Efficient Simulation of Macroscopic Molecular Communication: The Pogona Simulator

Jan Peter Drees
University of Wuppertal, Germany
jan.drees@uni-wuppertal.de

Lukas Stratmann
TU Berlin, Germany
stratmann@ccs-labs.org

Fabian Bronner
Paderborn University, Germany
fabian.bronner@ccs-labs.org

Max Bartunik
Friedrich-Alexander Universität Erlangen-Nürnberg
max.bartunik@fau.de

Jens Kirchner
Friedrich-Alexander Universität Erlangen-Nürnberg
jens.kirchner@fau.de

Harald Unterweger
University Hospital Erlangen
harald.unterweger@uk-erlangen.de

Falko Dressler
TU Berlin, Germany
dressler@ccs-labs.org

ABSTRACT

Molecular communication in pipe networks is a novel technique for wireless data exchange. Simulating such networks accurately is difficult because of the complexity of fluid dynamics at centimeter scales, which existing molecular communication simulators do not model. The new simulator we present combines computational fluid dynamics simulation and particle movement predictions. It is optimized to be computationally efficient while offering a high degree of adaptability to complex fluid flows in larger pipe networks. We validate it by comparing the simulation with experimental results obtained in a real-world testbed.

CCS CONCEPTS

• **Networks** → **Network simulations**; • **Computing methodologies** → *Model development and analysis.*

KEYWORDS

Molecular communication, particle movement, simulation tools, computational fluid dynamics

ACM Reference Format:
Jan Peter Drees, Lukas Stratmann, Fabian Bronner, Max Bartunik, Jens Kirchner, Harald Unterweger, and Falko Dressler. 2020. Efficient Simulation of Macroscopic Molecular Communication: The Pogona Simulator. In *The Seventh Annual ACM International Conference on Nanoscale Computing and Communication (NANOCOM '20), September 23–25, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3411295.3411297

NANOCOM '20, September 23–25, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8083-6/20/09…$15.00
https://doi.org/10.1145/3411295.3411297

1 INTRODUCTION

Molecular communication describes communication systems that use molecules, particles, or droplets as carriers of information between transmitters and receivers [5, 12]. The physical behavior of these carriers enables new applications which are especially interesting in the medical domain to communicate inside the human body, often referred to as *Body Area Networks* [2], but also in industrial contexts like chemical laboratories or where explosive materials might be susceptible to the radiation of transmitters. We see new testbeds making use of molecular communication evolving frequently with carriers either moving inside liquid-based [15] or gas-based [6] channels. To increase the transmission range, external drivers like pumps and ventilators induce a flow profile inside the channel so that carriers move along with their surrounding fluid. This adds additional complexity in terms of describing the molecular channel analytically and it creates new challenges in defining models for related simulators. When it comes to the simulation of molecular communication, several specific simulators are already available [7, 10, 11, 13]. In a previous study, we noted an oversimplification of flow profiles in these simulators [4] and proposed using a 3-dimensional vector field for particle-based simulations that defines the velocity for each carrier particle at a given position inside this field. As we see a gap between the use of flow-induced channels in real-world testbeds and their support by current molecular communication simulators, we implemented the suggested vector field approach, which basically splits the simulation in two parts: (i) the computation of a flow profile for a user-defined environment, which is exported as a vector field, (ii) the tracking of particles within this vector field, simulating carriers in a molecular channel. To compute the vector field, we suggest using open-source computational fluid dynamics solvers like *OpenFOAM*. This separation allows reusing the computed vector field as long as the geometry and flow attributes of the environment do not change. In several cases (performing multiple runs, changing injector and receptor settings) this saves the time consuming re-computation of the flow profile itself. To validate results taken from this simulator, we recreate the fluid-based scenario originally presented by Bartunik et al. [3] and perform several experiments with the testbed to

obtain data that we can compare our simulation results with. We named our implementation the *Pogona simulator* and will release it as open-source software.

Our main contributions can be summarized as follows:

- Details about our implementation of a particle-based simulator that uses vector fields for particle movement,
- recreation of a real-world testbed within this simulator,
- comparison of simulator to testbed data.

2 RELATED WORK

Several simulators that support molecular communication scenarios are already available. We have a special interest in how they simulate molecular movement. Next to their presentation in literature, we also set up each of them to study their source code and perform simulations.

nanoNS3 describes an analytical model to compute the propagation delay and received signal power of an injection of particles [11]. The used formulas allow to take a uniform flow within the environment into account. To do so, the area averaged flow rate u is computed as in Equation (1). G denotes the hydraulic conductance of the channel which depends on the shape of the environment and viscosity of the fluid; Δp is the pressure drop between transmitter and receiver. The averaged flow rate and length of the channel l allow the computation of the propagation delay τ (cf. Equation (2)).

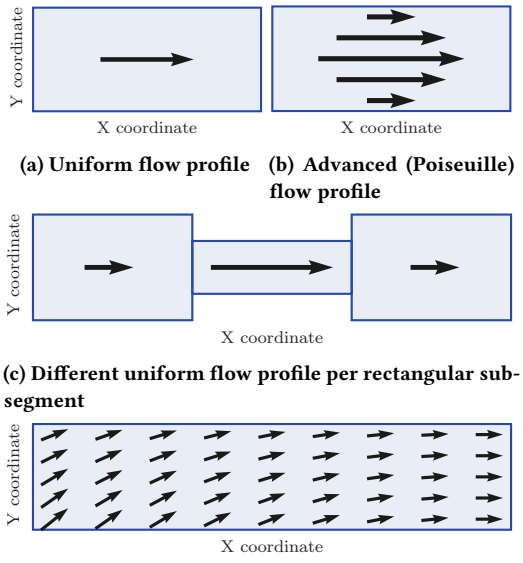
$$u = G \cdot \Delta p \tag{1}$$

$$\tau = \frac{l}{u} \tag{2}$$

nanoNS3 supports only a limited number of geometries such as straight and turning channels with differently shaped cross-sections. Furthermore, an averaged flow rate is used which cannot represent complex flow profiles that might lead to further effects like sedimentation. Figure 1a depicts the flow profile supported by nanoNS3, a location independent, uniform flow.

Particle-based simulators like BiNS2 compute the position of carriers in discrete time steps [7]. In each time step the simulator adds a 3-dimensional displacement to the particle positions. This displacement might be constant to depict a uniform flow profile comparable to Figure 1a. BiNS2 also supports a Poiseuille flow profile by considering the position of each particle inside a molecular channel with circular cross section. A mathematical function computes the amount of displacement by the distance between a particle and the middle of the channel. This enables a more advanced flow profile represented in Figure 1b.

AcCoRD, another particle-based simulator, describes the environment through different geometries glued together [13]. The intention of this approach is to model setups where subsegments are much larger than particles. A uniform flow is defined for each of these subsegments. This allows to define, for example, two chambers connected by a third segment. While it might be technically possible to push this concept to the edge and define very small subsegments to create a very fine-grained flow profile, it is unfeasible in practice. This is because each segment has to be defined manually, which does not only include the spatial attributes but also the direction and amount of flow inside it. The concept used in AcCoRD enables the simulation of a uniform flow per subsegment as shown in Figure 1c but is impractical for portraying complex flow profiles. Other simulators like BloodVoyagerS, which uses a static environment of the human cardiovascular system, follow a similar approach and define a flow for subsegments [10]. In this case segments reflect blood vessels.



(a) Uniform flow profile **(b) Advanced (Poiseuille) flow profile**

(c) Different uniform flow profile per rectangular subsegment

(d) Complex flow profile

Figure 1: Flow profiles with different levels of detail. Longer arrows denote higher flow rates.

Most simulators (nanoNS3, BiNS2, and AcCoRD) support the simulation of molecular movement due to diffusion. While this process seems to be well understood and implemented, the support of complex flow profiles ends at Poiseuille flow in straight channels and uniform flow in larger scaled subsegments. The proposed vector field approach targets the simulation of complex flow profiles like the one given in Figure 1d [4].

3 ARCHITECTURE

The Pogona simulator uses a microscopic approach where the superparamagnetic iron oxide nanoparticles (SPIONs) are represented as discrete information carriers. The underlying assumption is perfect advection: The particles are assumed to be small enough to move exactly the same as the surrounding liquid. Particle movement can then be calculated based on the liquid movement predicted by fluid simulation. The fluid flow itself is pre-computed using a computational fluid dynamics simulator and exported as a vector field. In each simulation time step, the flow at the current position of a particle is determined and applied as the movement of the particle towards its new position. By creating particles at the transmitter and detecting their presence at the receiver, information is propagated through a network of tubes. These core components of the Pogona implementation are covered in the following Section 3.1. Section 3.2 outlines ways in which its performance was improved.

3.1 Implementation

We chose OpenFOAM for the fluid simulation because it is the most widely used open source computational fluid dynamics software. It

can calculate the flow in a tube system by discretization in both time and space and then solving the Navier-Stokes equations numerically. The overall geometry of the tube is subdivided into a mesh of small chunks called "cells". In each time step, a subset of the Navier-Stokes equations are used to determine the flow and pressure in each cell, taking into account the influence of neighboring cells or mesh boundaries. Starting from an initial flow-less state, the OpenFOAM simulation reduces the residual Navier-Stokes error during each time step. When the OpenFOAM simulation has converged to a stable solution after thousands of these steps, the flow satisfies the equations with only minimal remaining error and is ready to be imported into the Pogona simulator.

The movement of each particle inside the tube system uses the geometry and the flow as imported from these computational fluid dynamics results. For simplicity, the center of each cell is considered to be a known data point, with the flow of the mesh cell as the associated function value. Interpolation is necessary to determine the flow of particles that are placed somewhere between these known points. Inverse distance weighting algorithms like Shepard's interpolation are used for this in order to avoid abrupt changes in speed.

At a transmitter, SPIONs are injected into the tube system with a pump. This is modeled as the creation or "spawning" of particles at an initial position. Where and when this spawning takes place is configurable in the simulation. A general model is the continuous or instantaneous creation of particles randomly distributed over a specified area, e.g., a pipe inlet or at a point source. The timing and amplitude of the injection pulses is controlled by a modulation component, which implements on-off keying in our case.

The receiver model is based on real-world measurements of the magnetic susceptometer used in the testbed. A dependency on the lateral position inside of the measurement coil was determined. This is then used as a weight for particles entering the detection area of the sensor. The lateral position of particles in the detection area is determined. Based on the relative influence of particles determined in the testbed at the position, each particle contributes to the overall sensor reading independently. The sensor reading at each time step is then written to a CSV file for further analysis.

3.2 Performance Improvements

When implementing the model as outlined above, the performance will be terrible. For each time step, the movement of each particle needs to be determined. That in turn requires determining the closest cells to get the associated flow to use in the interpolation, which means the entire mesh would need to be iterated over. Considering there will be thousands of time steps, thousands of particles, and a mesh consisting of millions of cells, the problem is obvious. Some of this cannot be avoided, for example because particles move independently of each other.

Determining the flow at an arbitrary lookup position can be optimized easily. Instead of global inverse distance weighting interpolation algorithms like Shepard's [14], the local variant of Franke and Nielson [8] does not need to consider all data points. That means that the data points closest to the lookup point need to be determined first. Instead of saving the data points in a list and comparing them pairwise to the lookup point, a spatial data structure

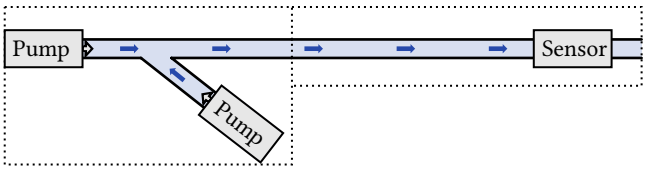


Figure 2: Schematic of the model setup, with the scene decomposition into objects shown as dashed lines.

is used. We chose the k-d tree, which according to Friedman et al. [9] allows determining the m closest neighbors with expected relationship $\log n$ to the overall number of data points n when the points are distributed homogeneously, which is the case in our computational fluid dynamics meshes.

For the time steps, the time resolution is the determining factor for the simulation run time. Unfortunately, the movement prediction is not as accurate with a lower resolution. In addition to the Euler integration method presented above, the Runge-Kutta 4 method which also takes the flow at several intermediate steps into account is implemented. It does not suffer as much from divergence issues and tends to offer an increased accuracy. What needs to be considered is that Runge-Kutta 4 will need to determine the flow at four different positions successively where Euler only needs one lookup. In our experiments, the time resolution could be decreased more than four-fold with Runge-Kutta 4, resulting in a net gain.

Sensor optimization hinges on the observation that a sensor will not need to consider particles outside of a certain detection area. The classic approach would be to check each particle against each sensor at the end of a time step to determine the influence on the sensor response, which scales linearly with the number of sensors in the simulation. However, there is an option for a computation-memory trade-off using pre-computation. At the beginning of the simulation, all mesh cells are checked against the sensor detection areas. Which sensors cover a mesh cell is saved in a list. At the end of the time step, the closest cell for each particle is determined with a k-d tree lookup. Which sensors cover this cell can be determined in constant time with a list lookup and only these will actually be informed of the particle movement. This requires additional memory proportional to the number of mesh cells. The closest data point as determined by the lookup can then be reused for the interpolation in the following time step, producing no computation overhead.

The complexity of the fluid simulation itself also needs to be taken into account. The simulation complexity will depend on the complexity of the flow under study. One issue here is the fact that with pumps turning on and off repeatedly throughout the simulation domain, the number of possible combinations needs to be considered. For an increasing number of pumps, the number of different fluid simulations grows rapidly.

We propose to address this issue with a scene system where the domain is split into largely independent objects. The computational fluid dynamics simulation of each object can be run independently. Consider the model setup shown in Figure 2, for example. At a certain distance downstream from the Y-piece, the flow in the tube is sufficiently close to the ideal parabolic flow profile. This flow profile at the Y-piece outlet is the same as at the inlet of the following

pipe, so both can be separated as shown in the figure. This results in a complexity reduction, since several pump flow combinations in the Y-piece object result in the same inlet flow speed of the tube. Changes in pump flow speeds are propagated downstream recursively. This causes the vector fields in affected objects to be swapped out for a computational fluid dynamics simulation with the updated flows.

Using this scene approach greatly improves the flexibility in the modeling phase. Instead of having to rerun several computational fluid dynamics simulations upon geometry changes, small adjustments like the bend of a tube only require rerunning the computational fluid dynamics simulation of the affected object. Some objects like the straight tube can even support variable lengths and therefore do not require any computational fluid dynamics simulation reruns. Going even further, hybrid models become possible. The flow inside a circular pipe can be determined analytically, so a tube object that does not depend on computational fluid dynamics results and instead reports the analytic solution directly is also supported.

Placing the objects correctly in the scene then becomes a challenge. To aid this setup, a Blender add-on is provided. It supports both the export of scene geometries as well as the import of simulated particle traces to facilitate the inspection of simulation results.

4 EVALUATION

We validated our simulator by comparing the simulated channel impulse response to that of the testbed described in Section 4.1. The scenario demonstrates how the Pogona simulator can be used with a complex injection geometry that would present a challenge for other simulators. The details of the modeling are presented in Section 4.2. Finally, Section 4.3 compares the results of measurement and simulation and discusses the observed differences.

4.1 Testbed

The testbed used to validate the simulator consists of a transmission channel with a constant background flow, a Y-connector as point of injection for information carriers, and the receiver. Various optimization steps and characteristics of the testbed were published previously [1, 3, 15]. The transmission channel is a tube with an inner diameter of 1.52 mm and has a variable length (distance from injection point to receiver) of 5 cm, 10 cm, 20 cm, 30 cm, and 40 cm. Inside the channel, a peristaltic pump (Ismatec ISM831C) provides a constant background flow of 10 mL/min sourced from a reservoir with distilled water.

Superparamagnetic iron oxide nanoparticles (SPIONs) are used as information carriers. As they were originally developed for medical applications, they are principally biocompatible, consisting of an iron core covered with a protective coating. The SPIONs used in this work were synthesized by the Section for Experimental Oncology and Nanomedicine (SEON) of the University Hospital Erlangen. They have a susceptibilty of 7.39×10^{-3} (SI Units) for a concentration of 1 mg Fe/mL and were used at a concentration of 7.5 mg Fe/mL. The average particle diameter is 47 nm with a zeta potential of −33 mV.

The nanoparticles are injected into the transmission channel at a Y-connector that is placed so that the injection occurs against the direction of background flow. This avoids a constant washing out of the SPIONs at the injection point. A second peristaltic pump (Ismatec ISM596D) controls the injection of the SPION-solution, which is sourced from a syringe. Data transmission scenarios are accomplished with concentration shift keying by using an interface to control the injection pump with a PC. The injection flow rate is 10 mL/min and the background flow rate is 5 mL/min.

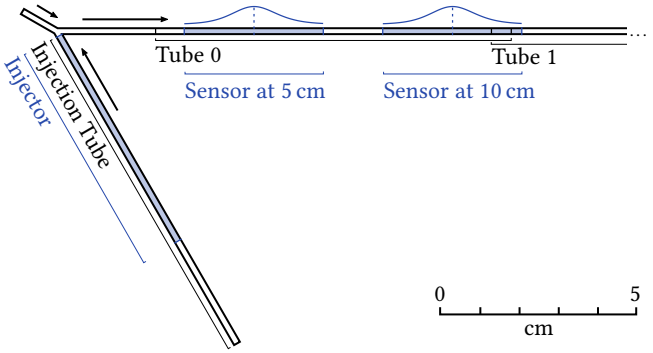


Figure 3: Arrangement of simulation components in the scene. The lateral sensitivity curve is shown in blue.

Due to their magnetic properties, SPIONs can be detected as a shift of coil inductance. To this end, a coil is wound around the transmission channel and fitted with a capacitor to create a resonant circuit. As the SPIONs pass through the coil, the resonance frequency of the circuit shifts dependent on the susceptibility of the nanoparticles. An inductance sensor (LDC1612 by Texas Instruments) is used to power the circuit and digitally acquire the resonance frequency. To reduce systematic environment influences, a second coil is used as differential reference.

For the presented evaluation, we injected 9 times 52.1 µL of particle solution in intervals of 16 s. This was repeated for each of the 5 channel lengths.

4.2 Simulation Method

We validate the simulator by comparing its channel impulse response to that of the testbed. In addition to a qualitative comparison, we particularly investigate the attenuation at increasing channel lengths and we compare the full width at half maximum (FWHM) and root mean square (RMS) delay spreads.

To obtain a valid comparison, the simulation scenario must resemble the testbed scenario as closely as possible. Table 1 lists the key simulation parameters. For this first experiment, we assume constant injection and background flow rates in accordance with the input parameters for the testbed pumps. Regarding the geometry, we adopt all known dimensions from the testbed. As illustrated in Figure 3, the Y-connector is modeled after the physical piece with an angle of 60° between outlet and injection inlet, and 150° between outlet and the inlet for the background flow. All other tubes are modeled without curvature. The mesh resolutions are chosen to give each element at least 20 cells per tube diameter. In the present scenario, this results in a total of about 2 610 000 cells for one Y-connector mesh and 6 instances of a 9 cm tube.

Table 1: Simulation parameters

Parameter	Value
Tube diameter	1.52 mm
Channel length	{5, 10, 20, 30, 40} cm
Injector length	6 cm
Background flow rate	5 mL/min
Injection flow rate	10 mL/min
Injection volume	52.1 µL
Number of particles per injection	5000
Step size	1 ms
Simulation time	36 s
Simulation runs	16

Figure 3 furthermore shows the setup of the simulation scene. We connect multiple meshes for the reasons explained in Section 3. The Y-connector is critical as this is where the flow from the injection tube merges into the background flow. When an injection is initiated, the injector component is filled with randomly positioned particles in a single time step to model the particle-filled tube of the testbed. The flow within the injection tube is switched on for the duration of the injection, which also affects the flow rates in connected objects downstream. After the injection, any remaining particles inside the injector are deleted to avoid unnecessary computations and such that these particles will not interfere with the distribution of particles of a potential subsequent injection. This approach requires the injection tube to be long enough to not run out of fast-moving particles while the injection is still in progress. Therefore, another straight tube is attached to the injection inlet of the Y-connector. The outlet of the Y-connector is extended by multiple straight tubes to accommodate all sensors for distances of up to 40 cm from the injection point. Any particles moving beyond the last sensor will be deleted once they enter a destructor component positioned behind the sensor.

4.3 Results and Discussion

Figure 4 shows the mean channel impulse responses we observed at the respective channel lengths listed in Table 1. We scaled all testbed and all simulated susceptibilities to fit inside the [0, 1] interval, with the maximum observed value of the testbed and simulator mapping to 1, respectively. As Figure 4 shows, our simulator matches the attenuation of the physical channel within the tested range of channel lengths quite well. We observe a maximum difference between mean scaled peak heights of 0.099, which occurs in the 30 cm condition; 95 % CI [0.31, 0.32] (simulator) and [0.17, 0.25] (testbed).

It can be surmised from Figure 4 that, in simulation, channel impulse responses tend to be narrower than in the testbed. This is also reflected by the observed FWHMs visualized in Figure 5, which show a consistently lower average value in simulation. Initially, the values in the simulator and testbed are relatively close; at 5 cm, the difference of mean FWHMs is 0.29 s; 95 % CI [0.95 s, 0.99 s] (simulator) and [1.18 s, 1.35 s] (testbed). At 20 cm, however, the values begin to diverge noticeably with a difference of 1.37 s; 95 % CI [1.56 s, 1.69 s] (simulator) and [2.81 s, 3.19 s] (testbed). The largest FWHM difference is reached at 40 cm with 3.28 s; 95 % CI [2.70 s, 3.04 s] (simulator) and [5.72 s, 6.58 s] (testbed). Consistent with the differences we observed in FWHM measurements, we also see lower RMS delay spreads in simulation. Here, the difference of means at 5 cm is 0.69 s; 95 % CI [0.45 s, 0.47 s] (simulator) and [1.08 s, 1.23 s] (testbed). The maximum difference is reached at 20 cm with 1.87 s; 95 % CI [1.62 s, 1.70 s] (simulator) and [3.43 s, 3.63 s] (testbed). After this, the offset between mean RMS delay spreads remains somewhat constant for the remaining observed channel lengths.

At present, it is difficult to trace these detected discrepancies back to any particular parameter or not-simulated physical phenomenon. For example, we assume diffusion and sedimentation to become stronger factors with increasing distances. Modeling the behavior

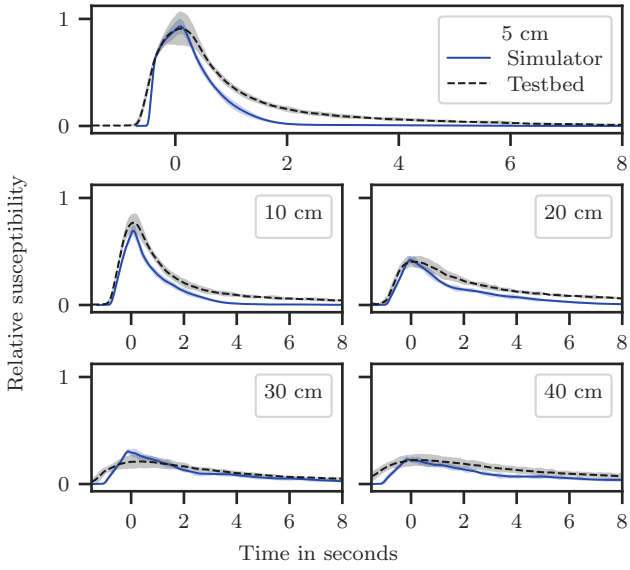


Figure 4: Comparison of the average channel impulse responses between testbed and simulator for varying distances between injection point and sensor, given an injection volume of 52.1 µL. Shaded areas indicate 95 % confidence intervals. All plots for the simulator and testbed are scaled relative to the respective overall maximum observed value.

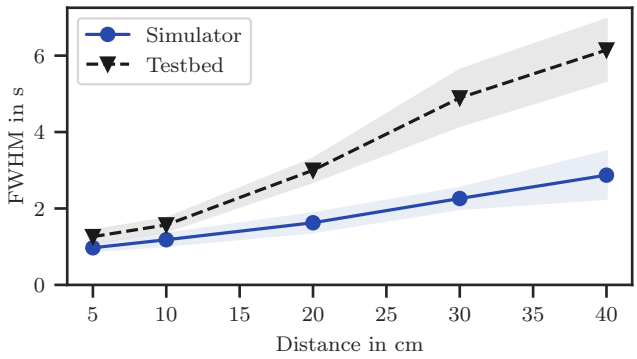


Figure 5: Average full width at half maximum (FWHM) in the testbed and in simulation for the tested channel lengths. Shaded areas indicate 95 % confidence intervals.

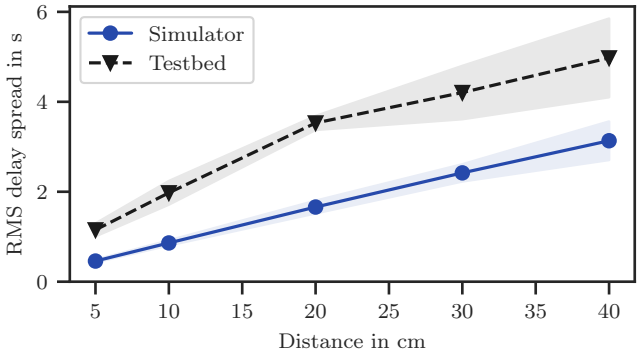


Figure 6: Average root mean square (RMS) delay spreads in the testbed and in simulation for the tested channel lengths. Shaded areas indicate 95 % confidence intervals.

of the peristaltic pumps may yield further interesting results. This is motivated by the observation that their flow rates are not perfectly constant and instead vary in pulses. Furthermore, the manufacturer specifies a reduction in flow rate at the end of an injection, which could affect the shape of the channel impulse response.

At this time, the Python implementation of our Pogona simulator takes 6 hours and 45 minutes to finish a single-threaded simulation run with the parameters listed in Section 4.2. We ran the simulations on an AMD Ryzen™ 7 3800X CPU.

5 CONCLUSION

On macroscopic scales, the motion of nanoparticles is influenced strongly by the geometry in which they move. In this study, we present first results of a comparison between a wet-lab Y-connector injection with the same scenario replicated in a novel particle-based molecular communication simulator using precomputed vector fields. We showed that our Pogona simulator is able to replicate the approximate shape of the channel impulse responses in the physical testbed. In future work, it should be possible to reduce the remaining discrepancies between testbed and simulation results further by considering additional physical effects. Furthermore, a more detailed look at different channel and injection parameters, such as tube diameters and flow rates, may help to paint a clearer picture of the potential fidelity of this simulation method. This is especially important in light of additional preliminary results that suggest turbulence at the injection point to be an important factor for some flow rates. Our short-term intention is also to improve our simulator's practical run time by porting much of the performance-relevant Python code to lower-level languages. Another possible performance improvement can be made by extending the implementation of the Runge-Kutta method for adaptive time stepping, since particles in a straight tube segment should not require step sizes quite as small as particles inside the Y-connector. Eventually, having an efficient simulator with support for complex geometries and scenes opens up the path to investigating larger-scale molecular communication networks, potentially with multiple injectors and mixing or separating flows. The ability to simulate such networks is particularly useful for analyzing higher-layer protocols. These include, for example, protocols for multiple access control or forwarding and routing messages to distant receivers.

ACKNOWLEDGMENTS

Reported research was supported in part by the project *MAMOKO* funded by the German Federal Ministry of Education and Research (BMBF) under grant numbers 16KIS0917 and 16KIS0913K.

REFERENCES

[1] Doaa Ahmed, Harald Unterweger, Georg Fischer, Robert Schober, and Jens Kirchner. 2019. Characterization of an Inductance-based Detector in Molecular Communication Testbed Based on Superparamagnetic Iron Oxide Nanoparticles. In *IEEE SENSORS 2019*. Montreal, Canada. https://doi.org/10.1109/SENSORS43011.2019.8956713

[2] Baris Atakan, Ozgur B. Akan, and Sasitharan Balasubramaniam. 2012. Body Area NanoNetworks with Molecular Communications in Nanomedicine. *IEEE Communications Magazine (COMMAG)* 50, 1 (Jan. 2012), 28–34. https://doi.org/10.1109/mcom.2012.6122529

[3] Max Bartunik, Maximilian Lübke, Harald Unterweger, Christoph Alexiou, Sebastian Meyer, Doaa Ahmed, Georg Fischer, Wayan Wicke, Vahid Jamali, Robert Schober, and Jens Kirchner. 2019. Novel Receiver for Superparamagnetic Iron Oxide Nanoparticles in a Molecular Communication Setting. In *6th ACM International Conference on Nanoscale Computing and Communication (NANOCOM 2019)*. ACM, Dublin, Ireland. https://doi.org/10.1145/3345312.3345483

[4] Fabian Bronner and Falko Dressler. 2019. Towards Mastering Complex Particle Movement and Tracking in Molecular Communication Simulation. In *6th ACM International Conference on Nanoscale Computing and Communication (NANOCOM 2019), Poster Session*. ACM, Dublin, Ireland, 36:1–36:2. https://doi.org/10.1145/3345312.3345490

[5] Stephen F. Bush, Janet L. Paluh, Guiseppe Piro, Vijay Rao, Venkatesha Prasad, and Andrew Eckford. 2015. Defining Communication at the Bottom. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications* 1, 1 (March 2015), 90–96. https://doi.org/10.1109/TMBMC.2015.2465513

[6] Nariman Farsad, Weisi Guo, and Andrew W Eckford. 2013. Tabletop Molecular Communication: Text Messages through Chemical Signals. *PLOS ONE* 8, 12 (Dec. 2013), 1–13. https://doi.org/10.1371/journal.pone.0082935

[7] Luca Felicetti, Mauro Femminella, Gianluca Reali, Paolo Gresele, and Marco Malvestiti. 2013. Simulating an in vitro experiment on nanoscale communications by using BiNS2. *Elsevier Nano Communication Networks* 4, 4 (Dec. 2013), 172–180. https://doi.org/10.1016/j.nancom.2013.08.003

[8] Richard Franke and Greg Nielson. 1980. Smooth interpolation of large sets of scattered data. *Internat. J. Numer. Methods Engrg.* 15, 11 (1980), 1691–1704. https://doi.org/10.1002/nme.1620151110

[9] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. 1976. An algorithm for finding best matches in logarithmic time. *ACM Trans. Math. Software* 3, 3 (1976), 209–226. https://doi.org/10.1145/355744.355745

[10] Regine Geyer, Marc Stelzner, Florian Büther, and Sebastian Ebers. 2018. BloodVoyagerS: Simulation of the Work Environment of Medical Nanobots. In *5th ACM International Conference on Nanoscale Computing and Communication (NANOCOM 2018)*. ACM, Reykjavík, Iceland, 5:1–5:6. https://doi.org/10.1145/3233188.3233196

[11] Yubing Jian, Bhuvana Krishnaswamy, Caitlin M. Austin, A. Ozan Bicen, Jorge E. Perdomo, Sagar C. Patel, Ian F. Akyildiz, Craig R. Forest, and Raghupathy Sivakumar. 2016. nanoNS3: Simulating Bacterial Molecular Communication Based Nanonetworks in Network Simulator 3. In *3rd ACM International Conference on Nanoscale Computing and Communication (NANOCOM 2016)*. ACM, New York City, NY, 17:1–17:7. https://doi.org/10.1145/2967446.2967464

[12] Tadashi Nakano, Michael J. Moore, Fang Wei, Athanasios V. Vasilakos, and Jianwei Shuai. 2012. Molecular Communication and Networking: Opportunities and Challenges. *IEEE Transactions on NanoBioscience* 11, 2 (June 2012), 135–148. https://doi.org/10.1109/TNB.2012.2191570

[13] Adam Noel, Karen C. Cheung, Robert Schober, Dimitrios Makrakis, and Abdelhakim Hafid. 2017. Simulating with AcCoRD: Actor-based Communication via Reaction–Diffusion. *Elsevier Nano Communication Networks* 11 (March 2017), 44–75. https://doi.org/10.1016/j.nancom.2017.02.002

[14] Donald Shepard. 1968. A two-dimensional interpolation function for irregularly-spaced data. In *23rd ACM National Conference (ACM '68)*. ACM, Las Vegas, NV, 517–524. https://doi.org/10.1145/800186.810616

[15] Harald Unterweger, Jens Kirchner, Wayan Wicke, Arman Ahmadzadeh, Doaa Ahmed, Vahid Jamali, Christoph Alexiou, Georg Fischer, and Robert Schober. 2018. Experimental Molecular Communication Testbed Based on Magnetic Nanoparticles in Duct Flow. In *19th IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC 2018)*. Kalamata, Greece, 1–5. https://doi.org/10.1109/SPAWC.2018.8446011