Efficient and reusable Software for Coupled Problems
Multi-Physics / Multi-Domain Problems

- Flooding
- Immiscible Two-Phase Flow
- Stokes Darcy Coupling
- Cell Biology
Multi-Physics / Multi-Domain Problems

- Flooding
  - Strong coupling of ground and surface water
  - Richards Equation and Shallow-Water Eq.
  - Coupling of 3D and 2D problems
- Immiscible Two-Phase Flow
- Stokes Darcy Coupling
- Cell Biology

Figure B-3: Groundwater System Involving the Hyporheic Zones (Alley et. al 2002)
Coupled surface and subsurface water
Multi-Physics / Multi-Domain Problems

- Flooding
- Immiscible Two-Phase Flow
  - Two immiscible fluid
  - Navier-Stokes Equation
  - Surface tension at the mutual interface
  - Buoyancy rise of a bubble
- Stokes Darcy Coupling
- Cell Biology

Rising bubble, Andrew Davidhazy (RIT)
Multi-Physics / Multi-Domain Problems

- Flooding
- Immiscible Two-Phase Flow
- Stokes Darcy Coupling
  - Free fluid flow (Navier-Stokes)
  - Porous media flow (Darcy)
  - Beaver-Joseph-Saffman condition
- Cell Biology
Multi-Physics / Multi-Domain Problems

- Flooding
- Immiscible Two-Phase Flow
- Stokes Darcy Coupling
- Cell Biology
  - Coupling of Membrane and Bulk processes
  - Reaction-Diffusion(-Convection) Systems
  - Coupling back to the geometry

Yeast budding
Outline

Introduction

Approaches for Multi-Domain Simulations

Reusable Software
   The DUNE Framework
   Rapid Prototyping via DUNE PDELab

Multi-Domain Software
   Multi-Domain interfaces in DUNE
   Multi-Domain concepts for PDELab

Applications
   Application 1: Stokes-Darcy-Coupling
   Application 2: Modelling a yeast cell’s budding process

Conclusions
Possible Multi-Domain Settings

Subdomains can be

- overlapping
- non-overlapping
- mixed-dimensional
Possible Multi-Domain Settings

Subdomains can be

- overlapping
- non-overlapping
- mixed-dimensional

All possible settings should be supported
Abstract Problem Definition

- Let $\Omega$ be a subdomain of $\mathbb{R}^d$ and $G$ a partition of $\Omega$ into subdomains
  
  $G(\Omega) = \{ \Omega^{(0)}, \ldots, \Omega^{(N-1)} \}$.

  The boundaries $\partial \Omega^{(i)}$ may have a complicated shape.

- Want to solve PDEs on $\Omega^{(i)}$
  
  $L_i(u_i) = f_i \quad + \quad \text{BC}$

- and PDEs on $\Gamma^{(i,j)}$
  
  $L_{i,j}(u_{i,j}) = f_{i,j}$

- and coupling conditions along $\Gamma^{(i,j)}$. 

Partition $G$ of $\Omega$ into two sub-domains with the interface $\Gamma^{(0,1)}$. 
Multi-Domain descriptions

Requirements
Given $\Omega$ we need a set $\{\mathcal{T}(\Omega^{(i)})\}$ of triangulations for the different subdomains.

- Individual meshes:
- Partition of a mesh:
- Cut-Cell meshes:
Multi-Domain descriptions

Requirements
Given Ω we need a set \( \{ \mathcal{T}(\Omega^{(i)}) \} \) of triangulations for the different subdomains.

- Individual meshes:
  - Given Ω we define a set of subdomains \( \mathcal{G} = \{ \Omega^{(i)} \} \).
  - For each \( \Omega^{(i)} \) we define an individual triangulation \( \mathcal{T}(\Omega^{(i)}) \).

- Partition of a mesh:

- Cut-Cell meshes:
Multi-Domain descriptions

Requirements
Given $\Omega$ we need a set $\{\mathcal{T}(\Omega^{(i)})\}$ of triangulations for the different subdomains.

- Individual meshes:
  $\Omega \rightarrow \{\Omega^{(i)}\} \rightarrow \{\mathcal{T}(\Omega^{(i)})\}$

- Partition of a mesh:
  - Given $\Omega$ we construct a triangulation $\mathcal{T}(\Omega)$.
  - The skeleton $\Gamma(\Omega) = \bigcap \partial E, E \in \mathcal{T}$ is aligned with subdomain boundaries $\partial \Omega^{(i)}$.
  - Triangulation of subdomains are constructed from $\mathcal{T}$ as $\mathcal{T}(\Omega) = \{E \in \mathcal{T} | E \subset \Omega^{(i)}\}$

- Cut-Cell meshes:
Multi-Domain descriptions

Requirements
Given $\Omega$ we need a set $\{ \mathcal{T}(\Omega^{(i)}) \}$ of triangulations for the different subdomains.

- **Individual meshes:**
  \[
  \Omega \rightarrow \{\Omega^{(i)}\} \rightarrow \{\mathcal{T}(\Omega^{(i)})\}
  \]

- **Partition of a mesh:**
  \[
  \Omega \rightarrow \mathcal{T}(\Omega), \bigcup \partial \Omega^{(i)} \subset \Gamma \rightarrow \{\mathcal{T}(\Omega^{(i)}) = \{E \in \mathcal{T} | E \subset \Omega^{(i)}\}\}
  \]

- **Cut-Cell meshes:**
  \[
  \text{Given } \Omega \text{ we construct a triangulation } \mathcal{T}(\Omega), \text{ without taking the subdomains into account.}
  \]
  \[
  \text{Triangulation of subdomains are constructed from } \mathcal{T} \text{ as the intersection } \mathcal{T}(\Omega) = \{E \cap \Omega^{(i)} | E \in \mathcal{T}\}
  \]
Multi-Domain descriptions

Requirements
Given $\Omega$ we need a set $\{\mathcal{T}(\Omega^{(i)})\}$ of triangulations for the different subdomains.

- **Individual meshes:**
  \[ \Omega \rightarrow \{\Omega^{(i)}\} \rightarrow \{\mathcal{T}(\Omega^{(i)})\} \]

- **Partition of a mesh:**
  \[ \Omega \rightarrow \mathcal{T}(\Omega), \bigcup \partial \Omega^{(i)} \subset \Gamma \rightarrow \{\mathcal{T}(\Omega^{(i)}) = \{E \in \mathcal{T} | E \subset \Omega^{(i)}\}\} \]

- **Cut-Cell meshes:**
  \[ \Omega \rightarrow \mathcal{T}(\Omega) \rightarrow \{\mathcal{T}(\Omega^{(i)}) = \{E \cap \Omega^{(i)} | E \in \mathcal{T}\}\} \]

Each of these concepts is available in DUNE
Introduction

Approaches for Multi-Domain Simulations

Reusable Software

Multi-Domain Software

Applications

Conclusions
Reusable Software

People think that computer science is the art of geniuses but the actual reality is the opposite, just many people doing things that build on each other, like a wall of mini stones.

— Donald E. Knuth

Flexibility  Seperation of data structures and algorithms.

Efficiency  Generic programming techniques.

Legacy Code  Reuse existing finite element software.
Flexibility + Efficiency

Separate data structures and algorithms.

Generic Programming techniques

…allows to write algorithms independent from the underlying data types and data structures.

- Static Polymorphism (C++ STL, boost, MTL, …)
- Iterators
- View Concept
The DUNE Framework

Current stable version is 2.2.0, available since June 4th 2012.

dune-common: foundation classes, infrastructure

dune-geometry: geometric mapping, quadrature rules

dune-grid: grid interface, visualization

dune-istl: (Iterative Solver Template Library)
generic sparse matrix/vector classes,
solvers (Krylov methods, AMG, etc.)

dune-localfunctions: generic interface for local finite element functions.
Abstract definition following Ciarlet.

http://www.dune-project.org/
The DUNE Framework

applications

discretization modules
  pdelab
  fem

extra grids

external modules

core modules
  grid
  istl
  localfunctions
Rapid Prototyping via DUNE PDELab

Aims and Features:

▶ Rapid prototyping: Substantially reduce time to implement discretizations and solvers for systems of PDEs based on DUNE.
▶ Simple things should be simple – suitable for teaching.
▶ Discrete function spaces spaces:
  ▶ Conforming and non-conforming,
  ▶ hp-refinement,
  ▶ general approach to constraints,
  ▶ generic generation of product spaces for systems.
▶ Operators based on weighted residual formulation:
  ▶ Linear and nonlinear,
  ▶ stationary and transient,
  ▶ FE and FV schemes requiring at most face-neighbors.
▶ Exchangeable linear algebra backend.
▶ User only involved with “local” view on (reference) element.
Rapid Prototyping via DUNE PDELab

Aims and Features:

▶ Rapid prototyping: Substantially reduce time to implement discretizations and solvers for systems of PDEs based on DUNE.
▶ Simple things should be simple – suitable for teaching.
▶ Discrete function spaces spaces:
  ▶ Conforming and non-conforming,
  ▶ hp-refinement,
  ▶ general approach to constraints,
  ▶ generic generation of product spaces for systems.
▶ Operators based on weighted residual formulation:
  ▶ Linear and nonlinear,
  ▶ stationary and transient,
  ▶ FE and FV schemes requiring at most face-neighbors.
▶ Exchangeable linear algebra backend.
▶ User only involved with “local” view on (reference) element.
Overview: Interface in PDELab

Efficient and reusable Software for Coupled Problems

Christian Engwer
Residual Formulation

We consider an unconstrained problem in weighted residual form:

\[ u_h \in U_h : \quad r_h(u_h, v) = 0 \quad \forall v \in V_h. \]
Residual Formulation

We consider an unconstrained problem in weighted residual form:

\[ u_h \in U_h : \quad r_h(u_h, v) = 0 \quad \forall v \in V_h. \]

For a particular test function space, we obtain a nonlinear algebraic problem:

\[ u \in U : \quad r_h \left( \text{FE}_{\Phi_{u_h}}(u), \psi_i \right) = 0, \quad i \in I_{V_h} \]

\[ \iff R(u) = 0 \]

with the nonlinear residual map \( R : U = K^{I_{V_h}} \to K^{I_{V_h}} \).
Evaluation of Residual Map

Using splitting and localization properties we obtain

\[
\begin{align*}
    r(u_h, v_h) &= \sum_{e \in E^0_h} \int_e A^{\text{vol}}(u_h, v_h) \\
    &+ \sum_{f \in E^1_h} \int_f A^{\text{skel}}(u_h, v_h) \\
    &+ \sum_{b \in B^1_h} \int_b A^{\text{bnd}}(u_h, v_h) \\
    &+ \sum_{e \in E^0_h} \int_e L^{\text{vol}}(v_h) \\
    &+ \sum_{f \in E^1_h} \int_f L^{\text{skel}}(v_h) \\
    &+ \sum_{b \in B^1_h} \int_b L^{\text{bnd}}(v_h).
\end{align*}
\]
Evaluation of Residual Map

Using splitting and localization properties we obtain

\[ \mathcal{R}(u) = \sum_{e \in E^0_h} \alpha^\text{vol}_{h,e}(u|_e) + \sum_{e \in E^0_h} \lambda^\text{vol}_{h,e} \]
\[ + \sum_{f \in E^1_h} \alpha^\text{skel}_{h,f}(u|_{l(f)}, u|_{r(f)}) + \sum_{f \in E^1_h} \lambda^\text{skel}_{h,f} \]
\[ + \sum_{b \in B^1_h} \alpha^\text{bnd}_{h,b}(u|_b) + \sum_{b \in B^1_h} \lambda^\text{bnd}_{h,b}. \]

To implement a discretization one has to implement a class providing methods for evaluation of \( \alpha^\text{vol}_{h,e}, \alpha^\text{skel}_{h,f}, \alpha^\text{bnd}_{h,b}, \lambda^\text{vol}_{h,e}, \lambda^\text{skel}_{h,f} \) and \( \lambda^\text{bnd}_{h,b} \).
Local Operators

Implement methods to evaluate $\alpha_{vol, h,e}$, $\alpha_{skel, h,f}$, $\alpha_{bnd, h,b}$, $\lambda_{vol, h,e}$, $\lambda_{skel, h,f}$ and $\lambda_{bnd, h,b}$. As well as the jacobian.

**Interface:**

```cpp
class LocalOperatorInterface {
    // $\alpha_{vol, h,e}$
    template<typename E, typename LFSU, typename X, typename LFSV, typename R>
    void alpha_volume (const E& e, const LFSU& lfsu, const X& x, const LFSV& lfsv, R& r);

    // $\alpha_{skel, h,f}$
    template<typename I, typename LFSU, typename X, typename LFSV, typename R>
    void alpha_skeleton (const I& f, const LFSU& lfsu_s, const X& x_s, const LFSV& lfsv_s,
                         const LFSU& lfsu_n, const X& x_n, const LFSV& lfsv_n, R& r_s, R& r_n);

    // $\alpha_{bnd, h,f}$
    template<typename I, typename LFSU, typename X, typename LFSV, typename R>
    void alpha_boundary (const IG& f, const LFSU& lfsu_s, const X& x_s, const LFSV& lfsv_s, R& r_s);

    ...
};
```

Christian Engwer
### Local Operators – Lines of Code

<table>
<thead>
<tr>
<th>Problem</th>
<th>Discretization</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Navier-Stokes</td>
<td>TH-FEM, DG</td>
<td>170, 867</td>
</tr>
<tr>
<td>Convection-Diffusion</td>
<td>CCFV, FEM, DG</td>
<td>247, 191, 710</td>
</tr>
<tr>
<td>Diffusion</td>
<td>CCFV</td>
<td>122</td>
</tr>
<tr>
<td></td>
<td>FEM</td>
<td>164</td>
</tr>
<tr>
<td></td>
<td>Mixed-FEM, DG</td>
<td>163, 616</td>
</tr>
<tr>
<td></td>
<td>MFD</td>
<td>129</td>
</tr>
<tr>
<td>Maxwell</td>
<td>Edge-FEM, DG</td>
<td>150, 118</td>
</tr>
<tr>
<td>Two-Phase Flow</td>
<td>CCFV</td>
<td>270</td>
</tr>
</tbody>
</table>
Introduction

Approaches for Multi-Domain Simulations

Reusable Software

Multi-Domain Software

Applications

Conclusions
Multi-Domain interfaces in DUNE

DUNE offers different approaches for multi-domain / multi-physics simulations

- **grid-glue**
  - Different Domains represented as different DUNE Grids.
  - Relates two unrelated grids to each other
  - Abstract interface to geometry of grid couplings
  - *Joint work with O. Sander, RWTH Aachen*

- **multi-domain-grid**
  - Different Domains as sub-sets of one DUNE Grid
  - Domain interfaces resolved by internal faces
  - **S. Müthing**

- **udg**
  - Domain interface given as a level-set function
  - Allows moving interfaces
  - Results in Cut-Cell meshes
  - Discretization using DG
  - *Joint work with F. Heimann, Uni Heidelberg*
Multi-Domain interfaces in DUNE

DUNE offers different approaches for multi-domain / multi-physics simulations

- **grid-glue**
  - Different Domains represented as different DUNE Grids.
  - Relates two unrelated grids to each other
  - Abstract interface to geometry of grid couplings
  - *Joint work with O. Sander, RWTH Aachen*

- **multi-domain-grid**
  - Different Domains as sub-sets of one DUNE Grid
  - Domain interfaces resolved by internal faces
  - S. Müthing

- **udg**
  - Domain interface given as a level-set function
  - Allows moving interfaces
  - Results in Cut-Cell meshes
  - Discretization using DG
  - *Joint work with F. Heimann, Uni Heidelberg*
Multi-Domain interfaces in DUNE

DUNE offers different approaches for multi-domain / multi-physics simulations

- **grid-glue**
  - Different Domains represented as different DUNE Grids.
  - Relates two unrelated grids to each other
  - Abstract interface to geometry of grid couplings
  - *Joint work with O. Sander, RWTH Aachen*

- **multi-domain-grid**
  - Different Domains as sub-sets of one DUNE Grid
  - Domain interfaces resolved by internal faces
  - *S. Mützing*

- **udg**
  - Domain interface given as a level-set function
  - Allows moving interfaces
  - Results in Cut-Cell meshes
  - Discretization using DG
  - *Joint work with F. Heimann, Uni Heidelberg*
Multi-Domain interfaces in DUNE

DUNE offers different approaches for multi-domain / multi-physics simulations

▶ grid-glue
  ▶ Different Domains represented as different DUNE Grids.
  ▶ Relates two unrelated grids to each other
  ▶ Abstract interface to geometry of grid couplings
  ▶ Joint work with O. Sander, RWTH Aachen

▶ multi-domain-grid
  ▶ Different Domains as sub-sets of one DUNE Grid
  ▶ Domain interfaces resolved by internal faces
  ▶ S. Mütting

▶ udg
  ▶ Domain interface given as a level-set function
  ▶ Allows moving interfaces
  ▶ Results in Cut-Cell meshes
  ▶ Discretization using DG
  ▶ Joint work with F. Heimann, Uni Heidelberg
dune-grid-glue

Relating unrelated grids

- Conforming, semiconforming, nonconforming

- Nonmatching geometries

- Overlapping geometries

- Mixed dimensions
dune-grid-glue

Relating unrelated grids

Example: nonoverlapping conforming coupling
dune-multidomain-grid

Sub-sets of an existing dune-grid

- MultiDomainGrid
  - Host Grid
  - Subdomain Partitioning Data
- SubDomainGrid 1
- SubDomainGrid 2
- SubDomainGrid 3
dune-udg

Level-set based cut-cells

Sub-domain boundary $\Gamma^{(i,j)}$, given as the iso-surface of a scalar function, e.g.

- micro-ct images
- level-set
- phase-field
Local triangulation of Cut-Cell grids based on modified Marching-Cubes algorithm.
Coupling via Intersections

Generalized Intersections:

\[ I = E \cap F, \quad E \in \mathcal{T}(\Omega^{(1)}), F \in \mathcal{T}(\Omega^{(2)}) \]

- Conform to DUNE Intersection interface as much as possible:
  - corresponding entities
  - geometric mapping into the world space / intersecting elements
  - normal vectors
- Intersections may be triangulated
- Access via iterators
- Unique direction (e.g. pointing from \( \Omega_1 \) to \( \Omega_2 \))
Coupling via Intersections

Generalized Intersections:

\[ I = E \cap F, \quad E \in \mathcal{T}(\Omega^{(1)}), \ F \in \mathcal{T}(\Omega^{(2)}) \]
Multi-Domain concepts for PDELab

Efficient and reusable Software for Coupled Problems
Multi-Domain concepts for PDELab

Efficient and reusable Software for Coupled Problems

Christian Engwer
Multi-Domain concepts for PDELab

Work in progress
Global Grid Function Spaces

Example Stokes Darcy

- We consider a global function space GFS(\(\Omega\))
- Coupling leads to a global System with . . .
  - local systems \(A_x\) for each subdomain
  - coupling operators \(C, D\)

\[
\begin{pmatrix}
  A_S & C \\
  D & A_D
\end{pmatrix}
= 
\begin{pmatrix}
  f_S \\
  f_D
\end{pmatrix}
\]
General Coupling Operators

- Following the PDELab concept of local operators
- Coupling condition contributes to global residual
- Contributions similar to skeleton in PDELab
- Generic code to reuse the same operator with different frameworks

\[
\text{coupling}(u_1, u_2) = \sum_{f \in \Gamma(1, 2)} \alpha_{\text{coupling} h, f}(u_1|_l(f), u_2|_r(f)) + \sum_{f \in \Gamma(1, 2)} \lambda_{\text{coupling} h, f}
\]

To implement the coupling one has to implement a class providing methods for evaluation of \(\alpha_{\text{coupling} h, f}\) and \(\lambda_{\text{coupling} h, f}\).

User only involved with "local" view on the intersection.
General Coupling Operators

▶ Following the PDELab concept of local operators
▶ Coupling condition contributes to global residual
▶ Contributions similar to skeleton in PDELab
▶ Generic code to reuse the same operator with different frameworks

\[
\mathcal{R}_{\text{coupling}}(\mathbf{u}_1, \mathbf{u}_2) = \sum_{f \in \Gamma^{(1,2)}} \alpha_{h,f}^{\text{coupling}} (\mathbf{u}_1|_{l(f)}, \mathbf{u}_2|_{r(f)}) + \sum_{f \in \Gamma^{(1,2)}} \lambda_{h,f}^{\text{coupling}}
\]

To implement the coupling one has to implement a class providing methods for evaluation of \(\alpha_{h,f}^{\text{coupling}}\) and \(\lambda_{h,f}^{\text{coupling}}\).

▶ User only involved with “local” view on the intersection.
General Coupling Operators

- Following the PDELab concept of local operators
- Coupling condition contributes to global residual
- Contributions similar to skeleton in PDELab
- Generic code to reuse the same operator with different frameworks

\[
\mathcal{R}_{\text{coupling}}(\mathbf{u}_1, \mathbf{u}_2) = \sum_{f \in \Gamma^{(1,2)}} \alpha_{h,f} (\mathbf{u}_1|_{l(f)}, \mathbf{u}_2|_{r(f)}) + \sum_{f \in \Gamma^{(1,2)}} \lambda_{h,f}
\]

To implement the coupling one has to implement a class providing methods for evaluation of \(\alpha_{h,f}\) and \(\lambda_{h,f}\).

- User only involved with “local” view on the intersection.
Introduction

Approaches for Multi-Domain Simulations

Reusable Software

Multi-Domain Software

Applications

Conclusions
Stoke-Darcy Coupling

- Darcy’s law describes flow in porous media
- Stokes describes free fluid flow at low reynolds numbers
- Coupling is described by the Beaver-Joseph-Saffman condition
  - experimentally obtained by [Beaver and Joseph, 1967]
  - refined by [Saffman 1971]
  - rigorous derived in [Jäger and Mikelić, 2000]
Mathematical Model

Stokes:

\[-\nabla \cdot (2\mu D(u) - p_1 I) + u \cdot \nabla u = f_1 \quad \text{in} \quad \Omega_1\]

\[\nabla \cdot u = 0 \quad \text{in} \quad \Omega_1\]

with \(D(u) = \frac{1}{2}(\nabla u + (\nabla u)^T)\), constant \(\mu\), source \(f_1\).

Darcy:

\[-\nabla K \cdot \nabla p = f_2 \quad \text{in} \quad \Omega_2\]

with permeability tensor \(K\) and source \(f_2\).

Plus suitable boundary conditions.

Coupling:

\[u \cdot n = -K \nabla p_2 \cdot n\]

\[((-2\mu D(u) + p_1 I)n) \cdot n = p_2\]

\[u \cdot \tau = -2\mu G(D(u)n) \cdot \tau\]

with material property \(G > 0\), normal \(n\) and tangential \(\tau\).
Settings

Three different Discretization
same coupling operator:

```cpp
typedef StokesDarcyCouplingOperator<CouplingParams> CouplingOP;
CouplingOP couplingop(couplingParams);
```

- Taylor-Hood Stokes
- SIPG Darcy
- Multidomain-Grid
- P2-P1 DG Stokes
- SIPG Darcy
- Multidomain-Grid
- P2-P1 UDG Stokes
- SIPG UDG Darcy
- Cut-Cell-Grid
Three different Discretization
same coupling operator:

```cpp
typedef StokesDarcyCouplingOperator< CouplingParams > CouplingOP;
CouplingOP couplingop (couplingParams);
```

- Taylor-Hood Stokes
- SIPG Darcy
- Multidomain-Grid

- P2-P1 DG Stokes
- SIPG Darcy
- Multidomain-Grid

- P2-P1 UDG Stokes
- SIPG UDG Darcy
- Cut-Cell-Grid
Settings

Three different Discretization
same coupling operator:

```cpp
typedef StokesDarcyCouplingOperator<CouplingParams> CouplingOP;
CouplingOP couplingop(couplingParams);
```

- Taylor-Hood Stokes
- SIPG Darcy
- Multidomain-Grid

- P2-P1 DG Stokes
- SIPG Darcy
- Multidomain-Grid

- P2-P1 UDG Stokes
- SIPG UDG Darcy
- Cut-Cell-Grid

Christian Engwer
Modelling budding yeast

- The eukaryotic organism *Saccharomyces cerevisiae* is known as budding, baker’s or brewer’s yeast
- It reproduces by mitotic cell division, in particular by a division process known as budding
Modelling budding yeast

- The eukaryotic organism *Saccharomyces cerevisiae* is known as budding, baker’s or brewer’s yeast
- It reproduces by mitotic cell division, in particular by a division process known as budding
- Budding is triggered by an intracellular pathway including processes on the cell membrane and in the cell body (cytosol)
- Important part: Formation of a Cdc42 cluster on the membrane
Modelling budding yeast

- The eukaryotic organism *Saccharomyces cerevisiae* is known as budding, baker’s or brewer’s yeast
- It reproduces by mitotic cell division, in particular by a division process known as budding
- Budding is triggered by an intracellular pathway including processes on the cell membrane and in the cell body (cytosol)
- Important part: Formation of a Cdc42 cluster on the membrane
- The interaction of the active, membrane-bound Cdc42 complex and the inactive Cdc42 complex in the cytosol can be modelled by a turing–like mechanism
Modelling budding yeast – The model

- Let \( u_1 \) and \( u_2 \) denote the concentrations of the inactive complex and active complex, respectively.

- Then the considered model reads

\[
\frac{\partial u_1}{\partial t} = d_{vol} \Delta u_1 \quad \text{in} \quad \Omega_0 \times (0, T],
\]
\[
\frac{\partial u_2}{\partial t} = d_{sur} \Delta u_2 + r_2(u_1|\Gamma, u_2) \quad \text{on} \quad \Gamma \times (0, T],
\]
\[
d_{vol} \nabla u_1 \cdot n = r_1(u_1|\Gamma, u_2) \quad \text{on} \quad \Gamma \times (0, T].
\]
Modelling budding yeast – The model

Let \( u_1 \) and \( u_2 \) denote the concentrations of the inactive complex and active complex, respectively.

Then the considered model reads

\[
\frac{\partial u_1}{\partial t} = d_{vol} \Delta u_1 \quad \text{in} \quad \Omega_0 \times (0, T],
\]

\[
\frac{\partial u_2}{\partial t} = d_{sur} \Delta \Gamma u_2 + r_2(u_1|\Gamma, u_2) \quad \text{on} \quad \Gamma \times (0, T],
\]

\[
d_{vol} \nabla u_1 \cdot n = r_1(u_1|\Gamma, u_2) \quad \text{on} \quad \Gamma \times (0, T].
\]

Transitions between active / inactive complex are described by nonl. reaction kinetics coupling the vol. and surface process:

\[
r_2(u_1, u_2) = -r_1(u_1, u_2) = k_1 \cdot u_1 u_2^2 + k_2 \cdot u_1 u_2 - k_3 \cdot u_2
\]
Modelling budding yeast – Discretization

Heterogeneous Coupling, implicit domains  [E., Westerheide, submitted]

- Surface problem discretized as volume equation
- Coupling conditions are imposed along the surface
- Globally coupled block system

\[ A = \begin{pmatrix} A_{vol} & C_1 \\ C_2 & A_{sur} \end{pmatrix} \]
Heterogeneous Coupling, implicit domains  [E., Westerheide, submitted]

- Surface problem discretized as volume equation
  - implicit surfaces FEM (using DG), see [Dziuk and Elliot, 2008]
- Coupling conditions are imposed along the surface
- Globally coupled block system

\[
A = \begin{pmatrix}
A_{vol} & C_1 \\
C_2 & A_{sur}
\end{pmatrix}
\]
Modelling budding yeast – Discretization

Heterogeneous Coupling, implicit domains  [E., Westerheide, submitted]

- Surface problem discretized as volume equation
  - implicit surfaces FEM (using DG), see [Dziuk and Elliot, 2008]
- Coupling conditions are imposed along the surface
  - General framework allows for very different strategies
    - Later example uses Neumann-Neumann type continuity of fluxes
- Globally coupled block system

\[
A = \begin{pmatrix} A_{vol} & C_1 \\ C_2 & A_{sur} \end{pmatrix}
\]
Modelling budding yeast – Discretization

Heterogeneous Coupling, implicit domains  [E., Westerheide, submitted]

- Surface problem discretized as volume equation
  - implicit surfaces FEM (using DG), see [Dziuk and Elliot, 2008]
- Coupling conditions are imposed along the surface
  - General framework allows for very different strategies
  - Later example uses Neumann-Neumann type continuity of fluxes
- Globally coupled block system

\[ A = \begin{pmatrix} A_{vol} & C_1 \\ C_2 & A_{sur} \end{pmatrix} \]

\[
\begin{align*}
a_\epsilon(u_h, v) &= \int_{\Omega_0} d_{vol} \nabla u_h \cdot \nabla v \, dx + \int_{\Gamma_{\Omega_0}} \epsilon \langle d_{vol} \nabla v \rangle \cdot [u_h] - \langle d_{vol} \nabla u_h \rangle \cdot [v] \, ds \\
&\quad + \eta_{\text{vol}} h^{-1} \int_{\Gamma_{\Omega_0}} [u_h] \cdot [v] \, ds.
\end{align*}
\]
Modelling budding yeast – Discretization

**Heterogeneous Coupling**, implicit domains  [E., Westerheide, submitted]

- Surface problem discretized as volume equation
  - implicit surfaces FEM (using DG), see [Dziuk and Elliot, 2008]
- Coupling conditions are imposed along the surface
  - General framework allows for very different strategies
  - *Later example uses Neumann-Neumann type continuity of fluxes*
- Globally coupled block system

\[
A = \begin{pmatrix}
A_{\text{vol}} & C_1 \\
C_2 & A_{\text{sur}}
\end{pmatrix}
\]

\[
a_\epsilon(u_h, v) = \int_{\Omega} \tilde{d}_{\text{sur}} \nabla u_h \cdot \nabla v \, dx + \int_{\Gamma_{\Omega}} \epsilon \langle \tilde{d}_{\text{sur}} \nabla v \rangle \cdot [u_h] - \langle \tilde{d}_{\text{sur}} \nabla u_h \rangle \cdot [v] \, ds
\]

\[
+ \eta_{\text{sur}} h^{-1} \int_{\Gamma_{\Omega}} [u_h] \cdot [v] \, ds.
\]
Modelling budding yeast – Results

Initial values
Modelling budding yeast – Results

Evolution
Modelling budding yeast – Results

Result at final time
Introduction

Approaches for Multi-Domain Simulations

Reusable Software

Multi-Domain Software

Applications

Conclusions
Conclusions & Future Work

- Multi-physics / multi-domain application require high flexibility of the simulation code.
- Generic Programming allow efficient \textit{and} flexible code.
- Extensions to DUNE allow
  - domain decomposition on top of the DUNE grid interface
  - mixing of different discretizations and grids on each domain
  - time dependent domains

Thank you for your attention.
Conclusions & Future Work

- Multi-physics / multi-domain application require high flexibility of the simulation code.
- Generic Programming allow efficient and flexible code.
- Extensions to DUNE allow
  - domain decomposition on top of the DUNE grid interface
  - mixing of different discretizations and grids on each domain
  - time dependent domains
- Generic GFS and assembler for coupled problems.
- Implement parallel coupling concepts (not discussed today).
- DOFs and Equations on the sub-domain interface.
Conclusions & Future Work

- Multi-physics / multi-domain application require high flexibility of the simulation code.
- Generic Programming allow efficient \textit{and} flexible code.
- Extensions to DUNE allow
  - domain decomposition on top of the DUNE grid interface
  - mixing of different discretizations and grids on each domain
  - time dependent domains
- Generic GFS and assembler for coupled problems.
- Implement parallel coupling concepts (not discussed today).
- DOFs and Equations on the sub-domain interface.

Thank you for your attention.