ISSAI

NAZARBAYEV UNIVERSITY | Institute of Smart Systems and Artificial Intelligence

A Particle-based COVID-19 Epidemic Simulator

Aknur Karabay (aknur.karabay@nu.edu.kz),
Askat Kuzdeuov, Madina Abdrakhmanova and Huseyin Atakan Varol.

25 June 2021, Friday

www.issai.nu.edu.kz

Particle Model

In our simulator, an individual is modelled as particle p with the following parameters:

$$p = [x, v, e, t, a, ts, ag, vs, hs]$$

where x – position of the particle on the 2D map,

v – the particle velocity,

e – the epidemic state of the particle,

t – the time of the particle in the current epidemic state,

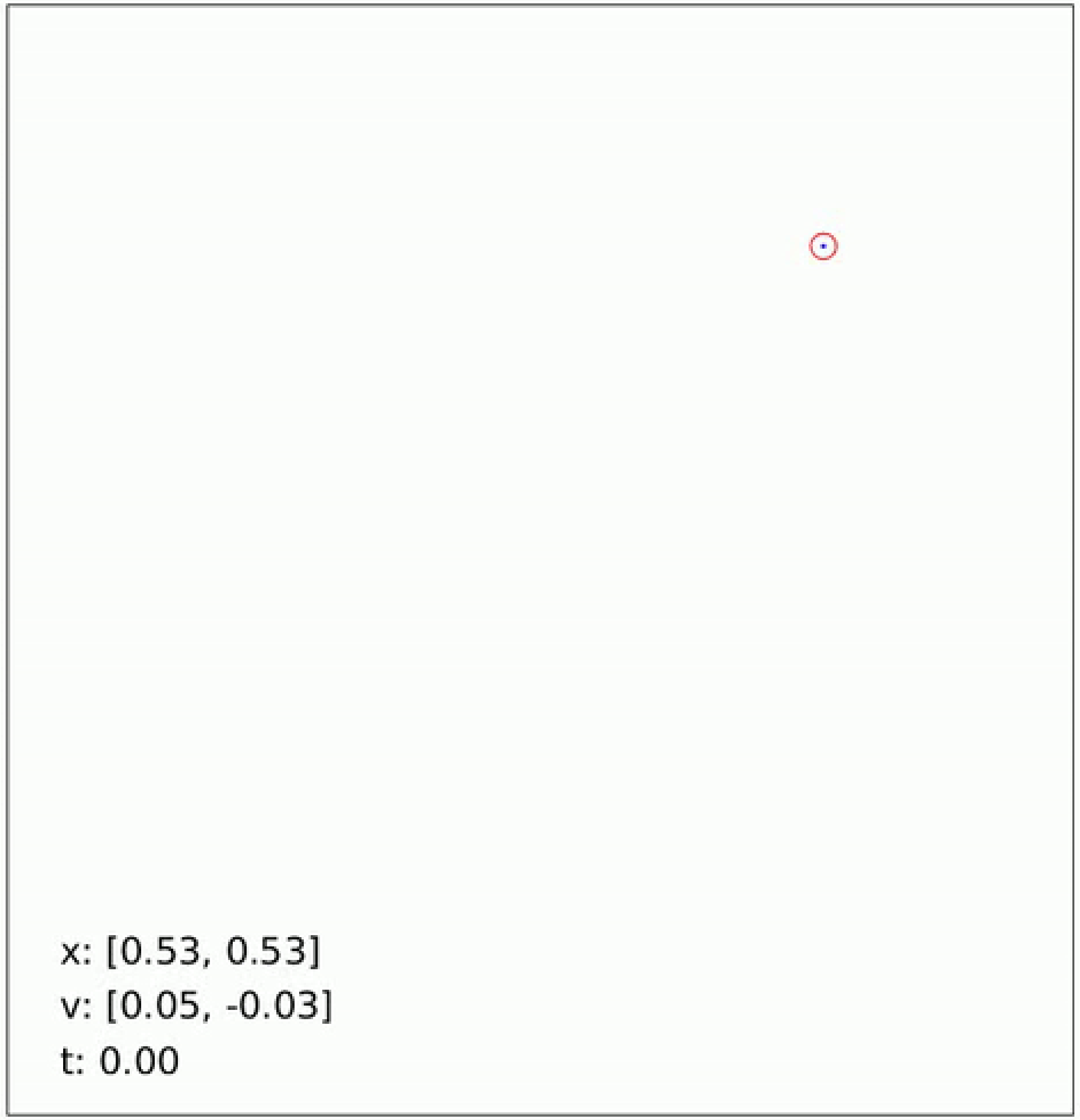
a – the availability of the contact tracing application,

ts – COVID-19 test result,

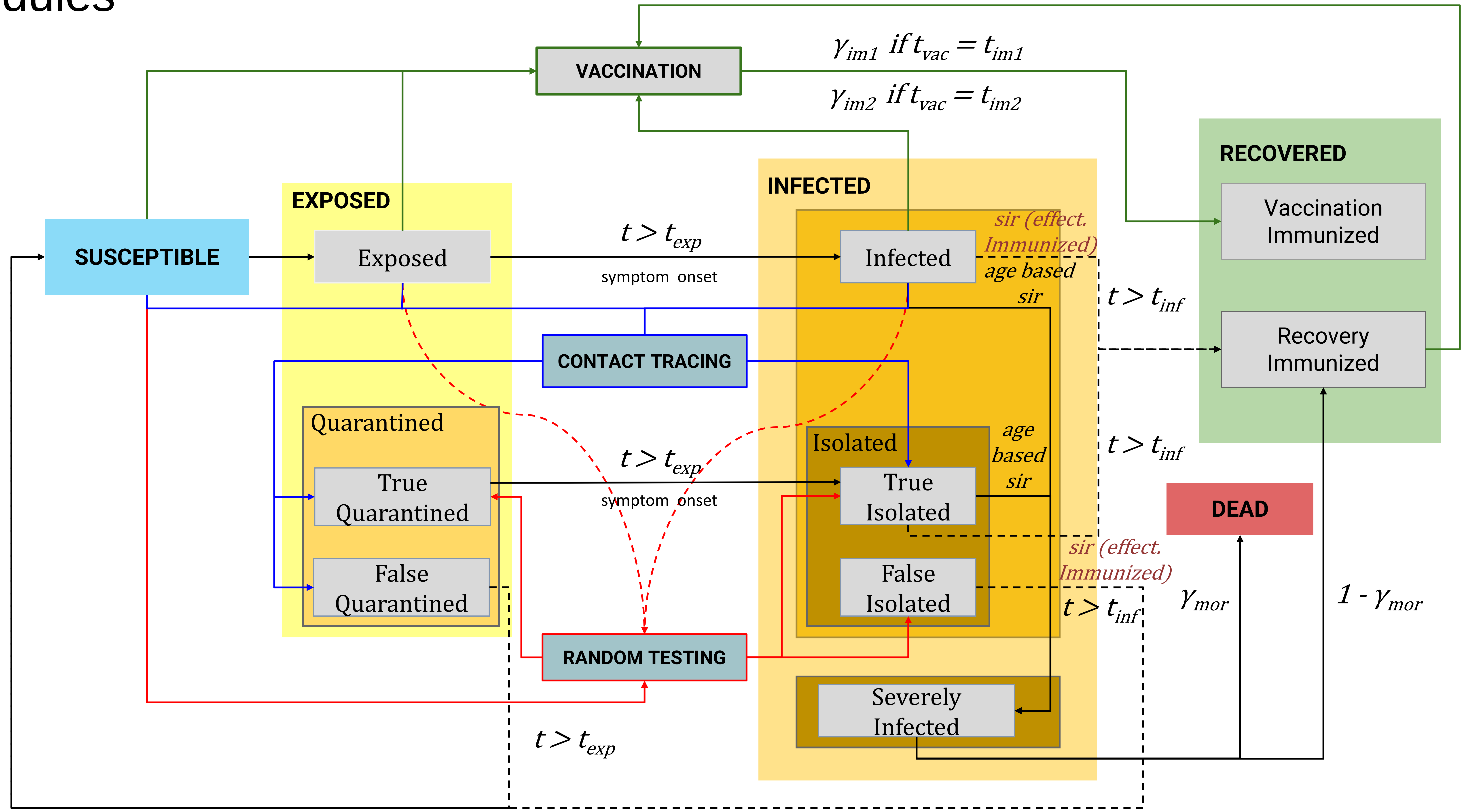
ag – the age group,

vs – vaccination status of the particle

hs – vaccination hesitancy of the particle.



Statechart of the Particle-based Simulator with Contact Tracing, Testing and Vaccination modules



A. Kuzdeuov *et al.*, "A Network-Based Stochastic Epidemic Simulator: Controlling COVID-19 With Region-Specific Policies," in *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 10, pp. 2743-2754, Oct. 2020, doi: 10.1109/JBHI.2020.3005160.

A.Kuzdeuov, A. Karabay, D. Baimukashev, B. Ibragimov, and H. A. Varol, “Particle-based covid-19 simulator with contact tracing and testing,” medRxiv, 2020. [Online]. Available: https://www.medrxiv.org/content/early/2020/12/08/2020.12.07.20245043

A.Karabay, A. Kuzdeuov, M. Lewis and H.A. Varol, “A Vaccination Simulator for COVID-19: Effective and Sterilizing immunization cases”, medRxiv, 2021. [Online]. Available: https://www.medrxiv.org/content/10.1101/2021.03.28.21254468v1.full

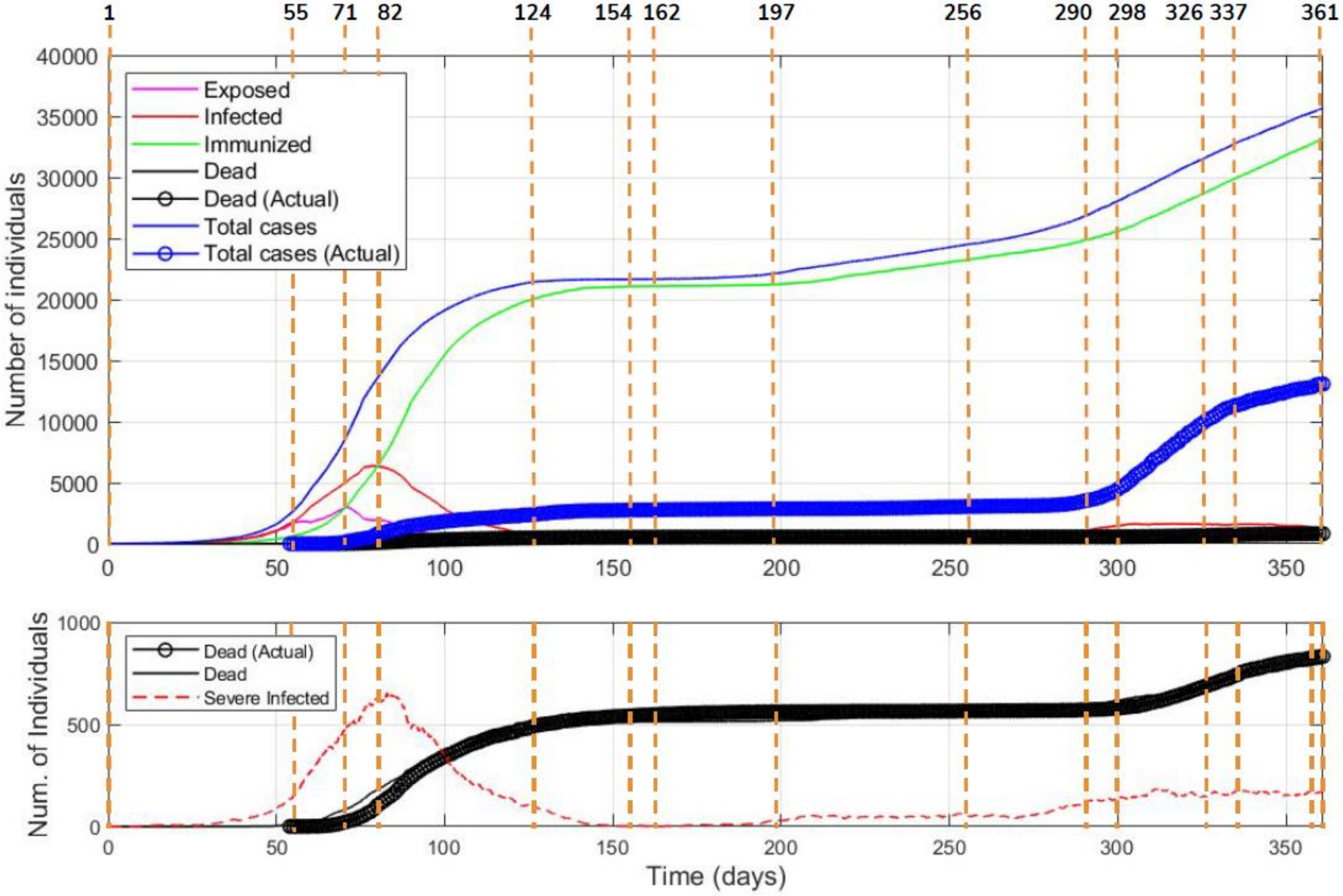
Link to the Github repository for the original Matlab source code files:

https://github.com/IS2AI/Particle-Based-COVID19-Simulator

Calibration model for province of Lecco, Italy.

TABLE IV: Major events and NPIs in Lecco province during the COVID-19 epidemic [25]

Day	Date	Event
0	1/1/2020	Start of the simulation.
55	24/2/2020	The COVID-19 data repository was launched [22].
71	11/3/2020	Lockdown in the province. Bars, restaurants are closed.
82	22/3/2020	Factories and all nonessential productions are closed.
124	4/5/2020	Easing lockdown between the regions.
154	3/6/2020	Unrestricted travel is allowed.
162	11/6/2020	International flights in Milan are resumed.
197	15/7/2020	International borders are closed. Restrictions are back.
256	14/09/2020	Schools are opened.
290	17/10/2020	Hybrid teaching in school. Food service restrictions.
298	25/10/2020	Nationwide restrictions. Night-time curfew [26].
326	23/11/2020	Lombardy still in red zone.
337	3/12/2020	Classified as very high risk region. Strict regime until 15 January [27].
361	27/12/2020	Start of vaccination. End of the simulation.



Particle Model

In our simplified model, an individual is modelled as particle $\boldsymbol{p}$ with the following parameters:

$$\boldsymbol{p} = [\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{e}, \boldsymbol{t}, \boldsymbol{ag}]$$

where $\boldsymbol{x}$ – position of the particle on the 2D map,

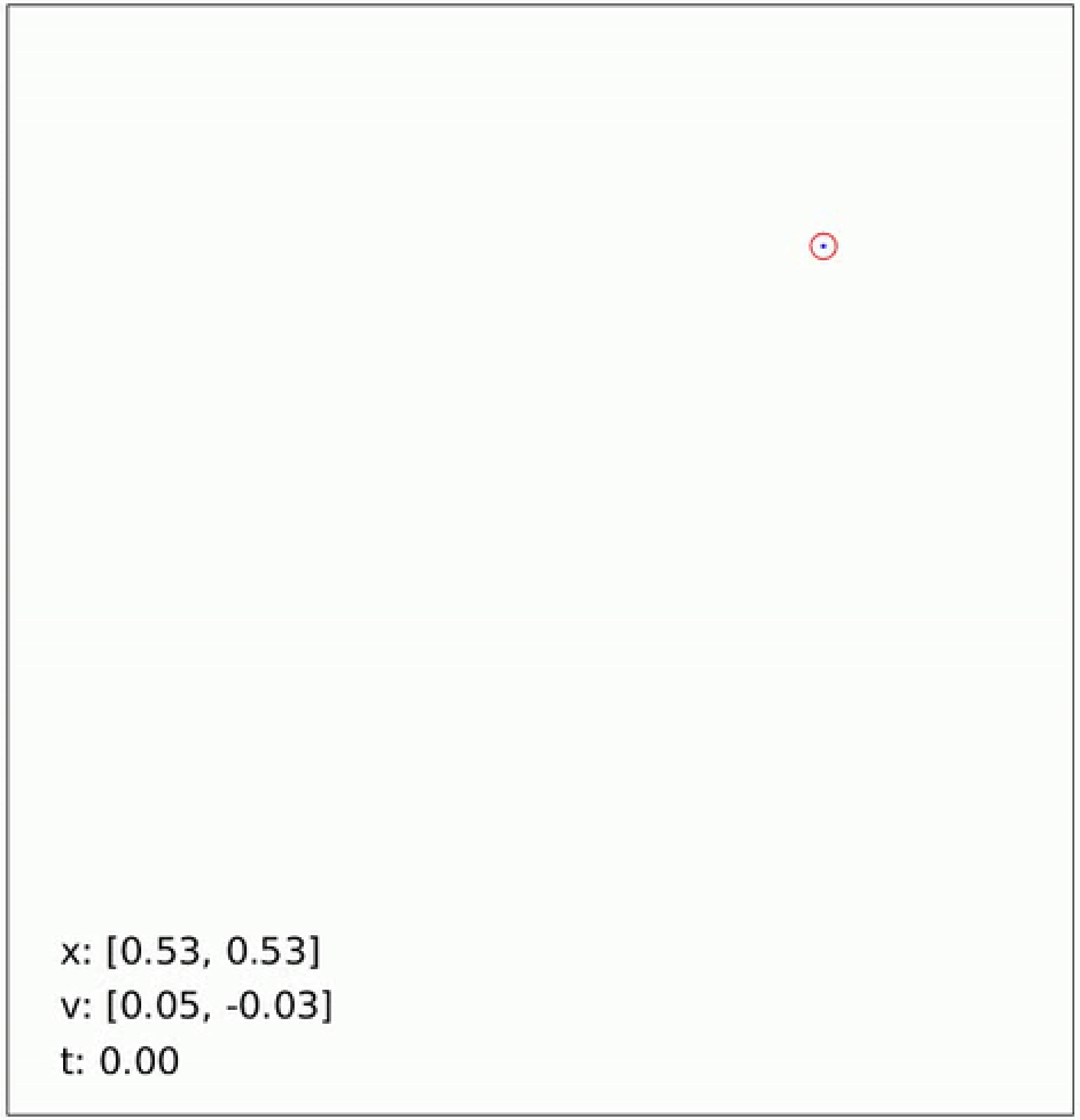
$\boldsymbol{v}$ – the particle velocity,

$\boldsymbol{e}$ – the epidemic state of the particle

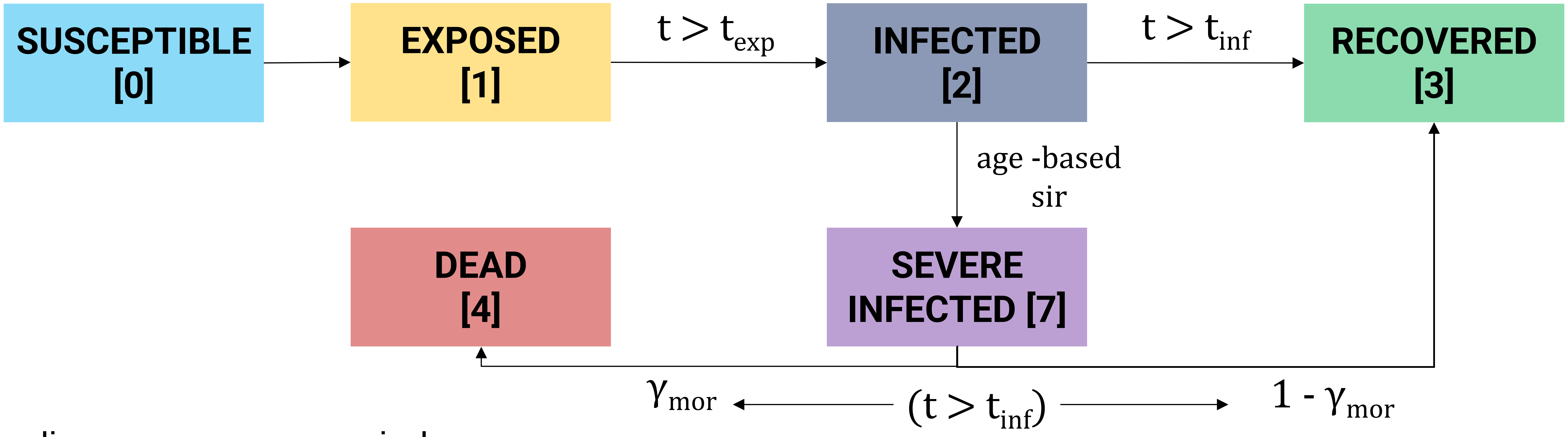
(susceptible (0), exposed (1), infected (2), severe infected (7), recovered (3), dead (4)),

$\boldsymbol{t}$ – the time of the particle in the current epidemic state,

$\boldsymbol{ag}$ – the age group



Statechart of the Particle-based SEIR simulator.



t_{exp} – disease exposure period

t_{inf} – disease infection period

t – the time of the particle in the current epidemic state,

age-based sir – rate of infected particles transitioning to the severe infected state

γ_{mor} – mortality rate

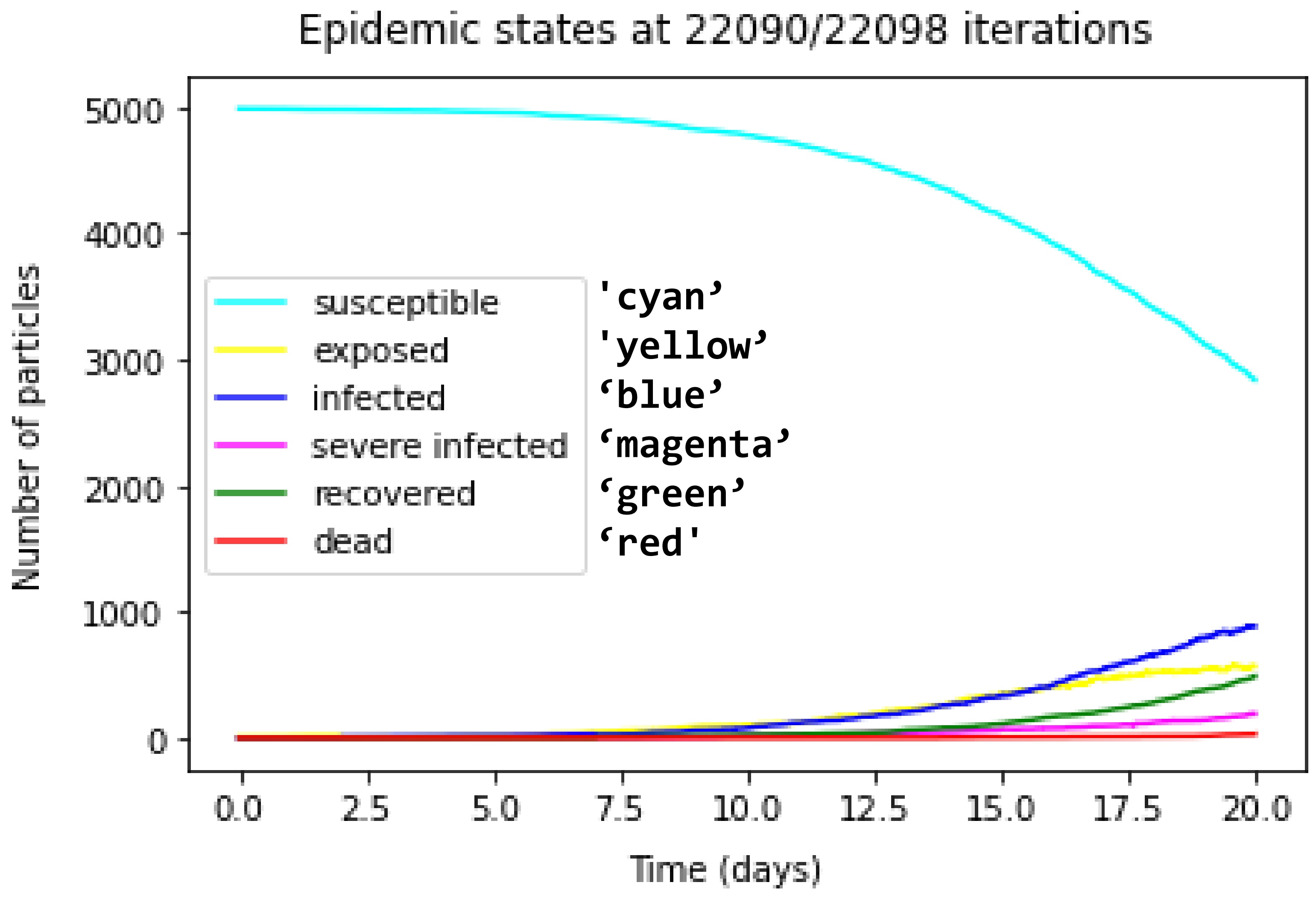
particles	This script contains Class Particles that defines the parameters and behaviours of our particles.
simulator	This script contains Class Simulator that defines the epidemic parameters and transition functions.
sir_epidemic_model	The main python file to run the SEIR epidemic simulator.

Task 1: Write a function of `Class Particles` called `update_coordinates`. The function takes `simulator`, an object of `Class Simulator`. At each iteration update the coordinate by the distance moved at the current iteration. (Hint: `simulator.delta_t` is the time a particle spends in each iteration.)

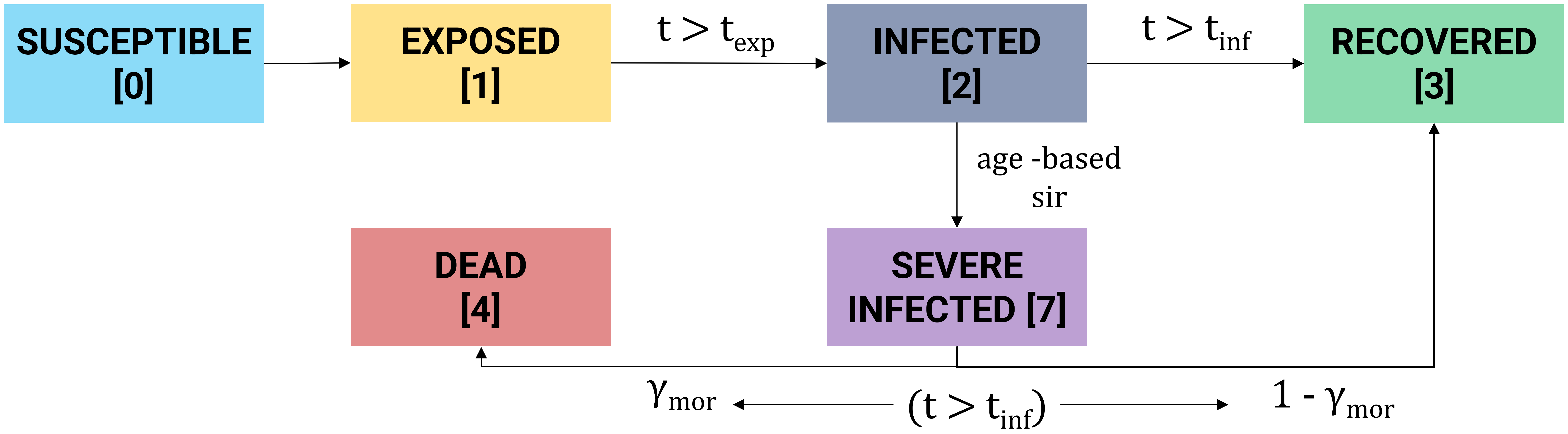
Note, the particles must stay inside of the 2D boundaries, set to [-1,1] for both dimensions. If a particle reaches one of the borders, it should be sent to the opposite side. For example, if x > 1, then update to x = -1.

Task 2: Write a function of `Class Particles` called `plot` that visualizes the epidemic curves for each state. The function takes two parameters: 1) `simulator,` an object of class Simulator; 2) `i`, the id number of the current iteration. The id should be featured in the title of the plot. The function should save the plot in .png format in the `plots` subdirectory under the name `states_i.png`, where `i` is the id number.

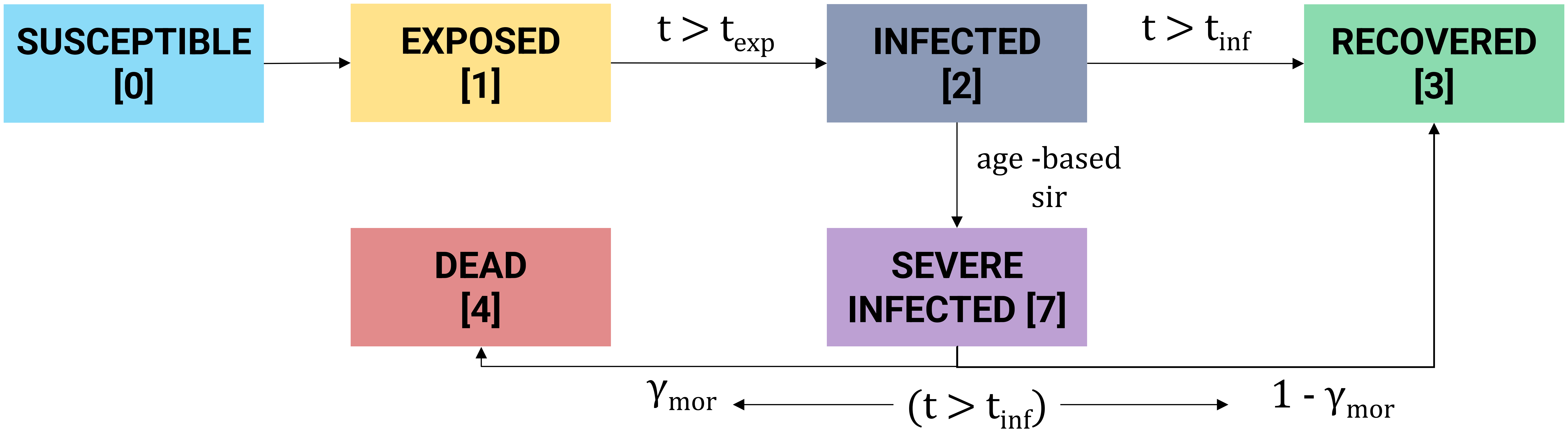
To test the function, you are provided with `data_for_plots.p`. See test # 2 in the main method of `particles.py.` The example below is the result of that test.



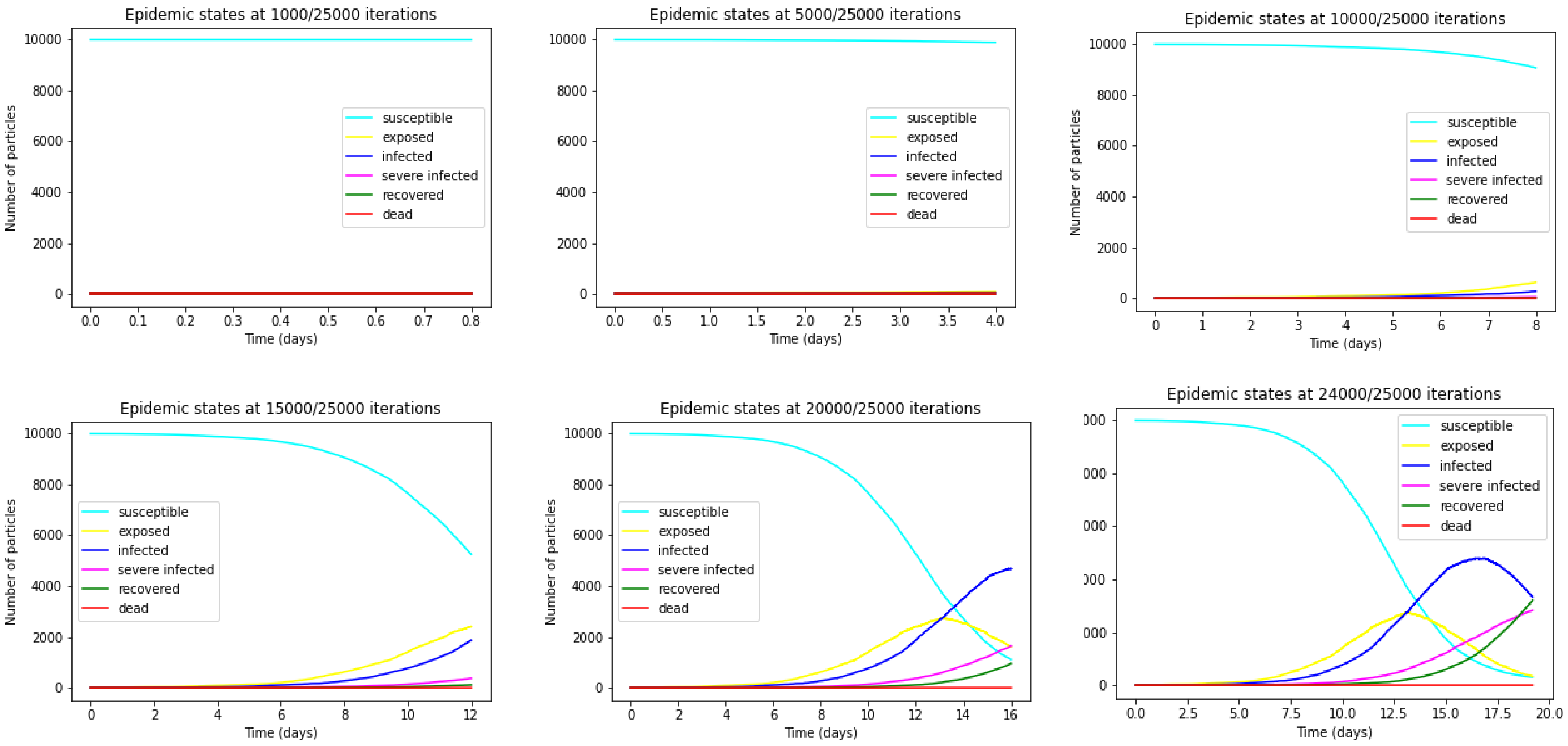
Task 3: Write a function of `Class Simulator` called `susceptible_to_exposed` that updates the epidemic status of particles from susceptible to exposed. The function takes two parameters: 1) `model`, an object of `Class Particles`; 2) `susceptible_contacted`, a list of indices of susceptible particles that were close to contagious particles (exposed, infected, severe infected) at the current iteration. Using these indices the function should: 1) update the corresponding elements in the `model.epidemic_state` array to the exposed state; 2) reset the corresponding elements in the `model.time_cur_state` array to 0.



Task 4: Write a function of `Class Simulator` called `infected_to_recovered` following the example code provided for the `exposed_to_infected` method. The function takes `model,` an object of `Class Particles.` The function should: 1) get the indices of the particles whose time at the current state have reached `T_INF` in `model.time_cur_state` array; 2) for these indices update the `model.epidemic_state` to recovered and `model.time_cur_state` to 0.



Results of the complete simulation code.



Thank you for your participation.