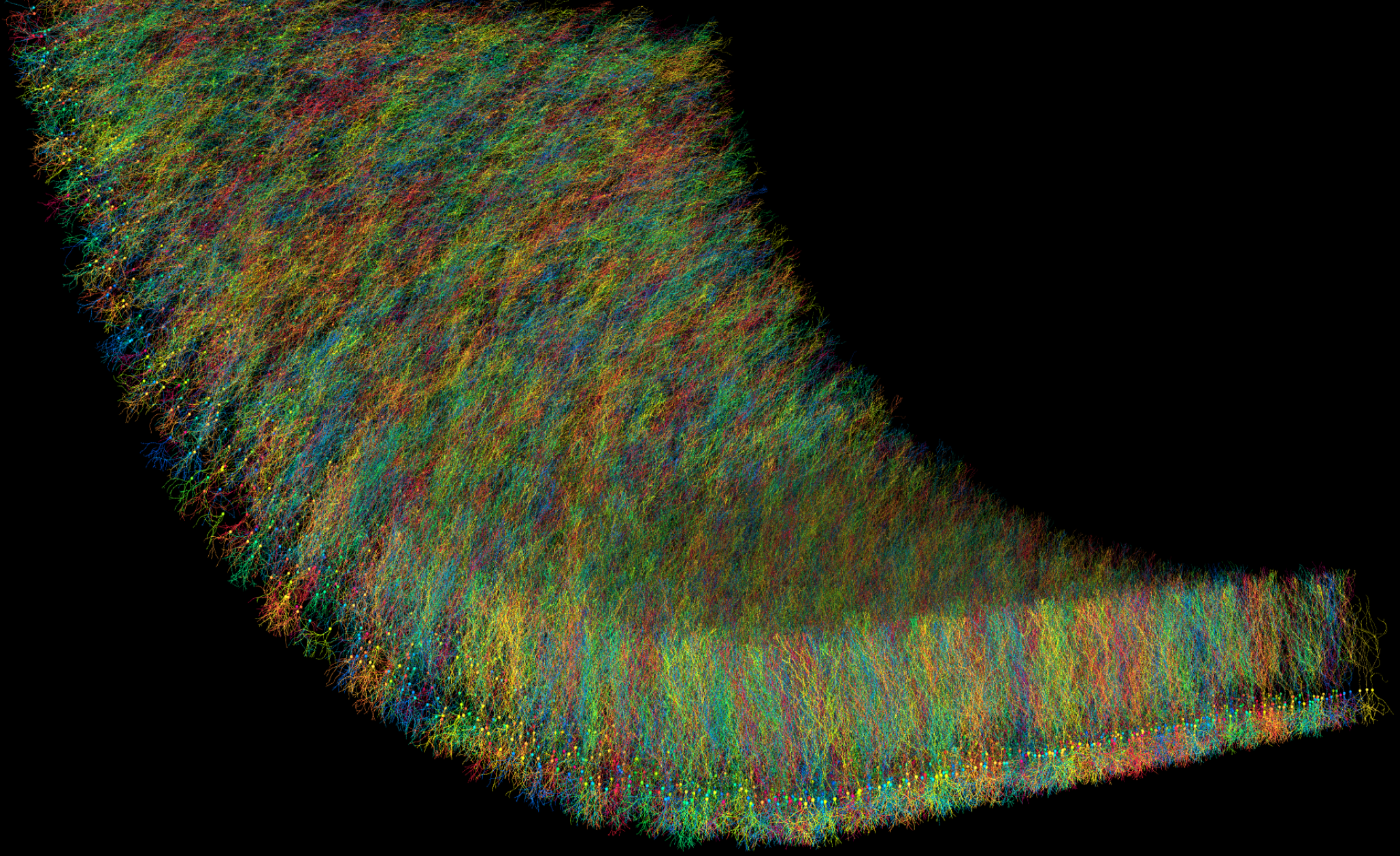


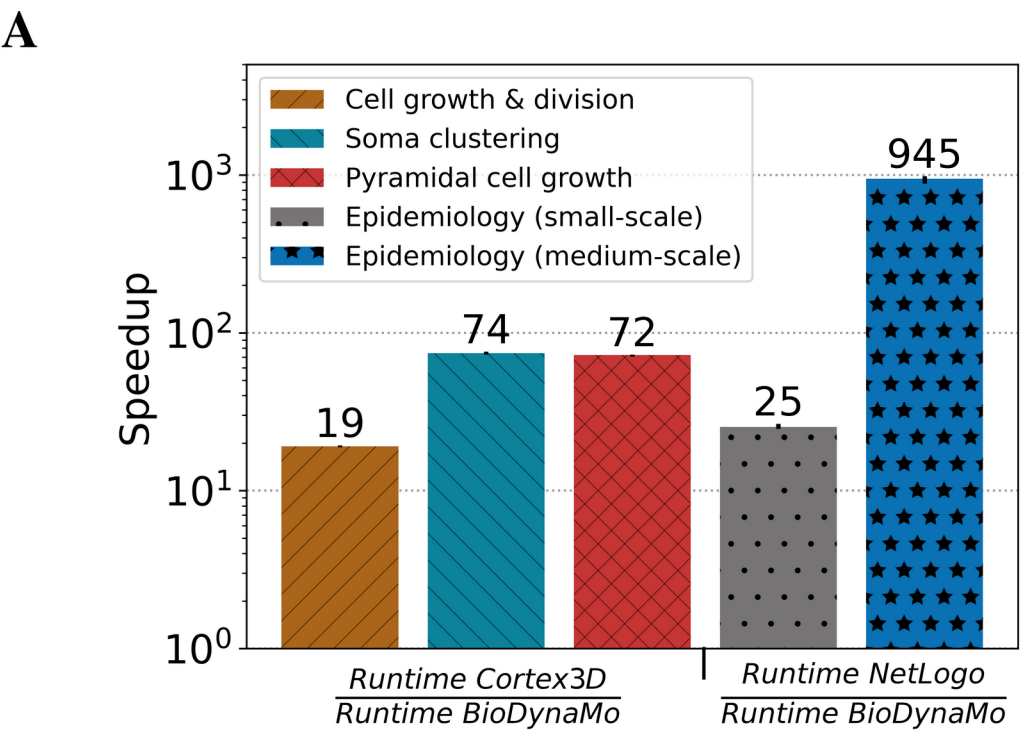
The BioDynaMo Platform: Current State and Future Developments

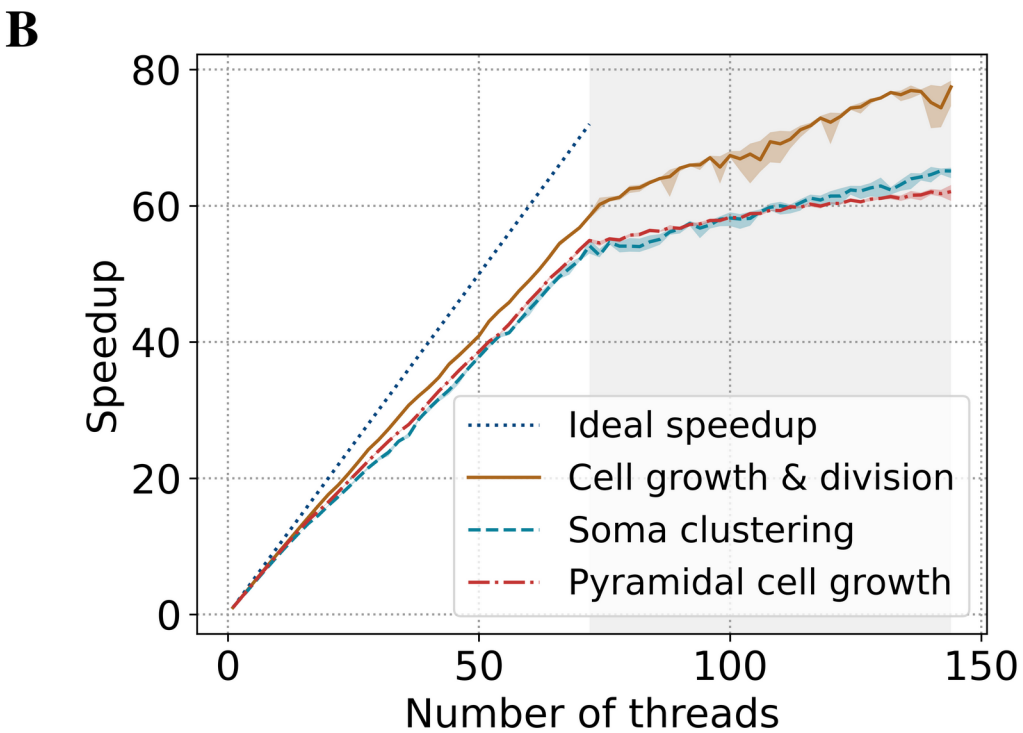
Lukas Breitwieser

BioDynaMo Workshop on Agent-Based Modeling 19.11.2021

BioDynaMo is a **high-performance and modular, agent-based** simulation platform written in C++.

BioDynaMo Performance 1/2





Source: Breitwieser et al. 2021, https://doi.org/10.1093/bioinformatics/btab649

BioDynaMo Performance 2/2

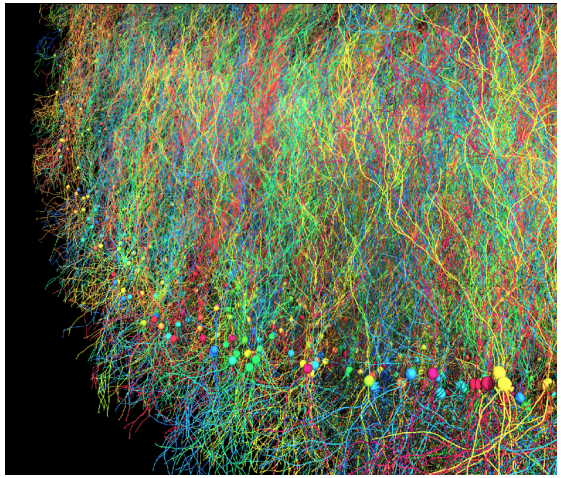
Table 1. **Performance data.** The values in column "Agents" and "Diffusion volumes" are taken from the end of the simulation. Runtime measures the wall-clock time to simulate the number of iterations. It excludes the time for simulation setup and visualization. The entries in column "System" correspond to Supplementary File S1 Table 5. Supplementary File S1 Table 6 contains more detailed performance data.

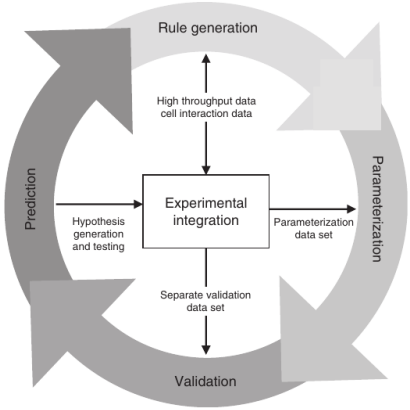
Simulation	Agents	Diffusion volumes	Iterations	System	Physical CPUs	Runtime	Memory
Neuroscience use case							
Single (Figure 4A)	1 494	250	500	A	1	0.16 s	382 MB
Large-scale (Figure 4C)	9 054 740	65 536	500	A	72	36 s	6.02 GB
Very-large-scale	1 018 644 154	5 606 442	500	B	72	1 h 26 min	436 GB
Oncology use case (Figure 5)							
2000 initial cells	4 177	0	312	A	1	1.05 s	382 MB
Large-scale	1 000 3925	0	288	A	72	1 min 42 s	7.42 GB
Very-large-scale	986 054 868	0	288	B	72	6 h 21 min	604 GB
Epidemiology use case (Figure 6C)							
Measles	2 010	0	1000	A	1	0.53 s	381 MB
Seasonal Influenza	20 200	0	2500	A	1	16.41 s	383 MB
Large-scale (measles)	10 050 000	0	1000	A	72	59.19 s	5.87 GB
Very-large-scale (measles)	1 005 000 000	0	1000	B	72	2 h 0 min	495 GB

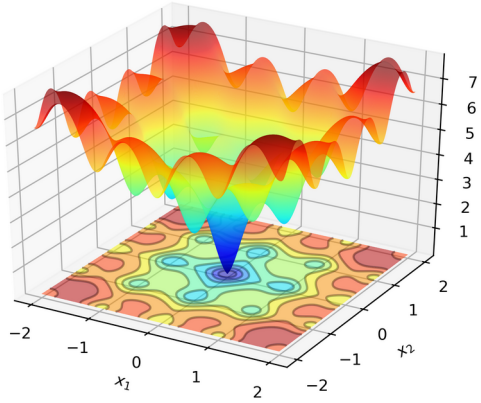
Source: Breitwieser et al. 2021, https://doi.org/10.1093/bioinformatics/btab649

Impact of Speed and Efficiency

- Simulate larger and more complex models
- Reduce development time
- Explore bigger parameter space
- Reduce cost

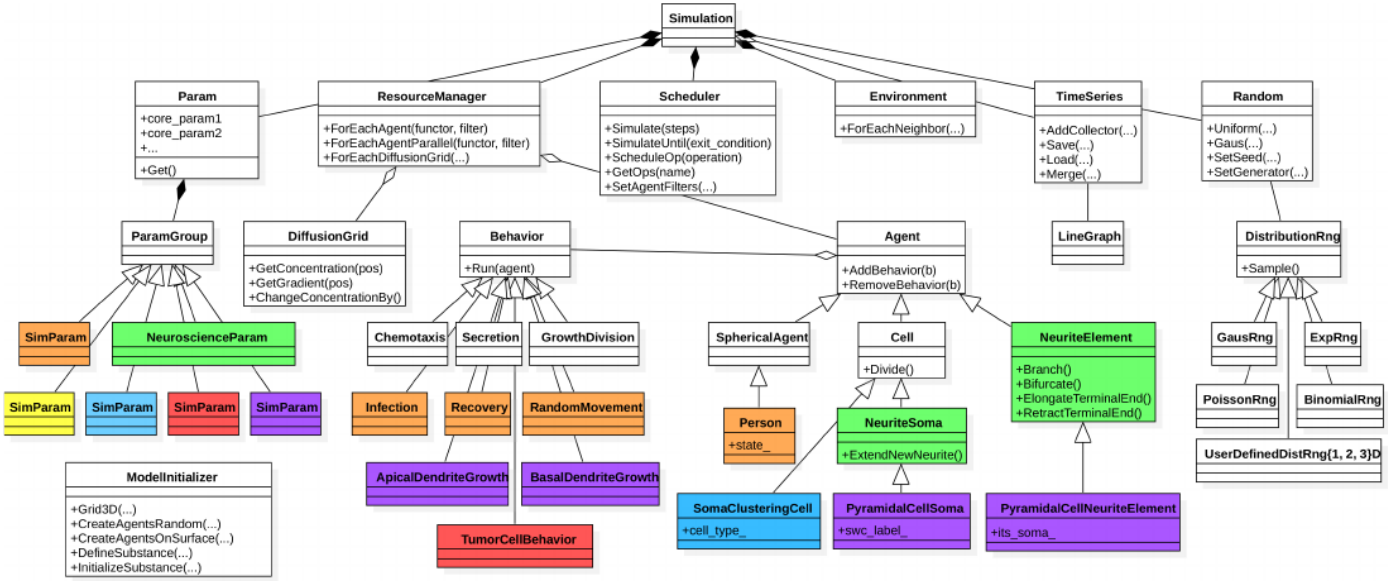




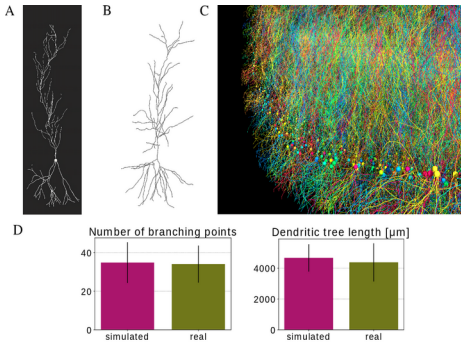


Images credit: https://github.com/kthohr/optim, https://doi.org/10.1093/bib/bbm024

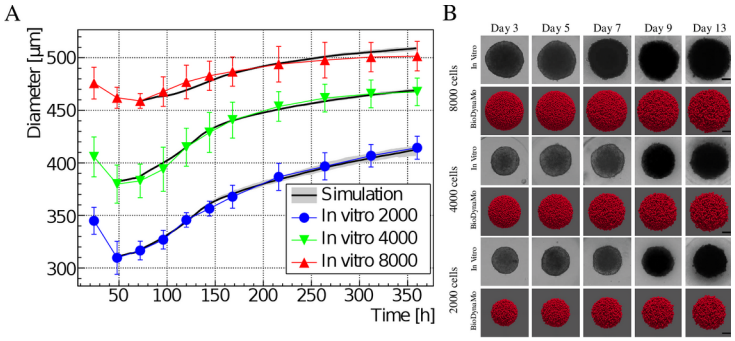
Modular Software Design



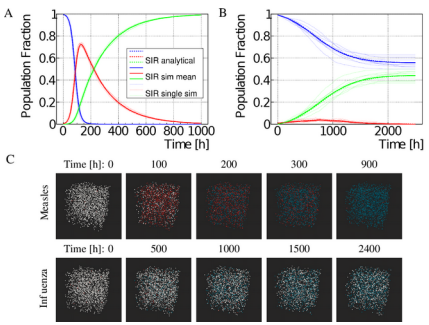
Neuroscience use case



Oncology use case



Epidemiology use case



Source: Breitwieser et al. 2021, https://doi.org/10.1093/bioinformatics/btab649

Features and abstraction layers

Simulation	*Simulation*
• **Agent geometry:** sphere, cylinder • **Agents**: Cell, NeuronSoma, NeuriteElement • **Behaviors**: Secretion, Chemotaxis, Proliferation, GeneRegulation • Extraculluar diffusion • Agent interaction force	*BioDynaMo's model building blocks*
• Generation of agent populations • Agent reproduction & mortality • Environment search • Multi-scale simulations • Dynamic scheduling • Statistical analysis • Parameter management • Parameter optimization • Hierarchical model support • Hybrid-modeling • Space boundary conditions	*BioDynaMo's high-level features*
• Parallelism & thread-safety • Performance optimizations • GPU support • Visualization • Web-based interface • Backup & restore of simulations • Quality assurance infrastructure	*BioDynaMo's low-level features*
OpenMP, ROOT, ParaView, Others	*Libraries*
Linux / MacOS	*Operating System*
(Multi-core) CPUs, GPU	*Hardware*

Source: Breitwieser et al. 2021, https://doi.org/10.1093/bioinformatics/btab649

Demo

Main BioDynaMo Article

Bioinformatics, 2021, 1–8
doi: 10.1093/bioinformatics/btab649
Advance Access Publication Date: 16 September 2021
Original Paper

OXFORD

Systems biology

BioDynaMo: a modular platform for high-performance agent-based simulation

Lukas Breitwieser [1,2,*], Ahmad Hesam[1,3,*], Jean de Montigny[1], Vasileios Vavourakis[4,5], Alexandros Iosif[4], Jack Jennings[6], Marcus Kaiser[6,7,8], Marco Manca [9], Alberto Di Meglio[1], Zaid Al-Ars[3], Fons Rademakers[1], Onur Mutlu[2,10,*] and Roman Bauer[11,*]

Bioinformatics Journal (Oxford)

Impact Factor
6.937
Mathematical & Computational Biology
3 out of 58

Editors-in-Chief
Janet Kelso
Alfonso Valencia

About the journal

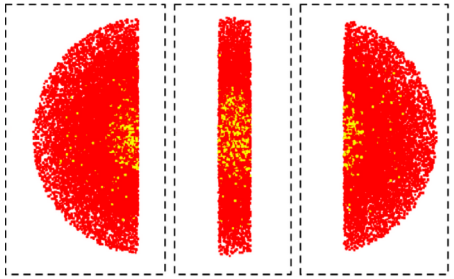
The leading journal in its field, *Bioinformatics* publishes the highest quality scientific papers and review articles of interest to academic and industrial researchers. Its main focus is on new developments in genome bioinformatics and computational biology…

Find out more

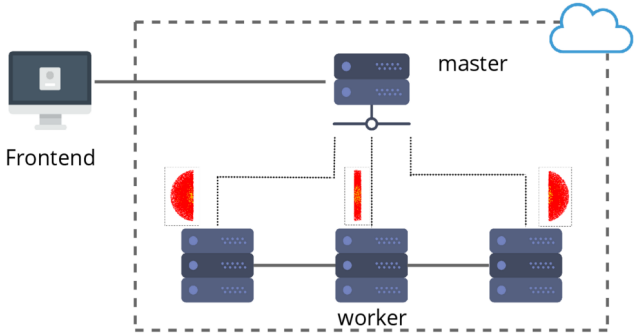
https://doi.org/10.1093/bioinformatics/btab649

Roadmap

- Release v1.02
- Improvements for continuum-based approach (see Tobias talk).
- Distributed runtime



Domain decomposition



Distributed system architecture

Summary

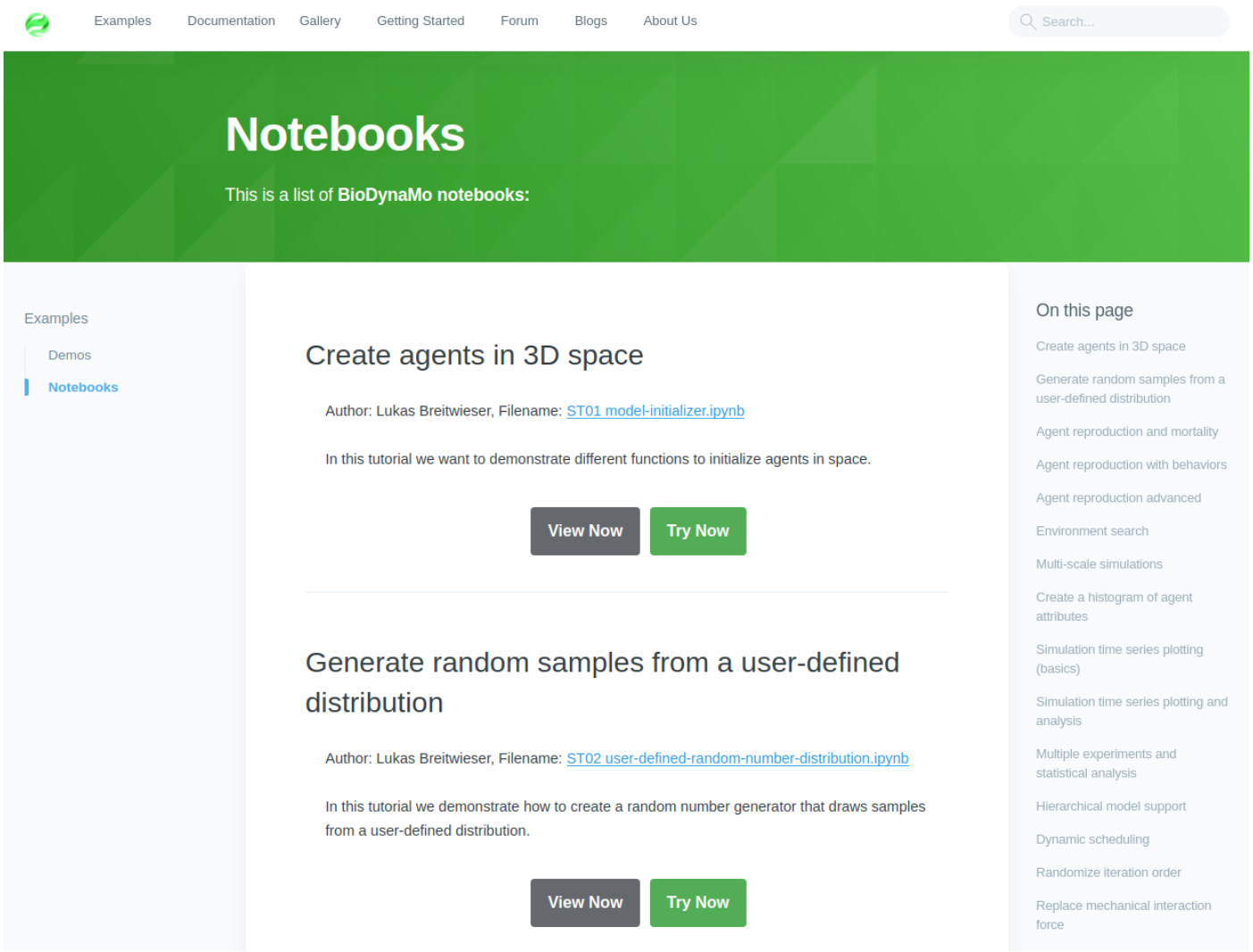
- Due to our tireless efforts, BioDynaMo is capable of simulating **billions of agents** in a **wide variety of research fields**.
- We added **numerous new features**
  - Parameter space exploration
  - Statistical analysis
  - Multiple execution modes (e.g. randomization)
  - Hierarchical model support
  - …
- Improved **ease-of-use and documentation**
  - 15 new BioDynaMo notebooks
  - 11 demos
  - Can be explored from within the web-browser without installing BioDynaMo

Questions?

Lukas.Breitwieser@cern.ch

Supplementary Material

15 BioDynaMo Notebooks



https://biodynamo.org/examples/notebooks

Create agents in 3D space

Author: Lukas Breitwieser

In this tutorial we want to demonstrate different functions to initialize agents in space.

Let's start by setting up BioDynaMo notebooks.

In [1]:
```
%jsroot on
gROOT->LoadMacro("${BDMSYS}/etc/rootlogon.C");
```

```
INFO: Created simulation object 'simulation' with UniqueName='simulation'.
```

We use `SphericalAgent`s with $diameter = 10$ for all consecutive examples.

In [2]:
```
auto create_agent = [](const Double3& position) {
  auto* agent = new SphericalAgent(position);
  agent->SetDiameter(10);
  return agent;
};
```

We define the number of agents that should be created for functions that require this parameter.

In [3]:
```
uint64_t num_agents = 300;
```

We define two helper functions that reset the simulation to the empty state and one to visualize the result.

In [4]:
```
void Clear() {
  simulation.GetResourceManager()->ClearAgents();
}
```

In [5]:
```
void Vis() {
  simulation.GetScheduler()->FinalizeInitialization();
  VisualizeInNotebook();
}
```

Create agents randomly inside a 3D cube

Cube: $x_{min} = y_{min} = z_{min} = -200$ and $x_{max} = y_{max} = z_{max} = 200$

By default a uniform random number distribution is used.

In [6]:
```
Clear();
ModelInitializer::CreateAgentsRandom(-200, 200, num_agents, create_agent);
Vis();
```

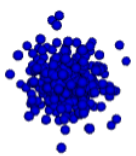
Create agents randomly inside a 3D cube using a gaussian distribution

Cube: $x_{min} = y_{min} = z_{min} = -200$ and $x_{max} = y_{max} = z_{max} = 200$

Gaussian: $\mu = 0, \sigma = 20$

Note the extra parameter rng passed to `CreateAgentsRandom`

In [7]:
```
Clear();
auto rng = simulation.GetRandom()->GetGausRng(0, 20);
ModelInitializer::CreateAgentsRandom(-200, 200, num_agents, create_agent, &rng);
Vis();
```



https://biodynamo.org/notebooks/ST01-model-initializer.html

Create a histogram of agent attributes

Author: Lukas Breitwieser

In this tutorial we will show how to create a histogram of all agent diameters in the simulation and fit a function to the data.

Let's start by setting up BioDynaMo notebooks.

In [1]:
```
%jsroot on
gROOT->LoadMacro("${BDMSYS}/etc/rootlogon.C");
```

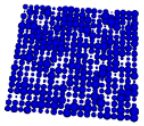
```
INFO: Created simulation object 'simulation' with UniqueName='simulation'.
```

We want to define a function that creates a cell at a certain position with diameters drawn from a gaussian distribution with $\mu = 20$ and $\sigma = 5$. The smallest diameter should be larger then 2.0.

In [2]:
```
simulation.GetResourceManager()->ClearAgents();
auto rng = simulation.GetRandom()->GetGausRng(20, 5);
auto create_cell = [&](const Double3& position) {
  Cell* cell = new Cell(position);
  double diameter = std::max(2.0, rng.Sample());
  cell->SetDiameter(diameter);
  return cell;
};
```

Now that we defined `create_cell` we can use it to create 400 cells on a plane with $z = 0$, $xmin = ymin = -200$, $xmax = ymax = 200$, and spacing = 20 in both dimensions.

In [3]:
```
auto f = [](const double* x, const double* params) { return 0.0; };
ModelInitializer::CreateAgentsOnSurface(f, {}, -200, 200, 20, -200, 200, 20,
                                        create_cell);
simulation.GetScheduler()->FinalizeInitialization();
VisualizeInNotebook(300, 300);
```



The next step is to create a histogram object with 100 bins in the interval [2, 40].

The second line creates a function which fills the histogram with the diameter of the given agent.

The third line calls the function `fill` for each agent, thus adding all diameters to the histogram.

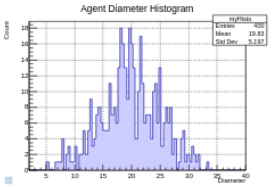
In [4]:
```
TH1F h("myHisto","Agent Diameter Histogram;Diameter;Count", 100, 2, 40);
auto fill = L2F([&](Agent* a, AgentHandle){ h.Fill(a->GetDiameter()); });
simulation.GetResourceManager()->ForEachAgent(fill);
```

Let's draw the final histogram.

Before we have to create a `TCanvas` object in order to display the result in this notebook.

We also modify the default color and create a grid.

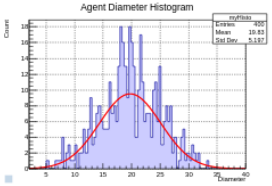
In [5]:
```
TCanvas c("", "", 400, 300);
h.SetFillColor(kBlue - 10);
c.SetGrid();
h.Draw();
c.Draw();
```



Finally, we can try to fit a function to the data in the histogram.

Since we drew samples from a gaussian random number generator when we created our cells, we expect that a gaussian will fit our data.

In [6]:
```
h.Fit("gaus", "S");
h.Draw();
c.Draw();
```




```
FCN=73.1872 FROM MIGRAD    STATUS=CONVERGED      78 CALLS          79 TOTAL
                     EDM=6.76682e-08    STRATEGY= 1      ERROR MATRIX ACCURATE
 EXT PARAMETER                                   STEP         FIRST
 NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
  1  Constant     9.50544e+00   7.09907e-01   2.21048e-03   4.02840e-04
  2  Mean         1.97314e+01   3.23457e-01   1.36014e-03  -3.65230e-04
  3  Sigma        5.40166e+00   3.17423e-01   6.23484e-05   2.34242e-02
```

https://biodynamo.org/notebooks/ST08-histograms.html

BioDynaMo Demos

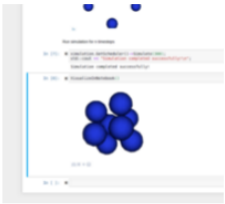
Binding Cells

Try Now

Cell Division

Try Now

Diffusion



Try Now

Multiple Simulations

Try Now

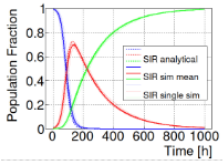
Parameters

Try Now

Pyramidal Cell

Try Now

Epidemiology



Try Now

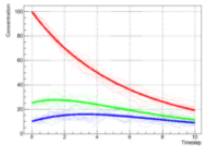
Gene Regulation

Try Now

Makefile Project

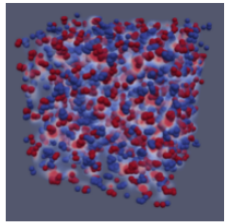
Try Now

Sbml Integration



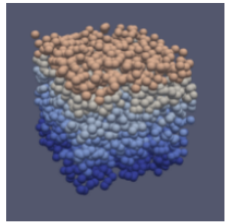
Try Now

Soma Clustering



Try Now

Tumor Concept



Try Now

ETH zürich CERN CERN openlab

https://biodynamo.org/examples/demos

Release v1.02

New Features

- Added parallelized optimization manager #205
- Added googletest support for simulations #187
- Improved biodynamo cli #215
- Added experimental version of the automated benchmarking suite #202
- Added CopyExecutionContext
- Added support for different execution orders
- Introduced execution context interface #189
- Reduced memory consumption of the BDM memory manager #186
- Added SphericalAgent
- Added toroidal space boundary condition
- Added RandomizedRm to randomize the iteration over all agents
- Added support for hierarchical agent-based models
- Added analysis classes to simplify data collection and plotting #177
- Improved random number generation
- Added `Scheduler::SimulateUntil(exit_condition)`
- Added class LambdaFunctor and function L2F to simplify functor creation #175
- Added octree and kd-tree as alternative environments #169

Bug Fixes

- Added simulation dependent diffusion time step #198
- Fixed bug in diffusion grid initialization #199
- Fixed MathArray::Norm and Normalize for zero vector #194
- Do not call `Rm::EndOfIteration` in `ExecCtxt::SetupIterationAll`
- Fixed errors in the static agent detection mechanism #191
- Fixed race condition in `DiffusionGrid::ChangeConcentrationBy`
- Changed BioDynaMo version from `vXX.YY-ZZ-gSHA` to `vXX.YY.ZZ-SHA` #216