

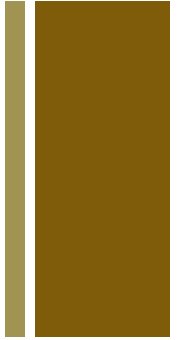


# Emory Updates

LifeV Workshop 2013

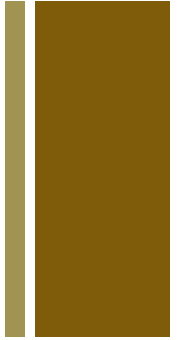


# The Emory team



- 1 supervisor (AV)
- 2 senior developers (TP, LB)
- 1 junior developer (Huanhuan Yang)
- 2 trainees (Boyi Yang, Jim Munch)
- Several users (mostly grads/undergrads)
- External collaborators (UV, Adrien Lefieux, Annalisa Quaini...)

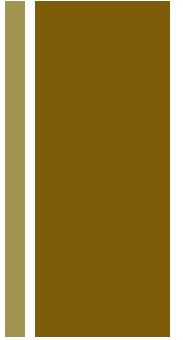
# + Projects



- CFD for blood flow problems (TP, AV, BY)
  - Pre-processing (mesh)
  - Post-processing (export, “secondary” computations)
  - **Validation**
  - Software usability, portability...
- Inverse problems / parameter estimation (LB, HY)
  - Verification
  - Efficiency

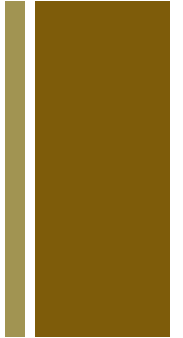


# Objectives (related to LifeV)



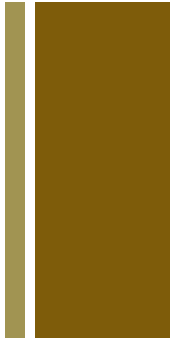
- Use LifeV in teaching labs (summer 2013)
  - Additional documentation
  - Website
  - Solved exercises
- Software distribution (2013)
  - Through cmcsforge
  - Through personal web pages? Github? ...
- Inject LifeV in the Biomed community
  - Workshop on software tools for Biomed Engineering

# + How to market LifeV?



- algorithms and data structures for the solution of PDE (with FEM)
  - Freely available! ([www.lifev.org](http://www.lifev.org))
  - strongly related to Trilinos (LifeV = “assembly” package for Trilinos-based FEM solvers)
  - “open” laboratory for **new ideas/methods**
  - **HPC technologies**
- a community of researchers
  - we make available (in principle) the tools we use to write papers

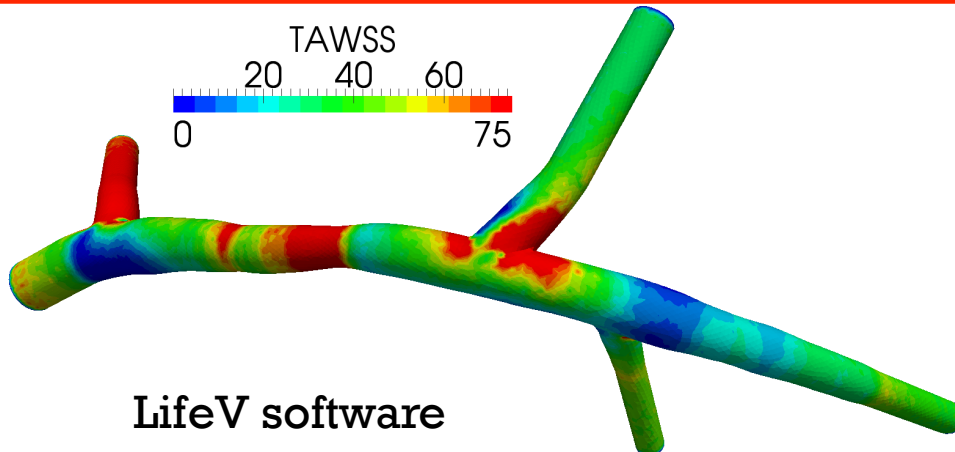
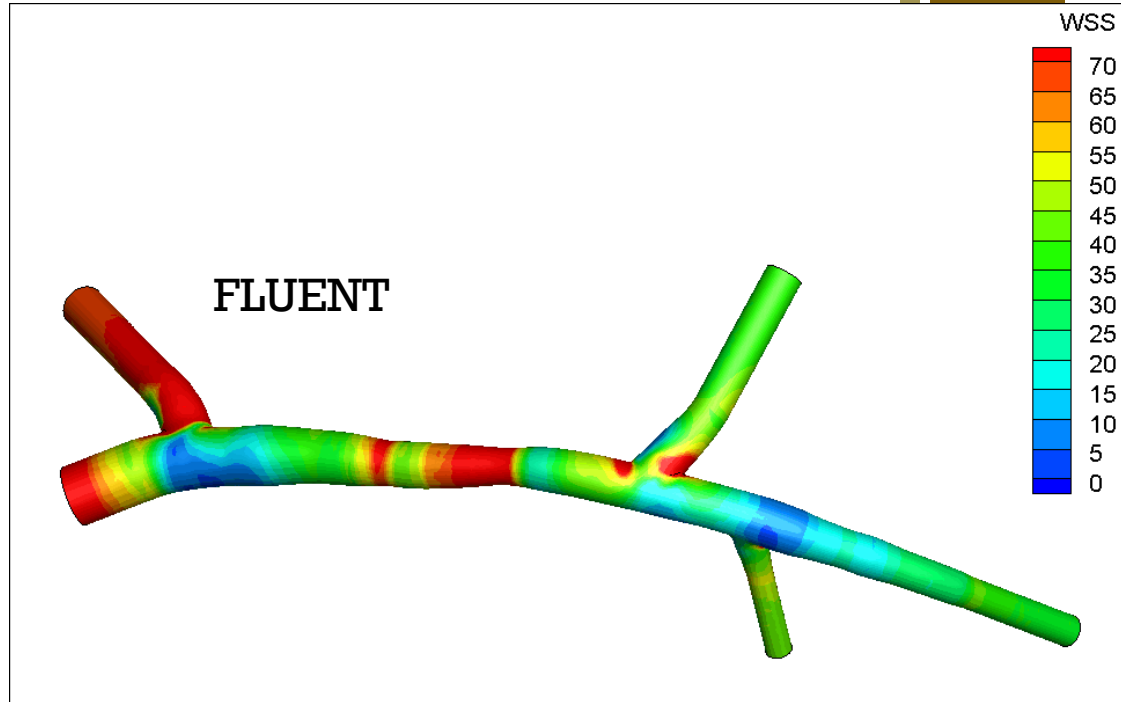
# + What LifeV is not (yet?)



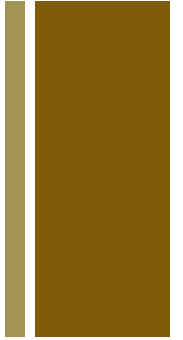
- teaching / educational tool
  - Easily solve easy problems
- FEM for dummies
  - As we teach it, we code it
    - e.g. assembly routines, ...
- seed for new collaborations
  - Enlarge the user base
- “software on demand”
  - Produce applications based on “LifeV as a library”
- collaborative development on a per-project basis
  - Shared code for shared papers
- ...

# + Promoting LifeV (I)

- Be competitive
  - Verification, validation
- Be better
  - e.g. mass conservation (avg. mass balance  $3e-7$  g/s), not achieved to the same level of accuracy by the Fluent solver



# + Promoting LifeV (II)



## ■ Be flexible

(tested machines, partial list)

- puma: 32 (2xDualCore) nodes, 128 cores, 8GB RAM per node (theoretically 256GB tot)
- crunch: 8xQuadCore, 32 cores, 200GB RAM
- Emory's HPC facilities
  - Ellipse cluster: 256 nodes, 1024 cores
  - facilities at Emerson Center: heterogeneous cluster, up to 240 cores
- XSEDE HPC resources
  - Trestles @ San Diego Supercomputer Center: 324 compute nodes, 10368 cores
  - Lonestar4 @ Texas Advanced Computing Center: 1888 nodes, 22656 cores
  - Steele @ Purdue University: 902 nodes, 7216 cores
- Collaboration with Italian HPC center Cilea
  - HP cluster Lagrange: 208 nodes, 1664 cores
- Amazon EC2
- ...



# + Promoting LifeV (III)



## ■ A clear and easy design

The basic steps of the simulation:

- i. Mesh generation/reading
- ii. Mesh  $\rightarrow$  matrices & vectors
- iii. Linear system solve
  - a. Preconditioner update
  - b. Solve
- iv. Processing & exporting

### (i) Preprocessing

Problem definition  
Mesh computation  
Time independent operations

Time Advancing{

### (ii) Assembly

### (iii) Solution at $t^k$

(iiia) Preconditioner construction

(iiib) Iterative solution

} //End Time Advancing

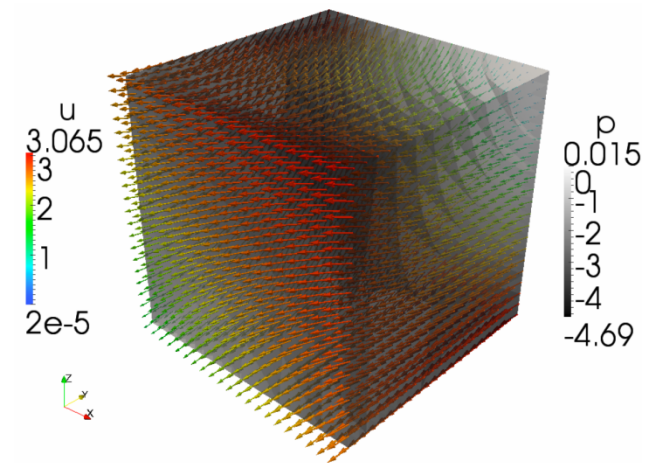
### (iv) Postprocessing

Non-primitive variable computation  
Visualization



# LifeV Application

```
NavierStokesApplicat  data  NavierStokesApplicat  write_boundary_point  data  TimeA
1 #include "TestNavierStokes.hpp"
2 #include <boost/scoped_ptr.hpp>
3
4 using namespace LifeV;
5
6 // Do not edit
7 int main(int argc, char **argv)
8 {
9     using namespace LifeV;
10 #ifdef HAVE_MPI
11     MPI_Init(&argc, &argv);
12     std::cout<< "MPI Initialization\n";
13 #endif
14
15     boost::scoped_ptr<TestNavierStokes> testPtr( new TestNavierStokes( argc, argv ) );
16     testPtr->run();
17
18     int result( EXIT_FAILURE );
19     if( testPtr->checkErrors() )
20     {
21         result = EXIT_SUCCESS;
22     }
23     testPtr.reset();
24
25 #ifdef HAVE_MPI
26     MPI_Finalize();
27     std::cout<< "MPI Finalization \n";
28 #endif
29
30     return EXIT_SUCCESS;
31 }
```



(C. R. Ethier, D. Steinman, 1994)

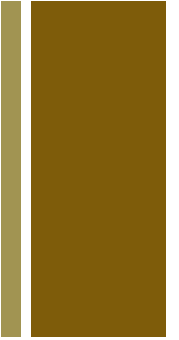


# LifeV Application

```
NavierStokesApplicat  data
45
46
47 void NavierStokesApplicatio
48 {
49     Debug() << "[NavierStok
50
51     readDataFile();
52
53     buildMesh();
54     buildFESpaces();
55     buildAssembler();
56     buildOperator();
57     this->buildExporter();
58
59     setupBDF();
60
61     this->initialize();
62
63     this->setBC();
64     this->timeLoop();
65 }
66
67
68 void NavierStokesApplicatio
69 {
70     Debug() << "[NavierStok
71
72     //-----
73     //  Data File
74     //-----
75     // boost::shared_ptr<Te
76     M_teuchosListPtr.reset(
77     Teuchos::updateParamete
78
```

```
48
49 class TestNavierStokes :
50 public NSApplication
51 {
52 public:
53
54     typedef LifeV::RossEthierSteinmanUnsteadyDec problem_type;
55
56
57 /** @name Constructors, destructor
58 */
59 //@{
60
61 TestNavierStokes( int argc, char** argv ) :
62     NSApplication( argc, argv ),
63     M_L2err_velocity(0.),
64     M_L2err_pressure(0.),
65     M_tolerance_velocity(0.),
66     M_tolerance_pressure(0.)
67 {}
68
69 virtual ~TestNavierStokes() {}
70 //@}
71
72 bool checkErrors();
73
74 virtual void initialize();
75 virtual void setBC();
76
77 virtual void postProcess( const Real& /*t*/ );
78
79 protected:
80     virtual void setProblemData();
81
82     void initializeErrorLog();
83     void printErrorLog();
84
85     LifeV::Real M_L2err_velocity, M_L2err_pressure;
86     LifeV::Real M_tolerance_velocity, M_tolerance_pressure;
87 };
88
```

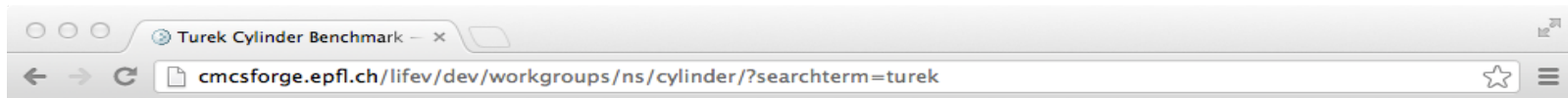
# + Promoting LifeV (IV)



- Let the software circulate!
  - Lifev.org web portal
  - Personal/private pages
  - Public portals (github)

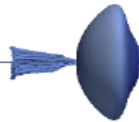


# Example: lifev.org



Accessibility Contact

# LifeV



Search Site

☐ only in current section

About LifeV

Development

Gallery

Downloads

Documentation

Members

Groups

Events

News

Log in Register

You are here: Home → Development → Workgroups → Navier-Stokes → Turek Cylinder Benchmark

Search

Search Site

Advanced Search...

Navigation

Home

About LifeV

Development

Roadmap

Workgroups

Core

Parallel

Serial

Mass Transport

Navier-Stokes

Meeting April 2006

NSPC and NSIP:  
preliminary analysis

Turek Cylinder  
Benchmark

NSPC and NSIP:  
preliminary analysis,  
continued

Turek Cylinder

## Turek Cylinder Benchmark

Results of the Turek cylinder benchmark computations of LifeV Navier-Stokes solvers  
updated 17 November 2006 (old version [here](#))

Testcase 3D-Z1 from [Schaefer, Turek](#). See there for details about geometry etc.

The solution is steady, and it is found by running the time dependent solvers with 1st order BDF (=backward Euler) until  $t=10$ , about where a stationary state is reached. The timestep is chosen as equal to the largest mesh width, which is 12.5 times the smallest mesh width. With a maximal inflow velocity of 0.45, this would give a maximal CFL of 5.625. Note that the velocity is low where the mesh is fine (i. e. near the cylinder).

The drag and the lift have been evaluated by numerical integration of the stress on the cylinder. The pressure difference was calculated using a  $L^2$  scalar product with two regularized delta functions located at the points of interest. The mass error is the integral over time of [inflow-outflow]. Implementation details can be found in [life-playground/benchmark/cylinder/turek](#). I didn't want to put the finest mesh into cvs, you can find it [here](#).

See the current results here, with the bounds indicated by Schaefer and Turek:

Solver	$N_{dof}$	$h=\Delta t$	drag	lift	$\Delta p$	mass error	cpu sec >per timestep<th>	Memory / MB
lower bound			6.05	0.0080	0.165			
upper bound			6.25	0.0100	0.175			
PC P1bubble- P1	36540	0.2	6.19	-0.0043	0.189	2.0e-6	42.9	46
	165236	0.1	6.10	-0.0094	0.172	2.3e-6	432.0	174

« January 2013 »

Mo	Tu	We	Th	Fr	Sa	Su
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

News

Gallery updated  
Sep 16, 2010

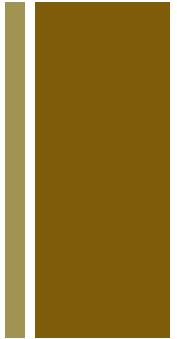
LifeV-Serial 1.0  
released  
Apr 27, 2010

LiveV workshop  
talks online  
Apr 22, 2010

More news...



# Example: “private” web portals



Tutorials - ECM2 group

https://sites.google.com/site/ecm2group/lifev/tutorials

Tutorials Updated Sep 24, 2012 8:28 AM

Tiziano Passerini

Search this site

ECM2 group

Home

LifeV

Examples

Tutorials

Sitemap

LifeV > Tutorials

LifeV in the Computer Lab

## Setting up the environment

### Notation

- Commands to be typed in the terminal will be preceded by a “greater than” symbol (>) and will be written in **bold and fixed width font**.
- Names of files and folders will be written in *fixed width font*.

### Tips

- to edit a text file named “filename”  
> **gedit filename**  
(if the file is not existing, it will be created when you save it)
- to inspect the content of the text file “filename” without opening the text editor “gedit”  
> **cat filename**  
(the content will be printed on screen)
- while typing in the terminal, press **TAB** once to activate the automatic completion of the



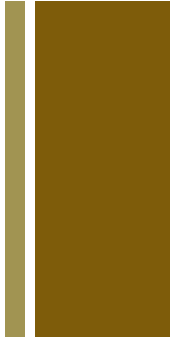
# Example: github

The screenshot displays a web browser with three tabs: 'tpass', 'tpass/meshUtilities - GitHub', and 'meshUtilities/Makefile.SAMPLE.in'. The active tab shows the GitHub repository page for 'tpass / meshUtilities'. The repository has 0 pull requests, 0 issues, 0 stars, and 0 forks. The 'Code' tab is selected, showing the 'Makefile.SAMPLE.in' file. The file was committed by 'tpass' 5 months ago with the message 'initial commit for application MeshInfo'. The file content is as follows:

```
1 #
2 # The paths to LifeV
3 #
4 LIFEPATH =
5 LIFELIBPATH = $(LIFEPATH)/lib
6 LIFELIBS = -llifefunctions -lllifefilters -lllifeoperators -lllifesolver \
7           -lllifefem -lllifealg -lllifearray -lllifecore -lllifemesh
8 LIFEINCLUDEPATH = $(LIFEPATH)/include
9 LIFELDFLAGS = -L$(LIFELIBPATH)
10
11 #
12 # The paths to BLAS/LaPACK
```



# The ECM2 module



- A collection of code experiments
- A code forge (preliminary stage for the library)
- Playground / applications
- Content
  - A NS solver, various preconditioners (block operators)
  - A solver for linear elasticity (time advance)
  - A monolithic FSI solver (block operators)
  - Classes to simplify the implementation of “applications”
  - Classes/routines for manifold handling (boundary mesh extractor, assembler, FESpace,...)
  - Classes for BC / problem set up
  - ...