This discrete-sectional code was originally developed by the authors under the guidance of Prof. Peter McMurry, using the discrete-sectional framework developed by Dr. Fred Gelbard and Dr. Nagaraja Rao. For any inquiries, suggestions or bug report, the reader is welcome to contact

Chenxi Li, chenxi20@sjtu.edu.cn

Runlong Cai, runlong.cai@helsinki.fi

A brief outline of how to use the code

- **Step 1**: The user specifies simulation conditions in *simulationEntry.m*
- **Step 2**: The program calls *refinemodelparams.m* and use user input to calculate other parameters used in the simulation.
- **Step 3**: The program enters *main.m*
  - Step 3.1: if the user chooses to calculate rate constants (i.e. the user has set `CAL_COEFFS` and `CAL_SINK_PARAMS` to true in *simulationEntry.m*), the program calls *makecollisionkernel.m*, *calcoagcoeffs.m*, *calsink.m*, *calevap.m*. The calculated rate constants are stored in the 'coefficients' folder.
  - Step 3.2: the program calls *GDEqns.m* in which the general dynamic equations are written
- **Step 4**: The program stores the simulation data in the 'results' folder.
- **Step 5**: The user does post analysis of the results. Go to the results folder for sample post analysis files.

Further instructions for test cases

There are in total six test cases prepared by the authors. Each of the test case is located in a separate folder with a full copy of the code. The rate constants and simulation results have been precalculated and are stored in the 'coefficients' and 'results' folder. If you change a parameter in the simulation that affects rate constants, or simply want to recalculate the rate constants to see how the program works, set `CAL_COEFFS` and `CAL_SINK_PARAMS` to 'true' in *simulationEntry.m*.

Test case 1:

This test case is designed to make sure that the GDE is properly constructed in our code. Since mass should be balanced in any condition, there is great flexibility in setting the simulation parameters in this test. For convenience, we set the external particle loss rates to zero, i.e. set `wLoss`, `svgLoss` and `dilution` to zero. We do so because the external particle loss rates by the whole system are in general hard to know. Additionally, we set `CONST_MONOMER` to false.

The reader can change other parameters in the model, i.e. change simulation time, use different collision kernels, and the total system mass should increase with the monomer production rate `RR`.

(Note: set the simulation domain large enough so that the largest particle do not go out of the simulation domain.)

Test case 2:

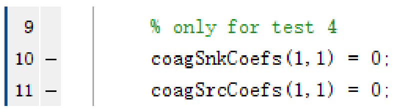
In the second test we are trying to examine the effect of bin width on simulation results, so the key parameter to change is `binsPer2`, i.e. how many bins it takes to double bin width ($\kappa = 2^{\frac{1}{\text{binsPer2}}}$). In the simulation entry scripts a-d and z, `binsPer2` are set to 5, 10, 20, 40 and 2, respectively. Note that each time you reset `binsPer2`, the rate constants need to be recalculated. This is because every time the bin configurations are changed, the matrix that contains the rate constants will change accordingly. The settings of other parameters are very flexible, as long as they are kept the same when the reader is trying to determine the effect of bin width.

Test case 3:
In this test, the parameter must be set in a way to achieve the asymptotic solution, i.e.

- `KERNEL` must be set to 'uniform'
- `RR` needs to be set to a non-zero value
- External particle loss, evaporation need to be turned off, i.e. `wLoss,` `svgLoss,` `dilution,` `satPressure` must be zero

Test case 4:
In this test, the parameter must be set in a stringent way to obtain the Poisson distribution.

- `KERNEL` must be set to 'uniform'
- `CONST_MONOMER` and `COAG_OFF` must be set to true
- Initial concentration of monomer and a cluster must be set to non-zero values (if they are set to zero, the clusters will not grow or there are no clusters to grow)
- External particle loss, evaporation need to be turned off, i.e. `wLoss`, `svgLoss`, `dilution,` `satPressure` must be zero
- You can choose a large `ND` and set `NS` to zero so that there are only discrete bins in the simulation domain
- This is the only test case where the Matlab functions other than the entry script is modified. In *GDEqns.m*, we added the following lines to prevent monomer-monomer collisions.



```
% only for test 4
coagSnkCoefs(1,1) = 0;
coagSrcCoefs(1,1) = 0;
```

Test case 5:
To achieve the self-preserving distribution,

- set `KERNEL` to 'free'
- set `RR` to 0
- set a non-zero initial concentration for monomers,
- External particle loss, evaporation need to be turned off, i.e. `wLoss`, `svgLoss`, `dilution`, `satPressure` must be zero

Test case 6:
Entry scripts 6a-6d contain parameter settings for the a) the collision-controlled limit, b) non-zero

wall loss rate c) non-zero evaporation rate, d) non-zero wall loss and evaporation rate. The reader can play with the values of `wLoss`, `satPressure` to better understand the effect of wall loss and evaporation. Another way to modify evaporation is to change `surfaceTension`, the bulk surface tension of the nucleating materials.