Python, Open-Source, and Numerical Solutions of Conservation Laws

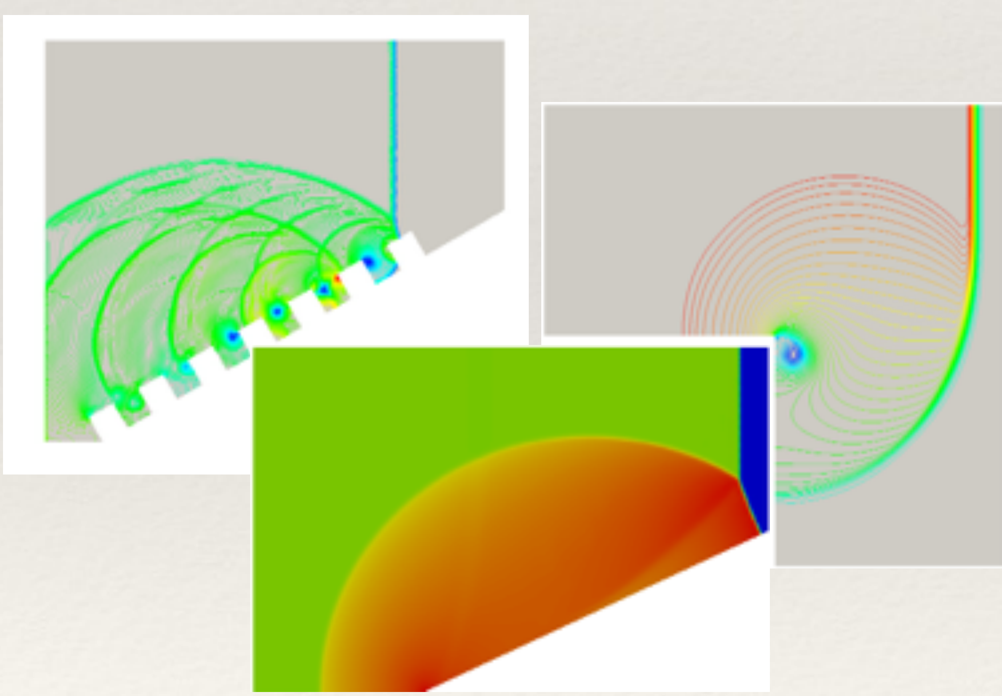
Yung-Yu Chen, Po-Hsien Lin,
Sheng-Tao John Yu

SOLVCON Project

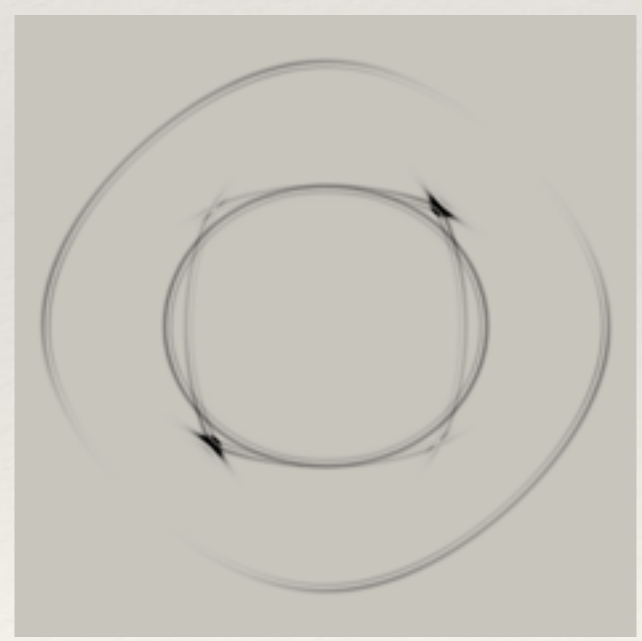
@ ISGC 2014 March 28

Introduction

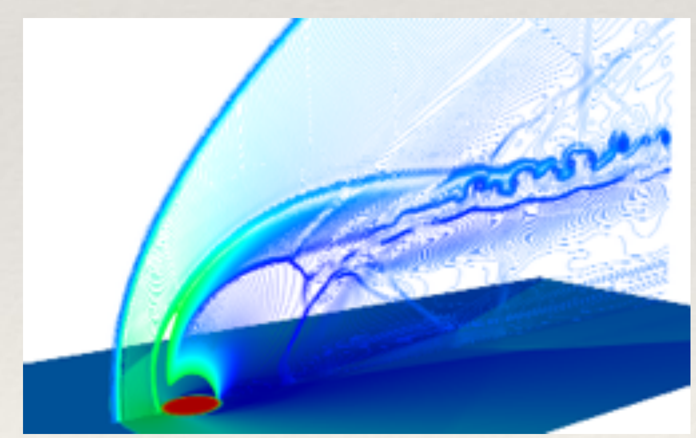
- Explore how to engineer numerical simulations of conservation laws and build a software framework (SOLVCON).
- Examples of the simulations:



Aerodynamics Benchmarks



Stress Waves in Anisotropic Solids



Supersonic Jet in Cross Flow

Goals

- We want multiple solvers for physical processes governed by the equations taking the quasilinear form:

$$\frac{\partial \mathbf{u}}{\partial t} + \sum_{k=1}^{3} \mathrm{A}^{(k)}(\mathbf{u}) \frac{\partial \mathbf{u}}{\partial x_k} = \mathbf{s}(\mathbf{u})$$

- We want an extensible system to modularize physical processes.
    - It is a merit provided by the space-time conservation element and solution element (CESE) method, and we want to explore its applicability.
    - Plan ahead for interoperation.
- We want parallel code (scale from 1,000 cores to more). We want hybrid parallelism for heterogeneous architecture.
- We want the code to be well maintained. We don't want spaghetti code patched from paper to paper. We want to code systematically.

It's an Engineering Problem

- The goals can be generalized to developing SOLVCON, a software framework for the CESE method. It's an engineering problem for scientific research.
- Predecessors of SOLVCON began as a mesh processing utility.
  - Two-/three-dimensional mesh and solution data need to be shared among gas-dynamics CFD (computational fluid dynamics) tools for reuse and analyses.
- The predecessors are redeveloped to a software framework for the space-time conservation element and solution element (CESE) method since 2008. We
  - wanted to explore the applicability of the CESE method, and thus
  - needed to develop solvers for multiple physical processes.
- All goals (extensibility, modularity, parallel computing, and maintainability) are achieved, but with many rough ends.

Python and Open-Source

- Python: Build everything upon the scientific Python ecosystem.
  - Various built-in and third party packages. More than 41,392 on https://pypi.python.org/pypi now.
  - ndarray (N-dimensional array) of NumPy.
  - Cython for interfacing between Python and C.
- Open-source: Open up everything, from source code to documents.
  - SOLVCON online documents can be a better vehicle than sole papers for propagating knowledge.
  - Frictionless collaboration.

Development Activities

1. Documenting for reproduction.
   - Dense knowledge in both natural and computer languages.
   - Propagate the knowledge with reproducibility.
2. Architecting and developing code.
   - Before writing it, we don't know what to expect.
   - Agile methodology must be employed for the uncertain nature.
3. Testing and validation.
   - We need unit tests and answer tests for development.
   - Running answer tests can take hours. We need a server farm.

Documenting

- SOLVCON documents should contain what's in conventional research papers.
- What we want is more than conventional papers.
  - Bi-directional linking between the code and the documents.
  - Some part of the documents should be runnable. It's related to validation.

Sphinx

- Sphinx (http://sphinx-doc.org/) is the default tool for Python documentation.
- ReadTheDocs (https://readthedocs.org/) is a service to host Sphinx documents.
- SOLVCON authors and organizes its documents with Sphinx.
    - The documents are published at http://www.solvcon.net/ (ReadTheDocs is behind the scenes).

Formulations (Under Development)

The governing equations of the hydro-acoustic wave include the continuity equation

$$\frac{\partial \rho}{\partial t} + \sum_{i=1}^{3} \frac{\partial \rho v_i}{\partial x_i} = 0 \tag{1}$$

and the momentum equations

$$\frac{\partial \rho v_j}{\partial t} + \sum_{i=1}^{3} \frac{\partial(\rho v_i v_j + \delta_{ij} p)}{\partial x_i} = \frac{\partial}{\partial x_j}\left(\lambda \sum_{k=1}^{3} \frac{\partial v_k}{\partial x_k}\right) + \sum_{i=1}^{3} \frac{\partial}{\partial x_i}\left[\mu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right], \quad j = 1, 2, 3 \tag{2}$$

where ρ is the density, v_1, v_2, and v_3 the Cartesian component of the velocity, p the pressure, $\delta_{ij}, i, j = 1, 2, 3$ the Kronecker delta, λ the second viscosity coefficien, μ the dynamic viscosity coefficient, t the time, and x_1, x_2, and x_3 the Cartesian coordinate axes. Newtonian fluid is assumed.

The above four equations in (1) and (2) have five independent variables ρ, p, v_1, v_2, and v_3, and hence are not closed without a constitutive relation. In the `bulk` package, the constitutive relation (or the equation of state) of choice is

$$K = \rho \frac{\partial p}{\partial \rho}$$

where K is a constant and the bulk modulus. We chose to use the density ρ as the independent variable, and integrate the equation of state to be

$$p = p_0 + K \ln \frac{\rho}{\rho_0} \tag{3}$$

where p_0 and ρ_0 are constants. Substituting (3) into (2) gives

$$\frac{\partial \rho v_j}{\partial t} + \sum_{i=1}^{3} \frac{\partial}{\partial x_i}\left[\rho v_i v_j + \delta_{ij}\left(p_0 + K \ln \frac{\rho}{\rho_0}\right)\right] = \sum_{i=1}^{3} \frac{\partial}{\partial x_i}\left[\delta_{ij}\lambda \sum_{k=1}^{3} \frac{\partial v_k}{\partial x_k} + \mu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right], \quad j = 1, 2, 3 \tag{4}$$

Jacobian Matrices

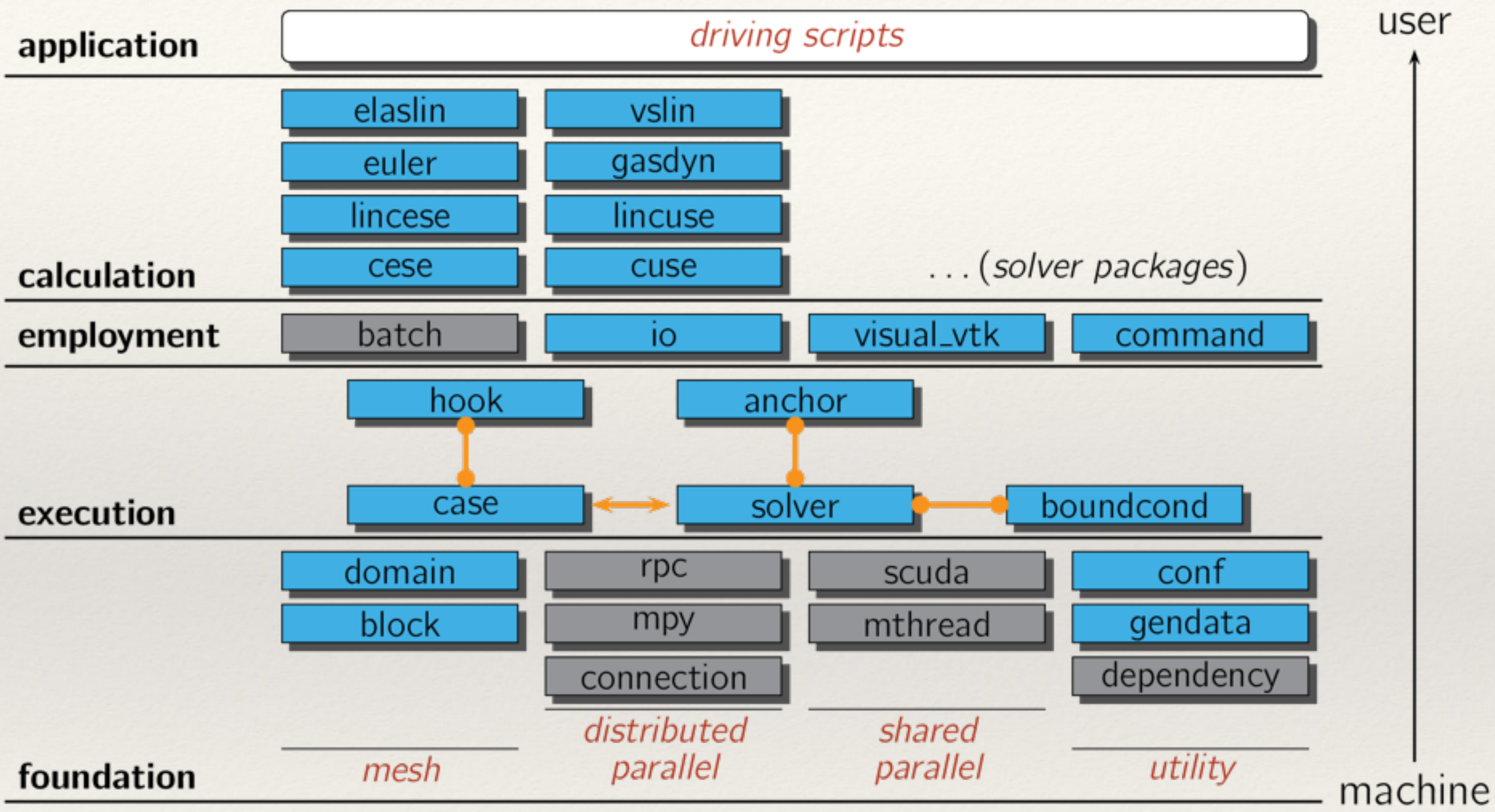
We proceed to analyze the advective part of the governing equations (1) and (4). Define the conservation variables

Authoring, Editing, and Reviewing

- We use Mercurial for version control and BitBucket (https://bitbucket.org/solvcon/solvcon) to host the project and its source code.
- Reviewing the documents goes through the pull-request flow of the code review.
- We treat the reStructuredText source files of the documents like the program source code.

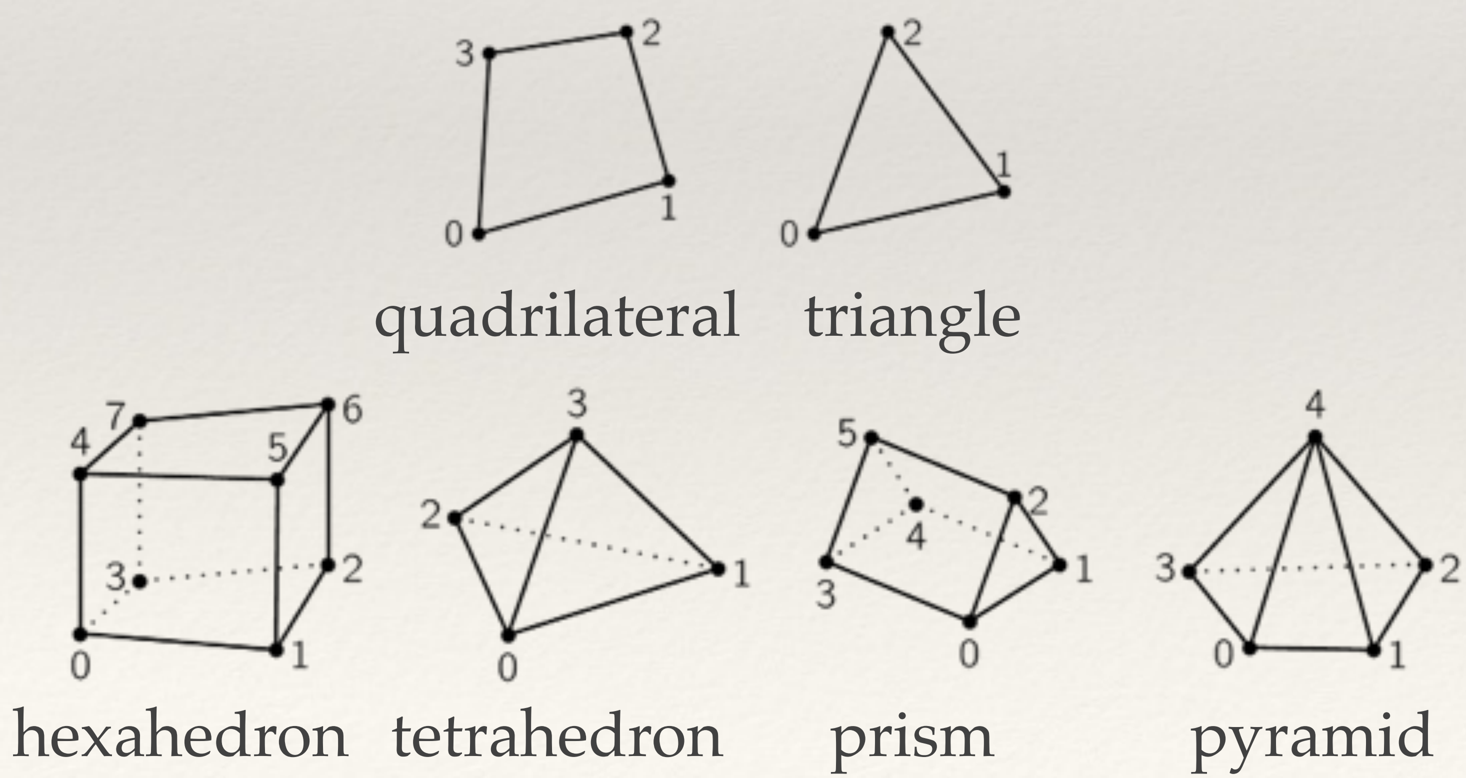
SOLVCON Architecture

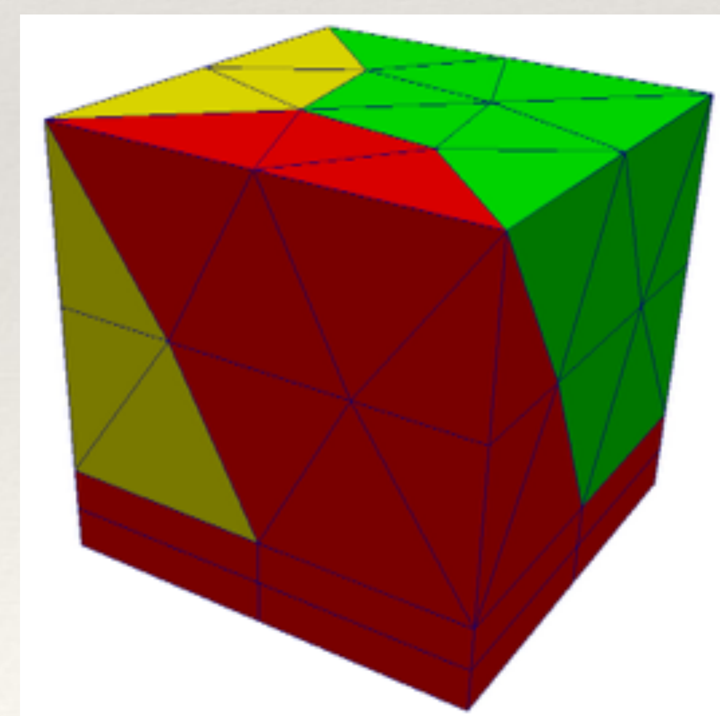
- Fundamentals of SOLVCON:
  1. Unstructured meshes of mixed elements. Data structure affects everything of the system.
  2. The space-time CESE method. It insists a two-nested-loop structure.
  3. Hybrid implementation of Python and C. It's for usability and flexible parallel computing.
- Components are organized into 5 layers (next page).


application
driving scripts
user
elaslin
vslin
euler
gasdyn
lincese
lincuse
cese
cuse
calculation
. . . (solver packages)
employment
batch
io
visual_vtk
command
hook
anchor
case
solver
boundcond
execution
domain
rpc
scuda
conf
block
mpy
mthread
gendata
connection
dependency
distributed parallel
shared parallel
mesh
utility
foundation
machine

Flexible Geometry

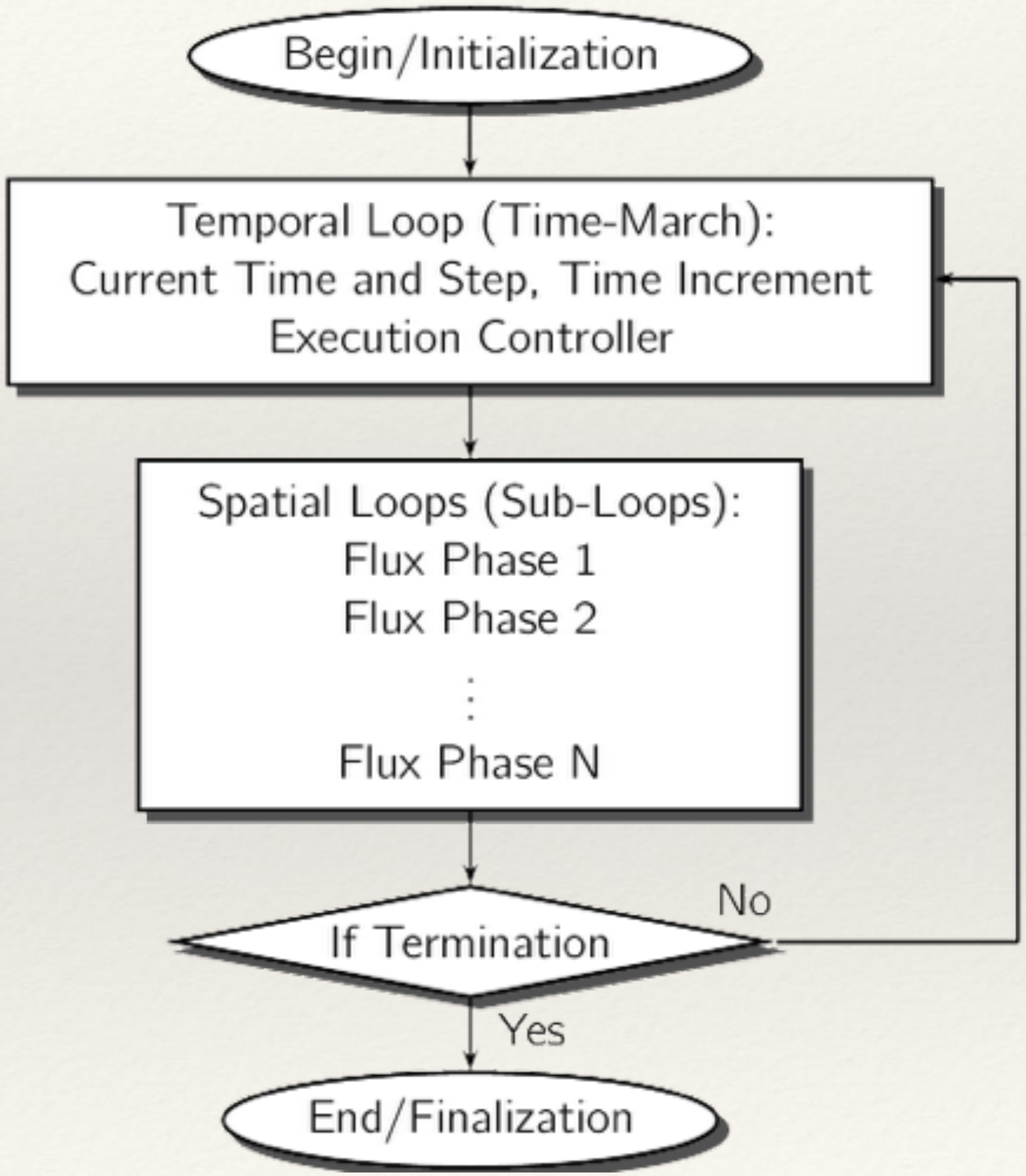
- We want the ability to model the most complex geometry.
- We chose to employ the unstructured meshes of mixed elements for spatial discretization.





Time-Accurate Solutions

- SOLVCON exclusively uses the space-time Conservation Element and Solution Element (CESE) method for time-accurate solutions.
- The pursuit of time-accurate solutions mandates the two-nested-loop structure shown at the right-hand side.
- All facilities can built around the execution flow.



Python: Scripting Is a Must

- There are too many parameters and analyses required.
- No interface is sufficient for the simulations: SOLVCON needs to be a library.
- Controlling a library by using low-level languages like C and C++ is too painful to be realistic.

Python: Architecting

- The outstanding capability to interface to C makes Python an ideal manager for low-level, high-performance C code.
  - C++ works as well but SOLVCON uses only C to avoid the complexity.
  - SOLVCON uses Cython (http://cython.org/) for interfacing.
  - NumPy (http://www.numpy.org/) is the key component for efficient data access through **arrays**.
- The hybrid approach allows us to architect the system with the high-level Python and still have the high performance of C.
  - Ideas can be quickly prototyped with Python. After results get validated, the fast C version can then developed iteratively.

Integrated Testing

- Code without verification can’t be taken as working code, no matter how carefully we wrote it.
- With the testing system, refactoring and quick prototyping become possible.
  - Many research codes don’t have a regression testing system, so while coding the developers can’t be sure how the changes impact the system.
- Two kinds of tests: unit tests and answer tests. Both need automation.

Unit Tests

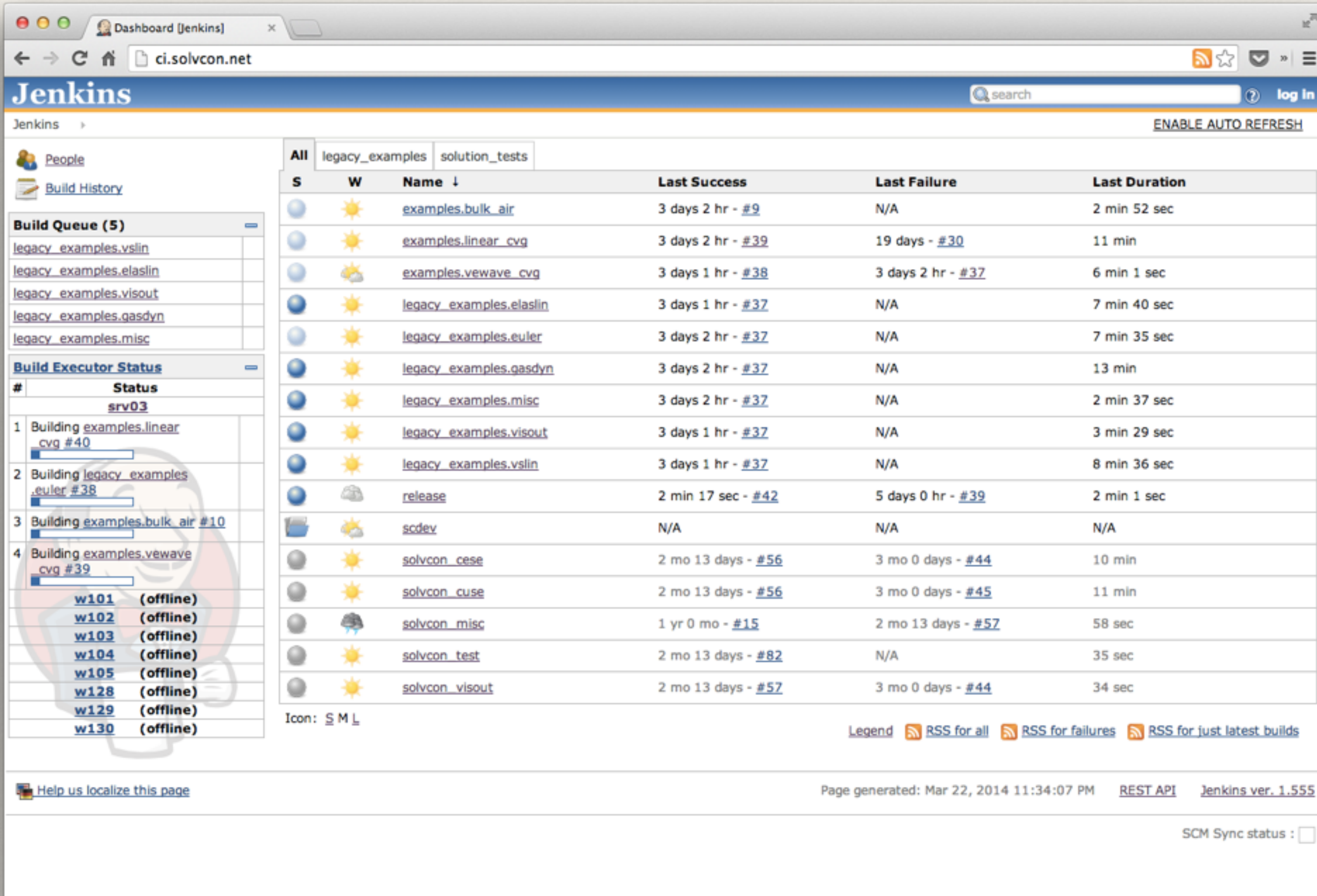
- Unit tests are short and quick, and check for simple output of elementary API behaviors.
- SOLVCON uses standard Python unittest module (http://docs.python.org/2/library/unittest.html).
- nose (https://nose.readthedocs.org/) is used as the automatic test runner.

```
yungyuc@nanoha:~/work/coding/solvcon
$ nosetests
.......................................................................
.......................................................................
.......................................................................
..................................................................
----------------------------------------------------------------------
Ran 279 tests in 5.290s

OK
yungyuc@nanoha:~/work/coding/solvcon
$ 
```

Answer Tests

- Answer tests check for the correctness of simulations. Usually an answer test is a meaningful simulation, and can serve as an example to how to use SOLVCON to simulate other problems.
- Jenkins (http://jenkins-ci.org/) is installed at http://ci.solvcon.net/ to automatically run the answer tests for every commit of SOLVCON in a testing farm.



Dashboard [Jenkins]

ci.solvcon.net

Jenkins

search

log in

Jenkins

ENABLE AUTO REFRESH

People

Build History

Build Queue (5)

legacy_examples.vslin
legacy_examples.elaslin
legacy_examples.visout
legacy_examples.gasdyn
legacy_examples.misc

Build Executor Status

#	Status
	srv03
1	Building examples.linear_cvg #40
2	Building legacy_examples.euler #38
3	Building examples.bulk_air #10
4	Building examples.vewave_cvg #39
	w101 (offline)
	w102 (offline)
	w103 (offline)
	w104 (offline)
	w105 (offline)
	w128 (offline)
	w129 (offline)
	w130 (offline)

All | legacy_examples | solution_tests

S	W	Name ↓	Last Success	Last Failure	Last Duration
		examples.bulk_air	3 days 2 hr - #9	N/A	2 min 52 sec
		examples.linear_cvg	3 days 2 hr - #39	19 days - #30	11 min
		examples.vewave_cvg	3 days 1 hr - #38	3 days 2 hr - #37	6 min 1 sec
		legacy_examples.elaslin	3 days 1 hr - #37	N/A	7 min 40 sec
		legacy_examples.euler	3 days 2 hr - #37	N/A	7 min 35 sec
		legacy_examples.gasdyn	3 days 2 hr - #37	N/A	13 min
		legacy_examples.misc	3 days 2 hr - #37	N/A	2 min 37 sec
		legacy_examples.visout	3 days 1 hr - #37	N/A	3 min 29 sec
		legacy_examples.vslin	3 days 1 hr - #37	N/A	8 min 36 sec
		release	2 min 17 sec - #42	5 days 0 hr - #39	2 min 1 sec
		scdev	N/A	N/A	N/A
		solvcon_cese	2 mo 13 days - #56	3 mo 0 days - #44	10 min
		solvcon_cuse	2 mo 13 days - #56	3 mo 0 days - #45	11 min
		solvcon_misc	1 yr 0 mo - #15	2 mo 13 days - #57	58 sec
		solvcon_test	2 mo 13 days - #82	N/A	35 sec
		solvcon_visout	2 mo 13 days - #57	3 mo 0 days - #44	34 sec

Icon: S M L

Legend RSS for all RSS for failures RSS for just latest builds

Help us localize this page

Page generated: Mar 22, 2014 11:34:07 PM REST API Jenkins ver. 1.555

SCM Sync status :

Current Work: Hydroacoustics

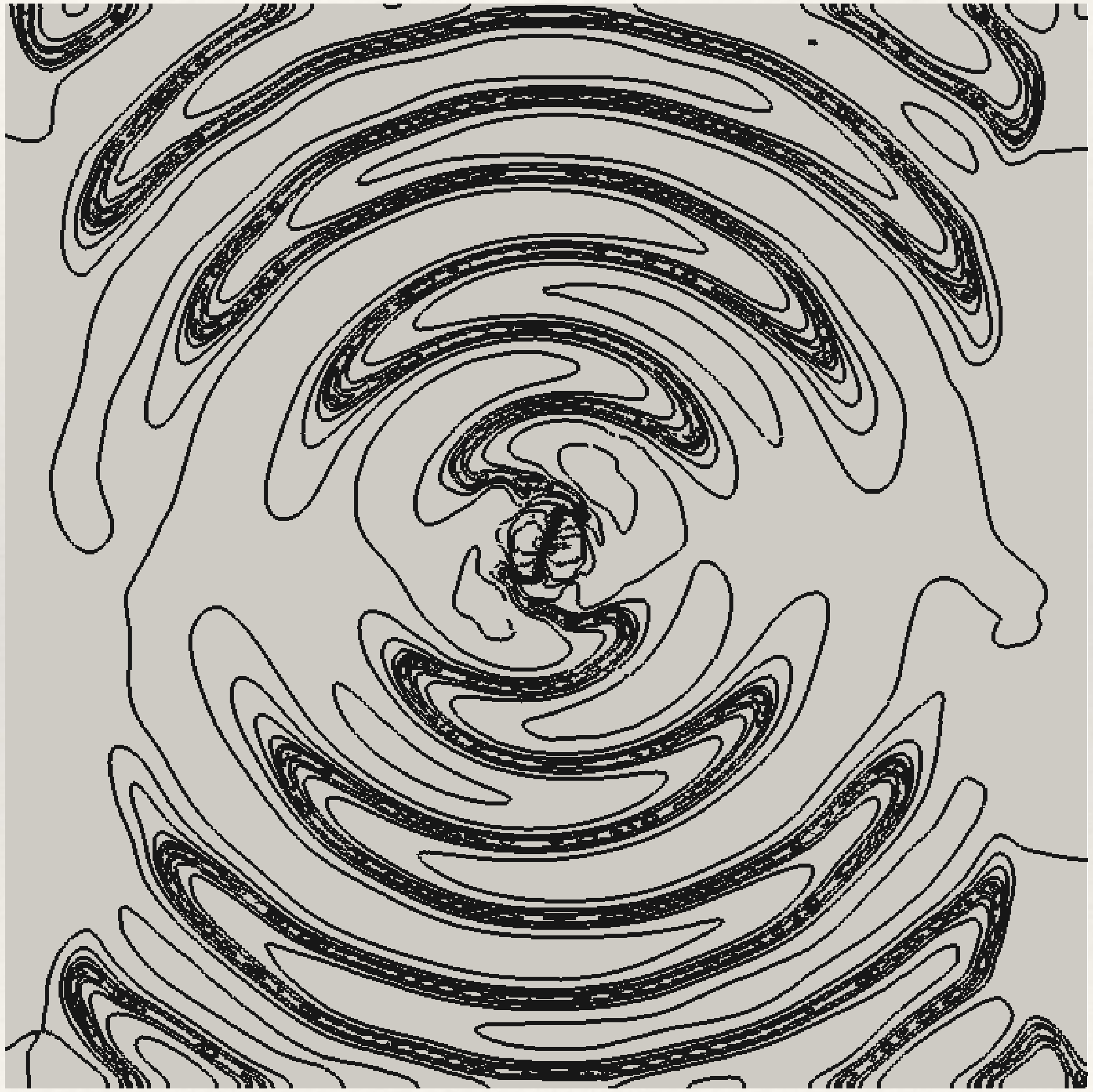
- Governing equations: conservation of mass and momentum.

$$\frac{\partial \rho}{\partial t} + \sum_{i=1}^{3} \frac{\partial \rho v_i}{\partial x_i} = 0$$

$$\frac{\partial \rho v_j}{\partial t} + \sum_{i=1}^{3} \frac{\partial (\rho v_i v_j + \delta_{ij} p)}{\partial x_i} =$$

$$\frac{\partial}{\partial x_j}\left(\lambda \sum_{k=1}^{3} \frac{\partial v_k}{\partial x_k}\right) + \sum_{i=1}^{3} \frac{\partial}{\partial x_i}\left[\mu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)\right], \quad j = 1, 2, 3$$

- Water passes a cylinder at Re = 89,000; decibel contour at RHS.
- Ongoing investigation.



Snapshot of the dB level of the acoustic field

Lin, et. al., TCFD, v28, issue1, pp51-63.

Challenges (Conclusions)

- The simulations were done with older version of SOLVCON; it takes a lot of efforts to upgrade (ongoing).
- Unfamiliarity of the collaborators to the authoring system (Sphinx): http://www.solvcon.net/en/latest/bulk/theory.html
  - Unification of code and document needs extra efforts.
- Feeding back to the main framework.
- Server farm for continuous integration.
- And more. **I'd love to learn your comments.**