Putting It All Together: Example

PFLOTTRAN

Glenn Hammond
August 8, 2016
Outline

- Introduction to PFLOTRAN
- Code development
  - IDEs
  - Refactoring
  - Testing
  - Debugging
- Open source deployment
  - Software repository
  - Documentation
  - User support
  - QA
- **Petascale** reactive multiphase flow and transport code
- **Open source** license (GNU LGPL 2.0)
- **Object-oriented** Fortran 2003/2008
  - Pointers to procedures
  - Classes (extendable derived types with member procedures)
- Founded upon well-known (supported) open source libraries
  - MPI, PETSc, HDF5, METIS/ParMETIS/CMAKE
- **Demonstrated performance**
  - Maximum # processes: 262,144 (Jaguar supercomputer)
  - Maximum problem size: 3.34 billion degrees of freedom
  - Scales well to over 10K cores
Application of PFLOTRAN

- Nuclear waste disposal
  - Waste Isolation Pilot Plant (WIPP) in Carlsbad, NM
  - DOE Used Fuel Disposition Program
  - SKB Forsmark Spent Fuel Nuclear Waste Repository (Sweden, Amphos21)

- Climate: coupled overland/groundwater flow; CLM
  - Next Generation Ecosystem Experiments (NGEE) Arctic
  - DOE Earth System Modeling (ESM) Program

- Biogeochemical transport modeling
  - U(VI) fate and transport at Hanford 300 Area
  - Hyporheic zone biogeochemical cycling
    - Columbia River, WA, USA
    - East River, CO, USA

- CO₂ sequestration
- Enhanced geothermal energy
- Radioisotope tracers
- Colloid-facilitated transport

Hammond and Lichtner, WRR, 2010
Numerical Methods

- **Spatial discretization**
  - Finite volume (2-point flux default)
  - Structured and unstructured grids
- **Time discretization:** backward Euler
- **Nonlinear solver**
  - Newton-Raphson
  - Line search/damping with custom convergence criteria
- **Linear solver:** direct (LU) or iterative (BiCGStab)
- **Multi-physics coupling**
  - Flow and transport/reaction: sequential
  - Transport and reaction: global implicit
  - Geomechanics and flow/transport: sequential
  - Geophysics and flow/transport: sequential
PFLOTRAN Computing Capability

- High-Performance Computing (HPC)
  - Increasingly mechanistic process models
  - Highly-refined 3D discretizations
  - Massive probabilistic runs
- Open Source Collaboration
  - Leverages a diverse scientific community
  - Sharing among subject matter experts and stakeholders from labs/universities
- Modern Fortran (2003/2008)
  - Domain scientists remain engaged
  - Modular framework for customization
- Leverages Existing Capabilities
  - Meshing, visualization, HPC solvers, etc.
  - Configuration management, testing, and QA

Data Assimilation

Xingyuan Chen, PNNL, 2011
IDEs

- Emacs
- gVim (w/plugins)
- Kate
- Visual Studio
- Xcode

Bottom line: Pick one that best suites your development style.
Benefits of IDEs during Development and Refactoring

- Automated generation of dependency lists
- Bookmarks
- Code completion
- Formatting (auto-indent, block indent, etc.)
- Global search and replace
- Syntax coloring/highlighting and checking
- Integrated compiler
  - Linkage of compile-time errors to lines in source
- Integrated debugger
Testing

- Unit testing
  - Equations of state
    - Viscosity
    - Density
    - Enthalpy / internal energy
    - Saturation pressure
  - Constitutive relations
    - Capillary pressure functions
    - Saturation functions
    - Relative permeability functions
- pFUnit
  - Open source Fortran unit testing framework
Testing (cont.)

- Regression testing (Did the solution change?)
  - Driven by custom python scripting
  - Regression module in PFLOTRAN used to sample solution variables at the end of a simulation
  - Locations (cell ids) are specified in a REGRESSION block

```
REGRESSION
   CELLS_PER_PROCESS 2
   CELLS
      29
/
END
```

- Variables are specified in the OUTPUT block
- .regression file compared to .regression.gold file
- Solutions outside absolute- or relative-change convergence tolerances are flagged.
Regression Entries in Input File

#=========================== regression ============================
REGRESSION
   CELLS_PER_PROCESS 2
   CELLS
       29
/
END

#=========================== output options ========================
OUTPUT
   VARIABLES
       LIQUID_PRESSURE
       LIQUID_SATURATION
       PERMEABILITY_X
       PERMEABILITY_Y
       PERMEABILITY_Z
       POROSITY
       PH
       TOTAL
       TOTAL_SORBED
       KD
       MINERAL_SATURATION_INDEX
/
END
.regression Output File

-- PRESSURE: Liquid Pressure --
  Max: 3.6987012374958E+05
  Min: -2.9546226998033E+04
  Mean: 1.7789073395768E+05
  29: 1.9047483535024E+05
  1: 3.6938752319775E+05
  31: 1.9834550275718E+05

-- RATE: Metatorbernite Rate --
  Max: 0.0000000000000E+00
  Min: 1.9999999998411E-11
  Mean: 2.6666666663803E-12
  29: 0.0000000000000E+00
  1: 0.0000000000000E+00
  31: 0.0000000000000E+00

-- GENERIC: LIQUID VELOCITY [m/d] --
  29: 8.5124089175370E-02 -1.2877090842582E-01 5.2164253201197E-04
  1: 1.7617702348986E-02 -9.6502560583815E-04 -8.0086946328361E-04
  31: 2.5578704112122E-01 2.2363909790982E-02 1.8169543176494E-02

-- SOLUTION: Flow --
  Time (seconds): 2.6570320129395E-02
  Time Steps: 14
  Newton Iterations: 28
  Solver Iterations: 28
  Time Step Cuts: 0
  Solution 2-Norm: 1.8527721282346E+06
  Residual 2-Norm: 3.7495263161587E-13
Testing (cont.)

- Tests can be launched through the PFLOTRAN makefile
  - `make rtest` (regression tests only)
  - `make utest` (unit tests only)
  - `make test` (regression and unit tests)

- Regression tests can be launched separately from the command line within `$PFLOTRAN_DIR/regression_tests`
  
  ```python
  python regression_tests.py <args>
  ```
python regression_tests.py --help

usage: regression_tests.py [-h] [--backtrace] [--advanced]
                        [-c CONFIG_FILES [CONFIG_FILES ...]] [-check-only]
                        [--check-performance] [--debug] [-d]
                        [-e EXECUTABLE] [-list-suites] [-list-tests]
                        [-m MPIEXEC] [-n]
                        [-r [RECURSIVE_SEARCH [RECURSIVE_SEARCH ...]]]
                        [-s SUITES [SUITES ...]] [-t TESTS [TESTS ...]]
                        [--timeout TIMEOUT] [-u]

Run a pflotran regression tests or suite of tests.

optional arguments:
  -h, --help                  show this help message and exit
  --backtrace                 show exception backtraces as extra debugging output
  --advanced                  enable advanced options for developers
  -c CONFIG_FILES [CONFIG_FILES ...], --config-files CONFIG_FILES [CONFIG_FILES ...]
                              test configuration file to use
  --check-only                diff the existing regression files without running
                              pflotran again.
  --check-performance        include the performance metrics ('SOLUTION' blocks) in
                              regression checks.
  --debug                    extra debugging output
  -d, --dry-run              perform a dry run, setup the test commands but don't
                              run them
  -e EXECUTABLE, --executable EXECUTABLE
                              path to executable to use for testing
  --list-suites              print the list of test suites from the config file and
                              exit
...
543.cfg – Regression Test Configuration File

[suites]
standard = 543_flow
    543_flow_dbase
    543_flow_eos_default
    543_flow_eos_constant
    543_flow_eos_exponential
    543_flow_and_tracer
    543_flow_and_tracer_dbase
    543_hanford_srfcplx_base
    543_hanford_srfcplx_base_restart
    543_hanford_srfcplx_base_restart_hdf5
    543_hanford_overwrite_restart
    543_hanford_srfcplx_param
standard_parallel = 543_flow-np8
    543_flow_and_tracer-np8
    543_hanford_srfcplx_param-np8

[default-test-criteria]
# default criteria for all tests, can be overwritten by specific tests
time = 500 percent
generic = 1.0e-12 absolute
concentration = 1.0e-9 relative
discrete = 0 absolute
rate = 1.0e-12 absolute
volume_fraction = 1.0e-12 absolute
pressure = 1.0e-12 relative
saturation = 1.0e-12 absolute

...

[543_flow-np8]
np=8

[543_hanford_srfcplx_param]
generic = 1.0e-12 relative
make test Screen Output

```
[fuji]pflotran-dev/src/pflotran(110): make test
make[1]: Entering directory `/home/gehammo/software/pflotran-dev/src/pflotran/unittests'

Running pflotran unit tests:

Time: 0.001 seconds
OK (38 tests)

make[1]: Leaving directory `/home/gehammo/software/pflotran-dev/src/pflotran/unittests'
make[1]: Entering directory `/home/gehammo/software/pflotran-dev/regression_tests'
/usr/bin/python regression_tests.py -e ./src/pflotran/pflotran --mpiexec /home/gehammo/local/bin/mpiexec - --suite standard standard_parallel - --config-files ascem/batch/batch.cfg ascem/1d/1d-calcite/1d-calcite.cfg'

Test log file: pflotran-tests-2016-07-29_10-16-31.testlog
Running pflotran regression tests:

----------------------------------------------------------------------
----------------------------------------------------------------------
----------------------------------------------------------------------

Regression test summary:
  Total run time: 185.067 [s]
  Total tests : 179
  Tests run : 179
  All tests passed.
```
Example regression test failure

- Perturb critical pressure for water equation of state by \textbf{10 billionths of a percent}

```diff
diff -r f9f01bbf557a src/pflotran/eos_water.F90
--- a/src/pflotran/eos_water.F90   Thu Jul 28 18:59:00 2016 -0700
+++ b/src/pflotran/eos_water.F90   Fri Jul 29 10:31:57 2016 -0700
@@ -893,6 +893,7 @@

tc1 = H2O_CRITICAL_TEMPERATURE ! K
pc1 = H2O_CRITICAL_PRESSURE ! Pa
+ pc1 = pc1 + 1.d-10*H2O_CRITICAL_PRESSURE ! perturb by 1e-10
vc1 = 0.00317d0 ! m^3/kg
utc1 = one/tc1 ! 1/C
upc1 = one/pcc1 ! 1/Pa
```
Running pflotran unit tests:
...F........................................
Time: 0.001 seconds

Failure in: testEOSWater_DensitySTP
  Location: [test_eos water.pf:157]
expected: +998.3234 but found: +998.3234; difference: |+0.4774847E-11| > tolerance:+0.1000000E-15.

FAILURES!!!
Tests run: 38, Failures: 1, Errors: 0

make[1]: Leaving directory `/home/gehammo/software/pflotran-dev/src/pflotran/unit tests'
make[1]: Entering directory `/home/gehammo/software/pflotran-dev/regression_tests'
/usr/bin/python regression_tests.py -e ../src/pflotran/pflotran --mpiexec /home/gehammo/local/bin/mpiexec \
   --suite standard standard_parallel \
   --config-files ascem/batch/batch.cfg ascem/1d/1d-calcite/1d-calcite

Test log file: pflotran-tests-2016-07-29_10-27-50.testlog
Running pflotran regression tests:

..........F.F...FFFFF..F.F........F.....FFFFF..FFFFF.........F....F.F......F.F..FFFFF.F.F....F.F......F.F....
......F..................F........F.............F...........

Regression test summary:
  Total run time: 178.551 [s]
  Total tests : 179
  Tests run : 179
  Failed : 37
pflotran-tests-2016-07-29_10-27-50.testlog

PFLOTRAN Regression Test Log
Date : 2016-07-29_10-27-50
System Info :
   platform : linux2
Test directory :
   /home/gehammo/software/pflotran-dev/regression_tests

PFLOTRAN repository status :
-------------------------------
$ hg parent
changeset: 10149:f9f01bbf557a
tag: tip
user: Glenn Hammond
date: Thu Jul 28 18:59:00 2016 -0700
summary: Modified seepage face BC in TH to prevent thermal conduction when boundary pressure is below the reference pressure (e.g. river stage below cell center).

$ hg status -q
M src/pflotran/eos_water.F90

PETSc information :
-------------------
* WARNING * This information may be incorrect if you have more than one
   version of petsc installed.
   PETSC_DIR : /home/gehammo/software/lib/petsc-git
   PETSC_ARCH : gnu-c-debug
   petsc repository status :
$ git log -1 HEAD
commit 9fc87aa74b00c10f6fbaa6e6828460251b027710
Author: Barry Smith <bsmith@mcs.anl.gov>
Date: Mon Jun 6 15:16:46 2016 -0500

   Add additional information to MATSOLVERMUMPS manual page
543_flow-np8...
  cd /home/gehammo/software/pflotran-dev/regression_tests/default/543
  /home/gehammo/local/bin/mpiexec -np 8 /home/gehammo/software/pflotran-dev/src/pflotran/pflotran -malloc 0 -successful_exit_code 86 -input_prefix 543_flow-np8
  # 543_flow-np8 : run time : 1.31 seconds
diff 543_flow-np8.regression.gold 543_flow-np8.regression
543_flow-np8... passed.

543_hanford_srfcplx_param...
  cd /home/gehammo/software/pflotran-dev/regression_tests/default/543
  /home/gehammo/software/pflotran-dev/src/pflotran/pflotran -malloc 0 -successful_exit_code 86 -input_prefix
543_hanford_srfcplx_param
  # 543_hanford_srfcplx_param : run time : 2.91 seconds
diff 543_hanford_srfcplx_param.regression.gold 543_hanford_srfcplx_param.regression
FAIL: LIQUID VELOCITY [m/d]:1 : 1.084136795e-11 > 1e-12 [relative]
FAIL: LIQUID VELOCITY [m/d]:31 : 7.3779567027e-12 > 1e-12 [relative]
FAIL: LIQUID VELOCITY [m/d]:31 : 1.7611798338e-12 > 1e-12 [relative]
FAIL: LIQUID VELOCITY [m/d]:29 : 2.25552177701e-12 > 1e-12 [relative]
FAIL: LIQUID VELOCITY [m/d]:29 : 1.61796082447e-11 > 1e-12 [relative]
FAIL: UO3.2H2O SI:Min : 4.37393289458e-12 > 1e-12 [relative]
FAIL: UO2(PO3)2 SI:Min : 4.34593859641e-12 > 1e-12 [relative]
FAIL: UO2SO4 SI:Min : 4.32535887832e-12 > 1e-12 [relative]
FAIL: Torbernite SI:Min : 8.7624584403e-12 > 1e-12 [relative]
FAIL: (UO2)3(PO4)2.4H2O SI:Min : 1.3083890044e-11 > 1e-12 [relative]
FAIL: UOF4 SI:Min : 4.3466543516e-12 > 1e-12 [relative]
FAIL: Saleeite SI:Min : 8.72937379374e-12 > 1e-12 [relative]
FAIL: Schoepite SI:Min : 4.37393289458e-12 > 1e-12 [relative]
543_hanford_srfcplx_param... failed.
pflotran-tests-2016-07-29_10-27-50.testlog

543_flow-np8...
  cd /home/gehammo/software/pflotran-dev/regression_tests/default/543
  /home/gehammo/local/bin/mpiexec -np 8 /home/gehammo/software/pflotran-dev/src/pflotran/pflotran -malloc 0 -successful_exit_code 86 -input_prefix 543_flow-np8
  # 543_flow-np8 : run time : 1.31 seconds
  diff 543_flow-np8.regression.gold 543_flow-np8.regression
543_flow-np8... passed.

543_hanford_srfcplx_param...
  cd /home/gehammo/software/pflotran-dev/regression_tests/default/543
  /home/gehammo/local/bin/mpiexec -np 8 /home/gehammo/software/pflotran-dev/src/pflotran/pflotran -malloc 0 -successful_exit_code 86 -input_prefix 543_hanford_srfcplx_param
  # 543_hanford_srfcplx_param : run time : 2.91 seconds
  diff 543_hanford_srfcplx_param.regression.gold 543_hanford_srfcplx_param.regression
FAIL: LIQUID VELOCITY [m/d]:1 : 1.084136795e-11 > 1e-12 [relative]
FAIL: LIQUID VELOCITY [m/d]:31 : 7.3779567027e-12 > 1e-12 [relative]
FAIL: LIQUID VELOCITY [m/d]:31 : 1.7611798338e-12 > 1e-12 [relative]
FAIL: LIQUID VELOCITY [m/d]:29 : 2.25552127701e-12 > 1e-12 [relative]
FAIL: LIQUID VELOCITY [m/d]:29 : 1.61796082447e-11 > 1e-12 [relative]
FAIL: UO3.2H2O SI:Min : 4.37393289458e-12 > 1e-12 [relative]
FAIL: UO2(PO3)2 SI:Min : 4.34539859641e-12 > 1e-12 [relative]
FAIL: UO2SO4 SI:Min : 4.32535887832e-12 > 1e-12 [relative]
FAIL: (UO2)3(PO4)2.4H2O SI:Min : 1.30878004044e-11 > 1e-12 [relative]
FAIL: UO2CO3 SI:Min : 4.36306510613e-12 > 1e-12 [relative]
FAIL: UO3.0.9H2O(alpha) SI:Min : 4.3793289458e-12 > 1e-12 [relative]
FAIL: Torbernite SI:Min : 8.7624584403e-12 > 1e-12 [relative]
FAIL: (UO2)3(PO4)2.4H2O SI:Min : 1.30878004044e-11 > 1e-12 [relative]
FAIL: UO2CO3 SI:Min : 4.36306510613e-12 > 1e-12 [relative]
FAIL: UO3.0.9H2O(alpha) SI:Min : 4.3793289458e-12 > 1e-12 [relative]
FAIL: Metatorbernite SI:Min : 8.7557249827e-12 > 1e-12 [relative]
FAIL: CaUO4 SI:Min : 4.38494539832e-12 > 1e-12 [relative]
FAIL: (UO2)3(PO4)2 SI:Min : 1.30887549659e-11 > 1e-12 [relative]
FAIL: UOF4 SI:Min : 4.34665543516e-12 > 1e-12 [relative]
FAIL: Saleeite SI:Min : 8.72937379374e-12 > 1e-12 [relative]
FAIL: Schoepite SI:Min : 4.37393289458e-12 > 1e-12 [relative]
543_hanford_srfcplx_param... failed.
Buildbot: pflotran.lbl.gov/pbbot/waterfall

Mac is still testing
Buildbot: pflotran.lbl.gov/pbbot/waterfall
Code Coverage

- Steps to generating code coverage statistics with Intel codecov
  1) Load Intel compilers
     - source /opt/intel/bin/compilervars.csh intel64
  2) Instrument files at compile time
     - CC_FLAGS += -prof-gen=srcpos -prof-dir <path>
     - FC_FLAGS += -prof-gen=srcpos -prof-dir <path>
  3) Run simulations
  4) Merge dynamic profile information (*.dyn) files
     - profmerge
  5) Generate code coverage results
     - codecov -spi pgopti.spi -dpi pgopti.dpi
  6) View results in <path>/CODE_COVERAGE.HTML
## Coverage Summary

<table>
<thead>
<tr>
<th></th>
<th>Files</th>
<th>Functions</th>
<th>Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>total</td>
<td>cvrd</td>
<td>uncvd</td>
</tr>
<tr>
<td></td>
<td>231</td>
<td>204</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>142,321</td>
<td>73,632</td>
<td>68,688</td>
</tr>
</tbody>
</table>

## Covered Files

<table>
<thead>
<tr>
<th>Name</th>
<th>Functions</th>
<th>Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>total</td>
<td>cvrd</td>
</tr>
<tr>
<td>auxiliary.F90</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>block_solve.F90</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>characteristic_curves.F90</td>
<td>157</td>
<td>89</td>
</tr>
<tr>
<td>checkpoint.F90</td>
<td>21</td>
<td>20</td>
</tr>
<tr>
<td>co2_span_wagner.F90</td>
<td>30</td>
<td>2</td>
</tr>
<tr>
<td>co2_span_wagner.F90</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>co2eos.F90</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>communicator_structured.F90</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>communicator_unstructured.F90</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>condition.F90</td>
<td>39</td>
<td>35</td>
</tr>
<tr>
<td>condition_control.F90</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>connection.F90</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>convergence.F90</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>coupler.F90</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>data_mediator.F90</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

## Uncovered Files

<table>
<thead>
<tr>
<th>Name</th>
<th>Functions</th>
<th>Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>total</td>
<td>total</td>
</tr>
<tr>
<td>block_tridiag.F90</td>
<td>6</td>
<td>255</td>
</tr>
<tr>
<td>checkpoint_surface.F90</td>
<td>5</td>
<td>263</td>
</tr>
<tr>
<td>co2_span_wagner_spline.F90</td>
<td>2</td>
<td>534</td>
</tr>
<tr>
<td>co2_sw_rtsafe.F90</td>
<td>2</td>
<td>56</td>
</tr>
<tr>
<td>communicator_base.F90</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>dataset_map hdf5.F90</td>
<td>10</td>
<td>503</td>
</tr>
<tr>
<td>gas_eos_mod.F90</td>
<td>5</td>
<td>35</td>
</tr>
<tr>
<td>hydrostatic_multi_phase.F90</td>
<td>10</td>
<td>375</td>
</tr>
<tr>
<td>immis.F90</td>
<td>30</td>
<td>1,965</td>
</tr>
<tr>
<td>immis_aux.F90</td>
<td>7</td>
<td>430</td>
</tr>
<tr>
<td>matrix_buffer.F90</td>
<td>9</td>
<td>196</td>
</tr>
<tr>
<td>miscible.F90</td>
<td>32</td>
<td>2,029</td>
</tr>
<tr>
<td>pm_immis.F90</td>
<td>16</td>
<td>149</td>
</tr>
<tr>
<td>pm_miscible.F90</td>
<td>15</td>
<td>144</td>
</tr>
<tr>
<td>pm_surface_flow.F90</td>
<td>11</td>
<td>102</td>
</tr>
</tbody>
</table>
CODE_COVERAGE.HTML

- Intel codecov
  - Poor coverage
    - dataset_map_hdf5.F90
  - Good coverage
    - material.F90
  - Excellent coverage
    - srcsink_sandbox_wipp_gas.F90

### Uncovered Files

<table>
<thead>
<tr>
<th>Files</th>
<th>Functions</th>
<th>Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>total</td>
<td>cvrd</td>
<td>uncvr</td>
</tr>
<tr>
<td>231</td>
<td>204</td>
<td>27</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Functions</th>
<th>Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>block_tridiag.F90</td>
<td>6</td>
<td>255</td>
</tr>
<tr>
<td>checkpoint_surface.F90</td>
<td>5</td>
<td>263</td>
</tr>
<tr>
<td>co2_span_wagner_spline.F90</td>
<td>2</td>
<td>534</td>
</tr>
<tr>
<td>co2_sw_rtsafe.F90</td>
<td>2</td>
<td>56</td>
</tr>
<tr>
<td>communicator_base.F90</td>
<td>1</td>
<td>17</td>
</tr>
<tr>
<td>dataset_map_hdf5.F90</td>
<td>10</td>
<td>503</td>
</tr>
<tr>
<td>material.F90</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gas_eos_mod.F90</td>
<td>5</td>
<td>35</td>
</tr>
<tr>
<td>hydrostatic_multi_phase.F90</td>
<td>10</td>
<td>375</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Functions</th>
<th>Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>srcsink_sandbox_wipp_gas.F90</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>
dataset_map_hdf5.F90

```
55, !***********************************************************************
56!
57 subroutine DatasetMapHDF5Init(this)
58!!
59! Initializes members of global dataset class
60!!
61! Author: Glenn Hammond
62! Date: 05/29/13
63!
64 implicit none
65
66 class(dataset_map_hdf5_type) :: this
67
68 call DatasetCommonHDF5Init(this)
69
thisHDF5_dataset_name = ''
71 nullify(thisHDF5Mapping)
72 thisHdf5_dims_global = 0
73 thisHdf5_dims_local = 0
74 nullify(thisHdf5Cell_ids)
75 nullify(thisHdf5Cell_ids_local)
76 thisFirst_time = .PETSC_TRUE
77 end subroutine DatasetMapHDF5Init
78
79 !***********************************************************************
80!
81 function DatasetMapHDF5Cast(this)
82!!
83!! Casts a dataset_base_type to dataset_map_hdf5_type
84!!
85!! Date: 05/23/13
86!!
87 implicit none
88
89 class(dataset_base_type), pointer :: this
90
91 class(dataset_map_hdf5_type), pointer :: DatasetMapHDF5Cast
92
93 nullify(DatasetMapHDF5Cast)
94 select type (this)
95 class is (dataset_map_hdf5_type)
96 DatasetMapHDF5Cast = this
97 end select
98 end function DatasetMapHDF5Cast
99```
material.F90

Error messaging
material.F90

Uncovered code blocks
Uncovered subroutine
Error message

srcsink_sandbox_wipp_gas.F90

Error message
Graphical Debuggers (A Must!)

- Debugging with `gdb` or print statements is a thing of the past.
- Use a graphical debugger
  - Allinea DDT
  - RogueWave TotalView
  - Visual Studio
- Memory debugging
  - Intel Inspector
  - Valgrind
Debugging with two Allinea DDT debuggers
OPEN SOURCE DEVELOPMENT
Community vs. Open Source Codes

- A *community* of users grows around a *fit* code.
  - “Community” does not imply “open source”.
  - “Open source” does not imply a “community”.

- Successful open source codes tend to be governed by *benevolent dictators* that address and protect the needs of the entire community while accommodating the desires of individuals.

- The community can drive the code to evolve beyond the original vision (*evolution*).
Advantages of Open Source

- **Encourages collaboration**
  - Development
  - Testing
  - Debugging

- **Transparency** exposes implementation details critical to scientific reproducibility, but excluded by journal publications.

- **More optimal use of funding**
  - Funding is pooled across a diverse set of projects/budgets.
  - What would have been spend on licensing fees can be redirected towards development.
  - Infinite benefit to those who are unfunded.

- The most fit codes tend to survive (**natural selection**).
Potential Disadvantages of Open Source

- Potential time sinks
  - Newbies
  - Lookie-loos
- Care must be taken to avoid exposing the details of unpublished research.
- One’s own capability can be leveraged against oneself by competitors (competing proposals).
- Dissention among the ranks (permanently forked repositories).
- Codes can be licensed as open source, but the source code remains inaccessible (disingenuous licensing).
PFLOTRAN Support Infrastructure

- Mercurial: distributed source control management tool
- Bitbucket: online PFLOTRAN repository
  - hg clone https://bitbucket.org/pflotran/pflotran-dev
  - Source tree
  - Commit logs
  - Wiki
    - Installation instructions
    - Quick guide
    - FAQ (entries motivated by questions on mailing list)
- Pull requests
- Issue tracker
- Buildbot: automated building and testing (regression and unit)
- Google Groups: pflotran-users and pflotran-dev mailing lists
- Google Analytics: tracks behavior on Bitbucket
PFLOTRAN Documentation

- **Current**
  - **User manual** stored in code repository (LaTeX)
  - **Design docs** stored in separate protected repository (LaTeX)
  - Bitbucket wiki for **quick guide** (description of input deck cards)

- **Future**
  - Prototyped
    - Markdown / MkDocs
    - reStructuredTest / Sphinx
  - Planned approach
    - **reStructuredText** and **Sphinx** to generate html and pdf to be hosted on pflotran.org
    - Use converter to adapt in the future (e.g. Pandoc)
Comments on Mercurial Distributed Source Control Management

- **Git vs. Mercurial**
  - Mercurial tends to be more *novice friendly*.
  - Git is clearly more popular. But as long as Mercurial continues to satisfy our needs, *why change?*

- **Functionality**
  - `hg revert`
  - `hg bisect`
  - `hgactivity extension`

# commits from 2007-2014.
LaTeX stored in PFLOTRAN code repository
PFLOTRAN is an open source, state-of-the-art massively parallel subsurface flow and reactive transport code. The code is developed under a GNU LGPL license allowing for third parties to interface proprietary software with the code, however any modifications to the code itself must be documented and remain open source. PFLOTRAN is written in object oriented, free formatted Fortran 2003. The choice of Fortran over C/C++ was based primarily on the need to enlist and preserve tight collaboration with experienced domain scientists, without which PFLOTRAN’s sophisticated process models would not exist.

PFLOTRAN employs parallelization through domain decomposition using the MPI-based PETSc framework with pfotran-dev tracking the developer version of PETSc (i.e. petsc-dev) available through Bitbucket.

**PFLOTRAN Performance**

**Installation Instructions**

Windows
Windows No METIS
Windows Cygwin GNU
Linux
Legacy Build

**Documentation**

Code Development
RADIOACTIVE_DECAY_REACTION Card

Specifies parameters for radioactive decay reaction. This reaction differs from the GENERAL_REACTION in that only one reactant species may be specified with a unit stoichiometry (i.e. the rate is always first order) and the reactant species is decayed in both the aqueous and sorbed phases.

Required Cards:

REACTION <string>
Reaction equation. Only one reactant species may be listed on the left side of the equation (i.e. or on the right side with a negative stoichiometry). The reactant's stoichiometry is fixed at 1.0. The forward rate is applied to that one species as a first order rate constant [1/sec]. Multiple species are supported as daughter products on the right hand side and stoichiometries can be specified.

RATE_CONSTANT or HALF_LIFE (but not both)

Optional Cards:

RATE_CONSTANT <float>
rate constant for 1st-order decay reaction [1/sec]. The rate constant may be calculated from -ln(0.5) / half-life

HALF_LIFE <float>
half life of species [sec]

Note that rate constant or half life units other than sec or 1/sec may be specified.

Examples:
Optional Cards:

RATE_CONSTANT <float>
  rate constant for 1st-order decay reaction [1/sec]. The rate constant may be calculated from \(-\ln(0.5) / \text{half-life}\).

HALF_LIFE <float>
  half life of species [sec].

Note that rate constant or half life units other than sec or 1/sec may be specified.

Examples:

```
RADIOACTIVE_DECAY_REACTION
REACTION A(aq) <-> B(aq)
RATE_CONSTANT 1.7584d-7 ! half life at 0.125 y
/
```

```
CHEMISTRY
PRIMARY_SPECIES
  A(aq)
  B(aq)
  C(aq)
/
... 
RADIOACTIVE_DECAY_REACTION
REACTION A(aq) <-> B(aq)
  ! Calculating forward rate from half-life
  ! rate = -ln(0.5) / half-life [1/sec]
  RATE_CONSTANT 1.75836d-9 ! 1/s  half life = 12.5 yrs
/
```

```
RADIOACTIVE_DECAY_REACTION
REACTION B(aq) <-> C(aq)
RATE_CONSTANT 8.7918d-10 ! 1/s  half life = 25. yrs
/
```

```
RADIOACTIVE_DECAY_REACTION
  ! Note that C(aq) simply decays with no daughter products
REACTION C(aq) <->
HALF_LIFE 5. y
/
... 
```
PFLOTRAN Bitbucket Commit Log

<table>
<thead>
<tr>
<th>Author</th>
<th>Commit</th>
<th>Message</th>
<th>Date</th>
<th>Builds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glenn Hammond</td>
<td>1ea80b8</td>
<td>Moved members of th_auxvar_type that are specific to ic...</td>
<td>2016-05-13</td>
<td></td>
</tr>
<tr>
<td>Jennifer Frede</td>
<td>54de14b</td>
<td>merge</td>
<td>2016-05-12</td>
<td></td>
</tr>
<tr>
<td>Jennifer Frede</td>
<td>52fa5ac</td>
<td>Added new member variables in the waste form which he...</td>
<td>2016-05-12</td>
<td></td>
</tr>
<tr>
<td>Jennifer Frede</td>
<td>aec86b7</td>
<td>Added an update to the fractional dissolution rate in the f...</td>
<td>2016-05-12</td>
<td></td>
</tr>
<tr>
<td>Glenn Hammond</td>
<td>eb8f365</td>
<td>merge</td>
<td>2016-05-12</td>
<td></td>
</tr>
<tr>
<td>Glenn Hammond</td>
<td>65f4ef3</td>
<td>merge</td>
<td>2016-05-11</td>
<td></td>
</tr>
<tr>
<td>Glenn Hammond</td>
<td>02959a8</td>
<td>merge</td>
<td>2016-05-03</td>
<td></td>
</tr>
<tr>
<td>Glenn Hammond</td>
<td>ece265</td>
<td>merge</td>
<td>2016-05-02</td>
<td></td>
</tr>
<tr>
<td>Jennifer Frede</td>
<td>eb7f6ed</td>
<td>Updated printing for when FMDM is called.</td>
<td>2016-05-12</td>
<td></td>
</tr>
<tr>
<td>Jennifer Frede</td>
<td>f9cafa3</td>
<td>Fixed issues with eff_dissolution_rate_equation and move...</td>
<td>2016-05-12</td>
<td></td>
</tr>
<tr>
<td>Jennifer Frede</td>
<td>14336a5</td>
<td>Updated and merged the revert.</td>
<td>2016-05-12</td>
<td></td>
</tr>
<tr>
<td>Jennifer Frede</td>
<td>0639702</td>
<td>Reverted pm_waste_form to last version where update in...</td>
<td>2016-05-12</td>
<td></td>
</tr>
<tr>
<td>Heeho Park</td>
<td>087a6ae</td>
<td>.hignore edited to remove previously added *.chk and *.h5</td>
<td>2016-05-11</td>
<td></td>
</tr>
<tr>
<td>Heeho Park</td>
<td>50e99ff</td>
<td>abaqus2pfotran.py edited comments</td>
<td>2016-05-11</td>
<td></td>
</tr>
<tr>
<td>Heeho Park</td>
<td>266e122</td>
<td>abaqus2pfotran.py generates boundary region snippets i...</td>
<td>2016-05-11</td>
<td></td>
</tr>
<tr>
<td>Heeho Park</td>
<td>b746801</td>
<td>merged</td>
<td>2016-05-11</td>
<td></td>
</tr>
<tr>
<td>Heeho Park</td>
<td>64c14c5</td>
<td>error messages added back on to abaqus2pfotran</td>
<td>2016-05-11</td>
<td></td>
</tr>
</tbody>
</table>
PFLOTRAN Bitbucket Commit

Corrected the waste form linear dissolution rate so that it is first calculated by getting the mass dissolved in the time step using an exponential decay equation given the fractional dissolution rate, and then take the mass and divide it by the timestep length to get the corrected linear rate. Changed when waste form volume gets updated (now in the dissolution routines) so that dV uses the same dt length (occurs in same timestep). Updated the regression files because the physics have changed.
PFLOTRAN Bitbucket Pull Request

Pull requests

#182  MERGED  jennifer_frederick  default  default

outputting mass in a specific region

Overview  Commits  Activity

Author  Jennifer Frederick

Reviewers

Description

- Added a TOTAL_MASS_REGIONS block under the MASS_BALANCE_FILE block under the new OUTPUT block, where the user can list region names. The mass balance file will output the total mass of aqueous, sorbed, and precipitated solute. Read routine implemented and header in mass balance file printed.

- Added the region total mass to mass balance file output by calling PatchGetMassInRegion if the mass balance region list object is associated. At time of input file read, the regions are not yet localized to patches, so another subroutine was made to point the mass balance region to the patch-region after localization. The total mass calculations are based on region cell ids. It seems in parallel, the cell ids are not correct?

- Renamed PatchGetMassInRegion to PatchGetCompMassInRegion so it represents what is getting calculated better (component mass, not mass in general). Changed loop over k out of region%num_cells instead of k out of size(region%cell_ids) in case the

Comments (2)

Glenn Hammond

I hate to do this to you, but I want to propose the following modification. After looking over the dependencies, I don't know that I want output_option_type to be dependent on region_type...it is the whole issue of minimizing the number of tentacles from one object in another. All that we need from region is the list of cell ids. Therefore, I propose that we replace the pointer to a region type in mass_balance_region_type derived type with a pointer to an array of PetscReals. We then point that pointer to the %cell_ids array in the
PFLOTRAN Bitbucket Blame
Welcome to the PFLOTRAN users mailing list.

<table>
<thead>
<tr>
<th>Title</th>
<th>Posts</th>
<th>Views</th>
<th>Author(s)</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>PFLTRAN and ice?</td>
<td>12</td>
<td>69</td>
<td>Roland Hendel +5</td>
<td>Jul 29</td>
</tr>
<tr>
<td>multiple continuum</td>
<td>16</td>
<td>84</td>
<td>Paolo Trinchero +3</td>
<td>Jul 27</td>
</tr>
<tr>
<td>comparison of pfltran and modflow</td>
<td>15</td>
<td>56</td>
<td>Linwei Hu +3</td>
<td>Jul 22</td>
</tr>
<tr>
<td>Functionality: CO2 flow and transport vs. legacy build</td>
<td>3</td>
<td>16</td>
<td>Igal Tsarfis +1</td>
<td>Jul 19</td>
</tr>
<tr>
<td>Output and regression with geomechanics</td>
<td>1</td>
<td>16</td>
<td>Karra, Satish</td>
<td>Jul 5</td>
</tr>
<tr>
<td>[Q] Triangles embedded between adjoining tetrahedra in a hybrid mesh..</td>
<td>14</td>
<td>21</td>
<td>Franz M Krumenacker +3</td>
<td>Jun 30</td>
</tr>
<tr>
<td>Write to file failed</td>
<td>6</td>
<td>27</td>
<td>Andy Ward +1</td>
<td>Jun 29</td>
</tr>
<tr>
<td>Water EOS for high temperature - above critical temperature</td>
<td>4</td>
<td>13</td>
<td>Paolo Orsini +2</td>
<td>Jun 29</td>
</tr>
<tr>
<td>Mineral precipitation/dissolution with prefactors</td>
<td>1</td>
<td>10</td>
<td>Hammond, Glenn E</td>
<td>Jun 23</td>
</tr>
<tr>
<td>please subscribe me to the PFLOTRAN mailing list</td>
<td>1</td>
<td>11</td>
<td>Wissmeier Laurin</td>
<td>Jun 17</td>
</tr>
<tr>
<td>FLOWRATE output</td>
<td>12</td>
<td>24</td>
<td>Romain P. +2</td>
<td>Jun 16</td>
</tr>
<tr>
<td>Running PFLOTRAN on Windows</td>
<td>5</td>
<td>14</td>
<td>Hammond, Glenn E +1</td>
<td>Jun 15</td>
</tr>
</tbody>
</table>
Hits on PFLOTRAN Bitbucket Site in 2015

# hits per day

Total # hits

1 747
PFLOTRAN QA Testing

python-scripted framework under development
PFLOTRAN Verification Testing

- Test cases for WIPP codes (BRAGFLO and NUTS) set up and executed with PFLOTRAN
  - E.g. BRAGFLO Case #8 “Well production at a specified bottom hole pressure”

PFLOTRAN results compared to BRAGFLO and WIPP version of TOUGH2 (TOUGH28W)
QUESTIONS?
MODERN FORTRAN
Why Object-Oriented Fortran 2003/2008?

- Why Fortran?
  - Experienced domain scientists remain engaged
  - Commonality among all domain scientists

- Why object-oriented?
  - Modular data structures
    - Eases code development and debugging – data locality
    - Nesting of processes and data
  - Tree structure enables self-contained simulations

- Why Fortran 2003/2008?
  - Classes (extendable derived types)
    - Member functions
    - Inheritance
  - Pointers to procedures
    - E.g. swapping equations of state

Diagram:
- Simulation
  - Realization
    - Process Model A
    - Process Model B
  - Solver
    - Timestepper
    - Newton Solver
    - Linear Solver
- State Variables
- Parameters
Object-Oriented Fortran

Fortran 77

subroutine X(a,b,c,d,e,f,...) 
...
common/array/a(ncomp,ncell)

do icell = 1, ncell  
  do icomp = 1, ncomp  
      a(icomp,icell) = ...
  enddo
enddo

OO Fortran 90

subroutine X(realization)
...
grid => realization%patch%grid
reaction => realization%reaction
cells => realization%patch...%cells

do icell = 1, grid%ncell  
  do icomp = 1, reaction%ncomp  
      cells(icell)%conc(icomp) = ...
  enddo
enddo
Modern Fortran

Fortran 90

```
select case(eos_type)
  case(WATER)
    call EvaluateWater(p,t)
  case(AIR)
    call EvaluateAir(p,t)
  case(CO2)
    call EvaluateCO2(p,t)
  case(CH4)
    call EvaluateCH4(p,t)
end select
```

Fortran 2003/2008

```
type, extends(eos_base) :: eos_C02
  ...
contains
  procedure :: Evaluate => EvaluateCO2
end type eos_C02

class(eos_C02) :: eos

call eos%Evaluate(p,t)
```

eos = equation of state