Improvements in Sequential Verification

Dr. George Mesina

International RELAP5-3D User Group Meeting

Date: Aug 13, 2015

Overview

- Basic idea of Sequential Verification
- Method – Primary variables, verification file
- Statistical Theory
- Detection
- Coverage
- Future

Sequential Verification (SV)

Software Verification

- Evaluates a software system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

Sequential Verification (SV)

- If sequence of code versions produces same calculations tracing back to initial development
- OR if changes to calculations are justified (bug fix, development, etc.)

- *I.E. No unexpected, unjustified differences in code calculations*

Which calculations do we compare?

Sequential Verification Theory – Basics

- **Primary** variables are the ones solved for in the governing equations
  - Secondary variables derive from them and, on the next time step, contribute to building the system solved for primary variables
  - Tertiary variables are output only. No feedback to primary vars.
- If a secondary variable is calculated incorrectly in one code version, but not another, on a given step
  - Some primary variable(s) will be wrong on the next step when the system is solved for primary variables.
- Secondary variables are unnecessary for finding differences.

Sequential Verification RELAP5-3D Primary Vars

- **Primary** variables are the ones solved for in the governing equations

Quantity	In manual	On file
Pressure	P	P
Liquid internal energy	U_f	Uf
Gas internal energy	U_g	Ug
Void fraction of gas	α_g	VOIDg
Noncondensable quality	X_n	QUALa
Liquid velocity	V_f	Vf
Gas velocity	V_g	Vg
Heat Structure Temperature	T	Temp
Neutron flux	φ	Flux
Timesteps sum	Δt, Δt_{kin}	dtsum
Trips	T_r	Trips
Control system value	Y	Cntrl

Sequential Verification Theory – Basics

- Calculate **L_1-norms** of the arrays of primary variables and write them on a ***verification file***.
- Compare verification files between 2 runs by different code versions
- If **L_1-norms** are exactly the same, want to conclude code's calculations are unaffected by code changes between versions

Verification File for RELAP5-3D

- Verification file displays ID and dumps of L_1-norms of primary variables

Dump: Dump #, Adv #, Time, L_1-norms in Sci & Hex

Header
* ID of Code & Computer
* Date/Time Compiled
* Date/Time of Run

Input Case # & Title

```
RELAP5-3D/Ver:4.3.1   inl425834-linux
Time compiled: Dec  8 2014 09:10:06
Date and Time of run: 14/12/10     09:06:02

Case   1  edhtrkm: edward's pipe problem base case with extras

Dump    1 Advancement=     110 time=1.0100000000000006439E-01
P=       4.9364464366256470E+07 4018789EF82EE17DF400000000000000
Uf=      1.9645487450034248E+07 40172BC42F733571C840000000000000
Ug=      5.4521337937904159E+07 40189FF6FCF80D3E5500000000000000
VOIDg=   7.0684077373541792E+00 4001C460CAD8AEB11000000000000000
QUALa=   0.0000000000000000E+00                                0
Boron=   0.0000000000000000E+00                                0
Vf=      2.0609367404278057E+02 40069C2FF60B4C740DFA000000000000
Vg=      2.3448536361144574E+02 4006D4F881944BA7E45F000000000000
RHSth=   7.5475342950407861E+04 400F26D357CB9911E9C0000000000000
SOLth=   6.7091414384136505E+04 400F06136A15142A4524000000000000
Error=   9.2690379142344780E-05 3FF184C58969992D0000000000000000
Temp=    1.1047787599936748E+05 400FAF8DE0417E9AA9B3E00000000000
Flux=    6.9442838715461731E+10 402302B1DA0BB7634000000000000000
dtsum=   3.0000000000000001E-03 3FF689374BC6A7EFA000000000000000
Time =   2.0200000000000001E-01 3FFC9DB22D0E56042000000000000000
Trips=   3.1979851464308495E+00 4000995793C881D39700000000000000
Cntrl=   3.7058322760846792E+06 4014C45F42356BE26176967E4A6D999A
Rdc:Crnt,Extrp,Mass,Prop,Qual=      0      2      0      2      0
Rpt:Air,DelP,Flip,Jpack,Vpack=      0      0      0      0      0

Dump    2 Advancement=     509 time=4.9999999999999944489E-01
P=       1.1610017826645510E+07 4016624F43A73E149100000000000000
Uf=      1.3706563288738387E+07 4016A24A8693D5848F80000000000000
Ug=      5.3792556235066078E+07 40189A67961E16A52E00000000000000
VOIDg=   2.0127747744320381E+01 4003420B4137FFE69F80000000000000
QUALa=   0.0000000000000000E+00                                0
Boron=   0.0000000000000000E+00                                0
Vf=      2.8891214895245306E+02 400720E98297FFDDD508000000000000
Vg=      9.1675057057330686E+02 4008CA6012B250D8B1C0000000000000
RHSth=   7.5475342950407861E+04 400F26D357CB9911E9C0000000000000
SOLth=   6.7091414384136505E+04 400F06136A15142A4524000000000000
Error=   1.0335807997476587E-04 3FF1B183E4B070579000000000000000
Temp=    1.0939814425864998E+05 400FAB5624EE2287CFAFD00000000000
Flux=    2.7820142401306078E+07 4017A8806E66BBFEC000000000000000
dtsum=   3.0000000000000001E-03 3FF689374BC6A7EFA000000000000000
Time =   1.0000000000000000E+00 3FFF000000000000000000000000000
Trips=   2.3019990000000000E+00 400026A7E73A365CB400000000000000
Cntrl=   8.6401335729741725E+05 4012A5E1AB6EFAFE408A91514BA66667
Rdc:Crnt,Extrp,Mass,Prop,Qual=      0      2      0      2      0
Rpt:Air,DelP,Flip,Jpack,Vpack=      0      0      0      0      0

CPU Time=  2.7300000000000002E-01 size     3013
```

Sequential Verification Theory – Basics

- Verification has two parts
  - Detection – Finding differences between versions for given input
  - Coverage – Exercising a wide variety of code capability via input

- Examine Detection first

- *Is comparing just* ***primary*** *variable* $\boldsymbol{L_1}$***-norms*** *sufficient?*

Sequential Verification – Questions

Comparing just **primary** variable $\mathbf{L_1}$**-norms** – 3 possible sources of error:

1. Calculations could differ only on timesteps not dumped to file
2. Two different arrays may have the same L_1-norm
3. It catches differences in primary and secondary variables (due to feedback into system solved for primary vars.), but not tertiary vars.

Answer to 1: Once a calculation has a difference, the difference does not disappear in later advancements (generally grows)

Answer to 2: Well-posed problems admit only one solution, so occurs only when quadruple precision sum insufficient (34 places)

Complex answer to #3…

Sequential Verification Theory

- Statistical Hypothesis Testing
  - H_0: For every test case “i” in the test suite, the two corresponding runs produce the same calculations
  - A_0: Code calculations are different some test case i
- Hypothesis Testing Table for Test

	H_0 is true **No differences exist**	**H_0 is false** **Differences do exist**
Accept H_0	Correct Report: “No differences”	Type II Error Miss actual differences
Reject H_0	Type I Error Detect non-existent differences	Correct Report: “Differences found”

- Goal of Hypothesis Testing is to control Type I Error at some level, α (generally 5%, 1%, or lower) while minimizing Type II Error
  - Type I errors are called false positives, Type II are false negatives

Sequential Verification – Application

- Statistical Hypothesis Testing
  - H_0: For every test case "i" in the test suite, the **two corresponding runs** produce the same calculations
- Stated this way because the test applies to more than just simple code runs. Can also test the following code features
  - Restart
  - Backup
  - PVM Coupling
  - MORE…

	H_0 is true No differences	**H_0 is false Differences**
Accept H_0	Correct	Type II Error
Reject H_0	Type I Error	Correct

Sequential Verification Theory

- Theorem 1: **SV** Verification File Test has level of significance, **α = 0**

  It always accepts the null hypothesis when it's true

  No Type I Error. No false positives.
- Interpretation: *If properly programmed,* ***SV*** *test will never report nonexistent code bugs*
  – No false positives
- Corollary: For testing restart, backup and PVM, the **SV** Test has level of significance, **α = 0**

What about Type II error?

Does ***SV*** *ever miss differences?*

	H_0 is true No differences	H_0 is false Differences
Accept H_0	Correct	Type II Error
Reject H_0	Type I Error	Correct

SV Theory

- Theorem 2: If L_1-norm calculated in quadruple precision & $N > 3$, then
  $P(\text{Accept } H_0 \mid \text{primary or secondary variables differ}) < 10^{-18}$

- The **diffem** test that applies the "diff" utility to compare output files to examine tertiary variables.
- Combine **SV** (Sequential Verification) test with **diffem**
- Theorem 3: $P(\text{Type II Error} \mid \text{SV \& diffem find no difference}) < 10^{-5}$

Probability of missing an actual error is **0.001%**

Recall: *Must program properly!*

	H_0 is true No differences	H_0 is false Differences
Accept H_0	Correct	Type II Error
Reject H_0	Type I Error	Correct

Verification Improvements – DETECTION

PROGRAM PROPERLY

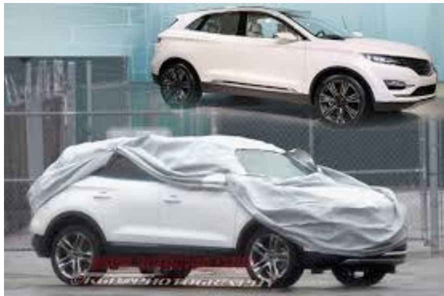
- Verification **programming errors** discovered and corrected
  - In the placement of the calls to verification subprograms
  - In the implementation of backup testing

RELAP5-3D Corrections/Improvements via verification

- Numerous issues with code **backup** were discovered in RELAP5-3D
  - Variables not saved/restored in subroutine MOVER
  - Some variables could not be saved/restored in MOVER had to be backed up elsewhere
- Variables missing from the **restart** file were identified and added
- **R5-Exec** issues were corrected
  - Time exchanges with RELAP5-3D were found and fixed
  - Time calculations were improved to quadruple precision as needed

Verification Improvements – COVERAGE

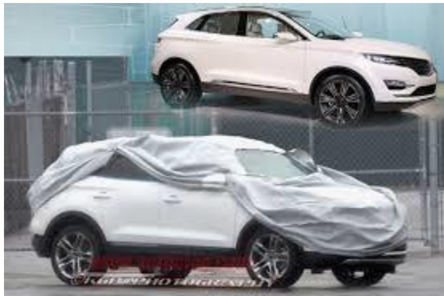
- **Coverage** is design and inclusion of **tests** that exercise code **features**
  - Nearly 200 code features are tested
  - Differences can be detected only by test cases
- Verification Test suite was expanded by 22 input decks
  - Now 65 input decks and 195 cases
  - Added many new input decks for PVM
  - Added tests that had revealed the issues listed on previous slide
- Features were added to the Makefile to test each capability by itself and in groups (such as PVM base cases and restart)



Verification Improvements – COVERAGE

Categories of covered code **features**

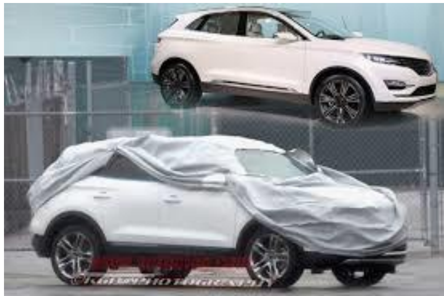
- Hydrodynamic components: pipes, separators, etc.
- Control volume flags: thermal stratification, mixture level, etc.
- Additional wall friction options: shape factor, viscosity ratio, etc.
- Junction flags: jet junction, CCFL, etc.
- Junction form loss: constant, abrupt area change, etc.
- Heat structure geometry type: rectangular, cylindrical, spherical
- Heat structure boundary conditions: adiabatic, convective, etc.
- Heat source options: radial factor shape, table, etc.
- Material properties: built-in, user input (functions and tables)
- Control Functions: arithmetic operations, controllers, etc.



Verification Improvements – COVERAGE

Categories of covered code **features**

- Trips: logical or variables
- General Tables: power, temperature, etc.
- Reactor kinetics: point, nodal
- Decay heat: No decay heat, ANS/ANSI Standard options
- Equation solvers: BPLU, PGMRES, LSOR, Krylov, etc.
- Timestep integration schemes: semi-implicit, nearly-implicit.
- Covered code features that do not fit these categories
  - noncondensables
  - cases with or without boron tracking
  - Certain developmental (card-1) options



COVERAGE – New Input

Input Model	Description
cpl_det	A simplified version of TYPPWR (test 40) that tests the detector model with pt. kinetics
cpl_det_new	Same as cpl_det (test 51) with modified weighting factors and attenuation coefficients.
cpl_new_sa	Version of TYPPWR (test 40) that tests detector model w nodal kinetics
cpl_pvm_core	Christensen model domain decomposed into two semi-implicitly coupled regions, one with the center of the pipe representing the core, the other with the upper and lower portions.
cpl_pvmcs	Edward's pipe problem adapted to test control system coupling
cpl_pvmeda	Edward's pipe problem split in half to test asynchronous coupling
cpl_pvmedca	Edward's pipe problem split in half to test asynchronous explicit conserving coupling
cpl_pvmedcs	Edward's pipe problem split in half to test synchronous explicit coupling
cpl_pvmnd	A version of TYPPWR (test 40) that tests nodal kinetics coupling
cpl_pvmnonc	Parallel pipes tests multiple connections to a coupling TDV and multiple noncondensables
cpl_pvmpt	A version of TYPPWR (test 40) that tests point kinetics coupling

COVERAGE – New Input

Input Model	Description
det	Tests the detector model.
det_new	Tests the detector model.
do_nothing	Tests if zero flow and zero heat transfer are maintained in a rectangular solid of 3x5 vols. constructed of 5 volume pipes connected by multiple junctions.
ht_expl_fluid	Tests explicit fluid-to-heat structure coupling
ht_imp_fluid	Tests implicit fluid-to-heat structure coupling
nothing_trans	Tests moving problems translational acceleration specified by both periodic and table input in a 3x3x5 rectangular solid built of 5 volume pipes connected by multiple junctions.
pvmcore	Tests ability of RELAP5-3D to run the vessel interior of a modified Christensen model[8, 9].
pvmcs	Edward’s pipe problem adapted to test control system
pvmnonc	Parallel pipes tests multiple connections to TDV and multiple noncondensables
pvmpt	A version of TYPPWR (test 40) that tests point kinetics
tdvtdj	Tests multiple connections to a TDV.

Future Sequential Verification Improvements

- Other capabilities to verify
  - Multi-case
  - Multi-deck
  - Input
    - Renodalization
    - Restart table & control variable deletion/addition
    - Should fail testing – compare output against comments in file
  - Physics-based testing:
    - activation/deactivation of models
    - Track activivation/deactivation correlation w failed time step