DOKUMENTACJA TECHNICZNA

INTEGRACJA OPROGRAMOWANIA ASPEN™ I FLUENT™

Piotr Baniukiewicz

wersja 1.0

data: 30 października 2010

Spis treści

ZMIANY

ver. 0.9 pierwsza wersja Release

ver. 0.9.3 Dodano:

- rozdział w dokumentacji dotyczący błędów numerycznych
- dyrektywę #PHASES
- dodano obsługę „velocity-inlet”
- zmieniono działanie paru instrukcji celem przygotowania do obsługi mieszanin
- poprawiono kompatybilność z Ansysem 12.xx

ver. 1.0 wersja obsługująca dwa wejścia oraz mieszaniny - dwa wejścia i dwa składniki we Fluencie. Zmieniono:

- usunięto niektóre instrukcje i dodano nowe

WSTĘP

Celem projektu była integracja oprogramowania Aspen™ oraz Fluent™ przy wykorzystaniu otwartego interfejsu komunikacji międzyprogramowej CAPE-OPEN. Projekt został wykonany w ramach projektu badawczego nr 507-10-013-9900/7 na podstawie umowy o dzieło z prawami autorskimi.

1. Zamawiający: prof. dr hab. Zdzisław Jaworski
2. Wykonawca: dr inż. Piotr Baniukiewicz

Treść umowy: "**Przygotowanie i skuteczne uruchomienie oprogramowania pomostowego (interfejsu) do integracji programów do symulacji procesów oraz numerycznej dynamiki płynów w oparciu o programy Aspen Plus i Fluent**"

Niniejsza instrukcja została podzielona na dwie części, pierwszą zawierającą informacje teoretyczne mogące przyczynić się zrozumienia mechanizmów działania narzędzi opartych o Cape-Open oraz drugą dotyczącą konkretnie realizowanego zadania.

Omówienie problemu i wyników

Głównym problemem, który należało przezwyciężyć było to, że Fluent™ nie wspiera standardu Cape-Open. W tym przypadku zaistniała potrzeba znalezienia innego sposobu przesyłania i odbierania danych z Fluenta. Powstały komponent jest więc nie tylko interfejsem pomiędzy tymi dwoma programami ale także interfejsem pomiędzy samym standardem Cape-Open a innymi metodami importu i eksportu danych, które normalnie nie są wspierane przez Cape-Open. Skuteczność tego rozwiązania oraz jego uniwersalność została potwierdzona nie tylko poprzez współpracę z programem Aspen™ ale także z innymi programami do symulacji procesów chemicznych, n.p. COFE (http://www.cocosimulator.org/). Jakakolwiek wzmianka w niniejszej dokumentacji o programie Aspen w kontekście Cape-Open będzie miała także odniesienie do dowolnego innego programu pracującego w tym standardzie.

Kompatybilność z Fluentem

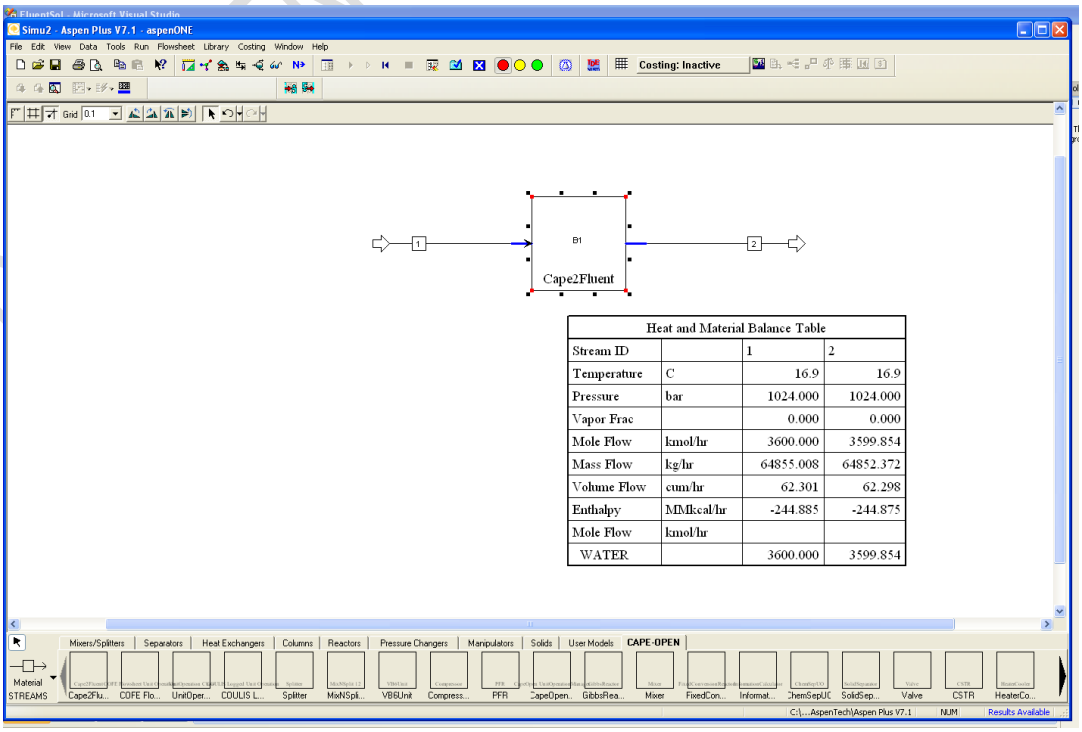
Ze względu na różnice w adresowaniu pamięci oraz kontekście pracy programu w systemie Windows nie jest możliwe "mieszanie" programów przeznaczonych na architekturę 32bit z programami 64bit. Oznacza to, że jeśli Aspen pracuje w 32 bitach, to kontrolka też musi być 32 bitowa i Fluent także. Tak, więc opracowany komponent będzie współpracował jedynie z Fluentem w wersji 32bit.

Ze względu na różnice w nazwach procesów komponent będzie współpracował jedynie z określonymi wersjami Fluenta. Zmiana wersji Fluenta będzie wymagała dokonania odpowiednich wpisów w pliku konfiguracyjnym, co może być wykonane samodzielnie przez użytkownika. W przypadku, gdy zmienią się funkcje realizowane przez Fluenta, będzie to wymagało zmian w samym kodzie kontrolki.

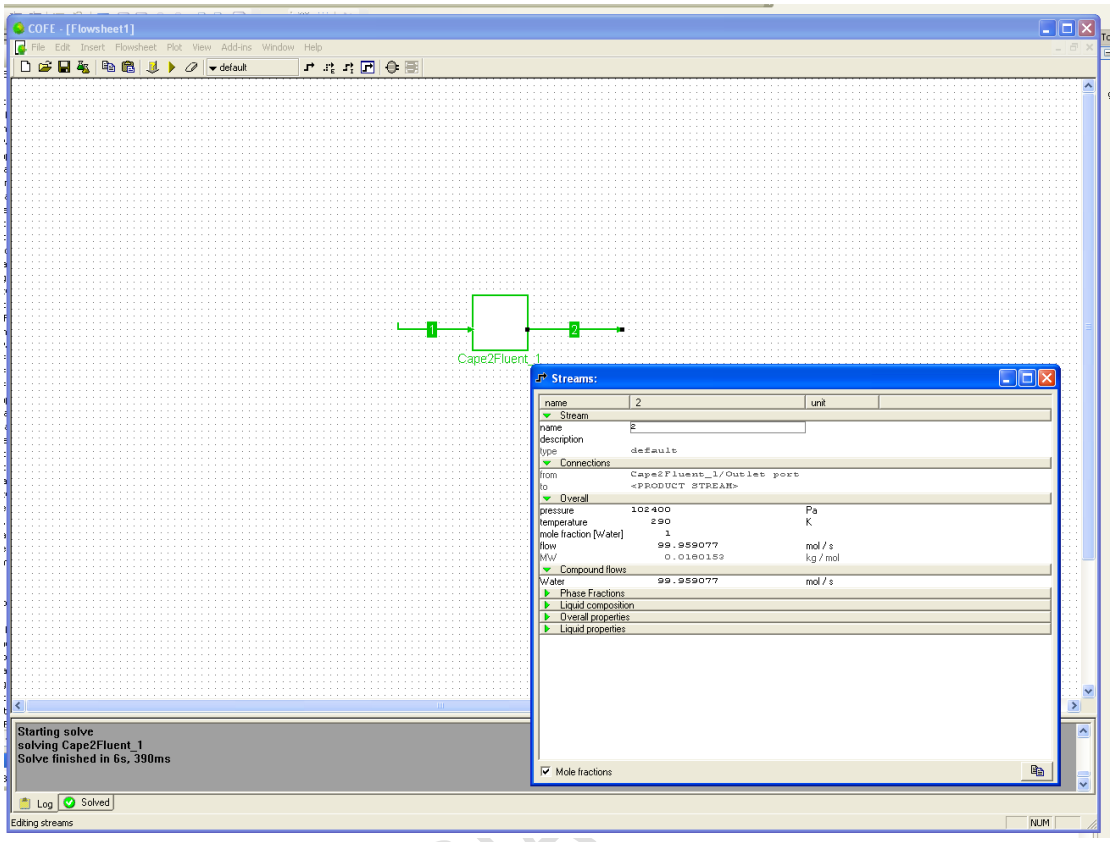
Od wersji 0.9.3 docelową wersją Fluenta będzie Ansys Fluent 12.xx.xx.

Wynik pracy

Wynikiem pracy jest komponent o nazwie Cape2Fluent umożliwiający współpracę tych dwóch programów oraz dokumentacja do niego.



Rys. 1 Widok okna programu Aspen Plus z kontrolką Cape2Fluent oraz wynikami symulacji



Rys. 2 Widok okna programu COFE z kontrolką Cape2Fluent oraz wynikami symulacji

PODSTAWY DZIAŁANIA KOMPONENTU CAPE2FLUENT

Model COM+ - programowanie komponentowe

COM (ang. Component Object Model) – standard definiowania i tworzenia interfejsów programistycznych na poziomie binarnym dla komponentów oprogramowania wprowadzony przez firmę Microsoft wraz z bibliotekami zapewniającymi podstawowe ramy i usługi dla współdziałania komponentów COM i aplikacji.

Podział projektu na logiczne komponenty jest podstawą analizy i projektowania obiektowego. Jest także podstawą oprogramowania komponentowego, tworzonego z dostarczanych w postaci binarnej (w przeciwieństwie do kodu źródłowego), wielokrotnie wykorzystywanych fragmentów, które mogą być stosunkowo łatwo łączone z komponentami pochodzą i od innych producentów. Co niezwykle ważne, technika tworzenia oprogramowania opartego na komponentach nie narzuca struktury aplikacji. Jest raczej modelem umożliwiającym programowanie, wykorzystywanie i niezależną ewolucję binarnych komponentów programowych niezależnych od aplikacji, która je wykorzystuje, a także od języka programowania, który został użyty do ich utworzenia.

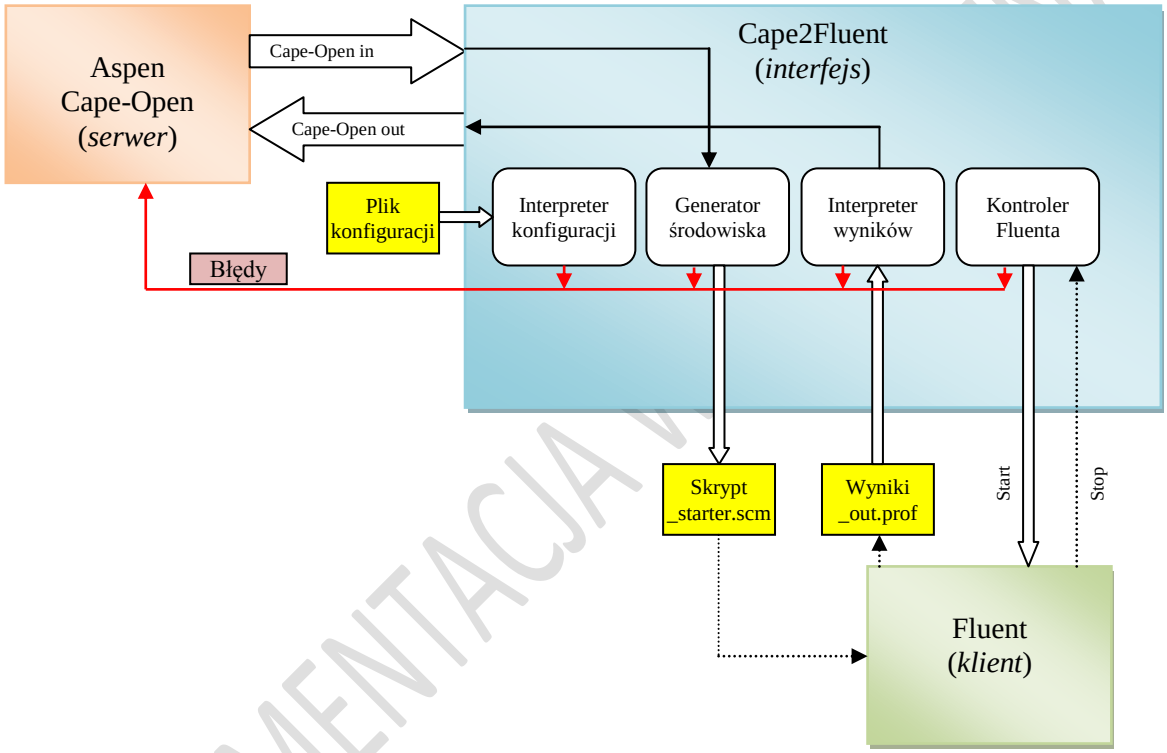
Standard Cape-Open

Standard Cape-Open jest zbiorem zunifikowanych standardów umożliwiających wymianę informacji pomiędzy programami do modelowania procesów chemicznych. Standardy te są oparte o technologię programistyczną COM+ lub CORBA. Cape-Open ma charakter otwarty i niekomercyjny.

W świetle poprzedniego rozdziału wprowadzającego do technologii programowania komponentowego, standard Cape-Open jest niczym więcej jak zbiorem interfejsów mających zapewnić współdziałanie aplikacji implementujących ten standard. W tym przypadku zbiór interfejsów oznacza istnienie abstrakcyjnych, to jest nie posiadających żadnych zaimplementowanych metod, klas. Interfejs Cape-Open wyznacza po prostu jakie funkcje powinny być zaimplementowane w komponencie oraz jakie funkcji są realizowane w programie aby te dwa moduły mogły ze sobą współpracować. Obowiązuje tu zasada wzajemnego zaufania - komponent "w ciemno" wywołuje pewne funkcje programu, gdyż ich istnienie wymusza właśnie standard i na odwrót, w komponencie muszą zostać zaimplementowane pewne funkcje o określonej nazwie i parametrach, które będą wywoływane przez program. Poszczególne funkcje są pogrupowane w Interfejsach, dzięki czemu możliwy jest dostęp do nich z poziomu innych programów (technologia COM+). Należy pamiętać, że mówimy tutaj o dostępie do już skompilowanych kodów. Nie jest to ta sama sytuacja jak w przypadku pisania programu gdzie mamy dostęp do funkcji na poziomie kodów źródłowych.

Architektura komponentu Cape2Fluent

Komponent Cape2Fluent ma za zadanie integrować programy Aspen™ i Fluent™ z wykorzystaniem standardu Cape-Open. Niestety jedynie Aspen™ oficjalnie wspiera ten standard, tak więc powstały komponent jest tak naprawdę interfejsem pomiędzy generalnie standardem Cape-Open a Fluentem. Oznacza to że po jednej stronie komponentu może być dowolny program wspierający ten standard a po drugiej stronie musi być zawsze Fluent™. Brak wsparcia Cape-Open przez Fluenta powoduje także szereg problemów, które zostaną omówione w dalszej części instrukcji. Sposób współpracy pomiędzy standardami został przedstawiony na Rys. 3.



Rys. 3 Schemat blokowy wymiany danych pomiędzy Aspenem a Fluentem za pośrednictwem kontrolki Cape2Fluent. Strzałki grube - akcje główne podejmowane przez serwer lub interfejs, strzałki cienkie - akcje wewnętrzne, strzałki przerywane - akcje podejmowane przez klienta. Bloki żółte - zewnętrzne pliki tworzone przez użytkownika, interfejs lub klienta.

Serwer

Przepływ danych pomiędzy programami wygląda następująco: po wstawieniu kontrolki[1] do schematu procesu technologicznego w Aspenie, kontrolka wczytuje plik konfiguracyjny. Czynność ta ma miejsce jednorazowo przy inicjalizacji kontrolki, tak więc jakiekolwiek zmiany w pliku konfiguracyjnym będą wymagały ponownego wstawienia kontrolki do schematu technologicznego. Podobnie jeśli w procesie analizy konfiguracji wystąpią błędy. Po ustawieniu konfiguracji symulacji w Aspenie i wykonaniu symulacji Aspen przekazuje cztery podstawowe parametry materiału do kontrolki. Tymi parametrami są:

1. Temperatura [-]
2. Ciśnienie [-]

[1] Pisząc komponent, kontrolka lub interfejs autor zawsze miał na myśli interfejs Cape2Fluent

3. Udział [-]
4. Strumień całkowity [mol/s]

Brak jednostek przy pierwszych trzech parametrach oznacza, że przekazywane są one liczbowo, bez uwzględnienia ich jednostek tak jak zdefiniuje je użytkownik w Aspenie (na przykład, 300K i 300 ˚C zawsze pojawi się w kontrolce, jako bezwymiarowa liczba 300)[2]. Czwarty parametr zawsze jest przeliczany na mol/s. Po przekazaniu parametrów serwer[3] będzie oczekiwał na informację zwrotną, którą może być błąd lub przeliczone wartości czterech parametrów uzyskane z klienta. Należy tu zwrócić uwagę na bardzo ważna sprawę wynikającą z zastosowanego podejścia do komunikacji interfejsu z klientem - nie ma tu możliwości przekazania informacji o błędzie, który wystąpi poza kontrolką (np we Fluencie podczas symulacji). Jest to duże ograniczenia, którego nie można ominąć, tak więc należy zwracać szczególną uwagę na przygotowanie i przetestowanie modeli Fluenta przed ich użyciem[4].

Interfejs

Koncepcja wymiany danych pomiędzy Fluentem a komponentem Cape2Fluent wykorzystuje pliki skryptów *scm*, które mogą zawierać instrukcje, które z kolei zostaną wykonane przez Fluenta po wczytaniu skryptu. Z poziomu plików *scm* można wykonać wszystkie te same operacje co z poziomu graficznego interfejsu użytkownika. Po zdefiniowaniu parametrów procesu w Aspenie, są one przekazywane do kontrolki, która z kolei generuje odpowiedni skrypt *scm,* następnie uruchamia Fluenta, wczytuje do niego skrypt i odbiera wyniki. W skrypcie zapisane są operacje takie jak:

- Definicja warunków brzegowych - służy przekazaniu wartości parametrów strumienia z Aspena do modelu Fluenta
- Wykonanie symulacji
- Zapisanie określonych wyników na dysk
- Zamknięcie Fluenta

Po wykonaniu symulacji kontrolka oczekuje, że na dysku pojawi się plik *_out.prof* zawierający wyniki symulacji odpowiadające żądanym wartościom czterech wcześniej wspomnianych parametrów materiału. Uruchomienie Fluenta związane jest z przekazaniem sterowania do Fluenta i tymczasowym "zawieszeniu" działania interfejsu i samego serwera.

Komponent Cape2Fluent składa się z czterech podstawowych bloków:

1. Interpretera konfiguracji
2. Generatora środowiska dla programu Fluent
3. Kontrolera Fluenta
4. Interpretera wyników uzyskanych z Fluenta

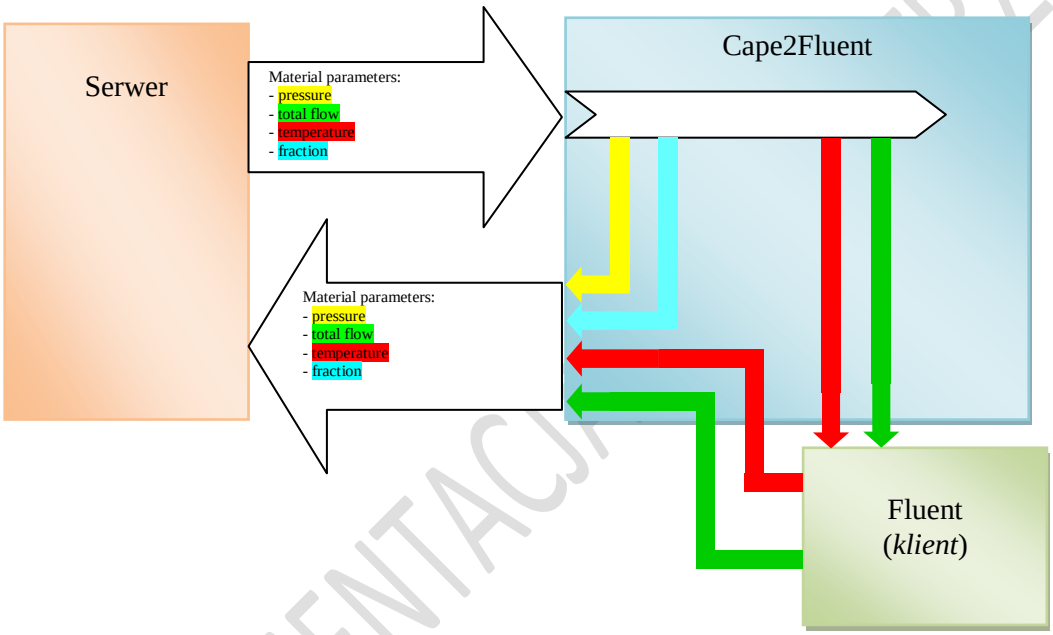
Interpreter konfiguracji ma za zadanie wczytać plik konfiguracyjny stworzony przez użytkownika, zdekodować go oraz wprowadzić do systemu zapisane wartości parametrów konfiguracyjnych[5].

Generator środowiska jest odpowiedzialny za wygenerowanie pliku *_starter.scm* przypisującego zadane z Aspena wartości parametrów materiału do określonej powierzchni wejściowej modelu Fluenta.

[2] Ten problem zostanie poruszony w dalszej części instrukcji
[3] Aspen lub inny program wspierający Cape-Open
[4] Problem omówiony w rozdziale Błędy i ich obsługa
[5] Patrz rozdział "Instrukcja użytkowania Cape2Fluent"

Kontroler Fluenta odpowiedzialny jest za uruchomienie programu Fluent i wczytanie do niego przygotowanego wcześniej skryptu. Moduł ten czeka także na zakończenie procesu obliczeń. W związku z brakiem sprzężenia zwrotnego pomiędzy Fluentem a interfejsem, jedynym sposobem na poinformowanie kontrolera, że proces obliczeń się zakończył jest zamknięcie procesu Fluenta. Jest to wykonywane w ostatniej linijce skryptu *_starter.scm*, jednakże jeżeli wcześniej wystąpi jakiś błąd, ta linijka nie zostanie wykonana, Fluent nie zostanie zamknięty i tym samym sterowanie nie zostanie zwrócone do interfejsu i serwera. Sytuacja ta spowoduje zawieszenie całego systemu[6].

Interpreter wyników wczytuje powstały plik z wynikami *_out.prof*, dekoduje go i przekazuje obliczone parametry z powrotem do serwera, jako wyjście z interfejsu. Jeżeli któryś z czterech parametrów strumienia nie był obliczany we Fluencie, zostanie on przepisany bezpośrednio z wejścia kontrolki, tak jak pokazano na Rys. 4.



Rys. 4 Schemat działania w przypadku niepełnego zdefiniowania problemu we Fluencie - tutaj z Fluenta uzyskano jedynie temperaturę i przepływ, pozostałe dwa parametry zostały przepisane z wejścia na wyjście bez zmiany.

[6] Problem omówiony w rozdziale Błędy i ich obsługa

Błędy i ich obsługa

Ze względu na brak sprzężenia zwrotnego pomiędzy Fluentem a kontrolką obsługa błędów jest mocno ograniczona. Jedyną informacją jaką może uzyskać kontrolka o stanie Fluenta jest to czy proces Fluenta jest czy go nie ma. Zakładając, że po skończonych obliczaniach i nagraniu pliku z wynikami Fluent automatycznie zamyka się (poprzez skrypt *_starter.scm*) jest to znak dla kontrolki że można przystąpić do analizy wyników. Nieco łatwiej jest obsłużyć błędy które mogą wystąpić w kontrolce zanim zostanie uruchomiony Fluent. Jednak tutaj wiele zależy od samego serwera, czy jest on w stanie odebrać nadprogramowe informacje z kontrolki. Niestety napotkano trudności w implementacji tego procesu w Aspenie, pomimo ze w programie COFE mechanizm ten działał poprawnie. W związku z tym na chwilę obecną w Aspenie obsługa błędów jest mocno ograniczona, co sprawia, że użytkownik nie dostanie pełnej informacji, co poszło źle a jedynie zobaczy skutki błędu np. jako błędne wyniki lub ich brak. Poniżej przedstawiono listę możliwych błędów oraz ich źródła.

1. Błędy pochodzące od interpretera konfiguracji
    a. Problem z otworzeniem pliku konfiguracyjnego
        i. Przyczyna: Plik zablokowany przez inny proces (np. edytor)
        ii. Przyczyna: Plik nie może być znaleziony
    b. Błędy w skrypcie
        i. Przyczyna: Literówki, błędy składniowe
2. Błędy pochodzące od generatora środowiska
    a. Problem z utworzeniem pliku *_starter.scm*
        i. Przyczyna: Plik zablokowany przez inny proces (np. edytor)
3. Błędy pochodzące od kontrolera Fluenta
    a. Problem z uruchomieniem Fluenta
        i. Przyczyna: Zła ścieżka do pliku fluent.exe podana w pliku konfiguracyjnym
        ii. Przyczyna: Próba uruchomienia Fluenta w innej wersji niż zalecana
        iii. Przyczyna: Próba uruchomienia Fluenta w wersji 64bit
4. Błędy pochodzące od interpretera wyników
    a. Problem z utworzeniem pliku *_out.prof*
        i. Przyczyna: Plik nie został utworzony na skutek błędu we Fluencie
        ii. Przyczyna: Plik nie zawiera wymaganych wielkości fizycznych
5. Błędy pochodzące od Fluenta
    a. Błędy powodujące przerwanie toku obliczeń
        i. Przyczyna: błędnie zdefiniowany model
        ii. Przyczyna: błędnie zdefiniowane nazwy powierzchni, wartości eksportowanych itp. w pliku konfiguracyjnym kontrolki
        iii. Przyczyna: Zablokowany plik _out.dat, który jest kasowany na początku każdej symulacji

Błędy są sygnalizowane na dwa sposoby: albo poprzez wyświetlenie okienka z komunikatem o błędzie albo poprzez przekazanie informacji do programu nadrzędnego o wystąpieniu błędu. Niestety Aspen nie odbiera tych komunikatów i nie interpretuje ich poprawnie, co powoduje całkowity brak informacji co aktualnie dzieje się w komponencie. Planuje się rozwiązanie tego problemu w następnych wersjach.

W przypadku wystąpienia błędu w procesie obliczeniowym prowadzonym we Fluencie nie nastąpi jego poprawne zamknięcie, co jest wymagane do zwrócenia sterowania do komponentu i serwera. W takim przypadku należy ręcznie zamknąć Fluenta. Uzyskane wyniki oczywiście nie będą poprawne.

W razie problemów z Fluentem, można podejrzeć automatycznie generowane pliki *_starter.scm* oraz *_out.dat*, zawierające skrypt inicjalizujący obliczenia oraz ich wyniki. Celem weryfikacji, skrypt inicjalizujący można uruchomić oddzielnie wczytując go do Fluenta poprzez funkcję Read Profile.

Błędy i ich obsługa v wersji 1.0

Od wersji 1.0 Cape2Fluent otrzymał bardziej rozbudowany mechanizm kontroli błędów. Błędy są logowane w pliku ErrorLog.txt tworzonym automatycznie w katalogu c:\. Ponadto są także przekazywane do ASPENA i można je podejrzeć w oknach zawierających wyniki symulacji lub też w pliku dziennika.

INSTRUKCJA UŻYTKOWANIA CAPE2FLUENT

Instalacja

Komponent dostarczany jest pod postacią pliku instalacyjnego, wykonywalnego exe. Instalacja polega na uruchomieniu pliku, wybraniu instalowanych komponentów (zaleca się zostawienie wyboru domyślnego) oraz wybraniu katalogu instalacyjnego (zaleca się krótkie nazwy bez spacji, np. C:\ cape2fluent). W katalogu instalacyjnym zostanie umieszczony plik cape2fluent.dll, dokumentacja oraz plik z parametrami konfiguracyjnymi *params.txt*. Zostaną stworzone także odpowiednie wpisy w menu Start.

Jeżeli zajdzie potrzeba ponownej instalacji programu (np nowszej wersji) zaleca się najpierw odinstalować starą wersję.

Wymagania i filozofia pracy

Komponent Cape2Fluent do poprawnej pracy wymaga:

1. Oprogramowania wspierającego Cape-Open
2. Oprogramowania Fluent™ w wersji 32bit
3. Pliku z parametrami *params.txt* znajdującego się w katalogu instalacyjnym.

Podstawowa konfiguracja komponentu znajduje się w pliku tekstowym *params.txt*. Konfigurację można dowolnie modyfikować uważając na niebezpieczeństwa związane ze słabą obsługą błędów. Konfiguracja jest odczytywana jednorazowo podczas wstawiania komponentu do arkusza programu do symulacji procesów lub tez podczas wczytywania arkusza zawierającego kontrolkę. Jakakolwiek zmiana parametrów wymaga ponownego wstawienia kontrolki do arkusza. W wersji 1.0, jeżeli nastąpiła jakś zmiana w pliku konfiguracyjnym, można ją zaktualizować w ASPENIE wyświetlając okienko kontrolki i wczytując skrypt ponownie.

Plik *params.txt* musi znajdować się w katalogu instalacyjnym programu Cape2Fluent jak i również jego nazwa nie może ulec zmianie. Inaczej mówiąc komponent zawsze będzie poszukiwał pliku *params.txt* w katalogu instalacyjnym.

Oprócz katalogu instalacyjnego, komponent wymaga także katalogu roboczego, w którym znajdować się będą/muszą:

- Automatycznie generowany plik *_starter.scm*
- Plik z wynikami _out.prof
- Model Fluenta (*.cas i *.dat)

Współpraca z Fluentem

Współpraca z Fluentem odbywa się poprzez automatyczną generację plików skryptowych zawierających instrukcje Fluenta. Od użytkownika wymagana jest znajomość słów kluczowych umożliwiających eksportowanie wybranych wielkości. Instrukcje Fluenta służące temu celowi można sprawdzić wczytując analizowany model. Należy zwrócić uwagę, że lista dostępnych komend zależy od wczytanego modelu oraz od ustawionych warunków brzegowych, tak więc zawsze należy sprawdzać daną instrukcję na modelu który zostanie użyty podczas symulacji. Listę słów kluczowych można sprawdzić wpisując w oknie Fluenta file, write-profile, nazwa, ENTER, ENTER. W razie potrzeby można tworzyć własne funkcje eksportujące[7]. Wymagana jest także znajomość nazewnictwa powierzchni wejściowych i wyjściowych w analizowanym modelu.

Jednostki

Interfejs nie otrzymuje żadnych informacji o tym, w jakich jednostkach są podawane cztery podstawowe parametry materiału. Oznacza to, że interfejs przekazuje do Fluenta liczby w takiej postaci, w jakiej je otrzymuje od serwera. Może to powodować trudne do wykrycia błędy symulacyjne. Na przykład w Aspenie temperaturę definiujemy, jako 100˚C, do Fluenta jest przekazywana liczba 100, która jest interpretowana, jako 100K a nie ˚C. Zaleca się, aby wszystkie parametry podawać w jednostkach podstawowych, czyli temperaturę w K, ciśnienie w Pa, udział jako udział masowy. Jedynym wyjątkiem jest strumień, który do interfejsu jest przekazywany zawsze po przeliczeniu na mol/s, niezależnie, w jakich jednostkach wpisze go użytkownik. Ponieważ Fluent wymaga strumienia masowego w kg/s, interfejs dokonuje odpowiedniego przeliczenia wykorzystując

[7] Wymagane, patrz rozdział jednostki

współczynnik skalujący #MOLEPERKG w pliku konfiguracyjnym. Całkowicie podstawową jednostką przepływu jest m/s. Fluent dokonuje konwersji z kg/s niejawnie[8]. Osobnym problemem są wielkości przepływu uzyskane w wyniku symulacji. Domyślnie dostępne są polecenia umożliwiające odczytanie przepływu w m/s. Interfejs jednak oczekuje wartości przepływu w jednostkach odpowiadających jednostce wejściowej, czyli kg/s. W tym przypadku należy zdefiniować własną funkcję wyjściową, dokonującą przeliczenia na strumień masowy. Funkcję tą definiuje się we Fluencie w Define/Custom Field Functions i powinna mieć ona postać

$$|V| \cdot density \cdot z - surface - area \tag{1}$$

gdzie poszczególne parametry są funkcjami Fluenta a w szczególności V oznacza *velocity magnitude*

Nazwa nadana własnej funkcji może być używana w skryptach jak każda standardowa funkcja programu[9].

Błędy numeryczne

Podczas procesu obliczeń prowadzonych w komponencie może nastąpić utrata precyzji obliczeń na styku komponent - Fluent. Jest to spowodowane przeliczeniami strumienia z jednostek masowych na prędkości. W przypadku wejścia masowego we Fluencie, wartość z Aspena strumienia wyrażona w mol/s jest przeliczana na kg/s za pomocą przelicznika podanego dyrektywą #MOLEPERKG. Ta wartość jest podawana do Fluenta, gdzie jest niejawnie przeliczana na m/s. Na wyjściu uzyskujemy m/s, które muszą zostać przeliczone powrotem na kg/s za pomocą własnej funkcji[10]. W przypadku słabej dyskretyzacji powierzchni wlotu i wylotu powstają dodatkowe błędy.

W przypadku podania prędkości na wejściu modelu w m/s (Boundary conditions „velocity-inlet"), wartości strumienia masowego mol/s z Aspena także muszą być wielokrotnie przeliczane, ale w tym przypadku użytkownik definiuje samodzielnie pole powierzchni wlotu i wylotu, unikając tym samym błędów dyskretyzacji. Jednostka strumienia podana w m/s jest podstawą jednostką stosowaną we Fluencie.

Celem zmniejszenia błędów numerycznych należy:

1. Przeliczniki w skrypcie podawać z jak największą dokładnością, nawet do 15 miejsc po przecinku
2. Największe błędy powstają przy małych wartościach – gra tutaj rolę ilość cyfr znaczących. Na przykład przepływ 10 mol/s daje 3% błędu, podczas gdy przepływ 1000 mol/s daje 0.03% błędu.
3. Lepiej stosować warunek brzegowy „velocity-inlet" niż „mass-flow-inlet"

Współpraca z ASPENEm

Od strony Aspena istotna jest znajomość nazw związków (komponentów) które są zdefiniowane w programie dlatego, że tymi nazwami należy się posługiwać podczas pisania skryptów konfiguracyjnych. Na przykład H3N - amoniak, H3PO4 - kwas fosforowy itp. Jeśli w dokumencie wystąpi odwołanie do nazwy komponentu to zawsze będzie to nazwa według Aspena.

[8] Patrz rozdział Błędy numeryczne
[9] Patrz przykładowy skrypt
[10] Patrz rozdział Jednostki

Konfiguracja komponentu

Konfiguracja komponentu odbywa się poprzez plik *params.txt*[11]. Plik ten ma charakter skryptu składającego się z prostych komend oraz ich parametrów analizowanych liniowo od początku do końca. Żelazne zasady pisania skryptów są następujące:

- Linie zaczynające się od ## traktowane są jako komentarz
- Linie zaczynające się od # traktowane są jako instrukcja
- Linie zaczynające się od dowolnego innego znaku niż wspomniane powyżej traktowane są jako parametry instrukcji.
- Po linii z instrukcją musi wystąpić linia zawierająca parametr dla tej instrukcji. Nie są dopuszczone puste linie ani komentarz pomiędzy instrukcją a jej parametrem
- Niektóre instrukcje wymagają więcej niż jednego parametru. Parametry umieszczamy jeden pod drugim a całość kończymy słowem kluczowym #END.
- Niektóre parametry mogą składać się z kilku członów. Poszczególne człony muszą wystąpić w określonej kolejności, w jednej linii i muszą być oddzielone jedną spacją.
- Rozróżniane są duże i małe litery
- Pierwszym znakiem w linii musi być #, ## lub dowolna litera lub cyfra dla parametrów. Nie może to być żaden znak biały (spacja, tabulator)
- W ścieżkach dostępu należy używać podwójnych \\ oraz obowiązkowo kończyć ścieżkę także znakiem \\
- Kolejność instrukcji jest dowolna

Instrukcje skryptu

Instrukcje skryptu można podzielić na obowiązkowe i opcjonalne. Obowiązkowe muszą wystąpić w skrypcie, opcjonalne mogą. Brak którejś z instrukcji obowiązkowych spowoduje błąd interpretacji skryptu i zatrzymanie działania komponentu w momencie jego inicjalizacji. Brak instrukcji opcjonalnej spowoduje przyjęcie parametru domyślnego.

Lista poleceń

#FLUENT_PATH

Definiuje bezwzględną ścieżkę dostępu do pliku fluent.exe
Parametry:
- ścieżka dostępu

Nie wymaga #END
Instrukcja obowiązkowa

#SUBPROCNAME

Definiuje subprocesu Fluenta
Parametry:
- nazwa subprocesu - zależna od wersji Fluenta. Dla wersji 6.3.26 powinno być *fl6326s.exe*, dla wersji 12.1.4 *fl1214s.exe*

Nie wymaga #END
Instrukcja obowiązkowa

[11] Ważne informacje w rozdziale Wymagania i filozofia pracy

fl6326s.exe

#DATA_PATH

Definiuje bezwzględną ścieżkę dostępu do katalogu roboczego
Parametry:
- ścieżka dostępu

Nie wymaga #END
Instrukcja obowiązkowa

#COMMAND_LINE

Definiuje parametry przekazywane do Fluenta podczas jego uruchamiania (tylko do celów testowych)
Parametry:
- parametry startowe

Nie wymaga #END
Instrukcja nieobowiązkowa

#CASE_NAME

Definiuje nazwę pliku 'case' zawierającego model. Plik ten musi znajdować sie w katalogu wskazywanym przez **#DATA_PATH**
Parametry:
- nawa pliku 'case' wraz z rozszerzeniem

Nie wymaga #END
Instrukcja obowiązkowa

#**EXPORTS**

Lista wartości exportu, wartości zapisana przy użyciu instrukcji Fluenta. Lista musi mieć postać: *instrukcja_Fluenta interpretacja jaki_komponent*, gdzie *instrukcja_Fluenta* jest słowem kluczowym Fluenta eksportującym dany parametr a *interpretacja* wskazuje, jaki to jest parametr i jak ma być interpretowany przez komponent, zaś *jaki_komponent* określa dla jakiego komponentu dany parametr się stosuje. Możliwe interpretacje to: *temperature*, *totalflow*, *pressure*, *fraction.* Jeśli nazwa komponentu będzie *all*, oznacza to że dany parametr zinterpretowany zgodnie z *interpretacja* zostanie przypisany do wszystkich komponentów chemicznych biorących udział w analizie.
Parametry:
- lista instrukcji Fluenta wraz z ich interpretacją oraz nazwą komponentu

Wymaga #END
Przykład:

```
#EXPORTS
velocity-magnitude totalflow all
total-temperature temperature H3N
#END
```

Instrukcja obowiązkowa

#INPUT_SURFACE

Definiuje nazwę powierzchni wejściowej oraz jej pole powierzchni w m^2. Jeżeli jest więcej wejść to nazwy umieszczamy jedna pod drugą. Powierzchnię w m^2 podajemy zaraz po nazwie i po spacji. Powierzchnia wejścia jest używana jedynie przy warunku brzegowym „*velocity-inlet*”, w przypadku innego warunku wartość powierzchni może być dowolna. Nazwa powierzchni musi się zgadzać z tymi zdefiniowanymi we Fluencie.
Parametry:
- nawa powierzchni wejściowej oraz jej powierzchnia w m^2

Wymaga #END

Przykład:
```
#INPUT_SURFACE
wlot 0.0314159265358979
wlotnh3 0.0314159265358979
#END
```
Instrukcja obowiązkowa

#OUTPUT_SURFACE

Definiuje nazwę powierzchni wyjściowej oraz jej pole powierzchni w m^2. Powierzchnię w m^2 podajemy zaraz po nazwie i po spacji. Powierzchnia wyjścia jest używana jedynie przy warunku brzegowym „*velocity-inlet*", w przypadku innego warunku wartość powierzchni może być dowolna. Nazwa powierzchni musi się zgadzać z tymi zdefiniowanymi we Fluencie.
Parametry:
- nawa powierzchni wejściowej oraz jej powierzchnia w m^2

Wymaga #END
Przykład:
```
#OUTPUT_SURFACE
wylot 0.0314159265358979
#END
```
Instrukcja obowiązkowa

#NUMOFITER

Definiuje ilość iteracji.
Parametry:
- ilość iteracji

Nie wymaga #END
Instrukcja obowiązkowa

#NUMOFTIMESTEPS

Definiuje ilość kroków czasowych.
Parametry:
- ilość kroków czasowych

Nie wymaga #END
Instrukcja nieobowiązkowa. W przypadku nie zdefiniowania ilości kroków czasowych obliczenia zostaną uruchomione poprzez solve/iterate. W przeciwnym wypadku poprzez solve/dual-time-iterate

#BOUND_COND

Definiuje typ warunków brzegowych używanych do nadania parametrów wejściowych.
Parametry:
- nawa warunku brzegowego zgodna z notacją Fluenta. Obecnie obsługiwane typy:
  - *mass-flow-inlet* – wymaga przeliczenia z m/s na kg/s wyjścia we Fluencie przy pomocy dedykowanej funkcji użytkownika
  - *velocity-inlet* – wymaga podania prawidłowego pola powierzchni dla wlotów i wylotu, wymaga podania gęstości dla poszczególnych faz

Nie wymaga #END
Instrukcja obowiązkowa

#ENERGY_EQ

Definiuje czy jest włączone obliczanie przepływu ciepła.
Parametry:
- "yes" albo "no" odpowiednio gdy są włączone równania albo nie.

Nie wymaga #END

Instrukcja obowiązkowa

#COMPONENTS

Definiuje związki chemiczne biorące udział w analizie - ich nazwę, gęstość oraz ilość moli na kilogram substancji. Składnia: *nazwa gęstość,ilosc_moli*. Nazwa musi się zgadać z nazwą w Aspenie. Pierwsza na liście musi być zawsze faza ciągła.
Parametry:

- nazwa zgodna z nomenklaturę Aspena
- gęstość
- ilość moli na jeden kg substancji.

Wymaga #END
Przykład:

```
#COMPONENTS
H3PO4 1362.885 10.204081632653061224489795918367
H3N 0.6894 58.719906048150322959483264826776
#END
```

Instrukcja obowiązkowa

#ASSIGNS

Definiuje powiązania poszczególnych komponentów uzyskiwanych z poszczególnych wejść modelu w Aspenie z odpowiednimi wejściami w modelu Fluenta. Oznacza że z nth wejścia Aspena zostanie pobrany dany materiał o nazwie *nazwa* w takim udziale molowym w jakim występuje i podany zostanie na daną powierzchnie wejściową we Fluencie
Składnia: *nazwa wejscie_aspen wejscie_fluent*
Nawa musi być zgodna z nazewnictwem komponentów w Aspenie, *wejście_fluent* musi także być zgodne z nazewnictwem powierzchni wejściowej we Fluencie (tak jak przy **#INPUT_SURFACE**). Wejście w modelu Aspena określone jest numerycznie i zgodne z numeracją w programie

Parametry:

- nazwa
- numer wejścia
- nazwa powierzchni

Wymaga #END
Przykład:

```
#ASSIGNS
H3N 1 wlotnh3
H3PO4 2 wloth3po4
#END
```

Instrukcja obowiązkowa

Przykładowy skrypt

```
## to jest plik z parametrami konfiguracyjnymi
## to jest linijka komentarza
#FLUENT_PATH
g:\\ANSYS Inc\\v120\\fluent\\ntbin\\ntx86\\fluent.exe
#DATA_PATH
f:\\data\\
#SUBPROCNAME
fl120l6s.exe
#EXPORTS
velocity-magnitude totalflow all
total-temperature temperature all
#END
#CASE_NAME
rura3vel.cas
#NUMOFITER
```

```
110
#BOUND_COND
velocity-inlet
#INPUT_SURFACE
wloth3po4 0.0314159265358979
wlotnh3 0.0314159265358979
#END
#OUTPUT_SURFACE
wylot 0.0314159265358979
#END
#ENERGY_EQ
no
#COMPONENTS
H3PO4 1362.885 10.204081632653061224489795918367
H3N 0.6894 58.719906048150322959483264826776
#END
#ASSIGNS
H3N 1 wlotnh3
H3PO4 2 wloth3po4
#END
```

W przykładzie tym ustawiono ścieżki dostępu do pliku Fluent oraz do katalogu z modelem, podano nazwę modelu *rura3vel.cas*. Eksportowane są wielkości *velocity-magnitude*, wynik jest interpretowany, jako przepływ oraz *total-temperature,* co jest interpretowane jako temperatura. Obie wielkości stosują się do wszystkich związków chemicznych biorących udział w analizie. Ustalono ilość iteracji na 110, warunki brzegowe na *velocity-inlet*, podano nazwy powierzchni wejściowych i wyjściowych odpowiadającym tym z modelu Fluenta. Zaznaczono, że symulacja przebiega z wyłączonymi równaniami transferu ciepła. W analizie biorą udział dwa związki: H3PO4 oraz H3N, nazwy są zgodne z nomenklaturą Aspena. Na końcu połączono pierwsze wejście z Aspena z powierzchnią *wlotnh3* oraz drugie wejście z powierzchnią *wloth3po4*. Oznacza to, że do pierwszego wejścia kontrolki Cape2Fluent podano strumień amoniaku a do drugiego wejścia strumień kwasu. Parametry fizyczne tych strumieni zostaną przepisane na odpowiednie wejścia w modelu Fluenta.