Testovací strategie

Tento dokument specifikuje pravidla a automatické mechanismy použité pro testování programu Flow123d během jeho vývoje. Dokument se vztahuje k vývoji verzí Flow123d 2.1.x.

1. Požadavky směrnice VDS 30

V této sekci sumarizujeme požadavky směrnice VDS na testování a způsob jejich naplnění v rámci vývoje Flow123d. Zbytek dokumentu pak popisuje realizaci uvedených testovacích podmínek.

1.1. Testovací podmínky

VDS 30: “Testovací podmínky určují, jak se má produkt testovat pro splnění požadavků kvality. Vyžadují testování vlastností, které jsou slíbeny v popisu produktu. Programy se musí testovat ve všech konfiguracích, které jsou vyjmenovány v popisu produktu a odsouhlaseny Komisí. Jestliže existuje několik variant, musí být testována každá. Popis produktu, uživatelská dokumentace a programy, musejí být testovány s ohledem na splnění požadavků a doporučení předchozích odstavců této kapitoly.”

Tento požadavek je realizován následovně. Množina integračních testů pokrývá všechny funkcionality popsané v dokumentaci. Zejména je pro každý model otestována správná odezva na jednotlivé fyzikální parametry, správnost všech podporovaných výstupních polí, a správná funkce všech typů okrajových podmínek. Pro každý integrační test je definován cíl testu a zdroj referenčního řešení (analytické řešení nebo kontrola klíčových hodnot). Všechny integrační testy musí být spustitelné a funkční na všech cílových platformách (konfiguracích), tj. Windows 64 bit a Linux 64 bit. Pro otestování obecně použitelných mechanismů, např. čtení sítě, vstupy polí, výstupy polí, jsou definovány samostatné integrační testy s využitím pouze některého z modelů. Pro klíčové třídy (síť, fields, vstupy, výstupy) jsou vytvořeny jednotkové testy, které též musí být funkční na všech cílových platformách.

1.2. Testovací protokol

VDS 30: “Testovací protokol má o každém testování obsahovat dostatek údajů, takže lze testování podle něj opakovat. Musí obsahovat:

- plán testování nebo specifikaci testování s testovacími případy (přičemž každý testovací případ má svůj cíl)
- všechny výsledky, spojené s testovacími případy, včetně všech poruch, které se v průběhu testování objevily

- totožnost personálu, který provedl testování”

Tyto požadavky směrnice jsou naplněny následovně. Testovací případy jsou dány zejména vstupními soubory integračních testů, které jsou součástí každé instalace. Každý integrační test má definován svůj cíl přímo ve hlavičce svého vstupního souboru. Vzhledem k jejich automatickému spouštění a požadavku na jejich funkčnost během vývoje nejsou protokolovány jejich výsledky. Taktéž není protokolován testující personál, nicméně každý test má uvedeného autora, který definoval jeho cíl a zdroj referenčních dat.

Kontrola pokrytí všech funkcí integračními testy je prováděna automaticky pomocí skriptu, který pro každý klíč ve vstupním souboru vyhledá integrační testy, které tento klíč používají včetně uvedení použité hodnoty.

1.3. Zpráva o testování

VDS 30: “Ve zprávě o testování musí být sumarizovány předmět a výsledky zkoušky (jak jsou uvedeny v záznamech o zkoušce). Zpráva o testování musí mít následující obsah:

1) Označení produktu

2) Počítačové systémy použité pro zkoušení (hardware, software a jejich konfigurace)

3) Použité dokumenty (včetně jejich označení)

4) Výsledky testování popisu produktu, uživatelské dokumentace, programů a dat

5) Seznam neshod s požadavky

6) Seznam neshod s doporučeními

7) Datum ukončení testování

Označení zprávy o testování (místo testování, označení produktu, datum zprávy o testování) a celkový počet stran musí být uvedeny na každé stránce zprávy o testování.”

Zpráva o testování zahrnuje výsledky testování programu na validačních úlohách.

2. Testovací podmínky

Testování programu Flow123d zahrnuje jednak *automatické testování* a dále *manuální testování*. Průběžné automatické testování je součástí vývojového procesu. Ten zahrnuje průběžné sestavování (continuous integration) jehož součástí je i vyhodnocení testů. Automatické testy jsou dvojího druhu: *jednotkové testy* (unit test) a *integrační testy* (integration test).

Jednotkové testy jsou specializované malé programy, které testují správnou funkci jednotlivých tříd programu obvykle izolovaně od ostatních částí programu. Jednotkový test je úspěšný pokud jsou při jeho běhu splněny všechny kontrolní podmínky, které test definuje. To zahrnuje i správné ošetření chybových stavů. Pomocí jednotkových testů jsou testovány pouze třídy, které mají obecnou použitelnost při vývoji numerických modelů. Třídy implementující jednotlivé numerické modely nejsou testovány pomocí jednotkových testů, nebo jen velmi omezeně.

Integrační testy naopak testují program jako celek. Integrační test je tvořen vstupními daty a referenčními výstupními daty. Program testem projde pokud dokončení výpočet bez chyb a jeho výstupy se od referenčních výstupů neodchylují více než o danou toleranci.

Při vhodně vytvořeném systému integračních testů není potřeba provádět ruční testování funkčnosti programu, neboť to je testováno průběžně automatickými integračními testy. Manuálně je však třeba testovat použití programu na komplexních reálných úlohách (*testy na validační úlohách*).

Systém automatických testů je prováděn a vyhodnocován na několika úrovních vývoje. Pro každou vývojovou větev je pro každou sadu změn provedeno ladící sestavení a systém testů spuštěn a vyhodnocen v ladicím režimu. Po každé změně v hlavní větvi (master) nebo větvích pro cílové verze (např. release_2.1.0) je provedeno kompletní optimalizované produkční sestavení včetně používaných knihoven, je proveden a vyhodnocen systém testů, sestavena generovaná část dokumentace a sestaven instalační balíček. Výsledkem sestavení je obraz pro nástroj Docker (Docker Toolbox). Tento obraz je dále doplněn o dokumentaci a testovací úlohy a zabalen jako instalační balíček zvlášť pro systémy Linux a pro systémy Windows.

V následujících sekcích je podrobně popsána metodika testování pro jednotlivé druhy testů.

2.1. Systém jednotkových testů

Cílem jednotkových testů je testování správné funkce jednotlivých tříd. V ideálním případě by test měl plně otestovat, že daná třída funguje podle své dokumentace. Nutnou (nikoliv postačující) podmínkou tohoto stavu je 100% pokrytí kódu, t.j. v průběhu jednotkového testu se projdou všechny řádky kódu, které při překladu vedou na nějaké instrukce. Míra pokrytí kódu Flow123d pomocí jednotkových testů i pomocí integračních testů je pravidelně automaticky vyhodnocována na integračním serveru. Jelikož plné pokrytí je nad možnosti malého vývojového týmu, je snaha dobře pokrýt alespoň obecně používané třídy. Jednotkové testy je nutné provádět jak v rámci ladicích sestavení tak pro produkční sestavení. Jednotkové testy jsou spouštěny s daným časovým limitem pro svůj běh a limitem na použitou paměť, v případě překročení limitů je program ukončen. Cílem je, aby chyby vedoucích např. k zacyklení programu nezpůsobily ukončení testovacího procesu.

Jednotkové testy jsou v podadresáři 'unit_tests', který je dále členěn stejně jako adresář se zdrojovými kódy. Jednotkové testy pro zdrojový soubor 'src/<module>/<class>.cc' jsou v souboru 'unit_tests/<module>/<class>_test.cpp'. Každý adresář s jednotkovými testy obsahuje také soubor 'CMakeLists.txt', kde je nutné definovat testy, které se mají provést. K tomu slouží CMake makra 'define_test(class_name)' a 'define_mpi_test(class_name n_processes)'. První makro definuje sekvenční test, druhé paralelní test, kde parametr 'n_processes' udává počet použitách MPI procesů. Obě makra mají ještě volitelné parametry 'time_limit' a 'memory_limit' pro zadání časového a paměťového limitu testu. Jelikož absolutní doba trvání testu se liší podle překladače a hardware, jsou limity interně přenásobeny faktorem, který je určen měřením pevného kódu během konfigurace (tento faktor je z podstaty odlišný pro ladicí a produkční sestavení). Vynucení limitů je realizováno pomocí skriptu 'exec_limit.py'.
Zdrojový kód testů využívá knihovnu GoogleTest s několika rozšířeními pro možnost paralelních testů pro testování tříd využívajících MPI. Více viz. 'unit_tests/HOWTO_TEST'.

Některé třídy implementují výkonově kritické funkce a je proto vhodné testovat i jejich rychlost. K tomu slouží výkonové jednotkové testy. Pokud tyto testy slouží pouze k testování výkonu a ne funkce a zejména pokud jejich provedení trvá delší dobu, je vhodné je provádět pouze pro produkční optimalizované sestavení. Části jednotkových testů, které slouží k testování výkonu jsou proto pod podmíněným překladem a aktivují se nastavením makra 'RUN_UNIT_BENCHMARKS=1'. Toto makro je automaticky nastaveno v hlavním 'CMakeLists.txt' při produkčním (release) sestavení.

Spouštění jednotkových testů (jakož i ostatních testů) se provádí pomocí skriptu 'exec_parallel' popsaného v přízoe A. Pro jednodušší spouštění unit testů během vývoje generuje hlavní 'CmakeLists.txt' v každém adresáři s jednotkovými testy make pravidla pro spouštění jednotlivých testů.

2.2. Systém integračních testů

Systém integračních testů je klíčový pro průběžné ověřování správné funkce programu. Je navržen tak, aby zaručil požadavky směrnice VDS30 na správnou funkci vlastností specifikovaných v dokumentaci. Dále slouží pro ladění nových algoritmů a některé testy jsou podrobně zdokumentovány, aby mohly sloužit jako výukové úlohy pro uživatele. Všechny požadavky kladené na systém integračních testů spolu se způsobem jejich realizace jsou prezentovány v sekci 2.2.3.

Systém intergračních testů se skládá z: testovacích úloh (vstupní data a referenční výstupní data), sada skriptů pro spouštění a vyhodnocování testů, sada pravidel zajišťující automatické spouštění testů v různých fázích vývojového procesu. Struktura testovacích úloh včetně analýzy pokrytí je popsána v sekci 2.2.4. Pomocné skripty jsou popsány v příloze A, pravidla pro automatické spouštění testů jsou popsána v rámci sekce 2.2.3.

2.2.3. Požadavky na systém integračních testů

Systém integračních testů musí splňovat následující požadavky.

- **Přenositelnost.** Každý integrační test lze spustit (simulace i vyhodnocení výsledků) pod systémy Linux a Windows. Všechny podpůrné skripty zajišťující spouštění testů a ověřování výsledků jsou realizovány v jazyku Python a jsou přenositelné mezi Linux a Windows. Přenositelnost je zajištěna při spouštění uvnitř kontejnerů nástroje Docker.
- **Nezávislost.** Každý integrační test je funkční i mimo prostředí, ve kterém je sestaven instalační balíček. Instalační balíčky jsou po sestavení a publikaci testovány v odděleném prostředí, které neobsahuje žádné vývojové nástroje a specializované knihovny.
- **Pokrytí.** Jelikož vstupní soubor se skládá z několika modulů, které lze konfigurovat nezávisle, roste exponenciálně počet možných vstupů s počtem modulů. Není tedy prakticky možné otestovat všechny možné vstupy. Testovací úlohy jsou tedy navrženy tak, aby: testovaly obecné mechanismy používané různými fyzikálními modely, testovaly detekci chyb, testovaly samostatnou funkci jednotlivých fyzikálních modelů, testovaly vybrané kombinace fyzikálních modelů. Zároveň je sledováno: pokrytí zdrojového kódu (které části zdrojového kódu byly aktivní alespoň jednom testu), pokrytí stromu struktury vstupů. Skript 'ist_coverage.py' je používán k ověření, že každá elementární volba vstupního souboru je použita v alespoň jednom testu.
- **Specifičnost.** Každý test by měl mít jasně specifikovaný cíl. Popis cíle a popis očekávaných výsledků je součástí vstupního souboru každé testovací úlohy.
- **Kompaktnost.** Testy jsou uloženy v repozitáři spolu se zdrojovým kódem, repozitář má velikostní limit a limit na velikost jednotlivých souborů. Navíc práce s velkým repozitářem a velkými soubory výrazně zpomaluje průběh testů a tím i samotný vývoj. Doporučení pro minimalizaci dat:
    - Sdílení sítí a dalších vstupních dat mimo YAML soubory.
    - Referenční data jednoho testu (YAML souboru) pod 1MB, ideálně do 100kB
    - Vystupovat pouze pole podstatná pro cíle testu.
    - Obecné funkcionality testovat obecně a neopakovat v testu každé rovnice.
- **Užitečnost.** Systém testů musí být snadno použitelný pro ladění. Možnost volat testy samostatně i hromadně, na dané množině procesorů, pod valgrindem, se sledováním pokrytí, paměťové náročnosti. Tato variabilita je zajištěna pomocí python scriptu 'runtest.py' vyvinutého speciálně pro spouštění testů.
- **Udržovatelnost.** Vytváření referenčních dat. Preference jednoduchých mechanismů, před komplexními skripty a GEO soubory. Důraz na dokumentaci. Skript 'runtest' umožňuje aktualizaci referenčních dat.

- **Příklad použití.** Některé testy jsou zpracovány včetně podrobného komentáře fyzikálního pozadí a přípravy vstupů, tak aby mohly sloužit jako součást dokumentace. Tyto úlohy slouží zároveň jako tutoriály, které jsou součástí uživatelské dokumentace.

2.2.4. Cíle integračních testů

Obecné testy

- funkce parametrů na příkazové řádce (výpisy struktury vstupů, logování, čtení CON souboru, funkce -i a -o parametrů)
- funkce výstupních metod (asi raději formou unit testu) výstup pro ruzna pole do různých formátů
- kompatibilita vstupu a výstupu v GMSH formátu

Testy proudění

- kompatibilita vstupu a výstupu v GMSH formátu
- Nezávislost na síti. 1D konstantní rychlostní pole pro různé sítě:
  1D, 2D, 3D, homogenní i heterogenní velikosti elementů.
  Invariance vůči rotaci, zrcadlení sítě, posun času. Analytické řešení.
- Funkce okrajových podmínek, otestování všech typů podmínek a jejich parametrů.
- Funkce objemových parametrů.
- Správná reakce na časovou změnu všech parametrů.
- Kombinace dimenzí: ekvivalence pukliny a kontinua pro puklinu kolmou a paralelní na delší osu kanálu.
- Větvení puklin.
- Oblast s vnitřní hranicí.
- Test robustnosti nastaveni solveru.

Testy difúzního transportu

- Správná rychlost transportu. 1D a ekvivalentní 2D a 3D oblasti. Transport po částech konstantním rychlostním polem, heterogenní porozita. Analytické řešení.
- Funkce zdrojů.
- Funkce okrajových podmínek
- Funkce difúze a disperze.
- Porovnání puklinového modelu s ekvivalentním kontinuem.
- Porovnání Explicitní metody a DG z hlediska numerické difúze a rychlosti výpočtu. Sada úloh s různou heterogenitou (rychlosti, velikosti elementů, porozity)
- Úloha 2D-1D (puklina napříč kanálem, nebo skloněná) s různou porozitou na puklině a v matrici.

- Transport rychlostním polem s nenulovou divergencí.

Testy konvektivního transportu

- Správná rychlost transportu. 1D a ekvivalentní 2D a 3D oblasti. Transport po částech konstantním rychlostním polem, heterogenní porozita. Analytické řešení.
- Funkce zdrojů.
- Funkce okrajové podmínky (součást ostatních testů)
- Porovnání puklinového modelu s ekvivalentním kontinuem.
- Úloha 2D-1D (puklina napříč kanálem, nebo skloněná) s různou porozitou na puklině a v matrici.
- Transport rychlostním polem s nenulovou divergencí.
- Transport pro porozitu proměnnou v čase.

Testy reakčního členu

- duální porozita
- sorpce (všechny druhy)
- rozpady a lineární reakce, oba ODE řešiče
- komplexní kombinace : duální porozita, lineární sorpce, rozpady

Testy vedení tepla

Není nutno testovat základní vlastnosti, které jsou sdílené s difúzivním transportem.

- Funkce objemových parametrů.
- Funkce okrajových podmínek.