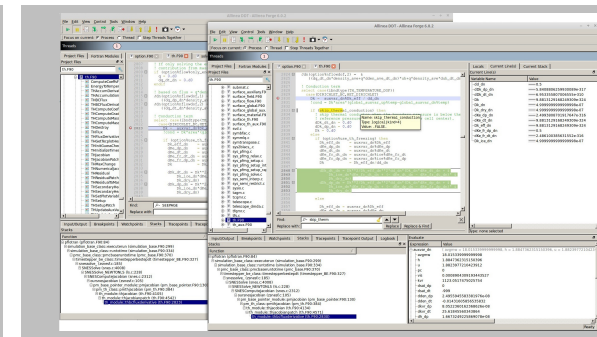


Exceptional service in the national interest



```
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.
```

last build	dev-std-gnu-linux	dev-std-gnu-mac	
	build successful	build successful	
current activity	waiting next in ~ 9 hrs 40 mins at 01:00	waiting next in ~ 9 hrs 40 mins at 01:00	
PDT	changes	dev-std-gnu-linux	dev-std-gnu-mac
01:12:50			
01:12:00			
01:04:54		test pflotran stdio	test pflotran stdio
01:04:36		build pflotran stdio	build pflotran stdio
		petsc stdio	petsc stdio
		set build script path	set build script path
		property 'pflotran_dir' set stdio	property 'pflotran_dir' set stdio
		property changes	property changes



Putting It All Together: Example

PFLOTRAN

Glenn Hammond

August 8, 2016



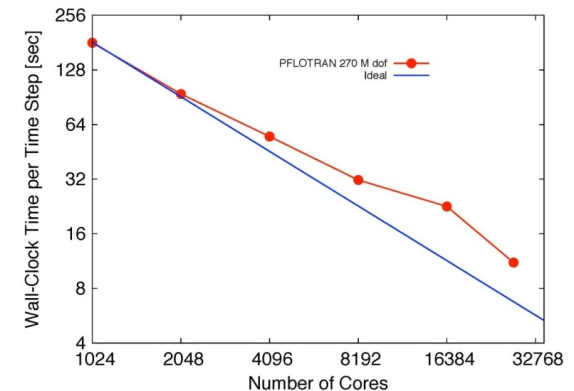
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. SAND2016-7655 PE

Outline

- Introduction to PFLOTRAN
- Code development
 - IDEs
 - Refactoring
 - Testing
 - Debugging
- Open source deployment
 - Software repository
 - Documentation
 - User support
 - QA

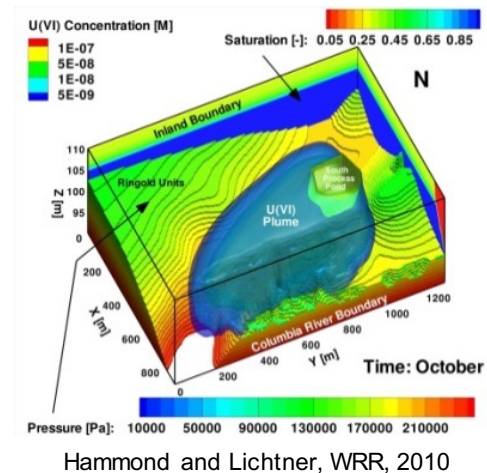
PFLOTRAN

- **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



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

Deep Borehole
Waste Disposal

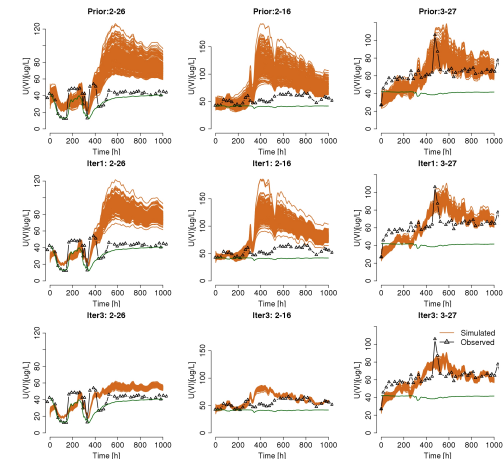


Emily Stein, SNL, 2015

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



PFLOTRAN Development Timeline



2000 2002 2004 2006 2008 2010 2012 2014 2016

Peter Lichtner

Glenn Hammond

Richard Mills

Chuan Lu

Jitu Kumar

Gautam Bisht

Satish Karra

Ben Andre

Nate Collier

Heeho Park

Paolo Orsini

Ayman Alzraiee

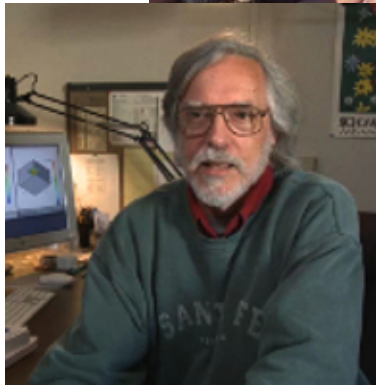
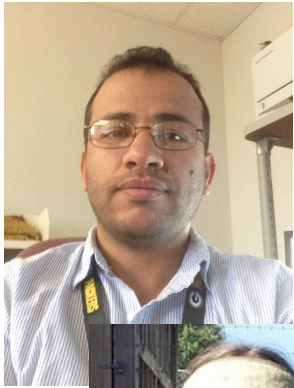
Jennifer Frederick 7

First release

SciDAC-funded rewrite

Process model refactor

PFLOTRAN Developers

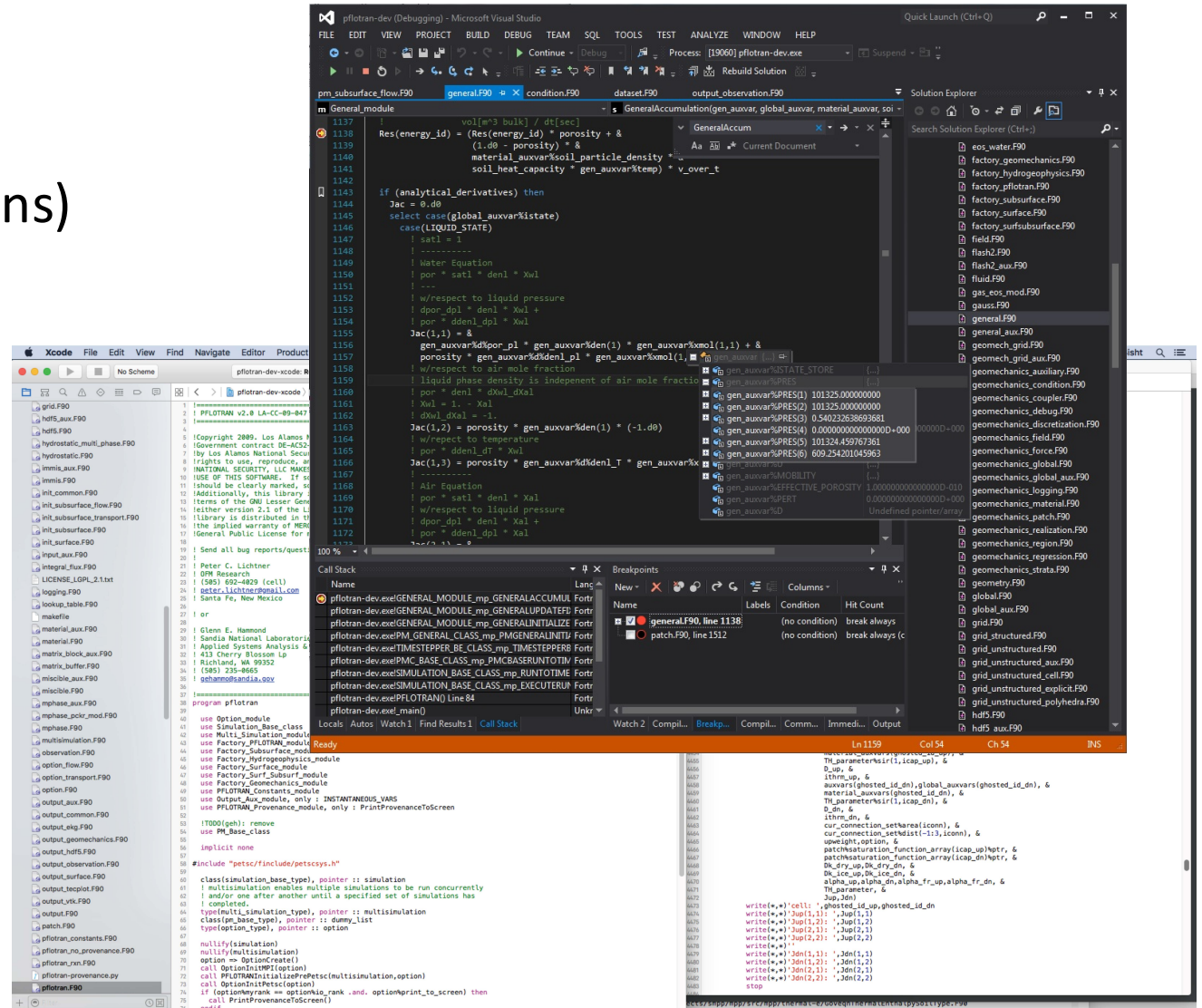


CODE DEVELOPMENT

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 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
-----
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.
```

Testing (cont.)

- Example regression test failure
 - Perturb critical pressure for water equation of state by 10 billionths of a percent

```
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/pc1      ! 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| > tol
erance:+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_test
s'
/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-calc
-----
Test log file : pflotran-tests-2016-07-29_10-27-50.testlog
Running pflotran regression tests :
.....F.F...FFFFF..F..F.....F.....F..FFFF.FFFF.....F.....F...FFFF.
...FF.....F.F...FF.F.....F..F.....FF.....
....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

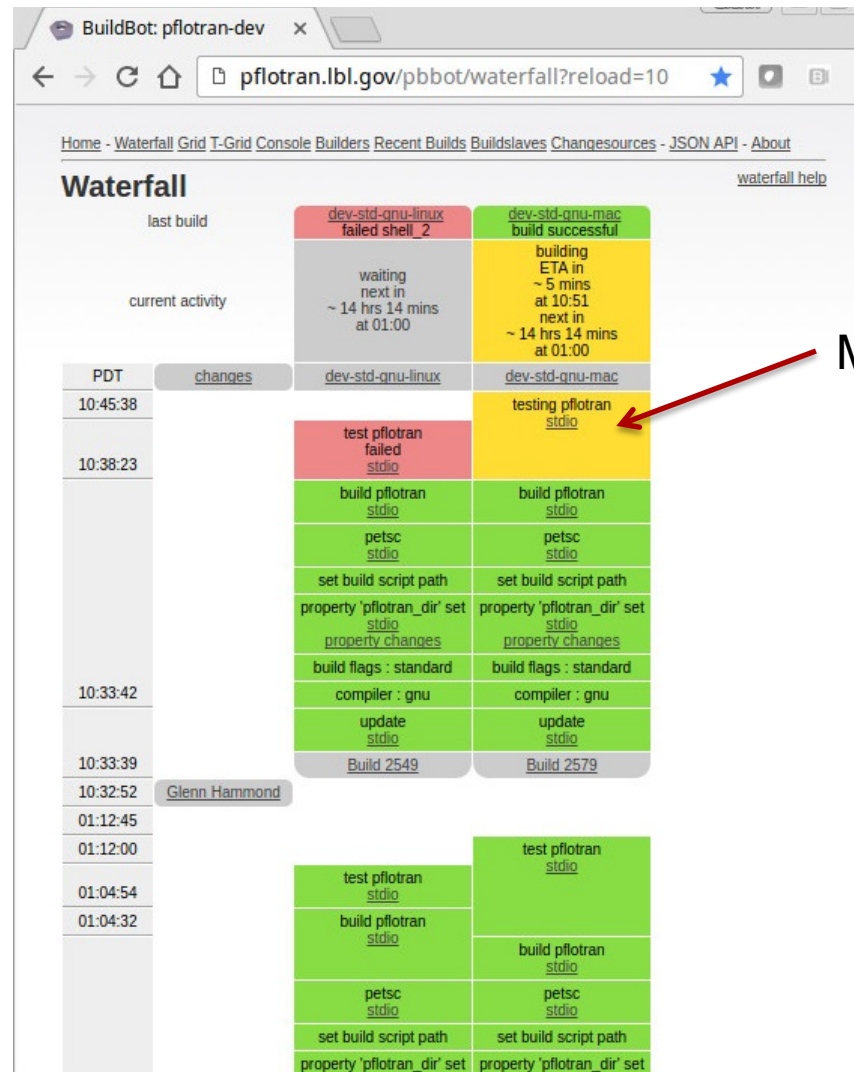
```
-----  
543_flow-np8...  
  cd /home/gehammo/software/pflotran-dev/regression_tests/default/543  
  /home/gehammo/local/bin/mpiexec -np 8 /home/gehammo/software/pflotran-dev/src/pflotran/pflotran -malloc 0 -  
successful_exit_code 86 -input_prefix 543_flow-np8  
  # 543_flow-np8 : run time : 1.31 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 : 1.76111798338e-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: UO2S04 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 : 1.30878004044e-11 > 1e-12 [relative]  
  FAIL: UO2C03 SI:Min : 4.36306510613e-12 > 1e-12 [relative]  
  FAIL: UO3.0.9H2O(alpha) SI:Min : 4.37498338731e-12 > 1e-12 [relative]  
  FAIL: Metatorbernite SI:Min : 8.75578249827e-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.
```

```
-----  
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.
```

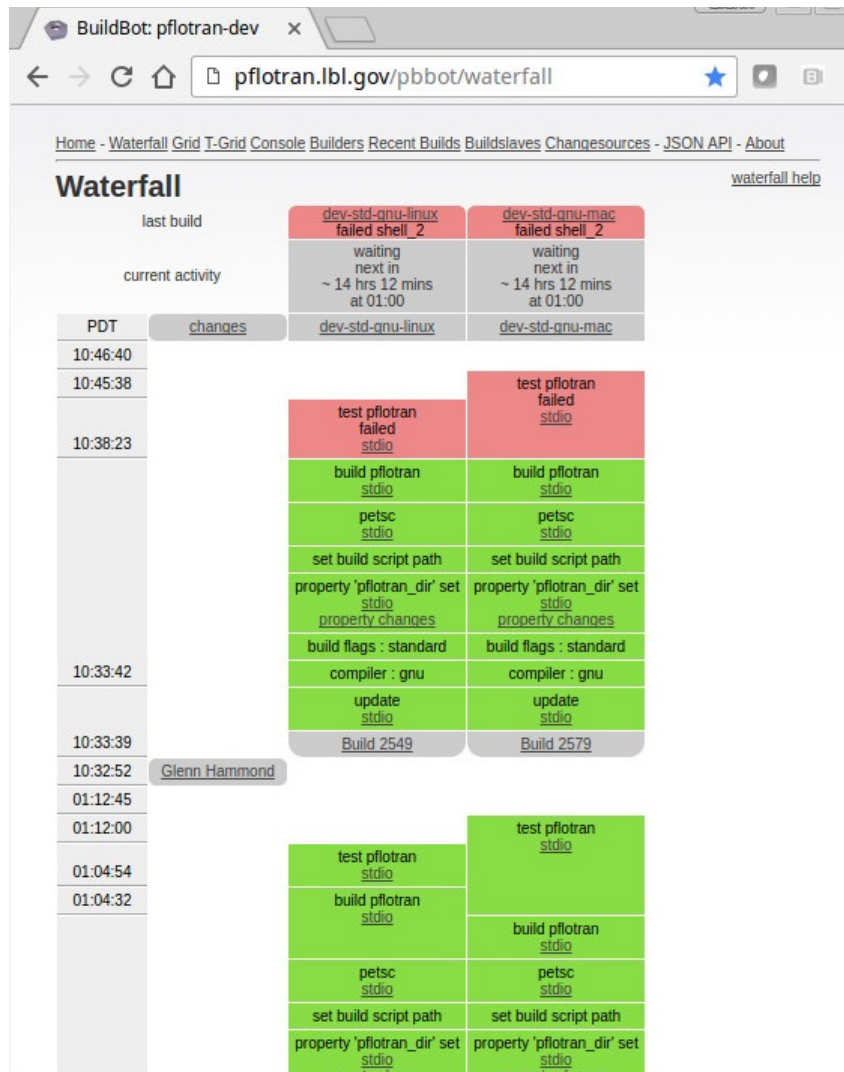
```
-----  
543_hanford_srfcplx_param...  
  cd /home/gehammo/software/pflotran-dev/regression_tests/default/543  
  /home/gehammo/software/pflotran-dev/src/pflotran/pflotran -malloc 0 -successful_exit_code 86 -input_prefix  
543_hanford_srfcplx_param  
  # 543_hanford_srfcplx_param : run time : 2.91 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 : 1.76111798338e-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: UO2S04 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 : 1.30878004044e-11 > 1e-12 [relative]  
  FAIL: UO2C03 SI:Min : 4.36306510613e-12 > 1e-12 [relative]  
  FAIL: UO3.0.9H2O(alpha) SI:Min : 4.37498338731e-12 > 1e-12 [relative]  
  FAIL: Metatorbernite SI:Min : 8.75578249827e-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



BuildBot: pflotran-dev x

pflotran.lbl.gov/pbbot/waterfall

Home - Waterfall Grid T-Grid Console Builders Recent Builds Buildslaves Changesources - JSON API - About

Waterfall

[waterfall help](#)

last build	dev-std-gnu-linux failed shell_2	dev-std-gnu-mac failed shell_2	
current activity	waiting next in ~ 14 hrs 12 mins at 01:00	waiting next in ~ 14 hrs 12 mins at 01:00	
PDT	changes	dev-std-gnu-linux	dev-std-gnu-mac
10:46:40			
10:45:38			
10:38:23	test pflotran failed stdio	test pflotran failed stdio	
	build pflotran stdio	build pflotran stdio	
	petsc stdio	petsc stdio	
	set build script path	set build script path	
	property 'pflotran_dir' set stdio property changes	property 'pflotran_dir' set stdio property changes	
	build flags : standard	build flags : standard	
10:33:42	compiler : gnu	compiler : gnu	
	update stdio	update stdio	
10:33:39	Build 2549	Build 2579	
10:32:52	Glenn Hammond		
01:12:45			
01:12:00			
01:04:54	test pflotran stdio	test pflotran stdio	
01:04:32	build pflotran stdio	build pflotran stdio	
	petsc stdio	petsc stdio	
	set build script path	set build script path	
	property 'pflotran_dir' set stdio property changes	property 'pflotran_dir' set stdio property changes	

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`

CODE_COVERAGE.HTML



generated by
Intel(R) C++/Fortran Compiler
code-coverage tool

Coverage Summary

Files				Functions				Blocks			
total	cvrd	uncvrd	cvrg%	total	cvrd	uncvrd	cvrg%	total	cvrd	uncvrd	cvrg%
231	204	27	88.31	3,090	2,076	1,014	67.18	142,321	73,632	68,689	51.74

Covered Files

Name	Functions			Blocks		
	total	cvrd	cvrg%	total	cvrd	cvrg%
auxiliary.F90	2	2	100.00	17	17	100.00
block_solve.F90	4	3	75.00	208	108	51.92
characteristic_curves.F90	157	89	56.69	3,097	1,120	36.16
checkpoint.F90	21	20	95.24	1,164	761	65.38
co2_span_wagner.F90	30	2	6.67	665	89	13.38
co2_sw.F90	5	1	20.00	368	5	1.36
co2eos.F90	22	2	9.09	234	25	10.68
communicator_structured.F90	9	6	66.67	84	29	34.52
communicator_unstructured.F90	9	6	66.67	81	29	35.80
condition.F90	39	35	89.74	3,402	2,114	62.14
condition_control.F90	6	4	66.67	855	435	50.88
connection.F90	8	7	87.50	213	190	89.20
convergence.F90	3	3	100.00	669	143	21.38
coupler.F90	12	11	91.67	240	200	83.33
data_mediator.F90	3	2	66.67	34	31	91.18

Uncovered Files

Name	Functions	Blocks
	total	total
block_tridiag.F90	6	255
checkpoint_surface.F90	5	263
co2_span_wagner_spline.F90	2	534
co2_sw_rtsafe.F90	2	56
communicator_base.F90	1	17
dataset_map_hdf5.F90	10	503
gas_eos_mod.F90	5	35
hydrostatic_multi_phase.F90	10	375
immis.F90	30	1,965
immis_aux.F90	7	430
matrix_buffer.F90	9	196
miscible.F90	32	2,029
pm_immis.F90	16	149
pm_miscible.F90	15	144
pm_surface_flow.F90	11	102

CODE_COVERAGE.HTML

Uncovered Files

Files				Fun	
total	cvrd	uncvrd	cvrg%	total	cvrd
231	204	27	88.31	3,090	2,076

- Intel codecov
 - Poor coverage
 - [dataset_map_hdf5.F90](#)
 - Good coverage
 - [material.F90](#)
 - Excellent coverage
 - [srcsink_sandbox_wipp_gas.](#)

Name
material.F90

Name	Functions	Blocks
	total	total
block_tridiag.F90	6	255
checkpoint_surface.F90	5	263
co2_span_wagner_spline.F90	2	534
co2_sw_rtsafe.F90	2	56
communicator_base.F90	1	17
dataset_map_hdf5.F90	10	503
gas_eos_mod.F90	5	35
hydrostatic_multi_phase.F90	10	375

Name	Functions			Blocks		
	total	cvrd	cvrg%	total	cvrd	cvrg%
srcsink_sandbox_wipp_gas.F90	5	5	100.00	68	67	98.53

dataset_map_hdf5.F90

Uncovered
subroutines

```
59) ! ***** !
60)
61) subroutine DatasetMapHDF5Init(this)
62) !
63) ! Initializes members of global dataset class
64) !
65) ! Author: Glenn Hammond
66) ! Date: 05/29/13
67) !
68)
69) implicit none
70)
71) class(dataset_map_hdf5_type) :: this
72)
73) call DatasetCommonHDF5Init(this)
74) this%h5_dataset_map_name = ''
75) this%map_filename = ''
76) nullify(this%mapping)
77) this%map_dims_global = 0
78) this%map_dims_local = 0
79) nullify(this%datatocell_ids)
80) nullify(this%cell_ids_local)
81) this%first_time = PETSC_TRUE
82)
83) end subroutine DatasetMapHDF5Init
84)
85) ! ***** !
86)
87) function DatasetMapHDF5Cast(this)
88) !
89) ! Casts a dataset_base_type to dataset_map_hdf5_type
90) !
91) ! Date: 05/03/13
92) !
93)
94) use Dataset_Base_class
95)
96) implicit none
97)
98) class(dataset_base_type), pointer :: this
99)
100) class(dataset_map_hdf5_type), pointer :: DatasetMapHDF5Cast
101)
102) nullify(DatasetMapHDF5Cast)
103) select type (this)
104)   class is (dataset_map_hdf5_type)
105)     DatasetMapHDF5Cast => this
106)   end select
107)
108) end function DatasetMapHDF5Cast
109)
```

material.F90

Error messaging

```
897) ! check to ensure that an id is not duplicated
898) error_flag = PETSC_FALSE
899) do i = 1, max_external_id
900)   if (id_count(i) > 1) then
901)     write(string,") i
902)     option%io_buffer = 'Material ID ' // trim(adjustl(string)) // &
903)       ' is duplicated in input file.'
904)     call printMsg(option)
905)     error_flag = PETSC_TRUE
906)   endif
907) enddo
908)
909) deallocate(id_count)
910)
911) if (error_flag) then
912)   option%io_buffer = 'Duplicate Material IDs.'
913)   call printErrMsg(option)
914) endif
915)
916) ! ensure unique material names
917) error_flag = PETSC_FALSE
918) do i = 1, size(array)
919)   if (associated(array(i)%ptr)) then
920)     length1 = len_trim(array(i)%ptr$name)
921)     do j = 1, i-1
922)       if (associated(array(j)%ptr)) then
923)         length2 = len_trim(array(j)%ptr$name)
924)         if (length1 /= length2) cycle
925)         if (StringCompare(array(i)%ptr$name,array(j)%ptr$name,length1)) then
926)           option%io_buffer = 'Material name "' // &
927)             trim(adjustl(array(i)%ptr$name)) // &
928)               '" is duplicated in input file.'
929)           call printMsg(option)
930)           error_flag = PETSC_TRUE
931)         endif
932)       endif
933)     enddo
934)   endif
935) enddo
936)
937) if (error_flag) then
938)   option%io_buffer = 'Duplicate Material names.'
939)   call printErrMsg(option)
940) endif
941)
```

material.F90

Uncovered
code blocks

```
1501) select case(ivar)
1502)   case(SOIL_COMPRESSIBILITY)
1503)     do ghosted_id=1, Material%num_aux
1504)       Material%auxvars(ghosted_id)% &
1505)         soil_properties(soil_compressibility_index) = vec_loc_p(ghosted_id)
1506)     enddo
1507)   case(SOIL_REFERENCE_PRESSURE)
1508)     do ghosted_id=1, Material%num_aux
1509)       Material%auxvars(ghosted_id)% &
1510)         soil_properties(soil_reference_pressure_index) = vec_loc_p(ghosted_id)
1511)     enddo
1512)   case(VOLUME)
1513)     do ghosted_id=1, Material%num_aux
1514)       Material%auxvars(ghosted_id)%volume = vec_loc_p(ghosted_id)
1515)     enddo
1516)   case(POROSITY)
1517)     select case(isubvar)
1518)       case(POROSITY_CURRENT)
1519)         do ghosted_id=1, Material%num_aux
1520)           Material%auxvars(ghosted_id)%porosity = vec_loc_p(ghosted_id)
1521)         enddo
1522)       case(POROSITY_MINERAL)
1523)         do ghosted_id=1, Material%num_aux
1524)           Material%auxvars(ghosted_id)%porosity_base = vec_loc_p(ghosted_id)
1525)         enddo
1526)     end select
1527)   case(TORTUOSITY)
1528)     do ghosted_id=1, Material%num_aux
1529)       Material%auxvars(ghosted_id)%tortuosity = vec_loc_p(ghosted_id)
1530)     enddo
1531)   case(PERMEABILITY_X)
1532)     do ghosted_id=1, Material%num_aux
1533)       Material%auxvars(ghosted_id)%permeability(perm_xx_index) = &
1534)         vec_loc_p(ghosted_id)
1535)     enddo
1536)   case(PERMEABILITY_Y)
1537)     do ghosted_id=1, Material%num_aux
1538)       Material%auxvars(ghosted_id)%permeability(perm_yy_index) = &
1539)         vec_loc_p(ghosted_id)
1540)     enddo
1541)   case(PERMEABILITY_Z)
1542)     do ghosted_id=1, Material%num_aux
1543)       Material%auxvars(ghosted_id)%permeability(perm_zz_index) = &
1544)         vec_loc_p(ghosted_id)
1545)     enddo
1546)   case(PERMEABILITY_XY)
1547)     do ghosted_id=1, Material%num_aux
1548)       Material%auxvars(ghosted_id)%permeability(perm_xy_index) = &
1549)         vec_loc_p(ghosted_id)
1550)     enddo
1551)   case(PERMEABILITY_YZ)
1552)     do ghosted_id=1, Material%num_aux
1553)       Material%auxvars(ghosted_id)%permeability(perm_yz_index) = &
1554)         vec_loc_p(ghosted_id)
1555)     enddo
1556)   case(PERMEABILITY_XZ)
1557)     do ghosted_id=1, Material%num_aux
1558)       Material%auxvars(ghosted_id)%permeability(perm_xz_index) = &
1559)         vec_loc_p(ghosted_id)
1560)     enddo
1561) end select
```


material.F90

Uncovered
subroutine

```
1816) subroutine MaterialUpdatePorosity(Material,global_auxvars,porosity_loc)
1817) !
1818) ! Gets values of material auxvar data using a vector.
1819) !
1820) ! Author: Glenn Hammond
1821) ! Date: 01/09/14
1822) !
1823)
1824) use Variables_module
1825) use Global_Aux_module
1826)
1827) implicit none
1828)
1829) #include "petsc/finclude/petscvec.h"
1830) #include "petsc/finclude/petscvec.h90"
1831)
1832) type(material_type) :: Material ! from realization%patch%aux%Material
1833) type(global_auxvar_type) :: global_auxvars(:)
1834) Vec :: porosity_loc
1835)
1836) PetscReal, pointer :: porosity_loc_p(:)
1837) class(material_auxvar_type), pointer :: material_auxvars(:)
1838) PetscInt :: ghosted_id
1839) PetscReal :: compressed_porosity
1840) PetscReal :: dcompressed_porosity_dp
1841) PetscErrorCode :: ierr
1842)
1843) if (soil_compressibility_index > 0) then
1844) material_auxvars => Material%auxvars
1845) call VecGetArrayReadF90(porosity_loc,porosity_loc_p,ierr);CHKERRQ(ierr)
1846) do ghosted_id = 1, Material%num_aux
1847) material_auxvars(ghosted_id)%porosity = porosity_loc_p(ghosted_id)
1848) call MaterialCompressSoil(material_auxvars(ghosted_id), &
1849) maxval(global_auxvars(ghosted_id)%pres), &
1850) compressed_porosity,dcompressed_porosity_dp)
1851) material_auxvars(ghosted_id)%porosity = compressed_porosity
1852) material_auxvars(ghosted_id)%dporosity_dp = dcompressed_porosity_dp
1853) enddo
1854) call VecRestoreArrayReadF90(porosity_loc,porosity_loc_p, &
1855) ierr);CHKERRQ(ierr)
1856) endif
1857)
1858) end subroutine MaterialUpdatePorosity
```

srcsink_sandbox_wipp_gas.F90

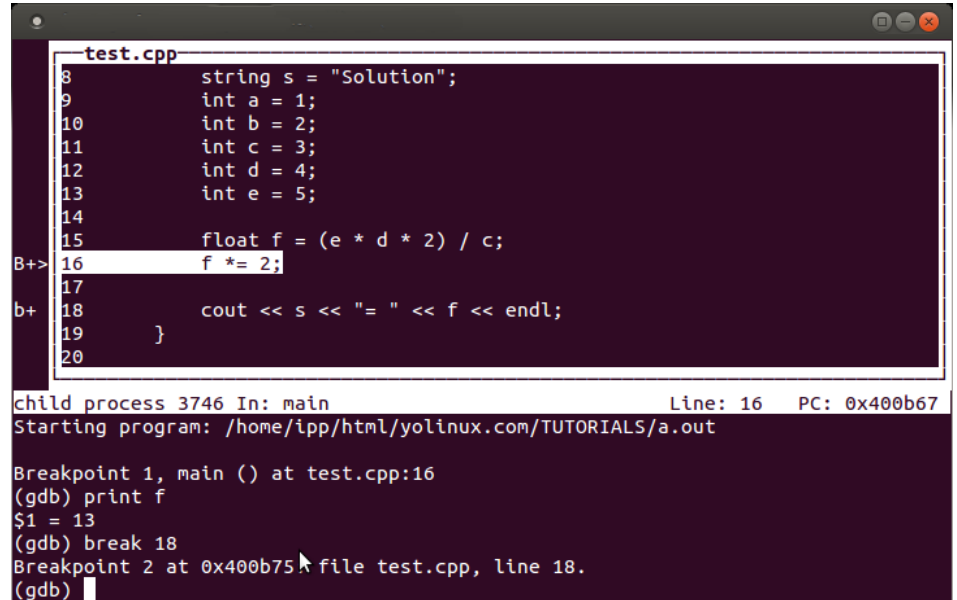
```
64) subroutine WIPPGasGenerationRead(this,input,option)
65) !
66) ! Reads input deck for WIPP gas generation src/sink parameters
67) !
68) ! Author: Glenn Hammond, Edit: Heeho Park
69) ! Date: 04/11/14, 05/15/14
70) !
71)
72) use Option_module
73) use String_module
74) use Input_Aux_module
75) use Units_module, only : UnitsConvertToInternal
76)
77) implicit none
78)
79) class(srcsink_sandbox_wipp_gas_type) :: this
80) type(input_type), pointer :: input
81) type(option_type) :: option
82)
83) PetscInt :: i
84) character(len=MAXWORDLENGTH) :: word, internal_units
85) PetscBool :: found
86)
87) do
88)   call InputReadPflotranString(input,option)
89)   if (InputError(input)) exit
90)   if (InputCheckExit(input,option)) exit
91)
92)   call InputReadWord(input,option,word,PETSC_TRUE)
93)   call InputErrorMsg(input,option,'keyword', &
94)     'SRCSINK_SANDBOX,WIPP')
95)   call StringToUpper(word)
96)
97)   call SSSandboxBaseSelectCase(this,input,option,word,found)
98)   if (found) cycle
99)
100)  select case(trim(word))
101)  case('INUNDATED_CORROSION_RATE')
102)    internal_units = 'unitless/sec'
103)    call InputReadDouble(input,option,this%inundated_corrosion_rate)
104)    call InputDefaultMsg(input,option,'inundated_corrosion_rate')
105)  case('INUNDATED_DEGRADATION_RATE')
106)    internal_units = 'unitless/sec'
107)    call InputReadDouble(input,option,this%inundated_degradation_rate)
108)    call InputDefaultMsg(input,option,'inundated_degradation_rate')
109)  case('HUMID_CORROSION_FACTOR')
110)    call InputReadDouble(input,option,this%humid_corrosion_factor)
111)    call InputDefaultMsg(input,option,'humid_corrosion_factor')
112)  case('HUMID_DEGREADATION_FACTOR')
113)    call InputReadDouble(input,option,this%humid_degradation_factor)
114)    call InputDefaultMsg(input,option,'humid_degradation_factor')
115)  case('H2_FE_RATIO')
116)    call InputReadDouble(input,option,this%h2_fe_ratio)
117)    call InputDefaultMsg(input,option,'h2_fe_ratio')
118)  case('H2_CH2O_RATIO')
119)    call InputReadDouble(input,option,this%h2_ch2o_ratio)
120)    call InputDefaultMsg(input,option,'h2_ch2o_ratio')
121)  case default
122)    call InputKeywordUnrecognized(word, &
123)      'SRCSINK_SANDBOX,WIPP-GAS_GENERATION',option)
124)  end select
125) enddo
126)
127) end subroutine WIPPGasGenerationRead
```

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

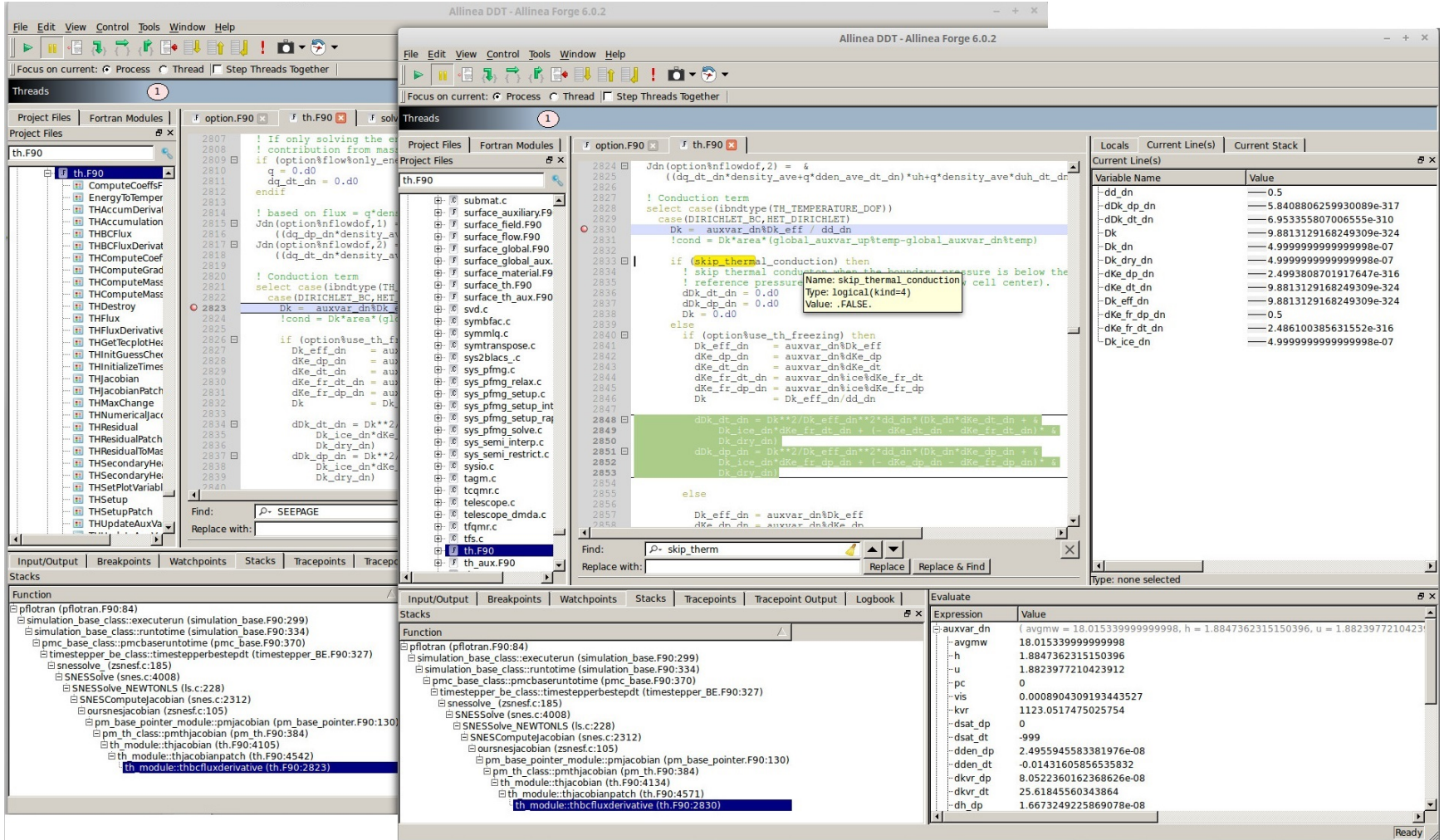


```
test.cpp
8      string s = "Solution";
9      int a = 1;
10     int b = 2;
11     int c = 3;
12     int d = 4;
13     int e = 5;
14
15     float f = (e * d * 2) / c;
B+> 16     f *= 2;
17
b+  18     cout << s << " = " << f << endl;
19     }
20

child process 3746 In: main                               Line: 16   PC: 0x400b67
Starting program: /home/ipp/html/yolinux.com/TUTORIALS/a.out

Breakpoint 1, main () at test.cpp:16
(gdb) print f
$1 = 13
(gdb) break 18
Breakpoint 2 at 0x400b75 file test.cpp, line 18.
(gdb) █
```

Debugging with two Allinea DDT debuggers



The image shows two instances of the Allinea DDT - Allinea Forge 6.0.2 debugger. The left instance shows the 'option.F90' file with a search for 'SEEPAGE'. The right instance shows the 'th.F90' file with a search for 'skip_therm'. Both instances show a 'Locals' window with a list of variables and their values.

Locals Window (Left Instance):

Variable Name	Value
-dd_dn	0.05
-dDk_dp_dn	5.8408806259930089e-317
-dDk_dt_dn	6.953355807006555e-310
-Dk	9.8813129168249309e-324
-Dk_dn	4.999999999999999e-07
-Dk_dry_dn	4.999999999999999e-07
-dKe_dp_dn	2.4993808701917647e-316
-dKe_dt_dn	9.8813129168249309e-324
-dKe_eff_dn	9.8813129168249309e-324
-dKe_fr_dp_dn	0.5
-dKe_fr_dt_dn	2.486100385631552e-316
-Dk_ice_dn	4.999999999999999e-07

Locals Window (Right Instance):

Variable Name	Value
auxvar_dn	(avgmw = 18.015330999999999, h = 1.8847362315150396, u = 1.8823977210423)
avgmw	18.015330999999999
h	1.8847362315150396
u	1.8823977210423912
pc	0
vis	0.0008904309193443527
kvr	1123.0517475025754
dsat_dp	0
dsat_dt	-999
dden_dp	2.49559458381976e-08
dden_dt	-0.01431605856535832
dkvr_dp	8.0522360162368626e-08
dkvr_dt	25.61845560343864
dh_dp	1.6673249225869078e-08

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 Sandia National Laboratories

- 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

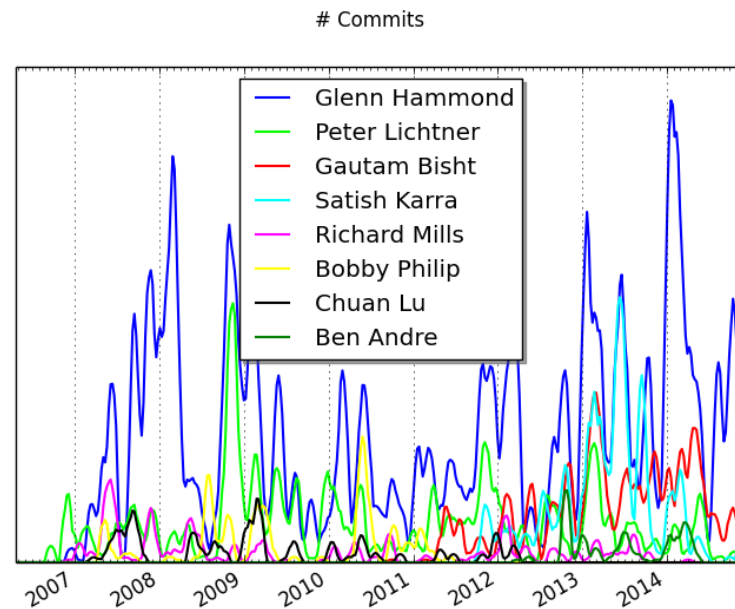
- Mercurial: distributed source control management tool
- Bitbucket: online PFLOTRAN repository
 - hg clone <https://bitbucket.org/pflotran/pflotran-dev>
 - Source tree
 - Commit logs
 - Wiki
 - Installation instructions
 - Quick guide
 - FAQ (entries motivated by questions on mailing list)
 - Pull requests
 - 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
 - reStructuredText / 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.

PFLOTRAN User Manual

PFLOTRAN

PFLOTRAN User Manual

A Massively Parallel Reactive Flow and Transport Model for Describing Surface and Subsurface Processes

Peter C. Lichtner^α, Glenn E. Hammond^β, Chuan Lu^σ,
Satish Karra^δ, Gautam Bisht^γ, Benjamin Andre^ζ,
Richard Mills^ε, Jitu Kumar^ρ

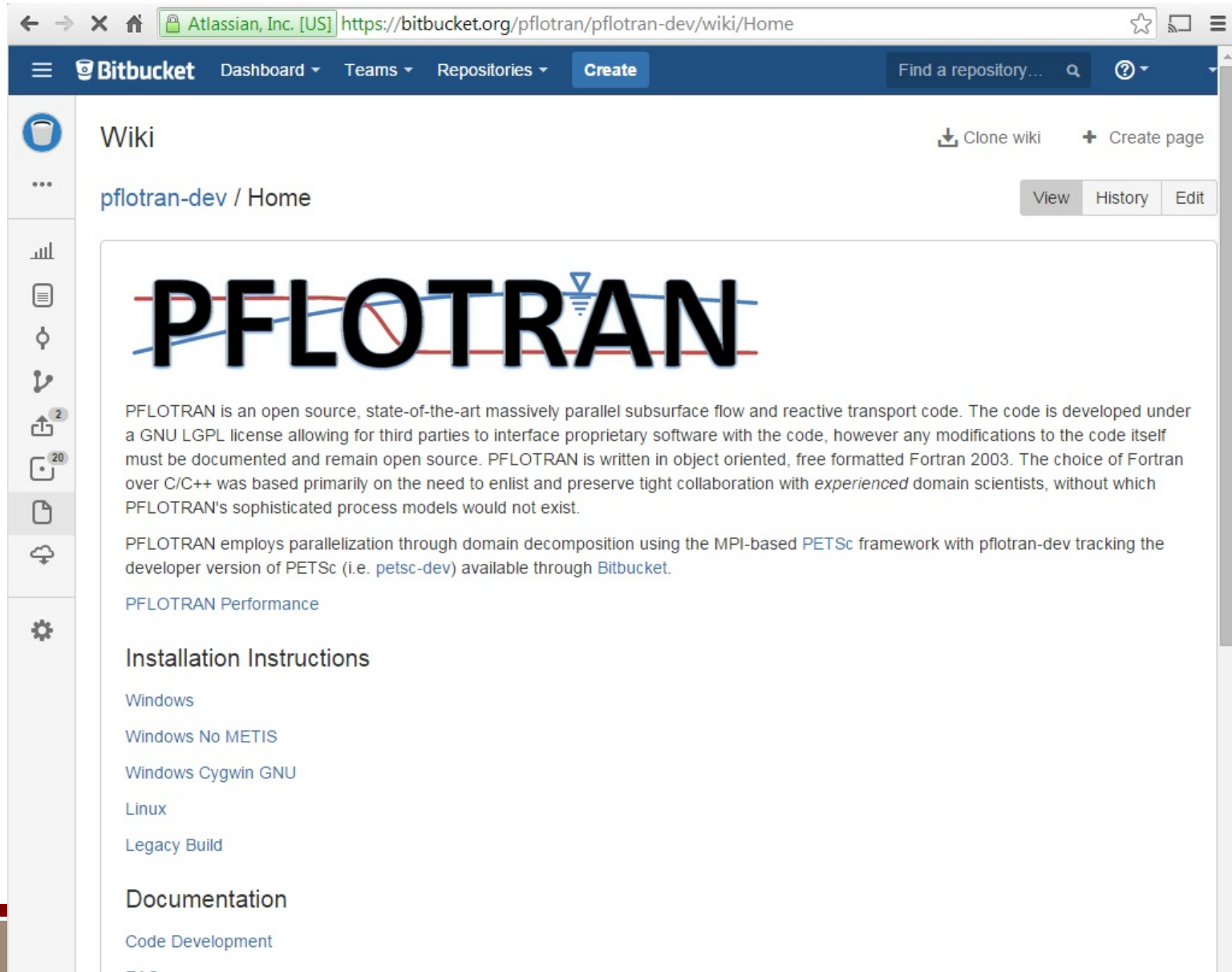
^α OFM Research peter.lichtner@gmail.com
^β SNL gehammo@sandia.gov
^σ IHEG luchuanen@163.com
^δ LANL satkarra@lanl.gov
^γ LBNL gbisht@lbl.gov
^ζ NCAR andre@ucar.edu
^ε Intel rtm@eecs.utk.edu
^ρ ORNL jitu1503@gmail.com

August 1, 2016



LaTeX stored in
PFLOTRAN
code repository

PFLOTRAN Bitbucket Site



The screenshot shows a web browser window displaying the PFLOTRAN Bitbucket Wiki page. The browser's address bar shows the URL <https://bitbucket.org/pflotran/pflotran-dev/wiki/Home>. The Bitbucket navigation bar includes links for Dashboard, Teams, Repositories, and a Create button. The main content area features the PFLOTRAN logo, a description of the software as an open-source massively parallel subsurface flow and reactive transport code, and a list of installation instructions for various operating systems and environments.

Wiki Clone wiki Create page

pflotran-dev / Home View History Edit

PFLOTRAN

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 pflotran-dev tracking the developer version of PETSc (i.e. `petsc-dev`) available through Bitbucket.

PFLOTRAN Performance

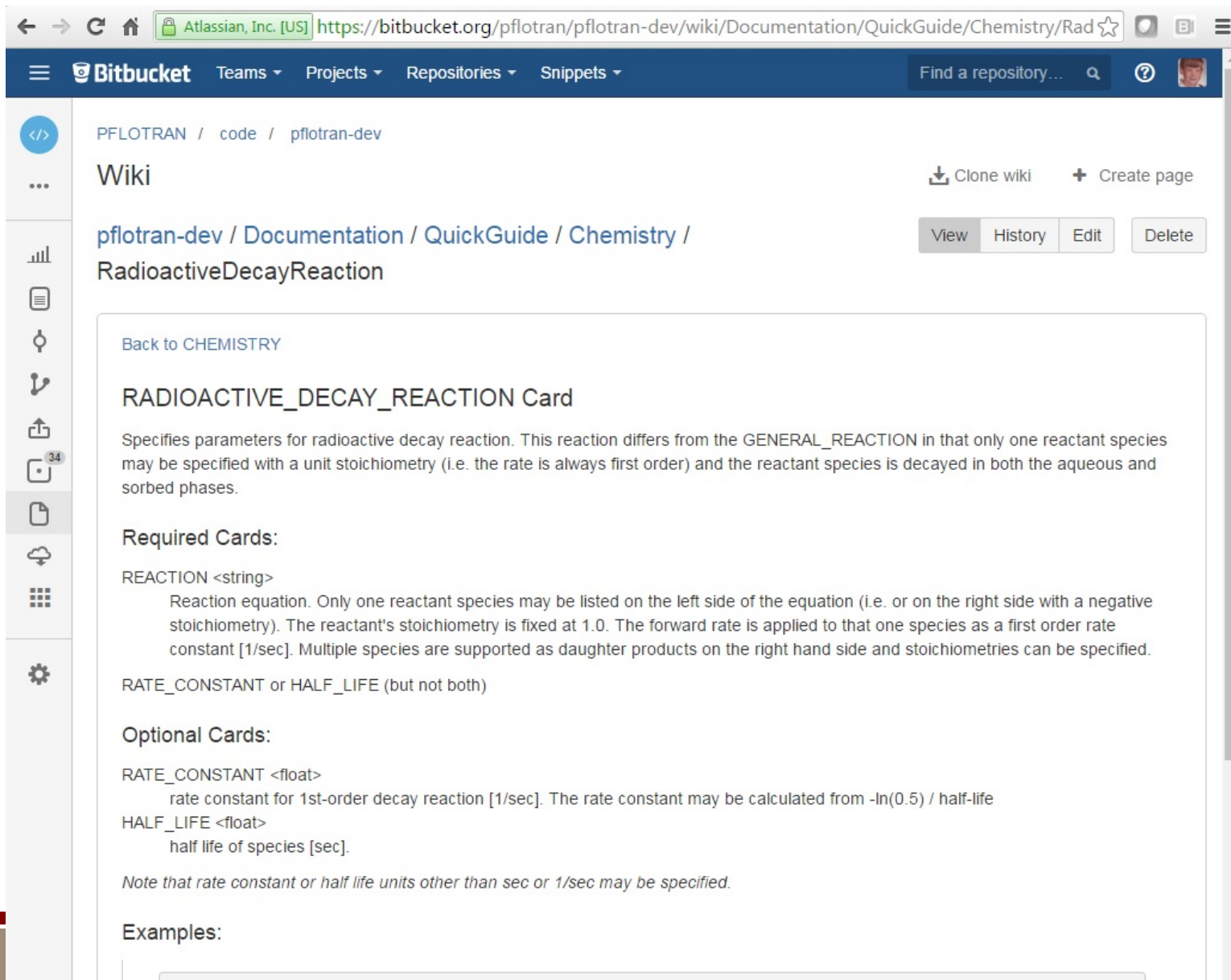
Installation Instructions

- [Windows](#)
- [Windows No METIS](#)
- [Windows Cygwin GNU](#)
- [Linux](#)
- [Legacy Build](#)

Documentation

- [Code Development](#)

PFLOTRAN Quick Guide



The screenshot shows a web browser displaying a Bitbucket wiki page. The address bar shows the URL: <https://bitbucket.org/pflotran/pflotran-dev/wiki/Documentation/QuickGuide/Chemistry/Rad>. The page title is "RadioactiveDecayReaction". The content includes a "Back to CHEMISTRY" link, a section for "RADIOACTIVE_DECAY_REACTION Card", a description of the reaction parameters, "Required Cards" (REACTION), and "Optional Cards" (RATE_CONSTANT and HALF_LIFE). A note at the bottom states: "Note that rate constant or half life units other than sec or 1/sec may be specified." The page also features navigation buttons like "View", "History", "Edit", and "Delete", and a sidebar with various icons.

PFLOTRAN / code / pflotran-dev

Wiki

Clone wiki Create page

View History Edit Delete

Back to CHEMISTRY

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 $-\ln(0.5) / \text{half-life}$

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:

PFLOTRAN Quick Guide (cont.)



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].

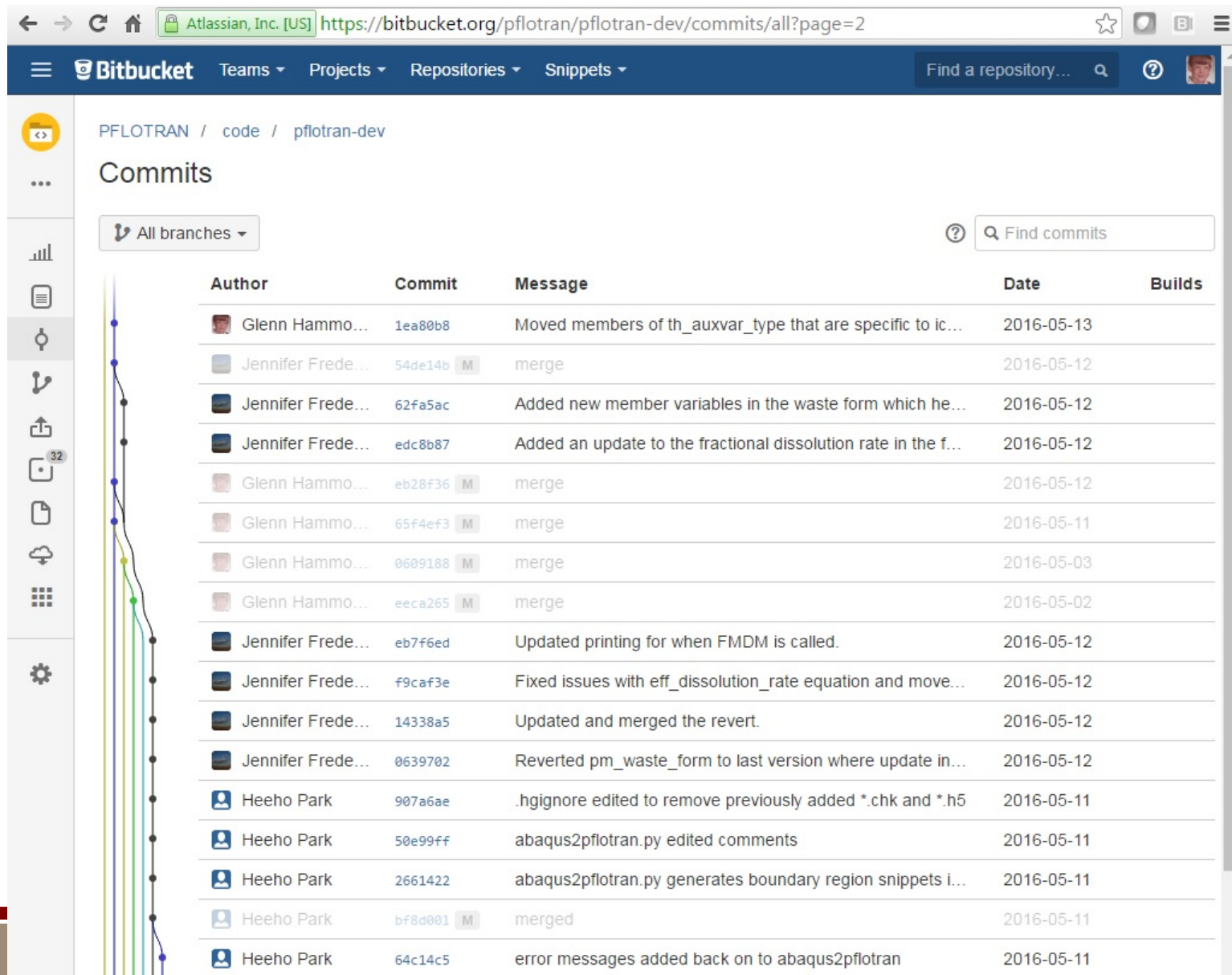
Note that rate constant or half life units other than sec or 1/sec may be specified.

Examples:

```
RADIOACTIVE_DECAY_REACTION
REACTION Tracer <-> Tracer2
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
/
...
/
```

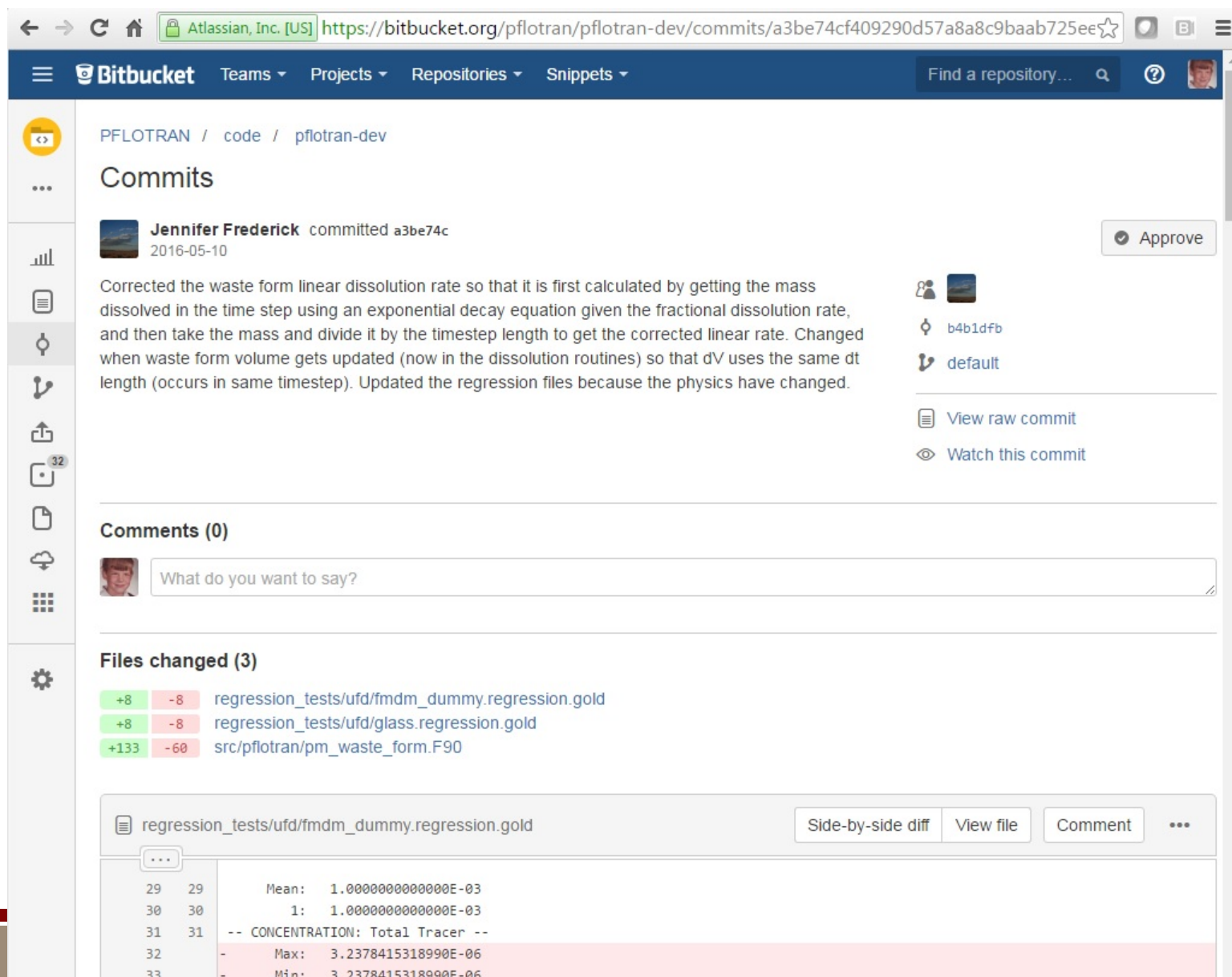
PFLOTRAN Bitbucket Commit Log



The screenshot shows the Bitbucket web interface for the PFLOTRAN repository. The browser address bar shows the URL: <https://bitbucket.org/pflotran/pflotran-dev/commits/all?page=2>. The page title is "Commits" and it shows a list of 20 commit entries. On the left side, there is a vertical navigation menu with icons for repository overview, commit history, and other actions. A commit history graph is visible on the left, showing the progression of branches over time. The commit list includes columns for Author, Commit ID, Message, Date, and Builds.

Author	Commit	Message	Date	Builds
Glenn Hammo...	1ea80b8	Moved members of th_auxvar_type that are specific to ic...	2016-05-13	
Jennifer Frede...	54de14b M	merge	2016-05-12	
Jennifer Frede...	62fa5ac	Added new member variables in the waste form which he...	2016-05-12	
Jennifer Frede...	edc8b87	Added an update to the fractional dissolution rate in the f...	2016-05-12	
Glenn Hammo...	eb28f36 M	merge	2016-05-12	
Glenn Hammo...	65f4ef3 M	merge	2016-05-11	
Glenn Hammo...	0609188 M	merge	2016-05-03	
Glenn Hammo...	eeeca265 M	merge	2016-05-02	
Jennifer Frede...	eb7f6ed	Updated printing for when FMDM is called.	2016-05-12	
Jennifer Frede...	f9caf3e	Fixed issues with eff_dissolution_rate equation and move...	2016-05-12	
Jennifer Frede...	14338a5	Updated and merged the revert.	2016-05-12	
Jennifer Frede...	0639702	Reverted pm_waste_form to last version where update in...	2016-05-12	
Heeho Park	907a6ae	.hgignore edited to remove previously added *.chk and *.h5	2016-05-11	
Heeho Park	50e99ff	abaqus2pflotran.py edited comments	2016-05-11	
Heeho Park	2661422	abaqus2pflotran.py generates boundary region snippets i...	2016-05-11	
Heeho Park	b78d001 M	merged	2016-05-11	
Heeho Park	64c14c5	error messages added back on to abaqus2pflotran	2016-05-11	

PFLOTRAN Bitbucket Commit




← → ↻ 🏠 Atlassian, Inc. [US] <https://bitbucket.org/pflotran/pflotran-dev/commits/a3be74cf409290d57a8a8c9baab725ee> 🔖


☰ **Bitbucket** Teams ▾ Projects ▾ Repositories ▾ Snippets ▾ Find a repository... 🔍 ? 👤

🔗 **PFLOTRAN** / code / pflotran-dev

Commits


 **Jennifer Frederick** committed a3be74c 2016-05-10 Approve

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.

 b4b1dfb
default

[View raw commit](#)
[Watch this commit](#)

Comments (0)

 What do you want to say?

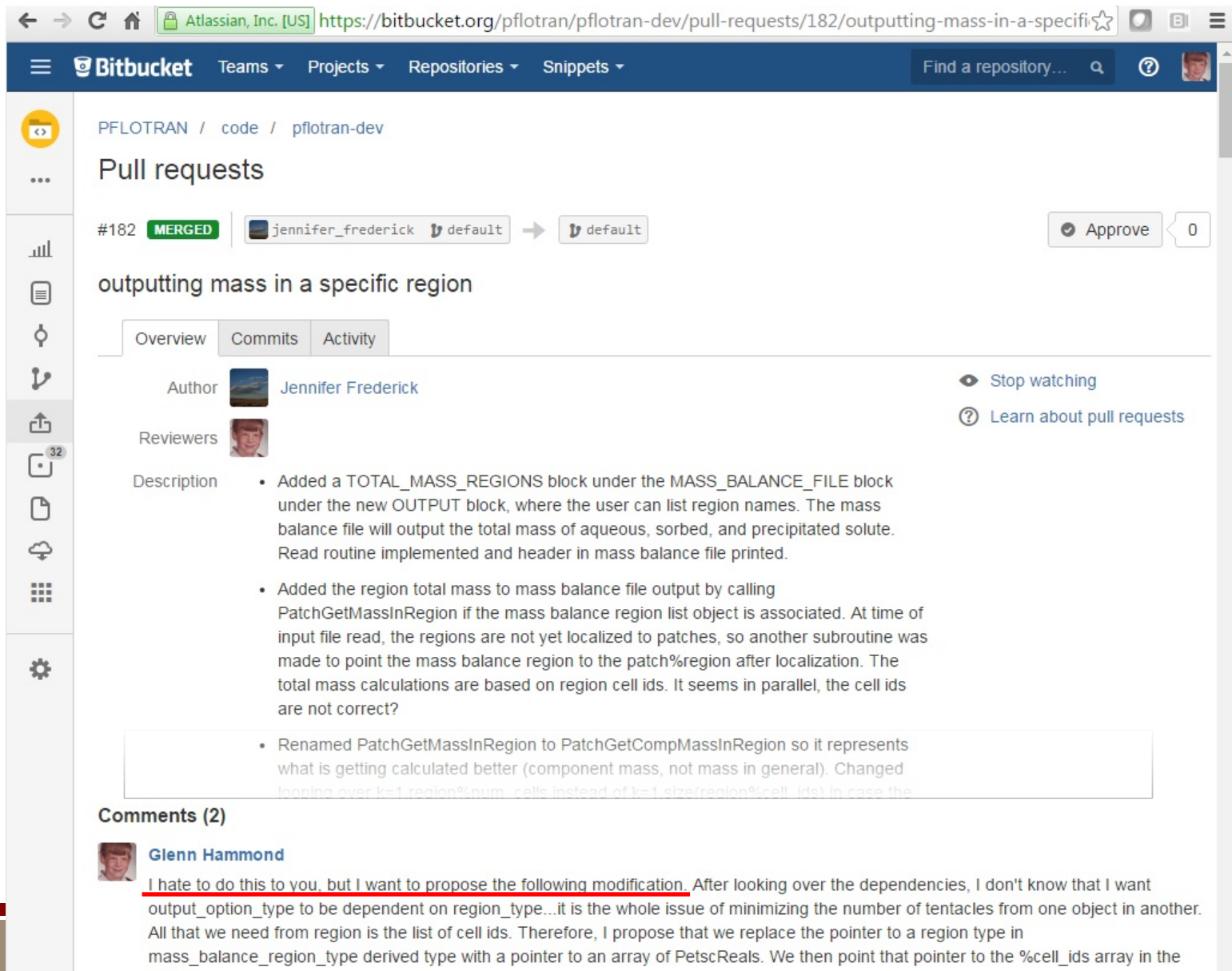
Files changed (3)

+8	-8	regression_tests/ufd/fmdm_dummy.regression.gold
+8	-8	regression_tests/ufd/glass.regression.gold
+133	-60	src/pflotran/pm_waste_form.F90

📄 regression_tests/ufd/fmdm_dummy.regression.gold Side-by-side diff View file Comment ⋮

```
29 29      Mean:  1.000000000000E-03
30 30      1:    1.000000000000E-03
31 31  -- CONCENTRATION: Total Tracer --
32 32  -      Max:  3.2378415318990E-06
33 33  -      Min:  3.2378415318990E-06
```

PFLOTRAN Bitbucket Pull Request




← → ↻ 🏠 Atlassian, Inc. [US] <https://bitbucket.org/pflotran/pflotran-dev/pull-requests/182/outputting-mass-in-a-specifi> ⭐

☰ Bitbucket Teams ▾ Projects ▾ Repositories ▾ Snippets ▾ Find a repository... 🔍 ⓘ 👤


🔗 PFLOTRAN / code / pflotran-dev


Pull requests

#182 **MERGED**  default → default Approve 0

outputting mass in a specific region

Overview **Commits** Activity


Author  Jennifer Frederick Stop watching

Reviewers  Learn about pull requests

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 looping over `k=1, region%num_cells` instead of `k=1, 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

← → ↻ ⬆️ Atlassian, Inc. [US] <https://bitbucket.org/pflotran/pflotran-dev/annotate/7837cd164398ac05c3218ec99d337cc> 🔍 📄 ☰

📁 Imported From IE ★ Bookmarks 📁 Other bookmark


Paolo Orsini 98016c5 2015-10-18	60	PetscInt :: gas_phase
Glenn Hammond 25019bc 2008-02-18	61	PetscInt :: oil_phase
Glenn Hammond 0974189 2008-05-02	62	PetscInt :: nflowdof
Satish Karra 4e5b21c 2012-04-02	63	PetscInt :: nflowspec
Satish Karra f6f3cfa 2012-10-05	64	PetscInt :: nmechdof
Ben Andre 9ebbd6 2014-01-03	65	PetscInt :: nsec_cells
	66	PetscBool :: use_th_freezing
	67	
Gautam Bisht ca1eb21 2014-07-30	68	PetscBool :: surf_flow_on
Gautam Bisht 367db46 2012-05-11	69	PetscInt :: nsurfflowdof
Gautam Bisht 4337f14 2012-06-08	70	PetscInt :: subsurf_surf_coupling
Gautam Bisht 977c4b8 2012-08-06	71	PetscInt :: surface_flow_formulation
Gautam Bisht 323f3dc 2012-10-06	72	PetscReal :: surf_flow_time, surf_flow_dt
Gautam Bisht 083a8fa 2012-10-31	73	PetscReal :: surf_subsurf_coupling_time
	74	PetscReal :: surf_subsurf_coupling_flow_dt
	75	PetscReal :: surf_restart_time
	76	PetscBool :: surf_restart_flag
Gautam Bisht 5f3bfdc 2013-02-08	77	character(len=MAXSTRINGLENGTH) :: surf_initialize_flow_filename
Gautam Bisht f8f3d8d 2013-06-11	78	character(len=MAXSTRINGLENGTH) :: surf_restart_filename
Satish Karra f53dbd2 2013-05-23	79	
Gautam Bisht 9577fe6 2016-01-30	80	PetscBool :: geomech_on
Satish Karra 7948370 2016-07-20	81	PetscBool :: geomech_initial
Gautam Bisht 9577fe6 2016-01-30	82	PetscInt :: ngeomechdof
	83	PetscInt :: n_stress_strain_dof
Satish Karra e0ca5eb 2013-07-03	84	PetscReal :: geomech_time
Gautam Bisht 9577fe6 2016-01-30	85	PetscInt :: geomech_subsurf_coupling
Satish Karra 5d60950 2013-09-26	86	PetscReal :: geomech_gravity(3)
Satish Karra 04446c4 2012-06-12	87	PetscBool :: sec_vars_update
Glenn Hammond 2e4437c 2011-03-08	88	PetscInt :: air_pressure_id
	89	PetscInt :: capillary_pressure_id
	90	PetscInt :: vapor_pressure_id
	91	PetscInt :: saturation_pressure_id
	92	PetscInt :: water_id ! index of water component dof
	93	PetscInt :: air_id ! index of air component dof
	94	PetscInt :: oil_id ! index of oil component dof
	95	PetscInt :: energy_id ! index of energy dof
	96	
	97	PetscInt :: nrandof
	98	
	99	PetscInt :: iflag
	100	PetscInt :: status
	101	PetscBool :: input_record
	102	!geh: remove once legacy code is gone.
	103	! PetscBool :: init_stage
	104	! these flags are for printing outside of time step loop
	105	PetscBool :: print_to_screen
	106	PetscBool :: print_to_file

PFLOTRAN User Support

← → ↻ 🏠 <https://groups.google.com/forum/#!forum/pflotran-users> ☆ 🔔 📧 ☰

Google Search for topics 🔍

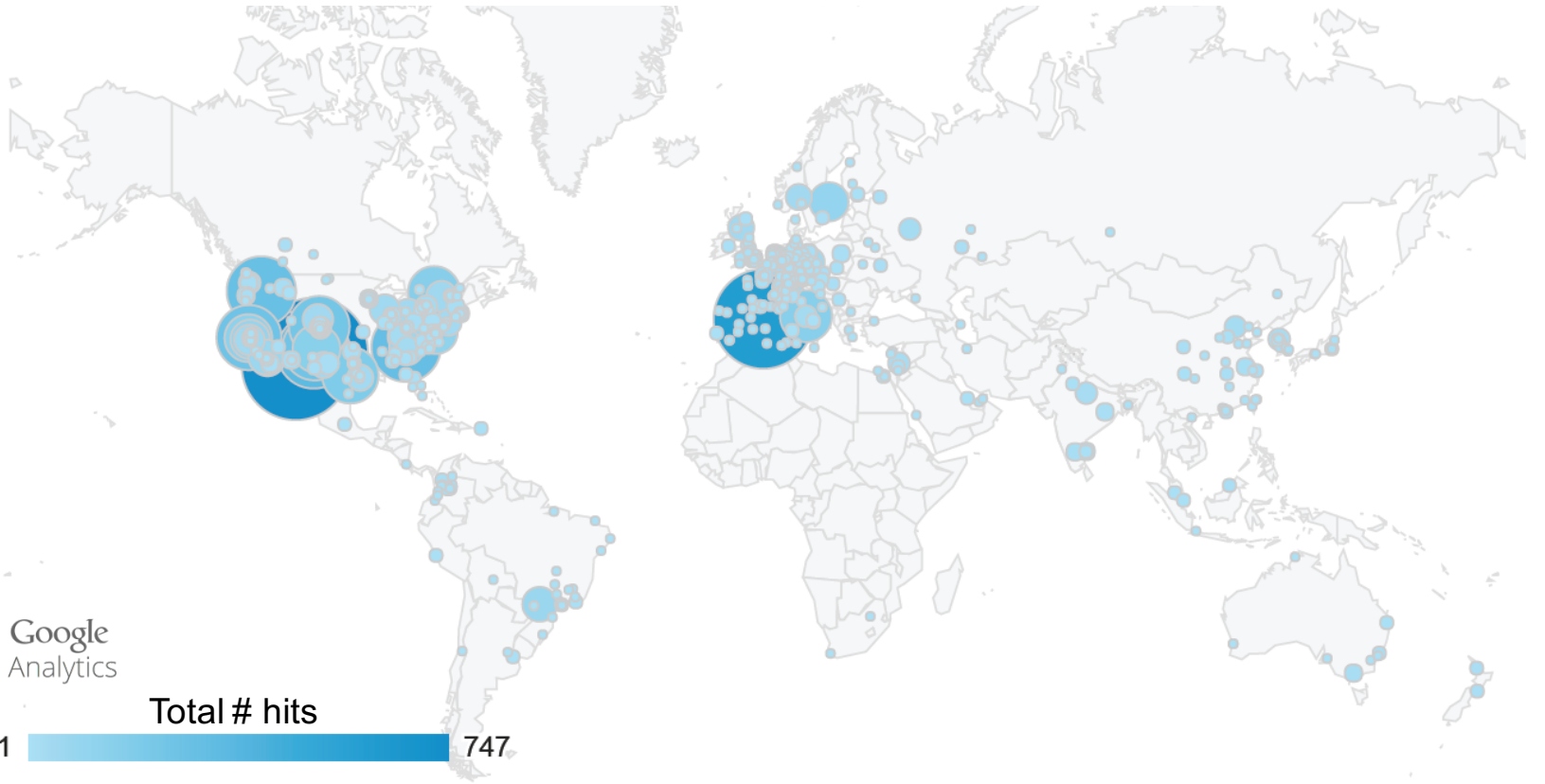
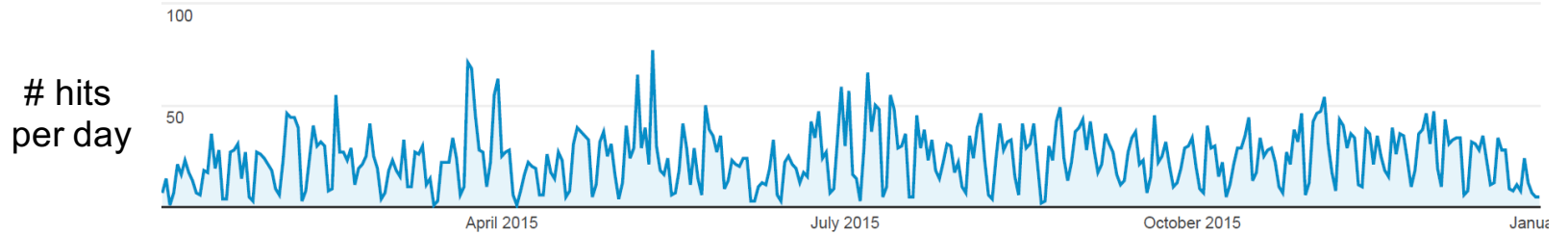
Groups **NEW TOPIC** ↻ Mark all as read Actions ▾ Filters ▾ 👤 ⚙️

▶ **pflotran-users** Shared publicly
60 of 642 topics (99+ unread) ☆  Manage · Members · About ▾

Welcome to the PFLOTRAN users mailing list.
[Edit welcome message](#) [Clear welcome message](#)

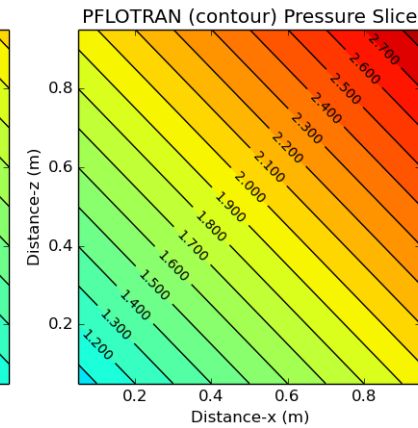
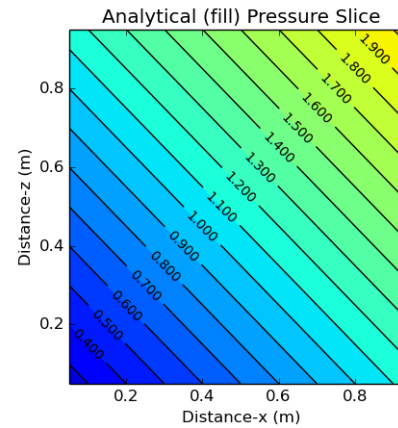
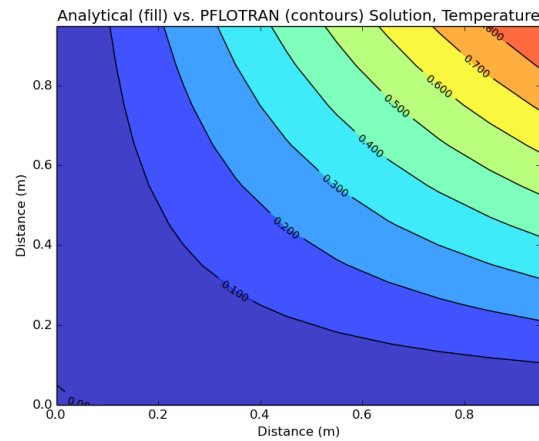
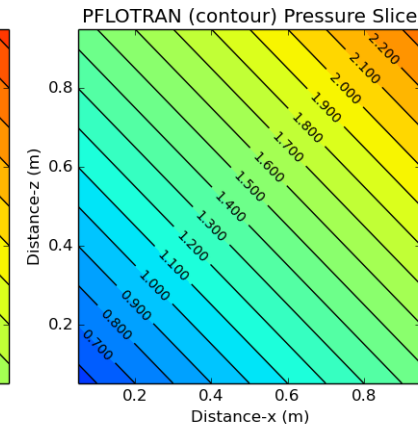
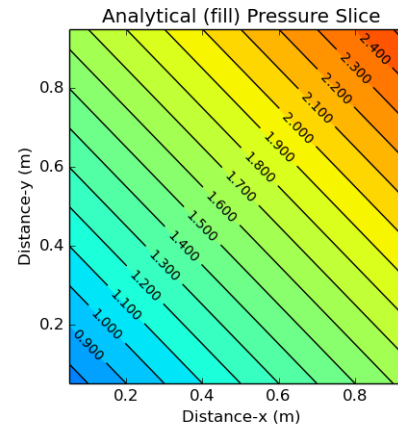
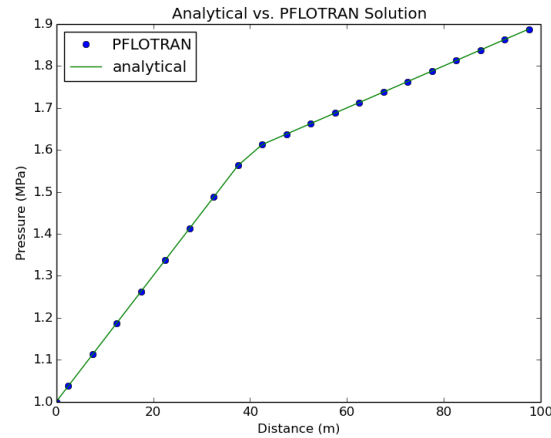
<input type="checkbox"/>	☆	🗨️	PFLOTRAN and ice?	12 posts	69 views	Roland Hendel +5	Jul 29
<input type="checkbox"/>	☆	🗨️	multiple continuum (16)	16	84	paolo trinchero +3	Jul 27
<input type="checkbox"/>	☆	🗨️	comparison of pflotran and modflow (15)	15	56	Linwei Hu +3	Jul 22
<input type="checkbox"/>	☆	🗨️	Functionality: CO2 flow and transport vs. legacy build (3)	3	16	Igal Tsarfis +1	Jul 19
<input type="checkbox"/>	☆	🗨️	Output and regression with geomechanics (1)	1	16	Karra, Satish	Jul 5
<input type="checkbox"/>	☆	🗨️	[Q] Triangles embedded between adjoining tetrahedra in a hybrid mesh...	14	21	Franz M Krumenacker +3	Jun 30
<input type="checkbox"/>	☆	🗨️	Write to file failed (6)	6	27	Andy Ward +1	Jun 29
<input type="checkbox"/>	☆	🗨️	Water EOS for high temperature - above critical temperature (4)	4	13	Paolo Orsini +2	Jun 29
<input type="checkbox"/>	☆	🗨️	Mineral precipitation/dissolution with prefactors (1)	1	10	Hammond, Glenn E	Jun 23
<input type="checkbox"/>	☆	🗨️	please subscribe me to the PFLOTRAN mailing list (1)	1	11	Wissmeier Laurin	Jun 17
<input type="checkbox"/>	☆	🗨️	FLOWRATE output (12)	12	24	Romain P. +2	Jun 16
<input type="checkbox"/>	☆	🗨️	Running PFLOTRAN on Windows (5)	5	14	Hammond, Glenn E +1	Jun 15

Hits on PFLOTRAN Bitbucket Site in 2015



PFLOTRAN QA Testing

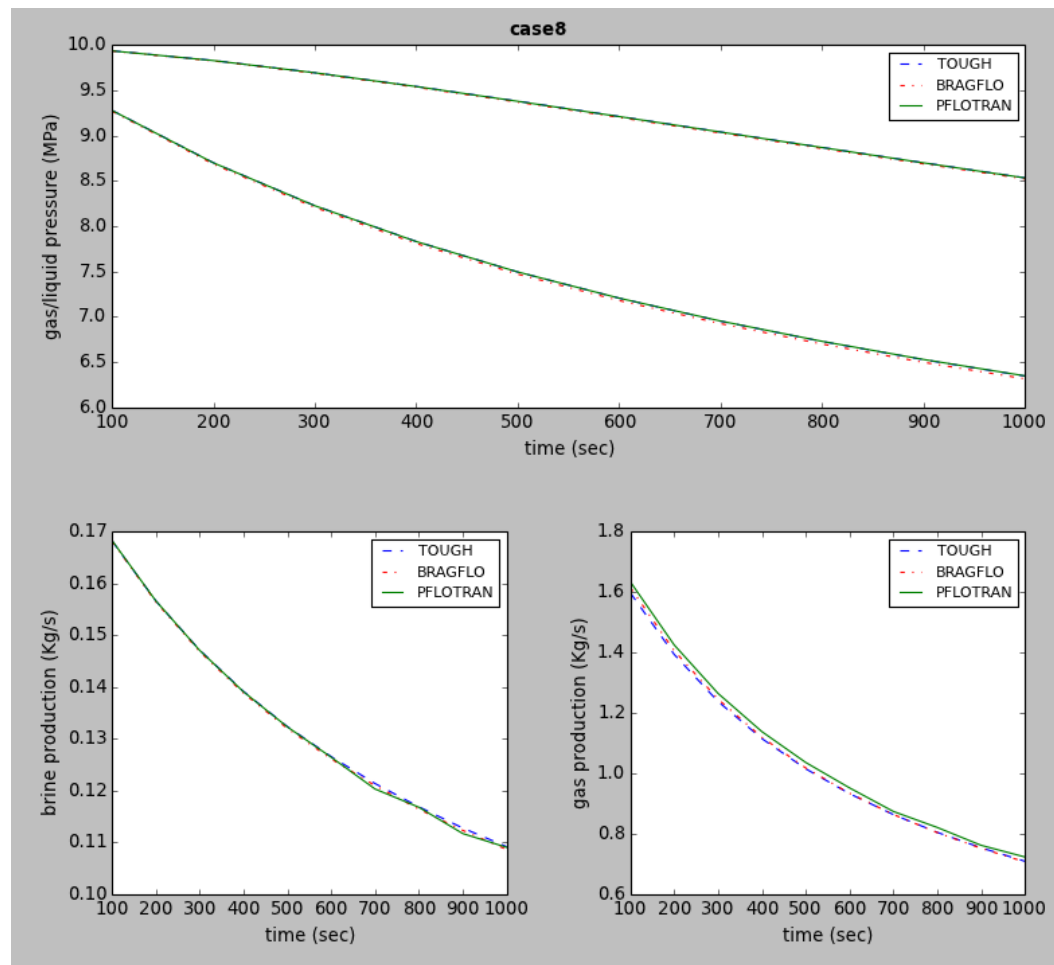
python-scripted framework under development



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)



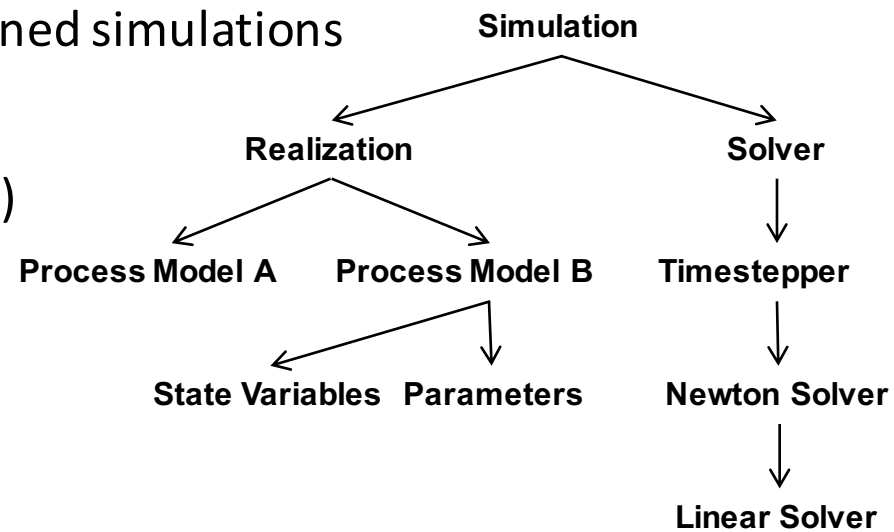
QUESTIONS?

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



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 => EvaluateC02
end type eos_C02

class(eos_C02) :: eos

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

eos = equation of state