

Exceptional service in the national interest

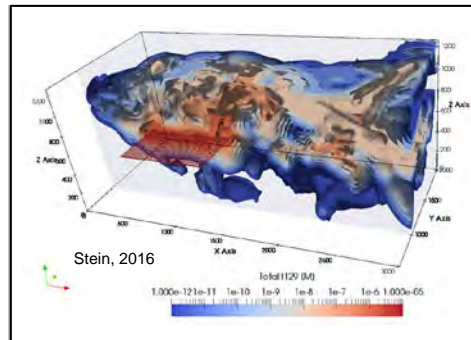


soft·ware ev·o·lu·tion

'sôf(t)wer/ / ,evə'loʊSH(ə)n/

Standard definition:

The gradual development of code, from a simple to a more complex form, due to **repeated** improvements and updates.



Maintaining Quality Assurance Within Software Evolution: Lessons Learned With PFOLOTRAN

Jennifer M. Frederick and Glenn E. Hammond

SAND2017-9211-C



Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

What is Software Evolution?

soft·ware ev·o·lu·tion

'sôf(t)wer/ / ,evə'loʊSH(ə)n/

Standard definition:

The gradual development of code, from a simple to a more complex form, due to ***repeated*** improvements and updates.

What is Software Evolution?

soft·ware ev·o·lu·tion

'sôf(t)wer/ / ,evə 'loʊSH(ə)n/

Standard definition:

The gradual development of code, from a simple to a more complex form, due to **repeated** improvements and updates.

New domain science:

- New process models
- Increasingly mechanistic process models
- Programming paradigms
- Novel numerical methods

What is Software Evolution?

soft·ware ev·o·lu·tion

'sôf(t)wer/ / ,evə 'loʊSH(ə)n/

Standard definition:

The gradual development of code, from a simple to a more complex form, due to **repeated** improvements and updates.

New computational science:

- Changing third party libraries
- Changing operating systems
- Changing programming language
- New or architecturally changing computer hardware

What is Software Evolution?

soft·ware ev·o·lu·tion

'sôf(t)wer/ / ,evə'loʊSH(ə)n/

Standard definition:

The gradual development of code, from a simple to a more complex form, due to **repeated** improvements and updates.

When software is constantly evolving,
how can we ensure software quality?

What is Software Quality Assurance?

soft·ware qual·i·ty as·sur·ance

'sôf(t)wer/ 'kwälədē/ ə 'SHoŏrəns/

IEEE standard:

A planned and systematic pattern of all actions necessary **to provide adequate confidence** that software conforms to established technical requirements.

When software is constantly evolving,
how can we ensure software quality?

What is Software Quality Assurance?

soft·ware qual·i·ty as·sur·ance

'sôf(t)wer/ 'kwälədē/ ə 'SHoŏrəns/

IEEE standard:

A planned and systematic pattern of all actions necessary **to provide adequate confidence** that software conforms to established technical requirements.

Software quality includes:

- **Correctness**
- **Reliability**
- **Efficiency**
- **Survivability**
- **Maintainability**
- **Testability**
- **Availability**
- **Portability**

What is Software Quality Assurance?

soft·ware qual·i·ty as·sur·ance

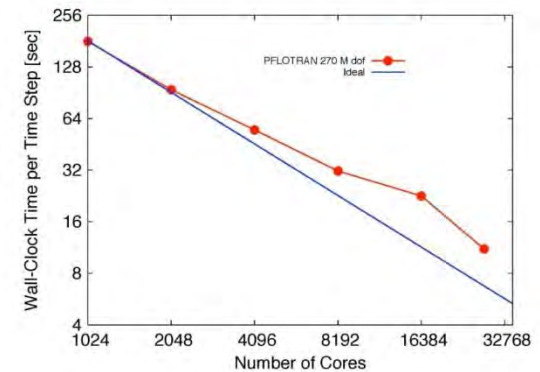
'sôf(t)wer/ 'kwälədē/ ə 'SHoŏrəns/

IEEE standard:

A planned and systematic pattern of all actions necessary **to provide adequate confidence** that software conforms to established technical requirements.

How does PFLOTRAN ensure software quality under an evolving software framework?

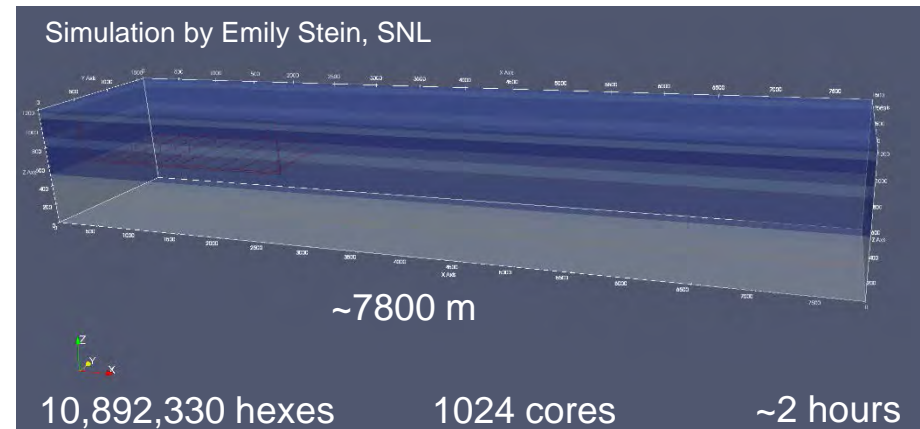
- Reactive multiphase flow and transport code for porous media
- Open source license (GNU LGPL 2.0)
- Object-oriented Fortran 2003/2008
 - Pointers to procedures
 - Classes (extendable derived types with member procedures)
- Founded upon well-known (supported) open source libraries
 - MPI, PETSc, HDF5, 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



PFLOTRAN

- Nuclear waste disposal
 - Waste Isolation Pilot Plant (WIPP) in Carlsbad, NM
 - DOE Spent Fuel and Waste Science & Technology 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
- CO₂ sequestration
- Enhanced geothermal energy
- Radioisotope tracers
- Colloid-facilitated transport

pa.sandia.gov



PFLOTRAN

Steps taken to minimize impacts of software evolution:

- Open source development
- Software configuration management
- Modular object oriented design
- Automated testing suites
- Online documentation



Open Source Development

- Google

“Open source is an adjective denoting software for which the original source code is made **freely available** and **may be redistributed and modified.**”
- Open source software refers to code that:
 - Is free
 - Is publicly available
 - Can be legally modified
 - Can be legally shared with anyone

Open Source Development

- PFLOTRAN has an open source GNU Lesser General Public License (LGPL).
 - The original or modified source may not be sold for profit.
 - Third-party software linked to or wrapped around PFLOTRAN (e.g. graphical user interfaces [GUIs], pre-/post-processing tools, etc.) may be proprietary.



Benefits of Open Source Software

- Encourages **collaboration**
 - Development, testing, debugging can be shared
- **Transparency** exposes implementation details critical to scientific reproducibility, but excluded by journal publications.
- More optimal use of funding
 - **Funding pooled** across diverse set of projects/budgets.
 - What would have been spent on **licensing fees** can be redirected toward development.
 - **Infinite benefit** to those unfunded
- The open source community can drive the code to evolve beyond the original vision (**evolution**).
- The most fit codes tend to survive (**natural selection**)

*critical for software
quality assurance*

Software Configuration Management

- PFLOTRAN employs the Git distributed source control management tool for configuration management.

- Git logs all changes to a code repository
 - version control



- Git allows developers to:
 - Clone the base repository
 - Modify and test code in a development branch
 - Merge changes back into base repository
 - Pinpoint problematic changesets (snapshots of code versions)

*critical for software
quality assurance*

Software Configuration Management

- The PFLOTRAN source code repository is hosted at Bitbucket.org.

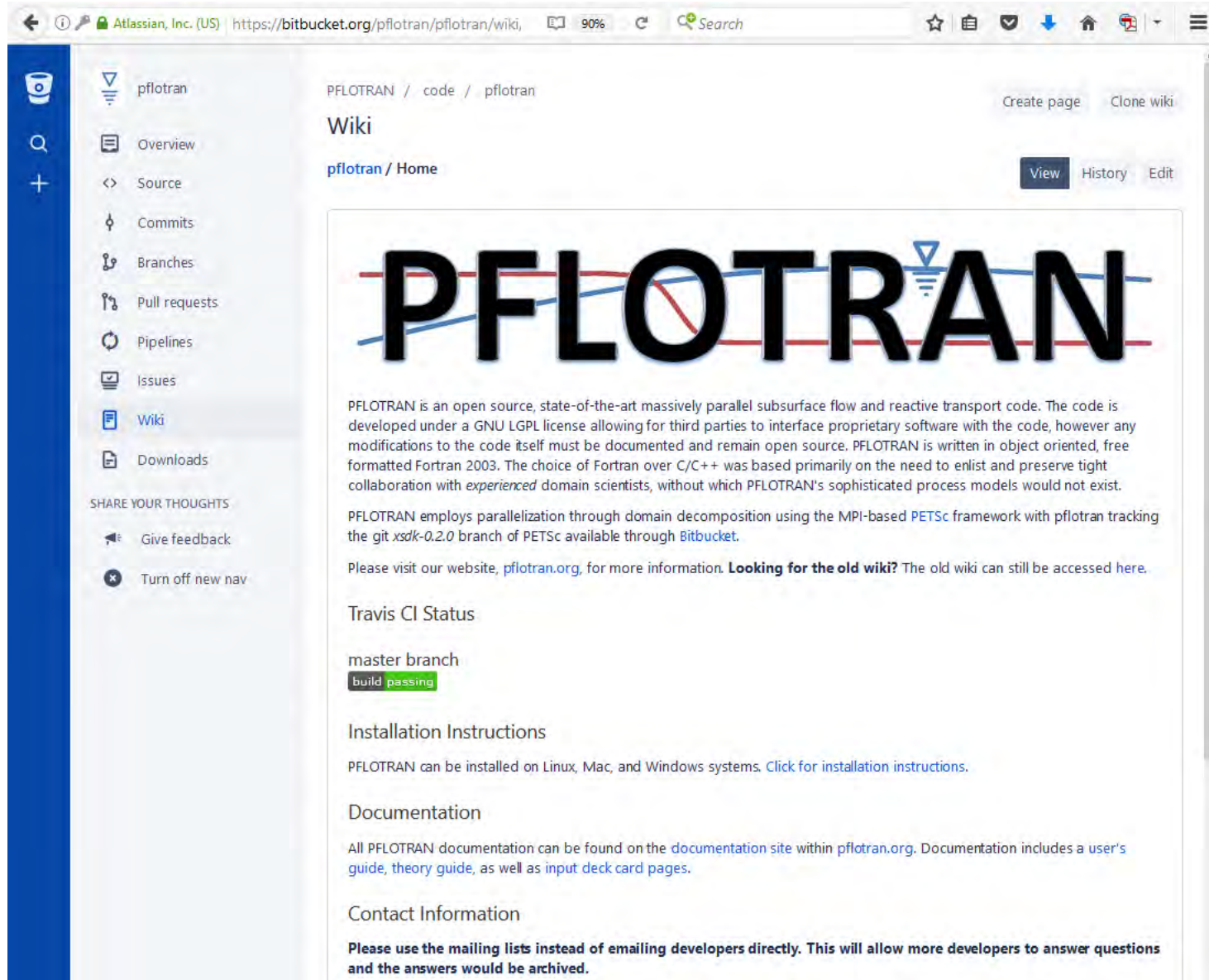
<https://bitbucket.org/pflotran/pflotran>

*critical for software
quality assurance
(availability)*

- Bitbucket is a web-based hosting service for software development projects that use Git.
- Provides:
 - Git operations (clone, fork, branch, etc.)
 - Wiki for information or documentation
 - Source tree
 - Pull requests
 - Commit logs
 - Issue tracker

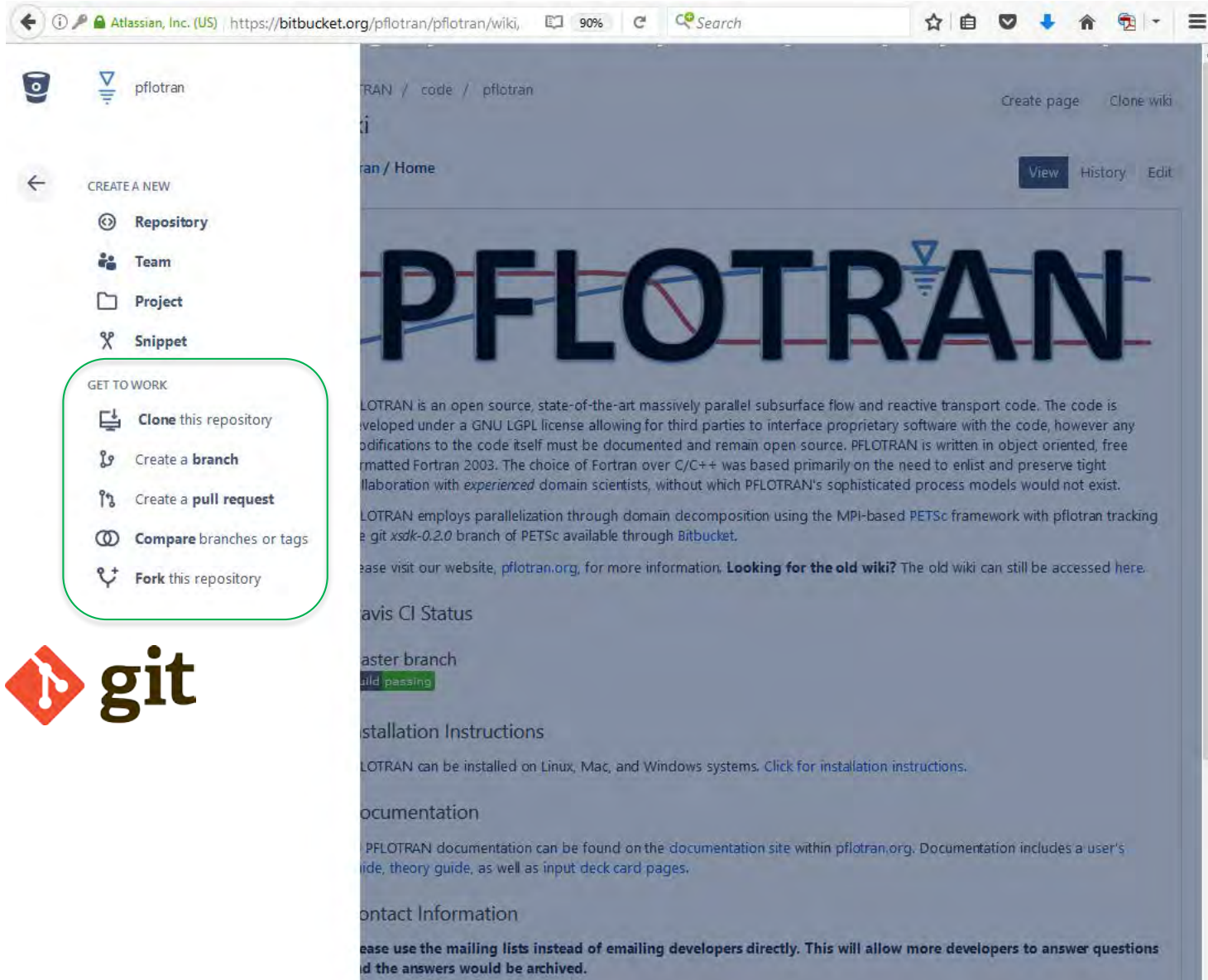


Software Configuration Management




The screenshot shows a web browser displaying the Bitbucket Wiki for PFLOTRAN. The browser's address bar shows the URL <https://bitbucket.org/pflotran/pflotran/wiki>. The page has a blue sidebar on the left with navigation links: Overview, Source, Commits, Branches, Pull requests, Pipelines, Issues, Wiki (highlighted), and Downloads. Below the sidebar is a section titled 'SHARE YOUR THOUGHTS' with links for 'Give feedback' and 'Turn off new nav'. The main content area has a breadcrumb trail 'PFLOTRAN / code / pflotran' and a 'Wiki' title. It includes buttons for 'Create page', 'Clone wiki', 'View', 'History', and 'Edit'. The central part of the page features a large 'PFLOTRAN' logo with a stylized blue and red graphic. Below the logo, the text describes PFLOTRAN as an open source, state-of-the-art massively parallel subsurface flow and reactive transport code, developed under a GNU LGPL license. It mentions that the code is written in object-oriented, free-formatted Fortran 2003 and was chosen over C/C++ for its ability to enlist and preserve tight collaboration with experienced domain scientists. The text also states that PFLOTRAN employs parallelization through domain decomposition using the MPI-based PETSc framework, with pflotran tracking the `git xsdk-0.2.0` branch of PETSc available through Bitbucket. A link to the website pflotran.org is provided for more information, along with a note about the old wiki being accessible [here](#). Below this, there is a 'Travis CI Status' section showing the 'master branch' with a 'build passing' status. The 'Installation Instructions' section states that PFLOTRAN can be installed on Linux, Mac, and Windows systems, with a link to [Click for installation instructions](#). The 'Documentation' section mentions that all PFLOTRAN documentation can be found on the [documentation site](#) within pflotran.org, including a [user's guide](#), [theory guide](#), and [input deck card pages](#). Finally, the 'Contact Information' section advises users to **use the mailing lists instead of emailing developers directly**, as this will allow more developers to answer questions and the answers will be archived.

Software Configuration Management



The screenshot shows the Bitbucket web interface for the PFLOTRAN repository. The left sidebar contains navigation options: 'Repository', 'Team', 'Project', and 'Snippet'. Below these is a 'GET TO WORK' section with a green border, containing links for 'Clone this repository', 'Create a branch', 'Create a pull request', 'Compare branches or tags', and 'Fork this repository'. The main content area displays the 'PFLOTRAN' title with a logo, followed by a description of the code as an open-source, state-of-the-art massively parallel subsurface flow and reactive transport code. It mentions the GNU LGPL license and the use of Fortran 2003. The page also includes sections for ' Travis CI Status', 'Master branch' (with a 'Build passing' status), 'Installation Instructions', 'Documentation', and 'Contact Information'. The Git logo is visible in the bottom left corner of the screenshot.

Git logo:  **git**

Software Configuration Management

Atlassian, Inc. (US) https://bitbucket.org/pflotran/pflotran/src 90% Search

PFLOTTRAN / code / pflotran

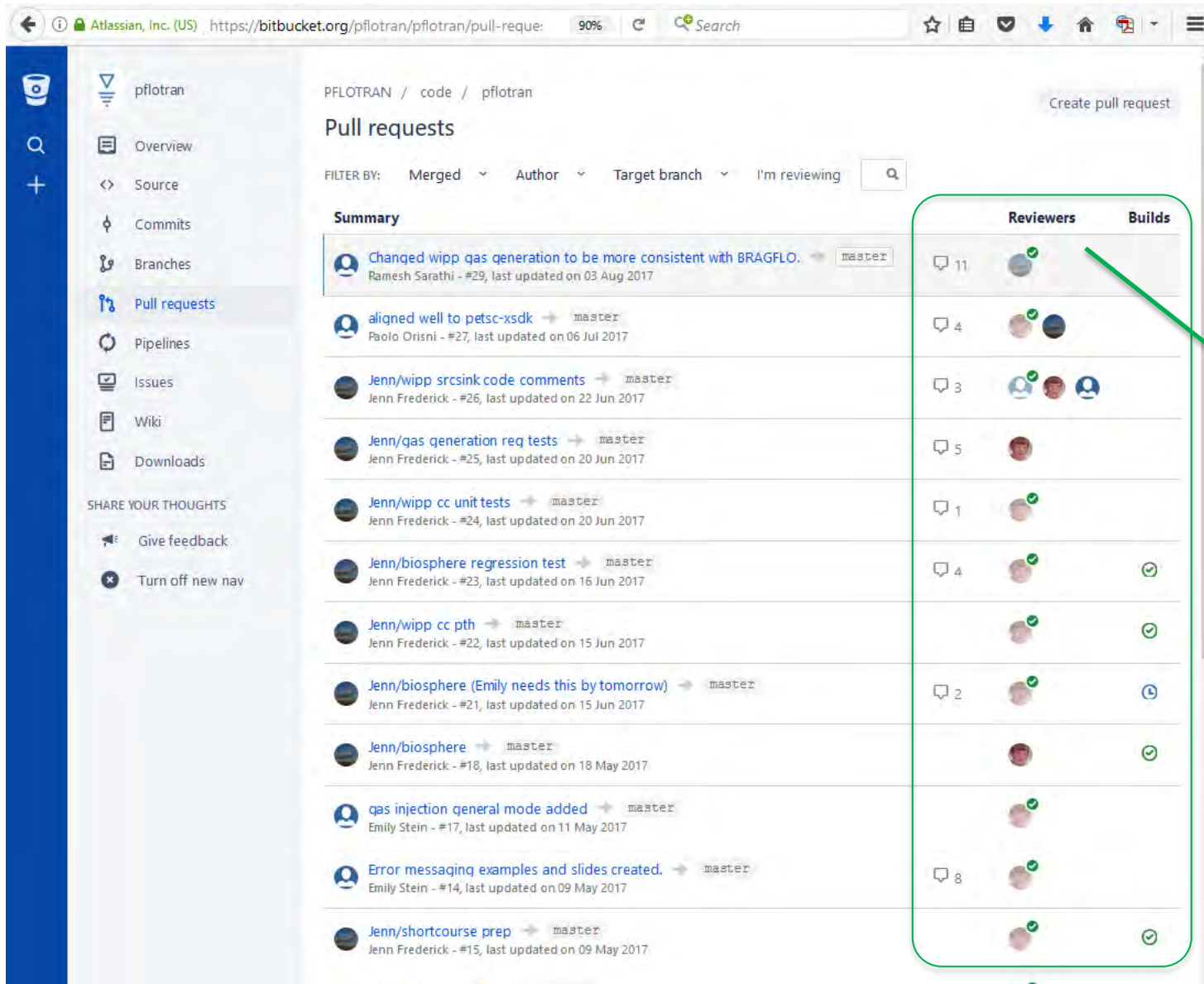
Source

master pflotran / New file

- .bitbucket-pipeline
- .travis
- database
- docs
- example_problems
- hanford
- regression_tests
- shortcourse/exercises
- src
- tools
- tpls

.gitattributes	455 B	2017-04-21	Updated .gitattributed to prevent auto conversion of local text files.
.gitignore	1.9 KB	2017-04-20	Added regression_tests, tpls, and src/pflotran/unittests to ignored
.hqeol	71 B	2016-04-28	Another try to fix +x.
.hqignore	1.9 KB	2016-05-11	.hgignore edited to remove previously added *.chk and *.h5
.travis.yml	688 B	2017-07-12	Removed 'next' branch from continuous integration.
bitbucket-pipelines.yml	206 B	2016-09-21	dummy commit to get the pipeline started
configure	624 B	2016-11-17	Fixed corrupt configure file.
makefile	2.3 KB	2017-01-26	Pflotran should also install its generated .mod files in the include

Software Configuration Management



The screenshot shows the Bitbucket web interface for the 'pflotran' repository. The left sidebar contains navigation links: Overview, Source, Commits, Branches, Pull requests (highlighted), Pipelines, Issues, Wiki, and Downloads. Below these are links to 'Give feedback' and 'Turn off new nav'. The main content area is titled 'Pull requests' and includes a 'Create pull request' button. A filter bar shows 'FILTER BY: Merged Author Target branch I'm reviewing'. A 'Summary' section is visible. The main list of pull requests is as follows:

Summary	Reviewers	Builds
Changed wipp gas generation to be more consistent with BRAGFLO. → master Ramesh Sarathi - #29, last updated on 03 Aug 2017	11	
aligned well to petsc-xsdk → master Paolo Orisni - #27, last updated on 06 Jul 2017	4	
Jenn/wipp srcsink code comments → master Jenn Frederick - #26, last updated on 22 Jun 2017	3	
Jenn/gas generation req tests → master Jenn Frederick - #25, last updated on 20 Jun 2017	5	
Jenn/wipp cc unit tests → master Jenn Frederick - #24, last updated on 20 Jun 2017	1	
Jenn/biosphere regression test → master Jenn Frederick - #23, last updated on 16 Jun 2017	4	
Jenn/wipp cc pth → master Jenn Frederick - #22, last updated on 15 Jun 2017		
Jenn/biosphere (Emily needs this by tomorrow) → master Jenn Frederick - #21, last updated on 15 Jun 2017	2	
Jenn/biosphere → master Jenn Frederick - #18, last updated on 18 May 2017		
gas injection general mode added → master Emily Stein - #17, last updated on 11 May 2017		
Error messaging examples and slides created. → master Emily Stein - #14, last updated on 09 May 2017	8	
Jenn/shortcourse prep → master Jenn Frederick - #15, last updated on 09 May 2017		

Pull requests are reviewed, commented on, and scrutinized by a second pair of eyes.

Software Configuration Management

Atlassian, Inc. (US) https://bitbucket.org/pfioTRAN/pfioTRAN/commits/a 90% Search

pfioTRAN

Overview

Source

Commits

Branches

Pull requests

Pipelines

Issues

Wiki

Downloads

SHARE YOUR THOUGHTS

Give feedback

Turn off new nav

PFIOTRAN / code / pfioTRAN

Commits

All branches

Find commits

Unique commit numbers and descriptions.

Author	Commit	Message	Date	Builds
Glenn Hammo...	c7ebfd2	Proposed fixes to wipp sr... glenn/wipp-gas-generat...	21 hours ago	✓
Emily Stein	40a7d23	Command line added to gas_inj... emily-shortcourse	2 days ago	✓
Satish Karra	43bc28b	Fixed averaging bug in str... satish/geomech-perm-po...	2 days ago	✓
Emily Stein	c305212	heater example in general mod... emily-shortcourse	2 days ago	
Jenn Frederick	1ee4b9b	Removed the print statements for debugging. Chan...	2 days ago	✓
Jenn Frederick	284458f	Added some temporary print statements and chan...	2 days ago	
Jenn Frederick	07c295c	Added print messages for some debugging, and c...	2 days ago	
Jenn Frederick	c44f678	Merge branch 'jenny/adding-output-for-ufd-decay'	2 days ago	
Jenn Frederick	bf741f4	Added PRINT_OUTPUT keyword to UFD_DECA... 1	2 days ago	
Satish Karra	ecd516e	Added setup of porosity... satish/geomech-perm-po...	2 days ago	✓
Satish Karra	7660182	Removed statements to e... satish/geomech-perm-po...	3 days ago	✓
Emily Stein	58bc082	first pass at multiphase heater emily-shortcourse	4 days ago	
Satish Karra	fff380d	Fixed inconsistencies in po... satish/geomech-perm-po...	4 days ago	✓
Glenn Hammo...	679d4d7	Fixed setting of Output procedur... glenn/wipp-reset	2017-08-18	✓
Glenn Hammo...	ced4a49	Added subsurface_reset.F90; up... glenn/wipp-reset	2017-08-18	
Glenn Hammo...	9faf4af	Initial coding of subsurface reset. glenn/wipp-reset	2017-08-18	!
Glenn Hammo...	2a30c1b	Merge branch 'master' into glen... glenn/wipp-reset	2017-08-18	✓
Glenn Hammo...	8f33d80	Switch solver for 543_river_het_map_dirich-np4 to in... glenn/wipp-reset	2017-08-18	✓
Glenn Hammo...	22c5f23	Merge branch 'glenn/wipp-reset'... glenn/wipp-reset	2017-08-18	✓
Glenn Hammo...	6d65e51	Added reading of cell indexed d... glenn/wipp-reset	2017-08-18	
Glenn Hammo...	3a4dd8f	Updated regression files that we... glenn/wipp-reset	2017-08-18	
Glenn Hammo...	395242c	Remove option%initial_time in fa... glenn/wipp-reset	2017-08-18	
Glenn Hammo...	4e166bf	Moved initialization of timestep... glenn/wipp-reset	2017-08-18	

Automatic builds each time the code has changed.

Modular Object Oriented Design

- Object oriented design involves the organization of data and procedures into a hierarchy of containers (objects).

- The use of objects:

- Improves data locality (modularity)
- Eases code refactoring (rewriting)
- Facilitates extensibility (for adding capability)

*critical for software
quality assurance
(maintainability)*

- PFLOTRAN uses object oriented modern
FORTAN 2003/2008

Automated Testing Suites

- As open source development fosters a growing community of developers, **the code can't break!**.
- **Unit tests**
 - Individual routines are executed in isolation.
 - Results are compared with a gold standard to *within a tolerance*.
- **Regression tests** – focus on changes in simulation results
 - Full simulations are executed.
 - Simulations results are sampled and compared to a gold standard to *within a tolerance*.
- **Verification tests**
 - Full simulations are executed, for which there is a known solution.
 - Simulation results are sampled and compared to an analytical solution *within a tolerance*.

```
if (abs(test_value - gold_standard) > tolerance) report_error()
```

Automated Testing Suites

- As open source development fosters a growing community of developers, **the code can't break!**.

Why a tolerance?

*Accommodates small variations
in software and hardware
configurations (Linux vs Mac)*

- **Unit tests**

- Individual routines are executed in isolation.
- Results are compared with a gold standard to ***within a tolerance***.

- **Regression tests** – focus on changes in simulation results

- Full simulations are executed.
- Simulations results are sampled and compared to a gold standard to ***within a tolerance***.

- **Verification tests**

- Full simulations are executed, for which there is a known solution.
- Simulation results are sampled and compared to an analytical solution ***within a tolerance***.

```
if (abs(test_value - gold_standard) > tolerance) report_error()
```


Automated Testing Suites: Regression Tests

This is what
a successful
run of the
unit tests
and
regression
tests looks
like:



```
[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.
```

- Example regression test failure:
 - Perturb the critical pressure for the 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
```

Automated Testing Suites: Regression Tests

This is what
a failed run
of the unit
tests and
regression
tests looks
like:



```
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/ld/ld-calcite/ld-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
```

Automated Testing Suites: Regression Tests

----- 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.
```

this test didn't use the
change in eos_water.f90,
so it passes

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.
```

this test used
the change in
eos_water.f90,
so it fails

Automated Testing Suites: Verification Tests

- A set of > 50 tests that verify the code against an analytical solution
- Automatic spatial convergence testing is also performed
- The tests can be run each time a major portion of the code changes

```
=====
3D, Steady, Thermal, BCs 1st kind, TH Mode
=====
Running PFLOTRAN simulation . . .
Relative Maximum Error: 1.0779999995 %
-- Test PASS --

===== Running STEADY THERMAL tests =====
===== GENERAL Mode =====
/home/jmfrede/software/pflotran-qa/qa_tests/thermal/steady

=====
1D, Steady, Thermal, BCs 1st kind, General Mode
=====
Running PFLOTRAN simulation . . .
Relative Maximum Error: 0.0 %
-- Test PASS --

=====
1D, Steady, Thermal, BCs 1st/2nd kind, General Mode
=====
Running PFLOTRAN simulation . . .
Relative Maximum Error: 1.32563943235e-14 %
-- Test PASS --

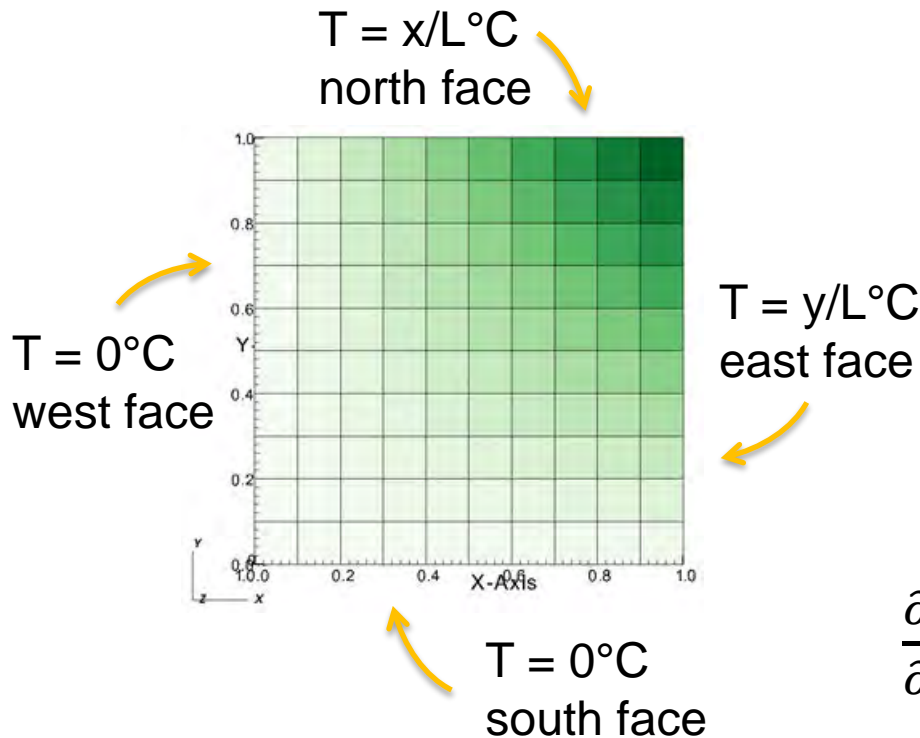
=====
2D, Steady, Thermal, BCs 1st kind, General Mode
=====
Running PFLOTRAN simulation . . .
Relative Maximum Error: 2.74129141815e-14 %
-- Test PASS --

=====
2D, Steady, Thermal, BCs 1st/2nd kind, General Mode
=====
Running PFLOTRAN simulation . . .
Relative Maximum Error: 99.9999999333 %
-- Test FAIL --

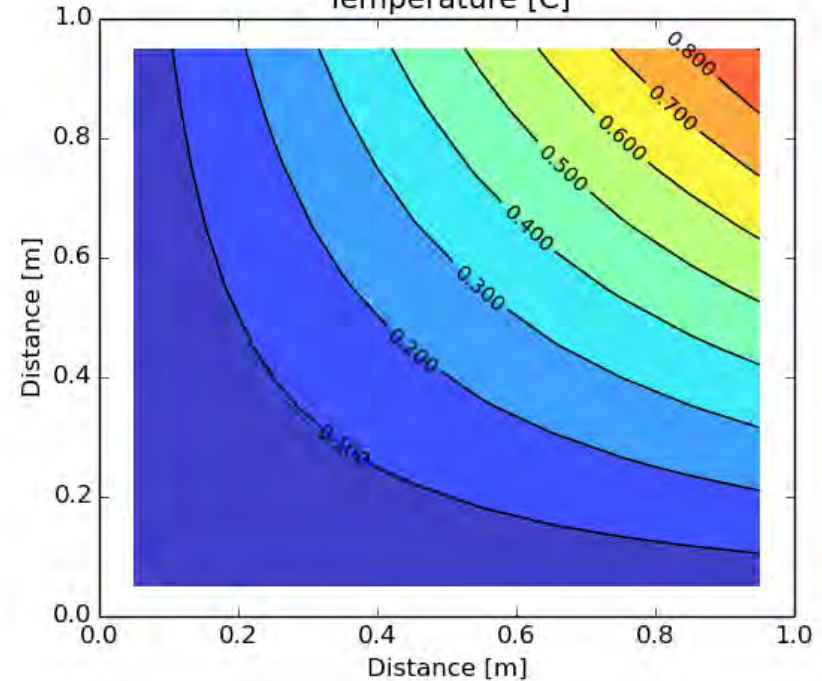
=====
3D, Steady, Thermal, BCs 1st kind, General Mode
=====
Running PFLOTRAN simulation . . .
Relative Maximum Error: 2.40048221534e-14 %
-- Test PASS --
```

Automated Testing Suites: Verification Tests

- 2D Domain (10x10 cells)
- Heat Conduction (steady state solution)
- Dirichlet (scalar) temperature boundary conditions



Analytical (fill) vs. PFLOTRAN (contour) TH Mode 0.00% error
Temperature [C]



governing equation

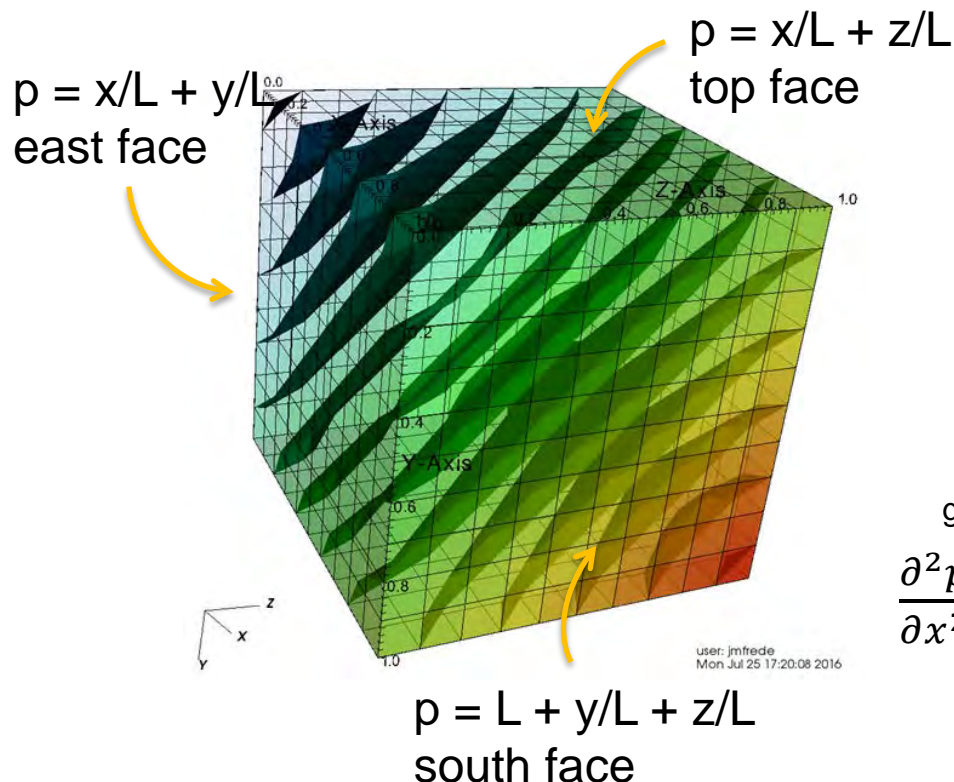
$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

analytical solution

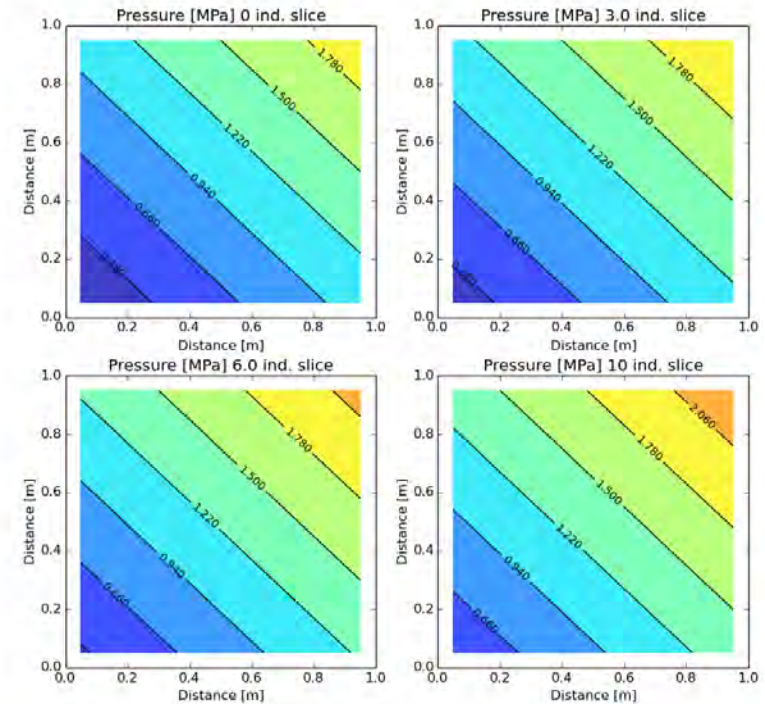
$$T(x, y) = T_0 \frac{x}{L} \frac{y}{L}$$

Automated Testing Suites: Verification Tests

- 3D Domain (10x10x10 cells)
- Fluid flow/pressure field (steady state solution)
- Dirichlet (scalar) pressure boundary conditions



Analytical (fill) vs. PFLOTRAN (contour) RICHARDS Mode 0.00% error



governing equation

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} = 0$$

analytical solution

$$p(x, y, z) = p_0 \left(\frac{x}{L} + \frac{y}{L} + \frac{z}{L} \right)$$

PFLOTRAN's Online Documentation

Documentation pages are version controlled
and also hosted on Bitbucket.org

documentation.pflotran.org

- We use a documentation generator program (Sphinx) to generate both the website and the PDF versions of the documentation:



The screenshot shows the PFLOTRAN online documentation page for the topic "1D Steady Flow (Pressure), BCs of 1st Kind". The page includes a table of contents, a problem description, and a 3D visualization of the domain. The problem description states: "This problem is adapted from Kolditz, et al. (2015). Thermo-Hydro-Mechanical-Chemical Processes in Fractured Porous Media: Modeling and Benchmarking. Closed Form Solutions, Springer International Publishing, Switzerland. Section 2.2.1, pg.27, 'A 1D Steady-State Pressure Distribution, Boundary Conditions of 1st Kind'." The 3D visualization shows a rectangular domain with a color gradient representing pressure distribution.

The screenshot shows the PFLOTRAN online documentation page for "Appendix B: Method of Solution". The page describes the solution methods used in the software, including the integrated finite volume discretization and the governing partial differential equations for mass conservation. The governing equation is given as:

$$\frac{\partial}{\partial t} A_j + \nabla \cdot \mathbf{F}_j = Q_j$$

The page also includes a table of contents and a quick search bar.

PFLOTRAN's Online Documentation

Documentation pages are version controlled
and also hosted on Bitbucket.org

documentation.pflotran.org

- We use a documentation generator program (Sphinx) to generate both the website and the PDF versions of the documentation:



The screenshots display the PFLOTRAN documentation website and its corresponding PDF files. The website interface includes a navigation menu on the left with sections like 'Theory Guide', 'User's Guide', 'QA Test Suite', and '1D Steady The...'. The main content area shows a 'CONTENTS' page with a table of contents listing various topics and their page numbers. Other screenshots show specific technical sections such as '2.2.1 Required Software Packages', '2.2.2 Installing PFLOTRAN', 'Brooks-Cory Saturation Function', 'Relative Permeability', and '1.1.3 Mode: MPMSE'. The PDF files are shown as overlapping windows, indicating that the documentation is version controlled and hosted on Bitbucket.org.

Conclusion



www.pflotran.org

- Although software tends to have a limited lifespan, **careful design and planning** in the development of a code **can significantly lengthen** the duration of a software application's **viable existence**.
- PFLOTRAN attempts to minimize the impact of software evolution through:
 - Open source development
 - Software configuration management
 - Modular object oriented design
 - Automated testing
 - Online documentation

