Improving the continuum modelling within BioDynaMo

Tobias Duswald, M.Sc.

19 November, 2021

Continuum modelling with BioDynaMo

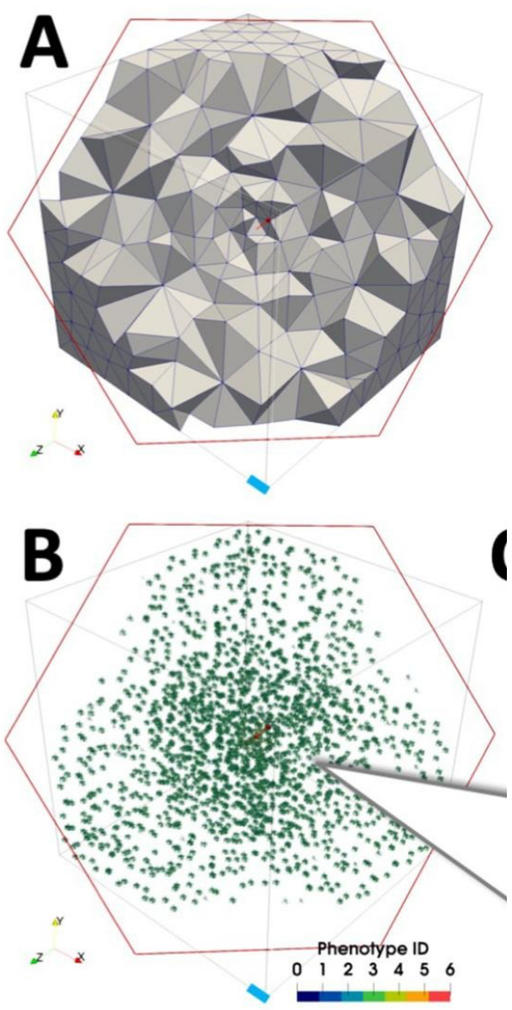
Slides overview

1. Motivation
2. Current State
3. MFEM Integration
4. A Conceptual Example: Diffusion

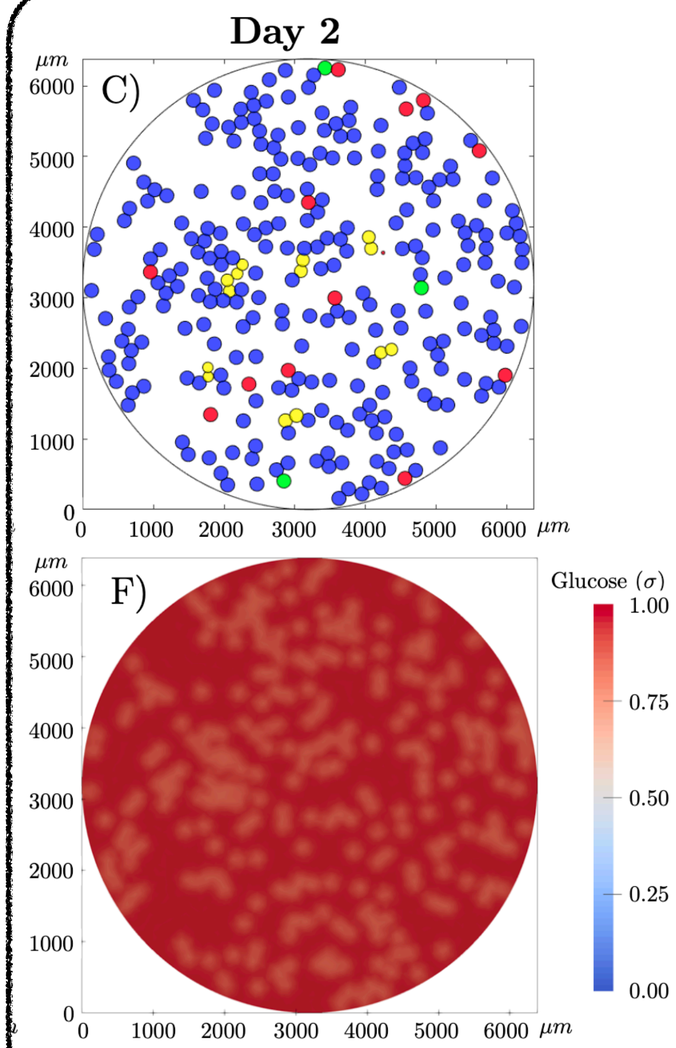
Motivation

Hybrid Modelling

ABM + continuum model



de Montigny, Methods, 2021
10.1016/j.ymeth.2020.01.006



Lima, bioRxiv, 2021
10.1101/2021.03.03.433731

Current State

Current State

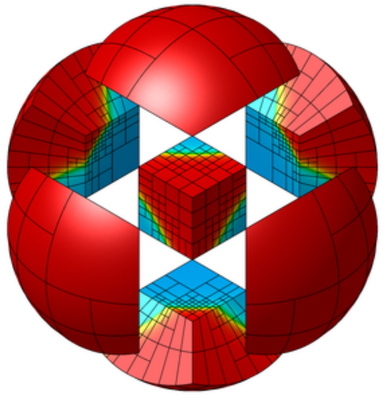
	Currently	Target
Equation	Heat / Diffusion equation	Arbitrary
Time discretization	Forward Difference	Explicit & implicit ODE solvers
Space discretization	Central difference	Finite elements
Geometry	Cube	Arbitrary but convex
ABM - Continuum interactions	Read of values, modify values	Read of values, density function

Conclusion

It’s unfeasible to support a variety of use cases with the current approach.
We need an interface to a sophisticated library handling
different geometries, solvers, and equations.

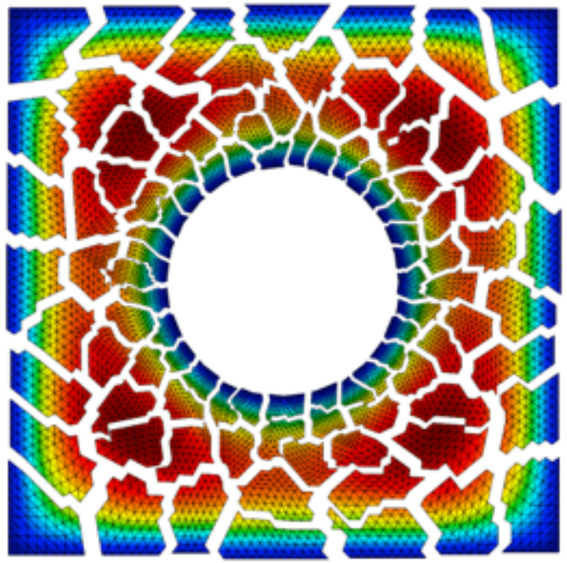
MFEM integration

What is MFEM? Why MFEM?

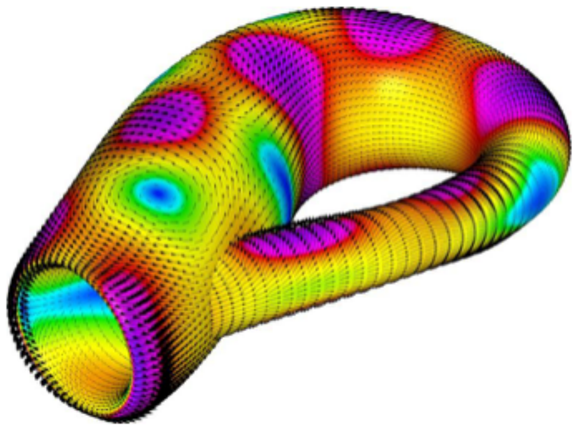


- “A finite element to linear algebra translator"
- Modern, object-oriented software design, C++
- Many options for parallelism
- Cmake compatible
- Active community & responsive developers
- Backed by Lawrence Livermore NL
- Licences compatible (BSD 3-Clause)
- 300K (cloc)

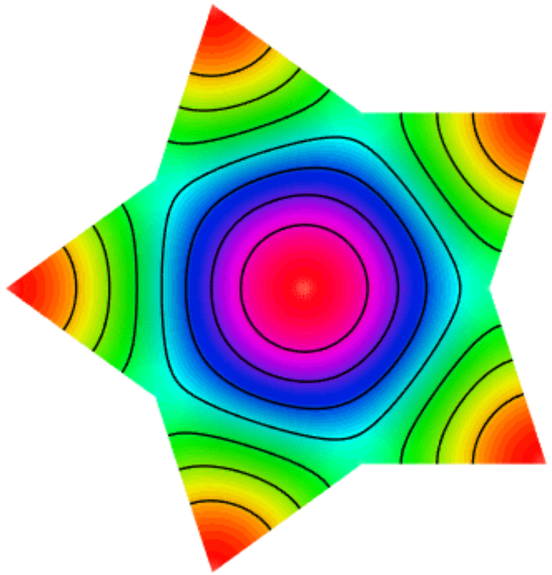
MFEM examples



$$-\Delta\Phi = 1$$



$$\nabla \times \nabla \times E + E = f$$



$$\frac{\partial \Phi}{\partial t} = \nabla(\kappa + \alpha\Phi)\nabla\Phi$$

<u>Target:</u> support time dependent diffusion processes based on the Method of Lines.

A Conceptual Example: Diffusion

Equation: $$\partial_t \Phi - \nabla D \nabla \Phi = f$$

FE expansion: $$\Phi(x,t) = \sum_{j=1}^{n} u_j(t) \Psi_j(x)$$

Derive ODE system: **Insert the expansion and apply** $\int_\Omega d\Omega \; \Psi_i(x)$

Result: $$M \frac{\partial \vec{u}}{\partial t} = D \vec{u} + \vec{f}$$

$$M \in \mathbb{R}^{n \times n},\; M_{ij} = \int_\Omega \Psi_i \Psi_j \, d\Omega \qquad D \in \mathbb{R}^{n \times n},\; D_{ij} = \int_\Omega \Psi_i \left(\nabla D \nabla \Psi_j \right) d\Omega \qquad \vec{f} \in \mathbb{R}^n,\; f_i = \int_\Omega \Psi_i f \, d\Omega$$

Good source:
https://tinyurl.com/fe-diffusion

MFEM example for diffusion

f=0

$$M \in \mathbb{R}^{n \times n},\ M_{ij} = \int_{\Omega} \Psi_i \Psi_j \, d\Omega$$

$$D \in \mathbb{R}^{n \times n},\ D_{ij} = \int_{\Omega} \Psi_i \left(\nabla D \nabla \Psi_j \right) d\Omega$$

```
M = new BilinearForm(&fespace);
M->AddDomainIntegrator(new MassIntegrator());
M->Assemble();
M->FormSystemMatrix(ess_tdof_list, M_matrix);

D = new BilinearForm(&fespace);
ConstantCoefficient d_coeff(d_value);
D->AddDomainIntegrator(new DiffusionIntegrator(d_coeff));
D->Assemble();
D->FormSystemMatrix(ess_tdof_list, D_matrix);
```

Usage example

```
ModelInitializer::DefineMFEMSubstanceOnMesh(
      mesh, kSubstance1, "kSubstance1", order, dimension,
      bdm::experimental::MFEMODESolver::kRK4Solver,
      bdm::experimental::PDEOperator::kDiffusion,
      InitializeGridValues, parameters);
```

Q & A

Backup

Notation:

$$x \in \Omega \subset \mathbb{R}^3 \qquad \Phi : (\Omega \times \bar{T}) \to \mathbb{R}$$

$$t \in \bar{T} = (0,T] \qquad f : (\Omega \times \bar{T}) \to \mathbb{R}$$

Equation:

$$\partial_t \Phi - \nabla D \nabla \Phi = f$$

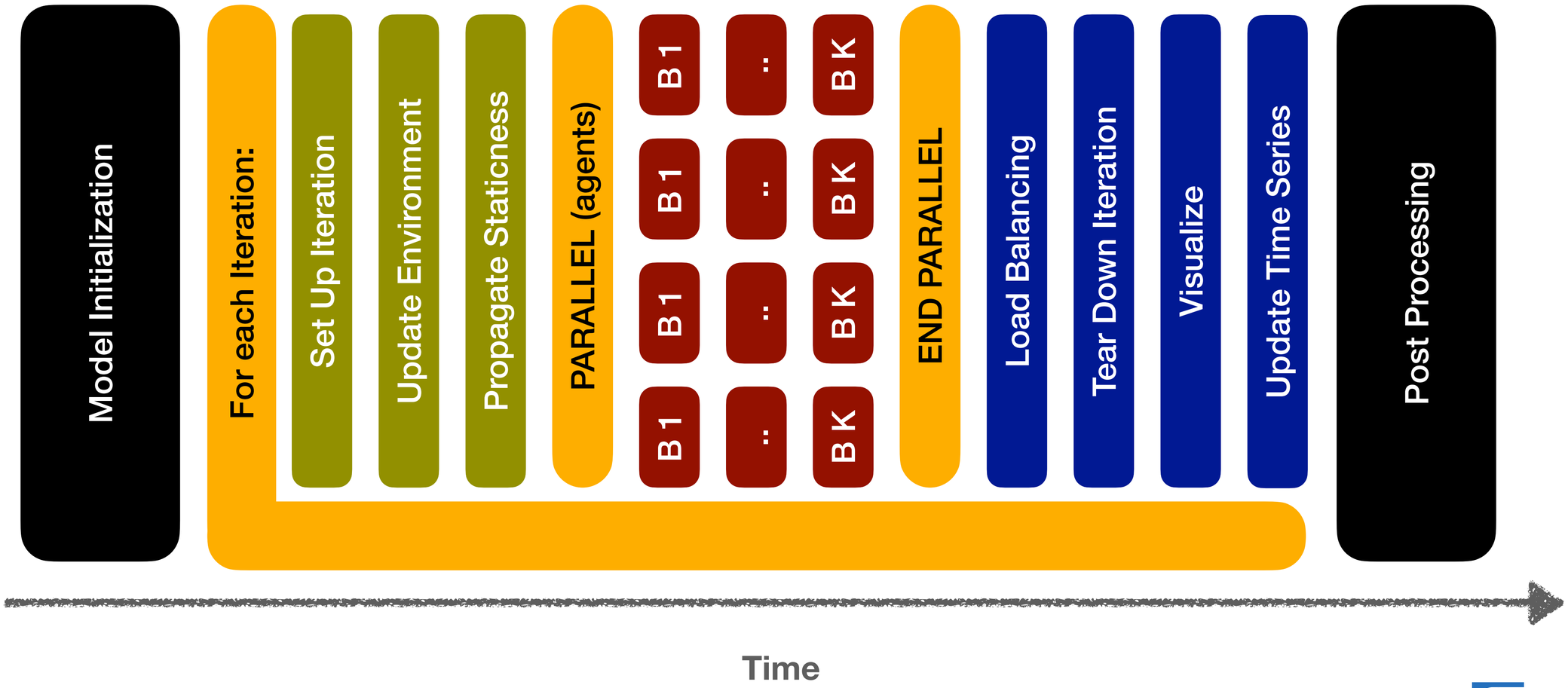
Boundaries:

$$\Phi(x,0) = g(x) \qquad \forall x \in \Omega$$

$$\frac{\partial \phi}{\partial n} = 0 \qquad \forall x \in \partial\Omega \ \wedge \forall t \in \bar{T}$$

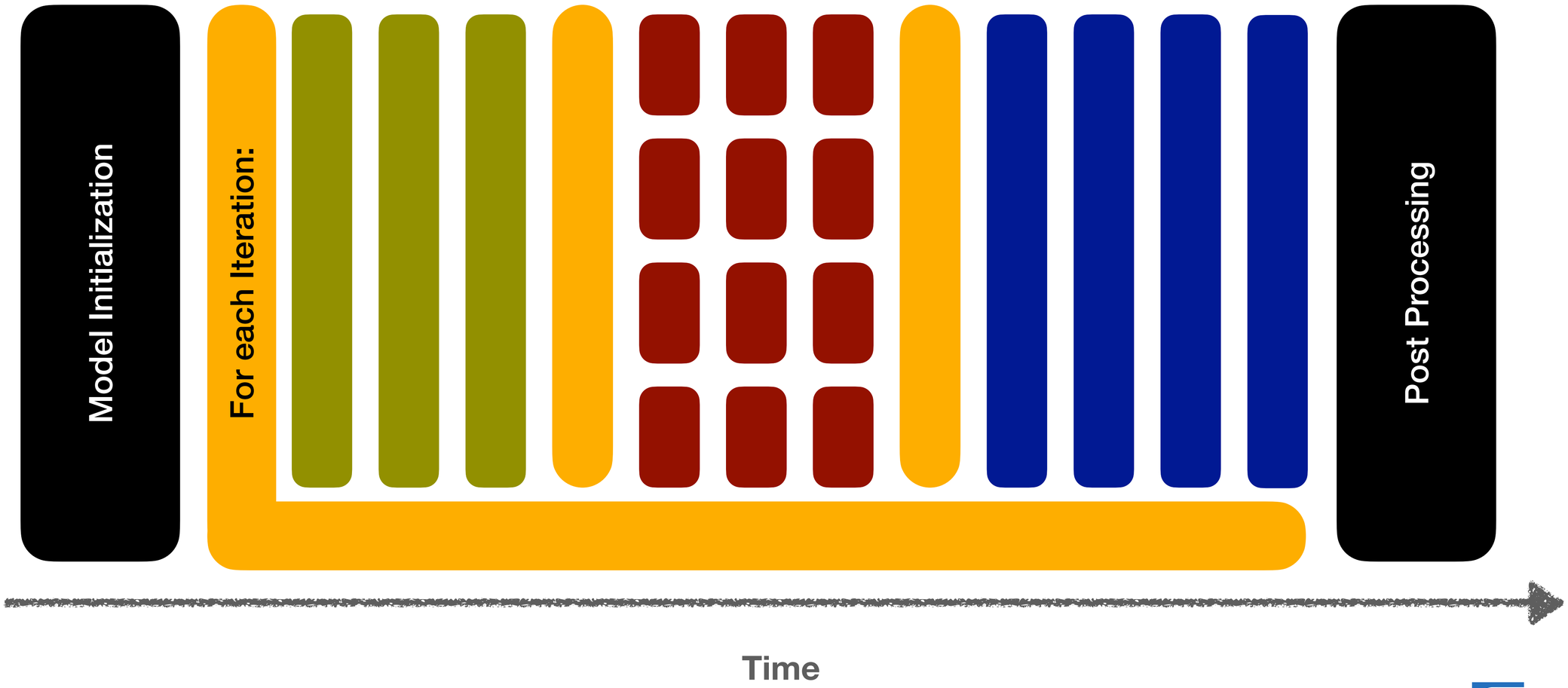
BioDynaMo's ABM engine

(Without user-defined operations)



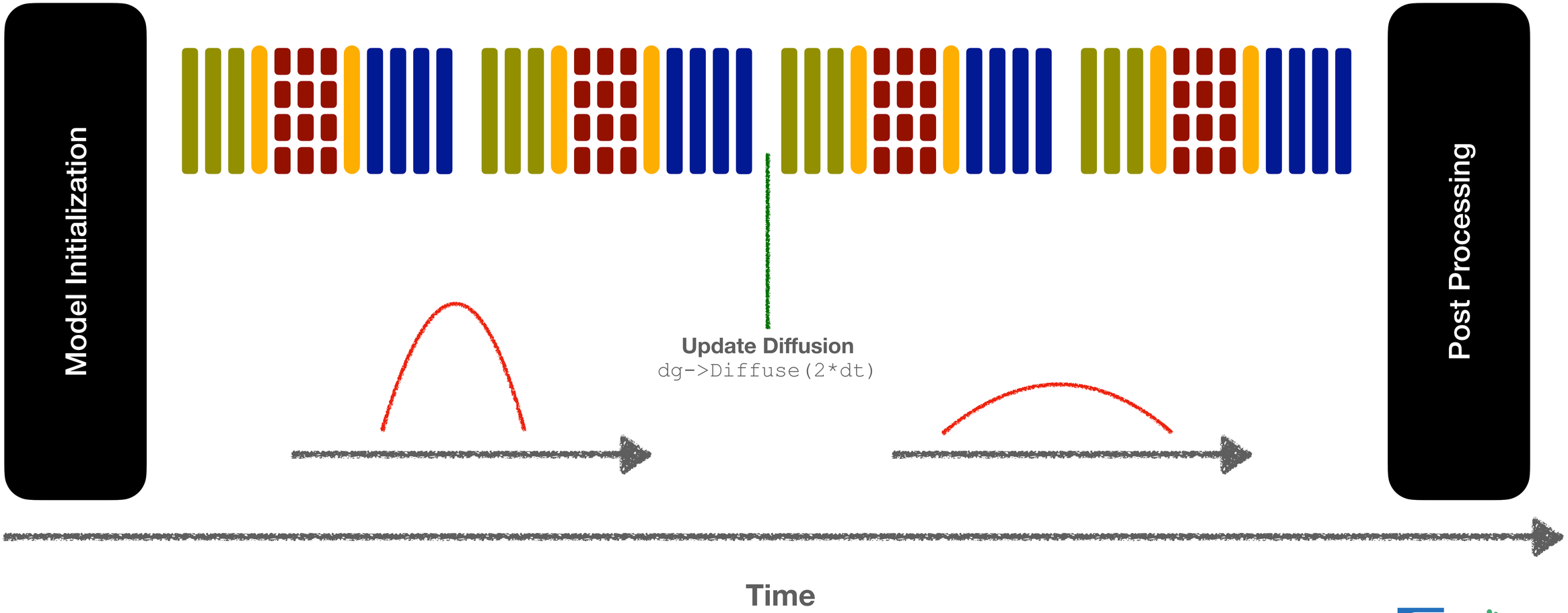
BioDynaMo's ABM engine

(Without user-defined operations)



BioDynaMo's ABM engine

(Without user-defined operations)



Method of Lines

Main idea

1. Get a clear understanding of the physics that you want to model and write down the corresponding PDE (incl. boundaries)

2. Discretise all but one dimension with FE, FV, or FD
   (Practically: discretise space, leave time continuous)

3. Derive the ODE system & construct matrices

4. Solve ODE system with established time-integrators

5. Critically reflect all steps and ideally discuss with colleagues

Spatial discretisation

MFEM - Finite Element Discretization Library

- Use a FE approach for spatial discretisation
- Use MFEM library to construct matrices and time integration
- MFEM offers various interfaces towards highly efficient solvers
- MFEM offers multiple options for parallelism

$$M \frac{\partial \vec{u}}{\partial t} = D \vec{u} + \vec{f}$$

$$M \in \mathbb{R}^{n \times n},\ M_{ij} = \int_{\Omega} \Psi_i \Psi_j \, d\Omega$$

$$D \in \mathbb{R}^{n \times n},\ D_{ij} = \int_{\Omega} \Psi_i \left(\nabla D \nabla \Psi_j \right) d\Omega$$

$$\vec{f} \in \mathbb{R}^{n},\ f_i = \int_{\Omega} \Psi_i f \, d\Omega$$

Software structure

bdm::TimeDependentScalarField3d

- Handles mfem related objects
- MFEM mesh
- ODE solver
- PDE operator
- Continuum values

mfem:: TimeDependentOperator

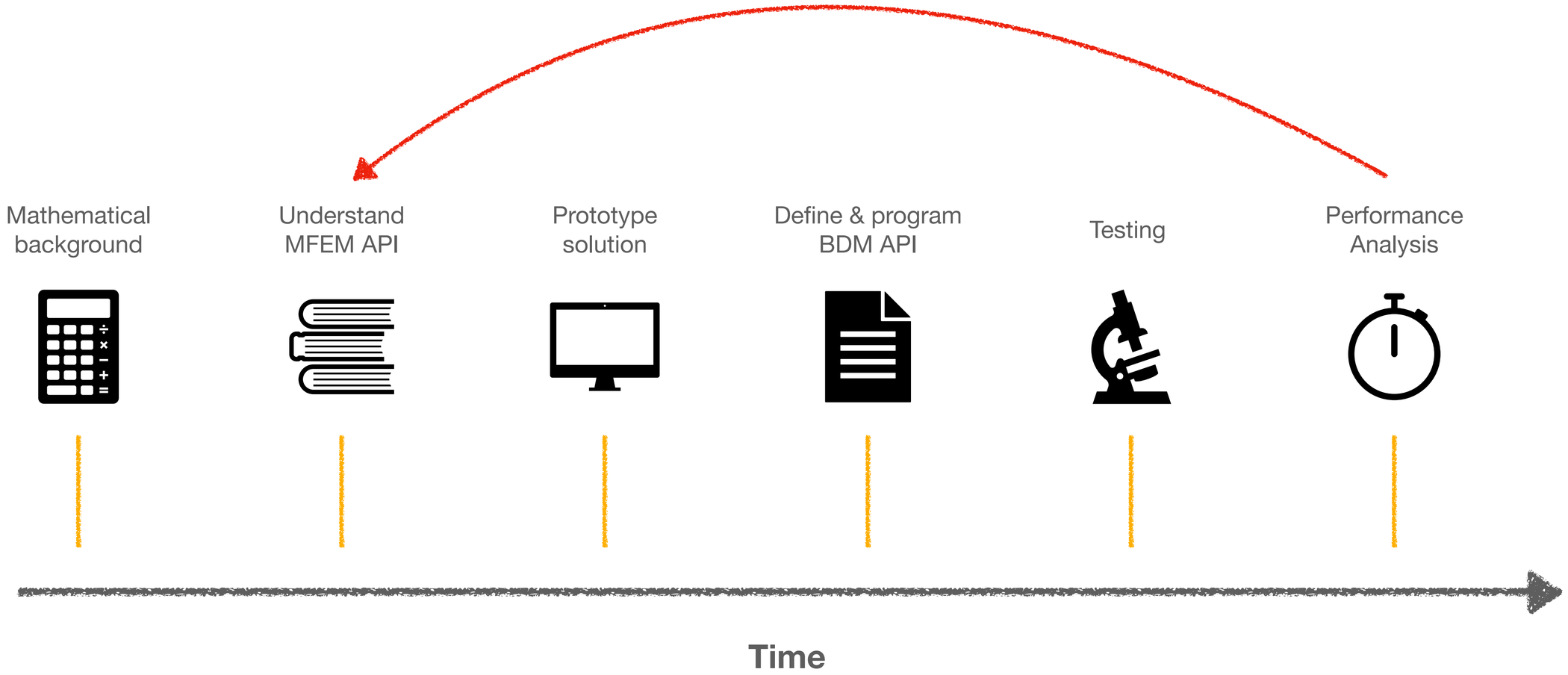
→ **bdm::MolOperartor**

→ **bdm::DiffusionOperator**

→ **bdm::ConductionOperator**

→ **bdm::DiffusionOperatorPerformance**

Development cycle



Coupling - Retrieving continuum information

Cycle 1

```
elem,ip  = mfem::Mesh.FindPoints(agent->pos);
value = mfem::GridFunction.GetValue(elem, ip);
```

Runtimes:

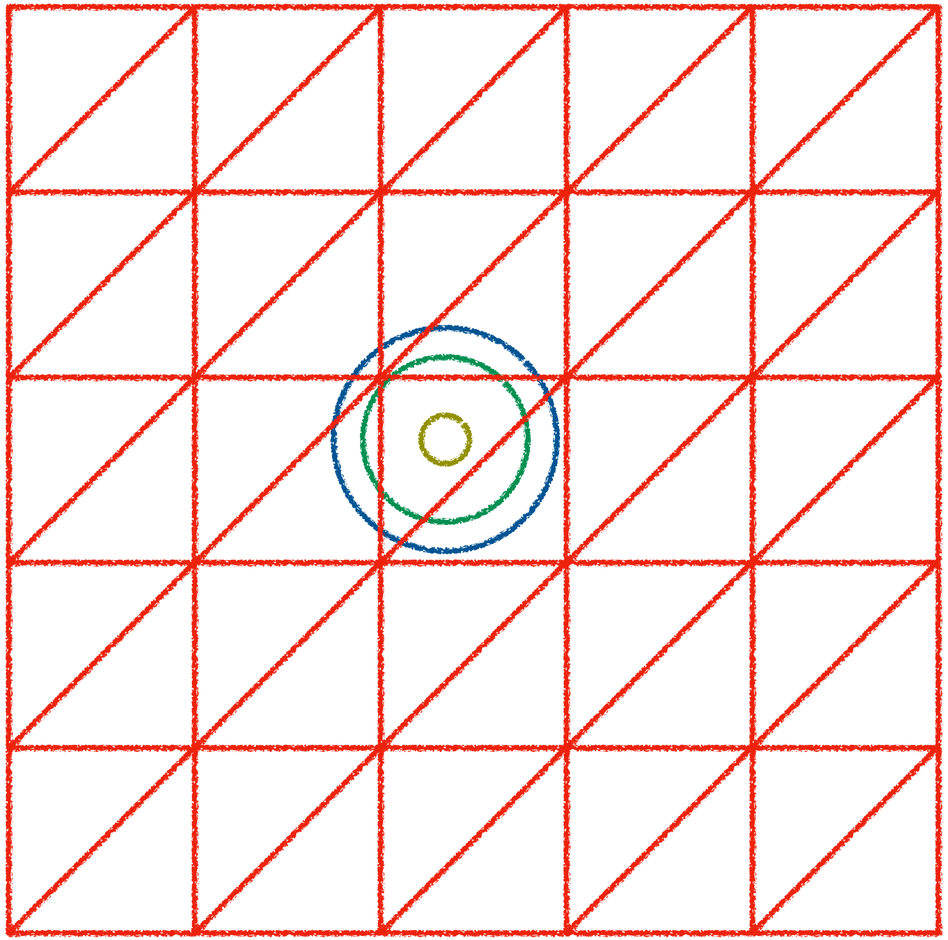
71 ns * n_elements * n_agents

Cycle 3

```
elem,ip  = ElementFinder.Octree(agent->pos);
value = mfem::GridFunction.GetValue(elem, ip);
```

Runtimes:

Static agents: 1 micro second
Moving agents: 8 micro seconds



Cycle2

```
If agent->HasElement():
   bdm::Continuum.VerifyElement(agent);
If not agent->HasElement():
   elem,ip  = mfem::Mesh.FindPoints(agent->pos);
agent->SaveElement(elem);
value = mfem::GridFunction.GetValue(elem, ip);
```

Runtimes:

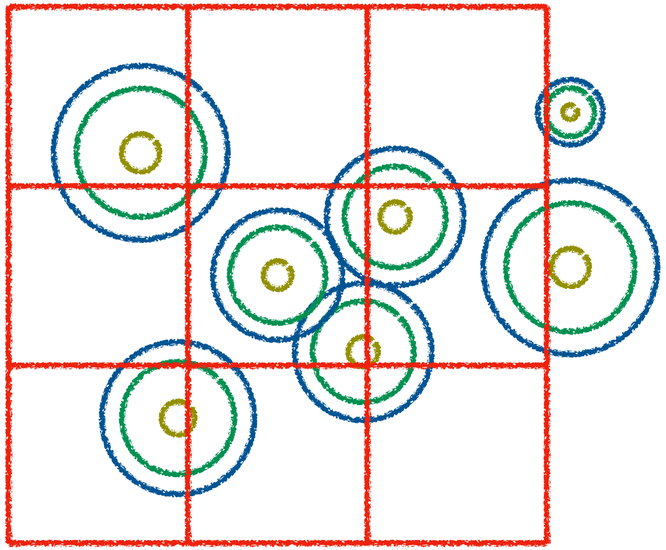
Static agent:
1 micro second
Moving agent:
55 micro seconds

Coupling - Retrieving agent information

Idea 1

1) Use uniform grid for efficient computation of a density function.
2) Pseudo Code:

```
sum = 0;
auto treat_neighbor = [&](agent* a){
  sum += Gaussian3D(x_in - a->pos);
};
ForAllNeighbors(treat_neighbor);
// + Normalisation
```



Idea 2

1) Use uniform grid for efficient computation of a density function.
2) Pseudo Code:

```
sum = 0;
auto treat_neighbor = [&](agent* a){
  sum += 1;
};
ForAllNeighbors(treat_neighbor);
// + Normalisation
```

Performance considerations

- How expensive is the treatment of ABMs and the PDE?
- Are all mesh resolutions a good choice?
- What kind of backends are available? BDM uses OMP at the moment.
- Possibly advanced splitting with MPI?