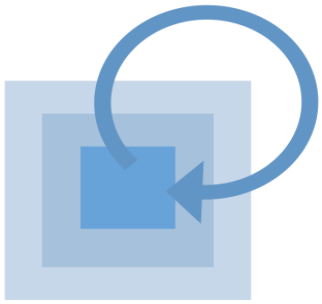

2011

Deliverable Report

D6.2 Feedback systems for container-molecular-electronic IT (M24)

Matrix for Chemical IT (MATCHIT)

Grant Agreement no. 249032

Deliverable 6.2: Feedback system for container-molecular-electronic IT

This deliverable focuses on objectives 6.1 (Extend DNA and Membrane Computing to a language for MATCHIT) and 6.4 (Extend the existing real-time feedback control interface to couple containers with electronics). It describes the simulation package of the MATCHIT-CTRL-software controlling real experiments.
Numerous new features (see manual in the supplementary material) were added to allow for optimization and to ease the control of real experiments.

Physical and chemical realism in real-time simulation

The ionic-chemistry library calculates (depending on the time-dependent electrode voltages), the time and spatial dependence of the concentrations of ionic species and the nonlinearly coupled local electrostatic potential, via an extended Poisson-Nernst-Planck framework, which takes finite ion size effects into account (this is joint work with the EU-project ECCell). From this, the forces to which the tracer particles in the simulation are subjected can be directly derived (proportional to the gradient of the potential). For the electrochemical problem solution, an attached Mathematica File summarizes the theoretical solution of the coupled equations for all mobile chemical species concentrations and the two potentials: electrostatic and chemical. The treatment is fully nonlinear, and treats finite size effects via an entropic excluded volume correction that is important because otherwise ions can pile up to unphysical concentrations at oppositely charged electrodes. The procedure is made efficient by mapping the 3D geometry to a 1D framework, extending theory developed recently for the PNP equations in membrane ion channels. Novel here is the transformation of coordinates derived for a general area function that allows a non-singular description of the general pseudo-1 D problem generalizing the case of hemispherical electrodes to electrodes embedded in a channel. We use the solution of Laplace's equation to derive the static 3D shape of potential curves and then use the full time dependent solution to assign potential values to different positions. This captures the main physical effects of the exceedingly complex interplay between electronic and migration and reaction behavior in aqueous solutions. The Mathematica solution structure must then be exported to C and coupled with sparse linear matrix equation solver libraries for integration with the MATCHIT-CTRL software. This is aided by Mathematica's inbuilt C export capability, but requires some additional matching with external libraries.

Simulating chemtainers as polygons with particles as content

The real-time simulation facility has been extended with an electrostatics based particle swarm chemtainer package. All geometries are taken from the real design-data used to produce the actual devices. This package is fully integrated into the MATCHIT-CTRL-software and usable in parallel with ongoing real experiments. The simulation itself is mapped via a camera-view into the user-interface and it thus represents an additional device in the normal experimental procedure. Particles are diffusing around, using a Brownian motion metaphor and these particles are assumed to be without mass. In addition to missing inertia also friction effects are neglected. It is assumed that the contributions to the overall potential at a certain spatial locations are simply sums of the individual electrodes. From this resulting potential a force vector is calculated. The particles are assumed to have an inner electric charge though this charge is not used to calculate particle-particle interactions. This is certainly a violation of physical properties but it is assumed that the concentration of the particle is low enough to neglect these interactions. Incorporated into the analytical solution of the electrode potentials are screening effects of the ionic double-layer at the surfaces of the electrodes. These particles can be enclosed by a flexible wall (which internally is realised as a polygon) which serves as a membrane with partial transport of materials through.

Fluid channels are represented by 2D-polygons which represent walls that are supposed to be of infinite height (in z-direction). These walls cannot be surpassed by the particles. To reduce the computational demand of particles hitting the walls a very simple solution is chosen: if a particle is about to cross such a wall the according move is not realised. This results in a higher probability of the particles to remain in the vicinity of walls which therefore exhibit some adhesive property. The initial focus of this development is less on a physically correct simulation platform but to investigate the power of feedback-control-loops to regulate the phenomena observed in experiments. The deeper reason for the feedback-controller is precisely to counteract perturbations in the microfluidic system (and this also can counteract some perturbations due to inexact simulations). Classical physically accurate particle based simulation methodologies are too slow to deal with real-time controller tasks and major simplifications must be made. However, we are working on embedding this work in a hierarchy of successively more accurate simulation methodologies, see previous section. In particular, the real-time demands also exclude partial differential equation systems because taking real microfluidic geometries into account would require a triangulation of the available physical space with a subsequent solution of the PDE system outside of currently available computing powers..

The following strong simplifications were made initially to still allow the development of a real-time simulation package: 1) Particle based, the goal is to have up to 1000 particles in real-time. 2) No solvent, in conventional particle based simulation systems the solvent calculations take the vast majority of the available compute power. 3) No mass, all particles are either randomly displaced or deterministically moving along given trajectories, low Reynolds numbers and laminar flow. 4) Attraction, repulsion, hydrophobic and electrostatic interactions are modeled via probabilities of random displacement acting as virtual forces. 5) Electrostatic interactions are super-imposable. Recent advances in calculating the field-strengths point to using concentration fields and take particles only as visualization means.

The simulation is controlled via a specific window in the MATCHIT-CTRL graphical user-interface, which is shown in Figure 1. Several “physical” properties can be specified, like the particle density, the spatial location where particles are placed when pressing the ‘Seed’ button, their hydrophylicity (currently not used), as well as particles diffusion-rate and mobility and a virtual size (this virtual size only serves graphical purposes to easily distinguish them and to calculate interactive repelling forces). How particles attract each other or how strong the react to the electrical fields can also be specified as the temperature and a cutoff-radius beyond which particles to not sense each other. The simulation as such is controlled with the buttons at the bottom of the control-cockpit.

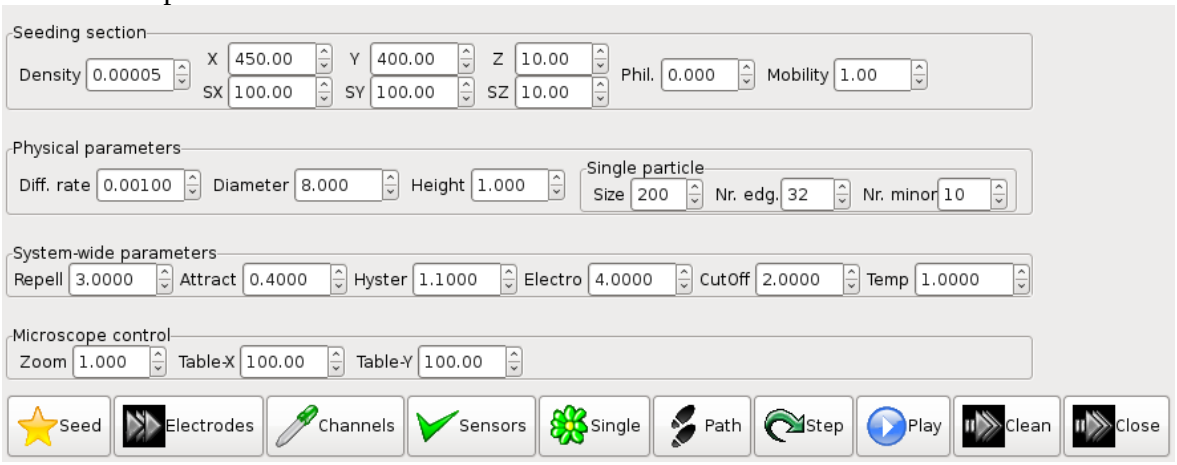

Figure 1: Simulation-cockpit to specify the parameters used for the particle simulation. Furthermore the simulation is controlled via the buttons at the bottom of the screen dump.

In addition to the camera-view the simulation with the actual particles shows the status of the electrodes in the design window, an example is shown in Figure 2.

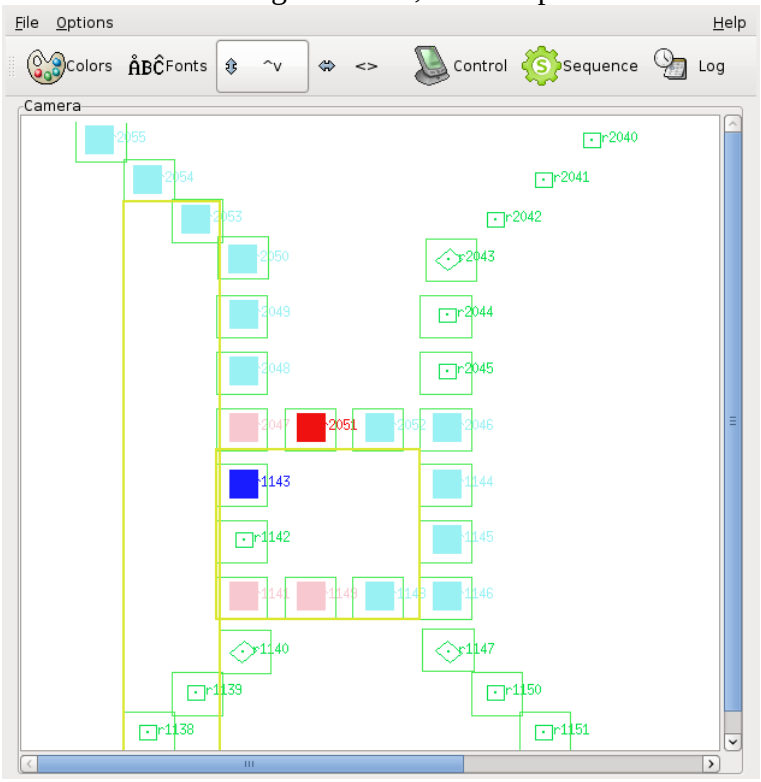

Figure 2: For illustration only: how electrode activation patterns look-like when simulating the movement of particle clouds under the influence of electric fields. Strong colors resemble active electrodes and faint color deactivated electrodes. The green large rectangles depict the sensors and the smaller rectangles the surface of the electrodes. In addition the electrode-identifiers are visible in this screen-shot.

An intermediate level programming language

An intermediate concurrent programming language was designed and implemented. This is work done in conjunction with the EU-project ECCell. This intermediate language level is powerful enough to run optimizations and to create a mobile algorithmic sphere which moves along with the real chemistry inside the channels. The programming language is identical for the simulation part in MATCHIT-CTRL and the experiments' control part. Because it has to deal with real world properties, it is connected to real designs in microfluidic chips and it is developed such that hardware specifications are hidden as much as possible. Two major ideas were prevalent in developing this intermediate language level: All position information in this language must boil down to electrodes. These electrodes are pooled in so called paths. Paths are abstractions of real micofluidic channels. Usually each electrode is accompanied by a sensor which has the shape of an electrode but with typically eight times the area. The second idea was to intersperse the programming language with the control-language. All commands inside the control-language are available to the programming-language as well. This means the programming-language is a superset of low-level commands: an interesting variant of current day programming paradigms in IT. With the ability to create online arbitrarily many new commands, simply by naming a shell-script, the intermediate programming-language can be extended at will for any given problem.
A third idea that guided the development of the intermediate programming-language was to restrict the number of language constructs as much as possible to ease the programming task for the experimenter. Though it supports all typical commands known from LISP (List Processing, http://en.wikipedia.org/wiki/LISP) it is not thought to be used for arbitrary problem-solving, like

programming a text-processing system. The language is processed by an embedded interpreter and as such cannot promise the same performance as a standalone pure programming language. Let alone, sustaining the control of the experiments while processing the language requires considerable resources, e.g. for image processing. This usually is not a problem because the observed chemistry is slow enough to not render this performance gap a problem.
To learn from other programming-language projects we looked at ERLANG (http://en.wikipedia.org/wiki/Erlang_(programming_language)) and LUA (http://en.wikipedia.org/wiki/Lua_(programming_language)). It turned out that the elegance of ERLANG could not be utilized here because of the strong hardware embedding in the Electronic Chemical Cell. LUA has the problem of not really supporting the special parallelization mode we do need here. The major approach to implement the intermediate programming language was done along the lines of LUA though. It turned out that the list-processing facility was easily implemented and only complex parts, like path-finding in the fluidic designs, needed to be completely written a new. In the following only the MATCHIT specifics are explained in more detail. See the accompanying user-manual for a complete description of all elements of the language.

The instance construct

In principle, instance can be many things, a cloud of particles, a chemtainer or a single chemical compound. An instance is bound to a certain droplet if the droplet is passing an electrode the instance was bound to via the 'control'-property.

instance:	Initialize a new instance. Formally instance is a name of a container. This instance can move along a series of electrodes or it is part of another instance. instance {name}
addprop:	Add a property to an instance or chemical. There are certain properties: addprop control {script-file} {central electrode} addprop content {list of chemicals or other instances (instances only)} addprop type ['controller', 'membrane'] addprop permeability {value} addprop concentration {value} addprop area {value of polygon area (instances only)} addprop address {name or identifier of expressed address}
rule:	Add a chemical production rule to an instance. rule {rule-name} {list of educt pairs (name, nr)} [to {rate}, back {rate}, equ {rate}] {list of product pairs (name, nr)}
drop:	Delete an instance or chemical because it vanished in the experiment or was pushed outside region of interest. drop {instance-name}
replicate:	Replicate a specific instance because of observation of a chemtainer splitting event. A new name for the copy is created automatically. This name also comprises the history of the parent copies. Replicate {instance-name}
chemical:	A chemical is part of a chemical reaction. Its properties are concentration, permeability and address.

This language allows connecting the abstract computations from the MATCHIT-calculus and the MATCHIT-automaton with the MATCHIT-CTRL-sofware.

Evolution and optimization is supported via the conditional execution of parts of the program and the commands maxinten, mininten, maxtime and mintime which act as sensors to the real experiments. It is thus conceivable to implement genetic algorithms on several experiments running in parallel in the microfluidics and being evaluated with a specified fitness criterion.