SocialArray: Many-Particle Simulation for Array-Based Social Interaction

Pengn Wang Xiaoda Wang Peter Luh

Neal Olderman Xuesong Lu Christian Wilkie

This article introduces a simulation tool to study a complex multi-agent system in social context. The individual-level model is extended based on self-propelled Brownian particle and social force model, and it mainly describes how agents/particles interact with each other, and also with surrounding facilities including obstructions and passageways. Most importantly, we introduce a set of arrays to define social relationship of agents/particles in a quantitative manner. Opinion dynamics is integrated with force-based interaction to study complex social phenomena including path-selection activities, social groups and herding effect. Very interestingly, the interaction of agents/particles does not only exist at physics-level, but also at consciousness and unconsciousness level by integrating advanced social-psychological studies in our modeling framework.[1]

1. Introduction

The agent-based model (ABM) is a computational research method to study social systems. This model-driven approach partly origins from statistical physics, investigating how individuals in free space or in lattice interact with each other and whether there is any converged pattern emerged at the macroscopic level. Since a society composed of many people is a typical system of many-particle, it is possible to apply the principles of statistical physics to study social behavior of many individuals. Recently, there has been a growing interest in using agent-based model and simulation to understand social phenomena such as economic market or political opinions [Peralta, Kertesz, Iniguez, 2022; Quang et. al., 2018].

The agent-based model refers to a system of many-particle that exhibits emergent characteristics when autonomous agents interact with each other. Basically, the ABM consists of agents, system space, and external environment. In our simulation platform, for example, the system space is a 2D planar space, and the environment is given as a structural layout that consists of obstructions (e.g., walls) and passageways (e.g., paths or exits), and agents are interacting with each other and moving within this structural layout. The agent is autonomous and interacts with surrounding others or the external environment based on a set of rules. The rules could be either force-based or logic-based, or a mixture of both, essentially describing how agents interact with each other.

Very importantly, the individual agent is formulated by merging knowledge in broad range from statistical physics to sociology and dynamical systems, and the macroscopic phenomena emerging from individual interactions are studied by numerical methods. The agent-based model and simulation contributes to study of collective behavior of either human crowd or other socialized living bodies like bird flock, fish school, or sheep herd. From the perspective of statistic physics the individual agent model is partly inspired from the self-propelled Brownian particles and

[1] The research program was supported in part by NSF Grant # CMMI-1000495 (NSF Program Name: Building Emergency Evacuation - Innovative Modeling and Optimization).

social force model (Helbing and Monlar 1995; Helbing, Farkas, Vicsek 2000; Lakoba, Kaup and Finkelstein 2005, Schweitzer et al., 1998b; Ebeling and Schweitzer, 2001). This model has been applied in many existing pedestrian simulators such as Menger, FDS+Evac, PTV Viswalk, MassMotion and so forth [Santos and Aguirre 2005; Ronchi, R. Lovreglio, M. Kinsey, 2020]. The model aims at investigating protypes of social group dynamics of self-propelled agents.

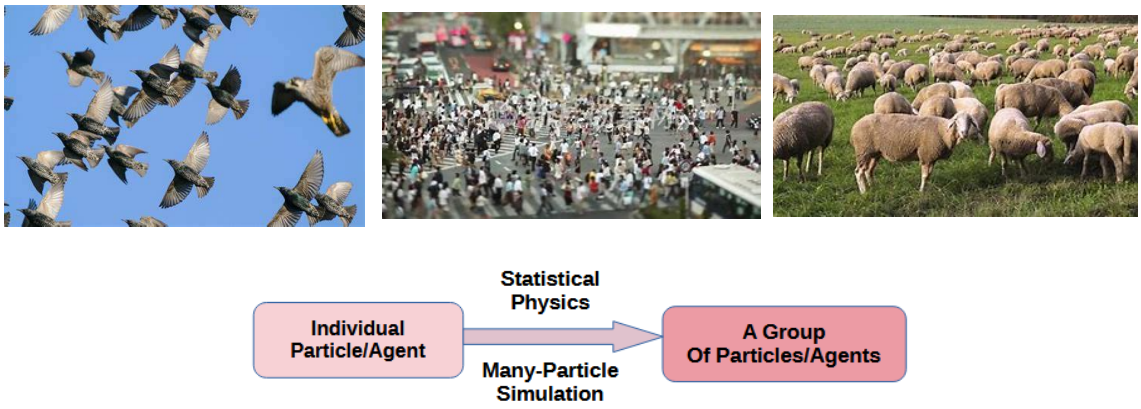


Figure 1. The Framework of Many-particle Simulation

The simulation is mainly implemented by a component as packed in a class called simulation class (simulation.py), and it computes interaction of agents with surrounding entities including walls, paths and exits. The agent model is described in agent.py, whereas walls, paths and exits are coded in obst.py. The core algorithm is still being developed and improved. This is an inter-discipline study topic, which refers to Brownian particles and statistical physics, complex dynamic systems, biological model, economic process as well as social and behavioral science. Your contributions or comments are much welcome. The program source code and numerical test cases are mainly uploaded online at https://github.com/crowdmodel/complexSocialOpinion.git and https://sourceforge.net/project/socialarray/ (https://git.code.sf.net/p/socialarray/git).

The program also consists of several functional components such as User Interface and Data/Visualization Tool.

***User Interface**: The user interface is written in tkinter in ui.py. Please run ui.py to enable a graphic user interface (GUI) where one selects the input files, initialize compartment geometry, and configure or start a simulation. An alternative method is using main.py to directly start a simulation without GUI. Currently there is a simple version of GUI and it needs to be improved in several aspects.*

***Data Tool**: This component is packed in data_func.py, and it reads in data from input files, and write data to output files. The input data is written by users in either csv files or fds input files [McGrattan et. al., 2021]. Agents must be specified in csv file while walls, paths or exits can be described either in csv file or read from standard fds input file. The simulation output is written into a binary file, which is compatible to the fds output data (fds prt5 data format). In the future we plan to visualize such data by smokeview [Forney, 2022], which is the standard tool to visualize fds output data.*

***Visualization Tool**: The visualization component is packed in draw_func.py and currently pygame (SDL for python) is used to develop this component. Users can select to visualize the simulation as it runs, or visualize the output data after the simulation is complete. If anyone is interested, please feel free to extend the module or try other graphic libraries to write a visulization component.*

As below we present the program flow chart in Figure 2. The input data is specified by a csv file to describe agents in the social context as well as compartment data such as walls, paths and exits. All the parameters to configure the simulation is also included in this csv file. The output data files include a binary data file, a text file and a npz data file (a special form in Numpy to store data array or matrix-like data). The solver for agent-based simulation is listed as below.

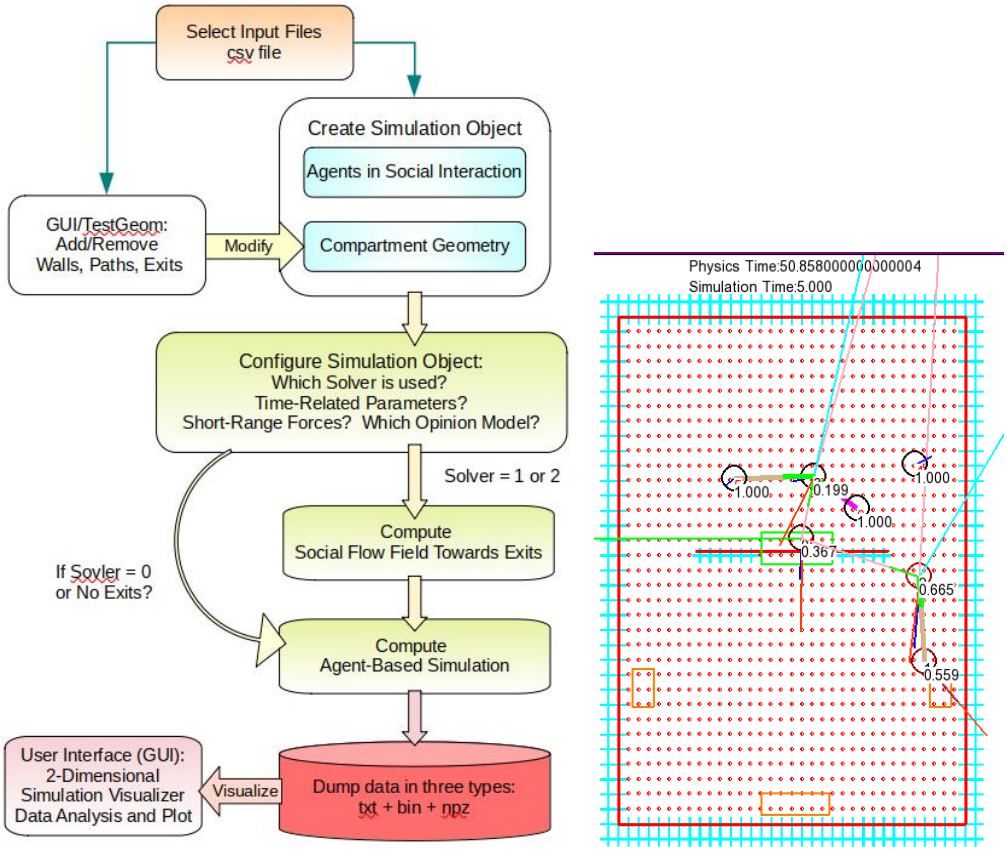


Figure 2. Program Flow Chart.

Table 1. Solver of Agent-Based Model

Solver = 2	*Compute egress flow field for all possible exits and agents select one exit based on exit-selection probability. Agents move to a selected exit when simulation time exceeds his/her pre-movement time. This is the primary solver we use in this simulation platform.*
Solver = 1	*Compute egress flow field for the nearest-exit strategy, and agents follow this flow field to use the nearest-exit to their locations. Agents move to the nearest exit when simulation time exceeds his/her pre-movement time .*
Solver = 0	*Do no use egress flow field, and agents find their ways by none-flow algorithms. In this solver agents may go to their destination site if no exit is specified in the input file. Currently, this solver has some problems to be solved. So users have no direct access to this solver by using GUI. In brief if no exit is specified in the input file, Solver=0 will be enabled automatically because egress flow field cannot be computed with no exit.*

Agent-based model (ABM) describes interactions among individual agents and their surroundings, and four types of entities are illustrated in Figure 2, which are agents, walls, paths and exits. The red lines are walls and green and red rectangular areas are paths and exits, respectively. Agents in social interactions are represented by the round disks. A discrete mesh field is also generated based on the position and shape of walls, paths and exits, where red dots in Figure 2 are open regions that agents can go through, and the crosses define the boundary regions in the mesh. The simulation object is created by reading in such entities from the input files as shown in Figure 2. The simulation object will be configured and computational process is executed to dump output data, which can be visualized by 2D viewer in our program. In the following sections we will sequentially introduce these topics.

The rest of this manual is organized as follows. In Section 2 we present how to prepare input data for agents, walls, paths and exits. Such data are used to initialize a simulation object, and configuration of a simulation object is introduced in detail in Section 3. Analysis and visualization of output data are mainly discussed in Section 4.

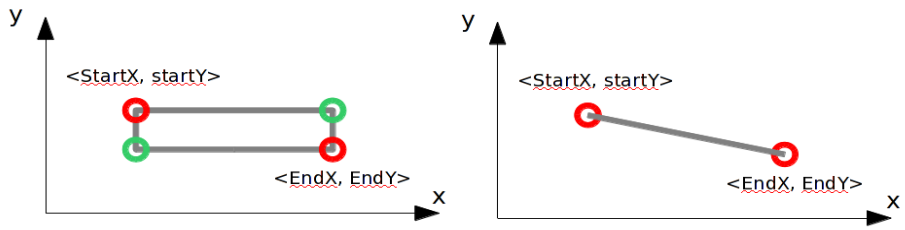
2. Preparing Input Data for Simulation

In this section we introduce how to prepare input csv data for our simulation platform. The csv (comma separated values) data is used to prepare the input data file, namely all the data cell in the input file are separated by comma. The csv data could be easily shown or modified by a table processing software such as Excel or GNUMeric, where comma are often omitted, and most text editor could also open and modify such csv data. In the latest version of the program we also intregrate a text-based csv editor and a treeview-based csv editor in the GUI such that users can easily modifly the input csv data in our program.

Walls:

Walls are obstructions in 2-dimensional space of simulation and they define the boundary of certain areas that agents cannot go through. &Wall is the identifier for the data array of walls. In other words, &Wall claims that this data array defines obstructions, and &Wall should be placed as the first element in this data array, namely, the most upper left element in the array-like data sheet. Please refer to Table 2 for more details.

In our program walls are defined in shape of either lines or rectangular areas. If any users are interested, please feel free to extend the wall types to circular or polyangular shapes. If users specify a line obstruction, it is expected to input the position of starting point and ending point of a line. If users specify a rectangular obstruction, it is expected to input the diagonal positions of two points to expand a rectangular area.



(a) Rectangular Type *(b) Line Type*

Figure 3. Create Walls in Rectangular Type or Line Type.

<name>: Name assigned to the wall, and it is arbitrarily given by users. Leave it blank if no name is assigned.

<startX, startY>: One diagonal point for rectangular obstruction; Or starting point for line obstruction.

<endX, endY>: The other diagonal point for rectangular obstruction; Or ending point for line obstruction.

<direction>: Direction assigned to the obstruction so that agents will be guided when seeing this obstruction, especially when they do not have any target path or exit. The direction means if the obstruction provides agents with any egress information or not (e.g., exit signs on the walls which direct agents to the location of exits). The value could be +1 for positive x direction, -1 for negative x direction, +2 for positive y direction and -2 for negative y direction. If no direction is given, the value is 0 by default, which means the obstruction does not provide any information of egress or exit locations. User may optionally leave it blank such that the default value 0 is used.

<shape>: Either rectangular or line obstruction in current program. The default value it 'rect', which means the rectangular shape. Use string 'line' if the obstruction is in line shape.

Table2. Data Array of Wall

&Wall	*1/startX*	*2/startY*	*3/endX*	*4/endY*	*5/direction*	*6/shape*
Wall Down	*0*	*0*	*10*	*0.5*	*0*	*rect*
Wall Top	*0*	*9.5*	*10*	*10*	*0*	*rect*
WallLeft	*0*	*0*	*0.5*	*10*	*0*	*rect*
WallRight	*9.5*	*0*	*10*	*10*	*0*	*rect*

Paths and Exit:

Paths are passageways that direct agents toward certain areas, and they may be placed over a wall so that agents can get through the wall. Paths can also be placed as a waypoint if not attached to any walls, and they can be considered as arrows or markers on the ground that guide agent towards any target site. In brief paths affect agent way-finding activities and they help agents to form a roadmap to exits. In current program paths are only specified as rectangular object, and the data array of paths is claimed by &Path or &Door, which should be written as the first element in the data array. Please refer to Table 3 for more details.

Exits are terminal locations and computation process of an agent is complete when the agent reaches an exit. It is optional to either place an exit over a wall, or anywhere independently without walls. In the program exits are only defined as rectangular areas. The data block of exits are claimed by &Exit, which should be written as the first element in the data block. Please refer to Table 4 for more details. The specific features of paths and exits are given as below.

<name>: Name assigned to the path/exit, and it is arbitrarily given by users and optionally visualized in the simulation window (pygame screen). Users could identify each path or exit by their names assigned in the input file. If no name is assigned, please leave the cell blank in the input csv file.

<startX, startY>: One diagonal point for rectangular path/ exit.

<endX, endY>: The other diagonal point for rectangular path/exit.

<direction>: Direction assigned to the path or exit so that agents will be guided when seeing this entity, especially when they do not have any target path or exit. The direction implies if the path or exit provides agents with any egress information such as exit signs or not. The value could be +1 for positive x direction, -1 for negative x direction, +2 for positive y direction and -2 for negative y direction. If no direction is given, the value is 0 by default, which means the path/exit does not provide any information of egress or exit locations. The direction is partly used in the way-finding algorithm when agents adapt their routes to exits. If users do not know how to specify the direction, simply leave it blank such that the default value 0 is used. In addition if users select solver=1 or 2 for simulation, the egress flow field will be calculated and the program will automatically assign the direction for each path, and the direction value in the csv file will be overwritten. Thus, users could simply leave it blank for solver=1 or 2.

<shape>: All the paths/exits are only in rectangular shape in current program. The default value it 'rect', which means the rectangular shape.

Table 3. Data Array of Path

&Path	1/startX	2/startY	3/endX	4/endY	5/direction	6/shape
Door Down	4.49	0	5.51	0.6	0	rect
Door Top	4.49	9.4	5.51	10.1	0	rect

Table 4. Data Array of Exit

&Exit	1/startX	2/startY	3/endX	4/endY	5/direction	6/shape
Exit Down	4.5	–0.3	5.5	0.3	0	rect
Exit Top	4.5	9.7	5.5	10.3	0	rect
Exit Right	9.3	4.3	10.3	5.9	0	rect

Agents:

Finally and most importantly, agents are the core entity in computation process. They interact with each other to form collective behavior of crowd. They also interact with above types of entities to form egress motion toward exits. The resulting program is essentially a multi-agent simulation of pedestrian crowd. Each agent is modeled by extending the well-known social force model. The model is further advanced by integrating several features including pre-movement behavior, group behavior, way-finding behavior and so forth. In current program the data block of agents are claimed by &Agent or &Ped, and they are written as the first element in the data block.

<name>: Name assigned to each agent and it is arbitrarily given by users and optionally visualized in the simulation window (pygame screen). The names of agents are given in the first column of the data block. Users may leave this column blank if no names are used.

<InitalX, InitialY>: Initial position of an agent in 2D planar space. Please note that such initial position is required to be given by users and no default values are used in program.

<InitialVx, InitialVy>: Initial velocity of an agent in 2D planar space. The default value is <0.0, 0.0>, which means that agents have zero velocity in the initial state, i.e., they are standing still in the initial position at time point t=0.0.

<tau>: Tau parameter in the social force model, or as usually called relaxation time in many-particle systems or statistical physics, and it critically affects how fast the actual velocity converges to the desired velocity. The default value is 0.6.

<tpre>: Time period for pre-movement phase of agents, which is also called milling process such that agents could sufficiently exchange opinions and form social groups. Within this time period agents do not select and move towards an exit. The default value is 10 seconds. The timeline model of pre-movement phase and movement phase is illustrated in Figure 3. In the pre-movement phase agnets are exchanging opinions rather than actually move to any exit. The pre-movement process is critically studied in this simulation platform, where various existing models in opinion dynamics are applied to aggregate pre-movement time of many individuals. In simulation platform of social array, behavior pattern of multiple agents is commonly divided into two phases: pre-movement phase and movement phase. In the pre-movement phase agents are collecting information rather than actually move to any destination exit. They exchange opinons with each other based on the social topology of them, and it is a milling process such that agents are self-organized into social groups in pre-movement phase. In our program several opinion dynamics models are used to aggregate pre-movement time of many individuals. As shown in Figure 4 the simulation starts at t=0. Let $tpre_i$ denote the pre-movement time interval of agent i, and it is updated by mixing itself with a weighted average of others given certain social relationship among individuals. When the simulation t >= $tpre_i$, agent i starts to move to an exit selected.

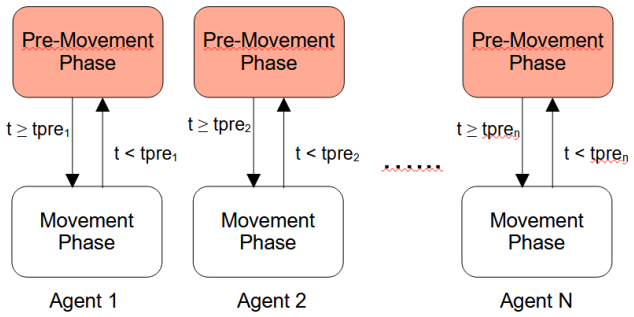


Figure 4. Time-Based State Transition Model: Pre-Movement Phase and Movement Phase.

<p>: Parameter p in opinion dynamics, and it indicates how an agent's opinion/decision is impacted by surrounding others, and it critically affects herding effect in collective behavior. The measurement of this parameter is within [0, 1], and an agent's opinion/decision completely follow others if p=1. In contrast, if the agent makes decision only based on his or her own opinion, the parameter p=0, and this usually implies that the agent is a leader in a group. Thus, we also call p by decision balance parameter because it determines how an individual agent keeps balance between his or her own opinion and others' opinion in decision making process. In sum our model and algorithm critically consider the social topology of many individuals in exit selection behavior. In a sense it is useful to describe leader-and-follower relationship, or more generally as called the herding effect in collective behavior.

<pMode>: This parameter indicates whether or how parameter p is dynamically updated. Currently there are three methods to be selected: random, fixed or stress. If 'random' method is used, it means that parameter p is randomly generated in the interval of [0, 1] by uniform distribution. If 'fixed' method is selected, parameter p is given by the initial value in the input csv file, and it keeps constants through the simulation. The 'stress' method will adapt parameter p by each agent's state (e.g., stress level). In principle the value of p increases if an agent feels more stressed. This method is being tested and it involves a computational model to adapt parameter p dynamically to surroundings.

<p2>: This parameter affects how much an agent tends to make a decision based on the information timely collected from the surrounding facilities such as the distance to the target exits or received guidance from the outside. The

measurement of this parameter is within [0, 1], and an agent's opinion/decision completely follows the received information if p2=1. In contrast, if the agent makes decision only based on his or her past experience and the new information is completely ignored, the parameter p2=0. In our algorithm parameter <p2> is the weight for an agent to timely integrate such new information into one's decision-making process.

<tpreMode>: The type of arousal level in the pre-movement phase. Some agents may actively exchange information with others and they are assigned with the highest level 3 in simulation. The level 2 of arousal implies that agents will passively interact with others, and they only receive information but not actively search for new information from surroundings. The level 1 is the lowest arousal level in our model, meaning that agents stay static in their initial positions (e.g., in sleep). The arousal level will increase irreversibly in the simulation process.

<moveMode>: The type of way-finding behavior. Some agents may actively move to a known exit while others may just search for new information. In current simulation the active agents are guided by the egress flow field and the searching agents will find their ways to exit by using solver=0.

<inComp>: a boolean variable to indicate if the agent is in computation loop or not. Normally it is given true. If users want to quickly remove an agent in computation loop, please assign it false for convenience.

<talkRange>: The range to determine when agents have opinion exchange. Such interaction could also be understood as herding effect or group opinion dynamics, which means agents exchange opinions by talking.

<talkTau>: Tau parameter when agents have opinion exchange, and it affects how fast the actual velocity converges to the desired velocity when agents are in states of talking. The default value is 0.6.

<talkProb>: The probability to determine if agents have opinion exchange. Such interaction could also be understood as herding effect or agents exchange opinions by talking.

<c2>: The weight parameter to determine whether an individual agent will follow others' behavior or opinion. The default value is 0. Please refer to Wang, 2016 for more details of the theoritical model.

<radius>: The radius of agents.

<mass>: The mass of agents.

<max speed>: The maximal moving speed that an agent is able to achieve in movement.

Table 5. Data Array of Agents

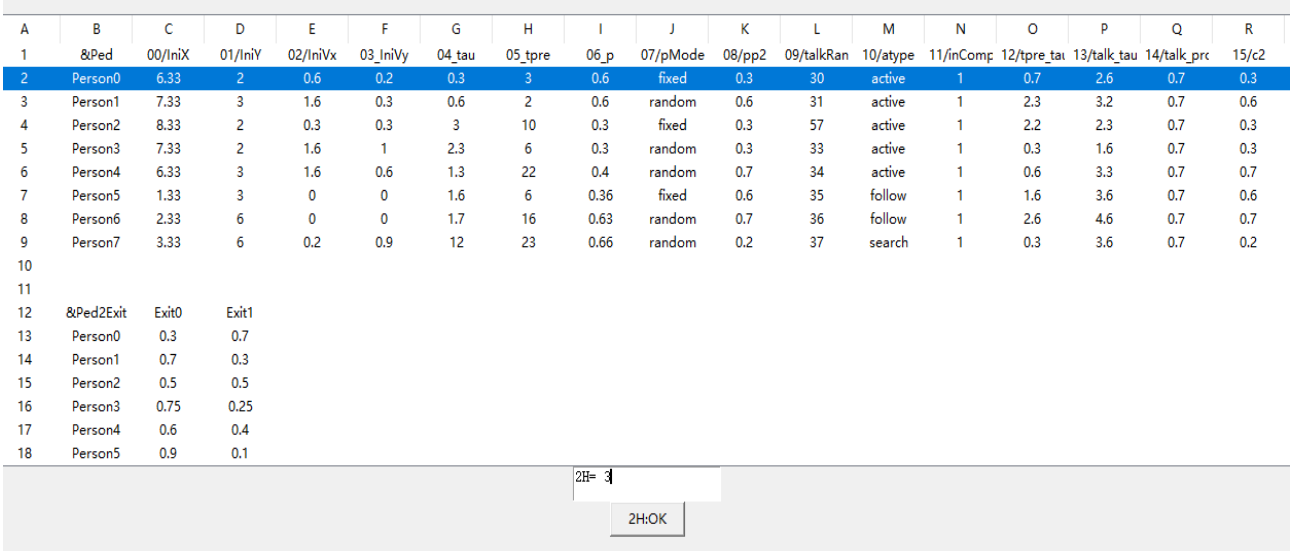
&Agent	01/ IniX	02/ IniY	03/ IniVx	04/ IniVy	05/ tau	06/ tpre	07/p	08/ pMode	09/ pp2	10/tpre Mode	11/move Mode	12/ inComp	13/talk Range	...
Tom	6.3	2	0.0	0.0	1.3	18	0	fixed	0.7	0	active	1	10	...
Tuck	7.3	3	0.6	0.2	0.6	7	0.6	random	0.6	2	active	1	20	...
James	7.3	6	0.3	0.7	0.6	6	0.6	stress	0.6	1	search	1	30	...

Additional Notice:

(1) The sequence of &Wall &Path &Exit &Agent could be arbitrarily changed within an input file. Users may either first specify walls, or paths or agents. However, data feature inside a data block could not be changed in sequence. For example, you must first give <startX>, and next <startY> for walls, paths and exit, and cannot change the sequence to be <startY> and <startX>, or move <endX> or <endY> before them. All the features for walls, paths, exits and agents are read in certain order, and such order cannot be altered in current version (version 2.3). Please check functions of readAgents(), readWalls(), readExits() and readDoors() in the source file data_func.py for more details. In addition new contributors are welcome to improve these functions in data_func.py such that this sequence can also be adjusted by users (e.g., using the DataFame in python pandas module).

(2) As above we exemplify all the input data by table-like data array, and users can select a table processing software such as Excel or GNUMeric to show such data. However, please remember that csv (comma separated values) is the basic format of the above data file, namely all the data cell in the input file are separated by comma. Those comma are omitted in Excel or GNUMeric, but are shown in any text editor. Please note that when users save the data in such table procressing softwares, the csv data could be slightly extended automatically, for example, adding many empty cell at the end of each data array. Usually such empty cells are useless and our program still works well with them. In the latest version of the program we also intregrate an easy text-based csv editor in the

GUI, where tab (/t) is used to align the data element in csv file, and no empty cells will be added in each line. There is also a treeview-based csv editor integrated in the GUI, and users can use it to modifly the input csv data and also no empty cells will be added at the end of each line.



A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	&Ped	00/IniX	01/IniY	02/IniVx	03_IniVy	04_tau	05_tpre	06_p	07/pMode	08/pp2	09/talkRan	10/atype	11/inComp	12/tpre_tau	13/talk_tau	14/talk_prc	15/c2
2	Person0	6.33	2	0.6	0.2	0.3	3	0.6	fixed	0.3	30	active	1	0.7	2.6	0.7	0.3
3	Person1	7.33	3	1.6	0.3	0.6	2	0.6	random	0.6	31	active	1	2.3	3.2	0.7	0.6
4	Person2	8.33	2	0.3	0.3	3	10	0.3	fixed	0.3	57	active	1	2.2	2.3	0.7	0.3
5	Person3	7.33	2	1.6	1	2.3	6	0.3	random	0.3	33	active	1	0.3	1.6	0.7	0.3
6	Person4	6.33	3	1.6	0.6	1.3	22	0.4	random	0.7	34	active	1	0.6	3.3	0.7	0.7
7	Person5	1.33	3	0	0	1.6	6	0.36	fixed	0.6	35	follow	1	1.6	3.6	0.7	0.6
8	Person6	2.33	6	0	0	1.7	16	0.63	random	0.7	36	follow	1	2.6	4.6	0.7	0.7
9	Person7	3.33	6	0.2	0.9	12	23	0.66	random	0.2	37	search	1	0.3	3.6	0.7	0.2
10																	
11																	
12	&Ped2Exit	Exit0	Exit1														
13	Person0	0.3	0.7														
14	Person1	0.7	0.3														
15	Person2	0.5	0.5														
16	Person3	0.75	0.25														
17	Person4	0.6	0.4														
18	Person5	0.9	0.1														

Figure 5. CSV Data Editor Integrated in GUI (DataTool)

(3) Please note that all the entities including walls, paths, exits and agents are also assigned with a unique ID number, and this number is automatically given by the program. The ID number starts from zero and it increases based on the sequences of entities specified in the input csv file. Both of the name and ID numbers could be visualized in simulation window (pygame screen). If users hope to change this ID number, the only method is adjusting the sequence of specifying these entities in the data array of input csv file.

(4) The shade area specifies the required parameters that users must input to initialize a simulation object. If not, the simulation object cannot be initialized. For other parameters, if users do not understant what they excatly means, please omit them in the csv input file and the default value will be automatically chosen by the program.

The above parameters are related to the theoretical model of agents in various scenarios such as in pre-movement phase or in talk state. In brief these parameters significantly affect how an agent interacts with others and how they decide the egress routes in collective behavior. Please refer to our article (Wang et. al., 2023) for more details of the theoretical model. As below we only highlight the key features of our model as shown in Figure 3.

3. Brief Introduction of Theoretical Model:

Brownian motion have long been studied on a diversity of fields, not only in physics of statistical mechanics and molecular dynamics, but also in biological models, finance process, and social systems. In the past twenty years, there has been a growing interest in studying the model in self-propelled feature and interaction force such that the model also fits into study of social phenomenon of many individuals. The model is mathematically described by

$$m_i (d\boldsymbol{v}_i/dt) = -\gamma \boldsymbol{v}_i + \gamma \boldsymbol{v}_i^0 + \sum_j \boldsymbol{f}_{ij} + \boldsymbol{\xi}_i \qquad (1)$$

where m_i is the individual mass, and the change of the individual instantaneous velocity $\boldsymbol{v}_i(t)$ is determined by the total force executed on the particle, including a (sliding) friction forces $-k\boldsymbol{v}_i$, driving force $k\boldsymbol{v}_i^0$, interaction force $\sum_j \boldsymbol{f}_{ij}$ and fluctuation $\boldsymbol{\xi}_i$. *A conceptual difference between the self-propelled Brownian particle and traditional Brownian particle is that the driving force is not exerted from external origin such as a gravitational or electrical field, but assumed to be associated with each single particle and self-produced. Thus, this implies that each particle has some kind of internal energy reservoir (Schweitzer et al., 1998; Ebeling et al., 1999). The resulting self-driven particles are a paradigm for many active or living systems, where they are a simplified and abstract representation of the most important dynamic behavior of cells, animals, or human pedestrians. In a mathematical sense, it is feasible to*

aggregate the first and second term in the right side of Equation (1) together as a self-driving force term, which is expressed by $\boldsymbol{f}_i^{drv} = \gamma(\boldsymbol{v}_i^0 - \boldsymbol{v}_i) = m_i(\boldsymbol{v}_i^0 - \boldsymbol{v}_i)/\tau_i$ *, where* $\boldsymbol{v}_i^0$ *is given by a self-produced field and* $\boldsymbol{v}_i$ *is the physical velocity of the particle. This term formalizes a linear feedback mechanism such that the physical velocity* $\boldsymbol{v}_i$ *converges towards desired velocity* $\boldsymbol{v}_i^0$ *with exponential convergence rate of* $\gamma = m_i/\tau_i$ *. This is a research branch that mainly studies the self-propelled feature of Brownian particle.*

Another major branch is using social force or social field to describe interaction of many particles. Traditionally, in statistical physics the interactive forces are often omitted such as ideal gas, and researchers are more interested in the fluctuation force in regards of the stochastic thermal effect. However, in modeling of social behavior of many living creatures it is favorable to have a system where the fluctuations are not significantly large compared to the systematic, deterministic motion. Thus, social force was introduced to represent a class of deterministic interaction among individuals (Helbing and Monlar, 1995; Helbing, Farkas, Vicsek, 2000). The resulting model was named by social force model, which has been widely applied in simulation of pedestrian crowd in the past twenty years such as in FDS+Evac (Korhonen, 2017, McGrattan et.. al., 2018), Menge (Kim, Guy and Manocha, 2010), MassMotion (Oasys, 2018), VisWalk and SimWalk. Consequently, each individual agent motion is described by physics laws as given below, where the traditional fluctuation force is ignored or integrated into other force terms, and boundary wall force is further included in simulation platform in order to confine motion of agents or particles within certain areas.

$$m_i(d\boldsymbol{v}_i/dt) = \boldsymbol{f}_i^{drv} + \sum_j \boldsymbol{f}_{ij} + \sum_w \boldsymbol{f}_{iw} \qquad (d\boldsymbol{r}_i/dt) = \boldsymbol{v}_i \tag{2}$$

This modeling approach is also called force-based method, where forces motivates agents to respond to surrounding stimuli, and the forces mainly consist of driving force $\boldsymbol{f}_i^{drv}$*, interaction force* $\boldsymbol{f}_{ij}$ *as well as boundary wall force* $\boldsymbol{f}_{iw}$*. Here we emphasize that these force components may also feature in randomness, describing the stochastic behavior on how agents perceive surroundings and integrate information for decisions. In a sense we may also understand that the traditional fluctuations are integrated into* $\boldsymbol{f}_i^{drv}$ *or other force components. In particular when such randomness is added in the driving force* $\boldsymbol{f}_i^{drv}$*, the problem becomes interesting because it refers to the discrete choice process in social-economic systems.*

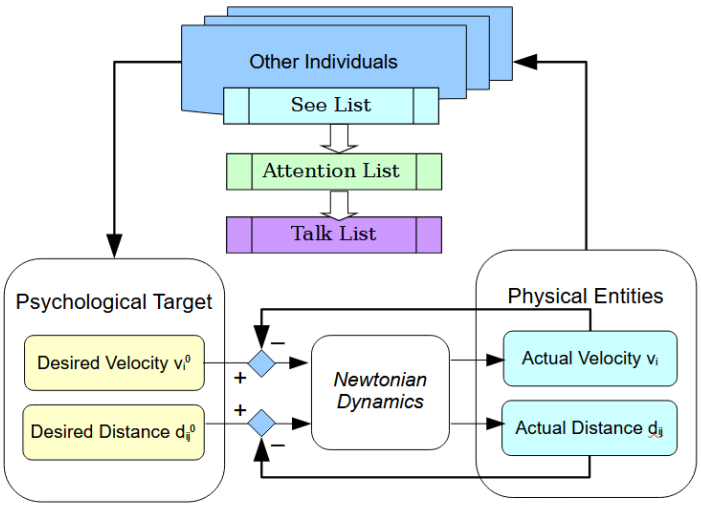


Figure 3. Agent-based model with Environmental Facilities.

<u>Forced-Based Model and Opinion Dynamics:</u>

The driving force represents an energy source that drives people to move to a destination location, and it is straightforward to use a linear form $\boldsymbol{f}_i^{drv} = m_i(\boldsymbol{v}_i^0 - \boldsymbol{v}_i)/\tau_i$ *as introduced above. This term could be further generalized by* $\boldsymbol{f}_i^{drv} = \boldsymbol{F}^{drv}(\boldsymbol{v}_i^0 - \boldsymbol{v}_i)$*, where* $\boldsymbol{F}^{drv}(.)$*is generally a monotonically increasing function. Here the model does not highlight how an individual gains energy from outside to realize motion. Rather, it emphasizes how an individual transforms the desirable motion in mind into the physical motion in reality. In other words, it is assumed that the individual always has enough energy to achieve desirable motion, and the key issue exists in the desired state in one 's opinion. The opinion thus becomes the essential subject of study.*

Now a natural question is where such opinion come from. From the perspective of Brownian particles $\boldsymbol{v}_i^0$ *comes from the self-produced field as mentioned above. Especially, in regards of the social-economic phenomenon we assume that the self-produced field* $\boldsymbol{v}_i^0$ *is mainly described by selection among several alternatives. For example, a group of people are discussing to choose model A or B when purchasing a car, or voting for several candidates in*

presidential election. The opinion of each individual agent is described as the selection probability among multiple discrete choices, and probability distribution evolves as individuals interact in social context, resulting in a class of stochastic process that characterizes the collective decision in human society. Consequently, the stochastic effect does exist, not due to ''thermal-like'' effect or fluctuations in the traditional Brownian motion, but from stochastic selection among several deterministic choices.

The interaction force $\boldsymbol{f}_{ij}$ describes the social-psychological tendency of two individuals to keep proper distance (as called the social-force), and if people have physical contact with each other, physical forces are also taken into account, i.e., $\boldsymbol{f}_{ij} = \boldsymbol{f}_{ij}^{soc} + \boldsymbol{f}_{ij}^{phy}$. In this article we mainly focus on the social force term. In Helbing et. al., 2002 and 2005 this term is given in an exponential form, and we will integrate a new concept of desired distance d_{ij}^0 into this force term. In a general sense we define this force term by $\boldsymbol{f}_{ij}^{soc} = \boldsymbol{F}^{soc}(d_{ij}^0 - d_{ij})$. The interaction of an individual with obstacles like walls is treated analogously, and denoted by $\boldsymbol{f}_{iw} = \boldsymbol{f}_{iw}^{soc} + \boldsymbol{f}_{iw}^{phy}$, and this force specifies boundary of people's motion. When an agent is sufficiently close to a path, door or exit, $\boldsymbol{f}_{iw}^{soc}$ also include attractions calculated between the agent and the path, door or exit, and such attractions are called door force in the output text file, and it is mainly used for Solver=0. As for Solver=1 or 2, because agents are guided by egress flow fields, it is not that useful to calculate such door forces.

In our agent-based model we will mainly focus on $\boldsymbol{f}_i^{drv}$ and $\boldsymbol{f}_{ij}^{soc}$ in the background of evacuation study. In addition agents can choose information used for the representation of social environments and are able to cast their attention purposely on certain details. The feature of selective attention is critically modeled by three lists in our simulation platform, which are see list, attention list and talk list. For simplicity, users could understand such three lists as three different sets, and the talk list is a subset of the attention list, and the attention list is a subset of see list. In other words, we assume that an agent first sees surrounding others such that all the agents in his or her viewsight are included in the see list. The member in the see list will be selectively added in the attention list if the agent will cast his or her attention to the member in the see list, and such selection process is calculated based on the social topology among agents. We will introduce this issue soon later. Finally, the agent may talk to some members in the attention list with certain probability, and such talking behavior forms the talk list for the agent. In practical computing these lists are represented by matrix-based data, and updated in the simulation process. They critically determine how an individual agent selects others in surrounding environment for social interactions.

In our agent-based model we will mainly focus on $\boldsymbol{f}_i^{drv}$ and $\boldsymbol{f}_{ij}^{soc}$ in 2-dimensional space. In addition agents are able to choose information used for the representation of social environments and to cast their attention purposely on certain details. The feature of selective attention is critically modeled by three lists in our simulation platform, which are seeing list, attention list and talking list. Here users could understand such three lists as three different sets in a mathmatical sense: The seeing list contains the attention list and the attention list likewise contains the talking list. In other words, we assume that an agent first sees surrounding others such that all the agents in his or her viewsight are included in the seeing list. The member in the seeing list will be selectively added in the attention list if the agent will cast his or her attention to the member. Furthermore the agent may talk to certain members in the attention list, and the member will be added in the talking list. In practical computing the lists are represented by matrix-based data, and updated through the simulation process. They critically determine the time-variant social topology among agents in 2-dimensional space.

An interesting topic is that the self-driving force and interaction force could be understood by stress theory in social-psychological study, which is caused by mismatch between psychological demand and realistic situation (Staal, 2004), and we summarize the mismatch in terms of velocity and interpersonal distance as shown in Figure 3: the psychological demands are abstracted as desired velocity $\boldsymbol{v}_i^0$ and desired interpersonal distance d_{ij}^0 while the physical reality is described by the physical velocity $\boldsymbol{v}_i$ and distance d_{ij}. The difference of two variables measures how much people feel stressed, by which people are motivated into certain behavior. The motivation is abstracted as the driving force and social force (Wang, 2016; Wang and Wang, 2021), and the entire process formalizes the stimuli-reaction model (S-R model) in psychological view of behaviorism.

Table 6. On Conception of Stress and Force-Based Model

	Opinion (Psychological Characteristics)	***Behavior (Physics-Based Characteristics)***	***Difference between subjective opinion and objective reality***	***Forced-Based Term***
Time-Related Characteristics	**desired velocities** $\boldsymbol{v}_i^0 = v_i^0 \boldsymbol{e}^0_i$	**actual velocities** $\boldsymbol{v}_i = v_i \boldsymbol{e}_i$	***Time-Related Stress: Velocity*** $\boldsymbol{v}_i^0 - \boldsymbol{v}_i$	***Driving Force*** $\boldsymbol{f}_i^{drv} = \boldsymbol{F}^{drv}(\boldsymbol{v}_i^0 - \boldsymbol{v}_i)$
Space-related Characteristics	**desired distance** d_{ij}^0	**actual distance** d_{ij}	***Space-related Stress: Distance*** $d_{ij}^0 - d_{ij}$	***Social Force*** $\boldsymbol{f}_{ij}^{soc} = \boldsymbol{F}^{soc}(d_{ij}^0 - d_{ij})$

Next we will emphasize that $\boldsymbol{f}_i^{drv}$ $\boldsymbol{f}_{ij}^{soc}$ and $\boldsymbol{f}_{iw}^{soc}$ are all subjective entities coming from people's opinions, and they are generated intentionally by people through foot-ground friction on physics basis. These forces essentially describe how an individual perceive and respond to the outside environmental stimuli. In particular $\boldsymbol{v}_i^0$ and d_{ij}^0 are non-physics entities for they exist in people's opinion, not in the physical world, and we have well explained the psychological background of the model in Wang, 2016, and also critically modified the key concept of social force in consistency of both physical laws and psychological principles.

Moreover, since $\boldsymbol{v}_i^0$ and d_{ij}^0 are non-physics entities that represent opinion of living creatures, it is reasonable to integrate opinion dynamic model with the above force-based method to study how agents interact in social context. In brief $\boldsymbol{v}_i^0$ and d_{ij}^0 of agents interact with surrounding others, and this feature is especially useful to characterize the collective motion of many agents (Viscek, 1995). Consequently, the interaction of agents could be summarized at two different levels.

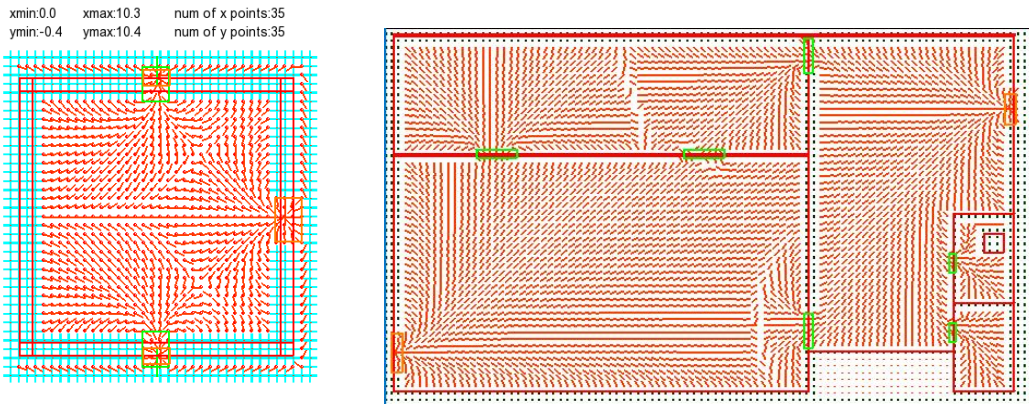
(a) Behavior-Level Interaction: The agents are autonomous whose behaviors are motivated from opinions, and thus have self-propelled motion. At the individual-level, such motion is propelled by agents' opinion. When they are aggregated into groups in collective behavior, and such group behavior forms a complex interactive system. Such forced-based interaction is largely inspired by the well-known Brownian particles and social force model. In principle of physics the behavior of each agent is abstracted as motion characteristics including position and velocity, which are mathematically generated from a sort of desired state at the opinion level.

(b) Opinion-Level Interaction: The opinions of individuals interact given a social topology, and group-level opinion emerges from the operation of linear or nonlinear rules that govern interactions among agents. Agents are assumed to be rational, but usually have only limited information for decision-making. They adapt using learning from past memories as well as surrounding others' opinions. By processing new information and interacting with others, opinions may change timely, and they are extensively represented by several features of agents, including exit-selection probability and pre-movement time, and we will discuss these topic as below.

Exit Selection Probability:

Way finding refers to how individual agents orientate themselves towards exits, especially within a multi-compartment layout. Several factors influence their decision making in way-finding activities such as guidance information or other individuals' choice, and it characterizes how agents perceive and integrate various information acquired from surrounding facilities and individuals (Tong and Bode, 2022). In our simulation platform agent i starts to move towards an exit when the simulation time $t>tpre_i$. The problem could be formalized in two steps: (a) How to choose a destination exit; (b) How to select a proper route to reach the exit.

If users select the default setup of using the nearest-exit strategy, it implies that each agent will select the exit nearest to their current locations, and no exit-selection probability is involved. This means that solver 1 is used in our program as listed in Table 1 such that the program calculates an egress flow field for the nearest-exit strategy, and agents follow this field to move towards the nearest-exit to their locations. The egress flow field for nearest-exit strategy is illustrated in Figure 5.



(a) *(b)*

Figure 5. Flow field of using the nearest exit.

Another solver deals with a more complicated scenario, where each individual's choice is determined by exit-selection probability distribution. This means that solver 2 is selected in our program as shown in Table 1. In other words we suppose that there are several candidate exits known by agents and we simulate how they select one among the known exits and choose a proper way to reach them.

The exit probability is initialized by a data array as exemplified in Table 7, where each row corresponds to each agent and the column refers to exit selection probability. The entire data array is claimed by &Agent2Exit or &Agent2Exit, which is written in the position of the first element. All the agents select exits based on the discrete probability distribution as described by the array. One thing to be emphasized is that if the agent does not know an exit, the corresponding data element is given by -1 or any negative value. In Table 7 the agent named by Luca does not know there is Exit2 in the given compartment layout. Thus, the in the exit selection algorithm the probability for Luca to use Exit2 keeps zero until he is told by others that there is Exit2 or he actively find this exit.

Table 7. Data Array of Exit-Selection Probability

&Agent2Exit	*Exit0*	*Exit1*	*Exit2*
agent0	*0.7*	*0.2*	*0.1*
David	*0.3*	*0.5*	*-1*
Irving	*0.0*	*-1*	*0.5*

In brief all the exits are classified in two types for each agent: known exits and unknown exits. An unknown exit will not be included in the exit-selection algorithm for the specific agent until the agent gets a way to know it. When the element of the data array is assigned with zero probability, it means that the agent does know the exit, but never goes there before. For example, in Table 7 Irving does know Exit0, but he never use it previously, and thus the historical usage frequency of that exit for Irving is simply zero, i.e, assigned with zero probability initially. This exit is definitely included in the computational loop of exit-selection algorithm, and thus the probability for Irving to select Exit0 is dynamically updated through the simulation process. This algorithm is being tested and it is inspired and developed based on the codework of FDS+Evac.

In the simulation process the prior probability distribution of using multiple exits will be dynamically updated when new information received. In a sense it is feasible to build up a probabilistic model to link different factors in a statistical sense to update the exit-selection probability (Wang et. al., 2022). This process refers to how an individual deals with the new information received from surroundings, and commonly there are two source of new information: (1) Information from surrounding facilities; (2) Information from surrounding people.

The egress layout and facilities determine a quantitative measure on utility of each exit, and thus utility model is useful to timely integrate such information into one's decision making process (Lovreglio, Fonzone and dell'Olio, 2016). For example, the utility of an exit is larger for an agent if it is closer to the agent's location, and thus it is more likely for the agent to select it. Extensive research has been conducted on how to properly develop such a utility function, and it refers to social and economic models and theories. In our algorithm parameter <p2> is the weight for an agent to timely integrate such new information into one's decision-making process.

More importantly, our model and algorithm considers the social topology of many individuals in exit selection behavior. In a sense it is useful to describe leader-and-follower relationship, or as more generally called the herding effect in collective behavior. In this situation the choice of others or a leader in social group are very important, and it may significantly impacts an agent's probability distribution of exit selection. In our algorithm parameter <p> is the weight for an agent to timely integrate such new information into one's decision-making process.

All in all, solver=2 considers a more realistic and complicated scenario which assumes that each individual's choice is made by integrating three major factors: (1) historical knowledge of building layout or prior information such as habit of using different path; (2) Judgment on current situation and integrate timely information for decisions (e.g., the distance to an exit) (3) Choices of other agents in given social topology. Because each individual's decision is interacting with others, the collective decision-making is critically studied with integration of various social-psychological findings, especially for de-individualization effect in social groups. To well describe such social group behavior users should well define parameter <p> and <p2> for each agent in csv file. As a result, the data array claimed by &agent2exit only gives the initial value of exit-selection probability distribution, and how the probability evolves dynamically in the simulation is critically determined by parameters of <p>, <pMode>and <p2> as mentioned before. To better understand how <p> <pMode> and <p2> works together in our simulation model and algorithm, we give the following equation where $Prob(\text{individual } i \text{ select exit } e)$ *is the probability of*

individual agent i selects exit e, and it forms a stochastic process which is updated by integrating three basic conventions: memory of past experience, bounded rationality, and herding instinct.

$$Prob(\text{individual i select exit e}) = (1-p)(1-p2)*Prob(\text{individual i select exit e by using prior experience}) + (1-p)*p2*Prob(\text{individual i select exit e by utility of the path}) + p*Prob(\text{individual i select exit e by hearing others' opinions})$$

$$0<=p<=1,\ 0<=p2<=1$$

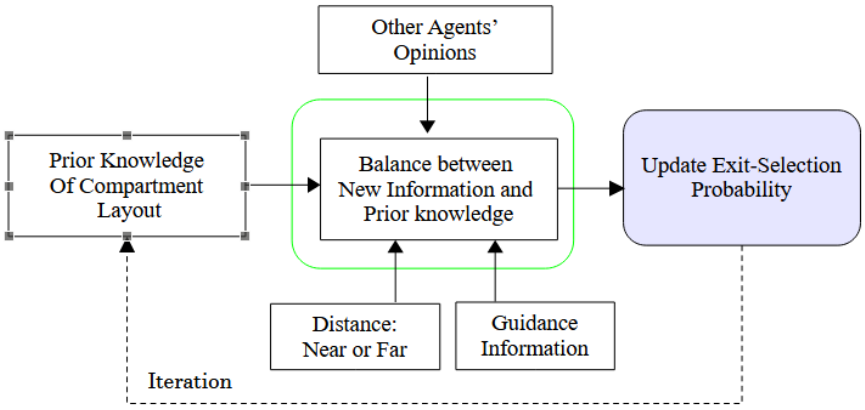


Figure 6. Information fusion for timely updating the exit-selection probability.

The operational choice is to generate the egress route based on the exit selected. There are a number of methods used in existing egress simulators. Our algorithm is partly learned from FDS+Evac, where the route is calculated by a two-dimensional flow solver. The computation result is a 2D flow field that guides agents to the exit selected. The flow field can be better explained as a social field related to social norms or other behavioral characteristics (Lewin, 1951; Helbing et. al., 2000; Wang, 2022), and we will further elaborate the idea in future. In this algorithm each exit is a sink point, and solver calculates the route as the crowd flow move to the sink (Korhonen et. al., 2008; Korhonen, 2016). The detailed discussion of the flow solver will not be included in this article, but we emphasize that this method is more suitable to describe human collective behavior on the background of social science and psychological theory.

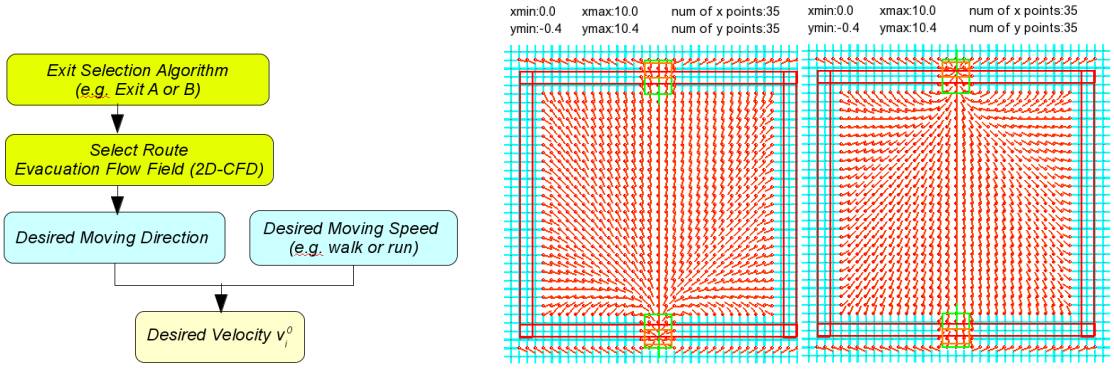


Figure 7. Simulation of crowd egress with exit-selection algorithm.

Social Topology of Agents:

In our simulation platform a data array is used for a quantitative measure of social relationship among agents, and it is claimed by &groupSABD as shown in Table 8. Here &groupSABD implies that the data array consists of four different sets of values to describe social relationship of agents, which are sequentially named by S, A, B, D, and they are separtated by the notation of '|' or ";" in the cell. The initial and most important value is S, meaning the social weight among agents, and it describes how an individual bring others' opinion in forming his or her own

opinion. The group social force are next described by A, B, D such that individuals are socially bonded with each other in physical positions, and thus help to form a common motive in collective motion. In principle opinion dynamics is a more fundamental feature to describe how individuals interact in social context, and thus it is suitable to first give matrix S as a quantitative measure of social relationship, and matrices A, B, D are next specified in consistency. If the element in matrtix S is zero, then the corresponding elements in A, B, D are given zero automatically, and the value in the cell is simply denoted by 0 as shown in Table 8.

In this data array shown in Table 8 we can generally identify a group which consists of individual 0, 1, 2. In this group individual 0 is completely a follower to individual 1 with a weight of 1.0, and individual 1 also cares about individual 0 with a weight of 0.2. Such valuation represents a kind of leader-and-follower or child-and-parents relationship in social topology. Individual 1 and 2 care about each other significantly (with weight of 0.5 and 0.7), and both of them are also socially bonded with individual 3 (with weight 0.3 and 0.3).

In contrast individual 5, 6, 7 are considered as another social group. In particular individual 5 and 6 compose a stable pair because they care about each other mutually (e.g., couples). Individual 7 also follow individual 3 and 5, but individual 3 and 5 does not care individual 7, and thus they form a sort of leader-and-follower pattern.

Individual 3 and 4 connect the above two groups. However, such connection are directional in a sense that individual 3 and 4 are widely impacted by individual 5, 6, 7 (second group) and individual 0, 1, 2 (first group) are moderately impacted by them.

In this social topology individual 5 and 6 have the leadership, and they directly impact individual 3, 4, 6 and 7 and also relies on individual 3 and 4 to indirectly affect individual 0, 1, 2. In addition individual 3 is the critical bridge who connects these two groups. If individual 3 moves out of this crowd, the two groups become isolated completely.

Table 8. Data Array of Social Groups

&groupSABD	Agent0	Agent1	Agent2	Agent3	Agent4	Agent5	Agent6	Agent7
Agent0	0	1|170|10|2	0	0	0	0	0	0
Agent1	0.2|60|6|2	0	0.5|30|10|3	0.3|20|20|1	0	0	0	0
Agent2	0	0.7|30|20|3	0	0.3|90|3|2	0	0	0	0
Agent3	0	0	0.3|30|6|2	0	0	0.1|10|2|1	0.6|60|2|1	0
Agent4	0	0	0	0.2|70|2|1	0	0.3|20|2|1	0	0.5|20|3|1
Agent5	0	0	0	0	0	0	1|70|3|1	0
Agent6	0	0	0	0	0	1|200|1|1	0	0
Agent7	0	0	0	0.5|26|8|1	0	0.5|3|1|1	0	0

In the following discussion we will introduce the group social force as specified by the parameter A, B, D. Now considering a system composed by *n* agents, and the entire agent state space is the union of each agent's state space, i.e., $U_i\{ \boldsymbol{r}_i(t), \boldsymbol{v}_i(t), tpre_i(t), prob_i(t)\}$. *where $\boldsymbol{r}_i(t)$ or $\boldsymbol{v}_i(t)$ are the instantaneous position and velocity of agent i as specified by Equation (1) and (2). The cohesive effect is included in the interaction force $\boldsymbol{f}_{ij}$ in Equation (1) and (2). Several force description in molecular dynamics are suitable to describe cohesive effect among agents, such as Lennard-Jones potential or Morse potential. The focus of our study is not comparing these forces and here we will briefly extend the traditional social force (Helbing, Farkas, and Vicsek 2000) to describe the cohesive effect. Given agent i and j, and the group force exerted on agent i from agent j is described as below.*

$$f_{ij}^{soc} = \frac{A_{ij}}{B_{ij}}\left(d_{ij}^{0} - d_{ij}\right)\exp\left[\frac{\left(d_{ij}^{0} - d_{ij}\right)}{B_{ij}}\right]n_{ij} \qquad (3)$$

In Equation (3) A_{ij} and B_{ij} are positive constants, which affect the strength and effective range about how two agents are attracted to each other and $\boldsymbol{n}_{ij}$ is the normalized vector which points from agent j to i. The above mathematcial description of interaction force integrates attraction into the traditional social force (Helbing, Farkas, and Vicsek 2000; Helbing et. al., 2002), and it is functional in a feedback manner to make the physical distance d_{ij} approaching towards the equilibrium distance d_{ij}^{0}. The parameter A, B, D are thus specified by A_{ij}, B_{ij} and d_{ij}^{0} from Equation (3). Due to the page limit we will not further describe the mathematical model of social groups in this manual. If users are interested in our social group model, please refer to Wang et. al., 2024 for more details, and a similar data array is presented in Section 4 of the article.

In an addition, there is another data array called exit2door, which is claimed by &exit2door in the input csv file. This data array specifies the path direction towards a given exit. For example, in Figure 5(b) there are two exits in the structural layout, and if the upper right one is selected by an agent, the path direction towards this exit will be given by the first row in the following data array. If the lower left exit is selected, the path direction will be given by the second row. The data array &exit2door is optionally specified by users. In fact if a flow solver is used in our

program, this array could be automatically generated by the egress flow field. Thus, users could usually omit this data array in the input csv file unless no egress flow field is calculated (solver=0) in the program.

Table 9. Data Array of Exit2Door Matrix

&Exit2Door	*door0*	*door1*	*door2*	*door3*	*door4*	*door5*
Exit 0	*1*	*1*	*-1*	*-1*	*-2*	*-2*
Exit 1	*-1*	*-1*	*-1*	*-1*	*2*	*2*

4. Configure Simulation Parameters and Run Simulation

As above we briefly introduce how to write a csv file to describe agents, walls, paths and exits. In addition another config.txt file is often used jointly with csv file to configure the simulation parameters. In our program there are mainly two methods to configure a simulation object. One way is using GUI, and it is the most user-friendly method to setup all the simulation parameters. The other one is using a script file, config.txt, which is to be placed in the same folder as the input csv file. In this section we will first explain how to use GUI (tkinter window) to set up the simulation parameters. Writing config.txt is briefly mentioned next.

In Tkinter GUI:

When tkinter window (GUI) is activated, users will see several panels including <QuickStart>, <Parameters> and so forth. Next, we will briefly introduce how to set up the simulation in the panels.

The default panel is called <QuickStart>. In this panel users select the input files to get a quick run of the simulation. Choose csv file to specify agent data. Users can optionally use fds file to create the compartment geometry, and the pedestrian features are described in csv file. If both csv and fds files are presented, the compartment structure will be created by fds file. If fds file is omitted, the compartment geometry should be described in csv file. The agents must be specified in csv file currently while the walls, paths and exits can either described by csv file or fds file. Please take a look at the examples for details.

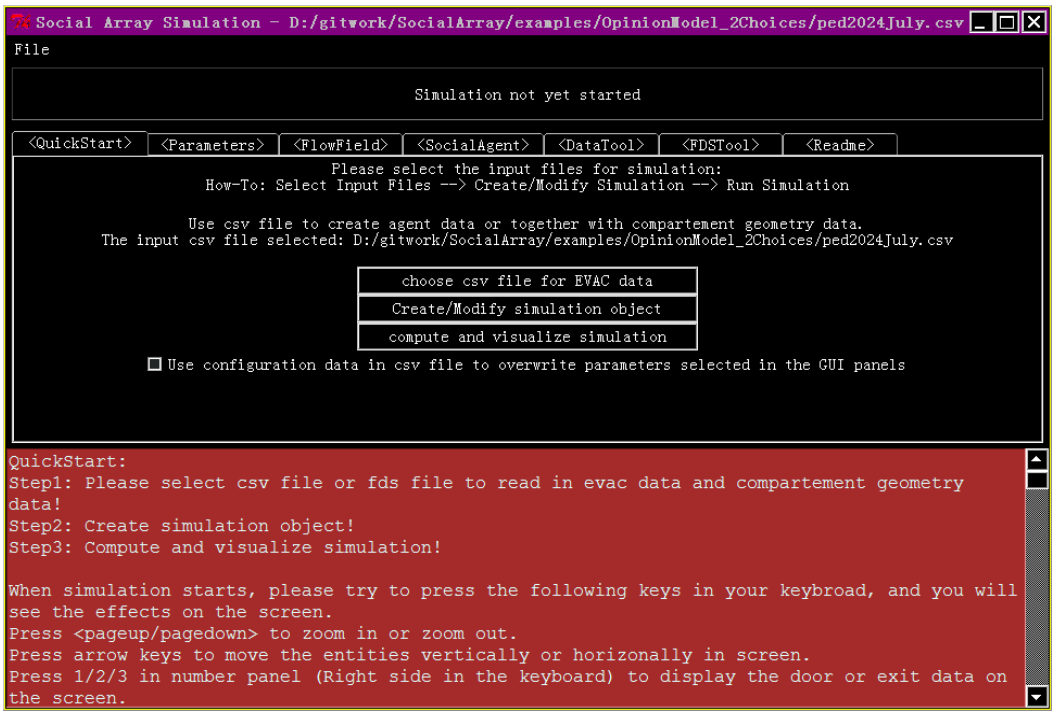


Figure 7. User Interface of CrowdEgress (QuickStart)

<Create/Modify simulation object>: After the input files are well selected, it is suggested that users first create the simulation object to see if the structural layout is created as expected. Another screen is next displayed by pygame, where users could check all the entities in the simulation, such as wall, paths, exits and initial potions of each agent. We call this screen TestGeom in this manual, and it means testing the geometric settings of compartment layout. In TestGeom user could add walls, paths or exits, or change the direction of paths or exits, or change the initial position of each agent. Whenever users change the input files or after an existing simulation is complete, users should re-click this button to initialize the simulation object. Otherwise the simulation object will remain to be the previous one in the memory and new simulation object will not be created.

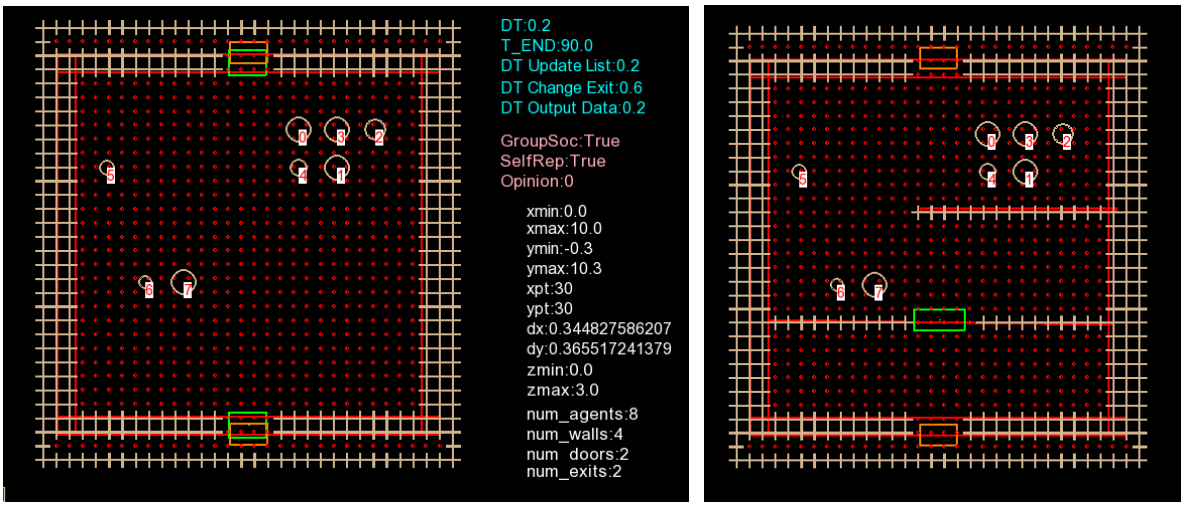
In Pygame Screen (TestGeom):

When pygame screen is activated, press keys of PageUp/PageDown in your keyboard to zoom in or zoom out the entities in screen. Use direction keys to move the entities vertically or horizontally in screen. In fact there are several sections in pygame screen, and these keys work in similar ways in all the sections. The first section we will introduce is called TestGeom as shown in Figure 8, where users can visualize compartment geometric settings and modify them manually.

In TestGeom users can add walls, paths or exits by selecting the corresponding items in the menu bar (See Figure 7 and 8). When the corresponding item is seleced, simply drag a rectangular region to create a new wall, path or exit in rectangular shape. A wall in line shape can also be created by drawing a line from its start point to end point. If users are not satisfied with the new entities created, please press <z> in the keyboard to remove it. Please be careful to remove entities because the last element in the list of walls, paths and exits will be removed by pressing <z>, not only for the new entities that users graphically created in testGeom. Sometimes it is necessary to adjust the exit-selection probability if new exits are added in TestGeom. Namely, the number of column in data array &Agent2Exit (See Table 1) should be equal to the total number of exits. If users do not manually update &Agent2Exit in the input csv file, the program will automatically add zero columns there.

When all the items in the menu bar are closed, users may also adjust the initial position of each agent or change the direction of a path or exit. Simply drag an agent to a new position, namely, drawing a line from the existing position to a new position, and the agent will be moved there. The direction of a path and exit is adjusted by drawing a line from outside to inside of the path or exit. However, if solver 1 or 2 is selected for the simulation, users do not need to care about this issue because such directions will be automatically updated by the egress flow solver.

Users can also dump geometric data (e.g., wall data and path data) into csv file by selecting the item <OutputData> in the menu bar. The output file is created as bldDataRev.csv for any modification of the compartment geometric in TestGeom. The data can also be briefly shown in the screen by selecting the item <ShowData> in the menu bar. If users click <Simulation>, then the simulation starts and the program goes to the second section called RunSimulation.



(a) TestGeom ***(b) Add paths or exits in TestGeom***

Figure 8. Initialize Simulation Object in Pygame Screen of TestGeom.

After the simulation object is initialized or modified, the next step is running the simulation. There are several options for users.

<Compute Egress Flow Field>: The first option is to compute the egress flow field and display it on the pygame screen, and agent-based simulation is not started in this option. In the pygame screen users will find how the mesh is created such as the number of x points and y points, the boundary value of mesh (i.e., min_x, max_x, min_y, max_y). Users will also see the flow field towards each exits. If solver 1 is selected, there is only one flow field computed, which is simply using the nearest-exit strategy (See Figure 4). If solver 2 is selected, there are several flow field computed, which corresponds to the roadmap towards each exit (See Figure 6). The nearest-exit strategy is also additionally computed in solver 2. Users could press <o> and <p> in the keyboard to switch from different flow field. Please also see the third panel of EgressFlow for more details.

<Compute Simulation and Dump Binary Data>: The second option is to compute the agent-based simulation with the flow field. The numerical result will be written into a binary file (.bin), a text file (.txt) and a npz data file (.npz). but not displayed timely in pygame screen. If your input csv file is named by agent.csv and simulation is complete at time of 2024-03-02_23_09_27 (in format of year, month, date, hour, minute and second), the output data files will be generated as agent_2024-03-02_23_09_27.bin, agent_2024-03-02_23_09_27.txt and agent_2024-03-02_23_09_27.npz. We will introduce these files in detail in the following section.

<Compute and Visualize Agent-Based Simulation>: The third option is to compute the agent-based simulation and visualize the result timely in pygame screen. This is the most common and important option such that users are able to directly observe how agents interact and move towards a selected exit in the compartment layout. As below we will introduce this pygame section as called RunSimulation. Users can directly observe the forces and movement trace of each agents, pause the simulation. The output data can also be saved as in the previous option, namely the numerical results are saved into a binary file (.bin), a text file (.txt) and a npz data file (.npz).

<u>In Pygame Screen (RunSimulation):</u>

In pygame section of RunSimulation the agent-based simulation is visualized on the pygame screen. User can pause the simulation, but cannot rewind it in current version (Version 2.3). A binary data file is optionally created when the simulation runs and users can also visualize the data after the simulation is finished. In both phases of TestGeom and RunSimulation, there are hot keys defined. Use pageup/pagedown to zoom in or zoom out the entities in screen. Use space key to pause the simulation. Use direction keys to move the entities vertically or horizonally in screen. Use 1/2/3 to display the path or exit data on the screen. Press I to show agent index number. Press S to show their stress level timely. The stress level is computed based on our work Wang and Wang 2020.

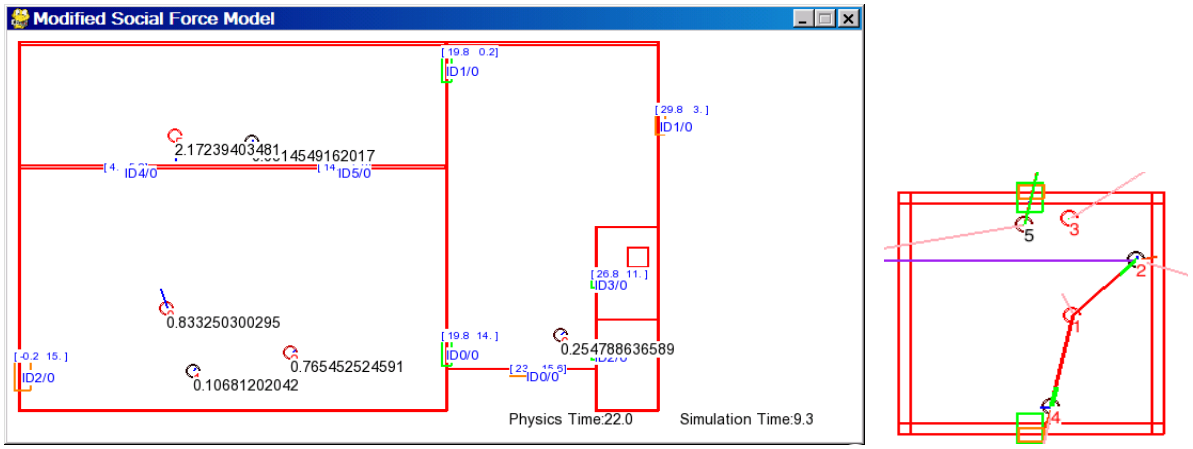


Figure 9. Agents with forces or stress level indicated.

The second panel is called Parameters, where users specify the basic parameters before the simulation starts. An important issue is that z-interval (min_z and max_z) should be specified when a FDS+Evac input is used to create the

compartment geometry setting. The reason is that FDS+Evac is a 3D simulator, and its input file may include several computation meshes as several floors in a building. However, our simulator CrowdEgress is a 2D simulator only for single-floor layout. Thus, if a FDS+Evac input file is used with several compartment floors, sometimes users should check or modify the min_z and max_z to determine which floor should be computed in the simulation. The default value in CrowdEgress is given as min_z=0.0 and max_z=3.0, and it is generally suitable for most FDS+Evac file with only one compartment floor. Other parameters are briefly introduced as below.

<Show Time in Simulation>: Show the computational time and simulation time when simulation starts.

<Show Stress Level in Simulation>: Indicate the difference between actual velocity and desired velocity. Please refers to Wang, 2021 for more details.

<Use nearest exit strategy>: Solver 1 is used if selected. Namely, each agent is guided to the nearest exit and exit-selection algorithm is not involved. Solver 2 is used if unchecked.

<Show compartment data in simulation>: Show the geometric data of paths and exits in the pygame screen such that users can identify each path and exit by their indice and default directions.

<Show forces on agents in simulation>: Visualize forces in the pygame screen. The forces are measured by several lines shown on each agent. The line in purple color is the wall force; line in green color represents the door force. The line in pink color is the social force.

<Dump data to a binary file>: Dump simulation data into a binary file which is compatible to fds prt5 data format. The data file is used to visualize the numerical result after simulation stops.

<Group behavior>: Use group social force in simulation. This is a little complicated issue and we will elaborate it in future.

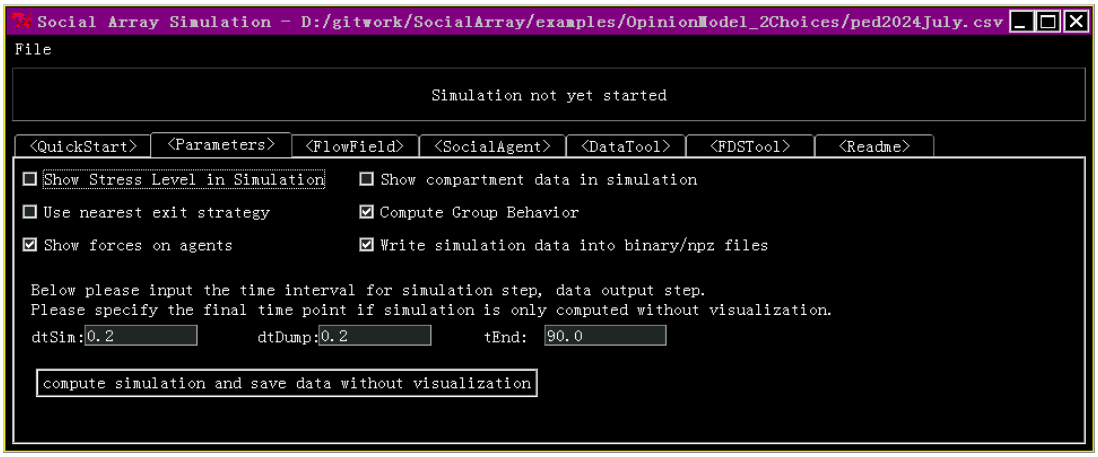
<Write data to a binary/npz file>: Write simulation data into a binary file which is compatible to FDS prt5 data format. Also, write the array-like or matrix-based data into a npz data file, which is a special form used in Numpy. These data files are used to visualize the numerical result after simulation is accomplished.

<dtSim>: The time interval (second) for each simulation step. In Figure 10, for example, the time interval is 0.2 second for each simulation step, and all the agents state including position, velocity and forces are updated numerically every 0.2 second.

<dtDump>: The time interval to write simulation data into the binary and npz data file. For example, in Figure 10 the simulation data is written into the binary and npz file every 0.2 second, which is the same as the default time interval for each simulation step. This means that all the simulation data in all time steps are saved into the binary and npz data files.

<tEnd>: The entire time span for simulation. In Figure 10, for example, the simulation starts at t=0.0 second and is accomplished at t=90.0. This value is only valid if simulation is computed without timely visualization, namely, if users selects the mode of <Compute Simulation and Save Data> as introduced before.

<Use config.txt to overwrite parameters selected in GUI panels>: If checked, all the above simulation parameters will be overwritten by reading the script of config.txt, which is placed in the same folder of the input csv file.



0

Figure 10. User Interface of CrowdEgress (Parameters)

The third panel is called EgressFlow, where users specify the basic parameters to calculate the egress flow field. The calculation is executed based on the solver selected. If solver=1, the egress flow field is calculated only for the nearest-exit strategy. If solver=2, several flow fields are calculated, and each one corresponds to an exit, and the total number is the number of exits plus one because the nearest-exit strategy is also calculated for solver=2.

In the panel of EgressFlow users are expected to input the dimensional measure of the calculation mesh, including the minimal and maximal value in x axis (x_min and x_max) and minimal and maximal value in y axis (y_min and y_max). The number of points are denoted by num_x and num_y. When the number of points increases, the computational mesh are refined and the calculation time is supposed to be increased also.

There are two buttons in this panel. One is <Compute Egress Flow Field>, and it is the same as in panel of QuickStart, and it generates a pygame screen, where users will find how the mesh is created such as the number of x points and y points, the boundary value of mesh (i.e., min_x, max_x, min_y, max_y). Users will check the flow fields in the pygame screen: if solver 1 is selected, only one flow mesh is computed for the nearest-exit strategy (See Figure 3). If solver 2 is selected, several flow fields are computed, which corresponds to the roadmap towards each exit (See Figure 4). The nearest-exit strategy is also additionally computed in solver 2. Users could press <o> and <p> in the keyboard to switch from different flow field.

Another button is especially useful for Solver 1 and it shows the computational result of crowd fluid dynamics. This fluid-based model is different from the agent-based model, and it assumes crowd movement as mass flowing in a two-dimensional space, and it is basically an analytical model at the macroscopic level, aggregating many particle-like individuals into fluid-like model of crowd. In a sense the fluid-based model (macro-level) is derived from homogeneous individual equation (micro-level) based on physics laws and *mathematical principles. The resulting model is a set of partial differential equations (PDE). The analytical solutions to these equations are often difficult in mathematics, but numerical solutions are obtained nowadays by advanced computing methods. Thus, we apply a simple version of fluid-based model in our program by extending 1D traffic problem (Lighthill and Whitham, 1955) to 2D crowd fluid problem. The computation process of this simple fluid-based model is included in Solver 1. In other words when users select <Use nearest exit strategy> in the <Parameters> panel, it means that Solver 1 is selected and this fluid-based model will be calculated along with the egress flow field.*

The solution could be visualized by using the second button in <EgressFlow>, i.e.,the button named by <Read output npz file and show crowd fluid model>. The computational result is stored in vel_flow1.npz in the example folder, and it is visualized in Figure 9.

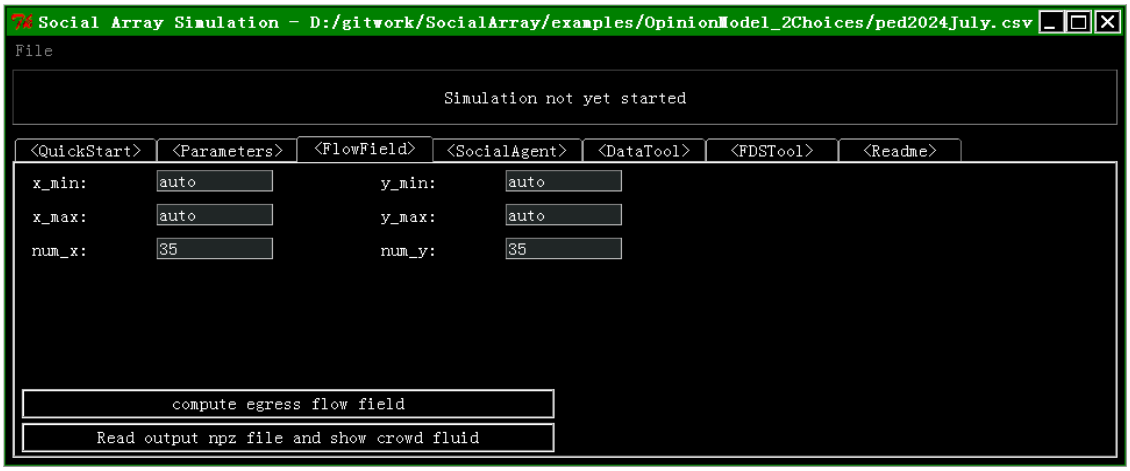


Figure 11. User Interface of CrowdEgress (EgressFlow)

As shown in Figure 9 the basic information of egress flow field is first displayed such that users could check if the mesh is generated as expected, including the number of x points and y points and the boundary value of mesh (i.e., min_x, max_x, min_y, max_y). When the pygame screen is next displayed, the crowd flow density is denoted by a small black dash in the mesh, and users will see how they move towards the nearest exit. In this process it is also easy to use mouse to check the numerical value of the flow velocity and density for each mesh unit. U and V are the flow velocity, and R represents the flow density. Ud and Vd are the desired flow velocity, which always point towards

the nearest exit. BLD means whether the mesh unit is free for crowd flowing or solid boundary generated from the compartment layout.

In this process users could also use <pageup/pagedown> to zoom in or zoom out the entities in screen. Use <space> key to pause the simulation. Use arrows to move the entities vertically or horizontally in screen. The general framework of the fluid-based analysis is illustrated as below.

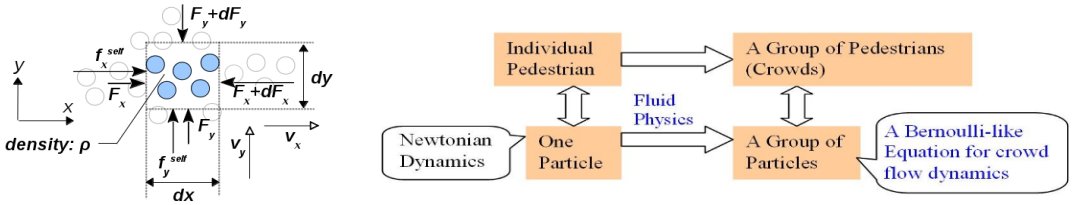


Figure 12. Fluid-Based Analysis of Crowd Movement

Last but not least, analog of fluid dynamics is only valid to crowd at medium or high density, where continuity hypothesis holds for crowd flowing in a planar space. If there are only sparse individuals, continuity hypothesis cannot hold, and fluid-based analysis is not actually suitable for such low-density crowd. The resulting fluid model provides a practical perspective to explain crowd behavior at bottlenecks (e.g., narrow passages), where crowd density are sufficiently large and short-range physical interactions are dominant among people. Such interactions are among the major cause of crowd disastrous events like stampede.

Furthermore, the discrete mesh of a compartment layout is also useful for other simulation algorithm, including a variety of discrete opinion dynamics model such as majority-rule model, voter model, Sznajd model, social impact model. Different from the fluid dynamics as introduced above, these models only take dicrete values at each lattice cell, often accepting only two different results. The evolution of binary results is also similar to celluar automata (CA), which is also used to simulate crowd motion in 2-dimensional lattice. Generally speaking, either deterministic or probabilistic rules can be applied in these discrete models, and interaction of each lattice cell follows logic rules, not force-based rule. If anyone is interested in applying any opinion dynamics model or CA model in our simulation platform, please contact me and I am glad to provide you guidance and help.

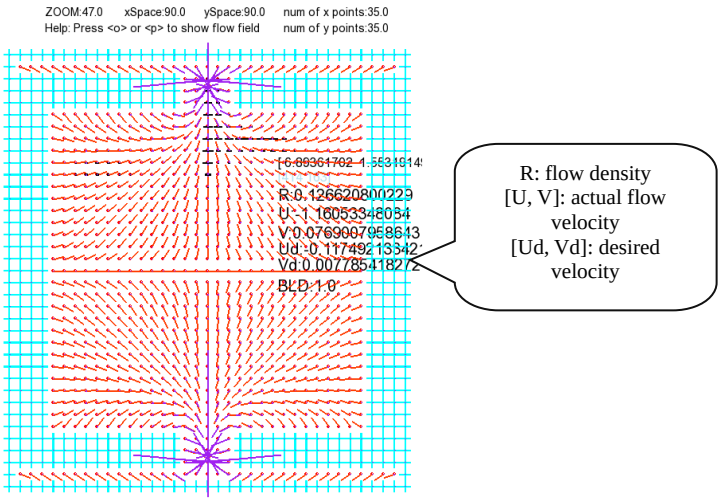


Figure 13. Computational Result of Crowd Fluid Model

The fourth panel is called SocialAgent, where users specify the basic parameters to calculate the agent-based simulation. Many parameters in this panel refers to the theoritical model of agents, and if users are interested, please browse our manuscript Wang et. al., 2024 for more details of the theoritical model.

<Selection of Short-Range Force>: Users could choose the short-range forces for agent interaction. Such forces are effective only if agents are sufficiently close to each other. As shown in Figure 15, value 0 is for social force in Helbing, I. Farkas, T. Vicsek 2000 and value 1 is for magnetic force in Okazaki, 1979.

<dtAtt>: The time interval (second) to update the attention list for all the agents. In Figure 15, for example, the time interval of dtAtt is 1.0 second, and the attention lists for all the agents are upated numerically every 1.0 second.

<dtExit>: The time interval (second) to upate the target exit for all the agents based on the exit-selection probability. In Figure 15, for example, the time interval of dtExit is 1.0 second, and all the agents will re-select the target exit numerically every 1.0 second. This parameter is not valid for solver=1, where the nearest-exit strategy is simply used, and agents do not re-select target exit dynamically.

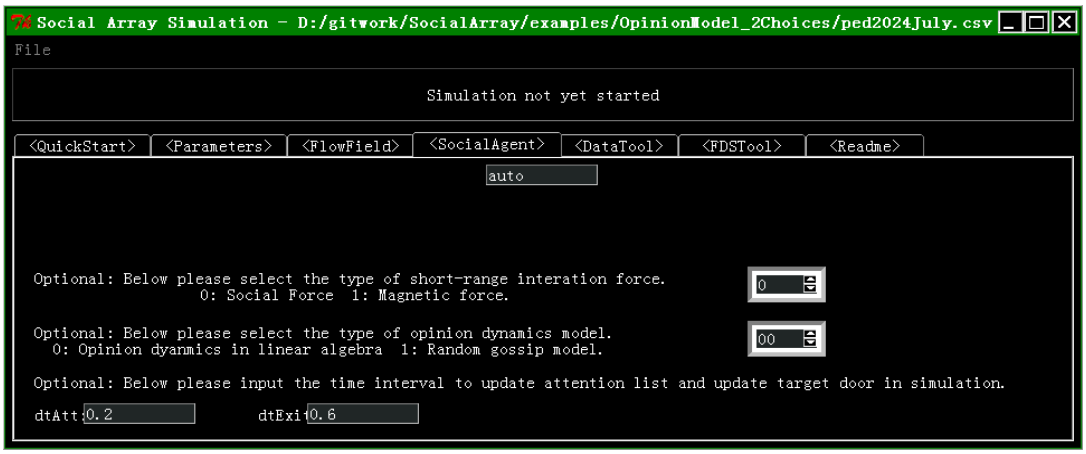


Figure 14. User Interface of SocialArray (SocialAgent)

5. Analysis and Visualization of Output Data

The output data of above agent-based simulation are classified in two forms, namely, the non-matrix data and matrix-based data. The non-matrix data is used to store agents' position velocity and forces for all the time steps. The social relationship and egress flow field are matrix-based data in our program. Both forms are also written into a text file in the simulation process. Users could easily check the text data by using a text editors. The output data files are listed as below.

Table 10. Output Data of Simulation.

Text Data	*The text file is a log of computation process and it records agents' position velocity and forces for all the time steps. It also includes other agent features such as exit selection probability, dynamics of pre-movement time, the matrices of social relationship, and so forth. In a sentse the text file contains all the data, which are included in the binary file and NPZ file.*
Binary Data	*The binary data file mainly records agents' position velocity and forces for all the time steps. It also record other non-matrix data for each agents, such as pre-movement time, stress level, exit selected and so forth. The binary data could be visualized in pygame after the simulation is complete. Use a button in the panel of SocialAgent and select the binary file (suffix of .bin file), and the data will be extracted and displayed in pygame section.*
NPZ Data	*The NPZ data file is a special form in NUMPY, and it is useful to store matrix-like data. In our program the NPZ data file is used to store the social relationship of agents as they move and interact. The computational results of egress flow fields are also saved in NPZ form. The filename are vel_flow1.npz and vel_flow2.npz for Solver 1 and Solver 2, respectively.*

The output data files for egress flow field are vel_flow1.npz and vel_flow2.npz, for Solver 1 and Solver 2, respectively. Such data files are stored in the example folders and they are not recalculated unless the compartment layout are changed. The agent-based simulation data include a binary data file, a npz data file and a text file, which all have

the same prefix name. If your input csv file is agent.csv and simulation is complete at time of 2024-03-02_23_09_27 (in format of year, month, date, hour, minute and second), the output data files will be generated as agent_2024-03-02_23_09_27.bin, agent_2024-03-02_23_09_27.txt and agent_2024-03-02_23_09_27.txt.

Users could direct open the text file by using any text editors and check the agent data there, including the agent's position and velocity and forces at each time step. Such data are 2D vectors, and in the text file we also list the vector's intensity before the vector. The text output file is illustrated as below.

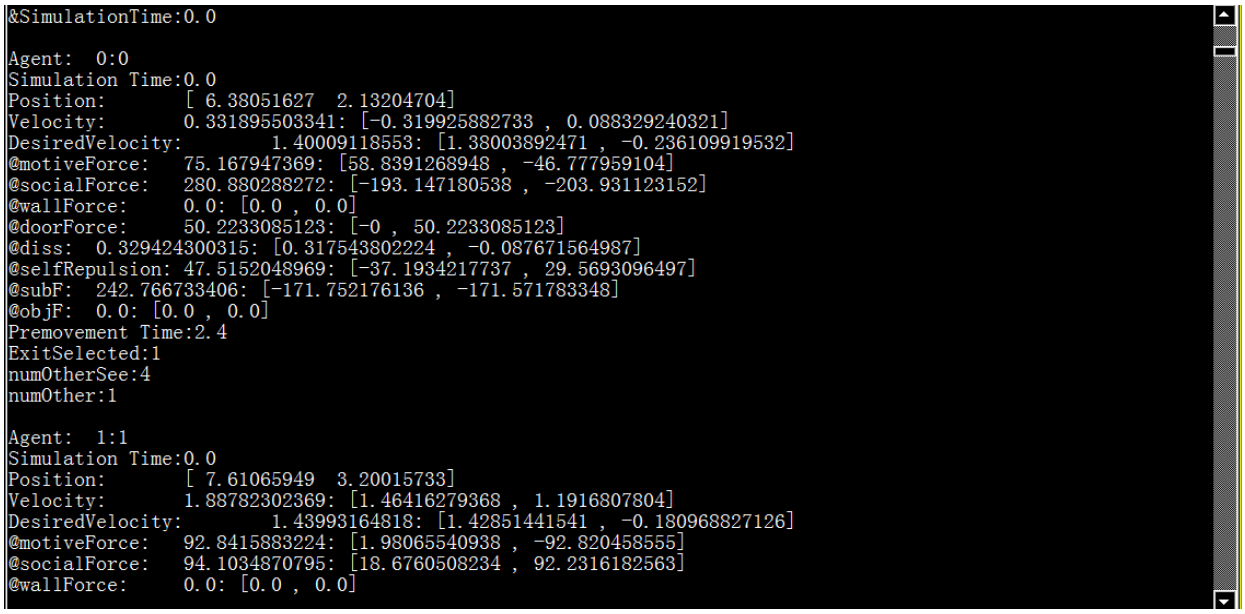


Figure 15. Output Data in Text File.

As for the npz data file, it is a special form used by numpy package in python, and it is easy to store the data array and matrix. This matrix-based data could be visualized together with the binary data file in our program.

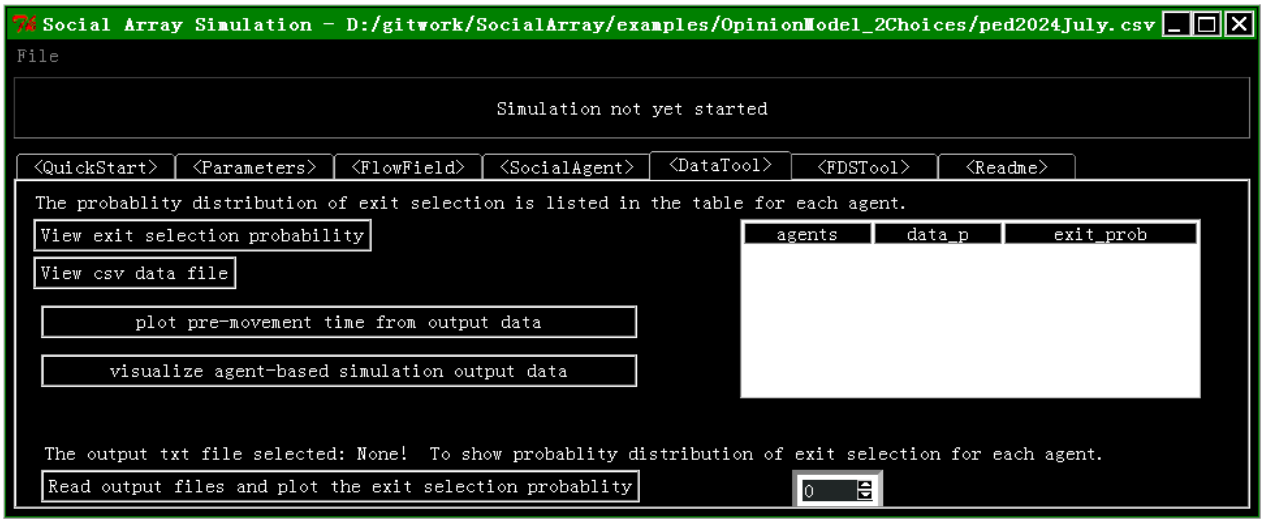


Figure 16. User Interface of SocialArray (DataTool)

<plot pre-movement time from output data>: Select the output binary file and the pre-movement time for each agent is plotted as exemplified in Figure 17. In Figure 17 each colorful line represents an individual's $tpre_i$ *and the value of* $tpre_i$ *dynamically changes as agents interact with each other. When the colorful line drops to x axis, it means that the individual reaches an exit and thus is removed from computational loop. In particular there is a*

gray slash line which has angle of π/4 to the x axis, and this gray line divides the x-y plane equally into two regions. The upper left region indicates the pre-movement phase because $tpre_i$ is larger than simulation time ($tpre_i$>t). The lower right region represents the movement phase since the simulation time t exceeds $tpre_i$ and thus individuals start to move to exits.

In particular it is noted that individual 2 (red line) sways the opinion between individual 0 and individual 3. He or she initially communicates with individual 1, implying that individual 1 is in his or her attention list, or even in talk list. Next he or she communicates with individual 3, and then back with individual 1 and 0, and again with individual 3. Finally, his or her opinion converges with individual 3, and the pre-movement time converges around 9 seconds with individual 3.

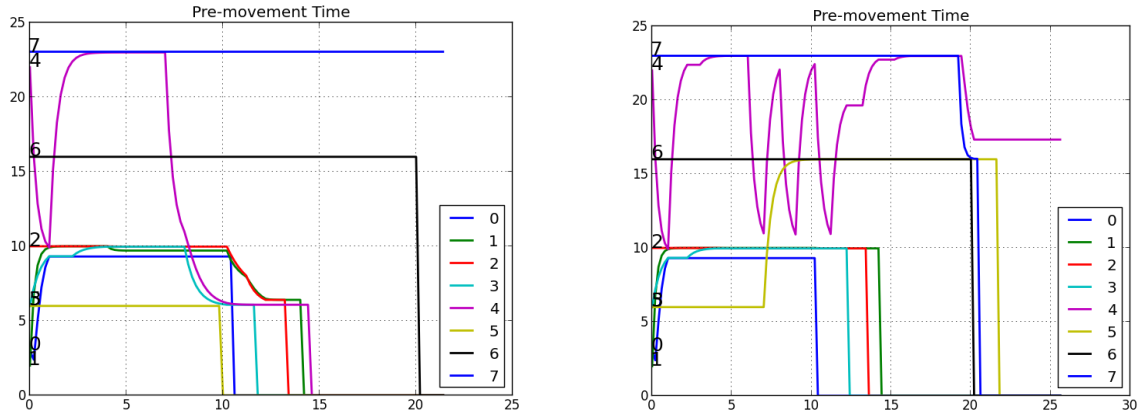


Figure 17. Visualization of pre-movement time from output data

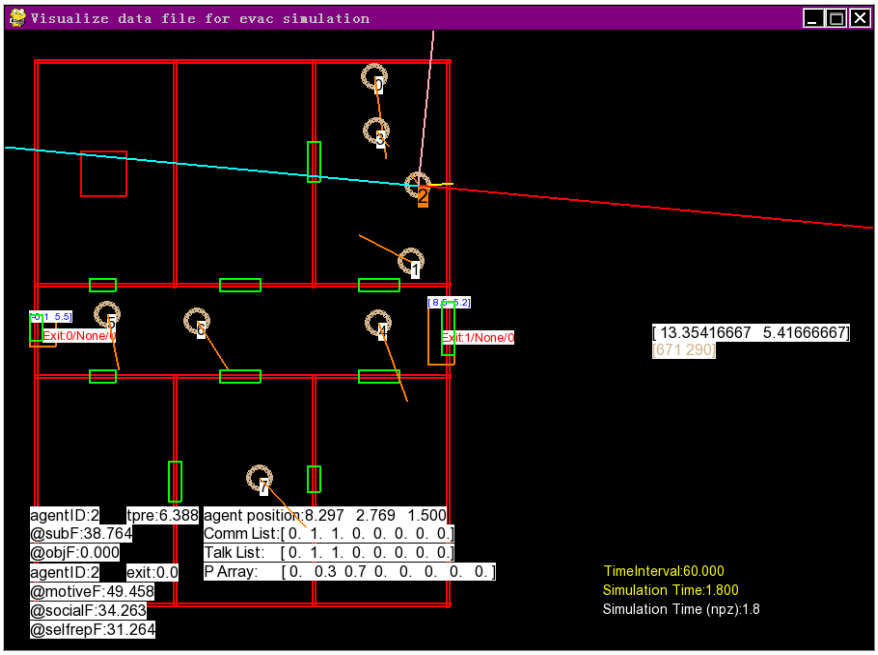


Figure 18. Visualization of simulation output data

<Visualize agent-based simulation output data>: Select the output binary file and the data will be extracted to show the simulation at each time step as illustrated in Figure 18. Users could press <Home/End> to forward or rewind the simulation process. Similar to testGeom and runSimulation as introduced before, press <pageup/pagedown> to zoom in or zoom out the entities in screen. Use space key to pause/continue the simulation process. Use direction keys to move the entities vertically or horizontally in screen.

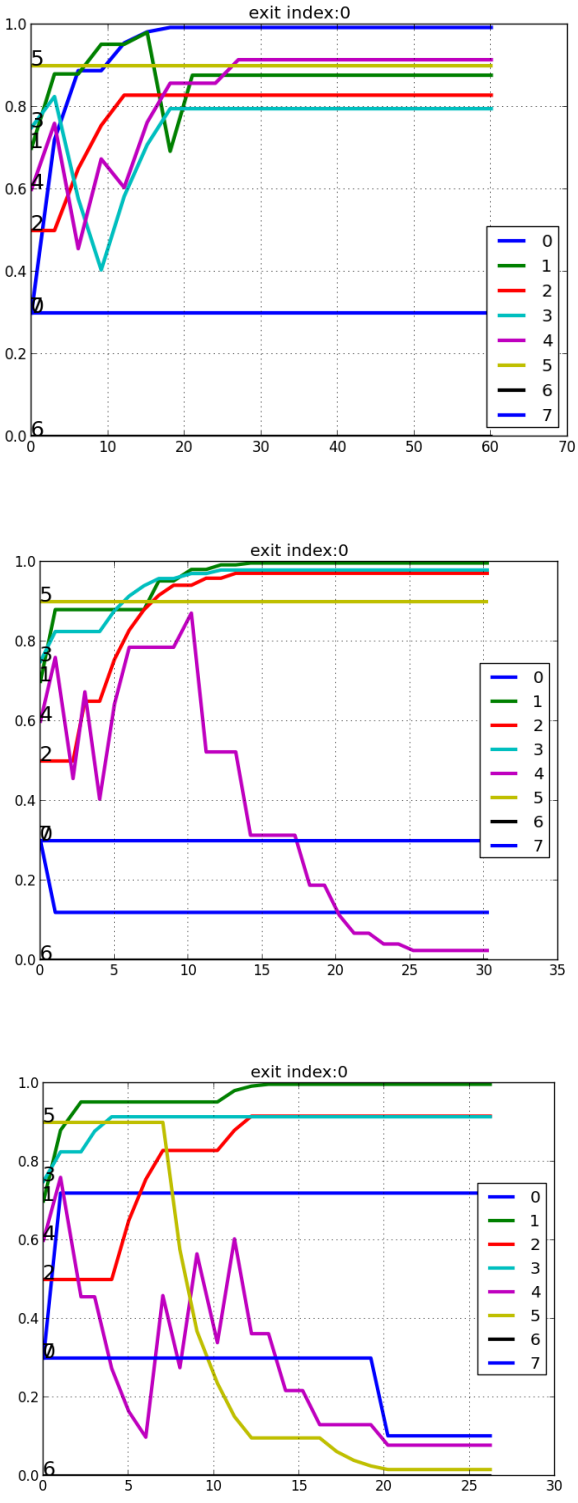


Figure 19. Visualization of Exit Selection Probability

<Visualize exit selection probability>: Select the output text file and the data will be extracted to show the exit selection probability evolves for each individual agent through the simulation process. The exits index number starts from 0, and users are able to check and find this index number in testGeom screen. In Figure 19 it seems that agent 2 and 3 have interaction in the exit selection process. For example, the probability for agent 3 (cyan line) to use exit0 is initialized with 0.75, and throughout the simulation it first drops and then increases. In addition, agent 4 (purple line) seems to sway the opinion of exit selection in different scenarios. Sometimes agent 4 goes with agent

3 to select exit 0 while sometimes goes with agent 5 to select exit 1. The social topology is important to affect how such exit selection probability evolves in timeline as individual agents interact. The social topology is expressed as shown in Figure 8. Please refer to Wang, et. al., 2022 for more details on the theoretical model.

There are several other panels including, DataTool, FDSTool and the GUI is under development. Thus, we will introduce them in the future.

Try Examples:

There are currently several examples in the repo of https://sourceforge.net/project/socialarray/ or https://github.com/crowdmodel/complexSocialOpinion.git . For instance there are standard example of a single room with two exits or three exits, which are basically used to test how agents socially interact and select different exits. The example of a room with one exit is also widely used to test crowd behavior at bottlenecks such as the faster-is-slower effect (Helbing, Farkas, Vicsek, 2000).

There are also some more complicated examples. A typical one is learned and extended from MassEgress project (Pan, 2006), and another one is from our IEEE Conference paper (Wang et. al., 2008). Some FDS+Evac test cases are also included. Users can also learn how to write the csv files from the examples.

In order to run such complicated examples it is suggested that users should first check the mesh parameters. Especially, the total number of x point and y points are to be properly given to build a mesh for flow computation. The flow solver may take some computational time to generate the egress flow field, especially if x point and y point are relatively large to generate a refined mesh.

In SocialArray simulation platform options are provided to users to either use existing FDS input files to create compartment geometries or specify them in a csv input file. In current version only one-floor crowd simulation is supported. So if there are multiple evacuation meshes in FDS input files, they should all belong to the same z interval in the vertical direction (z axis). By using FDS input files the walls are created by &OBST, and the paths are specified by &HOLE or &DOOR. The exits are obtained from &EXIT in FDS input files. If users want to find more about how FDS define a compartment area, please refer to FDS UserGuide for more information [McGrattan et. al., 2021].

If users do not use FDS input files, the above entities can alternatively be specified by using a csv file as introduced below, and users need to write data blocks in csv file, and such data blocks are identified by &Wall, &Path, &Exit and &Agent as the first element in the data block. If users import walls from a FDS input file, the walls are created as a rectangular type and it corresponds to &OBST in FDS input file.

Acknowledgments

The authors are thankful to Bo Xiong and Vivek Kant for helpful comments on earlier work in University of Connecticut. The author appreciates the research program funded by NSF Grant # CMMI-1000495 (NSF Program Name: Building Emergency Evacuation - Innovative Modeling and Optimization).

Also thank Topi for sharing his wonderful python script on Discussion Forum of FDS and thank Salah Benkorichi for sending the information to me! I mainly modified xyz.shape = (7,nplim) for evac binary data.

The program source code and numerical test cases are mainly included online at https://sourceforge.net/p/crowdegress/discussion/general/. If you have any comment or inquiry about the testing result, please feel free to contact me at wp2204@gmail.com or start an issue on the repository.

References

G. Deffuant, D. Neau, F. Amblard, G. Weisbuch, "Mixing Beliefs Among Interacting Agents," Advances in Complex Systems, Vol. 3, pp. 87-98, 2000.

W. Ebeling, F. Schweitzer, "Active Motion in Systems with Energy Supply," Integrative Systems Approaches to Natural and Social Dynamics – Systems Sciences 2000, Springer, Berlin, pp. 119–142, 2001.

D. Helbing, I. Farkas, T. Vicsek, "Simulating Dynamical Features of Escape Panic." Nature, Vol. 407, pp. 487– 490, 2000.

D. Helbing, P. Molnar, "Social force model for pedestrian dynamics," Physical Review E, vol. 51, no. 5, pp. 4282-4286, 1995.

G. P. Forney, "Smokeview, A Tool for Visualizing Fire Dynamics Simulation Data, Volume I: User's Guide", NIST Special Publication 1017-1 6th Edition, National Institute of Standards and Technology, Gaithersburg, MA, June 2022, 188 p.

S. Kim, S. J. Guy and D. Manocha, Interactive Simulation of Dynamic Crowd Behaviors using General Adaptation Syndrome Theory, Department of Computer Science, UNC- Chapel Hill http://gamma.cs.unc.edu/GAScrowd/ , 2010.

T. Korhonen, "Technical Reference and User's Guide for Fire Dynamics Simulator with Evacuation," (FDS+Evac, FDS 6.5.3, Evac 2.5.2), VTT Technical Research Center of Finland, 2016.

T. Korhonen, S. Hostikka, S. Heliovaara, H. Ehtamo, "FDS+Evac: Modelling Social Interactions in Fire Evacuation," Proceedings of the 4th International Conference on Pedestrian and Evacuation Dynamics, February 27-29m 2008, Wuppertal, Germany.

T. I. Lakoba, D. J. Kaup, N. M. Finkelstein, "Modifications of the Helbing-Molnar-Farkas-Vicsek Social Force Model for Pedestrian Evolution, Simulation, Vol. 81, Issue 5, pp. 339-352, May 2005.

K. Lewin, Field Theory in Social Science, New York, Harper, 1951.

K. McGrattan, S. Hostikka, R. McDermott, J. Floyd, C. Weinschenk and K. Overholt, "Fire Dynamics Simulator, User's Guide", NIST Special Publication 1019 6th Ed., National Institute of Standards and Technology, Gaithersburg, MA, 2022, 367 p.

R. Lovreglio, A. Fonzone, L. dell'Olio, A Mixed Logit Model for Predicting Exit Choice during Building Evacuations, Transportation Research Part A, 2016. DOI: 10.1016/j.tra.2016.06.018.

X. Pan, C. S. Han, K. Dauber, and K. H. Law, "Human and Social Behavior in Computational Modeling and Analysis of Egress," Automation in Construction, Vol. 15, pp. 448-461, 2006.

A. F. Peralta , J. Kertesz, G. Iniguez, Opinion dynamics in social networks: From models to data, 2022.

L. A. Quang, N. Jung, E. S. Cho, J. H. Choi and J. W. Lee, Agent-Based Models in Social Physics, Journal-Korean Physical Society, 2018. DOI: 10:3938/jkps.72.1272.

S. Okazaki, A Study of Pedestrian Movement in Architectural Space, Part 1: Pedestrian Movement by the Application on of Magnetic Models. Trans. of A.I.J., No.283, pp. 111-119, 1979.

M. Owen, E.R. Galea, P.J. Lawrence, The Exodus Evacuation Model Applied to Building Evacuation Scenarios, Journal of Fire Protection Engineering, 8(2) : 65-86 (1996).

F. Ozel, "Time Pressure and Stress as a Factor During Emergency Egress," Safety Science, Vol. 38, pp. 95-107, 2001.

E. Ronchi, R. Lovreglio, M. Kinsey, A Survey on Evacuation Model Awareness, Usage and Users, 2020.

G. Santos and B. E. Aguirre, A Critical Review of Emergency Evacuation Simulation Models, NIST Workshop on Building Occupant Movement during Fire Emergencies, June 9-10, 2004.

M. A. Staal, "Stress, Cognition, and Human Performance: A Literature Review and Conceptual Framework (NASA/TM – 204-212824)," August 2004, Hanover, MD: NASA Scientific and Technical Information Program Office.

Y. Tong and N. W. F. Bode, The principles of pedestrian route choice, Journal of The Royal Society Interface, Vol. 19: 20220061. 2022, https://doi.org/10.1098/rsif.2022.0061.

T. Vicsek, A. Czirok, E. Ben-Jacob, I. Cohen, O. Shochet, "Novel type of phase transition in a system of self-driven particles,'' Physical Review Letters 75. pp. 1226-1229. 1995.

P. Wang, "Understanding Social Force Model in Psychological Principle of Collective Behavior," Master Thesis, 2016. arXiv:1605.05146v10.

P. Wang and X. Wang, "Social Force in Pedestrian Crowd," arXiv:2105.05146v1 [physics.soc-ph], 2021.

P. Wang, X. Wang, P. B. Luh, Christian Wilkie, Timo Korhonen, Neal Olderman, "Simulation of Crowd Egress with Environmental Stressors, Technical Report, arXiv:2206.01393v6, [physics.soc-ph], May 2022.