notus
Computational Fluid Dynamics

Notus testing framework

Joris Picot   Stéphane Glockner

I2M
Université de Bordeaux, INP-Bordeaux, CNRS UMR 52 95

jpicot@enscbp.fr, glockner@ipb.fr
http://notus-cfd.org

26 january 2016

Content

1. Brief presentation of Notus

2. Verification and Validation (V&V)
    1. Definitions
    2. Notus tools

3. Porting and Performance (P&P)
    1. Definitions
    2. Notus tools

1. Notus – What is Notus?

Open-source project, started from scratch in 2015

- Modelisation and simulation of **incompressible fluid flows**
- **Massively parallel**
- 2D / 3D Finite Volume methods on staggered grids
- Multiphysics

Intended users – design

- **Mechanical community**:
  - easy to use
  - easy to adapt
  - proven state-of-the-art numerical methods
- **Mathematical community**:
  - open to new numerical schemes
  - fast and efficient framework for comparative and qualitative tests
- Industrials
- Students

What is ***not*** Notus

- A concurrent of
- A commercial tool
- A click button code

1. Notus – Some objectives

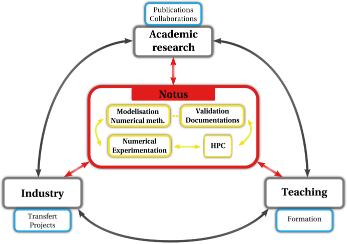
Objectives

- **Rationalise research efforts**
- Take advantage of synergies between Research / Teaching / Industry / HPC
- Provide benchmark methods on identified physical test cases
- Numerical toolbox
- Towards numerical experiments

Means

- A full-featured development environment (git, CMake)
- **Handle parallelism complexities** for easier programming
- **Be ported** on mesocentres (GENCI, PRACE, etc.)
- A thoroughly **validated and documented code**
  - **Verification and Validation** procedures
  - **Non-regression** procedures
  - **Portability and Performance** procedures

→ Become a reference code



2.1. Verification and Validation

Verification

- ***Proves that the discrete approach solves the continuous model precisely***
    - analyses the numerical solution of equations
    - quantifies and reduces of the numerical errors
    - computes spatial and temporal convergence orders

→ **mainly a mathematical and computing process, unlinked to physical problem**

Validation

- ***Analyses the capacity of a model to represent physical phenomena***
    - compares numerical solution to experimental results
    - identifies and quantifies errors and uncertainties of continuous and discrete models, and experience

→ **Accumulation of evidence that the code works!**

Methodology

- Set up a test case with exact solution
- Ensure correctness of the code
- Quantify numerical errors, order of convergence
- Comparison to expectations

Error sources

- Coding bugs
- Numerical stability conditions not satisfied
- Insufficient spatial or temporal convergence
- Non-converging iterative methods
- Rounding errors

Order of convergence

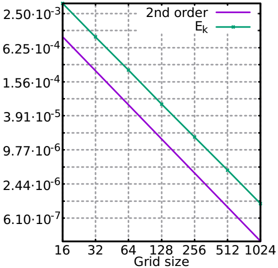
N discrete solutions $f_k (1 \leq k \leq N)$

Hypothesis: smoothed solution in the asymptotic convergence zone

$$f_{h \to 0} = f_k + C h_k^p + \mathrm{O}(h_k^{p+1})$$

$$p_k = \frac{\log\left(\frac{E_k}{E_{k-1}}\right)}{\log\left(\frac{h_k}{h_{k-1}}\right)}$$

where $E_k = f_{exact} - f_k$

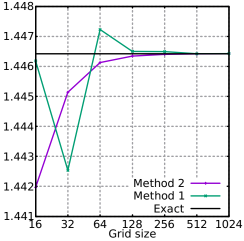


2.1. Verification and Validation – Validation

Test case based on physical phenomena

- In general, no reference solution
- Post processing of physical parameters (velocity plot, Nusselt numbers, lift, drag, etc.)
- Estimation of the influence of numerical parameters (time step, iterative method tolerances, etc.)
- Comparison with experiments and other codes
- Quantify error and uncertainty
  - Use of Richardson extrapolation technique to evaluate convergence order without exact solution

Convergence of physical parameters



- In Notus, tools have been coded to make V&V easier and more automated

1 – Grid convergence tool

- A ”per test case” analysis
- Automated by a Python script

Notus input file (.nts)

List of N grids (.json)

→ Python script →

1. Generates an input file (.nts) for each grid, and activates grid convergence options
2. Runs Notus N times
3. Collects error values and computes convergence orders

- Example of output

```
#      Temperature Linf error      Temperature L2 error        Mean temperature
   16 +3.797157e-03          +nan +4.490870e-03          +nan +1.441965e+00          +nan
   32 +9.624462e-04 +1.980142e+00 +1.288911e-03 +1.800842e+00 +1.445137e+00          +nan
   64 +2.507825e-04 +1.940269e+00 +2.916412e-04 +2.143887e+00 +1.446128e+00 +1.677552e+00
  128 +6.305541e-05 +1.991745e+00 +7.630639e-05 +1.934319e+00 +1.446343e+00 +2.208364e+00
  256 +1.643279e-05 +1.940043e+00 +1.892504e-05 +2.011508e+00 +1.446401e+00 +1.890371e+00
  512 +4.296278e-06 +1.935418e+00 +4.696270e-06 +2.010709e+00 +1.446415e+00 +2.065669e+00
 1024 +1.094218e-06 +1.973188e+00 +1.195573e-06 +1.973813e+00 +1.446418e+00 +1.988532e+00
```

- Save a reference value and set a tolerance for the non-regression tool

2.2. Notus V&V tools

2 – Validation and non-regression tool

- Monitor many test cases: does the code still converges? the same reference value is still obtained?
- Automated by the `notus_validation.sh` bash script

```
$ ./notus_validation.sh -h
Usage : notus_validation.sh [OPTIONS]
-s sequential validation (default: parallel)
-d 2/3 2D or 3D validation (default: 2D and 3D)
-l long validation (default: false); check for special keywords in case.nts and run the case several times
-h print usage

  Test case name                  Validated    Converged    Time iteration           Error
  ibd_laplacian_dirichlet.nts       FAIL
  poiseuille.nts                     OK           OK             356         1.3877787807814457E-17
  poiseuille_periodic.nts            OK           OK              69         1.3877787807814457E-16
  poiseuille_viscosity.nts           OK           OK            2989         0.0000000000000000E+00
  level_set_sheared_2D.nts           NO           N/A            200         3.8200452689984843E-08
  mof_analytic_periodic.nts          OK           N/A            141         2.2204460492503131E-16
  mof_minimization_sheared.nts       OK           N/A           1000         2.2204460492503131E-16
  vof_plic_periodic.nts              OK           N/A            141         3.3306690738754696E-16
  vof_plic_sheared.nts               OK           N/A           1000         3.3306690738754696E-16
  ball_equilibrium.nts               NO           OK            1128         1.4963675386815269E-07
  square_cavity.nts                  OK           OK             291         5.3942093847236805E-14
  driven_cavity.nts                  OK           OK            3449         5.1625370645069779E-15
  dam_break_mof.nts                  OK           N/A             50         2.5313084961453569E-14
  dam_break_vof_plic.nts             OK           N/A             50         3.3556490919295356E-14
  ...
```

- Easy to read: `OK`, `NO`, `FAIL`, etc.
- Short and long **list of V&V test cases files** (quick and full validation)

3.1. Portability and Performance

Portability

- ***Assess that numerical solutions are independant of computational details,*** such as:
  - compiler implementations and versions
  - libraries (MPI, etc.)
  - partitioning (number of processors)
  - computer architectures
- Associated to verification process

Performance

- ***Compare actual scalability to expectations***

Also:

- Determine optimal use of supercomputers (e.g. optimal number of nodes per processor)
- Compare the performance of differents parts of the code
  - partitiong
  - initialization
  - equation preparation
  - linear system solvers
  - post-processings
  - I/O

3.1. Portability and Performance – porting

Compilation

- Notus has been successfully built with the following configurations:
  - GNU compilers (5.2) and Open MPI (1.10)
  - Intel compilers (14.0—15.0) and SGI MPT (2.11) and BullxMPI (1.2.8.3)
  - IBM XL compilers (14.1) and MPI libraries (2.21.1)

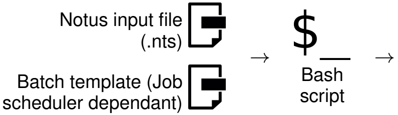
Architectures

- It runs on the following supercomputers and provide the same results up to computer precision (build scripts are provided to install the code):
  - curie at TGCC
  - occigen at CINES
  - turing at IDRIS
  - avakas at MCIA

3.2. Notus P&P tools – performance tools

1 – Performance tool

- A ”per test case” analysis
- Verify weak and strong scalability
- Automated by a bash script



Notus input file (.nts)

Batch template (Job scheduler dependant)

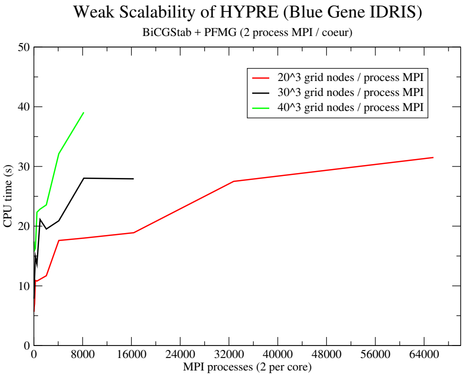
→ $_ Bash script →

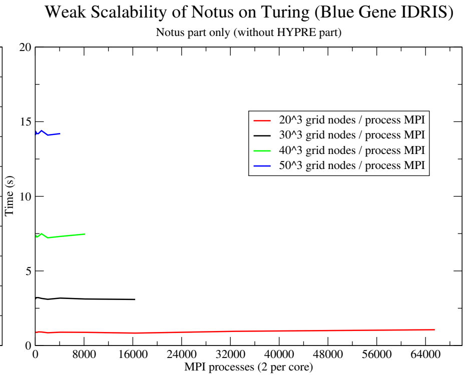
1. Generates input files and submission files
2. Submit jobs
3. The last job collects CPU time values

- Example of output

```
Np          Total        Hypre (velocity)     Hypre (pressure)       Notus
128      0.26000E+01       0.86140E+00         0.94443E+00       0.79417E+00
256      0.29297E+01       0.10660E+01         0.10462E+01       0.81751E+00
512      0.30754E+01       0.11369E+01         0.11025E+01       0.83590E+00
1024     0.38859E+01       0.16025E+01         0.13959E+01       0.88751E+00
2048     0.43207E+01       0.18807E+01         0.15359E+01       0.90404E+00
4096     0.47281E+01       0.22302E+01         0.16268E+01       0.87108E+00
8192     0.65902E+01       0.32613E+01         0.23815E+01       0.94744E+00
```

- Example of output





3.2. Notus – performance tools

2 – Non-regression tool *Under progress*

- All Notus features cannot be activated in one test case
  - → A list of relvelant test cases must be defined
- Save reference times for each test case
  - Reference times will be different according to hardware, software, etc.
- Write a bash script with easy reading: `OK`, `NO`, `FAIL`, etc.

Recap

Notus tools

- ”per test case” processes
  - Generate notus input files according
  - Collect outputs
  - External post processing
  - Present plots and tables
- Non-regression processes
  - Run a given set of test cases
  - Present validation results (`OK`, `NO`, `FAIL`, etc.)
  - Performance tool – *under progress*
    - Store lot of CPU times
    - Store hardware, software information (implementation, version, etc.)

- These tools relies as much as possible on portable technologies