Instrukcja 6 i 7
Model turbulencji $k-\epsilon$

Wstęp

Celem laboratorium jest samodzielna implementacja oraz sprawdzenie działania modelu turbulencji $k-\epsilon$ na przykładzie uderzenia strugi płynu w płaską powierzchnię przy liczbie Reynoldsa $Re = 23000$. Przy pomocy UDF-ów zaimplementujemy klasyczny model $k-\epsilon$. Następnie zweryfikujemy jego działanie w prostej konfiguracji (dwuwymiarowy kanał). Po udanej weryfikacji przeliczymy już właściwy przypadek z uderzającą strugą. Wyniki naszych symulacji porównamy z wynikami eksperymentalnymi ("Impinging jet studies for turbulence model assessment", Cooper *et. al*, Int. J. Heat Mass Transfer, 1992)[1].

1 Uśrednone równania pędu oraz model turbulencji $k-\epsilon$

Przepływy turbulentne są ze swojej natury przepływami nieustalonymi i dokładna ich symulacja wymaga ogromnych mocy obliczeniowych. Z tego względu najpowszechniejszym podejściem inżynierskim jest próba rozwiązania uśrednienionych w czasie równań Naviera-Stokes'a (tzw. równań RANS - Reynolds-Averaged Navier-Stokes). Ponieważ równania N-S są nieliniowe, każdy proces uśredniania generuje dodatkowe niewiadome, które trzeba w jakiś sposób powiązać w wielkościami średnimi (mówimy o tzw. problemie domknięcia) - właśnie na tym etapie pojawia się tzw. modelowanie turbulencji. Jedną z propozycji takiego modelu z lat 70. XX wieku jest wprowadzenie nowych zmiennych, k - tzw. kinetycznej energii turbulencji oraz ϵ - dyssypacji energii. Obie zmienne składają się na tzw. lepkość turbulentną μ_t, która to wielkość ma za zadanie modelować pozorne zwiększenie lepkości związane z faktem istnienia dodatkowych fluktuacji. Równania transportu wyglądają wówczas następująco:

$$\frac{\partial u_i}{\partial t} + u_k \frac{\partial u_i}{\partial x_k} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} - \frac{1}{\rho}\frac{\partial}{\partial x_k}\Big((\mu+\mu_t)(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i})\Big)$$

$$\frac{\partial u_k}{\partial x_k} = 0$$

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k u_j)}{\partial x_j} = \frac{\partial}{\partial x_j}\Big((\mu + \frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_j}\Big) + G_k - \rho\epsilon$$

$$\frac{\partial(\rho \epsilon)}{\partial t} + \frac{\partial(\rho \epsilon u_j)}{\partial x_j} = \frac{\partial}{\partial x_j}\Big((\mu + \frac{\mu_t}{\sigma_\epsilon})\frac{\partial \epsilon}{\partial x_j}\Big) + C_{1\epsilon}G_k\frac{\epsilon}{k} - C_{2\epsilon}\rho\frac{\epsilon^2}{k}$$

Równania na k oraz ϵ zawierają dodatkowe wielkości:

$$\mu_t = \rho C_\mu \frac{k^2}{\epsilon} \qquad G_k = \mu_t S^2 \qquad S = \sqrt{2S_{ij}S_{ij}} \qquad S_{ij} = \frac{1}{2}\Big(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i}\Big)$$

[1] dane eksperymentalne zostały ściągnięte z bazy ERCOFTAC-u *http://cfd.mace.manchester.ac.uk/ercoftac/* - można tam znaleźć również inne eksperymenty służące jako *benchmarki* dla innych symulacji

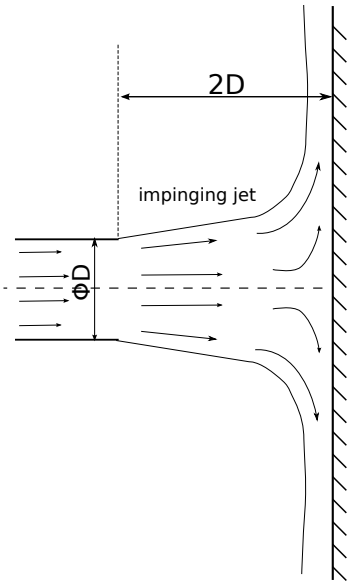
Ponadto model wymaga szeregu stałych:

$$C_\mu = 0.09 \qquad C_{1\epsilon} = 1.44 \qquad C_{2\epsilon} = 1.92 \qquad \sigma_k = 1.0 \qquad \sigma_\epsilon = 1.3$$

Zauważmy, że równania są rozwiązywane dla wielkości uśrednionych po odpowiednim okresie czasu, niemniej zakłada się, że nadal te uśrednione wielkości mogę mieć pewną zmienność w czasie.

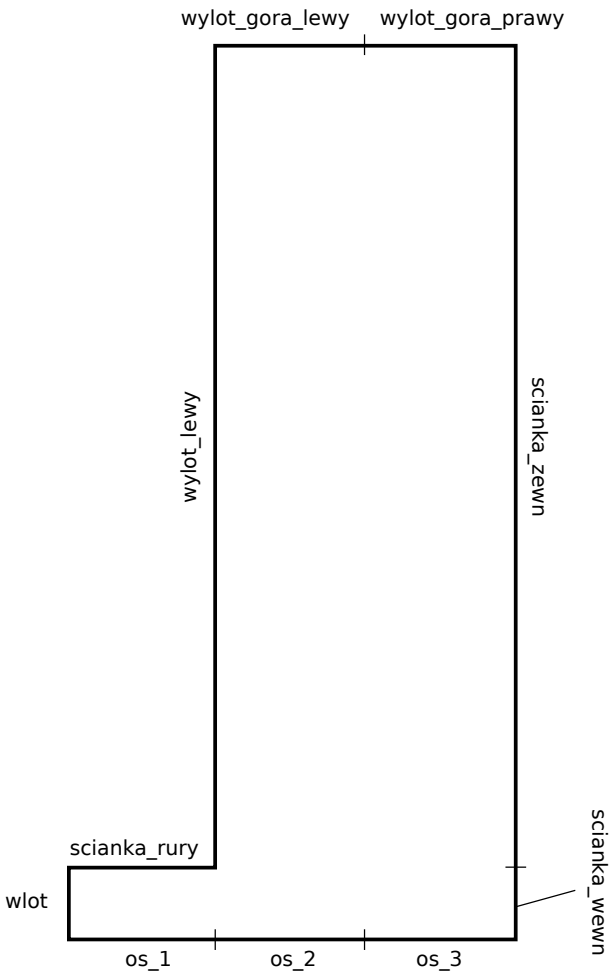
2 Geometria i warunki brzegowe

Będziemy symulować uderzenie strugi powietrza (*ang. "jet"*) w prostopadłą płytkę. Rozwinięty przepływ turbulentny wylatuje z rurki do swobodnej przestrzeni (wypełnionej tą samą substancją) i uderza o ściankę. Przepływ jest osiowosymetryczny. Geometrię ilustruje Rys. 1:



Rysunek 1: Struga uderzająca o ścianę.

W symulacjach będziemy używać siatki strukturalnej dostępnej w pliku *jet_mesh.msh*. Ze względu na osiową symetrię problemu, siatka obliczeniowa stanowi połowę obszaru zaznaczonego na Rys. 1. Zawiera ona dolot z rury długosci 1 średnicy oraz jest wysoka na 6 średnic. Średnica rury, z której wylatuje struga to $\phi = 20mm$. Siatka została wygenerowana tak, by szacunkowe wartosci y^+ na ściankach materialnych były bliskie 1. Lepkość dynamiczna powietrza wynosi $\mu = 1.7894 \cdot 10^{-5} Pa \cdot s$. Prędkość została dobrana tak, aby uzyskać pożądaną wartość liczby Reynoldsa. Dokładny profil wlotowy zostanie wczytany z pliku. Wszystkie brzegi siatki zostały przedstawione na Rys. 2. Odpowiednie warunki brzegowe dla każdego z obszarów zostały opisane w Tabeli 1. Szczegółowe wyjaśnienia implementacji UDF dla warunków brzegowych będą omówione w dalszej części instrukcji.



Rysunek 2: Nazwy krawędzi siatki obliczeniowej.

3 Implementacja modelu turbulencji

Zazwyczaj posługujemy się modelami turbulencji, które już zostały zaimplementowane w solwerze FLUENT i aktywujemy je za pośrednictwem opcji *viscous* → *turbulent*. Tym razem jednak zaimplementujemy nasz model własnoręcznie. Złoży się na to kilka elementów:

- dodanie do solwera k oraz ϵ jako UDS (user-defined scalars)
- implementacja równań na k i ϵ sprowadzonych do postaci równań konwekcji-dyfuzji
- sprzężenie równań na k i ϵ z równaniami pędu za pośrednictwem członu lepkościowego - lepkość będzie dostarczana w postaci UDF-a, który uzupełni lepokść laminarną μ o lepkość turbulentną μ_t policzoną ze zmiennych k i ϵ

Tablica 1: Opis warunków brzegowych

Nr	Nazwa	Warunek brzegowy	Dodatkowe uwagi
1	wlot	velocity_inlet	wczytać profil wlotowy z pliku *przeplyw_w_rurze.prof*, ustawić wlotowe $k = 0.5$ oraz $\epsilon = 517$
2	os_1	axis	-
3	os_2	axis	-
4	os_3	axis	-
5	scianka_wewn	wall	
6	scianka_zewn	wall	$k = 0$, ϵ wyznaczone za pomocą UDF
7	scianka_rury	wall	
8	wylot_gora_prawy	pressure_outlet	zerowy strumień dla k i ϵ, po kilkuset iteracjach patrz pkt 4.1: "poprawa zbieżności obliczeń"
9	wylot_gora_lewy	pressure_outlet	
10	wylot_lewy	pressure_outlet	

- implementacja UDF-ów zawierajacych dodatkowe definicje: warunek brzegowy dla ϵ, lepkość turbulentną itp.

3.1 Struktura równań konwekcji-dyfuzji we Fluencie

Aby zaimplementować dodatkowe równania dla naszego modelu turbulencji, najpierw musimy przeanalizować, w jaki sposób należy we Fluencie implementować równania konwekcji-dyfuzji. Takie równanie dla wielkości skalarnej ϕ_k przybiera następującą postać:

$$\frac{\partial(\rho\phi_k)}{\partial t} + \frac{\partial}{\partial x_i}\Big(\rho u_i\phi_k - \Gamma_k\frac{\partial\phi_k}{\partial x_i}\Big) = S_{\phi_k}, \quad k = 1, ..., N$$

Gdzie Γ_k oznacza dyfuzyjność natomiast S_{ϕ_k} człon źródłowy. Oba współczynniki mogą przybrać konkretne wartości zadane we Fluencie, ale też mogą zostać zdefiniowane za pomocą UDF-ów. Właśnie tą drugą drogą otrzymamy sprzężenie ze zmiennymi przepływu. Należy jeszcze tylko równania na k jak i ϵ sprowadzić do postaci równań konwekcji-dyfuzji:

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial}{\partial x_j}\Big(\rho u_j k - (\mu + \frac{\mu_t}{\sigma_k})\frac{\partial k}{\partial x_j}\Big) = G_k - \rho\epsilon$$

$$\frac{\partial(\rho\epsilon)}{\partial t} + \frac{\partial}{\partial x_j}\Big(\rho u_j \epsilon - (\mu + \frac{\mu_t}{\sigma_\epsilon})\frac{\partial \epsilon}{\partial x_j}\Big) = C_{1\epsilon}G_k\frac{\epsilon}{k} + C_{2\epsilon}\rho\frac{\epsilon^2}{k}$$

3.2 Implementacja modelu $k-\epsilon$ w postaci UDF

Implementacja nowego modelu turbulencji będzie wymagała stworzenia następujących UDF-ów:

1. DEFINE_SOURCE(k_production, c, t, dS, eqn) - człon źródłowy dla k, zależny od ϵ, pochodnych prędkości oraz μ_t. Do policzenia pochodnych posłużymy się odpowiednimi makrami.
2. DEFINE_SOURCE(epsilon_production, c, t, dS, eqn) - człon źródłowy dla ϵ.

3. DEFINE_PROPERTY(k_diffusivity, c, t) - dyfuzyjność dla k - konieczne będzie użycie makra *C_UDMI(c,t,0)* aby wydobyć wartość μ_t z user-defined memory.

4. DEFINE_PROPERTY(epsilon_diffusivity,c,t) - dyfuzyjność dla ϵ - podobnie jak wyżej, będziemy potrzebować μ_t.

5. DEFINE_PROPERTY(turbulent_viscosity,c,t) - sposób wyliczania lepkości całkowitej $\mu + \mu_t$, w tej funkcji wpiszemy do user-defined memory wartość μ_t oraz zwrócimy lepkość całkowitą. Sama lepkość dynamiczna ma wartość $\mu = 1.7894 \cdot 10^{-5} Pa \cdot s$.

6. DEFINE_PROFILE(eps_bc,t,i) - warunek brzegowy dla ϵ na ściance materialnej. Jest on zadany następującym wzorem:

$$\epsilon = 2\frac{\mu}{\rho}\frac{k}{y_c^2}$$

   W tym przypadku y_c jest odległością środka komórki obliczeniowej od brzegu materialnego[2]. Ten UDF będzie musiał zatem korzystać z makra *THREAD_T0(t)*, aby móc uzyskać dane z komórek, gdy będziemy iterować się po brzegu.

Przykładowe UDF-y są dokładnie opisane na końcu instrukcji.

3.3 Weryfikacja zaimplementowanego modelu

Aby mieć możliwość sprawdzenia działania naszego modelu turbulencji, wykonamy symulację możliwie najprostszego przypadku: przepływu w dwuwymiarowym kanale. Wlot zadamy jako profil paraboliczny a następnie będziemy obserwować, jak rozwija się profil turbulentny. Jeśli nasza implementacja będzie poprawna, otrzymane wyniki będą zgodne z wynikami referencyjnymi z Rys. 3. Wszystkie zabiegi zostaną opisane krok po kroku. Do wykonania zadania będzie potrzebny plik z siatką *kanal_2d.msh*.

1. Do Fluenta wczytaj siatkę *kanal_2d.msh*: *File* → *Read* → *Mesh*.

2. Sprawdź siatkę: *Mesh* → *Check*. Gdyby sprawdzenie zawiodło, wpisz do konsoli zasugerowane polecenie do naprawy siatki.

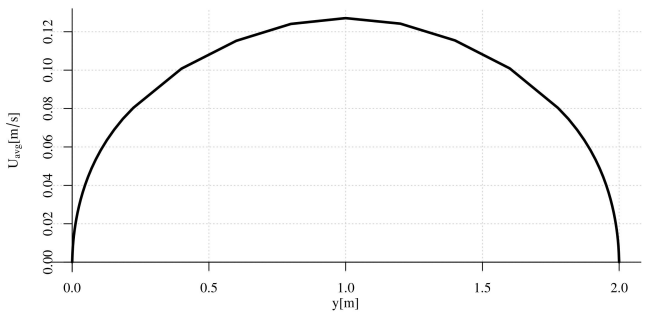
3. W katalogu roboczym otwórz plik *main.c*, który zawiara wszystkie niezbędne UDF-y. Uzupełnij go o UDF *DEFINE_PROFILE(predkosc_wlot, t, i)*, który zada paraboliczny profil prędkości wlotowej wg wzoru $u(y) = 0.15y(2 - y)$.

4. Plik *main.c* musi zostać zinterpretowany przez Fluenta: *Define* → *User-Defined* → *Functions* → *Interpreted* → *main.c* → *OK* → *Interpret* → *Close*.

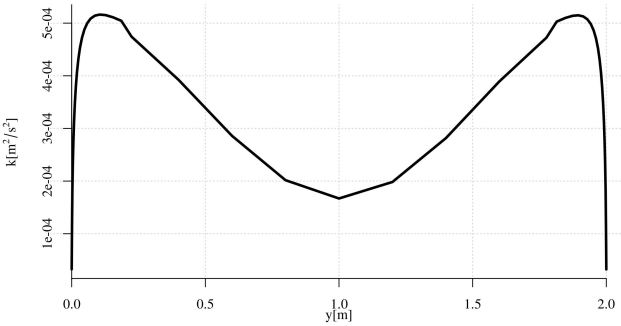
5. Wejdź do menu *Define* → *User-Defined* → *Scalars*. Zmień liczbę dodatkowych skalarów na 2. Wyłącz opcję *Inlet Diffusion*. Sprawdź, czy dla obu indeksów *Flux Function* jest wybrana jako *mass flow rate*. Zatwierdź *OK*.

6. Wejdź do menu *Define* → *User-Defined* → *Memory*. Ustaw liczbę dodatkowych zmiennych dla każdej komórki na 1. Zatwierdź wybór.

[2] nietrudno zauważyć, że taki warunek brzegowy zawiera pewien podstawowy błąd metodologiczny: mianowicie wartość brzegowa ϵ staje się zależna od wartości k w środku komórki przylegającej do tego brzegu. Ale w praktyce daje i tak w miarę sensowne wyniki.



(a)



(b)

Rysunek 3: Profile prędkości średniej (a) i energii kinetycznej turbulencji (b) dla przepływu w dwuwymiarowym kanale.

7. Przejdź do zakładki *Models*. Upewnij się, że masz wybrany model *Laminar*. Model turbulencji zostanie wprzęgnięty do solwera poprzez modyfikację lepkości laminarnej.

8. Przejdź do zakładki *Materials*. Wybierz *air* → *Create/Edit* i w polu *Viscosity* wybierz *user-defined* → *turbulent_viscosity* → *OK*. W tym samym oknie musimy wybrać dyfuzyjności dla energii kinetycznej turbulencji oraz dla dyssypacji energii. W polu *UDS Diffusivity* wybierz *defined-per-uds* i wyedytuj je tak, aby dla *uds-0* przypisać *k_diffusivity*, zaś dla *uds-1* przypisać *epsilon_diffusivity*. Na koniec zaakceptuj przez kliknięcie *Change/Create*.

9. Następnie przejdź do *Cell Zone Conditions* → *Edit*, zaznacz opcję *Source Terms* i przejdź do zakładki o tej samej nazwie. Dla *User Scalar 0* wybierz *udf k_production*, a dla *User Scalar 1* wybierz *udf epsilon_production*. Zakończ *OK*.

10. Przejdź do zakładki *Boundary Conditions*. Upewnij się (lub ewentualnie zmień), że pola *dol*

oraz *gora* mają warunek brzegowy typu *wall*, zaś wylot warunek *pressure-outlet*. Dla powierzchni wlotowej wybierz warunek typu *velocity-inlet*.

11. Zajmij się edycją warunku *velocity_inlet*:
    - W zakładce *Momentum* upewnij się, że *Velocity Specification Method* jest ustawione na *Magnitude, Normal to Boundary*, ponadto *Velocity Magnitude* ustaw na *udf_predkosc_wlot*.
    - W zakładce *UDS* musimy ustawić stałe wartości $k = 3 \cdot 10^{-4}$ oraz $\epsilon = 3 \cdot 10^{-5}$: *User Scalar 0* $\rightarrow$ 3e-4 oraz *User Scalar 1* $\rightarrow$ 3e-5. Oba oczywiście na *constant*.
12. Zajmij się edycją warunku *pressure_outlet*:
    - W zakładce *Momentum* upewnij się, że *Gauge Pressure* jest ustawione na wartość 0.
    - W zakładce *UDS* warunki wylotowe dla obu UDS-ów ustaw na *Specified Flux* i wartość 0 - oznacza to jednorodny warunek brzegowy type Neumana - pochodna w kierunku normalnym do wylotu znika, k oraz ϵ mają ustaloną wartość.
13. Zajmij się edycją warunków *wall* - przejdź do zakładek *UDS*:
    - *UDS 0* ustaw na *Specified Value* ze stałą wartością 0
    - *UDS 1* ustaw na *Specified Value* ze wartością wyliczaną za pomocą UDF-a *eps_bc*
14. W zakładce *Solution Controls* zmień współczynniki podrelaksacji dla *UDS*-ów na 0.3.
15. W zakładce *Monitors* zmień wymagania w stosunku do wszystkich residuali na $1e-5$.
16. Przejdź do zakładki *Solution Initialization* i zmień sposób inicjalizacji na *Standard Initialization*. Ustaw wartość ciśnienia na 0, wartość prędkośći na 0.15 m/s, energii kinetycznej turbulencji na $3 \cdot 10^{-4}$, a dyssypacji na $3 \cdot 10^{-5}$. Naciśnij *Initialize*.
17. Przejdź do zakładki *Run Calculation*, zmień liczbę iteracji na 3000, kliknij *Calculate* i podziwiaj, jak obliczenia zbiegają.
18. Porównaj swoje wyniki z wynikami referencyjnymi z Rys. 3. Aby tego dokonać, należy utworzyć wykresy obu wielkości: *Plots* $\rightarrow$ *XY Plot* $\rightarrow$ *Set Up...*. Następnie *Y Axis Function* ustawić na *Velocity...* i *Velocity Magnitude*, *X Axis Function* ustawić na *Curve Length*, kliknąć *Plot*. Aby obejrzeć k, wybór *Y Axis Function* zmienić na *User Defined Scalars...* i *Scalar-0* i ponownie *Plot*.
19. Jeśli zajdzie taka konieczność, zwiększ liczbę iteracji.

4 Obliczenia dla strugi

Po upewnieniu się, że nasz model turbulencji działa poprawnie, możemy zająć się naszą główną symulacją. Do wykonania ćwiczenia będą potrzebne następujące pliki:

- *jet_mesh.msh* - plik z siatką,
- *przeplyw_w_rurze.prof* - profile do rury dla u_x, k, ϵ

- *profil_k_0*, *profil_k_2*, *profil_k_4*, *profil_k_6* - wartości k w przekrojach $r/D = 0, 1, 2, 3$
- *profil_U_0-profil_U_6* - wartości U_{avg} w wybranych przekrojach $r/D = 0, 0.5, 1, 1.5, 2, 2.5, 3$

Zasadniczo, przebieg ustawień wygląda tak jak w poprzednim podpunkcie, trzeba jednak uwzględnić odpowiednie różnice:

1. W zakładce *General* ustawić analizę na *axi-symmetric*.
2. Wczytać siatkę *jet_mesh.msh*.
3. Wczytać profil wlotowy: Wejdź do menu *Define* → *Profiles* → *Read* → *przeplyw_w_rurze.prof* → *Close*.
4. Nanieść nowe warunki brzegowe (patrz: Tabela 1). W szczególności:
   - Przejdź do zakładki *Boundary Conditions*. Upewnij się (lub ewentualnie zmień), że trzy fragmenty osi mają warunek brzegowy typu *axis*, zaś wszystkie wyloty warunek *pressure-outlet*. Dla powierzchni wlotowej wybierz warunek typu *velocity-inlet*.
   - Kliknij *Edit* dla warunku wlotowego. Przełącz *Velocity Specification Method* na *Components* i wybierz dla *Axial Velocity* z menu profil *wylot axial-velocity* i podobnie uczyń dla prędkości promieniowej. Następnie przełącz się do zakładki *UDS*, przełącz dla obu wielkości typ warunku na warunek Dirichleta *Specified Value* i ich wartości poniżej odpowiednio *wylot turb-kinetic-energy* oraz *wylot turb-diss-rate*. Kliknij *OK*.
   - Pozostałe warunki brzegowe nanieś tak jak poprzednio: pamiętaj o warunkach brzegowych dla ϵ na ściankach materialnych oraz o warunkach wylotowych dla k oraz ϵ
5. Po uwzględnieniu warunków brzegowych oraz dodatkowych ustawień solwera (rząd metody, wartość residuów), przejdź do zakładki *Solution Initialization* i zmień sposób inicjalizacji na *Standard Initialization*. Ustaw wartość ciśnienia na 0, wartość osiowej składowej prędkośći na 16.79 m/s, składowej promieniowej na 0 m/s, energii kinetycznej turbulencji na 1.05, a dyssypacji na 500. Naciśnij *Initialize*.
6. Przejdź do zakładki *Run Calculation*, zmień liczbę iteracji na 100, kliknij *Calculate*, poczekaj na wyniki i przejdź do następnego podpunktu.

4.1 Poprawa zbieżności obliczeń w trakcie działania solwera

Standardowe warunki wylotowe powodują, że trudno jest uzyskać dobrą zbieżność obliczeń. Dla przypomnienia: na wszystkich wylotach z obszaru zadaliśmy warunek typu *pressure_outlet* oraz zerowe strumienie k oraz ϵ. Po wykonianiu 100 iteracji zmodyfikujemy warunki dla obu wielkości: należy sczytać średnią prędkość U_{avg} na każdym z wylotów, a następnie zadać konkretne wartości dla obu zmiennych:

$$k = \frac{3}{2}(U_{avg} t_i)^2 \qquad \epsilon = \frac{k^{1.5}}{l_0} \qquad l_0 = 0.1D \qquad t_i = 0.01$$

Do wyznaczenia wartości k oraz ϵ posłużyliśmy się dodatkowymi zmiennymi: t_i-intensywnością turbulencji oraz l_0 - długością charakterystyczną.
Wszystko wykonujemy w sposób nastepujący:

7. Przejdź do zakładki *Report* → *Surface Integrals*. Kliknij *Set Up...*. Następnie wybierz *Area Weighted Average*. Jako zmienną ustaw *Velocity* → *Axial Velocity*, trzymając klawisz *Ctrl*, zaznacz obszary, z których policzymy wydatek: *wylot_lewy, wylot_gora_lewy, wylot_gora_prawy* i kliknij *Compute*. Znając strumień masy na każdym kawałku brzegu, możemy policzyć nowe wartości k i ϵ dla każdego brzegu i dokonać zmian w *Boundary Conditions* → *brzeg...* → *UDS* → *User Scalar* → *Specified Value* dla każdego brzegu.

8. Po uwzględnieniu poprawek, wykonaj jakieś 2000 iteracji.

4.2 Analiza jakości wyników uzyskanych w symulacjach

Po uzyskaniu zbieżności powinniśmy porównać nasze wyniki z wynikami pochodzacymi ze wspomnianego na wstępie eksperymentu. Będziemy tworzyć wykresy porównawcze dla serii przekrojów. Procedura wygląda następująco:

10. Najpierw musimy utworzyć odpowiednie powierzchnie, które pozwolą nam sczytać wartości dla różnych promieni. Robimy to w zakładce: *Surface* → *Iso Surface*. Jako izo-wartość wybieramy *Mesh* → *Y coordinate*. Należy utworzyć 6 równoodległych powierzchni dla $y = 0.01m$ do $y = 0.06m$ i adekwatnie je ponazywać (najlepiej od *cut_1* do *cut_6* aby były zgodne z indeksami plików z wynikami eksperymentalnymi).

11. Teraz możemy porównywać wyniki z obu metod: *Reports* → *XY Plot*. Należy wczytać pliki z wynikami eksperymentalnymi: *Load_File* i, trzymając *Ctrl*, wybrać wszystkie pliki *profil_U...* oraz *profil_k...*. Teraz możemy swobodnie kreślić wszystkie porównania na dowolnie wybranych przekrojach. Należy jeszcze pamiętać o poprawnym wyborze wektora *Plot Direction*.

Krótki przewodnik po UDF-ach

Przykładowe UDF-y

Pokazujemy przykładowe dwa UDF-y, które służą odpowiednio do wyliczania naprężeń stycznych na poziomych ściankach dla płynu ze zmienną lepkością oraz do wyliczenia samej lepkości. Tutaj dla przykładu proponujemy płyn, którego lepkość zależy od stężenia C pewnej substancji, zdefiniowanej jako *UDS_0*. Obie wielkości wyrażają się następującymi wzorami:

$$\tau = \mu(C)\frac{\partial u_x}{\partial n}\Big|_{brzeg} \qquad \mu(C) = \mu_0 + (\mu_\infty - \mu_0)(1 - e^{-kC})$$

k,μ_0 oraz μ_{infty} to stałe materiałowe. Kanał jest ustawiony poziomo, górna ścianka znajduje się na wysokości $y_{top} = 0.05$, dolna na 0. Algorytm będzie wyliczał naprężenie, licząc w przybliżony sposób pochodną w kierunku normalnym do brzegu za pomocą różnic skończonych. Ponieważ na ściance obowiązuje warunek braku poślizgu, pochodna w kierunku normalnym to po prostu wartość predkości w środku komórki przyległej do brzegu podzielona przez odległość środka komórki od tego brzegu (oznaczoną jako h):

$$\tau \approx \mu(S)\frac{u_x|_{brzeg}}{h}$$

Podstawowe operacje oraz typy danych w UDF-ach

Zanim przedstawimy UDF-y, warto jeszcze omówić podstawowe typy danych używane w UDF-ach:

`real` - liczba rzeczywista

`face_t` - brzeg komórek objętości skończonych

`cell_t` - komórka objętości skończonych

`thread` - tzw. wątek - określa zbiór komórek albo brzegów (face-ów) należących do pewnej kategorii (np. do określonego obszaru widzianego z zewnątrz jako jeden odcinek i opatrzonego jednym warunkiem brzegowym)

Bardzo często stosuje się pętle po całych wątkach - używa się w tym celu specjalnej konstrukcji pętli: `begin_f_loop {...} end_f_loop` dla face-ów albo `begin_c_loop {...} end_c_loop` dla iteracji po komórkach.

Przykład 1

Pierwszy UDF implementuje sposób wyliczania lepkości. Konieczne jest m.in. wydobycie stężenia z UDS-a.

```
DEFINE_PROPERTY(viscosity_cdep,c,t)
{
      real mu_0 = 0.001, mu_inf = 10, k=150;      // stałe materiałowe
      real mu_c, C;

     // wydobywamy wartość stężenia z UDS-a o indeksie 0
     C = C_UDSI(c,t,0)

     mu_c = mu_0 +(mu_inf - mu_0)*(1-exp(-k*C));

     // wpisanie mu_c do user-defined-memory o indeksie 0 - tą drogą będzie
     // ona dostępna dla innych UDF-ów bez konieczności powtórnego wyliczania
     C_UDMI(c,t,0) = mu_c;

     // zwrócenie lepkości do programu głównego
     return mu_c;
}
```

Przykład 2

UDF wyliczający naprężenia na ściankach:

```
real DEFINE_PROFILE(tau_wall,t,i)
{
      real x[ND_ND] ;     // wektor 2-wymiarowy, do którego trafią
                           //  współrzędne środka komórki
```

```
 real h, mu_c,ux, tau;
 face_t f;     // face - potrzebny do pętli
 cell_t c0;     // komórka przyległa do brzegu


begin_f_loop(f,t)
{

c0 = F_C0(f,t);     // znajdujemy komórkę przylegającą do face'a f
C_CENTROID(x,c0,THREAD_T0(t));
// do c przypisujemy współrzędne środka komórki
// x[0] zawiera współrzędną x, x[1] zawiera współrzędną y

// wyliczamy h zależnie od naszego położenia w kanale
if( x[1] >= 0.025) h = 0.05 - x[1];
else h = x[1];

// wydobywamy wartość lepkości z UDM-a o indeksie 0
mu_c = C_UDMI(c0,THREAD_T0(t),0);

ux = C_U(c0,THREAD_T0(t));  // wydobywamy prędkość

tau = mu_c * ux/h;

// wyliczone tau przypisujemy do profilu
F_PROFILE(f,t,i) = tau;
}
end_f_loop(f,t)

}
```

Przytoczony przykład moze nie należy do najwymyślniejszych (zwłaszcza, że istnieją zaimplementowane funkcje, które są w stanie wyliczyć wartości naprężeń albo zwrócić odpowiednie pochodne), ale ilustruje podstawowe sposoby używania makr w UDF-ach.

Makra stosowane w UDF-ach

Ja widać w powyższych przykładach, we Fluencie funkcje mają w istocie formę makr, dlatego ich składnia na pierwszy rzut oka może się wydawać zawoalowana. W trakcie pisania makr często musimy wywoływać inne makra, które pozwolą nam na wydobycie wartości pewnych zmiennych. Opis makr niezbędnych do napisania UDF-ów do naszego ćwiczenia zamieszczony jest w Tabeli 2.

Tablica 2: Spis makr użytecznych przy pisaniu UDF-ów. Niektóre makra stanowią w istocie nagłówki funkcji, inne natomiast są wywoływne tak jak funkcje typu *void*, jeszcze pozostałe zachowują się jak zwykłe zmienne - można z nich wyekstrahować wartości jak i je nadpisać.

Nr	Makro	Opis
1	void DEFINE_PROFILE(*nazwa_makra,t,i*)	Makro służące do definiowania profili wartości dla warunków brzegowych. We FLUENCIE to makro będzie widoczne pod nazwą *nazwa_makra*. Argumenty t oraz i to robocze zmienne FLUENTa: wątek oraz numer równania
2	real F_PROFILE(*f,t,i*)	Do tego makra przypisuje się wartość funkcji, która ma być przyjęta na brzegu. f to aktualny *face*, do którego przypisywana jest wartość, *t,i*-zmienne robocze
3	void F_CENTROID(*x,f,t*)	Pod zmienną x przypisywane są współrzędne środka *face'a f*. Argument x musi być wektorem o rozmiarze równym wymiarowości problemu (tutaj: tablicą dwuelementową).
4	void C_CENTROID(*x,c,THREAD_T0(t)*)	Pod zmienną x wpisywane są współrzędne środka komórki. Indeks komórki przechowywany jest pod zmienną c. Dodatkowy argument w postaci makra *THREAD_T0(t)* przekształca wątek powierzchniowy na komórkowy (komórki leżą zazwyczaj na innych wątkach niż *face'y*).
5	cell_t F_C0(*f,t*)	Makro zwraca komórkę, ktorej brzegiem jest face f. Jeśli face nie leży na brzegu, wówczas można użyć jeszcze makra F_C1, aby dotrzeć do drugiej komórki.
6	real C_UDSI(*c0, THREAD_T0(t),s_id*)	Makro zwracające wartość user-defined scalar'a o indeksie *s_id* z komórki *c0*.
7	real C_UMDI(*c,t,m_id*)	Makro przypisujące prawą stronę do user-defined memory o indekse *m_id* w komórce c na wątku t.
8	DEFINE_SOURCE(*nazwa_zrodla, c,t, dS,eqn*)	Definiuje człon źródłowy do różnych równań transportu (w naszym przypadku do user-defined scalarów). *c,t*-komórka i wątek, *dS*- pochodne czlonu produkcji po zmiennych przeplywowych (u nas ignorowane, wpływają na tempo zbieżności), *eqn* -numer równania (dostarczany przez solwer).
9	real C_U_G(c,t)[0]	Pochodna $\frac{\partial u_x}{\partial x}$. *c,t* - komórka i wątek.
10	real C_U_G(c,t)[1]	Pochodna $\frac{\partial u_x}{\partial y}$. *c,t* - komórka i wątek.
11	real C_V_G(c,t)[0]	Pochodna $\frac{\partial u_y}{\partial x}$. *c,t* - komórka i wątek.
12	real C_V_G(c,t)[1]	Pochodna $\frac{\partial u_y}{\partial y}$. *c,t* - komórka i wątek.
13	DEFINE_PROPERTY(*nazwa_zmiennej, c, t*)	Makro wyliczające wartość zmiennej *nazwa_zmiennej* w komórce c na wątku t.