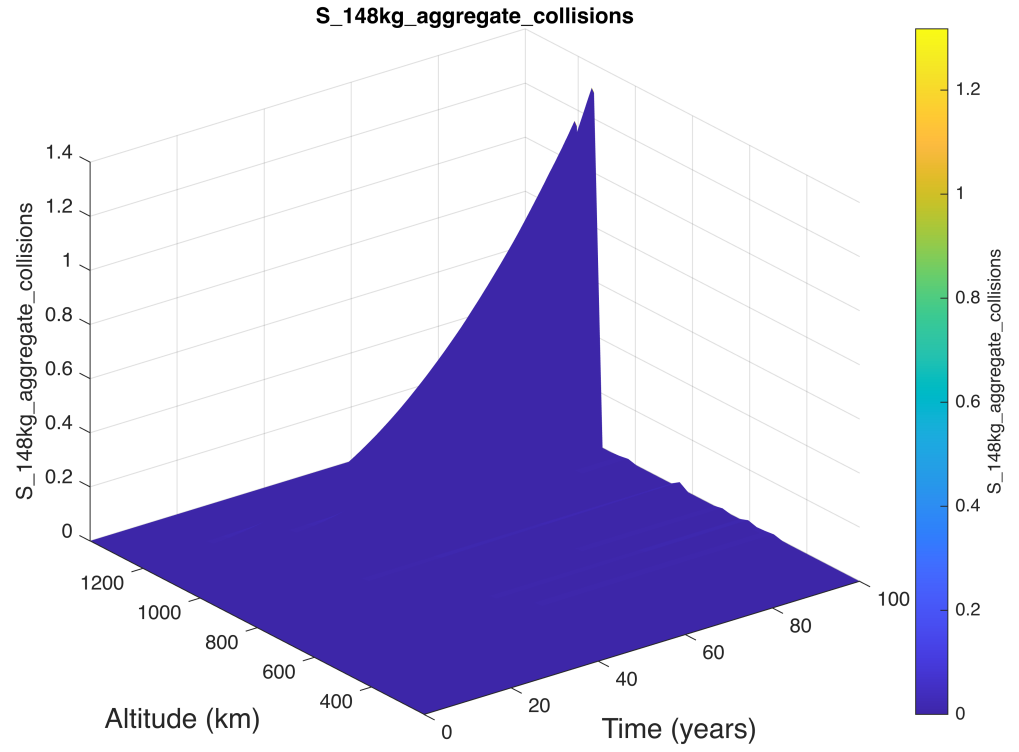


```
for ii = 1:length(scen_properties.indicator_var_list)
    if contains(scen_properties.indicator_var_list(ii).name, "collisions")
        disp(scen_properties.indicator_var_list(ii).name)

my_sim.per_shell_indicator(scen_properties.indicator_var_list(ii).name,
'constraint_type', 'upper')
    end
end
```

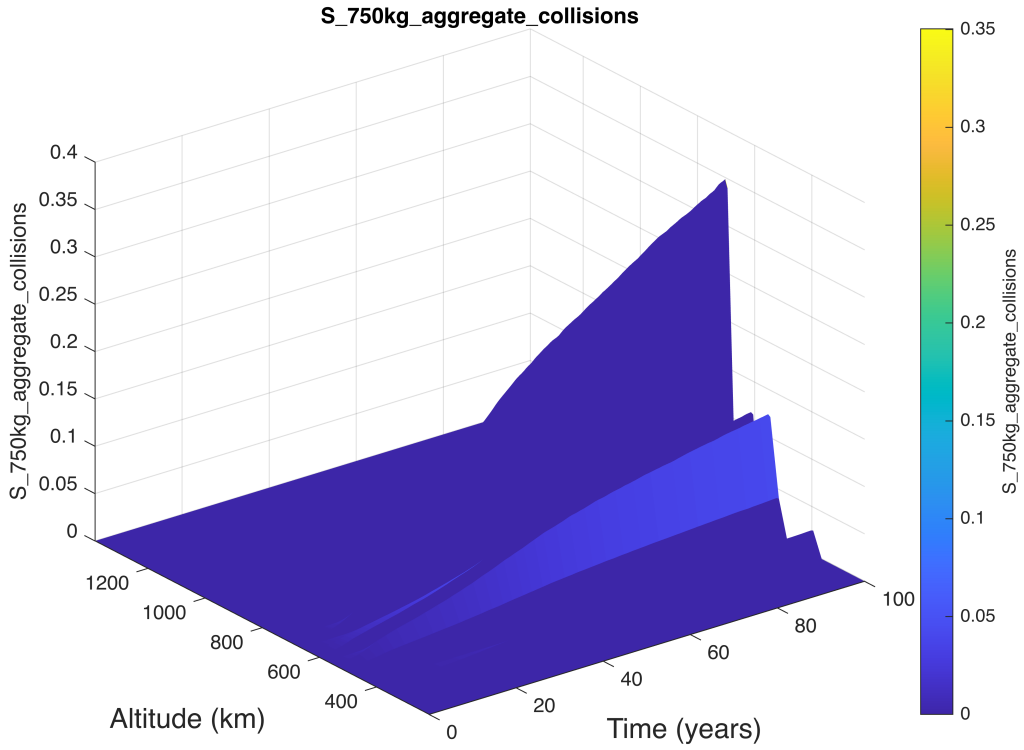
```
active_aggregate_collisions_percentage
Producing visuals for the evolution of indicator.
```



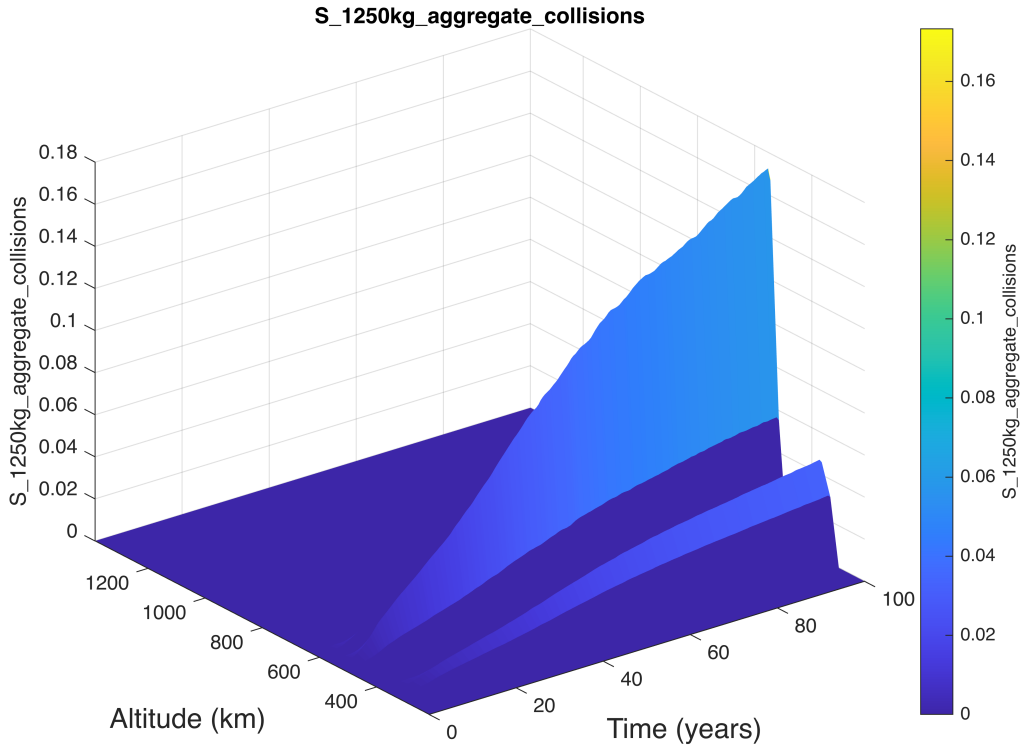

```
S_148kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



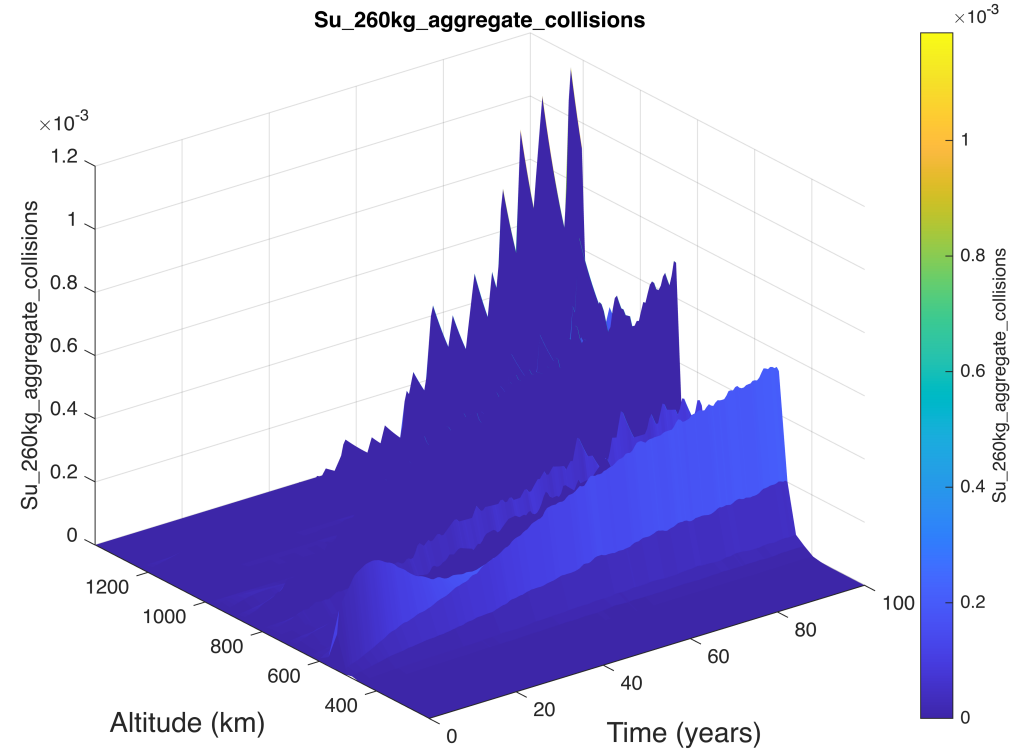

```
S_750kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



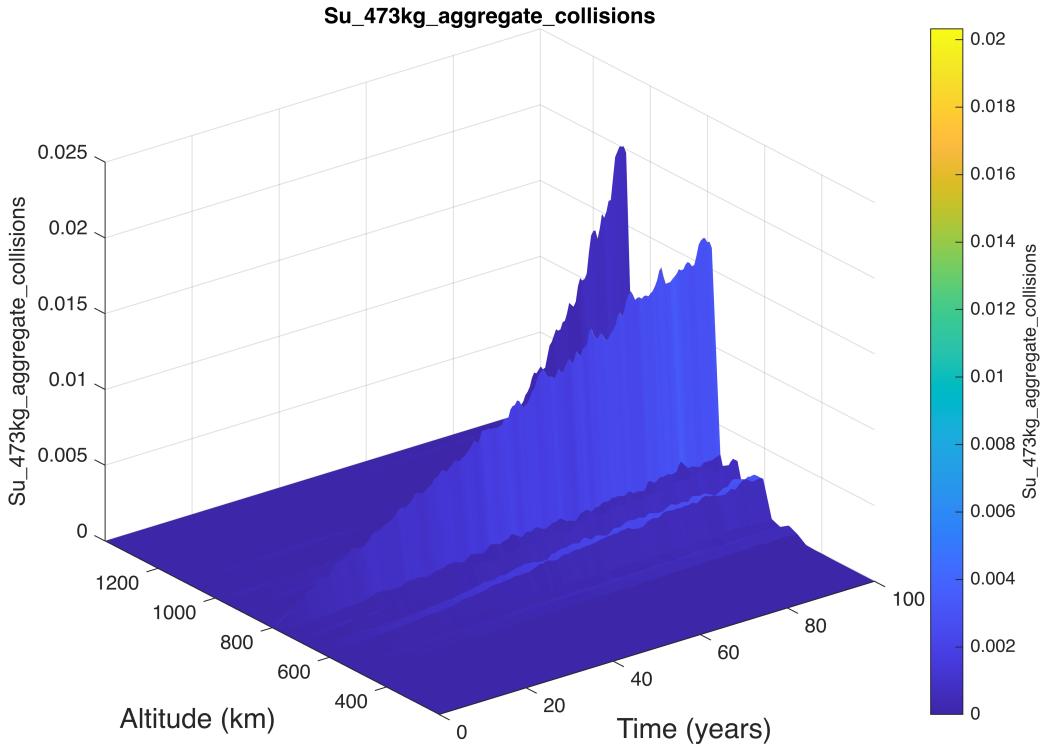

```
S_1250kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



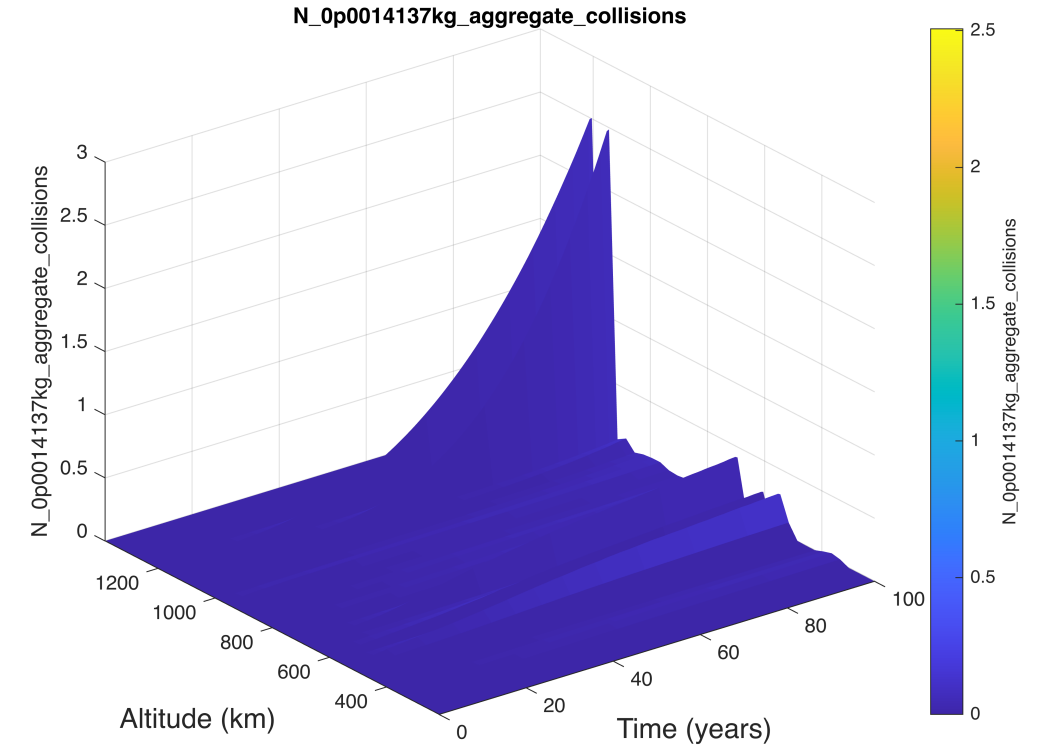

```
Su_260kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



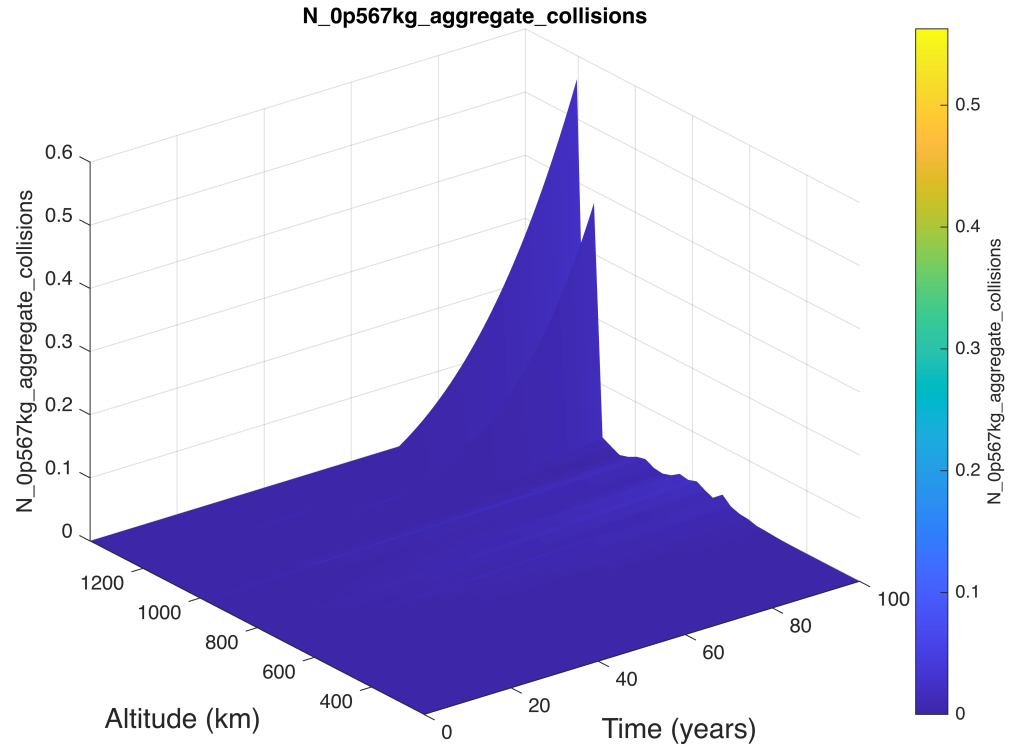

```
Su_473kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



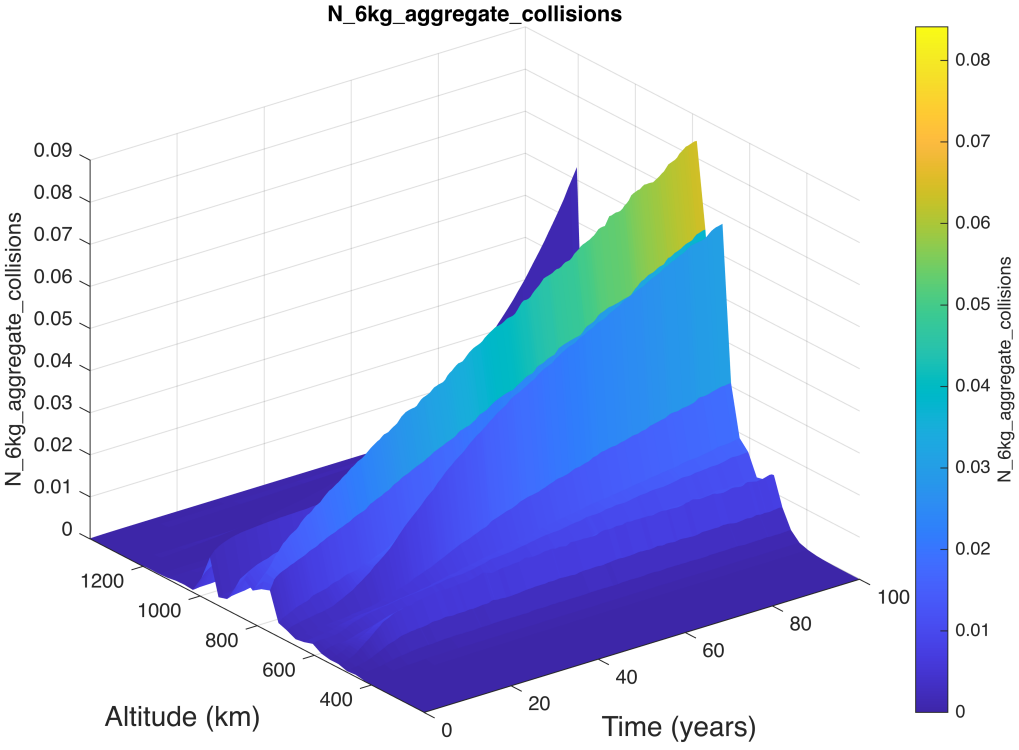

```
N_0p0014137kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



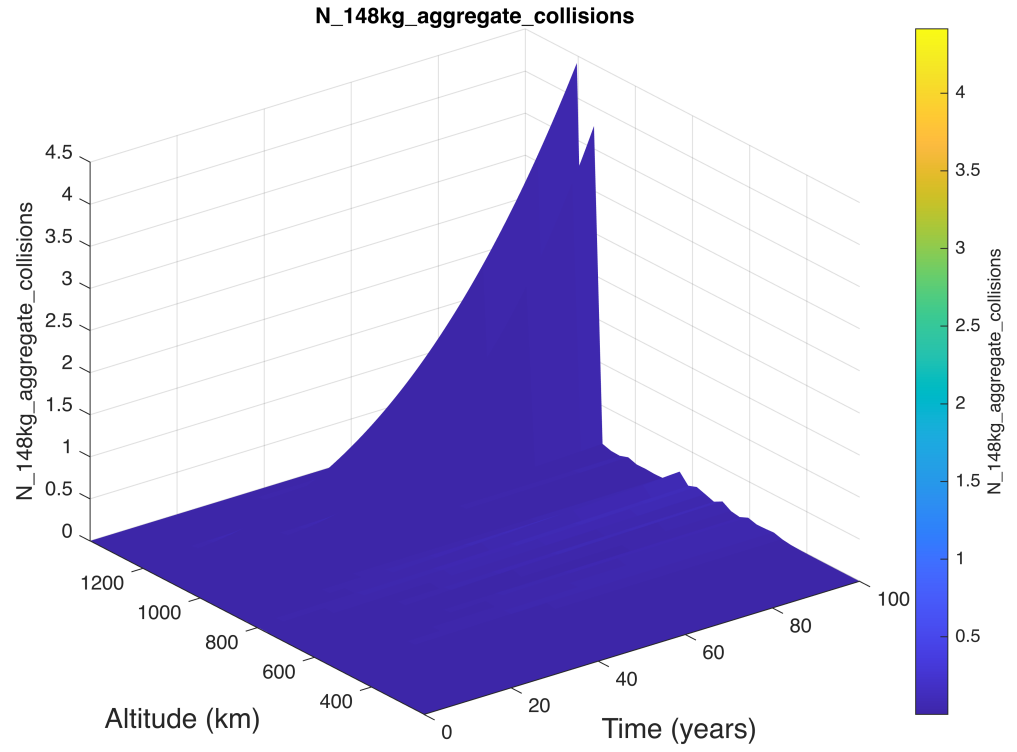

```
N_0p567kg_aggregate_collisions
Producing visuals for the evolution of indicator.
```



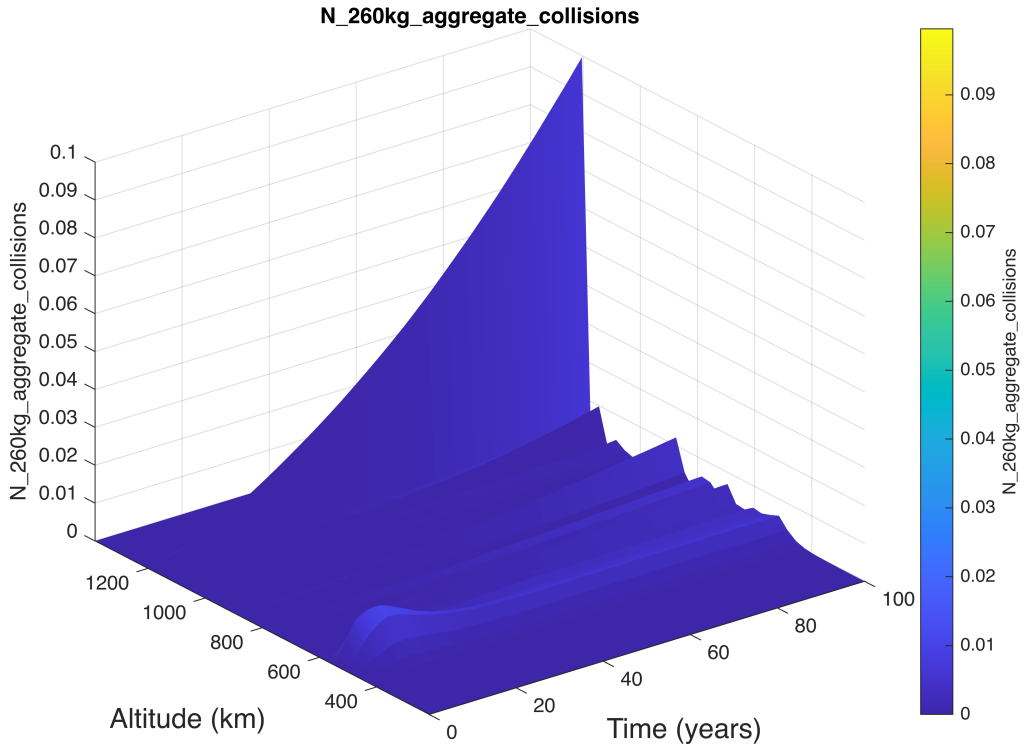
N_6kg_aggregate_collisions
Producing visuals for the evolution of indicator.



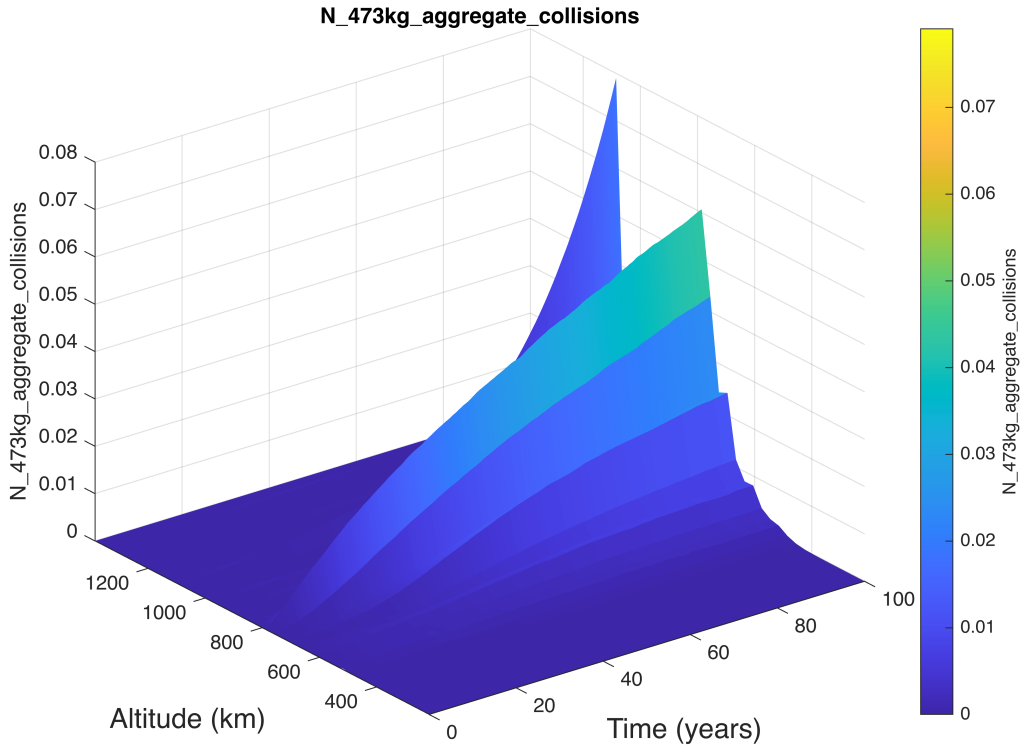
N_148kg_aggregate_collisions
Producing visuals for the evolution of indicator.



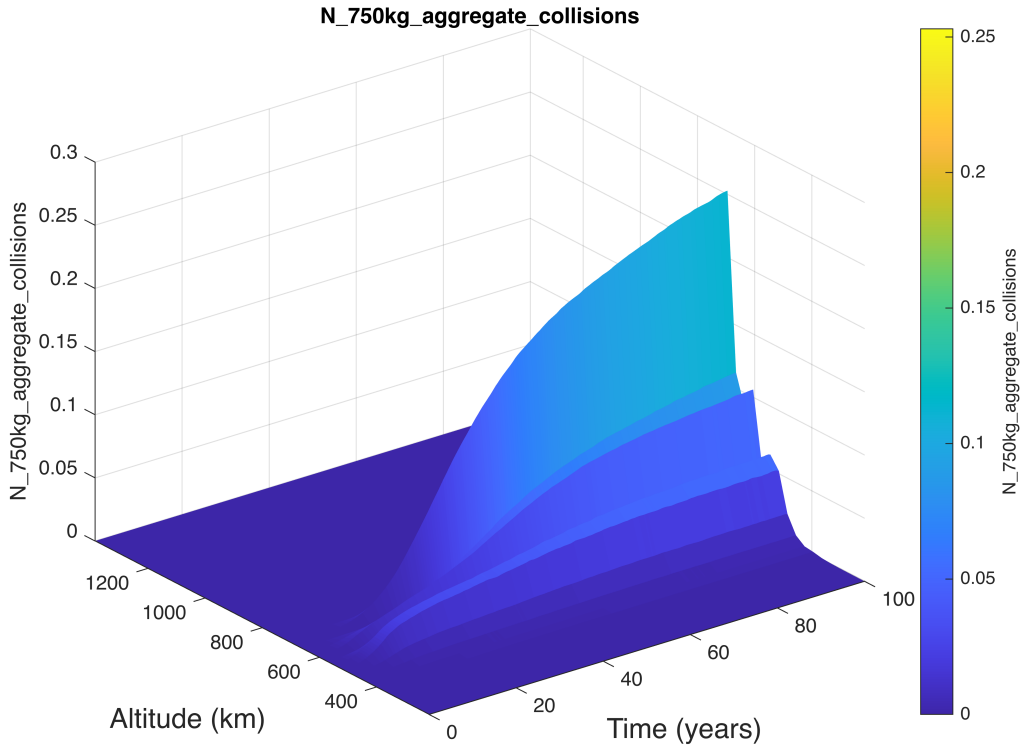
N_260kg_aggregate_collisions
Producing visuals for the evolution of indicator.



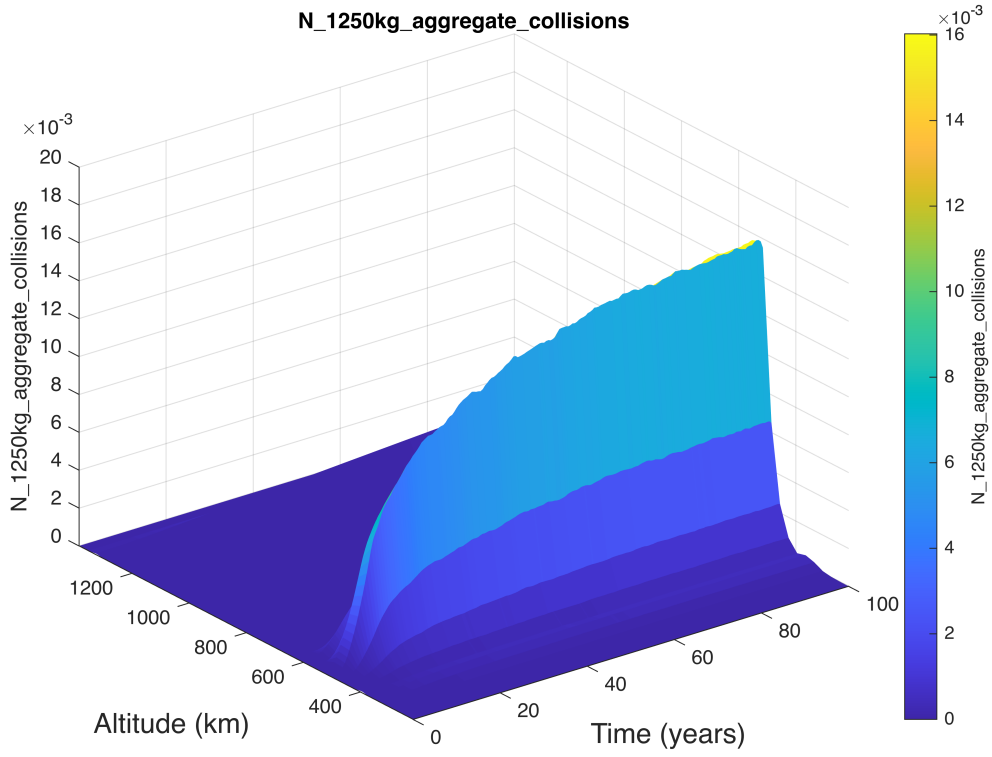
N_473kg_aggregate_collisions
Producing visuals for the evolution of indicator.



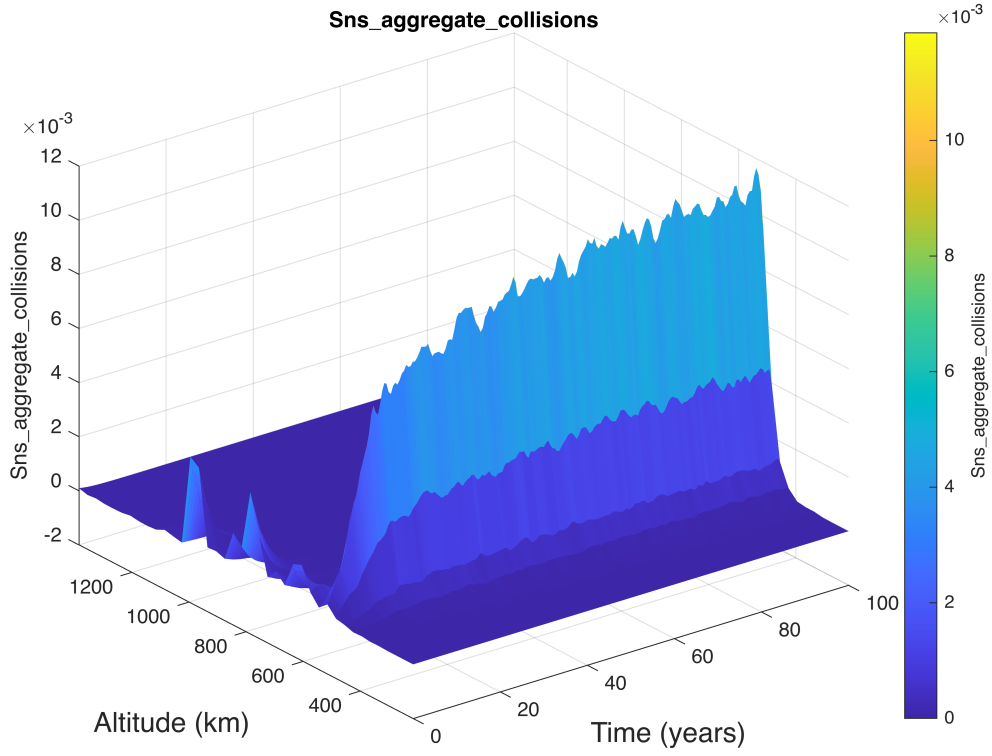
N_750kg_aggregate_collisions
Producing visuals for the evolution of indicator.



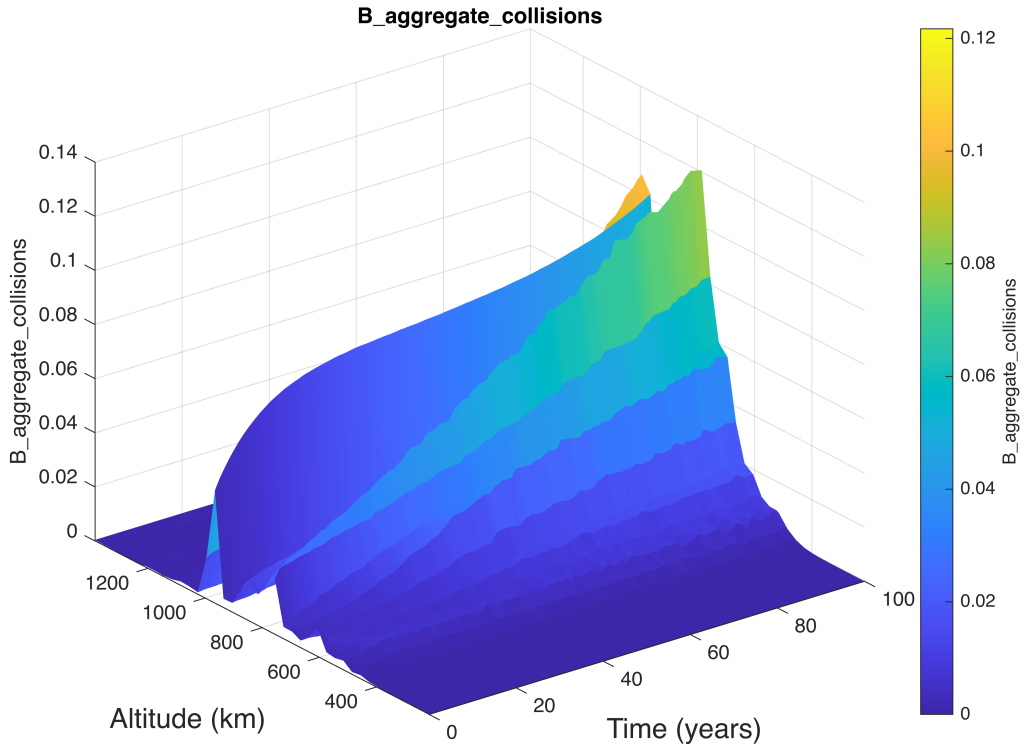
N_1250kg_aggregate_collisions
Producing visuals for the evolution of indicator.



Sns_aggregate_collisions
Producing visuals for the evolution of indicator.



B_aggregate_collisions
Producing visuals for the evolution of indicator.




```
Warning: Negative data ignored
Warning: Negative data ignored
```

Close all figures created (or run this line in your command line)

```
% close all
```