# SOLVCON Dev Plan

# 2018Q1

Yung-Yu Chen
yyc@solvcon.net
20180302

# Background

「歌は命」

「歌は希望」

「歌は愛」

「歌は神秘」

「歌は元気」

so is code

# SOLVCON Goal

- A Python-based application framework equipped with C++11-based solvers for continuum mechanics
  - "libmarch" provides high-performance solvers, and loosely couples to the high-level SOLVCON framework
    - easy-to-use (as a research code or a black-box tool) and dependable
    - high-performance and massively-parallel
- What was SOLVCON: A Python-based software framework for calculating conservation laws for multi-physics using parallel computing

# Foundation: Open Source

- SOLVCON is an open-source software (OSS) project.
- SOLVCON uses BSD license.  The only condition for using it (including its source code) is to add the license to your software.
  - Note: SOLVCON is NOT the copy-left “Free Software”, which is cumbersome in many commercial applications.

# History

- 2005-2009: Various predecessors
- 2009-2011: Project starts
- 2011: Python/C (ctypes) hybrid architecture settles
  - Scale up to 2000 cores, 66M elements
  - Multiple physics: Euler equations, anisotropic velocity-stress equations, viscoelastic model, etc.
- 2012-2016: Python/C (Cython) experimentation
  - Strengthen the wrapping layer
- 2016-now: Python/C++11 (pybind11) hybrid architecture
  - C++ template generic programming to replace C macro
  - Create “libmarch” to enable OO to low-level computing kernel
  - Attempt to handle mesh processing and generation

# Immediate Deliverables

- Test 3D Euler solver (march::gas)
- Distill the CESE solver (march::cese)
- (Re-)enable message-passing parallelism (march::mp)
- **Develop granular flow solver march::gran**
- **Develop Navier-Stokes solver (together in march::gas)**

# Working System

- The open-source way: release early, release fast
- Seminars for software development
  - Internal speakers analyze and organize project progress
  - External speakers introduce new information and provide training
- What happens in code remains in code.  Whenever possible, discuss code online using GitHub, emails, etc.
  - F2F meetups are precious and should be used efficiently

# Skills and Tools

- Speak programming languages: Python and C++11
- Speak mathematics
  - Use LaTeX to exchange information / notes
- Automate everything: code and notes
  - Version control: GitHub
  - References: Zotero
  - Data: we need to build a computer farm

# Project Driven Learning

- The only effective way to learn programming: do it and get corrected
- When entering a new area, it takes weeks of hard work to make even a code review
  - The same amount of time to improve to the quality for checking in

# Write Down Everything

- Critical always
- Source code needs to be written down and then executed.  Ideas don't run.
- Research needs to be published.
- Types:
  - Code/paper development and planning: GitHub issue tracking
  - Equations and manuscript: LaTeX files in Git repository
  - Plotting and schematics: Python and PsTricks/LaTeX.  Excellently reproducible but hard to produce at the first time.
- Topics falling outside the organized types go with emails.
- Last resort: verbal communication.  Good for pathfinding, but it's trackless and error-prone.  Write down agenda before and minutes and action items after.

# Distant Projects

- C++11/Python hybrid API for mesh manipulation and generation
- Visualizer (of course basic ones) and Qt-based GUI
- Prototype solving kernels in Python
- Then, use JIT compilation for the Python solving kernels

# Research Team Status Quo

- Computer / server farm (facility)?
- Version control?
- Testing strategy / coverage?
- Code review?
- Continuous integration?