Life(V): A modern and unified C++ implementation of finite-element and spectral-elements methods in 1D, 2D and 3D: overview and applications

Christophe Prud'homme

Université Joseph Fourier Grenoble 1

ICIAM'07
Zürich, July 18, 2007

Overview

LifeV

Consortium

http://www.lifev.org

- MoX, politecnico di Milano (Italy)
- Reo Team, INRIA Rocquencourt (France)
- CMCS, EPF Lausanne (Switz.)

Copyright and License

LifeV

Acknowledgements

Contributors

MoX A. Quarteroni, L. Formaggia, A. Veneziani, T. Passerini, ...

INRIA J-F. Gerbeau, M. Fernandez, ...

CMCS A. Quarteroni, G. Fourestey, S. Deparis, C. Winkelmann, ...

and

- M. Prosi formerly at Mox and CMCS
- C. Prud'homme formerly at CMCS and project manager of LifeV

LifeV

Features

Features

- LifeV is a C++ library that provides many features and tools. E.g. flexible and powerful numerical mesh manipulation and, in particular, the handling of moving meshes.
- Abstract 3D FE framework for manipulation and implementation of FE. As of now (Q0, Q1, P1, P2, P1-Bubble, RT).
- Stabilized numerical methods for handling convective dominated transport processes.
- Navier-Stokes solvers based on pressure-correction algebraic splitting methods.
- State of the art numerical methods for FSI.

LifeV

Spirit

Spirit

- ++ Collaborative effort
- ++ Suited for testing new numerical schemes / FE
- ++ (L)GPL, in particular open-source + free software
- – Developers coordination
- – Performances not always at the State of the Art
- – Developers are not computer scientists

Life

http://cmcstrac.epfl.ch:6789/projects/cmcslifev

- Université Joseph Fourier Grenoble 1
- CMCS, EPF Lausanne (Switz.)

Copyright and License

Life

Acknowledgements

Developers and users

CMCS/EPFL : E. Burman, S. Deparis, C. Winkelmann, G. Pena, B. Stamm

ASN/EPFL : G. Steiner

Life

Some Features

- Support 1D, 2D, 3D and basic entities: simplices and product of simplices
- Support various point sets on these basic entities: equispaced points, quadrature points, interpolation points (Gauss-Lobatto, Fekete, WarpBlend)
- Support various polynomial sets(Lagrange, Dubiner/BA, Legendre/BA, ...)
- Support continuous and discontinuous Galerkin methods
- Support for parallel computation and interfaces to GMM, PETSc/SLEPc, Trilinos
- Provide mathematical concept for higher order abstraction
- Interpolation in 1D, 2D, 3D
- Provide a language embedded in C++ ala FreeFem++

Debian/GNU/Linux

Scientific Computing Project

Objectives:

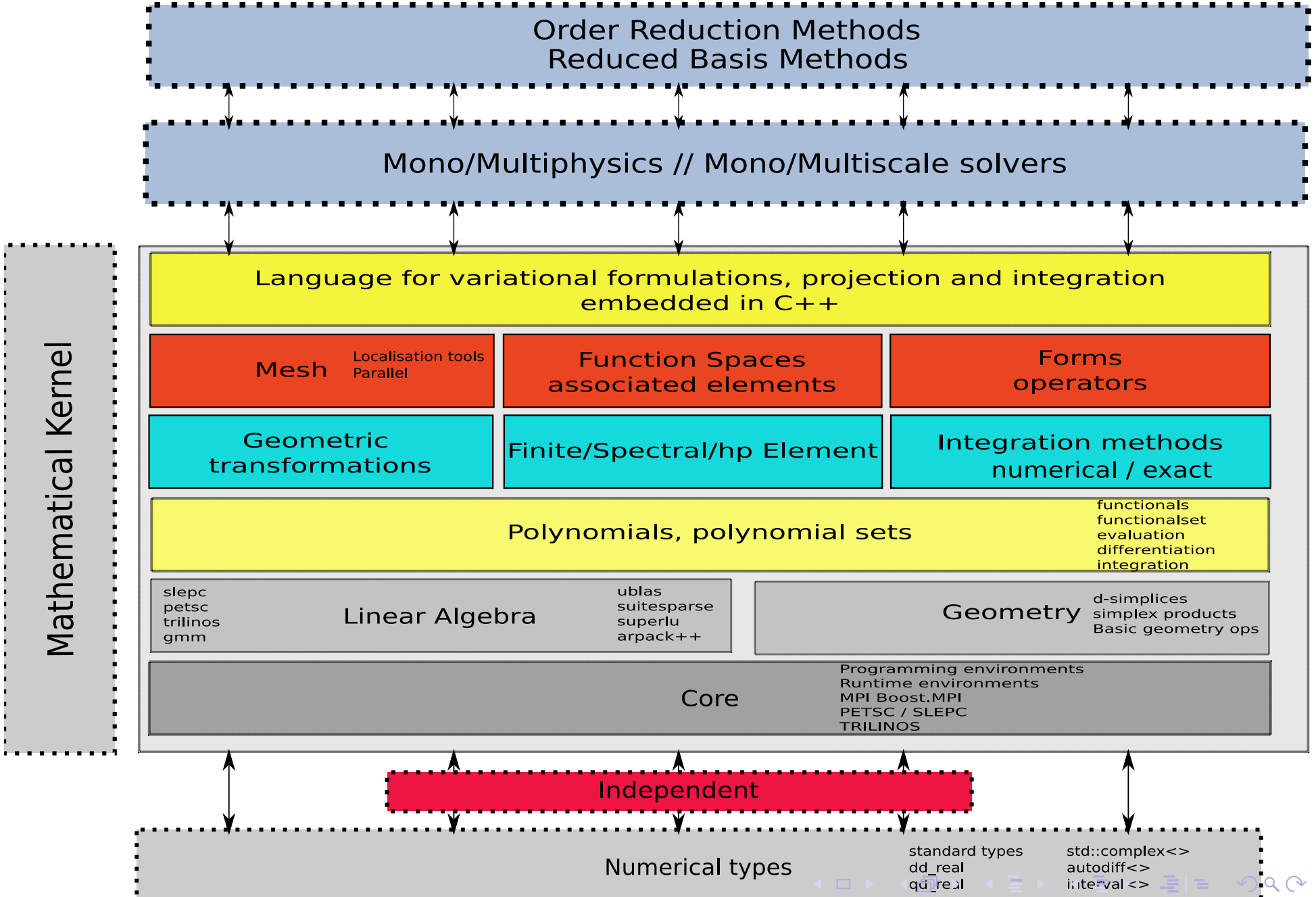
alioth.debian.org/projects/pkg-scicomp/

- Package state of the art, free software or opensource libraries/programs
- Provide feedback, bugreports to upstream
- Collaborate with other Debian groups: mpi, octave, petsc...
- Welcome non Debian developers

Packages

arpack*	cimg	freefem*	glpk	gmsh	netgen
opencv	openmx	parmetis	arprec/qd	qhull	slepc
suitesparse	sundials	superlu	tetgen	xmds	(life(v))

Architecture



Polynomials

Express the polynomials of $\mathbb{P}_k(K), K \subset \mathbb{R}^d, d = 1, 2, 3, ...$ in the Dubiner/Legendre basis $\{\phi_i\}_{i=1,\dim\mathbb{P}_k(K)}$, (see Kirby, Sherwin/Karnyadakis)

$$p = \sum_i (p, \phi_i)_K \phi_i, \quad p \in \mathbb{P}_k(K)$$

- Hierarchical L_2 orthonormal basis:
  - Trivial to extract a basis of $\mathbb{P}_l(K) \subset \mathbb{P}_k(K), l \leq k$
  - Exact integration (mass, stiffness,...)
  - Ease construction of polynomial families
- Algebraic representation of polynomials(vector) and polynomial sets (matrices) (reshape, scalar products, matrix/vector products, matrix/matrix products,...)
- Provide policies for polynomials with values in $\mathbb{R}, \mathbb{R}^d, \mathbb{R}^{d\times d}$

Polynomials

Construction

- Dubiner, Legendre, and associated boundary adapted versions(cG)
- Lagrange (cG, dG, EF, ES)
- $\mathbb{P}_{1,2}$-Bubble
- Raviart-Thomas (2D, 3D)
- Nedelec
- Divergence Free
- Crouzeix-Raviart
- More to come (Morley, Hermite,...)

Key Ingredient

Hierarchical L_2 orthonormal basis

Mesh

- Set of elements/convexes and an associated geometric transformation (P_1,P_2)
- Geometric entitites(convex and associated subentities) are stored using Boost.MultiIndex

Simple Parallel Strategy

- Each processor does the (element-wise) partitionning using Metis (BGL planned)
- Sorting over process id key allows for iterating over associated elements

Example

```
elements(mesh [, processid]);
markedelements(mesh, marker [, processid]);
```

Function Spaces

- Product of N-function spaces (a mix of scalar,vectorial, matricial and different basis types)
- Use Boost.MPL and Boost.Fusion heavily
- Get each function space and associated "component" spaces
- Associated elements/functions of N products and associated components, can use different backend (gmm,petsc/slepc,trilinos)

Example

```
typedef FunctionSpace<Mesh,vector<Lagrange<2,2,Vectorial>,
                                  Lagrange<2,1,Scalar> > > space_t;
space_t Xh( mesh );
space_t::functionspace<0>::type Uh (Xh.functionSpace<0>());
space_t::element_type x( Xh );
space_t::subelement<1>::type p( x.element<1>() ); // view
```

Operators and Forms

- Operators/Forms represented by full, blockwise(, matrix free) matrices/vectors
  - Full matrix $\begin{pmatrix} A & B^T \\ B & C \end{pmatrix}$
  - Matrix Blocks A, B^T, B, C
  - Free matrix: need to store variational expression
- The link between the variational expression and the algebraic representation

Example

```
typedef BilinearForm<X1,X2,MatrixPetsc> f1_t;
typedef BilinearForm<X1,X2,Block<MatrixPetsc,4> > f2_t;
f1_t a( Xh, Vh, M );
f2_t a( Xh, Vh, block( A, B, Bt, C ) );
```

A Language for PDEs

Automatic Computational Mathematical Modeling

- Generate automatically(ACMM) operators, matrices, vectors with variational formulations(FEM,SEM) for an easy integration with linear algebra software (Petsc, Trilinos, ...)
  See also FFC (FeniCs project) and Sundance
- High level language (DSEL) close to mathematical formulation (see also Freefem++/3D)

Example

```
// a : X1 × X2 → ℝ   a = ∫Ω ∇u · ∇v
form (X1,X2,M)=integrate( elements(mesh), im,
                                  gradt(u)*trans(grad(v)));
// l : X1 → ℝ   l = ∫∂Ω20 cos(πx)v
form(X1,L)=integrate( markedfaces(mesh,20), im,
                              cos(pi*Px())*id(v) );
```

A Language for PDEs

Enablers and Features

- Meta/Functional - programming (Boost.MPL...): high order functions, recursion, ...
- Crossing Compile-time to Run-time (Boost.fusion...)
- Lazy evaluations (allow for multiple evaluation engines)

Features: Use the *C++* compiler/language optimizations

- Optimize away redundant calculations (*C++*)
- Optimize away expressions known at compile time(*C++*)

A Language for PDEs

Enablers and Features

Example: a Navier-Stokes code

```
//standard terms
integrate( elements(mesh), im,
           nu*trace(gradt(u)*trans(grad(v)))+
           trans(idt(u))*id(v)/dt+
           (idv(u)*trans(gradt(u)))*id(v) +
           - idt(p)*div(v) + id(q)*divt(u) ) );
  // stabilization for equal-order approximation
+integrate(internalfaces(mesh), im,
           maxface( coeff * pow(h(),3.0) / max( h() *
                    sqrt( trans(beta)*(beta) ), (nu)))
           * ( jump(grad(q)*N())*jumpt (gradt(p)*N())));
```

A Language for PDEs

Enablers and Features

Example: a Linear-Elasticity code

```
AUTO( deft, 0.5*( gradt(u)+trans(gradt(u)) ) );
AUTO( def, 0.5*( grad(v)+trans(grad(v)) ) );
form( Xh, Xh, D ) =
 integrate( elements(mesh), im,
     lambda*divt(u)*div(v)  +
     2*mu*trace(trans(deft)*def) ) +
 on( markedfaces(*mesh,(nDim==2)?1:23),
     u, *F, constant(0)*one() );

MeshMover<mesh_type> meshmove;
meshmove.apply( mesh, u );
```

A Language for PDEs

Features

Compile-time Error Checks

- Verification that operations make sense with respect to rank (Scalar, Vectorial, Matricial) (static assertions)

```
id(u)+id(v) // u and v must be of the same type
```

- Check that terms can be handled (miscompilation)

```
cos(idt(u)) // Error
cos(idv(v)) // Ok!
```

- Check for exact integration (miscompilation)

```
cos(Px()*pi)*idt(u)*id(v) // Error
2*idt(u)*id(v) // Ok!
```

Applications

Applications

- Accuracy Benchmarks (test everything that can be tested)
  - Elliptic problems — Details
  - Hyperbolic problems — Details
  - Navier-Stokes problems — Details
- Efficient parallel preconditioner framework — Details
- Mono/Multi fluids flows — Details
- Hemodynamics — Details

Some On Going/Future Projects

Numerical Methods

Numerical Features

- More finite elements
- Wavelets
- P1 finite element preconditioner for high order methods
- Variable p
- Error estimators and hp-adaptation (bamg,3D?)

Navier-Stokes

- Generalization of the previous preconditioner to discontinuous pressures
- Algebraic factorization methods

Some On Going/Future Projects

Library features

Language

- More complicated expressions handling (many statements)
- Expression optimisation/simplification
- Features required by reduced-basis methods

Tools

- Better visualization of high order polynomials (gmsh?)
- More parallelisation and optimisation (assembly), e.g. use openmp

Some On Going/Future Projects

Applications

Applications

- Kernel for reliable real-time simulations (using RBOBM)
- Merge with LifeV: multiscale (3D/0D/1D), FSI, MT, ...
- Shallow water application
- Poisson-boltzmann application
- Spray simulation
- ...

END

Benchmark — Equations

$$\mu\Delta u = 0 \quad \mathrm{D},$$
$$\mu\Delta u + \sigma u = 0 \quad \mathrm{DR},$$
$$\mu\Delta u + \beta \cdot \nabla u + \sigma u = 0 \quad \mathrm{DAR}$$

with μ, σ and β constant or space-dependant:

	nD	**2D**	**3D**
$\mu(x,y,z) =$	1	$x^3 + y^2$	$x^3 + y^2 z$
$\beta(x,y,z) =$	$(1)_{i=1,n}$	$(x^3 + y^2, x^3 + y^2)$	$(x^3 + y^2 z, x^3 + y^2, x^3)$
$\sigma(x,y,z) =$	1	$x^3 + y^2$	$x^3 + y^2 z$

C++

Benchmark — Results

Dim	NEls	$\mathbb{P}_k$/Q	NDoF	D		DR		DAR	
				const	xyz	const	xyz	const	xyz
2D	20742	$\mathbb{P}_1$/3	10570	0.08	0.13	0.08	0.13	0.13	0.17
		$\mathbb{P}_2$/4	41881	0.2	0.23	0.2	0.23	0.25	0.29
	82460	$\mathbb{P}_1$/3	41 629	0.36	0.48	0.37	0.48	0.51	0.66
		$\mathbb{P}_2$/4	165717	0.81	0.99	0.83	1.02	1.07	1.42
	330102	$\mathbb{P}_1$/3	165850	1.47	2	1.5	2	2.06	2.6
		$\mathbb{P}_2$/4	661801	3.38	4.06	3.42	4.22	4.22	5.45
	517454	$\mathbb{P}_1$/3	259726	2.44	3.25	2.48	3.24	3.29	4.28
		$\mathbb{P}_2$/4	1036905	5.5	6.57	5.86	7.17	6.98	8.78
3D	8701	$\mathbb{P}_1$/4	1723	0.06	0.07	0.06	0.07	0.08	0.11
		$\mathbb{P}_2$/15	12825	0.42	0.46	0.43	0.48	0.49	0.58
	69602	$\mathbb{P}_1$/4	12239	0.49	0.73	0.5	0.75	0.69	1.09
		$\mathbb{P}_2$/15	96582	3.47	3.77	3.6	4.22	4.3	4.9
	554701	$\mathbb{P}_1$/4	92071	4.05	5.88	4.55	6.36	5.96	8.55
		$\mathbb{P}_2$/15	748575	31.27	34.42	32.35	36.16	35.82	42.08
	1085910	$\mathbb{P}_1$/4	178090	8.59	12.32	8.96	12.62	11.74	17.54

Table: Timings are in seconds on an AMD64 opteron 2.2Ghz running linux 2.6.9 with g++-4.0.1 and standard optimization. Quadrature rules to get exact integration for $\mathbb{P}_1$ and $\mathbb{P}_2$ functions. Q is the number of quadrature nodes used. (i) `xyz` just a little more expensive than `const` and (ii) little overhead when adding new terms (D→DR→DAR) to formulation.

Some More Benchmarking

Testing Exact Integration

Construct mass matrix with P1, P2, P5 and P8 in 2D

```
// Quad
vf_D = integrate( elements(*mesh),   IM, idt(u)*id(v));
// Exact
vf_D = integrate( elements(*mesh),   IMExact, mass(u,v));
```

Order	Elt	Dof	Quadrature	Exact
1/4	20786	10592	0.08	0.05
2/9	20786	41969	0.28	0.15
5/36	20786	260816	8.06	1.48
8/81	20786	666737	75.68	6.45

Testing '0' Terms

Influence of terms that should be 0

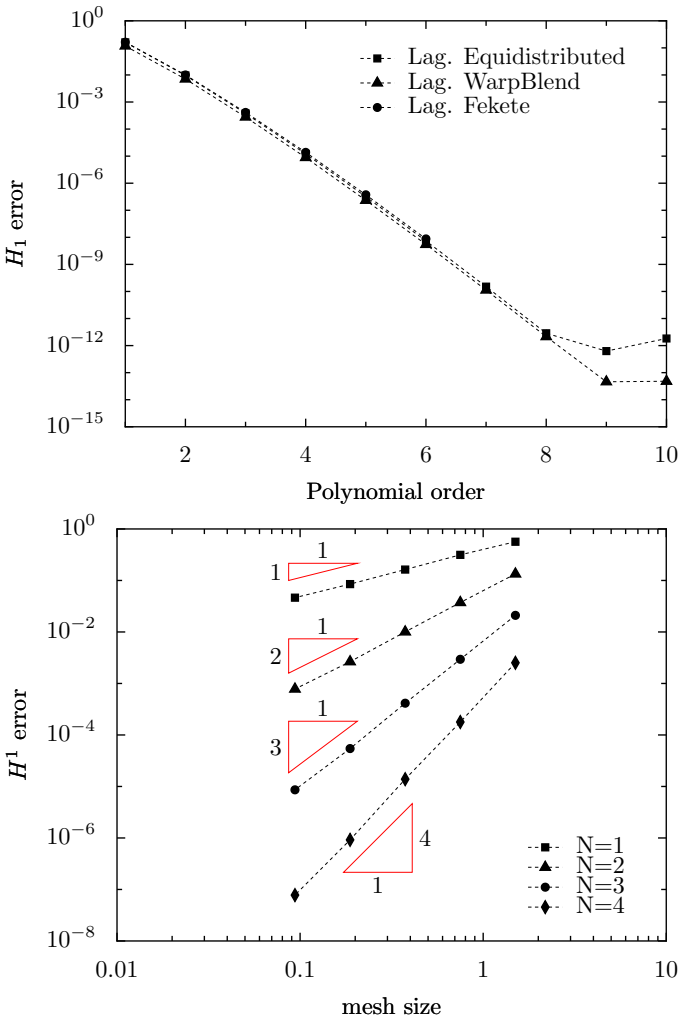
```
// 2D -> dz(.)-> 0
vf_D = integrate( elements(*mesh),
IM, dxt(u)*dx(v)+dyt(u)*dy(v)+dzt(u)*dz(v));
vf_D = integrate( elements(*mesh),
IM, dxt(u)*dx(v)+dyt(u)*dy(v));
```

Order	Elt	Dof	Without '0' Terms	With '0' Terms
1/4	20786	10592	0.11	0.11
2/9	20786	41969	0.48	0.5
1/4	20786	41618	0.47	0.45
2/9	20786	165673	1.92	1.94

Methodological Development

Elliptic Problems

Return



Collaboration with G. Pena and G. Steiner

Recover known results from literature (dependence in h and N)

- Results:
  - Multidomain (disc. across interfaces)
  - Multidomain (cont. across interfaces)
- Interpolation point sets: Equidistributed, Legendre-Gauss-Lobatto, Warpblend, Fekete
- High precision Gauss type quadrature rules

Methodological Development

Hyperbolic Problems: a problem with an internal layer

Return...

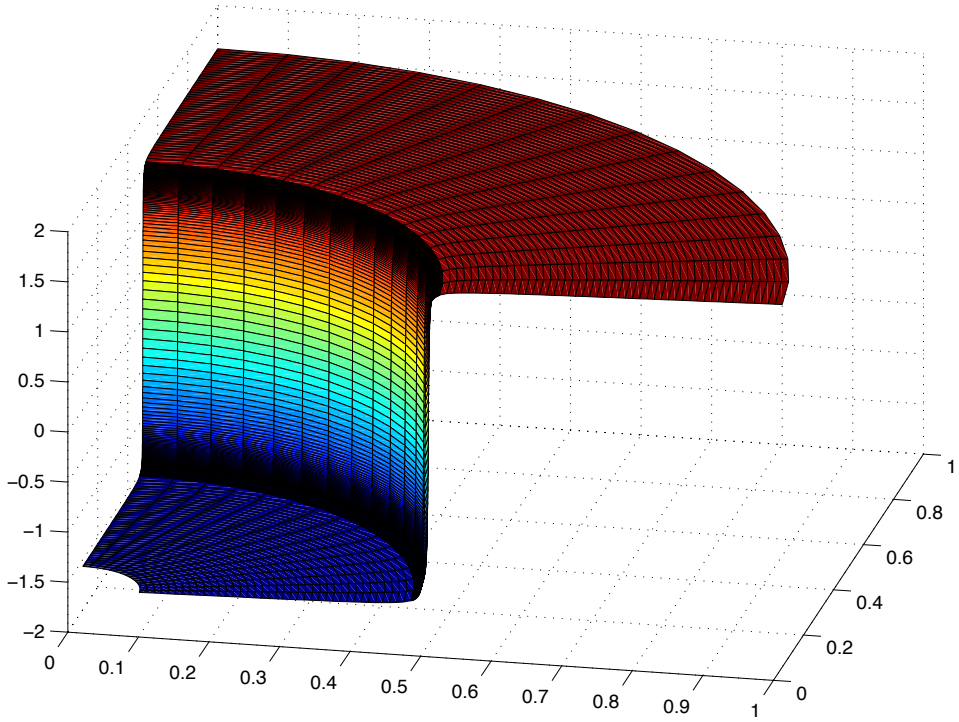


Figure: Exact solution. Internal layer problem

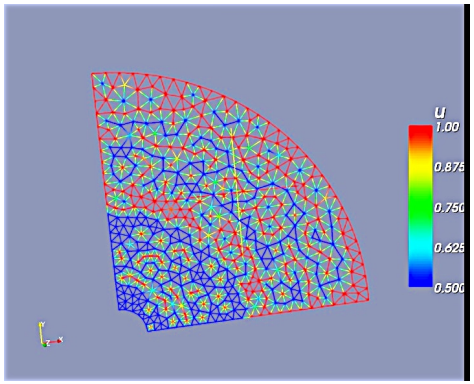
Stabilisation methods comparisons (B. Stamm)

Recover known results from literature (dependence in h and N)

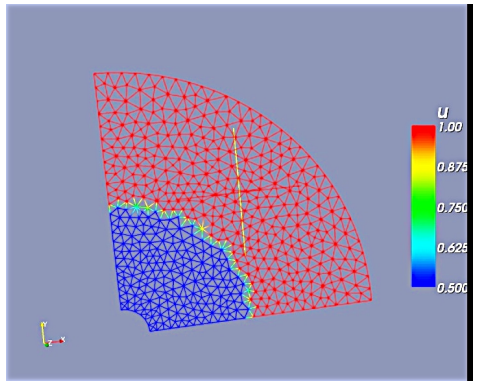
Methodological Development

Hyperbolic Problems: a problem with an internal layer

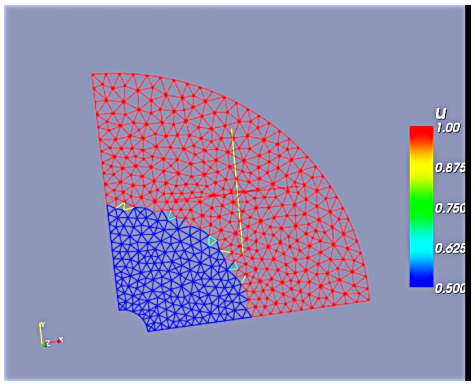
Return...



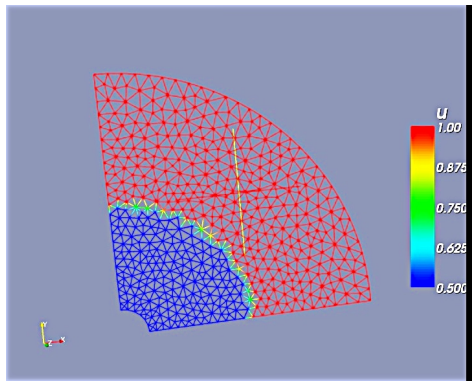
(a) cg



(b) cip



(c) dg upwind



(d) supg

Figure: Comparisons

Stabilisation methods comparisons (B. Stamm)

Recover known results from literature (dependence in h and N)

Methodological Development

Hyperbolic Problems: a problem with an internal layer

Return...

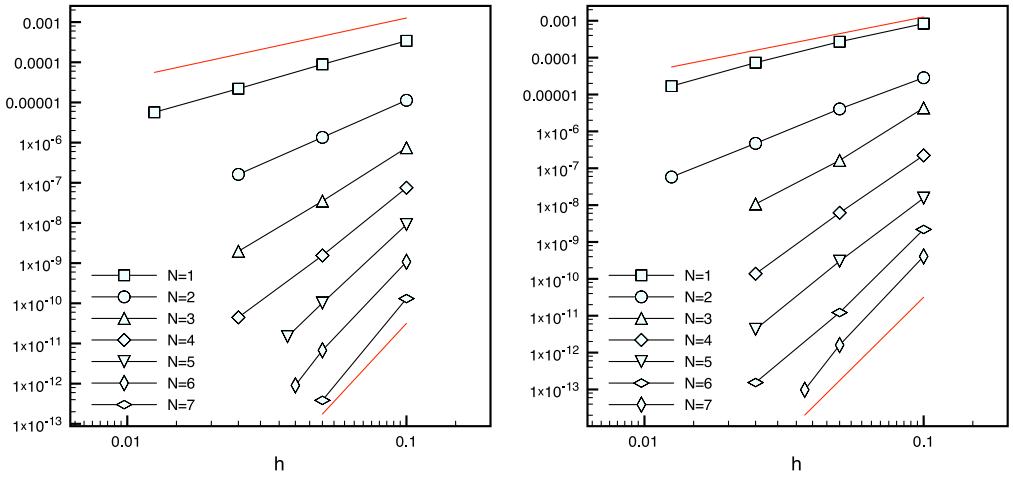


Figure: DG-Upwind and CIP Comparisons up to order 7. Red line are the optimal slopes (1.5 for $N = 1$ and 7.5 for $N = 7$)

Stabilisation methods comparisons (B. Stamm)

Recover known results from literature (dependence in h and N)

Optimal convergence of DG and CIP

Denote u the exact solution (with regularity $r >= 1$), then for all $1 \leq s \leq min(N + 1, r)$

$$||u - u_h||_{L_2} \leq C \frac{h}{N}^{s-1/2} |u|_{s,T_h}$$

Methodological Development

Hyperbolic Problems: a problem with an internal layer

Return...

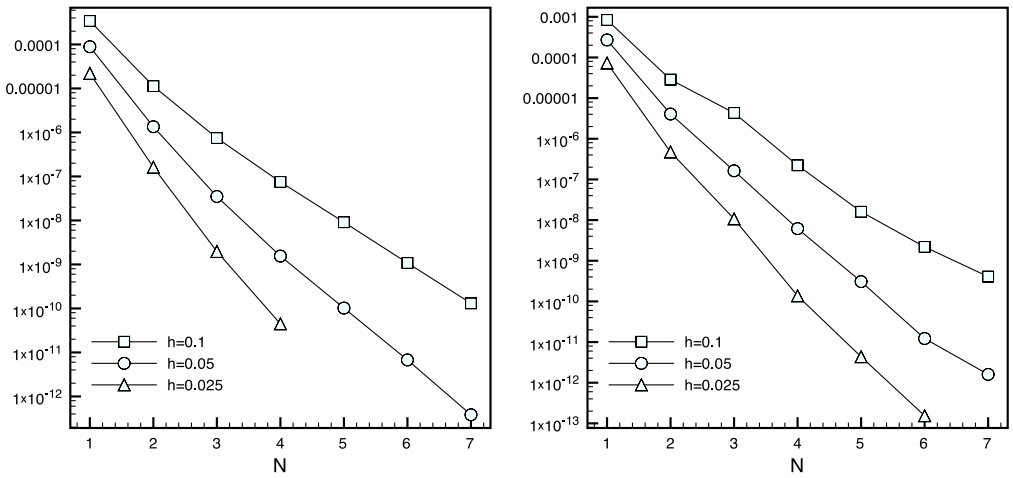


Figure: DG-Upwind and CIP Comparisons up to order 7. Exponential convergence.

Stabilisation methods comparisons (B. Stamm)

Recover known results from literature (dependence in h and N)

Optimal convergence of DG and CIP

Thus for smooth problems $\|u - u_h\|_{L_2} \leq C \frac{h}{N}^{N+1/2} |u|_{s,\mathcal{T}h}$ and thus exponential convergence...

Advection Reaction

Advection Reaction Code

Return

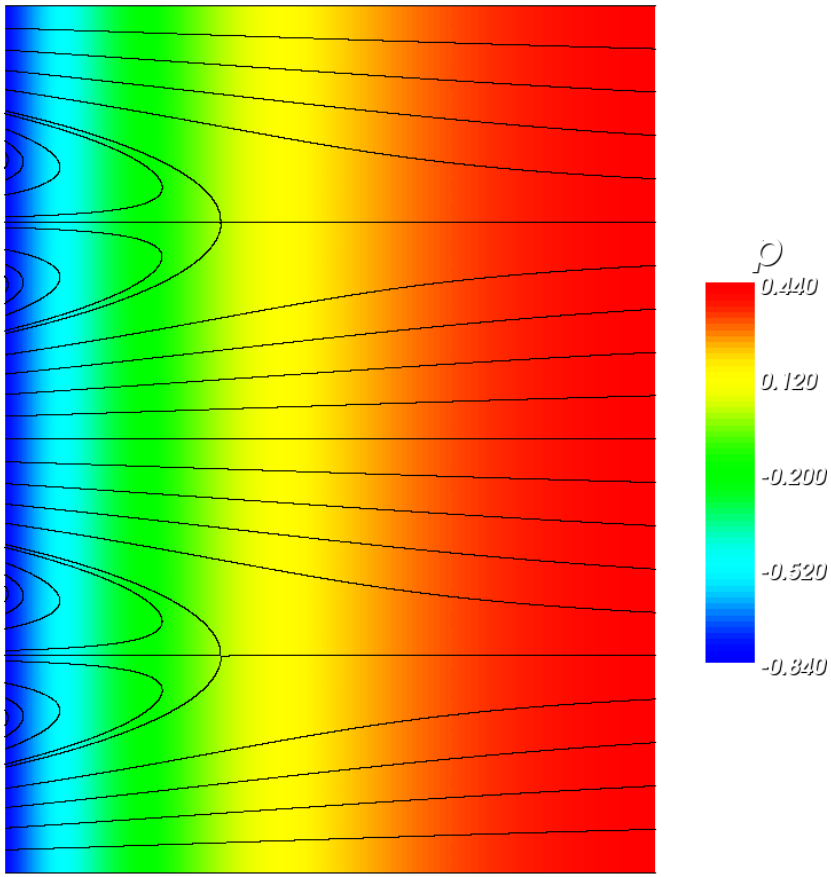
Advection code

```
// CIP stabilisation terms
M_operator = integrate( internalfaces(mesh), im,
     val( M_stabcoeff*vf::pow(hFace(),2.0) *
          abs(trans(beta)*N()) ) *
    (jumpt(gradt(phi)) * jump(grad(phi))) );

// volume terms
M_operator += integrate( elements(mesh), im,
    ( val(sigma)*idt(phi) +
      gradt(phi)*val(beta) ) * id(phi) ) +
// boundary terms
           integrate( boundaryfaces(mesh), im,
    - chi( trans(beta)*N() < 0 ) * /* inflow */
      val(trans(beta)*N()) * idt(phi) * id(phi));
```

Methodological Development

Navier-Stokes Problems

Return

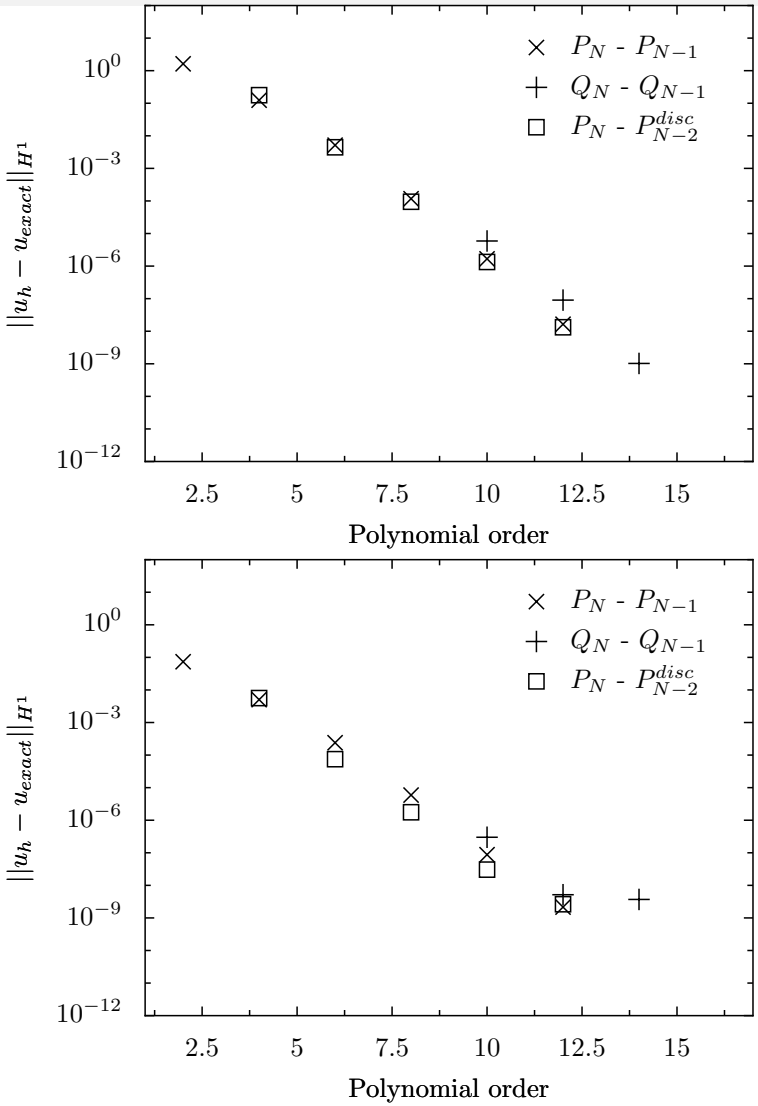


Steady Stokes with forcing term and non homogeneous Dirichlet boundary conditions

Methodological Development

Navier-Stokes Problems

Return



Steady Stokes with forcing term and non homogeneous Dirichlet boundary conditions

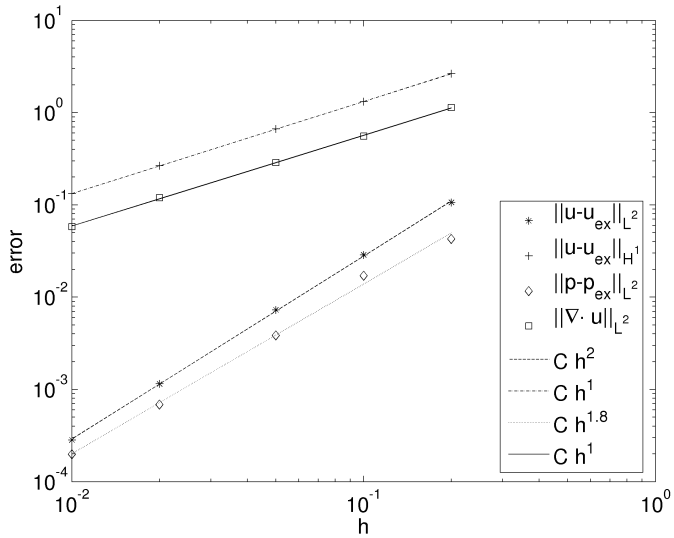
Kovasznay benchmark (G. Pena)

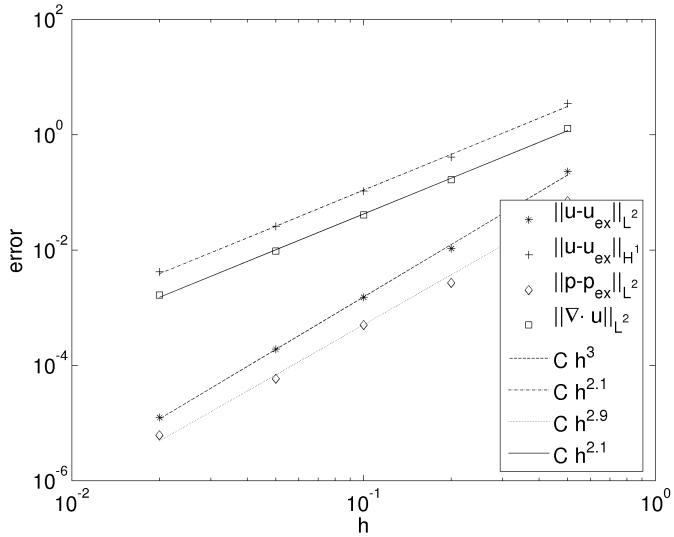
- Standard Galerkin discretization with strongly imposed bc
- Approximation spaces: $\mathbb{P}_N - \mathbb{P}_K$, $\mathbb{P}_N - \mathbb{P}_K^{disc}$, $\mathbb{Q}_N - \mathbb{Q}_K$ and $\mathbb{Q}_N - \mathbb{Q}_K^{disc}$
- Polynomial basis for the Galerkin method: Lagrange with Warpblend points

Methodological Development

Navier-Stokes Problems

Return





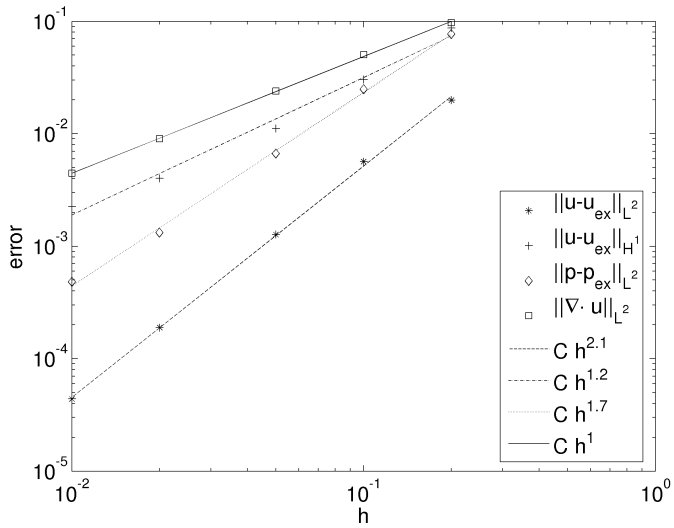
Steady Stokes with forcing term and non homogeneous Dirichlet boundary conditions

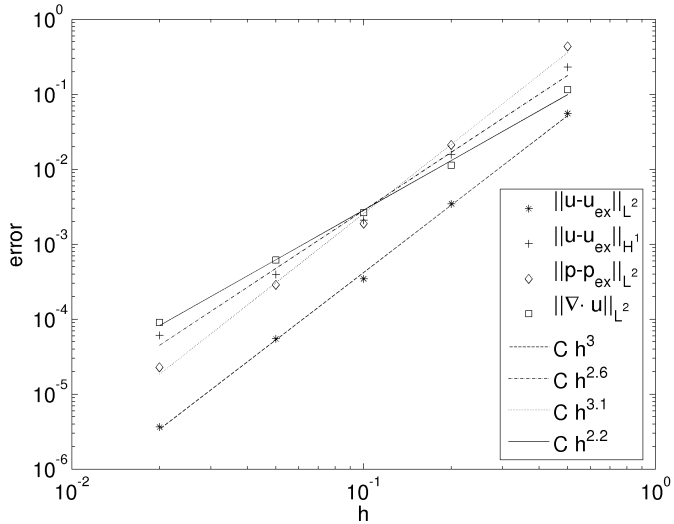
Kovasznay benchmark (C Winkelmann)

- Standard Galerkin discretization with weakly imposed bc
- Approximation spaces: $\mathbb{P}_1 - \mathbb{P}_1$, $\mathbb{P}_2 - \mathbb{P}_2$
- CIP stabilisation on the pressure

Methodological Development

Navier-Stokes Problems





3D time dependent Navier stokes benchmark

Kim-Moin benchmark (C Winkelmann)

- Standard Galerkin discretization with weakly imposed bc
- Approximation spaces: $\mathbb{P}_1 - \mathbb{P}_1$, $\mathbb{P}_2 - \mathbb{P}_2$
- Optimal convergence order
- Nonlinear Navier-Stokes (fixed point)
- Coupled linear system in u-p possible thanks to CIP

Methodological Development

Navier-Stokes Preconditioners

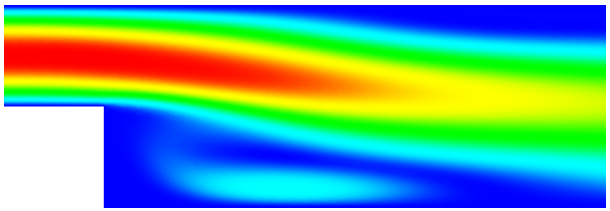
Return

Preconditioners for Stokes and Navier-Stokes equations (G. Pena)

- Block triangular preconditioner (Silvester, Elman, Kay, Walthen)

$$\begin{bmatrix} F & B \\ 0 & S^* \end{bmatrix}$$

- Schur complement approximation: $S^* = A_p F_p^{-1} M_p$ (Kay-Loghin)
- The exact block from the original problem, F, is used

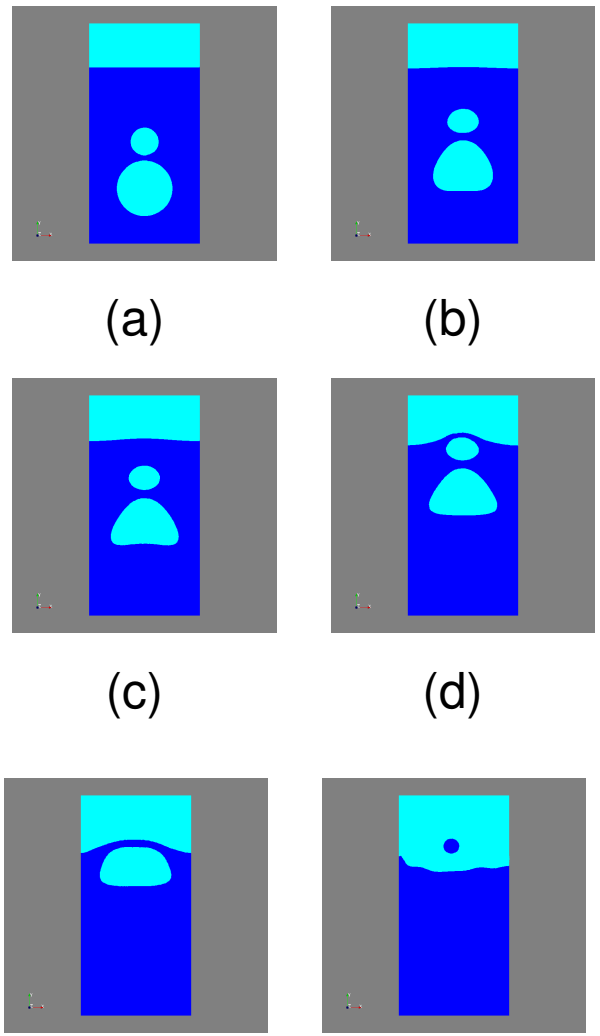


Recovered mesh size and polynomial order independence for $P_N \times P_{N-1}$ up to $N = 7$ for a backward facing step test case

Mono/Multi-fluid Flows

Mono/Multi-fluid Flows

Return



- N(=2) incompressible fluids flows
- Navier-Stokes equations in all Ω
- Variable density and viscosity

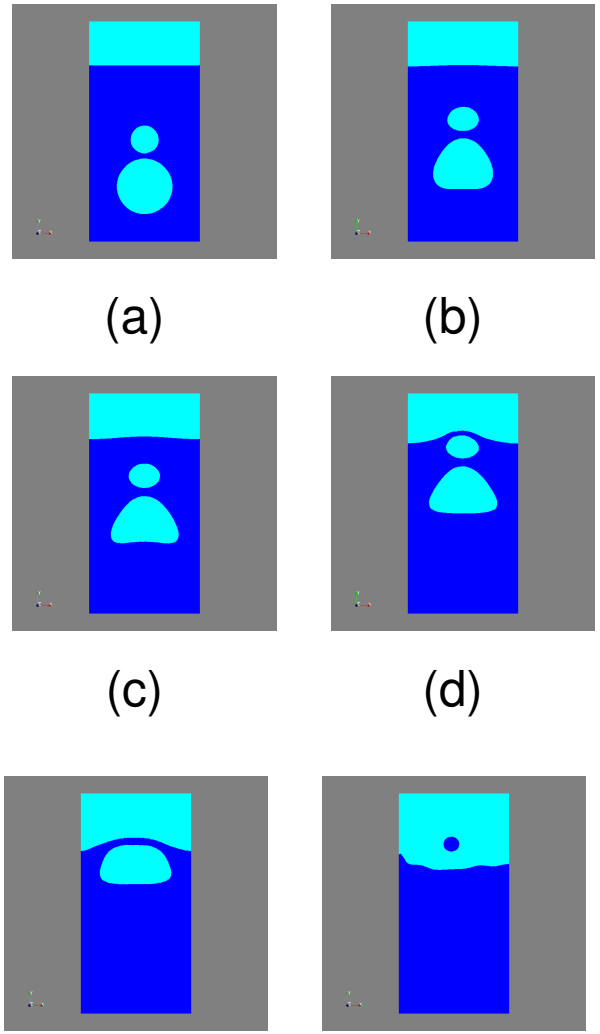
Level set approach: C. Winkelmann

- Advection of the level set function
- Explicit determination of the interface
- Integration on the interface (surfacic tension)
- Reinitialisation of the level set function

Mono/Multi-fluid Flows

Mono/Multi-fluid Flows

Return



- N(=2) incompressible fluids flows
- Navier-Stokes equations in all Ω
- Variable density and viscosity

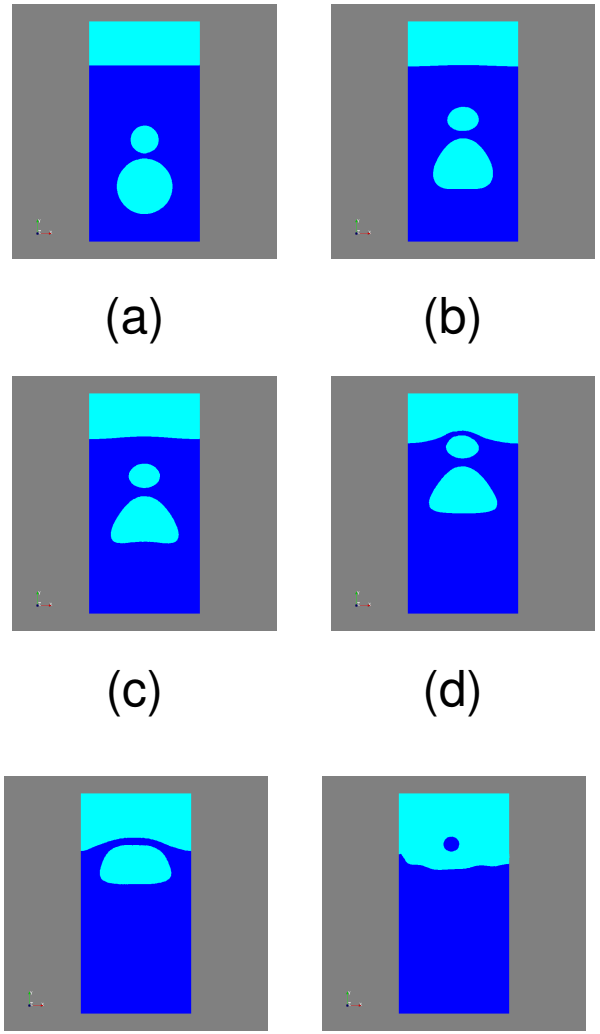
Level set approach: C. Winkelmann

- Fixed point iterations for all nonlinearities (convection, weak coupling Navier-Stokes - Level-Set)
- P2/P1 not stabilized for NS and P1 stabilized for CIP for levelset

Mono/Multi-fluid Flows

Mono/Multi-fluid Flows

Return



- N(=2) incompressible fluids flows
- Navier-Stokes equations in all Ω
- Variable density and viscosity

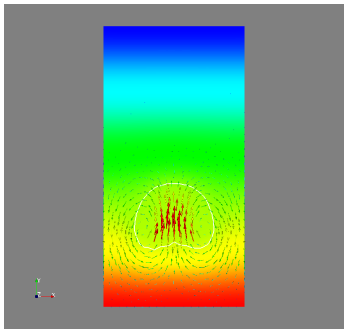
Level set approach: C. Winkelmann

- Changes in topology: 2 bubles (a-d) 1 buble (e), 1 droplet (f)
- Filament in (e) takes back the form of a circular bubble due to surfacic tension

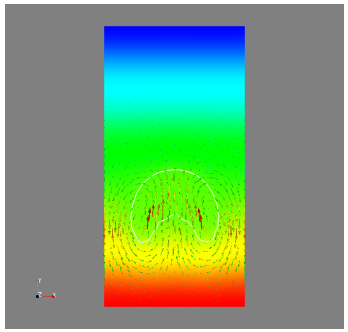
Mono/Multi-fluid Flows

Mono/Multi-fluid Flows

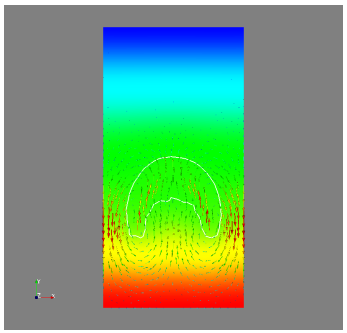
Return



(a) $t = 10s$



(b) $t = 15s$



(c) $t = 20s$

- N(=2) incompressible fluids flows
- Navier-Stokes equations in all Ω
- Variable density and viscosity

Bubble benchmark: C. Winkelmann

- Quantities of the benchmark recovered reliably
- Skirt formation

"Proposal for quantitative benchmark computations of bubble dynamics" by S. Hysing, S. Turek, D. Kuzmin (Dortmund), N. Parolini, E. Burman (EPFL), S. Ganesan et L. Tobiska (Magdeburg) (in preparation)

Mono/Multi-fluid Flows

Mono/Multi-fluid Flows

Return...

Computations using the levelset function

```
Mass = integrate( elements(*mesh), im,
           chi(idv(phi)>0) ).evaluate();

CenterOfMass = integrate( elements(*mesh), im,
           Py()*chi(idv(phi)>0) ).evaluate()/Mass;

RiseVelocity = integrate( elements(*mesh), im,
           idv(uy)*chi(idv(phi)>0) ).evaluate()/Mass;

perimeter = integrate( elements(*mesh), im,
           (35.0/(32.0*bandWidth)) *
           sqrt(gradv(phi)*trans(gradv(phi))) *
           pow(1.0-pow(idv(phi)/bandWidth,2.0),3.0) *
           chi(abs(idv(phi))<=bandWidth) ) ).evaluate();
```

Advection Reaction

Advection Reaction Code

Return

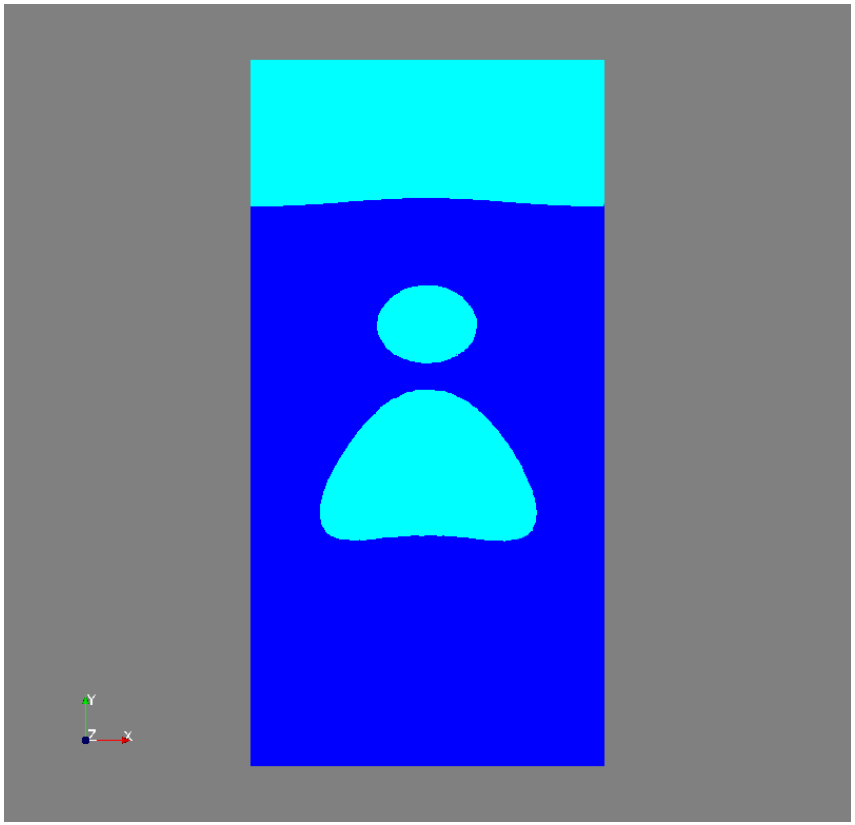
Advection code

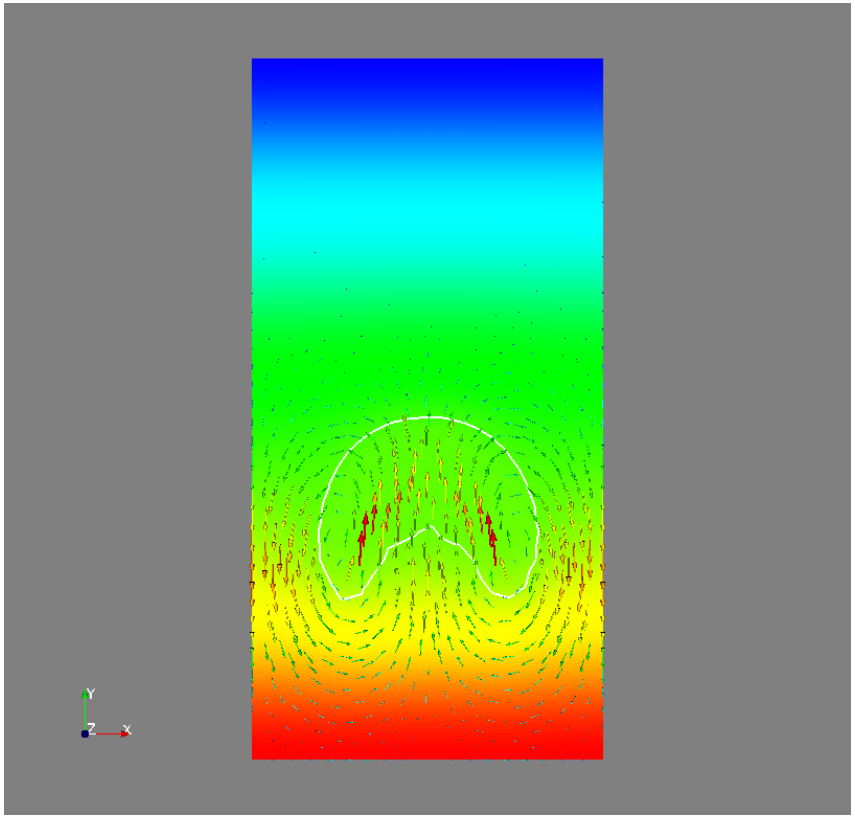
```
// CIP stabilisation terms
M_operator = integrate( internalfaces(mesh), im,
     val( M_stabcoeff*vf::pow(hFace(),2.0) *
          abs(trans(beta)*N()) ) *
    (jumpt(gradt(phi)) * jump(grad(phi))) );

// volume terms
M_operator += integrate( elements(mesh), im,
    ( val(sigma)*idt(phi) +
      gradt(phi)*val(beta) ) * id(phi) ) +
// boundary terms
           integrate( boundaryfaces(mesh), im,
    - chi( trans(beta)*N() < 0 ) * /* inflow */
      val(trans(beta)*N()) * idt(phi) * id(phi));
```

Mono/Multi-fluid Flows

Mono/Multi-fluid Flows

Return





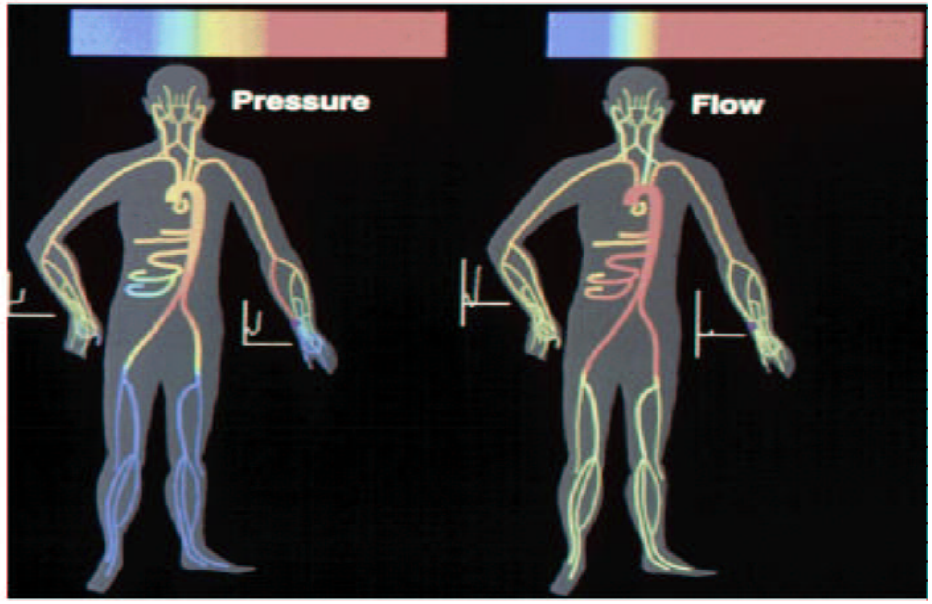
Hemodynamics

Medical Side

Return...

Vascular network is a robust system

Presence of occlusions and stenoses (local scale) can change significantly the global circulation (upstream and downstream)



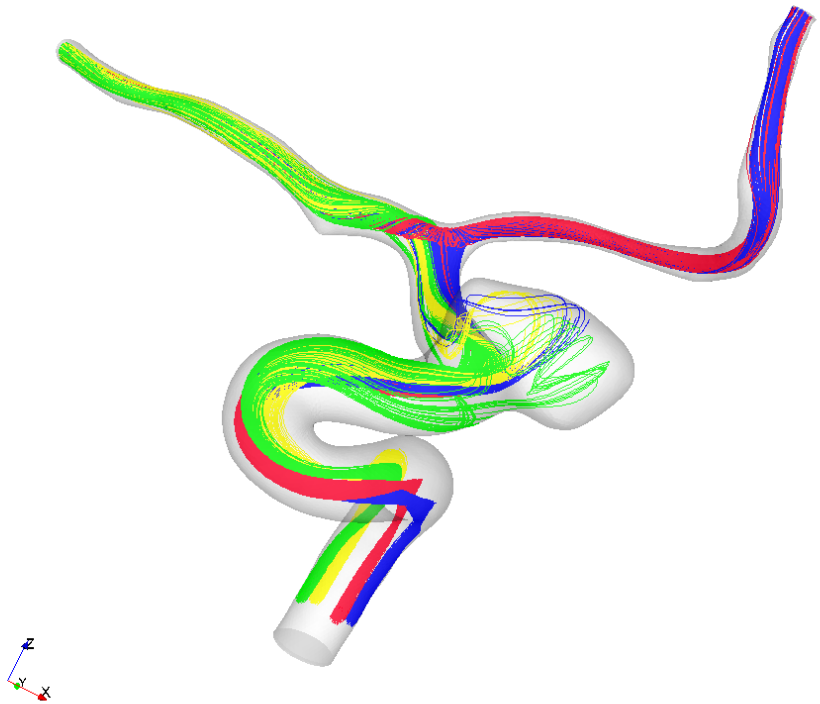
Compensatory mechanisms

peripheral districts can be reached by the blood following different pathways. Some vessels are unused in physiological conditions and activated only when needed.

Hemodynamics

Simulating the cardiovascular district

...Return...



Which data are to be prescribed on the artificial boundaries and how?

Boundary data should

- Account for the presence of the upstream/downstream vascular network
- Ensure correct solutions (absence or reduction of spurious effects)

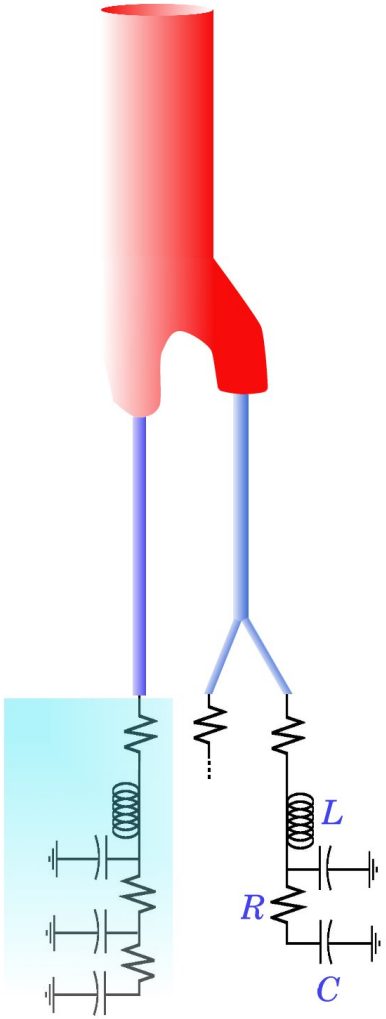
Integration of 0D/1D/3D models

Geometric multiscale approach

Hemodynamics

A hierarchy of models

...Return...



3D Navier-Stokes (F) + 3D ElastoDynamics (V-K)

1D Euler (F) + Algebraic pressure law

0D lumped parameters (system of linear ODEs)

Hemodynamics

Mass Transport

...Return...

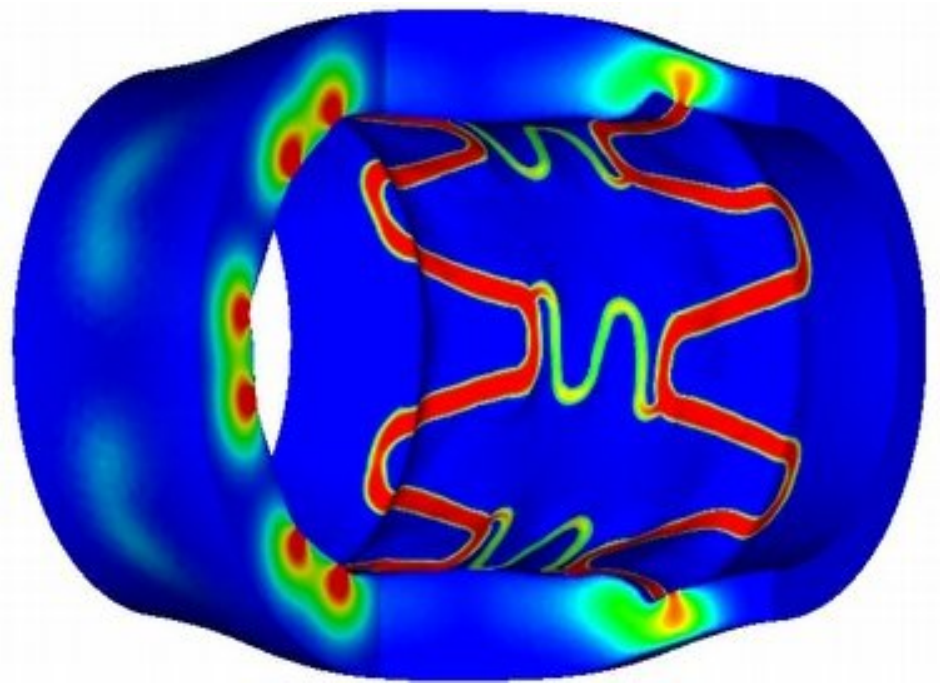


Figure: Concentration distribution after 15 hours

M. Prosi

- Navier-Stokes
- Convection-Diffusion-Reaction
- Darcy
- Different grids(interpolation)
- Multi-domains transport
- Physiological data and setting, results validation

Hemodynamics

Fluid-Structure Interaction

...Return...

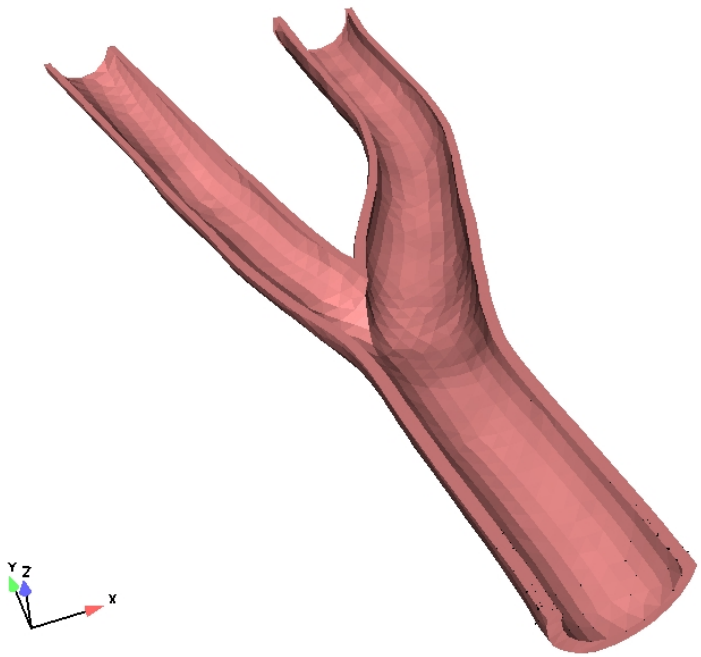


Figure: Carotid displacement

G. Fourestey, M. Fernandez & S. Deparis

Fluid/structure Interaction

- ALE Navier-Stokes
- (Non-)Linear Elasticity
- Implicit non-linear coupling
- Coupling operator preconditionning
- Parallel (due really soon now)

Mono/Multi-fluid Flows

Mono/Multi-fluid Flows

Return

- N(=2) incompressible fluids flows
- Navier-Stokes equations in all Ω
- Variable density and viscosity

Incompressible Navier-Stokes equations in the domain $\Omega \subset \mathbb{R}^3$

$$\begin{aligned}
\rho\partial_t\vec{u} + \rho(\vec{u}\cdot\vec{\nabla})\vec{u} - \vec{\nabla}\cdot(2\mu\epsilon(\vec{u})) + \vec{\nabla}p &= \rho\vec{g} \quad \text{dans } \Omega\times(0,T)\\
\vec{\nabla}\cdot\vec{u} &= 0 \quad \text{dans } \Omega\times(0,T)\\
\partial_t\phi + \vec{u}\cdot\vec{\nabla}\phi &= 0 \quad \text{dans } \Omega\times(0,T)\\
\rho(\phi) = \begin{cases} \rho_- & \phi<0 \\ \rho_+ & \phi>0 \end{cases}, \quad \mu(\phi) &= \begin{cases} \mu_- & \phi<0 \\ \mu_+ & \phi>0 \end{cases}\\
\vec{u} &= \vec{0} \quad \text{sur } \partial\Omega\times(0,T)\\
\vec{u} = \vec{u}_0 \quad \text{et} \quad \phi &= \phi_0 \quad \text{dans } \Omega \text{ à } t=0
\end{aligned}$$

Jump condition at the interface Γ

$$[\![\vec{u}]\!]_\Gamma = \vec{0}, \qquad [\![2\mu\epsilon(\vec{u}) - p\mathbf{I}]\!]_\Gamma\cdot\vec{n}_\Gamma = \kappa\sigma\vec{n}_\Gamma$$