J Mol Model (2019) 25: 5
https://doi.org/10.1007/s00894-018-3889-z

SOFTWARE REPORT

# A simulation environment for polymeric nanoparticles based on multi-agent systems

**Alexandre de O. Zamberlan[1] · Guilherme C. Kurtz[1] · Tomas L. Gomes[1] · Rafael H. Bordini[2] · Solange B. Fagan[1]**

Received: 28 May 2018 / Accepted: 3 December 2018 / Published online: 18 December 2018


**Abstract**

Production and characterization of polymeric nanoparticles, as colloidal dispersions, are processes that require time and technical skills to make the results accurate. Computational simulations in nanoscience have been used to help in these processes and provide agility and support to reach results: stability and quality in dispersions. Multi-Agent System for Polymeric Nanoparticles (MASPN) is an innovative and original simulation environment with features to demonstrate interactions of particles from physical-chemical parameters, ensuring Brownian motion of particles and attractive and repulsive behaviour. The MASPN environment has been designed and has been built according to the feature-driven development (FDD), as software methodology, and a multi-agent systems approach. In addition, we have used the event-driven simulation package *algs4*, the JASON agent building environment, all integrated by Java language. This paper aims to present the relation of the *algs4* package and the JASON tool, both integrated into the MASPN environment to generate Brownian motion with elastic and inelastic collisions. The MASPN environment as a simulation tool emerges as a result, including the following features: graphical interface; integrated physical-chemical parameters; Brownian motion; JASON and *algs4* integration; and distribution charts (size, zeta potential, and pH).

**Keywords** Simulation software · Artificial intelligence · Nanotechnology · Particle collisions · Cooperative systems

## Introduction

Nanoscience is the study of phenomena and manipulation of materials at atomic, molecular, and macromolecular scale, in which the properties differ significantly from those on a larger scale. Nanotechnology is concerned with the design, characterization, production, and application of nanoscale structures ($10^{-9}m$) [1].

Nanoscience and nanotechnology (NN) use computational resources in the production and characterization of nanoparticles, such as nanowires, nanocarriers, nanosensors, etc. An example of a computational resource is event-driven simulation [2], which could be used to observe (and even predict) the behavior of nanostructured systems, such as the agglomeration effect. This effect indicates instability in nanoparticulate systems, given that the structures no longer have inherent nanoscale properties. Nanoparticles can carry a pharmaceutical load and, because of their size, can escape the detection of the immune system [3]. On the other hand, if a dispersion formed by nanoparticles presents agglomeration, the system will be unstable [4].

Research in computing science has generated several methods based on behavior derived from nature, particularly in multi-agent systems (MAS) that support intelligent collective behavior [5]. Thus, MAS simulation appears as an alternative approach to investigate the agglomeration effect of nanoparticles, since particles and environment of a nanostructured system have characteristics and functionalities that are suitable for modeling and implementation as MAS

✉ Alexandre de O. Zamberlan
alexz@ufn.edu.br

Guilherme C. Kurtz
guilhermekurtz@ufn.edu.br

Tomas L. Gomes
tloureirog@gmail.com

Rafael H. Bordini
rafael.bordini@pucrs.br

Solange B. Fagan
sfagan@ufn.edu.br

[1] Universidade Franciscana, Santa Maria, RS, Brazil

[2] PUCRS, Porto Alegre, RS, Brazil

simulations. An important point in research in the very small scale universe is the capacity for self-organization of matter, similar to what happens with biological entities that are often modeled in MAS, i.e., mechanisms respond to stimuli from the environment without conscious control [1].

This work presents the integration of the multi-agent system for polymeric nanoparticles (MASPN) with multi-agents techniques and event-oriented paradigm to support in the observation of agglomeration effects in polymeric nanoparticles (PNPs). So, the focus is the integration of the *algs4* package with JASON and particle interactions according to Brownian motion.

The reason to work with PNPs, nanocapsules, is because they are used for drug delivery in several treatments. In addition, PNPs are generally coated with non-ionic surfactants, which reduce immunological interactions and intermolecular interactions between chemical groups of different PNPs. Polymers must protect the drug from degradation or metabolism, promoting sustained release of the drug [4].

The research presented can be considered innovative and interdisciplinary: there is no MAS simulation of polymeric nanoparticle systems, and the proposed tool has the flexibility of the simulation because it is possible to deal empirical parameters such as zeta potential and size (particle), and pH (environment).

The paper is structured as follows. The Background section introduces some important concepts around nanoparticles and simulations. Sections "Development of the simulation tool" and "Results and discussion" present the research methodology and some images of the MASPN tool as achieved results, respectively. Finally, conclusions and discussions for future work are presented.

## Background

MASPN uses input parameters related to the agents (particles) and the environment (dispersion). Size, electric charge surface (or zeta potential), and drug content are the particles parameters, while pH is parameter of the environment. MASPN embraces concepts of event-driven simulation and particle collision system. In addition, all of these are integrated with a multi-agent system approach.

In this section, related work is also discussed. No relevant work was found which used MAS applied in the simulation of bodies in the areas of biology and chemistry in nanoscience.

### Physico-chemical parameters

Particle size and size distribution (percentage of particles with specific size) are important characteristics of nanoparticle systems [6]. Zeta potential is used to characterize the charge property of a nanoparticle's surface and reflects the particle's electric potential. It is related to the particle composition and the medium in which it is dispersed [7]. A particle is electrically charged when it has a small amount of unbalanced charge. Electric force between two charges with the same sign is repulsive, while the electric force between two charges with opposite signs is attractive. Thus, charged particles are stabilized when there is repulsion, and they will be agglomerated when there is attraction [4].

Stability of nanoparticulate systems may be obtained by monitoring the pH over time. The pH of dispersion determines the zeta potential of colloidal dispersions, which may have an impact on their stability. The pH of the dispersion can be an important element which to control the nanoparticles size and, consequently, its biodistribution [6]. Association rate (or drug content) refers to the amount of drug associated with nanoparticles [8]. Drug content influences the agglomeration effect (or instability of concentration), because the nanocrystals formed in formulations agglomerate and precipitate during storage of nanocapsules in the course of time [9].

Due to the nanoscale, atoms are inter-facial, due to the high surface-to-volume ratio, the relationship of physico-chemical parameters to agglomeration behavior indicates a migration from volume chemistry to surface chemistry.

### Event-driven simulations - EDS

Computer science provides several resources to assist in the measurement and manipulation of nanomaterials. Thus, computational nanotechnology contributes to molecular modeling, nanostructural simulation, and nanoinformatics, which benefits from computing technology, for example high-performance computing. Molecular modeling involves computational methods and techniques to simulate the behavior of atomic and molecular systems [10].

There are different simulation methods, such as Monte Carlo (stochastic behavior), Brownian dynamics, and molecular dynamics (both with dynamic behavior). Molecular dynamics efficiently evaluates different structural properties and dynamic quantities that cannot be obtained by Monte Carlo, because it has no temporal correlation [11]. In general, Brownian dynamics is considered an efficient method for simulation of large polymeric molecules or colloidal particles [2].

The event-driven simulation paradigm is appropriate to address the motion of molecules in gas, dynamics of chemical reactions, atomic diffusion, one-dimensional gravitational systems, etc. [2]. Scientists use such systems to understand and predict properties of physical systems. For example, Einstein used the model to explain the Brownian

motion of pollen grains immersed in water [2]. Hence, EDS applications recognize molecular dynamics because this paradigm suggests that the hard body model of interaction between particles obey to the following [2]:

1. a defined structure: $n$ moving particles confined in a box, where each particle has position, velocity, mass, charge, and radius;
2. a predicted physical behavior: particles interact by elastic and inelastic particle–particle collisions and particle–wall elastic collisions;
3. other existing forces (or not): environment or box (solvent) is a continuous viscous medium that dissipates energy while the particle (solute) moves through it.

In simulation systems for polymeric nanoparticles, inelastic collisions occur when bodies of opposite electrical charge collide (attraction process) and elastic collisions occur when bodies with electric charge with the same signal collide (repulsive process). These collisions are influenced by elements such as zeta potential and size particle, and environment pH. There are three equations to coordinate collisions of hard bodies [2]:

1. conservation of linear momentum;
2. conservation of kinetic energy;
3. after a collision, the normal force acts perpendicular to the surface of the collision point.

## Elastic and inelastic collisions

Perfectly elastic collision is defined when there is no loss of kinetic energy[1], in other words, there is no dissipation of mechanical energy and the kinetic energy of the objects remains the same after impact [12].

When a collision occurs, and the kinetic energy of the system is not conserved (there is mechanical energy dissipation), the concept of inelastic collision arises [13]. If there is no action of external forces on the system, the principle of conservation of linear momentum[2] will be obeyed, although there is no conservation of kinetic energy. In isolation, as in colloidal systems (the focus of this study), this dissipation occurs due to some properties of the collided bodies, such as structure and molecular composition.

Equations 1 and 2 illustrate the velocity of two bodies after inelastic collision in one dimension, in other words, components in the same line to the plane at the point of contact.

$$v_a = \frac{C_R m_b(u_b - u_a) + m_a u_a + m_b u_b}{m_a + m_b} \quad (1)$$

$$v_b = \frac{C_R m_a(u_a - u_b) + m_a u_a + m_b u_b}{m_a + m_b} \quad (2)$$

where $v_a$ is the final velocity of the first body after impact, $v_b$ is the final velocity of the second body after the collision, $u_a$ is the initial velocity of the first body before the collision, $u_b$ is the initial velocity of the second body before impact, $m_a$ is the mass of the first body, $m_b$ is the mass of the second body, $C_R$ is the resolution coefficient, where 1 is elastic collision and 0 is collision perfectly inelastic.

For two-dimensional and three-dimensional collisions, the velocities in the formulas of Eqs. 1 and 2 are the perpendicular components to the tangent line or plane at the point of contact.

In inelastic collisions, the collided particles remain together, that is, the kinetic energy is lost joining two bodies. In addition, it is necessary to consider the conservation of the linear momentum, which occurs if the surface has friction equal to zero[3]. However, partially inelastic collisions are the most common form of collisions in the real world, because the objects involved in collisions do not bind, but some kinetic energy is still lost [12].

## Multi-agent system

The object-oriented paradigm provides resources for event-driven simulations, with frameworks or packages with numerous functionalities. However, in a simulation system in which particles interact by exchanging information between them and perceiving the environment (feedback), the MAS paradigm becomes a very attractive alternative: "...the simulation studies the modeling of the operation of a physical or conceptual system over time and for more than 20 years, the MAS field and the simulation field were combined in lines of research..." [14].

An agent internal architecture is associated with the type of task the agent will perform and its role in the multi-agent society. Thus, what defines an agent and the society where it is situated are the interactions with the environment and the internal processes that make possible the accomplishment of these interactions [5]. Different architectures have been proposed in order to specify the agents (and the society in which they are inserted) with a level of intelligence and autonomy. In this way, the architectures can be classified

[1] Kinetic energy is energy that is related to the state of motion of a body. If a body is at rest, the kinetic energy is zero.

[2] In classical mechanics, the conservation of linear momentum is implied by Newton's laws and is the result of the multiplication of mass by the velocity of a body. In addition, the conservation of linear momentum (with modified formula and appropriate definitions) is valid in electrodynamics, quantum mechanics, quantum field theory, and general relativity [13].

[3] In event-driven simulations in nanoparticulate colloidal systems, the surface (environment) is an aqueous medium that friction capacity is zero.

according to the mechanism used by the agent to select actions [15]. Definitions and properties that characterize the notion of agency do not divide the world between entities that are agents (or not), but serve as tools to analyze systems, as well as to specify, design, and implement systems whose basic elements are agents [16].

Simulation environments for nanostructures can be considered essentially reactive, that is, suitable for reactive agent architectures. A significant feature in agent-oriented theory is autonomy, which also exists in small-scale structures such as atoms and molecules, despite the strong interaction [17]. Regarding the organization of a reactive or cognitive multi-agent system, there are events, constraints, and interactions that also occur in a nanoscale environment.

In agent-based computing, the behavior of an agent occurs through the cycle of perceiving, planning, and acting [18], very similar to what occurs in event-oriented simulations. For example, a particle can be considered an agent that interacts with other particles and the wall of a box. The box is the environment containing the solvent (which would dissipate or not the interaction energy of the particles). In this way, the particles must perceive other particles, the walls of the box, and the environment solvent. Next step, the particle plans next actions calculating the elastic or inelastic collision forces, and act updating trajectory and velocity data.

### Related work

Drug delivery research can be considered one of the most promising areas in nanoscience investigations. Ntika, Kefalas, and Stamatopoulou [3] show that drug delivery and MAS areas can be combined. The relationship with this research is mainly due to the bibliographical references used for simulation building tools, and statistical model for the evaluation process, as well as the layers of abstractions.

In the research discussed in [18], it was evaluate the behaviour of interacting particles obeying a dynamics of self-organisation, as in a hematopoietic system (intracellular processes and events interacting among cells).

In the research conducted by Hogg [19], groups of robots were collectively controlled (individual versus collective behavior in response to the environment and other agents). Generally, computational studies of controlling groups of robots complement studies to individually controlled one. This can be extended to a system with particles that exhibits coordinated behavior in response to different stimuli, namely agglomeration effect generated by physical-chemical properties of the environment, such as pH.

The study discussed by Dan [20] brought contributions in three points: (i) how simulations were conducted; (ii) how size distribution and particle positioning were implemented; and (iii) how specific particle and environmental parameters were manipulated.

Finally, the research presented in this work can be considered innovative and interdisciplinary. First, there is no MAS simulation for nanoparticulate systems. Secondly, the flexibility of the simulation is based on empirical input parameters such as zeta potential and size (particle) and pH (environment).

## Development of the simulation tool

Computational resources (methodologies, languages, and technologies) used for the construction of the MASPN were:

1. Feature-driven development (FDD): an iterative and incremental software development process. It is an agile method for developing software. There are five processes, divided into two phases (design and planning; construction). The first phase has three processes in order to develop an overall model by listing and planning the features and system requirements. The second phase discusses the construction of the project, always interactively, incrementally, and using visual diagrams [21];
2. *algs4*[4] package: implementations of various functions for scientific computing and event-driven simulation;
3. JASON[5]: multi-agent systems development platform, with many user-customizable features. It is an interpreter for the AgentSpeak(L) language [22];
4. NetBeans[6]: a software development platform written in Java. The NetBeans platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans platform, including the NetBeans integrated development environment (IDE), can be extended by third-party developers.

The FDD methodology was selected for dealing well with heterogeneous systems integration. This is mainly because it is a flexible and customizable methodology.

From the classes and methods available in the *algs4* package, implementations were made to define the best parameters of these methods, as well as the most adequate ones, guaranteeing greater fidelity in the animations of interactions among particles (Brownian motion). The *algs4*

[4] http://algs4.cs.princeton.edu/code/javadoc/

[5] http://jason.sourceforge.net/wp

[6] https://netbeans.org/

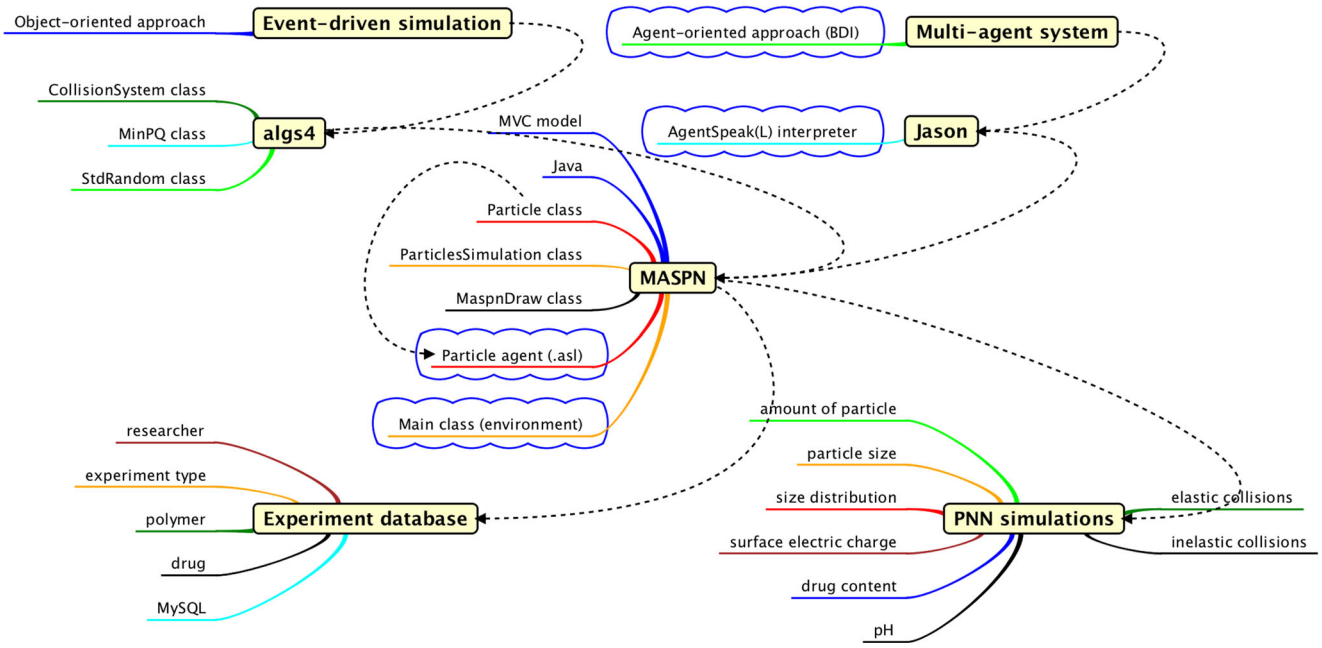


**Fig. 1** MASPN mind map: technologies, resources, and properties

package has classes for event-driven simulation and to determine the ordered sequence of collisions. The *MinPQ* class is used to create a priority queue for future events, ordered by time. For collision resolution, the package has physical formulas that specify the behavior of the particle after a particle–particle or particle–wall elastic collision (conservation of linear momentum, conservation of kinetic energy, and after a collision, the normal force acts perpendicular to the surface of the collision point).

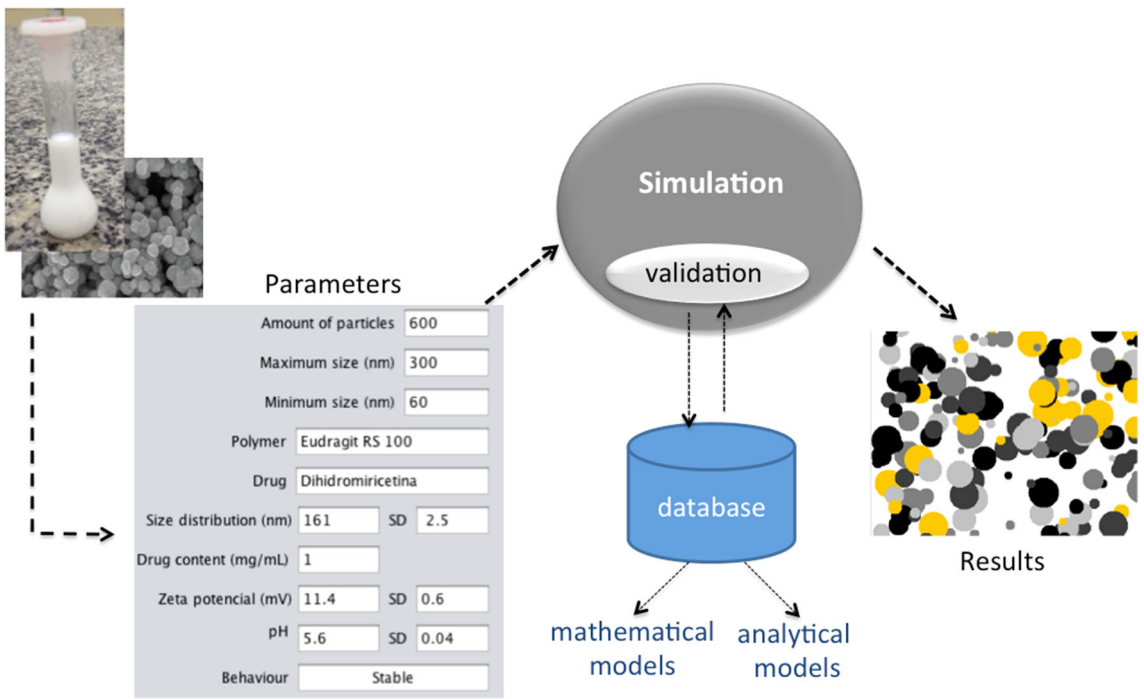


**Fig. 2** Process flow for agglomeration evaluation [17]

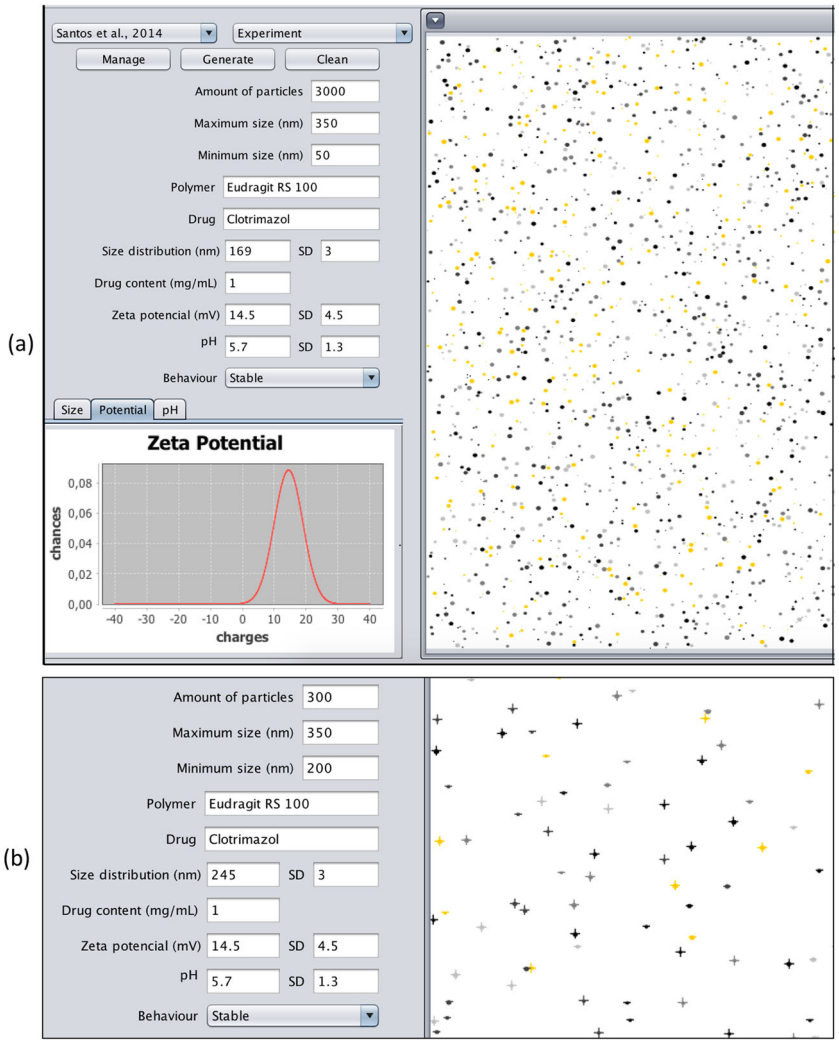


**Fig. 3** **a** MASPN main interface. **b** MASPN showing the absolute charge of the particles

Several agent-based systems development tools have been made available. Each one has particular features and functionalities. There are comparative studies that evaluate performance, robustness, usability, etc. [23]. JASON was chosen because:

1. it can be easily integrated with the Java technology;
2. it allows distributed and parallel programming;
3. it is the interpreter of the AgentSpeak(L) that obeys the logical paradigm with declarative programming;
4. there is continuous update and maintenance process;
5. it has examples in several domains;
6. there is a collection of tutorials and documentation that supports new projects.

Technologies, features, and properties involved in the construction of MASPN can be seen in Fig. 1.

## Results and discussion

As mentioned, MASPN manages experiments, performs simulations, generates distribution charts and performs 2D animations. Figure 2 illustrates how the process takes place, from the production and characterization phases to the evaluation phase of colloidal dispersion results. MASPN users can change values in the parameters and can follow the updated behavior of the particles. The parameters shown in the figure are the same as those used in the simulations of the built tool. We decided on a reduced model to aid in the evaluations of the initial simulations.

Figure 3a shows MASPN main interface in which it is possible to enter values to the parameters or select values embedded in experiments from scientific papers. After that, it is possible to generate a simulation and observe

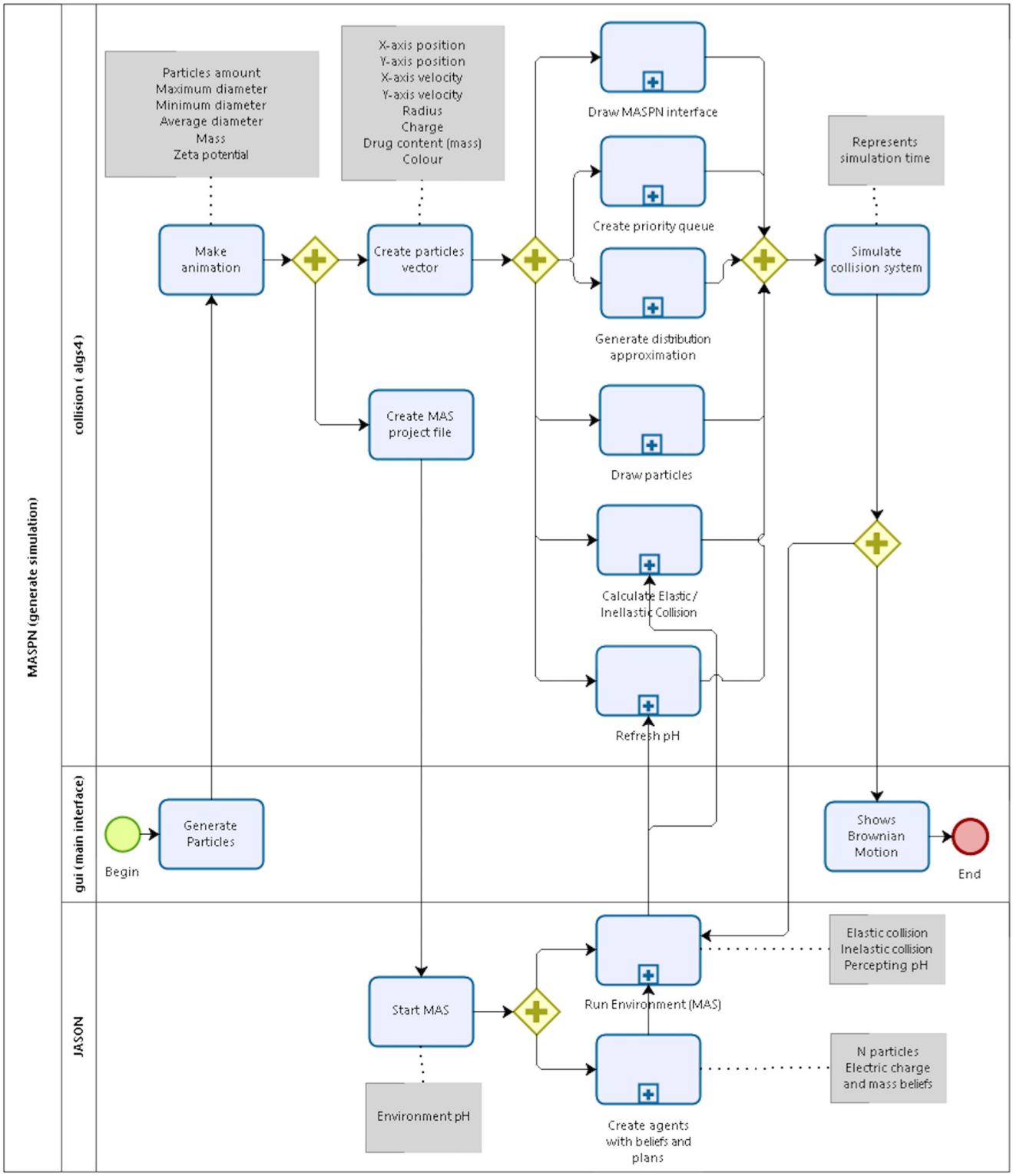


**Fig. 4** MASPN process modelling diagram

interactions of particles according to Brownian motion (elastic and inelastic collisions).

MASPN has some features implemented. The *Manage Experiments* and *Generate Graphics* features were detailed in [17]. The *Generate Size Distribution* and *Generation Position Particles* features were implemented according to the object-oriented approach and the *algs4* package methods. The

**Fig. 5** MAS project

```
MAS maspn {

    infrastructure: Centralised

    environment: MAS

    agents:
        particle1   particle.asl   [beliefs="charge(12),mass(1)"];
        particle2   particle.asl   [beliefs="charge(10),mass(0.8)"];
        //...
        particle_n  particle.asl   [beliefs="charge(13),mass(0.9)"];

    aslSourcePath:
        "src/asl";
}
```

*Pattern Recognition* feature was designed but not yet implemented.

Once an experiment has been chosen to be simulated in MASPN, the main process obeys the flow shown in Fig. 4. The process has three levels: (i) GUI (Graphical User Interface); (ii) collision; (iii) JASON.

The main process begins in the GUI with the "Generate Particles" task. Here, the user triggers the simulation in the graphical interface. The next action is to "Make (or create) animation", with information about particle amount, maximum and minimum particle diameter, mean distribution, mass, and zeta potential of the particles. In parallel, two tasks are triggered: "Create particles vector" and "Create MAS project file".

1. The "Create particles vector" task contains the ParticleSimulation class (Fig. 6 shows the class diagram), in which all the parameters of the main interface are received so that the Brownian motion of the particles is executed. For each particle parameter, such as position in the Cartesian coordinate system, velocity, mass and radius, the *uniform* method from the *StdRandom* class in the *algs4* package is used. Then, the six sub-processes are executed concurrently: "Draw MASPN interface"; "Create priority queue"; "Generate distribution approximation"; "Draw particles"; "Calculate Elastic/Inelastic collisions"; "Refresh pH";
2. The "Create MAS project file" task triggers the beginning of the MAS with pH values. Here, we generate a project file (*mas.mas2j*), which is the main JASON project file. Figure 5 illustrates what the project file looks like. This point is fundamental, since each particle has specific properties that must be mapped as beliefs. In this example, the instantiation of each

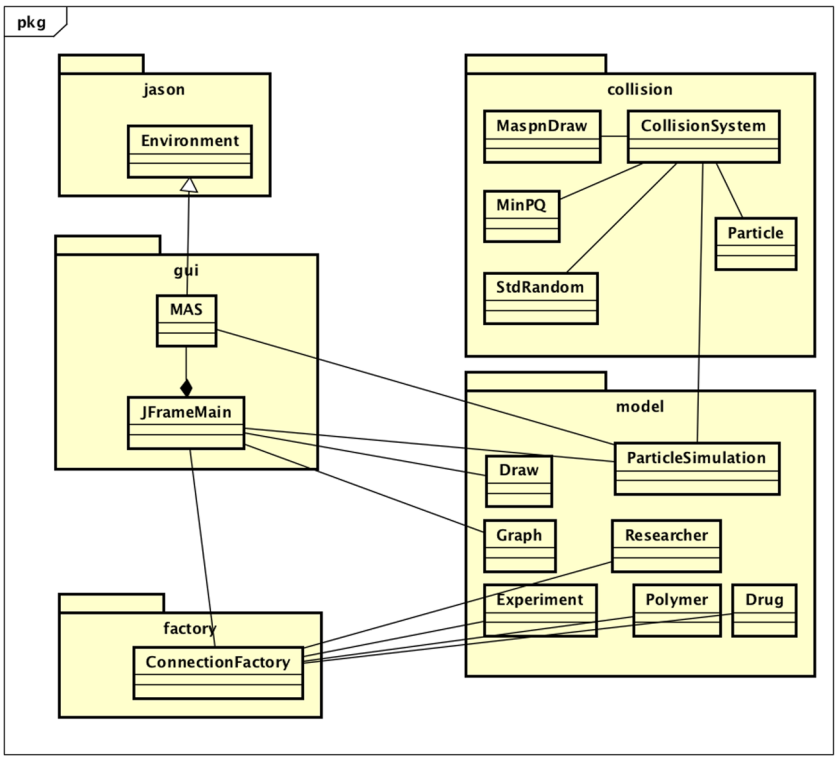


**Fig. 6** Class diagram for MASPN

**Table 1** Comparative performance: amount of particles versus maximum size

| | Maximum size (nm) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Computer 1 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 |
| amount of particle (∼) | 10000 | 8000 | 5400 | 3400 | 2500 | 2000 | 1500 | 1300 |
| Computer 2 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 |
| amount of particle (∼) | 1200 | 8000 | 5500 | 3500 | 2700 | 2000 | 1600 | 1300 |

particle with charge and mass is done, and both values are inserted as beliefs of agents.

After building a particle, it is inserted into the vector. Finally, the vector is sent to simulation ("Simulate collision system" task). Brownian motion simulation is initiated, obeying a maximum execution time.

Once the multi-agent system is started, the values of the physical-chemical parameters must be sent to the environment class (MAS) and to the agents in the MAS. The "Run Environment (MAS)" process ensures that particle control will not be centralized but distributed to agents, including collisions control (elastic and inelastic). This process is responsible for triggering the initial events of the simulation. It is also responsible for receiving and executing all actions sent by the agents (particles) to be performed in the environment. The multi-agent system adds the pH perception to the agents, as it is an environment parameter. In addition, always at the end of the *executionAction* method (within the "Run Environment MAS" task), there is a method call to check if the pH has been updated ("Refresh pH").

The *Particle* class (within the "Draw particle" task) represents a particle moving in the unit box, with a given position, velocity, radius, and mass. Methods are provided to move the particle and predict and resolve elastic collisions between other particles and vertical or horizontal walls. This data type is mutable because position and velocity change. The *CollisionSystem* class (within the "Draw particle" task) represents a collection of particles moving in the unit box, according to the laws of elastic collision. This event-based simulation relies on a priority queue. The *MinPQ* class (within the "Create priority queue" task) represents a priority queue of generic keys. It supports the usual insert and delete-the-minimum operations, along with methods for peeking at the minimum key, testing if the priority queue is empty, and iterating through the keys. This implementation uses a binary heap. The insert and delete-the-minimum operations take logarithmic amortized time [2]. The *min*, *size*, and *is-empty* operations take constant time.

Construction takes time proportional to the specified capacity or the number of items used to initialize the data structure. The *StdRandom* class (within the "Draw particle" task) provides static methods for generating random numbers from various discrete and continuous distributions, including Bernoulli, uniform, Gaussian, exponential, Pareto, Poisson, and Cauchy [2].

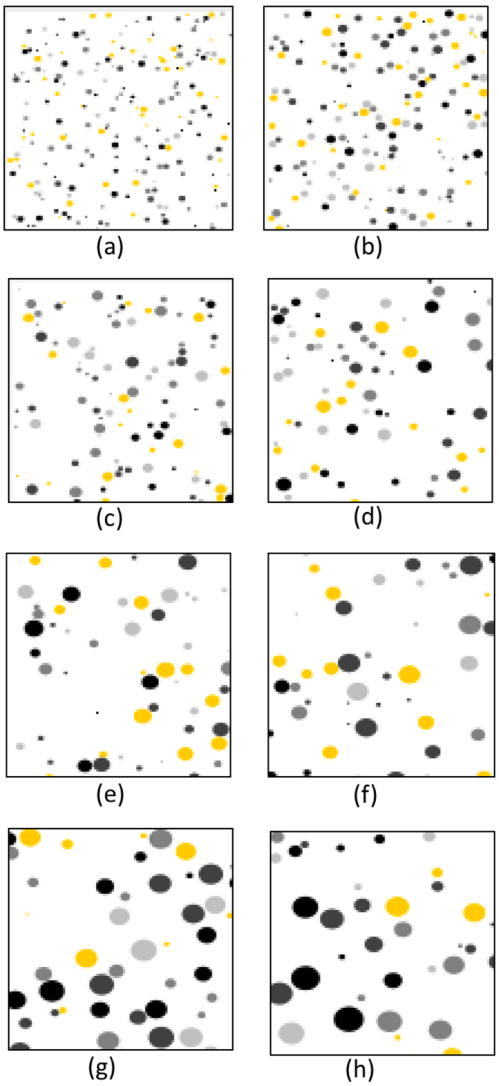


**Fig. 7** Amount of particles versus maximum size: (**a**) 10000 - 200 nm; (**b**) 8000 - 300 nm; (**c**) 5400 - 400 nm; (**d**) 3400 - 500 nm; (**e**) 2500 - 600 nm; (**f**) 2000 - 700 nm; (**g**) 1500 - 800 nm; (**h**) 1300 - 900 nm

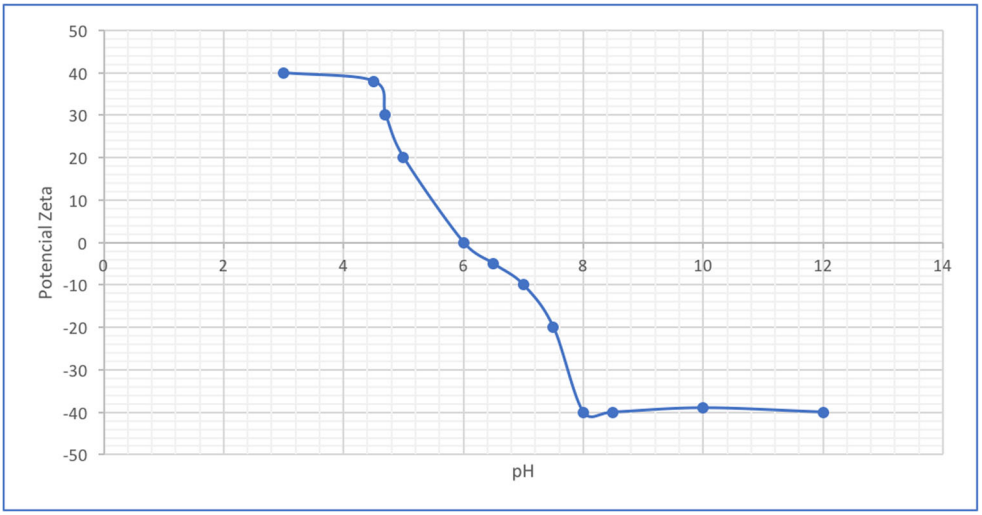


**Fig. 8** Behavior of zeta potential from pH variation

The charge parameter uses the *Gaussian* method because it takes into account the standard deviation. The *uniform* and *Gaussian* methods are static methods for generating random numbers (continuous distributions). At this point, Fig. 3b shows the substitution of the circle of each particle by the absolute charge of the particle; thus, proving that particle distribution and their charges are in agreement with the simulation model. If the zeta potential distribution graph is observed, it is possible to notice the equivalence.

The class diagram (Fig. 6) provides an overview of the package organization, classes, and the dependency relationships among them. In the diagram, it is possible to see the MASPN integration with JASON and *algs4*. At this point, we emphasize that the diagrams have mostly a conceptual function.

Some limitations have occurred, for example the maximum number of agents instantiated. For initial performance testing, we have used two computers:

1. Computer: Mac-book Pro, 2.7 GHz, Intel i7 processor, 8 GB 1867 MHz DDR3 memory and Intel Iris Graphics 6100 1536 MB;
2. Computer: Avell Titanium, 2.6 GHz, Intel i7 processor, 16 GB 2133 MHz DDR4 memory and NVIDIA GFORCE GTX 950M 2048 MB.

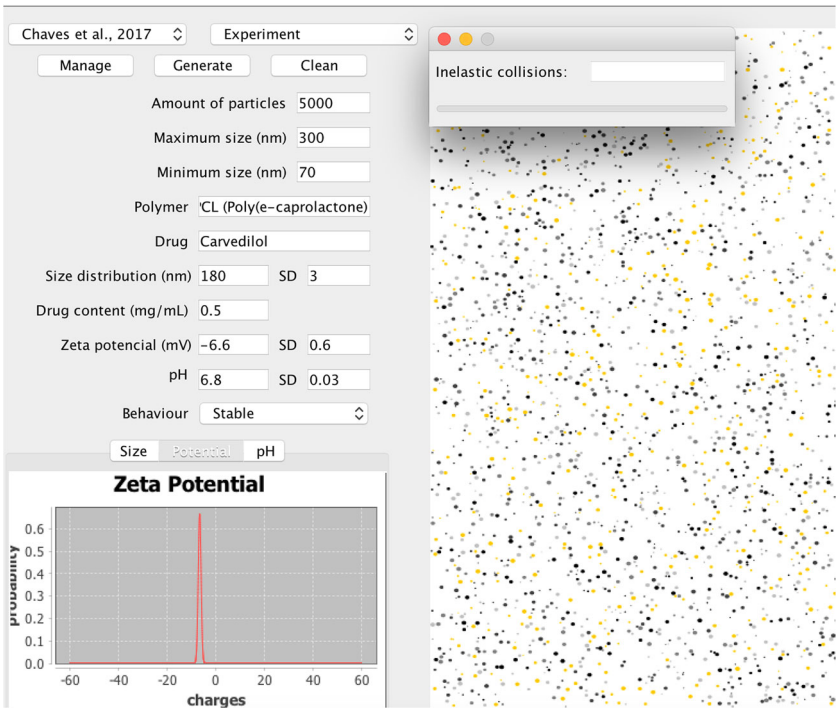


**Fig. 9** Experiment published in [26], with original values

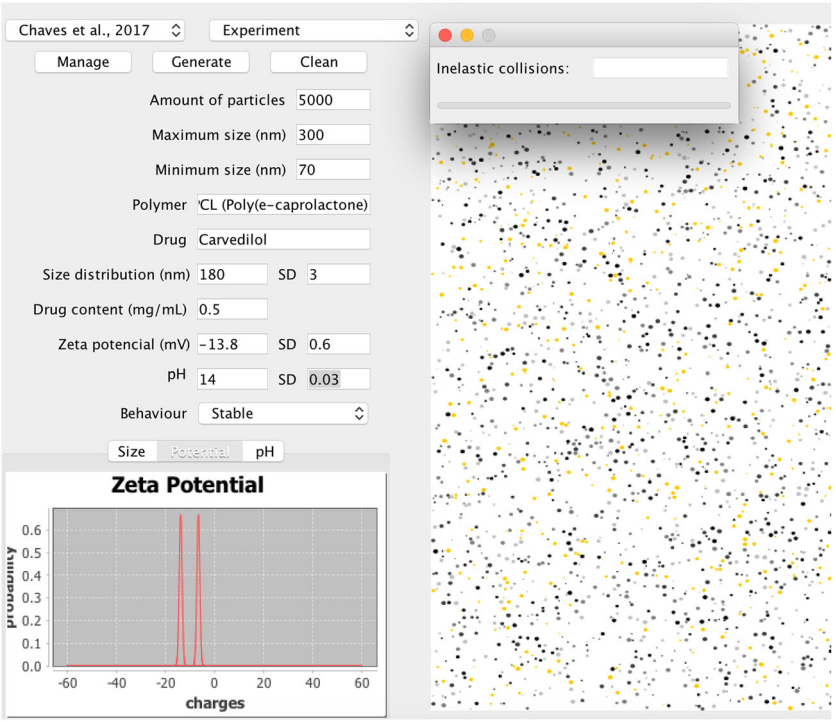


**Fig. 10** Experiment published in [26], with pH value equal to 14

All results are shown in Table 1.

Regardless of the equipment, the result is very similar. We believe that the NetBeans tool, when allocating memory to the program, tries to assign a RAM memory default value. For this reason, the results are similar on both computers. We have evaluated the ratio of particle size (mean size) versus amount of particles, and we realized that the lower the average particle size distribution, the more particles can be handled. So, the ability to handle particles at the nanoscale is conditioned by the average particle size.

The images in Fig. 7 were scaled to the proportion *100x100* pixels, following the relation presented in Table 1.

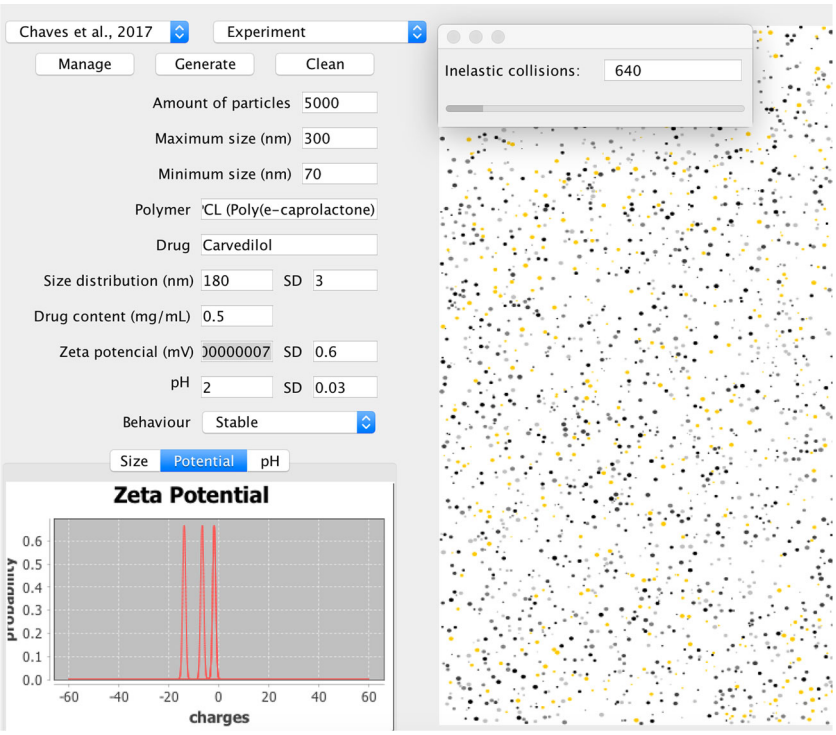


**Fig. 11** Experiment published in [26], with pH value equal to 2

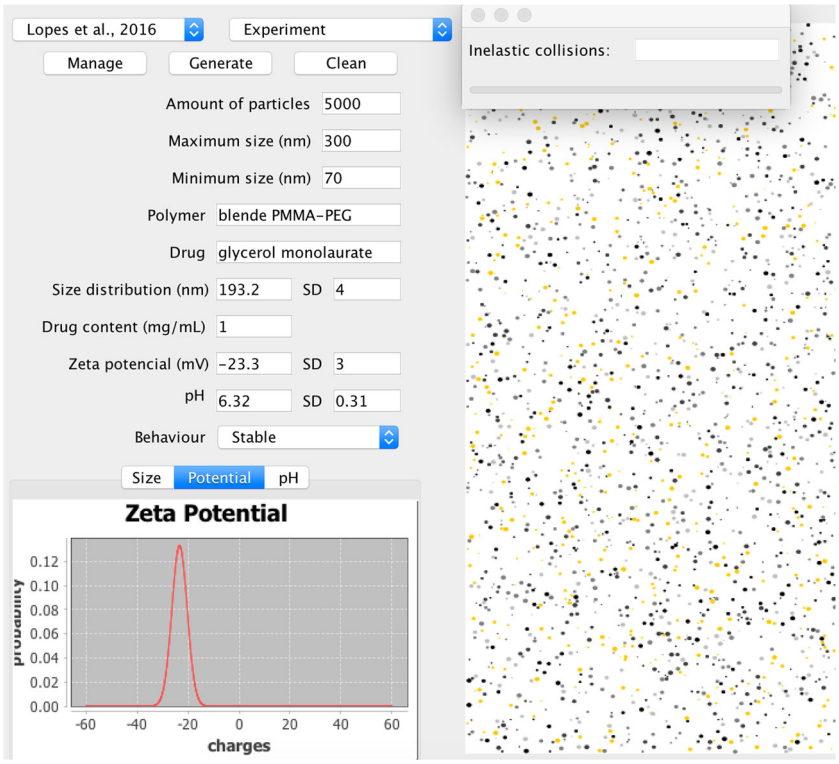


**Fig. 12** Experiment published in [27], with original values

We decided to use the minimum particle size of 50 nm in all comparisons, varying only the maximum size. Also, we decided to use grayscale and yellow for the animation of the particles. This decision was made to facilitate visual analysis.

The greater the difference between the minimum and maximum particle size, for example, 50 nm and 900 nm, the quality of the animation will also be impaired. That is because particles of minimum size will not be drawn correctly. Within this context, we believe that increasing the

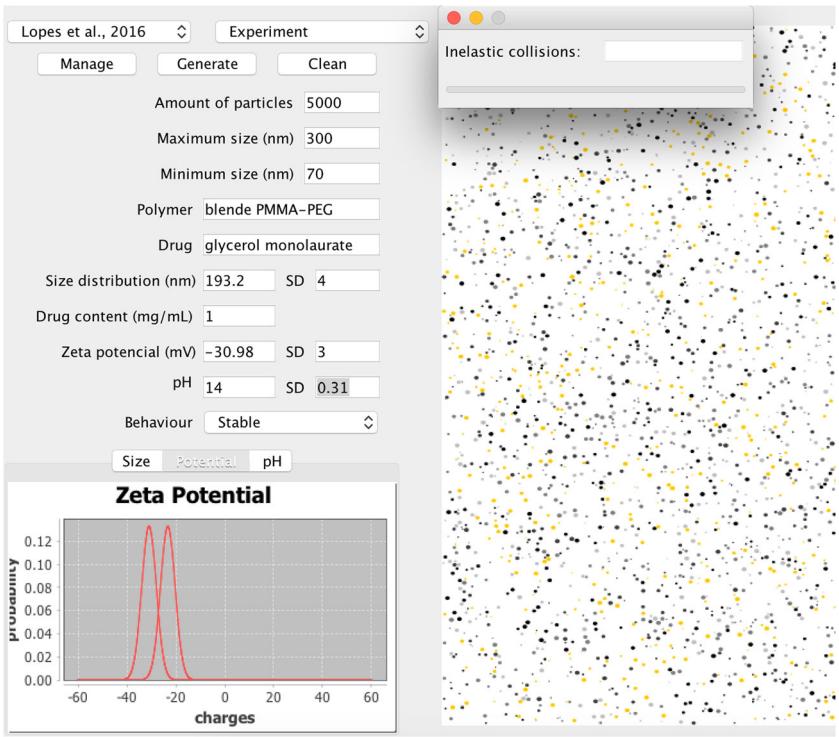


**Fig. 13** Experiment published in [27], with pH value equal to 14

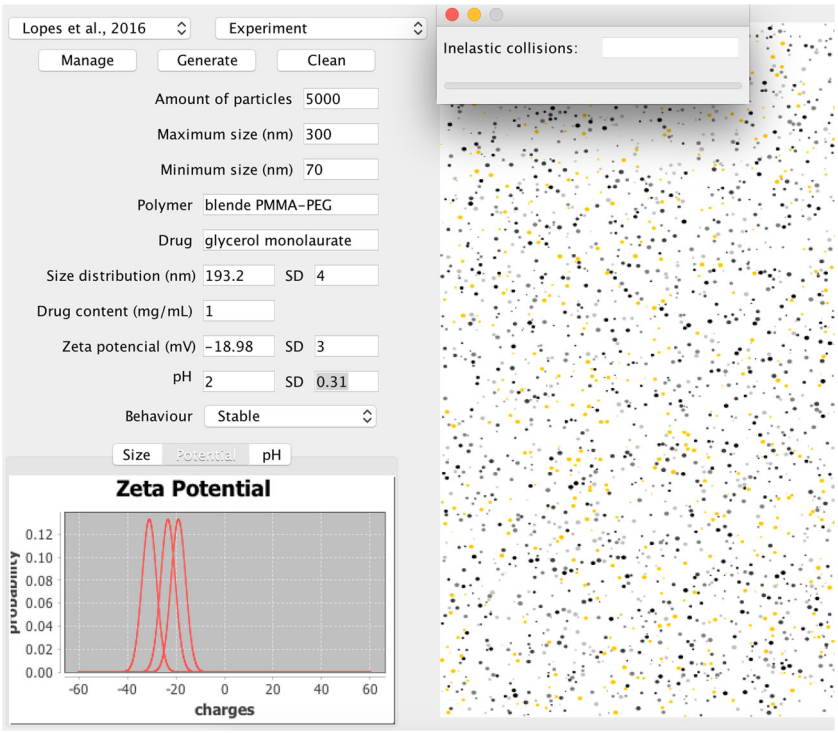


**Fig. 14** Experiment published in [27], with pH value equal to 2

computational resources or using parallel and distributed programming techniques could support simulations with more particles in MASPN.

## Comparison between MASPN and real experiments

To ensure fidelity to the reality of colloidal dispersion, we address the influence of pH on zeta potential, since the most important factor affecting the electric charge is pH. A zeta potential value used without an environment definition (pH, ionic strength, concentration of any additives) is a nonsense number [24]. Dispersion with pH less than 7 is considered acidic, and pH is higher than 7, basic or alkaline. The investigation carried out in [6] shows that if a particle in a dispersion with negative zeta potential had its pH changed (more alkaline), it would be more negative; and it would have neutral potential if acid was added to the dispersion. In general, zeta potential is positive

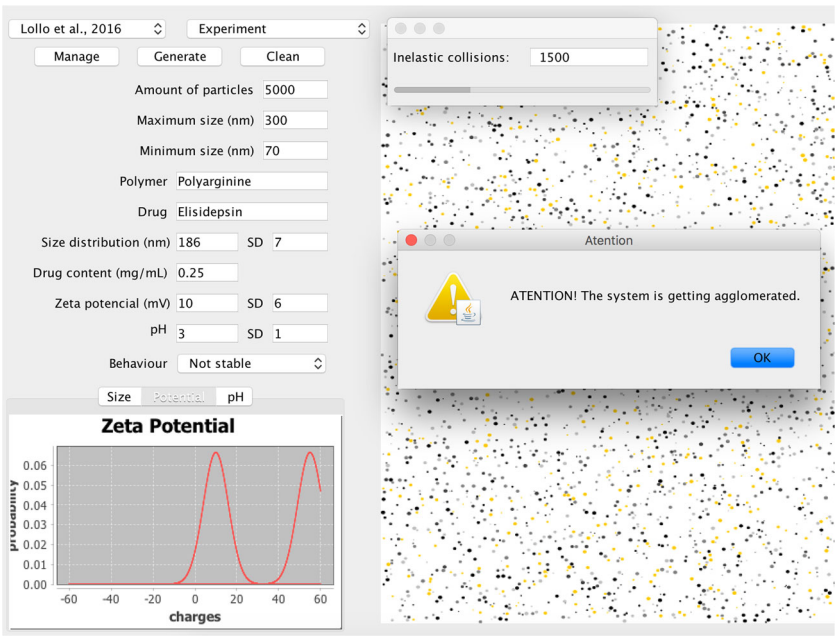


**Fig. 15** Experiment in [28], with manual zeta potential change (from 55 to 10 $mV$)

when pH is less than 6, and negative if pH is higher. In the example of Fig. 8, we can observe that if the pH of the dispersion is less than 5 or greater than 8, there is sufficient charge to produce stability. However, if the pH of the system is between 5 and 8, the dispersion may be unstable.

In this research, we do not consider the steric hindrance property, which has relation to volume and isolation of a particle. Steric hindrance can obstruct the approach even though the particles have different charges [25]. This property is influenced by the choice of polymer type.

In [26], there are values for pH equal to 6.8 and standard deviation of 0.03; zeta potential equal to -6.6 and standard deviation of 0.6 (Fig. 9). By manually changing the pH value to 14.0 while maintaining the standard deviation, the zeta potential value changed to -13.8 with the same standard deviation (Fig. 10).

Subsequently, by manually modifying the pH value to 2 with the same standard deviation, the zeta potential value changed to -1.81. After 5 min of simulation, the amount of inelastic collisions was 640 (Fig. 11).

In [27], the value for pH is equal to 6.32 and standard deviation equal to 0.31; zeta potential -23.3 and standard deviation equal to 3.0 (Fig. 12). By manually changing the pH value to 14.0 while maintaining the standard deviation, the zeta potential value changed to -30.98 with the same standard deviation (Fig. 13).

Subsequently, by manually modifying the pH value to 2 with the same standard deviation, the zeta potential value changed to -18.98. We note that the three figures show that the system is not agglomerated (Fig. 14).

In the research presented in [28], we assume that if the system is not agglomerated and the zeta potential is 55 $mV$, it would be logical that the pH value was between 2 and 5. However, we manually switched the zeta potential value to 10 $mV$, keeping the standard deviation and we realized that the system went into the agglomerate state, as shown in Fig. 15.

## Conclusions

Using agents as particles helps to simulate the particle–environment interaction because ambient pH values must be perceived by the particles. In this case, the pH has a very important relation with colloidal system because it can excite dispersion or not, i.e., speed and surface charge of the particles alter.

This research uses the JASON tool, agent-oriented approach, and provides features to aid in simulations of inelastic collisions. First, because in order to generate or produce inelastic collisions, the colliding particles need to interact, share or exchange information of electric charge (zeta potential), mass (drug content), speed, and direction. Second, the behavior of the particles (velocity, for example) is conditioned to ambient information, such as pH value.

The MAS paradigm is based on natural systems, which shows intelligent behavior from the interaction of its elements, as in an anthill (the colony has an intelligent behavior, while the ant does not) and the neurons (simple cells, but interaction and organization emerges complex and intelligent behavior). At this point, we do not intend to say nanoparticles have intelligent behavior, but rather complex, as they interact with an organization scheme and a very reactive environment.

The MAS area of research is consolidated, with methodologies and tools for the projection and implementation of multi-agent systems, both reactive and cognitive. However, even if the reactive architecture best fits the research, it is possible to design and implement nanoparticle simulation based on the intentional theory of agents based on mental states. This theory is based on the intentional approach, since it has a strong conceptual appeal (as abstraction), being quite natural for designers and analysts who use the agent-oriented approach. In the MASPN environment, we opted for a reactive architecture, but implemented with a tool for cognitive programming.

Another pertinent conclusion is that the Brownian motion of a colloidal system on the macro scale is equivalent on the nanometer scale because with the integration of the physicochemical parameters of the MASPN in the functions of the *algs4* package, it was possible to guarantee fidelity of the elastic collisions system interactions, while MAS contributed to the inelastic collisions system.

This research is relevant because the approach is innovative and interdisciplinary in a way that helps to understand the agglomeration effect on polymer particles by means of a simulated environment. Future work prospects are to introduce high-performance computing techniques to increase the particle handling capacity. Designing and implementing an online Web-based version can also lead to wider use of the system.



## References

1. Dowling A, Clif R, Grobert N (2004) Nanoscience and nanotechnologies: opportunities and uncertainties. Tech. rep., The Royal Academic of Engineering, London
2. Sedgewick R, Wayne K (2011) Algorithms, 4th edn. Addison-Wesley Professional, Boston
3. Ntika M, Kefalas P, Stamatopoulou I (2013) In: 2013 17th International Conference System Theory, Control and Computing (ICSTCC). IEEE, Sinaia, pp 777–782

4. Jo DH, Kim JH, Lee TG, Kim JH (2015) Size, surface charge, and shape determine therapeutic effects of nanoparticles on brain and retinal diseases. Nanomed Nanotechnol Biol Med 11(7):1603
5. Wooldridge M (2001) Introduction to multiagent systems. Wiley, New York
6. Jain K, Mehra NK, Jain NK (2014) Potentials and emerging trends in nanopharmacology. Curr Opin Pharmacol 15:97. https://doi.org/10.1016/j.coph.2014.01.006. http://www.sciencedirect.com/science/article/pii/S1471489214000071. Cardiovascular and renal
7. Bagul R, Mahajan V, Dhake A (2012) New approaches in nanoparticulate drug delivery system: a review. Int J Curr Pharm Res 4(2):29
8. Mohanraj V, Chen Y (2006) Nanoparticles—a review. Trop J Pharm Res 5(1):561. http://www.tjpr.freehosting.net
9. Pohlmann AR, Mezzalira G, de Garcia Venturini C, Cruz L, Bernardi A, Jäger E., Battastini AM, da Silveira NP, Guterres SS (2008) Determining the simultaneous presence of drug nanocrystals in drug-loaded polymeric nanocapsule aqueous suspensions: a relation between light scattering and drug content. Int J Pharm 359(1):288
10. Neto OPV (2014) Intelligent computational nanotechnology: The role of computational intelligence in the development of nanoscience and nanotechnology. J Comput Theor Nanosci 11:1
11. de C. Barbosa R, Krott LB, Barbosa MC (2016) Structural behavior of an anomalous fluid under hydrophobic, hydrophilic and heterogeneous confinement, Journal of Physics - VIII Brazilian Meeting on Simulational Physics 686. https://doi.org/10.1088/1742-6596/686/1/012004. http://www.if.ufrgs.br/barbosa/water-plates-mixture.pdf
12. Halliday D, Walker J, Resnick R (2010) Fundamentals of physics. Wiley, New York
13. Ho-Kim Q, Kumar N, Lam CS (2004) Invitation to contemporary physics. World Scientific, New York
14. Uhrmacher AM, Weyns D (2009) Multi-agent systems: Simulation and application. Computational analysis, synthesis, and design of dynamic models series. CRC Press, Boca Raton
15. Bordini R, Hübner JF, Wooldridge M (2007) Programming multi-agent systems in AgentSpeak using Jason. Wiley, Liverpol
16. Zamberlan A (2002) Em direção a uma técnica para programação orientada a agentes BDI. Master's thesis, PUCRS
17. Zamberlan A, Dalcin AJ, Kurtz G, Bordini R, Raffin R, Fagan S (2016) Simulation environment for polymeric nanoparticle: experiment database. Disciplinarum Sci 17(3):429
18. d'Inverno M, Howells P, Montagna S, Roeder I, Saunders R (2009) Multi-Agent Systems: Simulation and application. In: Uhrmacher AM, Weyns D (eds) Computational Analysis, Synthesis, and Design of Dynamic Models Series. CRC Press, Boca Raton. chap. 13, pp 389–418
19. Hogg T (2007) Coordinating microscopic robots in viscous fluids. Auton Agent Multi-Agent Syst 14(3):271. https://doi.org/10.1007/s10458-006-9004-3
20. Dan N (2014) Nanostructured lipid carriers: Effect of solid phase fraction and distribution on the release of encapsulated materials. Langmuir 30(46):13809
21. Pressman RS (2010) Software Engineering: a Practitioners Approach, 7th edn. McGraw-Hill Education, New York
22. Rao A (1996) Agents Breaking Away. Springer, Eindhoven, pp 42–55
23. Petreska I, Stamatopoulou I (2013) In: Proceedings of the 6th Balkan Conference in Informatics, BCI '13. ACM, New York, pp 53–60. https://doi.org/10.1145/2490257.2490289
24. Rafiee A, Alimohammadian MH, Gazori T, Riazi-rad F, Fatemi SMR, Parizadeh A, Haririan I, Havaskary M (2014) Comparison of chitosan, alginate and chitosan/alginate nanoparticles with respect to their size, stability, toxicity and transfection. Asian Pac J Trop Dis 4(5):372
25. Li Z, Cheng E, Huang W, Zhang T, Yang Z, Liu D, Tang Z (2011) Improving the yield of mono-DNA-functionalized gold nanoparticles through dual steric hindrance. J Amer Chem Soc 133(39):15284. https://doi.org/10.1021/ja205712a. PMID: 21894982
26. dos Santos Chaves P, Ourique AF, Frank LA, Pohlmann AR, Guterres SS, Beck RCR (2017) Carvedilol-loaded nanocapsules: Mucoadhesive properties and permeability across the sublingual mucosa. Eur J Pharm Biopharm 114:88. https://doi.org/10.1016/j.ejpb.2017.01.007. http://www.sciencedirect.com/science/article/pii/S0939641117301054
27. Lopes LQS, Santos CG, de Almeida Vaucher R, Raffin RP, Santos RCV (2016) Nanocapsules with glycerol monolaurate: Effects on *Candida albicans* biofilms. Microb Pathog 97:119. https://doi.org/10.1016/j.micpath.2016.05.016. http://www.sciencedirect.com/science/article/pii/S0882401016301413
28. Lollo G, Gonzalez-Paredes A, Garcia-Fuentes M, Calvo P, Torres D, Alonso MJ (2017) Polyarginine nanocapsules as a potential oral peptide delivery carrier. J Pharm Sci 106(2):611. https://doi.org/10.1016/j.xphs.2016.09.029. http://www.sciencedirect.com/science/article/pii/S0022354916417697