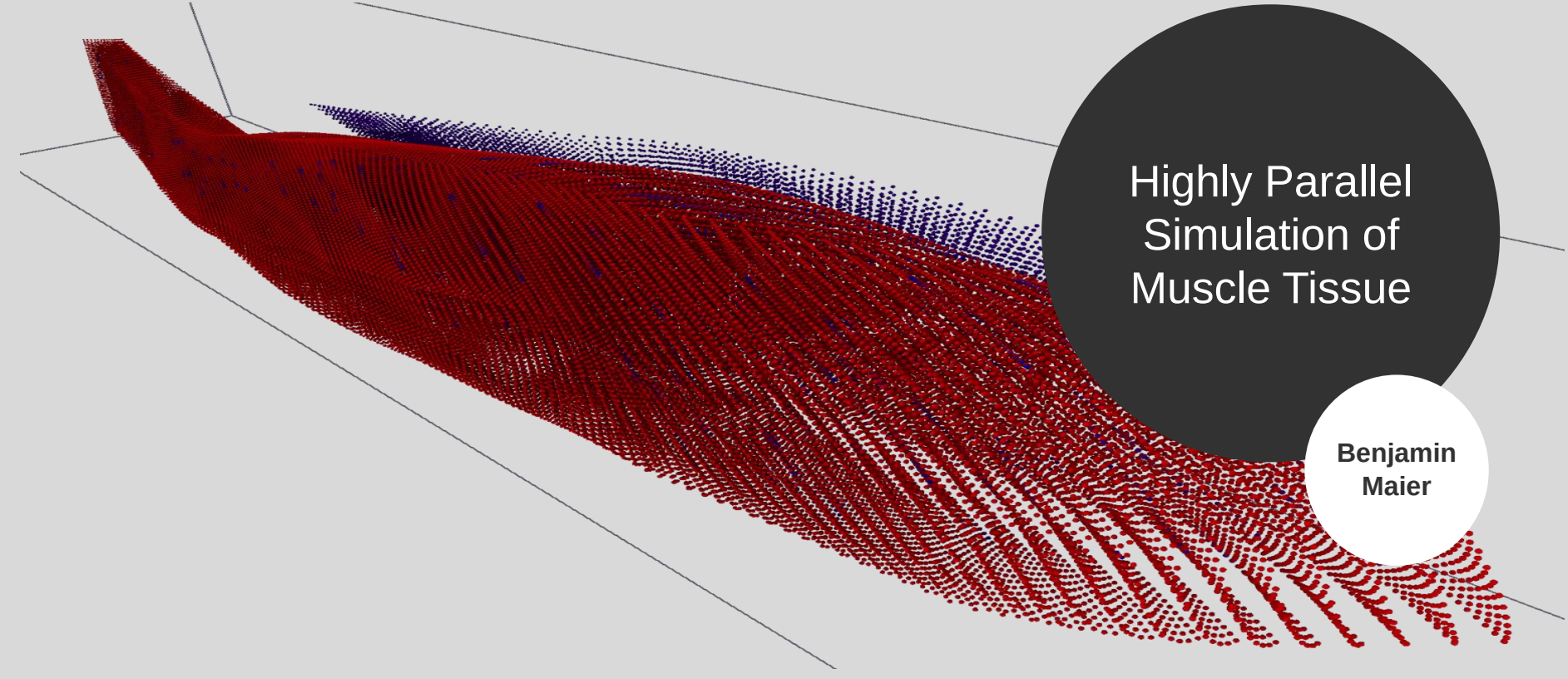
University of Stuttgart
Germany

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND



Highly Parallel Simulation of Muscle Tissue

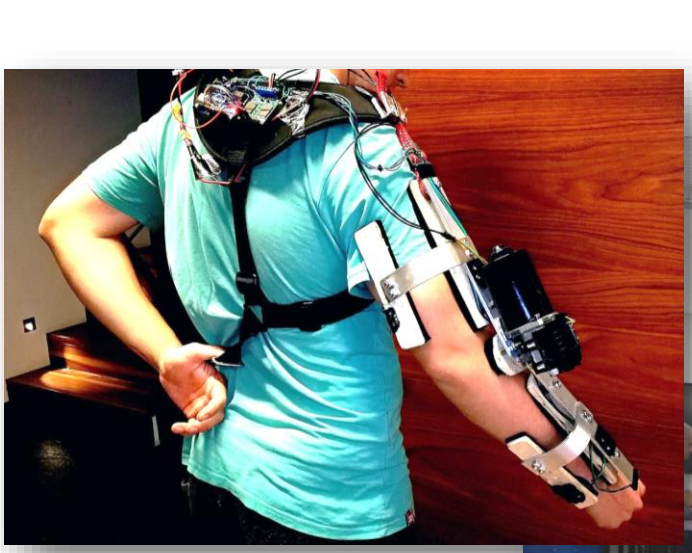
Benjamin Maier

IPVS
Simulation of Large Systems
Institute for Parallel and Distributed Systems

DFG Deutsche Forschungsgemeinschaft

IRTG
Soft Tissue Robotics

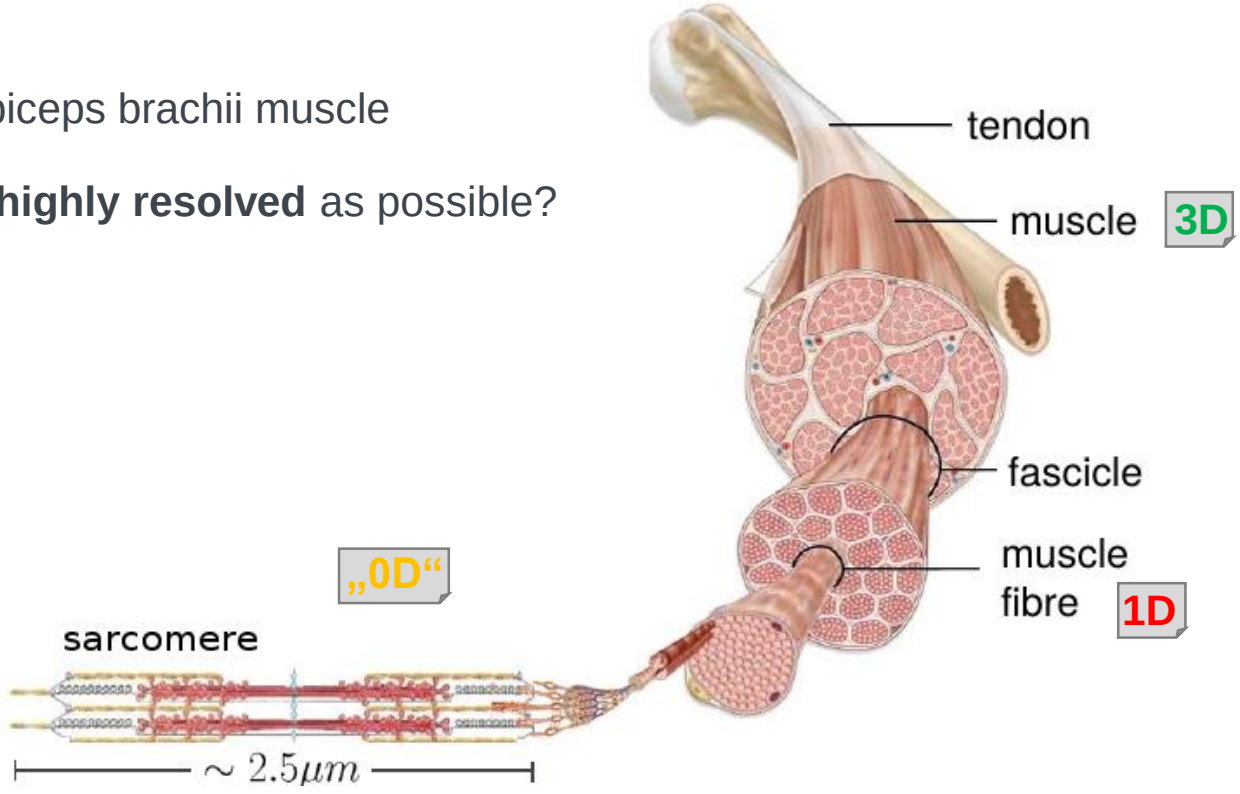
Motivation



Motivation

Anatomy of skeletal muscle

- **Multi-scale** architecture
- 270,000 muscle fibers in biceps brachii muscle
- How can we **simulate** as **highly resolved** as possible?
- Answer:

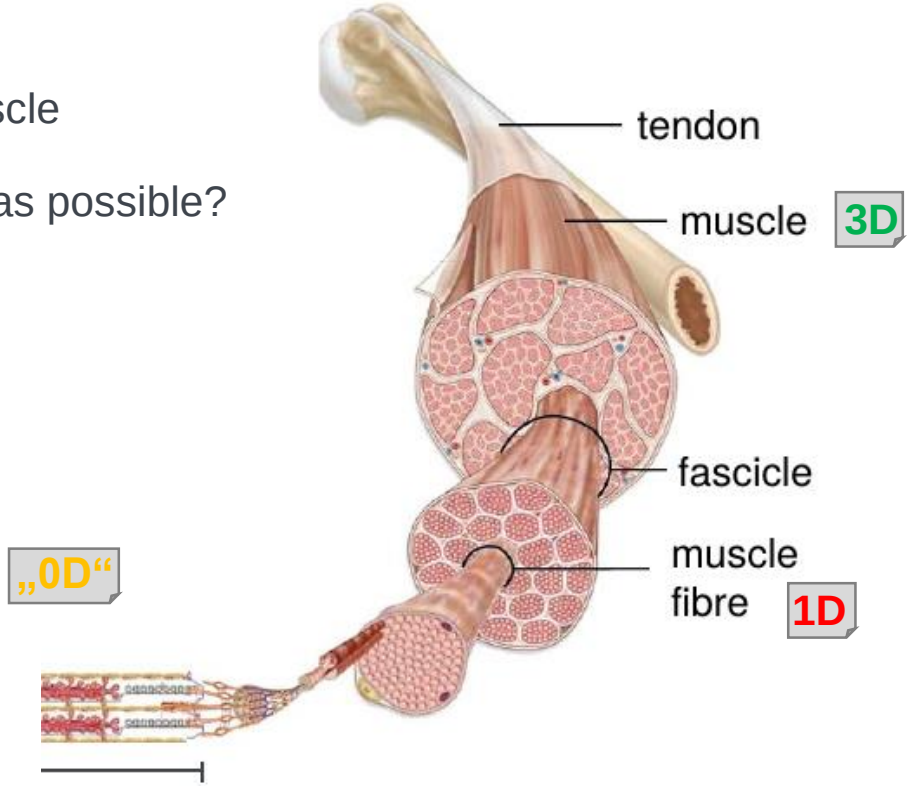


Motivation

Anatomy of skeletal muscle

- **Multi-scale** architecture
- 270,000 muscle fibers in biceps brachii muscle
- How can we **simulate** as **highly resolved** as possible?
- Answer:

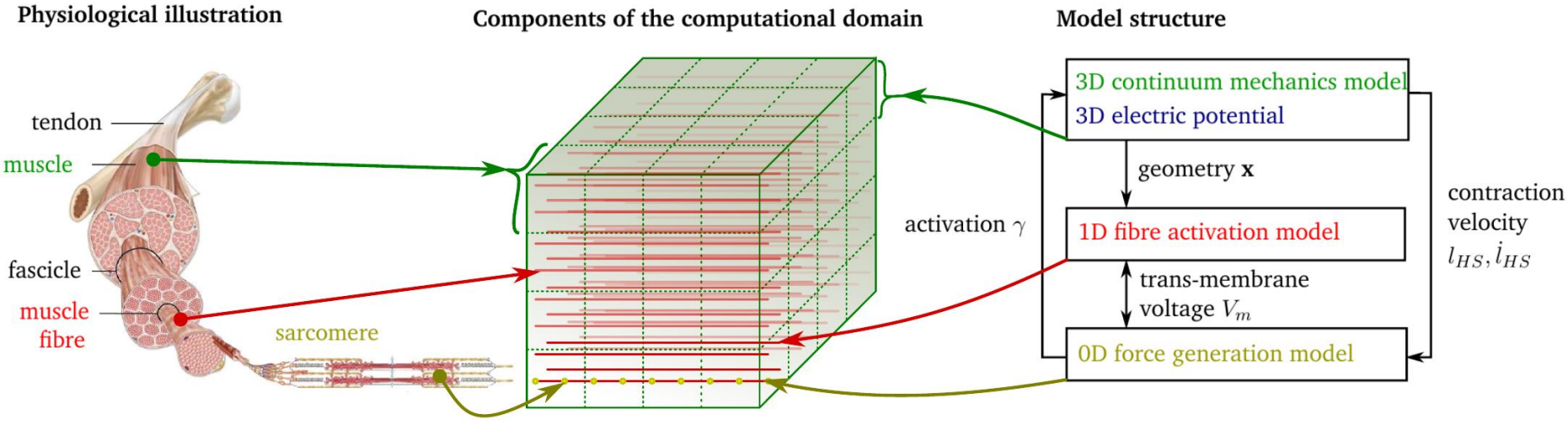
1. High Performance Computing
2. Efficient software



Contents

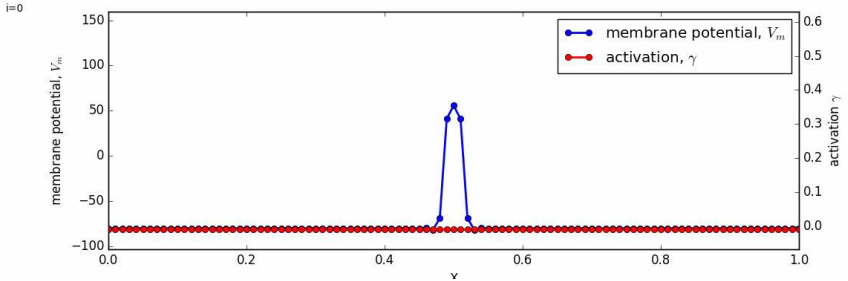
Contents

Mathematical Description

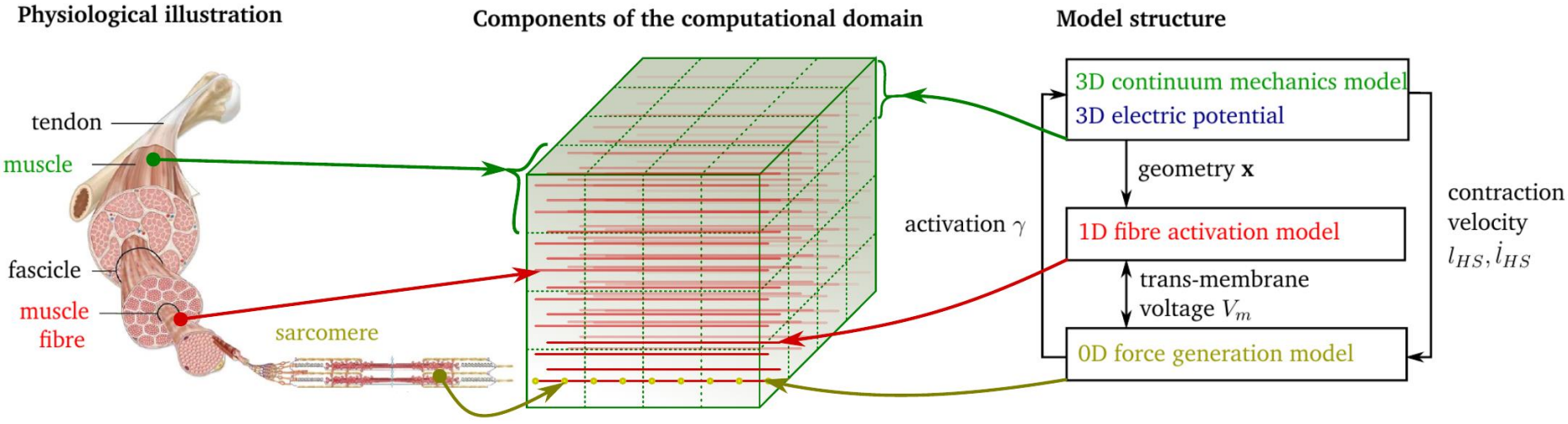


$$\frac{\partial V_m}{\partial t} = \underbrace{\frac{\sigma_{\text{eff}}}{A_m C_m} \frac{\partial^2 V_m}{\partial s^2}}_{\text{diffusion term}} - \frac{1}{C_m} I_{\text{ion}}(V_m, \mathbf{y})$$

$$\underbrace{\frac{\partial \mathbf{y}}{\partial t} = G_{\mathbf{y}}(V_m, \mathbf{y})}_{\text{reaction term}}$$



Mathematical Description



$$\frac{\partial V_m}{\partial t} = \underbrace{\frac{\sigma_{\text{eff}}}{A_m C_m} \frac{\partial^2 V_m}{\partial s^2}}_{\text{diffusion term}} - \frac{1}{C_m} I_{\text{ion}}(V_m, \mathbf{y})$$

$$\underbrace{\frac{\partial \mathbf{y}}{\partial t} = G_{\mathbf{y}}(V_m, \mathbf{y})}_{\text{reaction term}}$$

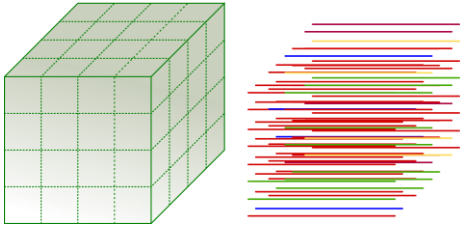
$$\operatorname{div}\left((\boldsymbol{\sigma}_e + \boldsymbol{\sigma}_i)\operatorname{grad}(\phi_e)\right) + \operatorname{div}\left(\boldsymbol{\sigma}_i \operatorname{grad}(V_m)\right) = 0$$

$$\int_\Omega \left(\boldsymbol{\sigma}_{\text{passive}}(\boldsymbol{\varepsilon}) + \boldsymbol{\sigma}_{\text{active}}(\gamma)\right) : \delta\boldsymbol{\varepsilon} \, \mathrm{d}\mathbf{x} = \mathbf{f},$$

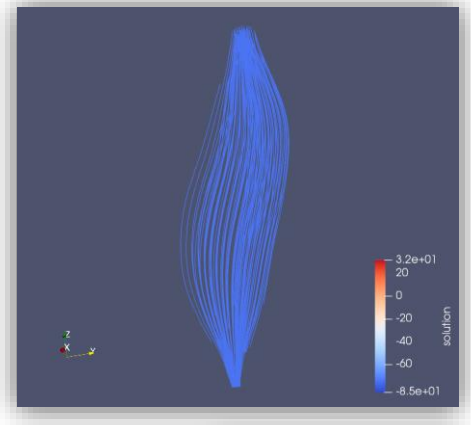
Incompressibility

Mathematical Description – Variants

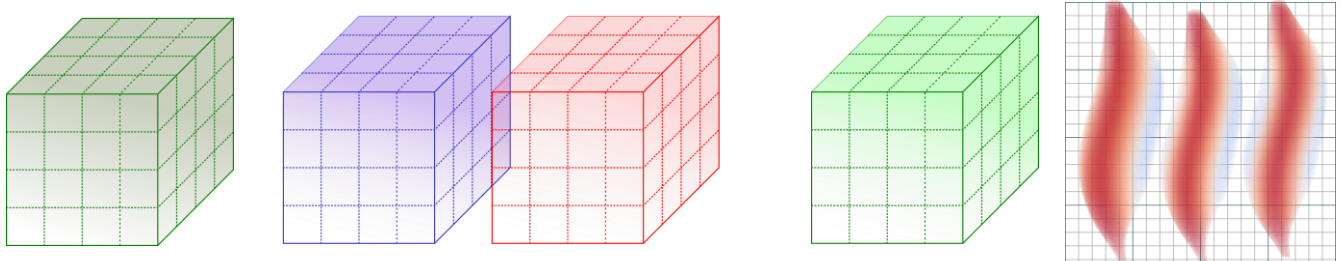
- Formulation A: *Multi-scale model*



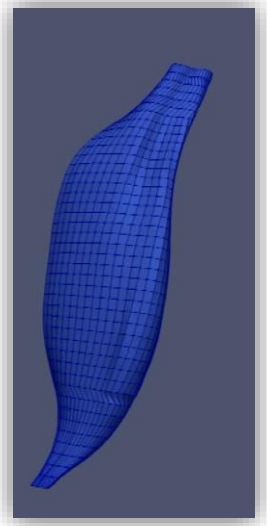
CM (3D) Fibers (1D), different motor units (MUs)



- Formulation B: *Multi-Compartment*, homogenized electrical potentials

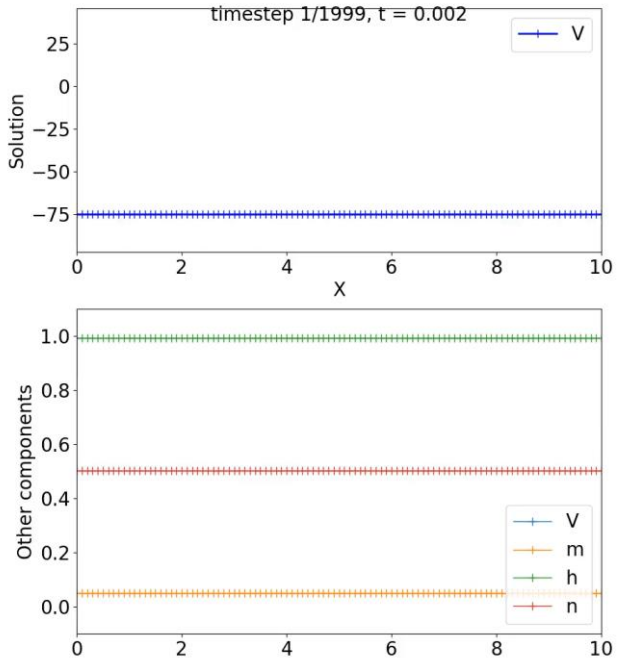


CM (3D) MU1 (3D) MU2 (3D) … MUn (3D)

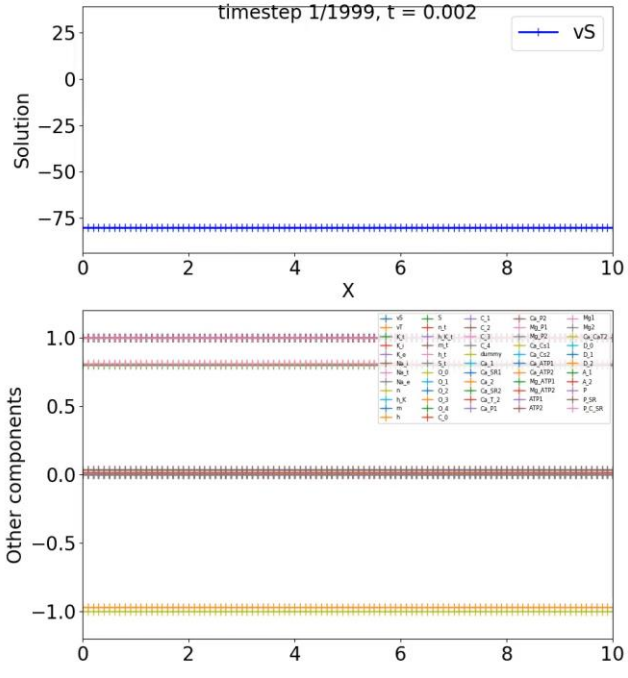


Mathematical Description – Variants

Fiber model: 0D-1D Monodomain Equation



Hodgkin, Huxley 1952



Shorten et al. 2007

Contents

Contents

Steps Towards a Highly Parallel Simulation

Improvements of Numerical Schemes

- Solution of reaction-diffusion equation using operator splitting

$$\frac{\partial V_m}{\partial t} = \frac{\sigma_{\text{eff}}}{A_m C_m} \frac{\partial^2 V_m}{\partial s^2} - \frac{1}{C_m} I_{\text{ion}}(V_m, \mathbf{y})$$

Steps Towards a Highly Parallel Simulation

Improvements of Numerical Schemes

- Solution of reaction-diffusion equation using operator splitting

$$\frac{\partial V_m}{\partial t} = -\frac{1}{C_m} I_{\text{ion}}(V_m, \mathbf{y})$$

Steps Towards a Highly Parallel Simulation

Improvements of Numerical Schemes

- Solution of reaction-diffusion equation using operator splitting

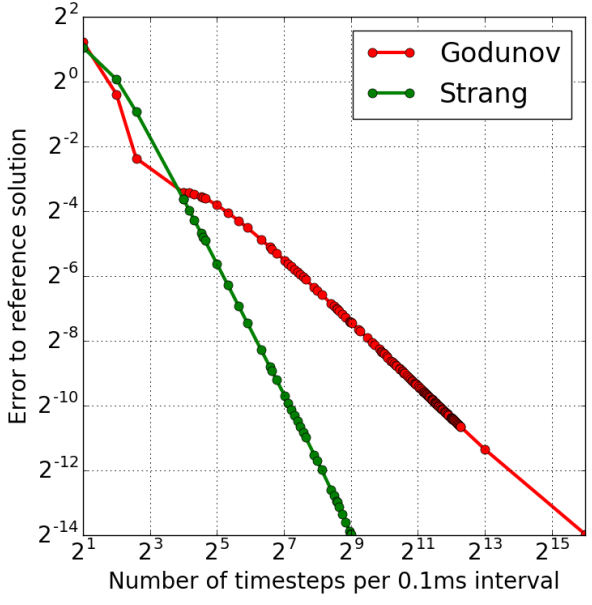
$$\frac{\partial V_m}{\partial t} = \frac{\sigma_{\text{eff}}}{A_m C_m} \frac{\partial^2 V_m}{\partial s^2}$$

Steps Towards a Highly Parallel Simulation

Improvements of Numerical Schemes

- Solution of reaction-diffusion equation using operator splitting

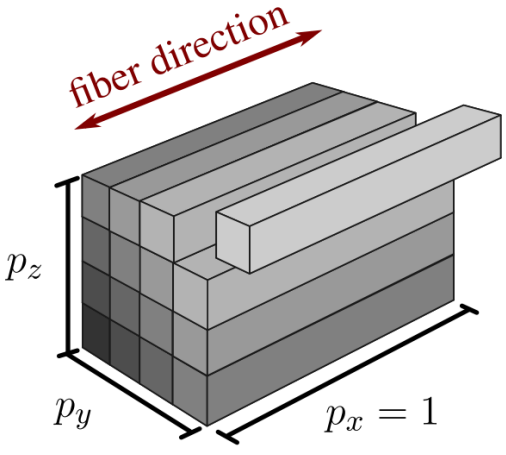
$$\frac{\partial V_m}{\partial t} = \frac{\sigma_{\text{eff}}}{A_m C_m} \frac{\partial^2 V_m}{\partial s^2} - \frac{1}{C_m} I_{\text{ion}}(V_m, \mathbf{y})$$



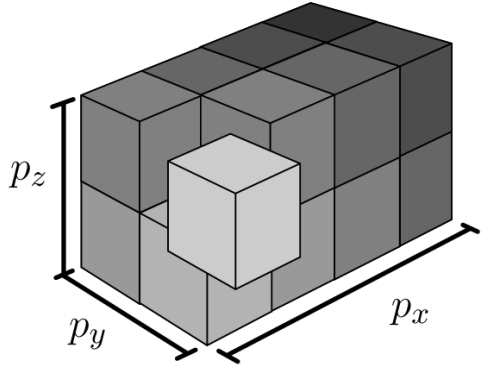
- Godunov: $h^2 \propto dt$
  Strang: $h \propto dt$

Steps Towards a Highly Parallel Simulation

Parallel-Computing Strategies



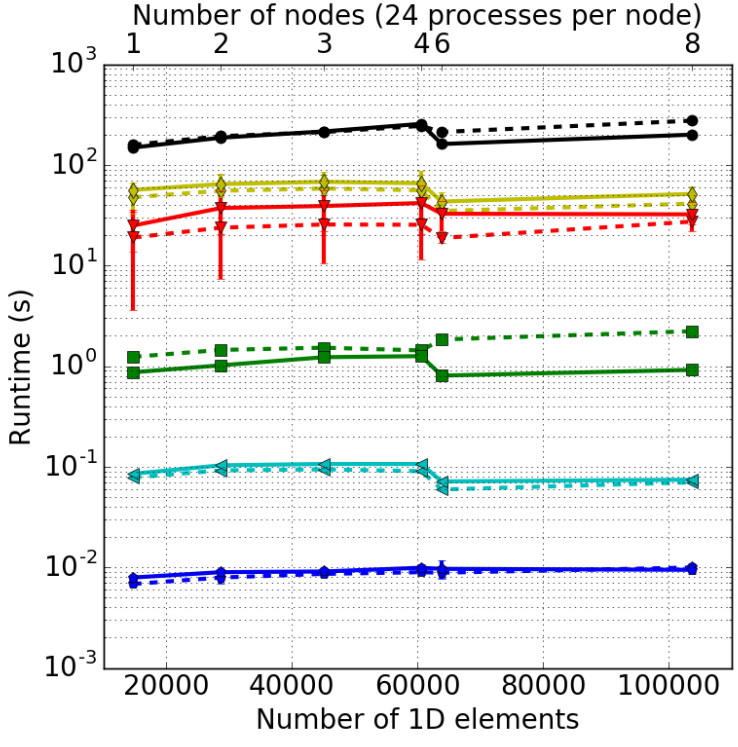
- Minimise communication for the 1D muscle fiber model

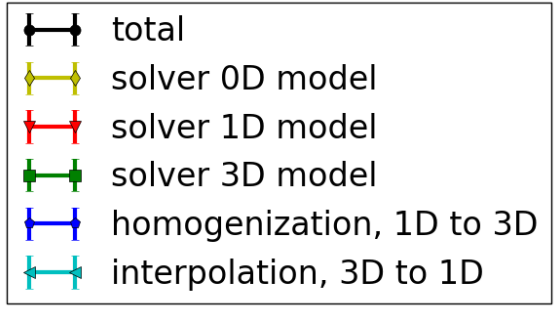


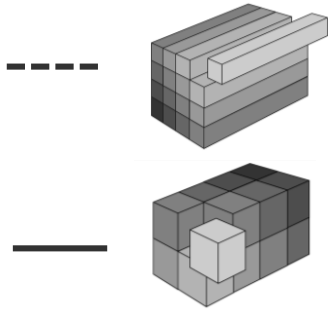
- Minimise communication for the 3D model

Steps Towards a Highly Parallel Simulation

Weak-Scaling Study

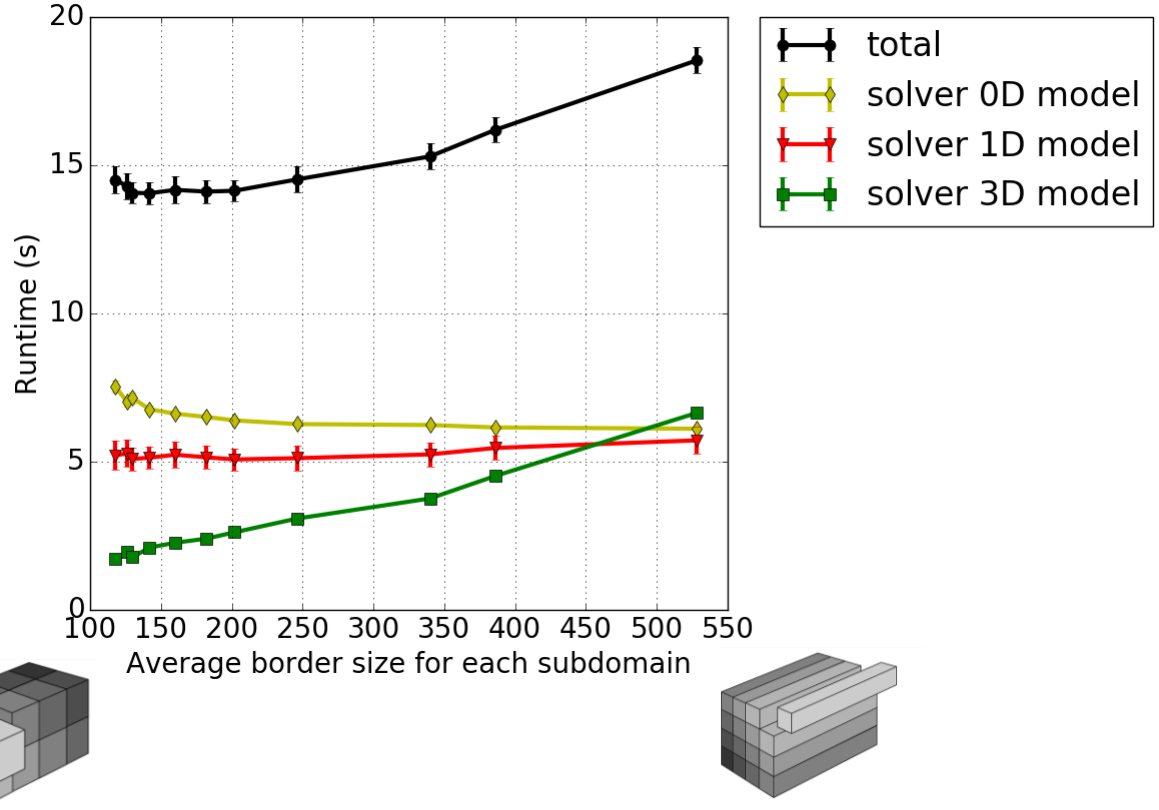






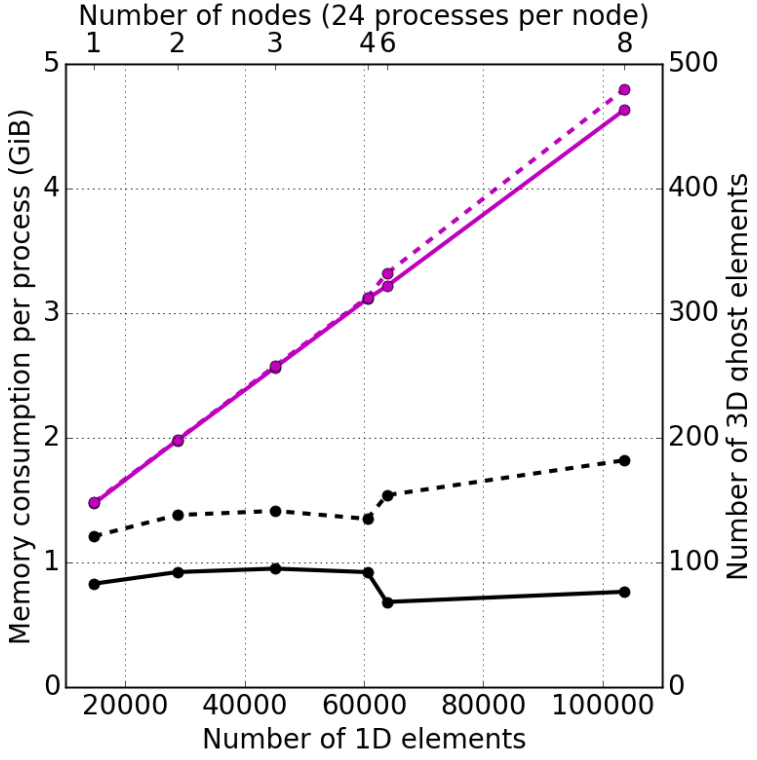
Steps Towards a Highly Parallel Simulation

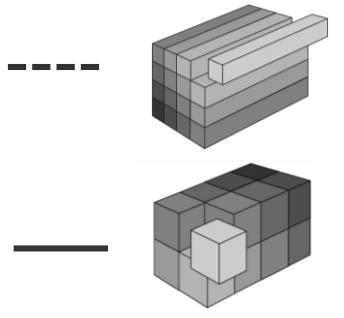
Parallelisation Strategy Study



Steps Towards a Highly Parallel Simulation

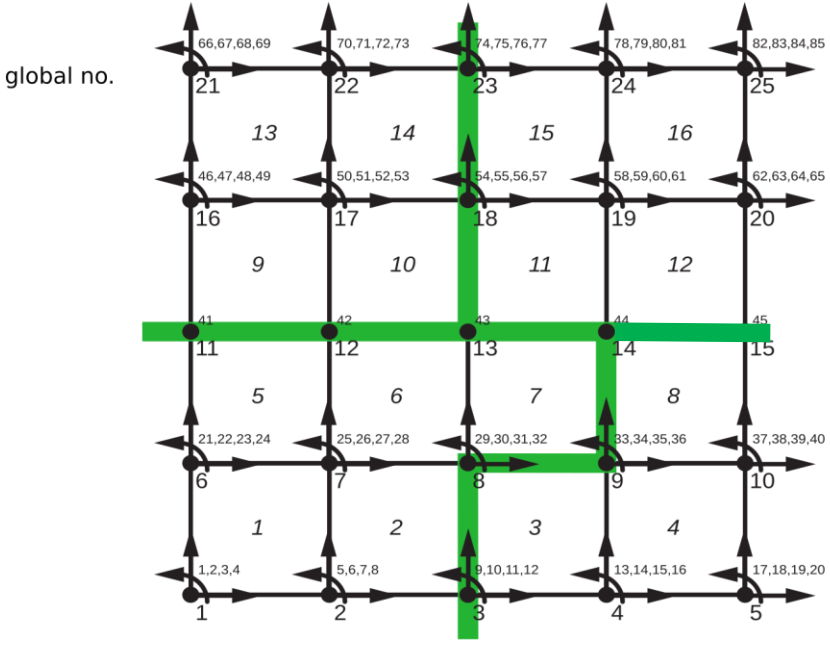
Memory Consumption

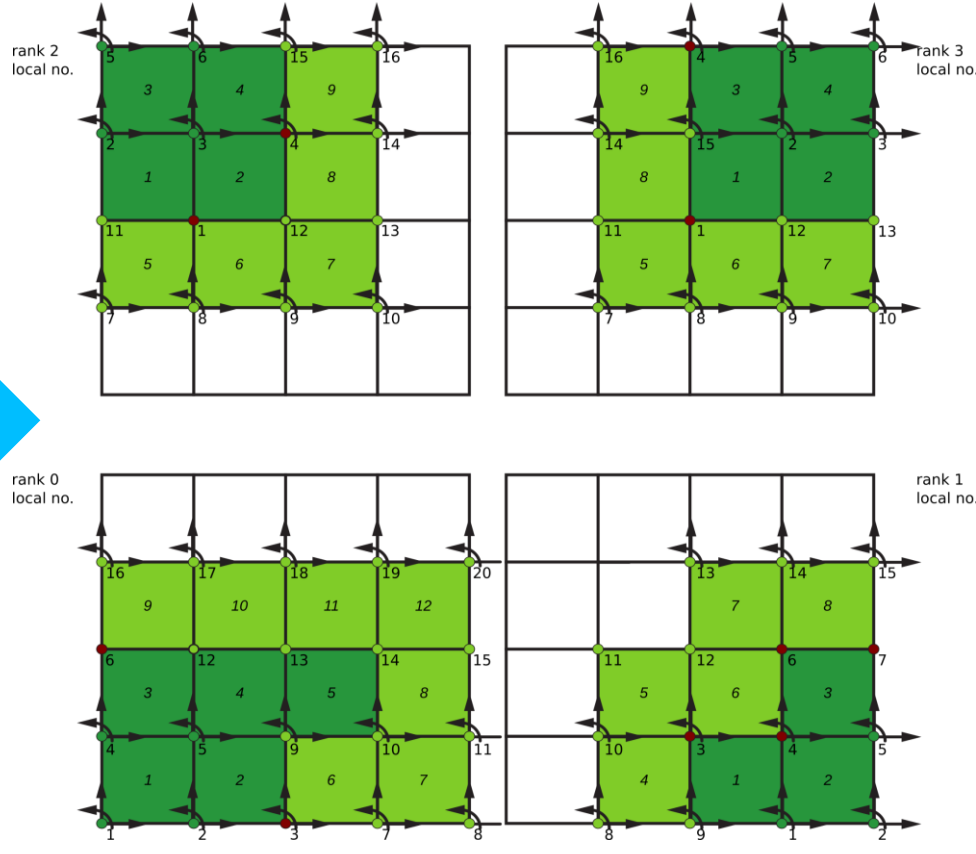




Steps Towards a Highly Parallel Simulation

Domain decomposition





Contents

- Mathematical Description
- Steps Towards a Highly Parallel Simulation
- Obtaining fiber geometry
- Software Framework *opendihu*
- Results

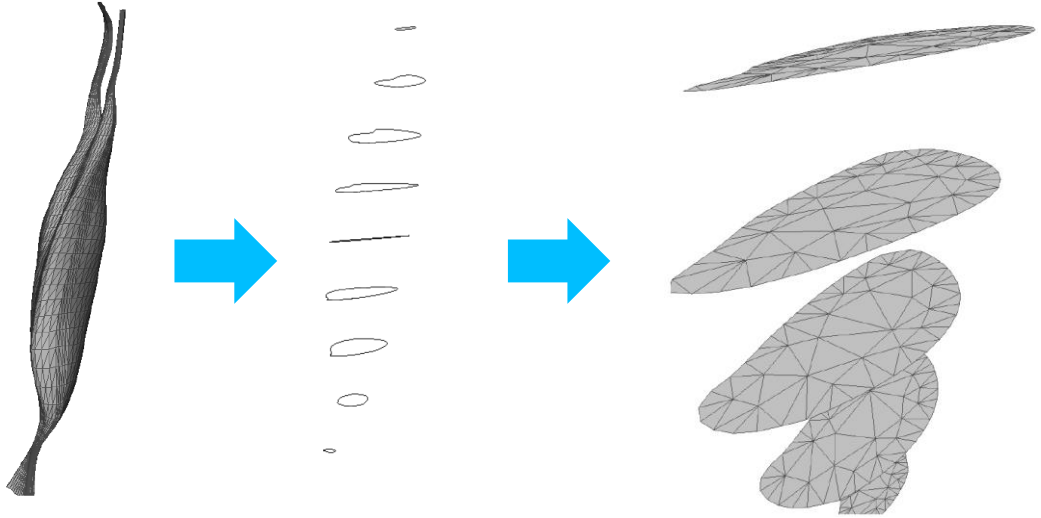
Contents

- Mathematical Description
- Steps Towards a Highly Parallel Simulation
- **Obtaining fiber geometry**
- Software Framework *opendihu*
- Results

Obtaining fiber geometry

- Fiber geometry inside a skeletal muscle can be estimated by streamlines of a potential flow problem [1]
- Algorithm to estimate 270,000 fibers for biceps brachii:

1. Create slices of surface mesh
2. Triangulate slices

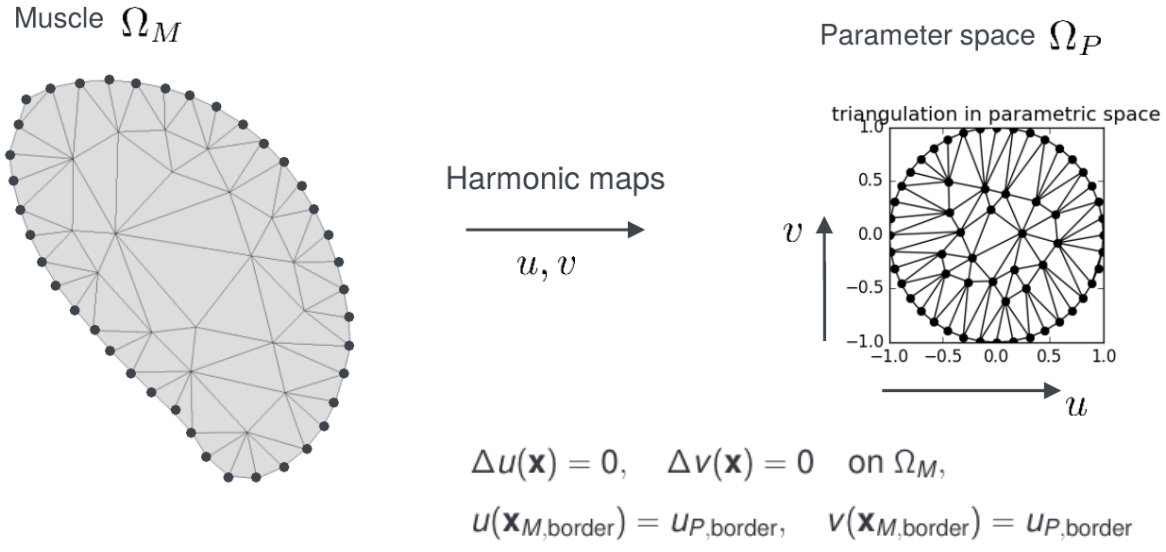


[1] Hon Fai Choi and Silvia S. Blemker. Skeletal muscle fascicle arrangements can be reconstructed using a laplacian vector field simulation. PLOS ONE, 8(10):1–7, 10 2013.

Obtaining fiber geometry

Algorithm to estimate 270,000 fibers for biceps brachii

1. Create slices of surface mesh
2. Triangulate slices
3. Compute harmonic maps from muscle domain to parameter space



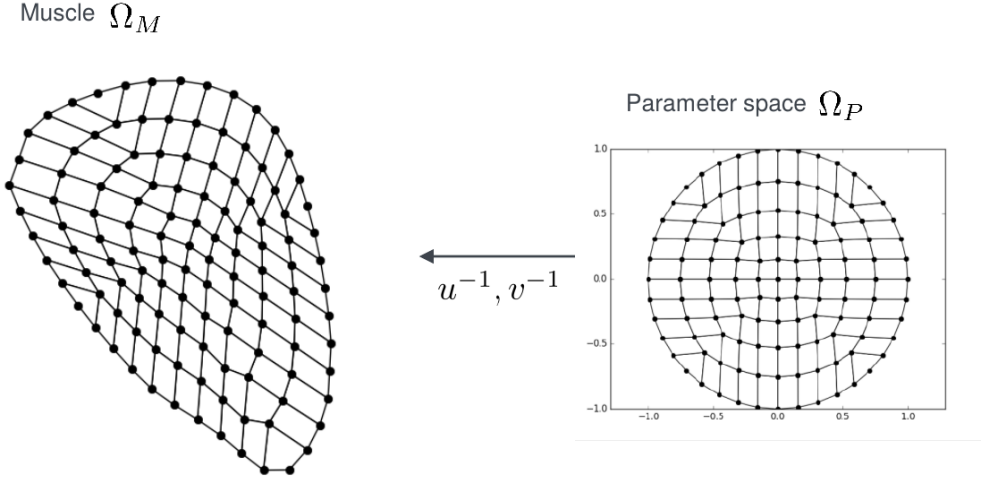
$$\Delta u(\mathbf{x}) = 0, \quad \Delta v(\mathbf{x}) = 0 \quad \text{on } \Omega_M,$$

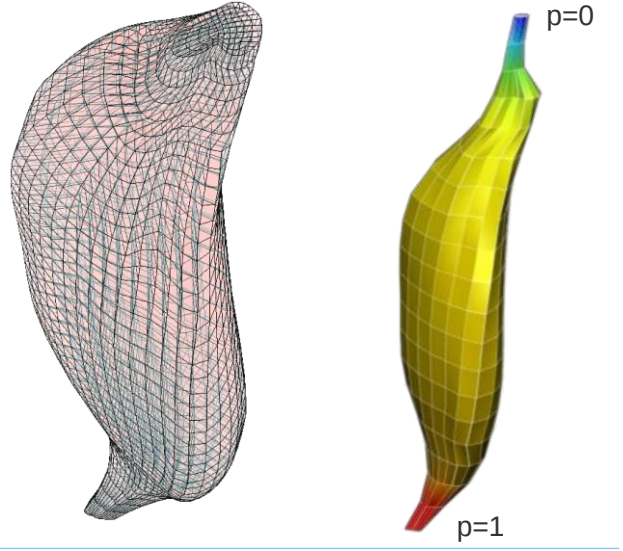
$$u(\mathbf{x}_{M,\text{border}}) = u_{P,\text{border}}, \quad v(\mathbf{x}_{M,\text{border}}) = u_{P,\text{border}}$$

Obtaining fiber geometry

Algorithm to estimate 270,000 fibers for biceps brachii

4. Project back a regular grid in parameter space
5. Form 3D mesh from 2D meshes on slices
6. Solve Laplacian flow problem: $\Delta p(\mathbf{x}) = 0, p(\mathbf{x}_{\text{bottom}}) = 1, p(\mathbf{x}_{\text{top}}) = 0$

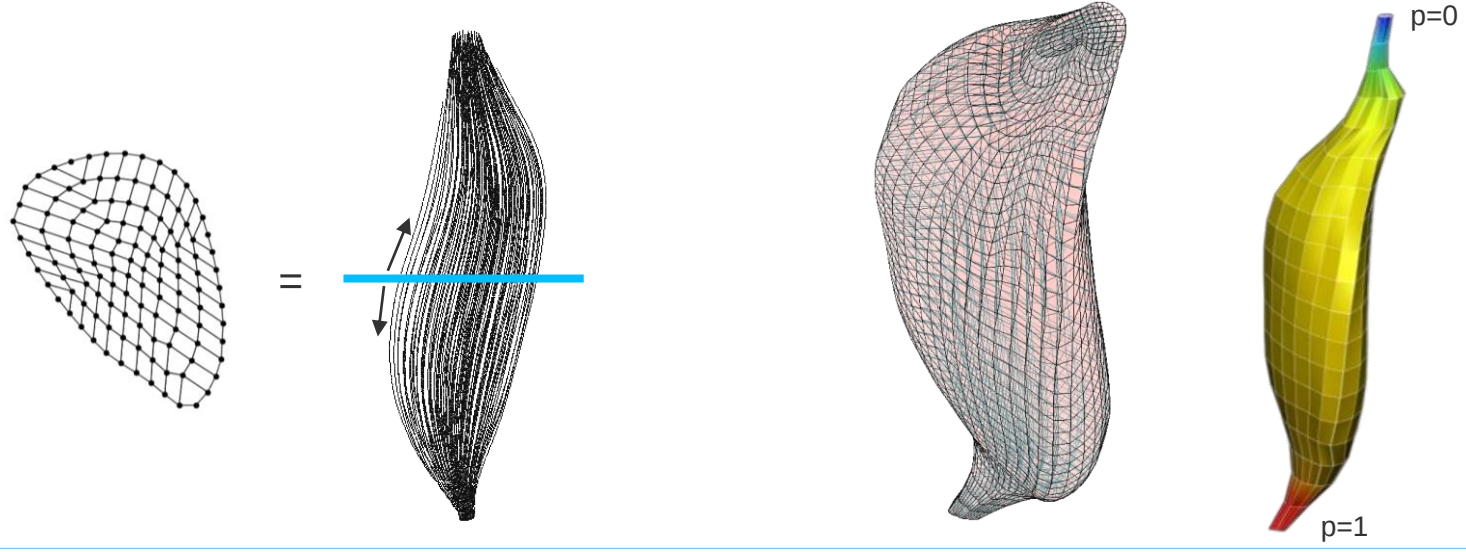




Obtaining fiber geometry

Algorithm to estimate 270,000 fibers for biceps brachii

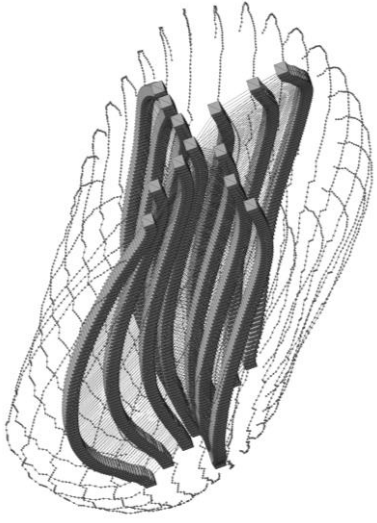
4. Project back a regular grid in parameter space
5. Form 3D mesh from 2D meshes on slices
6. Solve Laplacian flow problem: $\Delta p(\mathbf{x}) = 0, p(\mathbf{x}_{\text{bottom}}) = 1, p(\mathbf{x}_{\text{top}}) = 0$
7. Trace streamlines from center



Obtaining fiber geometry

Algorithm to estimate 270,000 fibers for biceps brachii

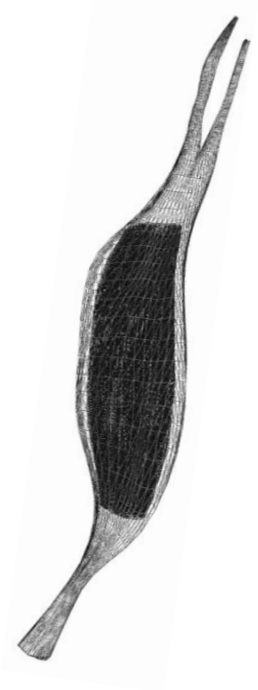
- Problem: Not feasible for 270,000 fibers
- Solution: Parallel algorithm
  - Perform steps 1.-6.
  - Trace streamlines from a „cross“ → creates 8 subdomains
  - In each subdomain in a separate process, perform steps 1.-6.
  - 1 process → 8 processes → 64 processes → …

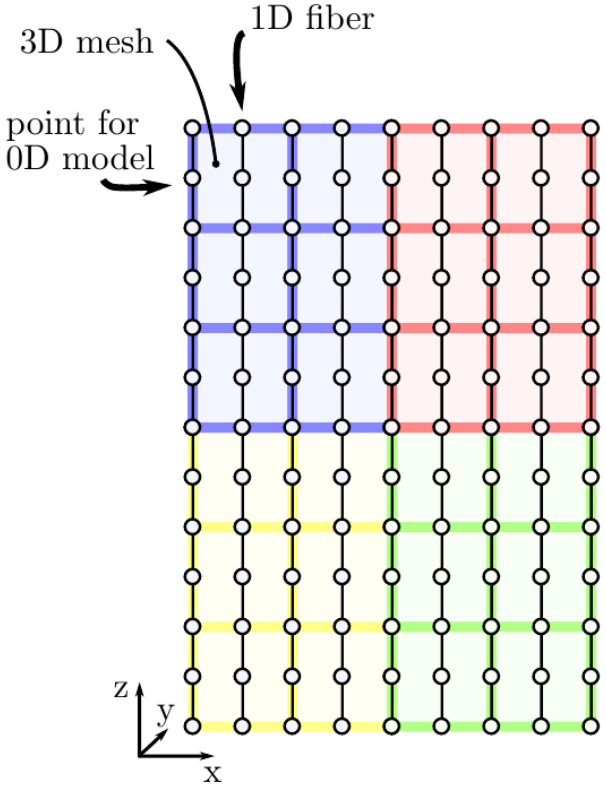


Obtaining fiber geometry

Algorithm to estimate 270,000 fibers for biceps brachii

- Result: structured 3D mesh and 1D fiber meshes
- 1D fibers aligned with 3D mesh
- Algorithmic advantages:
  - Parallel partitioning
  - Fast setup
  - Equal sizes
  - Neighbours implicitely given





Contents

- Mathematical Description
- Steps Towards a Highly Parallel Simulation
- Obtaining fiber geometry
- Software Framework *opendihu*
- Results

Contents

- Mathematical Description
- Steps Towards a Highly Parallel Simulation
- Obtaining fiber geometry
- **Software Framework *opendihu***
- Results

Software framework *opendihu*

- *opendihu*: „Digital Human“, Open source
- Computational framework, efficient for small and high number of processes
- Distributed memory parallelism
- Building upon existing libraries:
  MPI, PETSc, Python interpreter, …

Software framework *opendihu*

Differences to OpenCMISS Iron

	OpenCMISS Iron	opendihu
Language	Fortran 90/95: • no object-orientation • no polymorphism	C++14
Lines of code files with >1k lines	402k in 135 files 68 files	71k in 508 files 9 files
Unit tests	no	yes
Geometry input Model input	Hardcoded CellML	Binary files, Exfiles, Python scripting CellML
Data output format	Exelem/Exnode	Exelem/Exnode, VTK (Paraview), Python scripting, ADIOS for in-situ visualization
Meshes	Unstructured (parallel)	Unstructured (serial), Structured (parallel)

Software framework *opendihu*

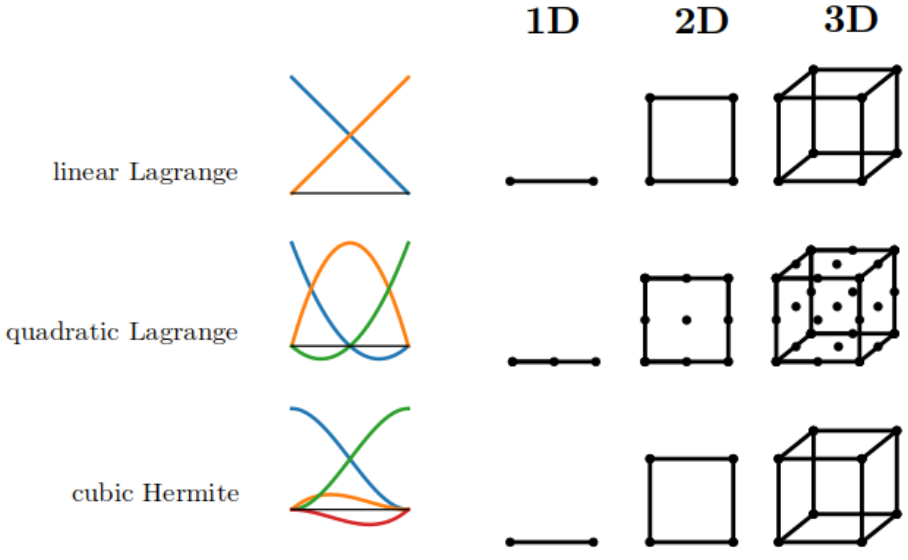
Nested solvers

- Laplace problem: $\Delta u = 0$

FiniteElementMethod<…>

- Mesh::Structured<2>
- BasisFunction::Lagrange<2>
- Quadrature::Gauss<2>
- Equation::Static::Laplace

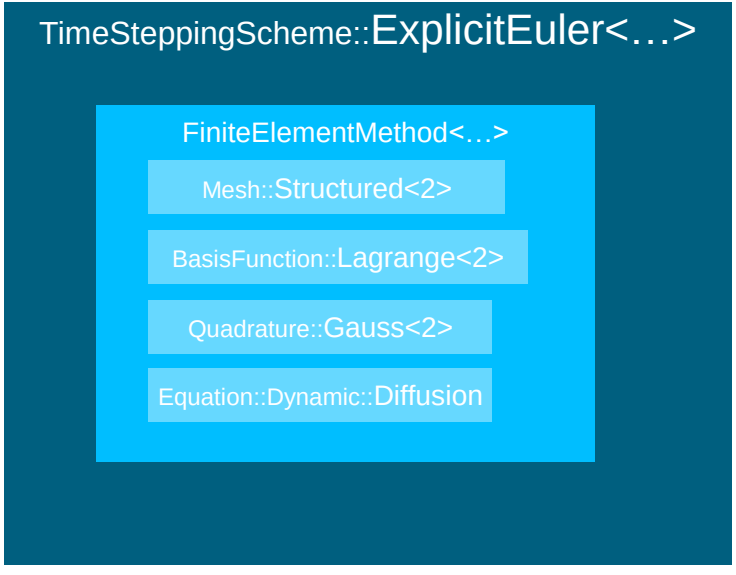
Options:



Software framework *opendihu*

Nested solvers

- Diffusion problem: $u_t - \Delta u = 0$



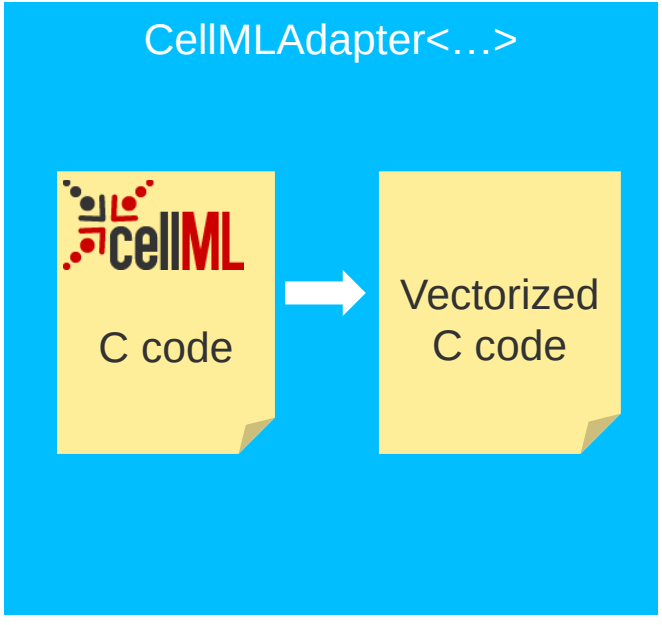
Options:

- ExplicitEuler
- Heun
- ImplicitEuler
- CrankNicolson

Software framework *opendihu*

Nested solvers

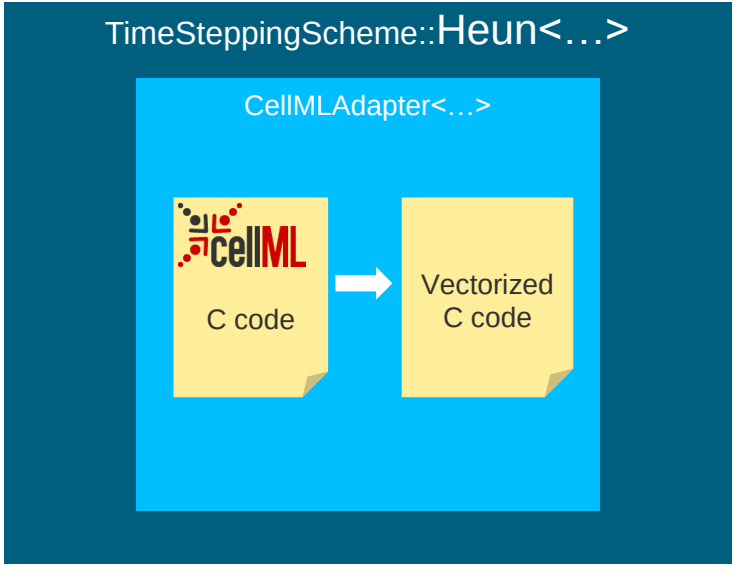
- Monodomain: $u_t - \Delta u = R(u, y)$
  $y_t = f(y, t)$



Software framework *opendihu*

Nested solvers

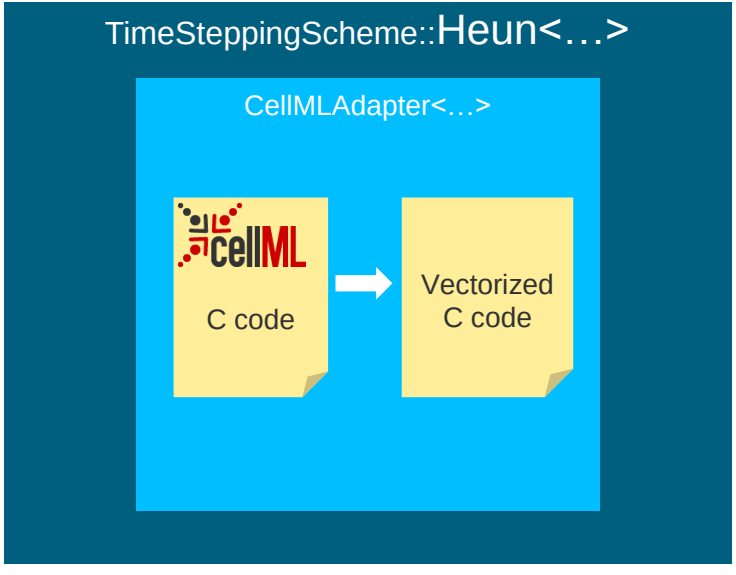
- Monodomain: $\mathrm{u}_t - \Delta u = R(u, y)$
  $y_t = f(y, t)$

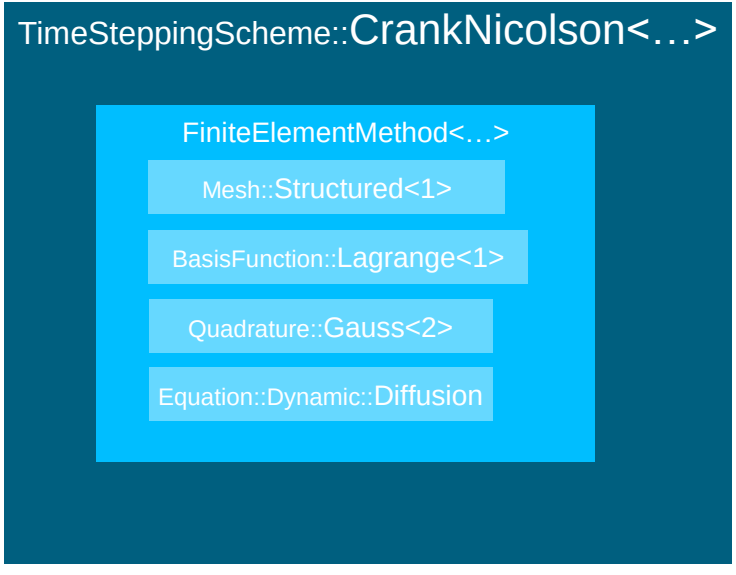


Software framework *opendihu*

Nested solvers

- Monodomain: $\mathrm{u}_t - \Delta u = R(u, y)$
  $y_t = f(y, t)$

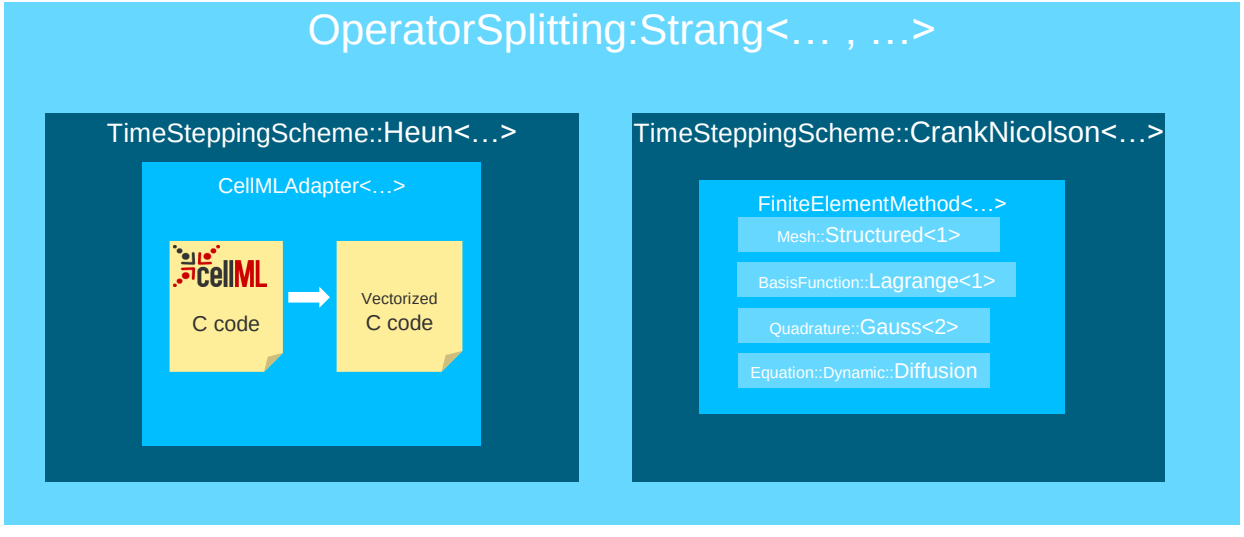




Software framework *opendihu*

Nested solvers

- Monodomain: $$\mathrm{u}_t - \Delta u = R(u, y)$$
$$y_t = f(y, t)$$

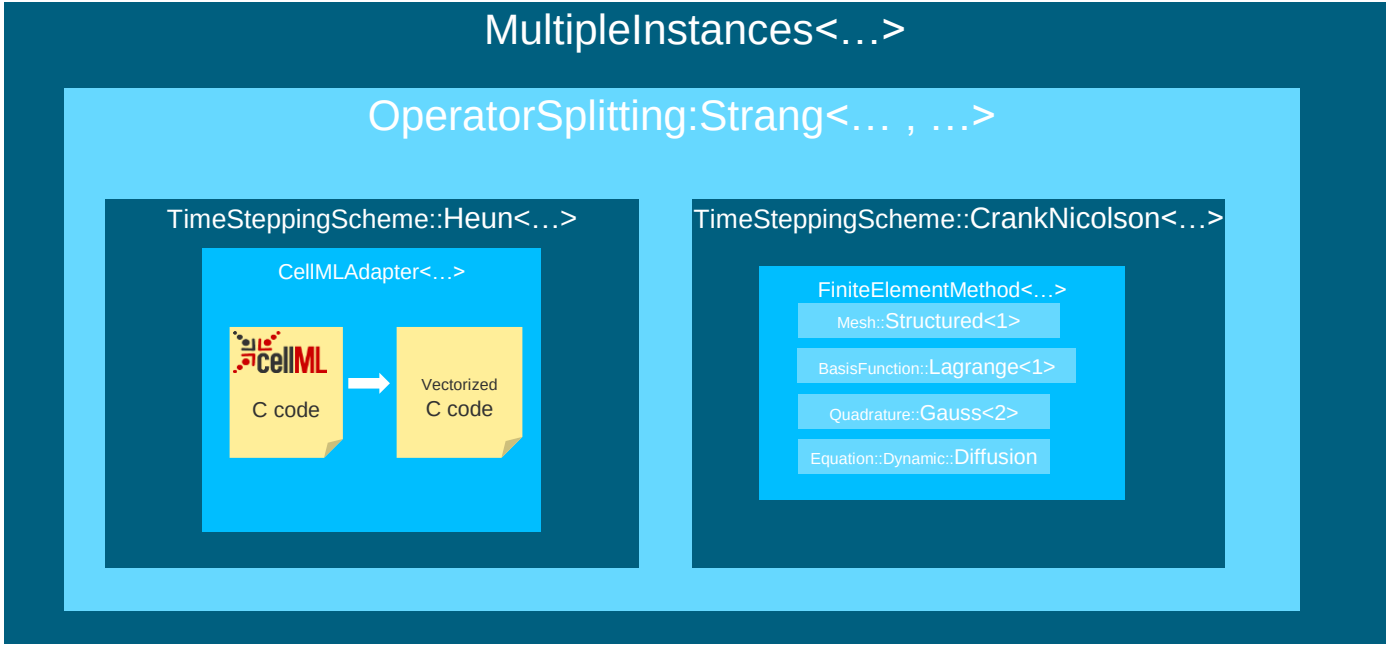


Software framework *opendihu*

Nested solvers

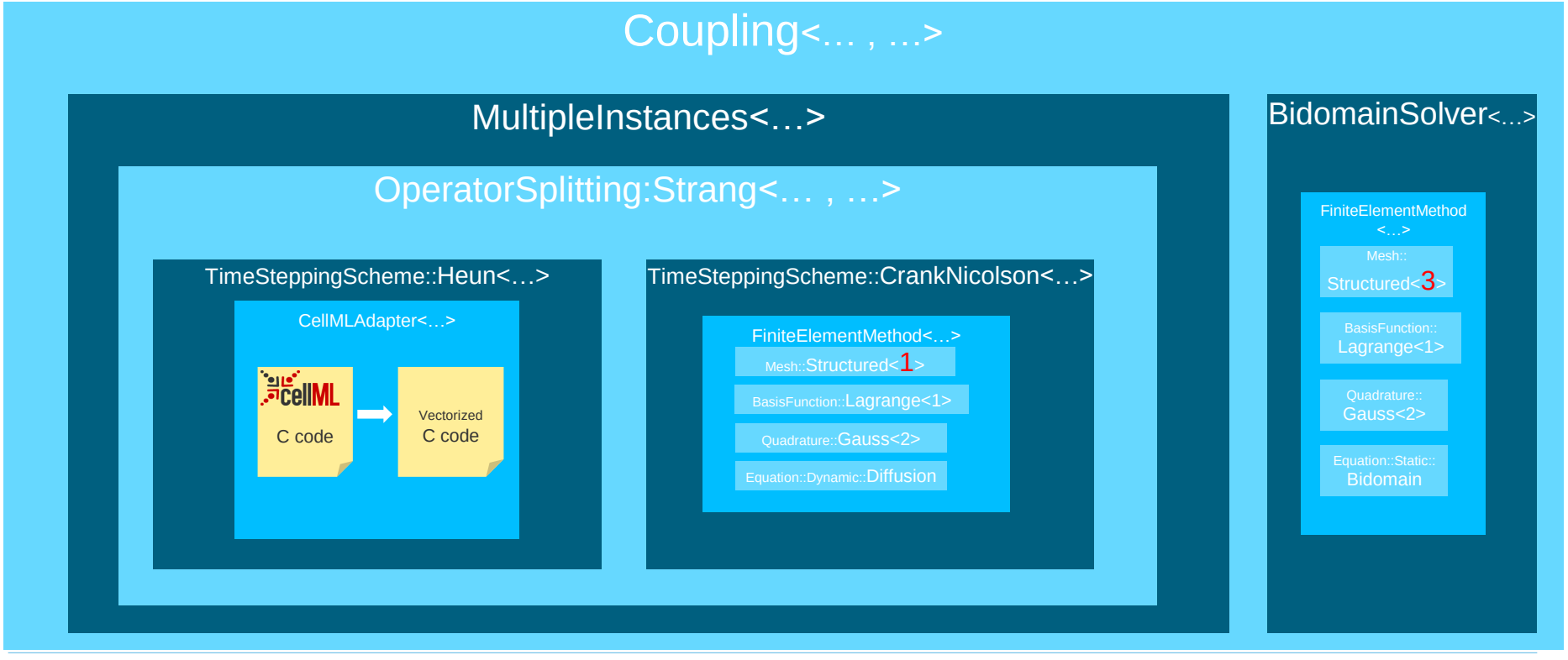
- Monodomain:

$$u_t - \Delta u = R(u, y)$$
$$y_t = f(y, t)$$



Software framework *opendihu*

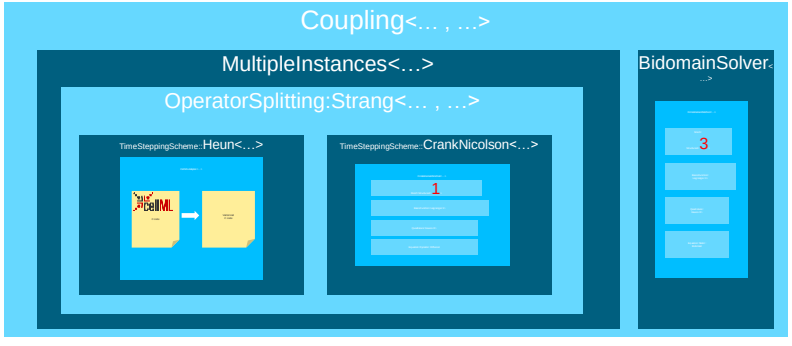
Nested solvers



Software framework *opendihu*

Configuration

- C++ main file:
  ~50 lines



- Compile and link to core library

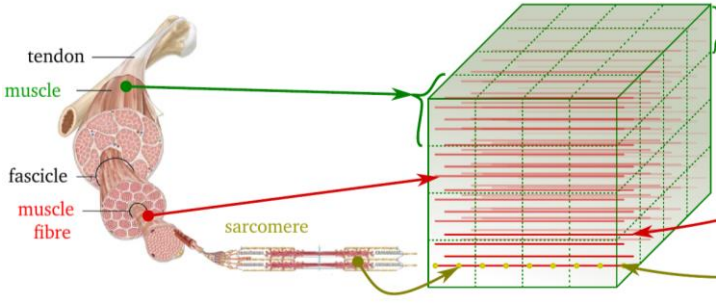
- Python script
  <1k lines

```
"StrangSplitting": {
  "timeStepWidth": dt_splitting,
  "endTime": 10.0,
    "MultipleInstances": {
      "Heun" : {
        "timeStepWidth": dt_0D,
        ...
        "CellML" : {
          "sourceFilename": cellml_file,
```

- Numerical and model parameters
- Mesh, node positions
- Local input for parallel execution
- Preprocessing: using numpy, scipy, etc.
- Command line arguments available

Software framework *opendihu*

Source-to-source compiler



Monodomain equation

$$\frac{\partial V_m}{\partial t} = \underbrace{\frac{\sigma_{\text{eff}}}{A_m C_m} \frac{\partial^2 V_m}{\partial s^2}}_{\text{diffusion term}} - \frac{1}{C_m} I_{\text{ion}}(V_m, \mathbf{y})$$

$$\underbrace{\frac{\partial \mathbf{y}}{\partial t} = G_{\mathbf{y}}(V_m, \mathbf{y})}_{\text{reaction term}}$$

cellML

C code

Original CellML code

- Array-of-struct

```
ALGEBRAIC[66] = pow(1.00000+ 0.120000*exp(

ALGEBRAIC[68] =  CONSTANTS[53]*ALGEBRAIC[6

OC_RATE[3] =  - CONSTANTS[9]*((ALGEBRAIC[5

OC_RATE[2] = (ALGEBRAIC[59]+ALGEBRAIC[61]+
```

Transformed code for multiple instances at once

- Struct-of-array

```
for(int i = 0; i < 501; i++)
{
 ALGEBRAIC[31563+i] =  ( ( CONSTANTS[39]*pow(OC_STATE[7515+i], 3
}

for(int i = 0; i < 501; i++)
{
 ALGEBRAIC[31062+i] = OC_STATE[501+i]*((OC_STATE[2505+i] -  OC_
}
```

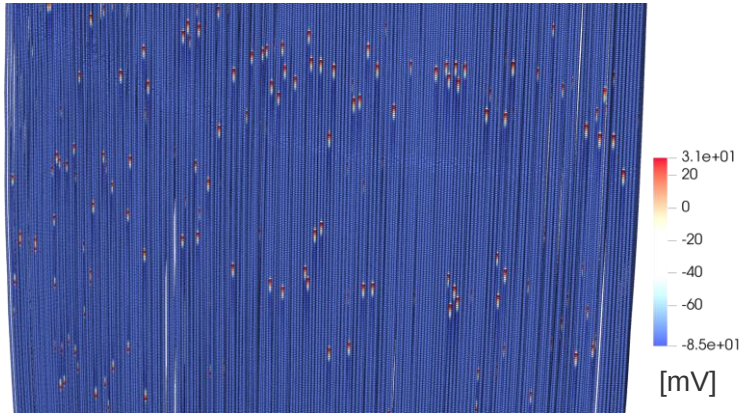
Contents

- Mathematical Description
- Steps Towards a Highly Parallel Simulation
- Obtaining fiber geometry
- Software Framework *opendihu*
- Results

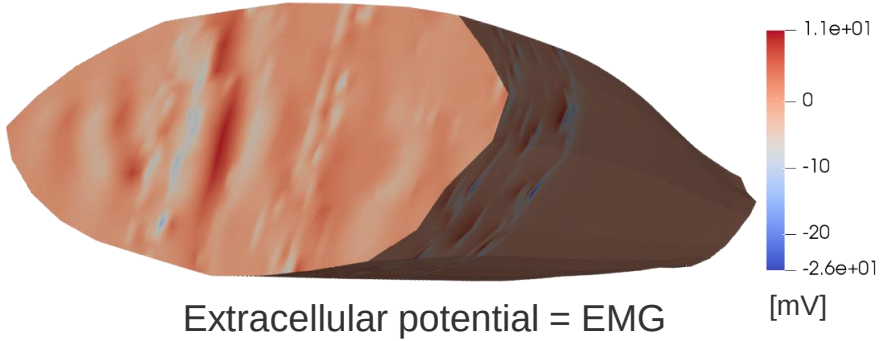
Contents

- Mathematical Description
- Steps Towards a Highly Parallel Simulation
- Obtaining fiber geometry
- Software Framework *opendihu*
- **Results**

Results



Transmembrane voltage

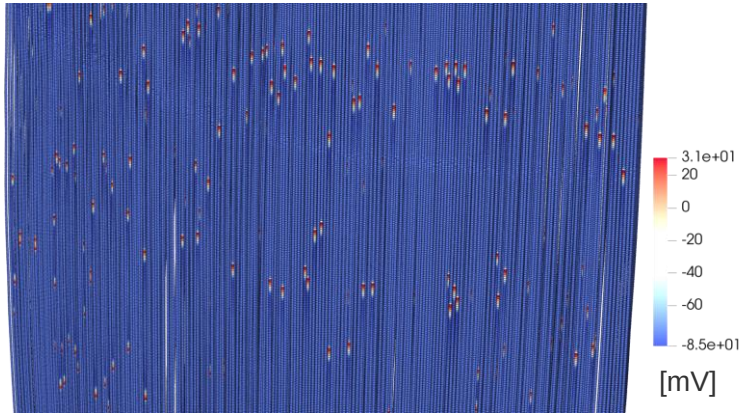


Extracellular potential = EMG

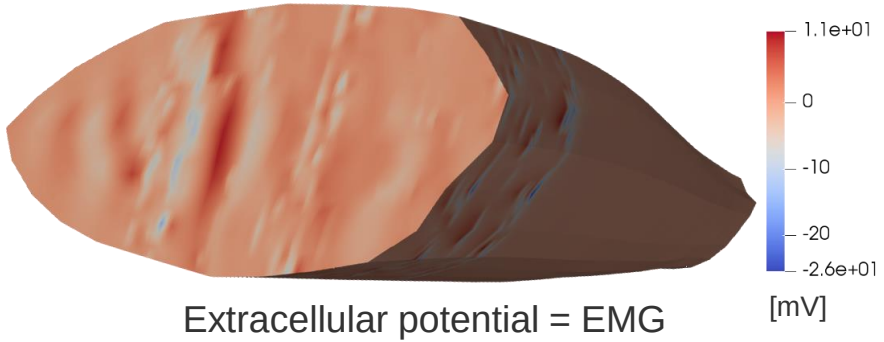
- Typical runtimes:
  - 625 fibers, 1480 FE/fiber, t = 5 ms, 2 processes on my laptop: 21 min
  - 1369 fibers, 1480 FE/fiber, t = 100 ms, 150 processes on a shared memory server: 7.5 h
  - 2700 fibers, 50 FE/fiber, t = 0.3 ms, 1 process, (only 0D+1D models)
    literature [1] with OpenCMISS Iron: 3h

[1] M. Mordhorst, T. Heidlauf, and O. Röhrle. Predicting electromyographic signals under realistic conditions using a multiscale chemo-electro mechanical finite element model. Interface Focus, 5(2):1–11, February 2015.

Results



Transmembrane voltage



Extracellular potential = EMG

- Typical runtimes:
  - 625 fibers, 1480 FE/fiber, t = 5 ms, 2 processes on my laptop: 21 min
  - 1369 fibers, 1480 FE/fiber, t = 100 ms, 150 processes on a shared memory server: 7.5 h
  - 2700 fibers, 50 FE/fiber, t = 0.3 ms, 1 process, (only 0D+1D models)
    literature [1] with OpenCMISS Iron: 3h. *opendihu:* 4min 40s (Speedup 38)

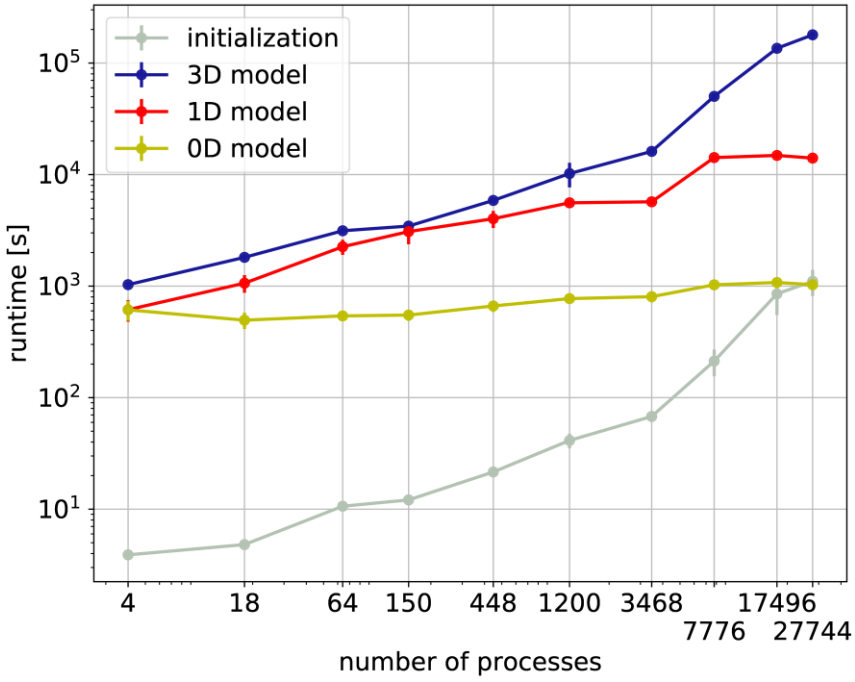
[1] M. Mordhorst, T. Heidlauf, and O. Röhrle. Predicting electromyographic signals under realistic conditions using a multiscale chemo-electro mechanical finite element model. Interface Focus, 5(2):1–11, February 2015.

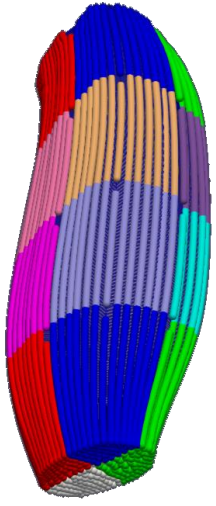
Results

Weak scaling: Increase problem size and number of processes

Monodomain + static Bidomain, t = 1s, HazelHen, 24 proc./node

0D: Heun, dt=1.5e-6s, 1D: Crank-Nicolson, conj. gradients, dt=3e-6s, 3D: GMRES, restart=30, dt=1e-3s



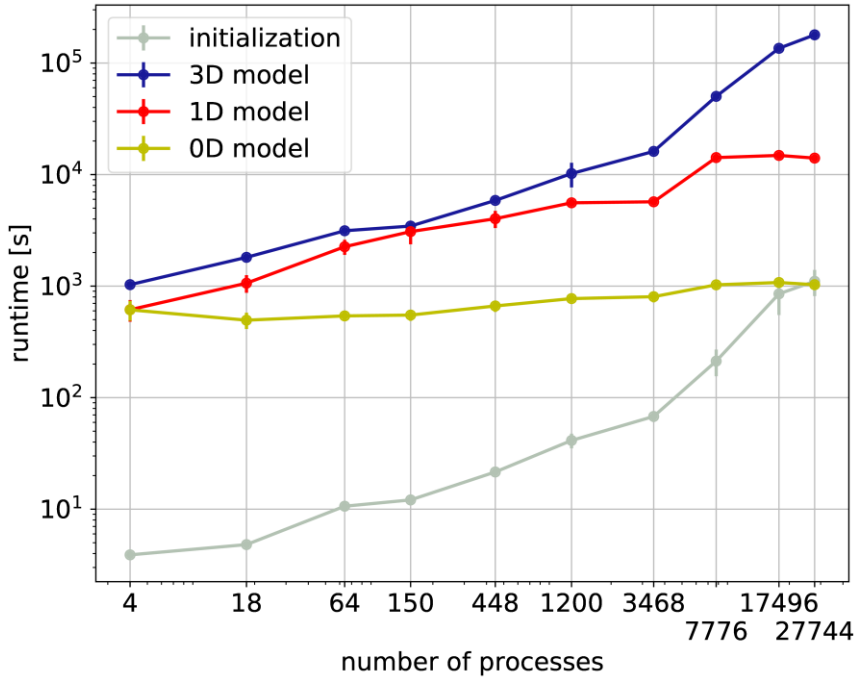


Results

Weak scaling: Increase problem size and number of processes

Monodomain + static Bidomain, t = 1s, HazelHen, 24 proc./node

0D: Heun, dt=1.5e-6s, 1D: Crank-Nicolson, conj. gradients, dt=3e-6s, 3D: GMRES, restart=30, dt=1e-3s



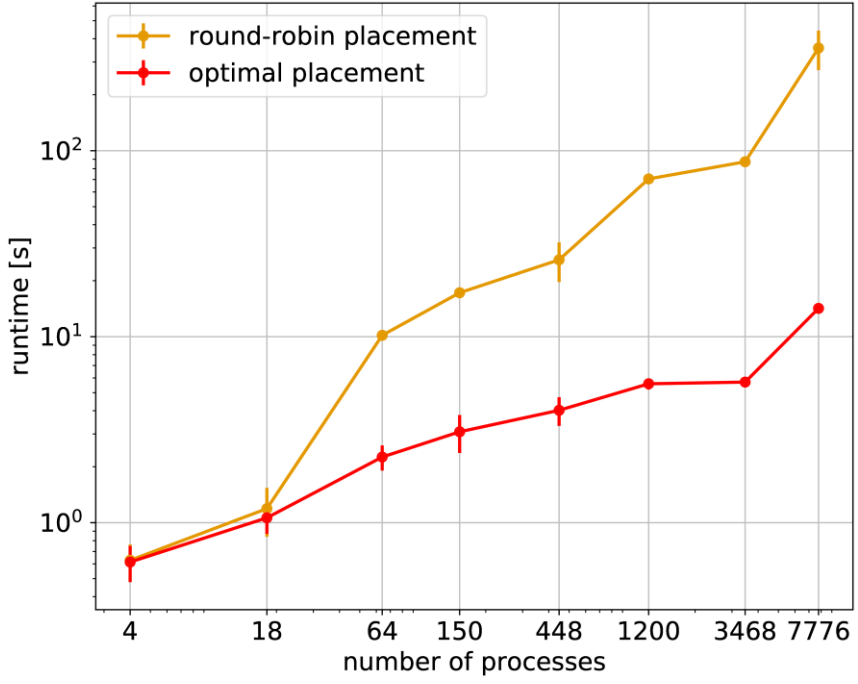
partitioning $x \times y \times z$	fibers	fibers per process	total degrees of freedom
$2 \times 2 \times 1 = 4$	49	12.3	297,984
$3 \times 3 \times 2 = 18$	169	9.4	1,032,160
$4 \times 4 \times 4 = 64$	625	9.8	3,797,216
$5 \times 5 \times 6 = 150$	1369	9.1	8,303,280
$7 \times 8 \times 8 = 448$	4489	10.0	27,201,100
$10 \times 10 \times 12 = 1200$	11,881	9.9	72,124,720
$17 \times 17 \times 12 = 3468$	34,969	10.1	212,187,268
$18 \times 18 \times 24 = 7776$	76,729	9.9	464,686,624
$27 \times 27 \times 24 = 17,496$	182,329	10.4	1,102,902,304
$34 \times 34 \times 24 = 27,744$	273,529	9.9	1,656,579,616

Results

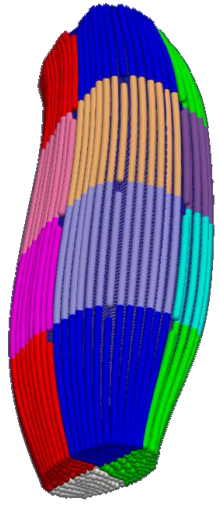
Weak scaling: Increase problem size and number of processes

Monodomain + static Bidomain, t = 1s, HazelHen, 24 proc./node

0D: Heun, dt=1.5e-6s, 1D: Crank-Nicolson, conj. gradients, dt=3e-6s, 3D: GMRES, restart=30, dt=1e-3s



partitioning $x \times y \times z$	fibers
$2 \times 2 \times 1 = 4$	49
$3 \times 3 \times 2 = 18$	169
$4 \times 4 \times 4 = 64$	625
$5 \times 5 \times 6 = 150$	1369
$7 \times 8 \times 8 = 448$	4489
$10 \times 10 \times 12 = 1200$	11,881
$17 \times 17 \times 12 = 3468$	34,969
$18 \times 18 \times 24 = 7776$	76,729
$27 \times 27 \times 24 = 17{,}496$	182,329
$34 \times 34 \times 24 = 27{,}744$	273,529

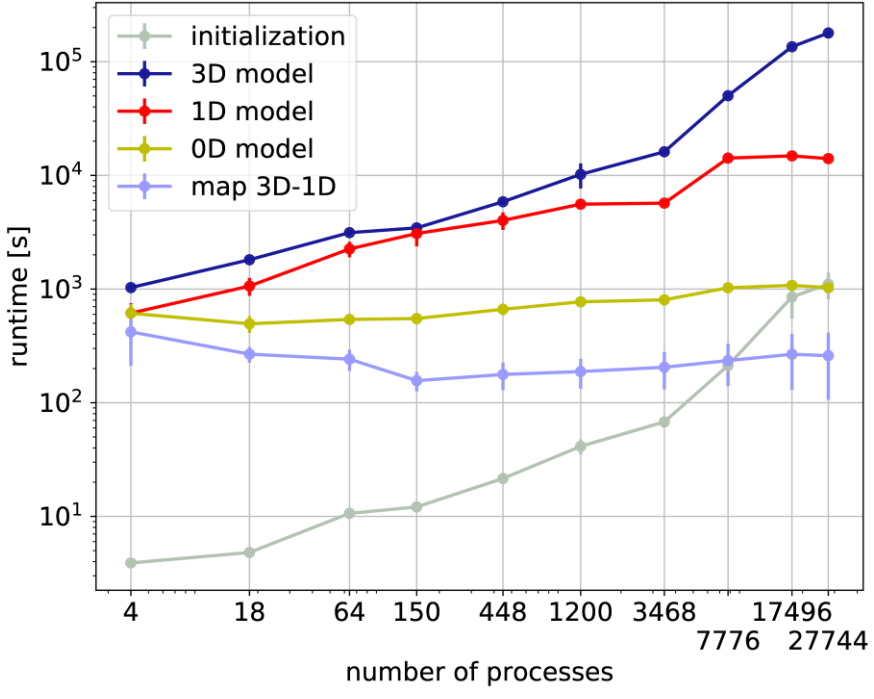


Results

Weak scaling: Increase problem size and number of processes

Monodomain + static Bidomain, t = 1s, HazelHen, 24 proc./node

0D: Heun, dt=1.5e-6s, 1D: Crank-Nicolson, conj. gradients, dt=3e-6s, 3D: GMRES, restart=30, dt=1e-3s



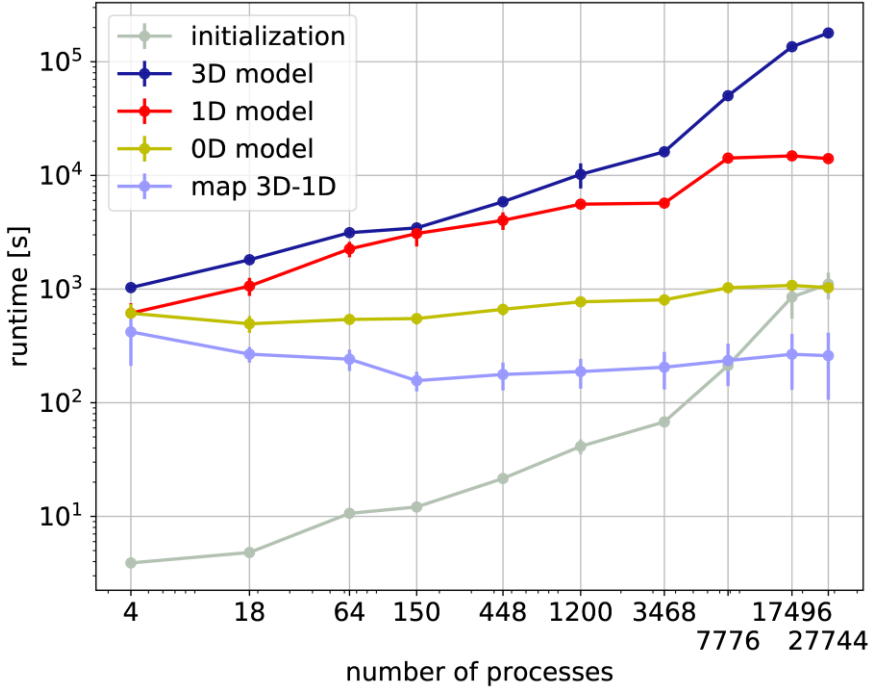
partitioning $x \times y \times z$	fibers	fibers per process	total degrees of freedom
$2 \times 2 \times 1 = 4$	49	12.3	297,984
$3 \times 3 \times 2 = 18$	169	9.4	1,032,160
$4 \times 4 \times 4 = 64$	625	9.8	3,797,216
$5 \times 5 \times 6 = 150$	1369	9.1	8,303,280
$7 \times 8 \times 8 = 448$	4489	10.0	27,201,100
$10 \times 10 \times 12 = 1200$	11,881	9.9	72,124,720
$17 \times 17 \times 12 = 3468$	34,969	10.1	212,187,268
$18 \times 18 \times 24 = 7776$	76,729	9.9	464,686,624
$27 \times 27 \times 24 = 17,496$	182,329	10.4	1,102,902,304
$34 \times 34 \times 24 = 27,744$	273,529	9.9	1,656,579,616

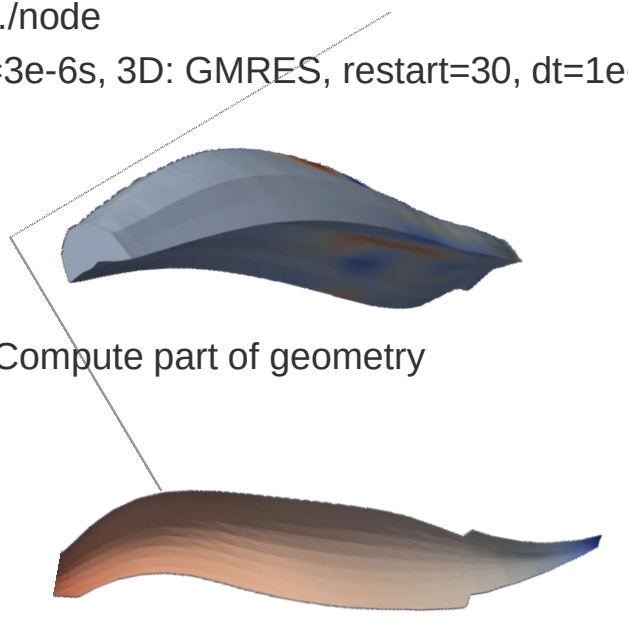
Results

Weak scaling: Increase problem size and number of processes

Monodomain + static bidomain, t = 1s, HazelHen, 24 proc./node

0D: Heun, dt=1.5e-6s, 1D: Crank-Nicolson, conj. gradients, dt=3e-6s, 3D: GMRES, restart=30, dt=1e-3s





- Compute part of geometry
- Compute full geometry, only output surface
- Future work: in-situ visualization

Conclusion and Outlook

Conclusion

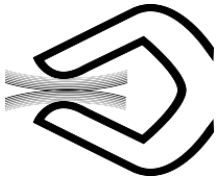
- High Performance Computing suitable for biophysical simuation of muscular activation with high number of fibers
- Structured grids have performance advantages
- *opendihu* is an efficient and flexible software framework for this task

Future work

- Contraction (linear or finite elasticity)
- Comparison of discretization parameters and models

Acknowledgements

Deutsche
Forschungsgemeinschaft



IRTG
Soft Tissue Robotics

Simulation of Large Systems
Institute for Parallel and Distributed Systems

University of Stuttgart
Universitätsstr. 38
70569 Stuttgart

Thank you!

Benjamin Maier

e-mail Benjamin.maier@ipvs.uni-stuttgart.de
phone +49 (0) 711 685-88247
fax +49 (0) 711 685-78247

University of Stuttgart
Institute for Parallel and Distributed Systems
Universitätsstraße 38
D-70569 Stuttgart