Wydział Nauk Technicznych

UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN
The Faculty of Technical Sciences
POLAND, 10-957 Olsztyn, M. Oczapowskiego 11
tel.: (48)(89) 5-23-32-40, fax: (48)(89) 5-23-32-55
URL: http://www.uwm.edu.pl/edu/sobieski/ (in Polish)

NUMERYCZNE METODY MECHANIKI

OpenFOAM – część 2

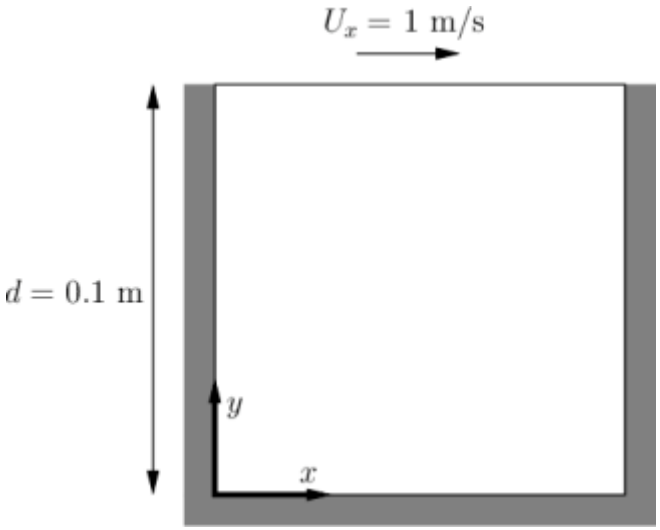
Publikacja została napisana w wyniku odbywania przez autora stażu w Holandii, współfinansowanego przez Unię Europejską w ramach Europejskiego Funduszu Społecznego (Program Operacyjnego Wiedza Edukacja Rozwój), zrealizowanego w projekcie Program Rozwojowy Uniwersytetu Warmińsko-Mazurskiego w Olsztynie (POWR.03.05.00-00-Z310/17).

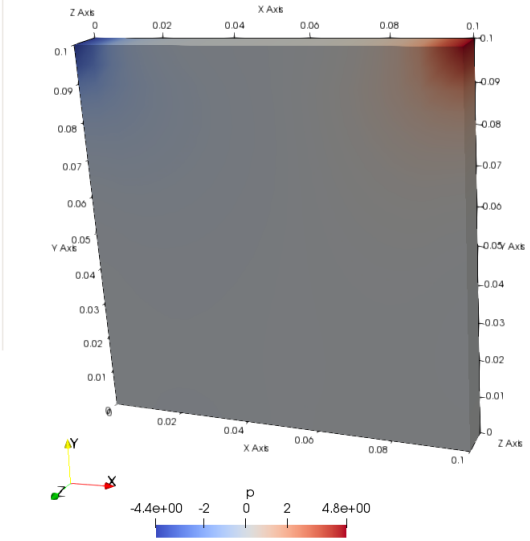
Wojciech Sobieski

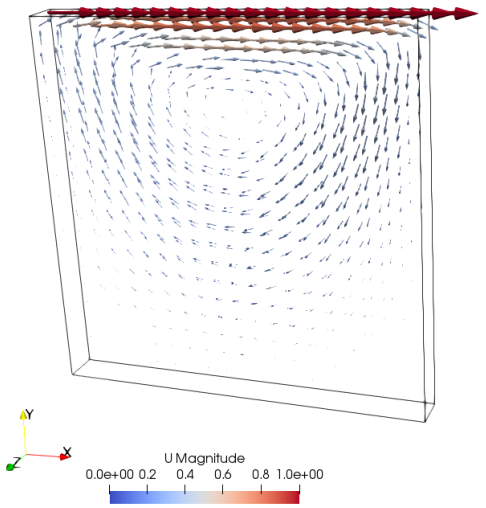
Utrecht, 2019

Cavity

Cavity – najprostszy możliwy przykład ilustrujący sposób pracy w środowisku OpenFOAM.

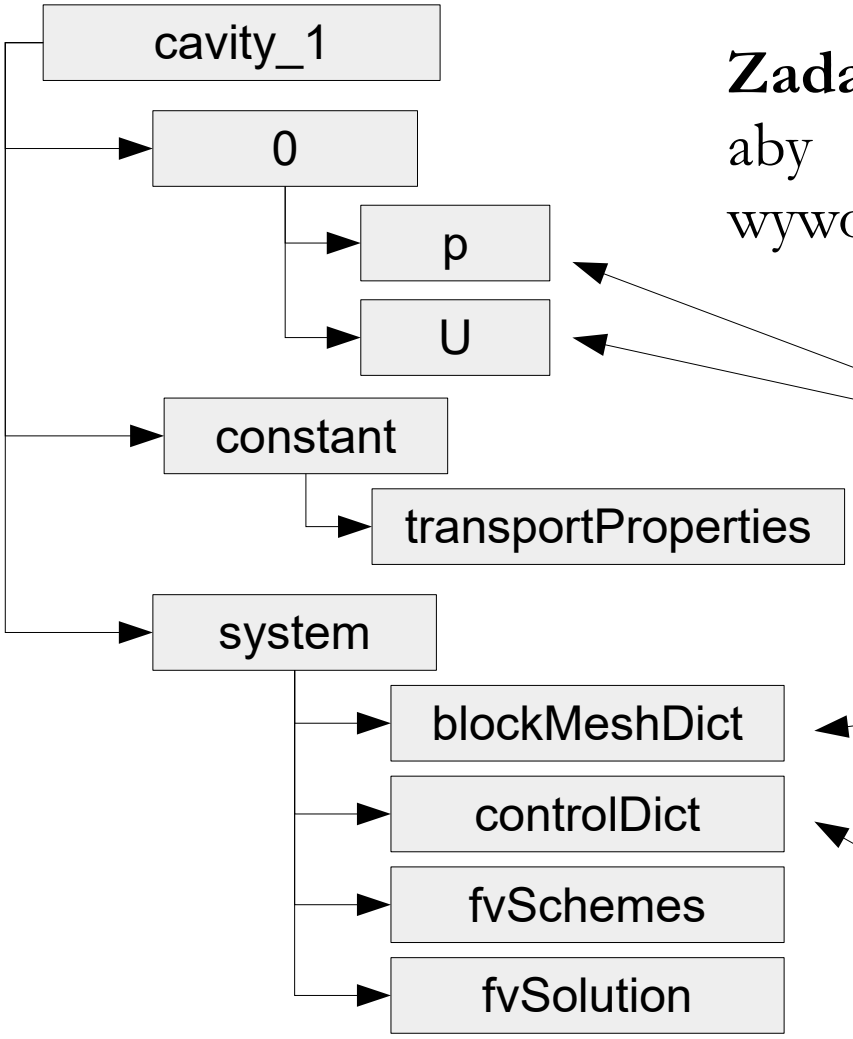






W tej części spróbujemy zmodyfikować przykład **cavity**.

Zadanie 1



Zadanie 1 – zmodyfikować przykład, tak, aby wydłużyć obszar obliczeniowy oraz wywołać przepływ (laminarny) wzdłuż osi X.

Zmienić nazwy i typy brzegów (**boundaryField**).

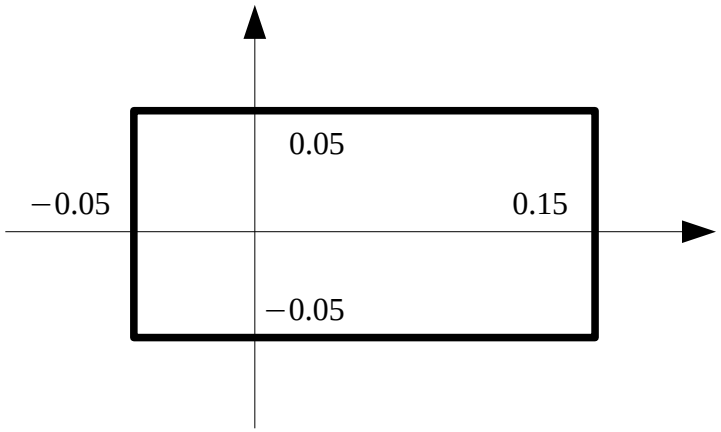
Zmienić współrzędne węzłów (**vertices**)
Zmienić rozmiar siatki (**blocks**).
Zmodyfikować nazwy
oraz typy brzegów (**boundary**).

Zmniejszyć krok czasowy.

Zadanie 1 – słownik blockMeshDict

bez skalowania

```
convertToMeters 1.0;

vertices
(
    (-0.05 -0.05 -0.005)
    (0.15 -0.05 -0.005)
    (0.15 0.05 -0.005)
    (-0.05 0.05 -0.005)
    (-0.05 -0.05 0.005)
    (0.15 -0.05 0.005)
    (0.15 0.05 0.005)
    (-0.05 0.05 0.005)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (200 100 1) simpleGrading (1 1 1)
);
```



punkty jak na rysunku (w kierunku osi Z ±0.005)

siatka stosowna do proporcji długości boków

Rozmiary domeny wg: Sato M., Kobayashi T.: A fundamental study of the flow past a circular cylinder using Abaqus/CFD. 2012 SIMULIA Community Conference.

Zadanie 1 – słownik blockMeshDict

było

```
convertToMeters 0.1;

vertices
(
    (0 0 0)
    (1 0 0)
    (1 1 0)
    (0 1 0)
    (0 0 0.1)
    (1 0 0.1)
    (1 1 0.1)
    (0 1 0.1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
);
```

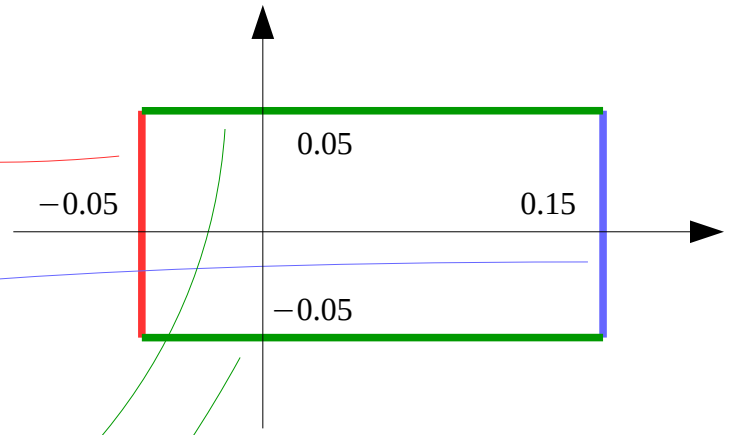
Poprzednio dolny zakres osi X definiowało 0 występujące w pierwszej kolumnie – wystarczy zamienić tę wartość na obecną minimalną wartość zakresu tej osi (-0.05). Analogicznie podmieniamy pozostałe współrzędne.

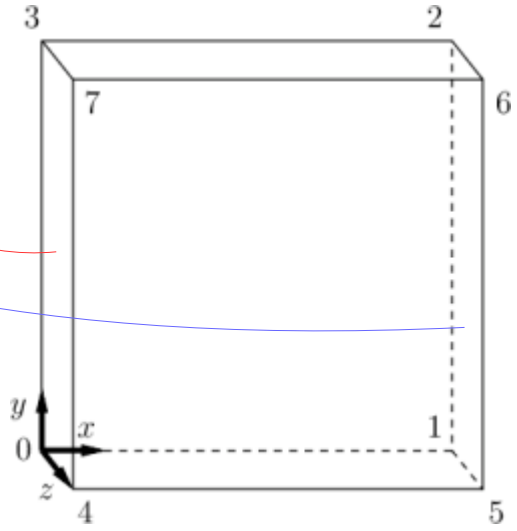
jest

```
convertToMeters 1.0;

vertices
(
    (-0.05 -0.05 -0.005)
    (0.15 -0.05 -0.005)
    (0.15 0.05 -0.005)
    (-0.05 0.05 -0.005)
    (-0.05 -0.05 0.005)
    (0.15 -0.05 0.005)
    (0.15 0.05 0.005)
    (-0.05 0.05 0.005)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (200 100 1) simpleGrading (1 1 1)
);
```

Zadanie 1 – słownik blockMeshDict

```
boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (1 2 6 5)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (0 1 5 4)
            (2 3 7 6)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);
```





Korzystamy z rysunku tego samego bloku co w przykładzie **cavity** (topologia układu się nie zmieniła!).

Zadanie 1 – słownik blockMeshDict

```
boundary
(
    movingWall
    {
        type wall;
        faces
        (
            (3 7 6 2)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);
```

było

jest

```
boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (1 2 6 5)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (0 1 5 4)
            (2 3 7 6)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);
```

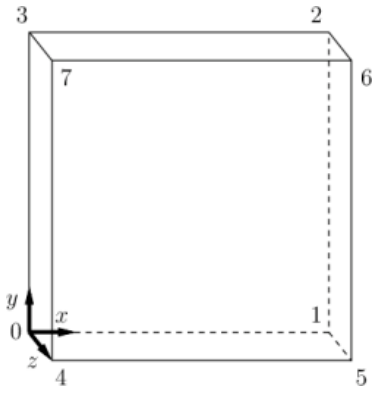
Poprzednia ścianka ruchoma staje się zwykłą ścianką (nadal typu **wall**).

Poprzednia ścianka lewa (**wall**) staje się wlotem (**patch**).

Poprzednia ścianka prawa (**wall**) staje się wylotem (**patch**).

Poprzednia ścianka dolna (**wall**) pozostaje bez zmian (**wall**).

Przód i tył pozostaje bez zmian (**empty**).



Zadanie 1 – słownik p

było

```
boundaryField
{
    movingWall
    {
        type            zeroGradient;
    }

    fixedWalls
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}
```

jest

```
boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    fixedWalls
    {
        type            zeroGradient;
    }
    frontAndBack
    {
        type            empty;
    }
}
```

W słowniku **blockMeshDict** zdefiniowano cztery nazwy – teraz trzeba dla nich określić warunki brzegowe.
Warunek **zeroGradient** oznacza zerowanie strumienia danego pola na kierunku prostopadłym do brzegu.

Zadanie 1 – słownik U

było

```
boundaryField
{
    movingWall
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }

    fixedWalls
    {
        type            noSlip;
    }

    frontAndBack
    {
        type            empty;
    }
}
```

jest

```
boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }
    outlet
    {
        type            zeroGradient;
    }
    fixedWalls
    {
        type            noSlip;
    }

    frontAndBack
    {
        type            empty;
    }
}
```

Na wlocie zdefiniowano jednorodne pole prędkości o wartości równej 1 [m/s] w kierunku osi X oraz 0 [m/s] w pozostałych kierunkach.

Zadanie 1 – słownik controlDict

```
application     icoFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         1.0;
deltaT          0.005;
writeControl    timeStep;
writeInterval   20;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
```

jest

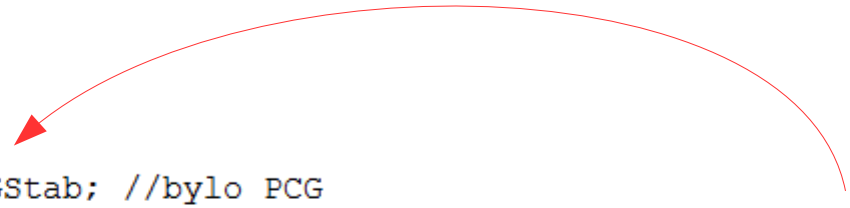
było

```
application     icoFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         0.1;
deltaT          0.000001;
writeControl    timeStep;
writeInterval   10000;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
```

Przy wyższych wartościach kroku czasowego obliczenia nie były zbieżne.

Zadanie 1 – słownik fvSolution

```
solvers
{
    p
    {
        solver          PBiCGStab; //bylo PCG
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0.05;
    }

    pFinal
    {
        $p;
        relTol          0;
    }
```

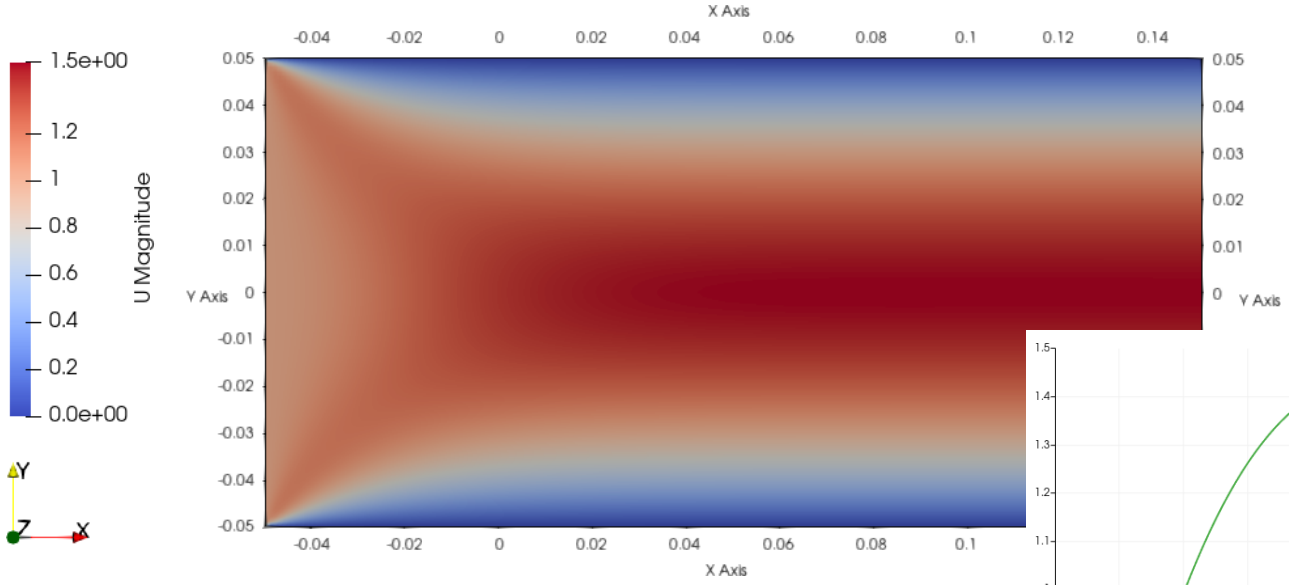


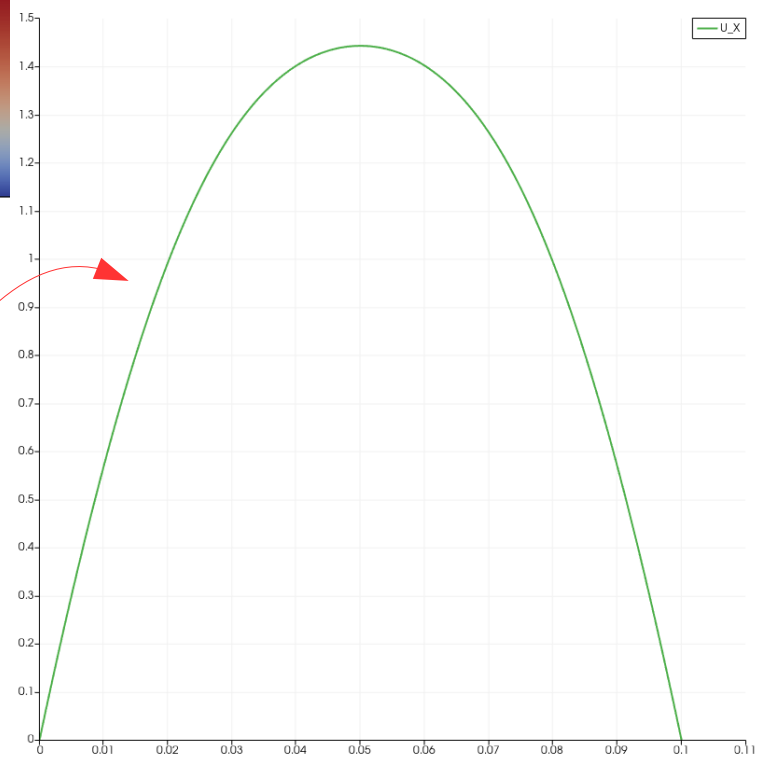
W przykładzie zmieniono solwer na nowszy, z godnie z rekomendacją zawartą w dokumentacji.

Solver	Keyword
Preconditioned (bi-)conjugate gradient	`PCG/PBiCG`†
Stabilized Preconditioned (bi-)conjugate gradient (recommended over PCG/PBiCG)	`PBiCGStab`
Solver using a smoother	`smoothSolver`
Generalised geometric-algebraic multi-grid	`GAMG`
Diagonal solver for explicit systems	`diagonal`

†`PCG` for symmetric matrices, `PBiCG` for asymmetric

Zadanie 1 – wyniki

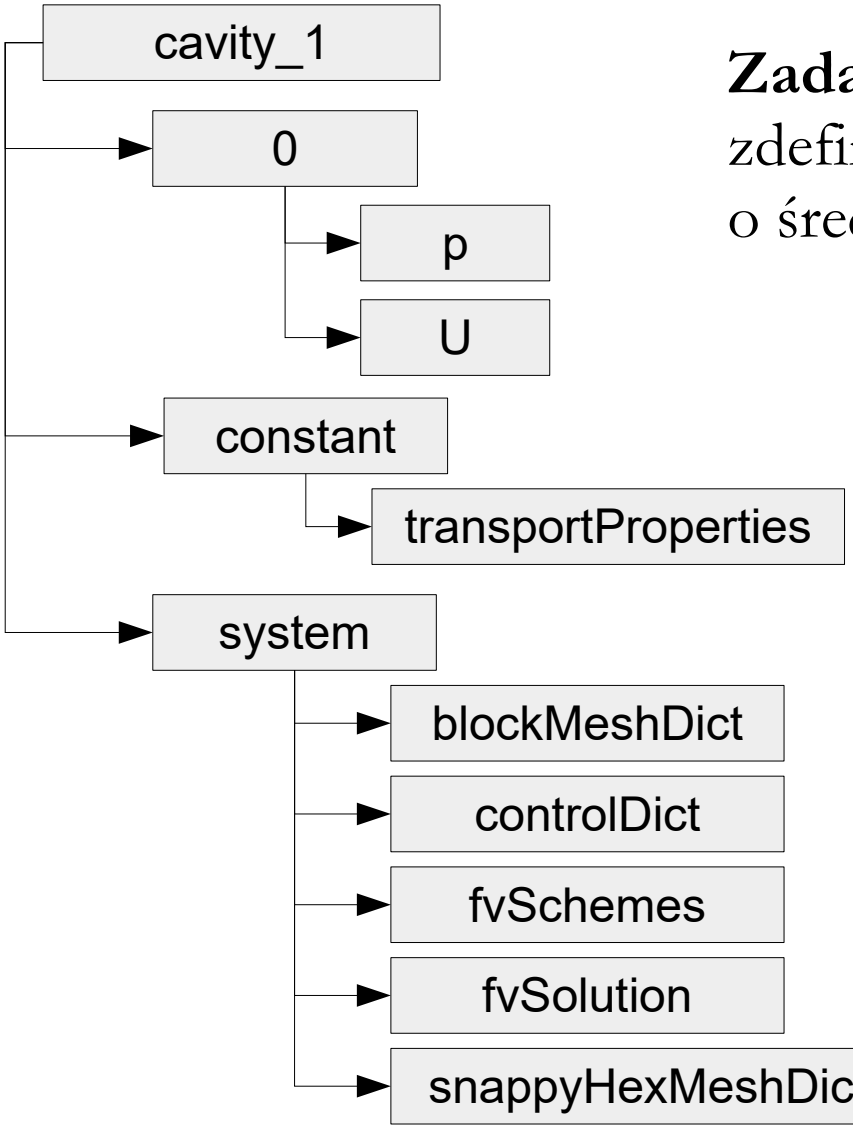




Pole modułu prędkości oraz profil prędkości w kierunku Y dla X = 0.1 [m]

Zgodnie z oczekiwaniami,
profil prędkości jest paraboliczny.

Zadanie 2



Zadanie 2 – w środku układu współrzędnych zdefiniować przeszkodę kołową (cylindryczną) o średnicy 8 [mm].

Wszystko będzie takie samo jak w Zadaniu 1.

Jedyny problem to dodanie przeszkody.

Nowe narzędzie do generacji siatki.

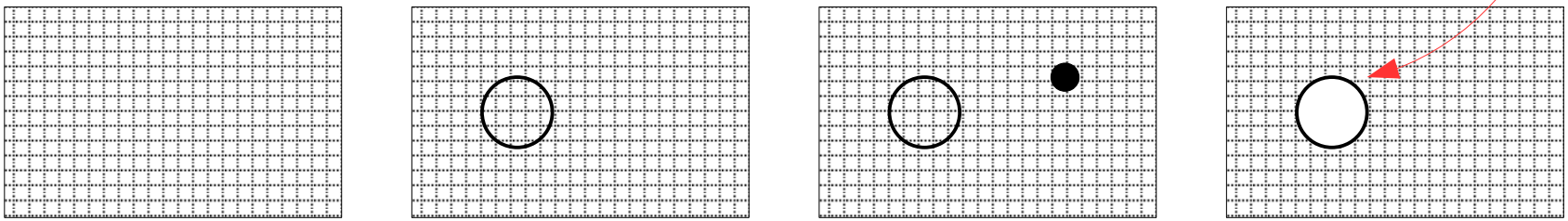
Zadanie 2

snappyHexMesh – narzędzie pakietu OpenFOAM służące do generacji zaawansowanych siatek na bazie siatki tła oraz obiektów zdefiniowanych poprzez kształty podstawowe lub powierzchnie siatkowe (np. pliki *.stl).

Kroki:

- generacja siatki tła za pomocą narzędzia **blockMesh**
- definicja geometrii obiektu
- wskazanie dowolnego punktu wewnątrz siatki
- generacji siatki końcowej za pomocą narzędzia **snappyHexMesh**

Uwaga: powstanie nowy brzeg („cylinder”)!



Zadanie 2 – słownik blockMeshDict

```
convertToMeters 1.0;

vertices
(
    (-0.05 -0.05 -0.005)
    (0.15 -0.05 -0.005)
    (0.15 0.05 -0.005)
    (-0.05 0.05 -0.005)
    (-0.05 -0.05 0.005)
    (0.15 -0.05 0.005)
    (0.15 0.05 0.005)
    (-0.05 0.05 0.005)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (200 100 1) simpleGrading (1 1 1)
);

edges
(
);
```

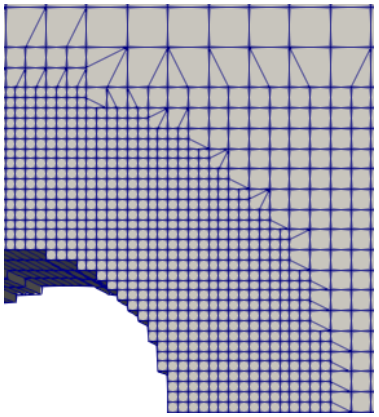
Słownik **blockMeshDict** pozostaje bez zmian – program **blockMesh** ma stworzyć wyłącznie siatkę tła, a w siatce tej nie ma jeszcze obiektu „cylinder” (o którym więcej za chwilę).

```
boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (1 2 6 5)
        );
    }
    fixedWalls
    {
        type wall;
        faces
        (
            (0 1 5 4)
            (2 3 7 6)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);

mergePatchPairs
(
);
```

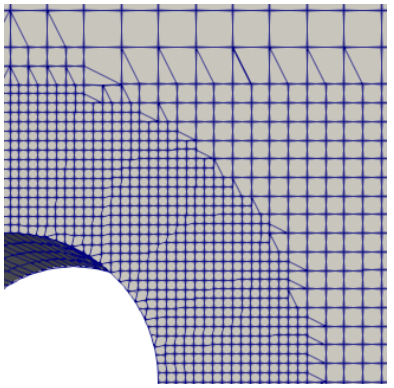
Zadanie 2 – słownik snappyHexMeshDict

```
// Which of the steps to run
castellatedMesh true;
snap            true;
addLayers       true;


// Geometry. Definition of all surfaces.
geometry
{
    cylinder
    {
        type searchableCylinder;
        point1 (  0  0  -0.005);
        point2 (  0  0   0.005);
        radius     0.004;
    }

}
```

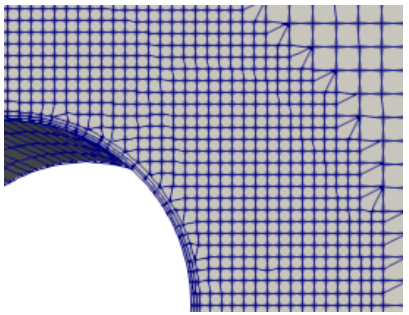
Zagęszcza siatkę i usuwa komórki leżące po drugiej stronie obszaru, w którym zdefiniowano punkt wewnętrzny.



Zmienia kształt komórek brzegowych i dopasowuje je do geometrii.



Dodaje strukturalną siatkę przyścienną na powierzchni wskazanych obiektów.



Zadanie 2 – słownik snappyHexMeshDict

Komentarze:

- w Internecie znaleźć można sporo dobrych filmów lub prezentacji o snappyHexMesh
- program snappyHexMesh wymaga istnienia w podkatalogu „system" pliku „snappyHexMeshDict"
- plik „snappyHexMeshDict" można skopiować z innego przykładu, np. „/tutorials/mesh/snappyHexMesh/flange"
- podczas konfiguracji słownika „snappyHexMeshDict" dobrze jest zmienić nazwę katalogu „0" na inną – inaczej, jeśli nie będzie spójności w nazwach brzegów – nie da się zrobić wizualizacji narzędziem paraFoam (podczas uruchamiania program będzie się zamykał)
- po generacji siatki warto podejrzeć plik „boundary", który pojawi się w podkatalogu „constant/polyMesh" – pojawią się w nim nazwy wszystkich brzegów, które powinny być zdefiniowane w słownikach p i U (i innych, w zależności od użytego solwera).
- najczęstsze błędy w działaniu programu snappyHexMesh wynikają z niepoprawnych wymiarów (np. złego skalowania) lub z istnienia stykających się powierzchni (np. dotykających się w punkcie dwóch sfer)
- wyniki symulacji łatwo można konwertować do innych formatów, np. formatu VTK poleceniem (wtedy można w ParaView wczytać i porównać wyniki kilku różnych symulacji):

```
foamToVTK
```

Zadanie 2 – słownik snappyHexMeshDict

metoda bezpośrednia

```
geometry
{
    cylinder
    {
        type searchableCylinder;
        point1 (  0  0  -0.005);
        point2 (  0  0   0.005);
        radius     0.004;
    }

}
```

W przypadku prostych kształtów można podać geometrię bezpośrednio za pomocą liczb.

W jednym słowniku może być wiele obiektów o różnym typie, ale każdy musi mieć inną nazwę.

W tym przypadku nazwą nowego brzegu będzie słowo „cylinder”.

metoda pośrednia

```
geometry
{
    cylinder.stl
    {
        type triSurfaceMesh;
        scale 0.001;
        name cylinder;
    }

};
```

Geometrię można wczytać z pliku (plików).

Plik *.stl (lub inny dopuszczalny format) należy umieścić w podkatalogu „constant/triSurface”
(trzeba najpierw stworzyć ten katalog).

Nazwę nowego brzegu definiuje się parametrem **name**.

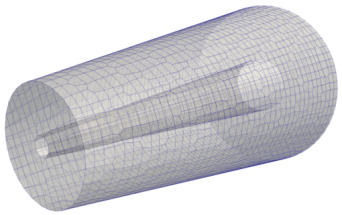
Jeśli zachodzi potrzeba, to współrzędne zawarte w pliku można przeskalować (tu trzeba było to zrobić, gdyż FreeCAD domyślnie stosuje milimetry, a nie metry).

Zadanie 2 – słownik snappyHexMeshDict

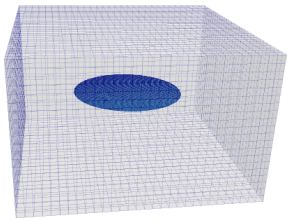
Dopuszczalne rodzaje obiektów:

- searchableBox – czworościan
- searchableCone – stożek (również z współosiowym otworem) (a)
- searchableCylinder – cylinder
- searchableDisk – dysk o zerowej grubości (b)
- searchableExtrudedCircle – wyciągnięty kształt kołowy
- searchablePlane – nieskończona płaszczyzna (c)
- searchablePlate – skończona płaszczyzna (prostokąt) (d)
- searchableRotatedBox – czworościan obrócony względem osi (e)
- searchableSphere – sfera
- searchableSurfaceCollection – zbiór powierzchni
- searchableSurfaceWithGaps – zbiór powierzchni

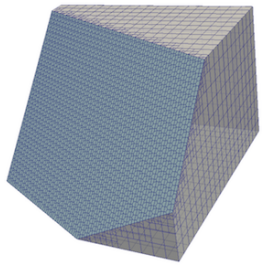
a)



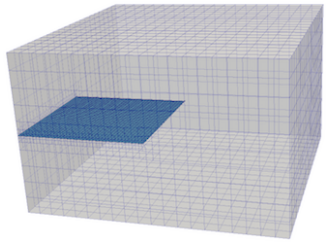
b)



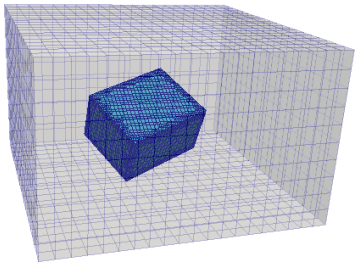
c)



d)



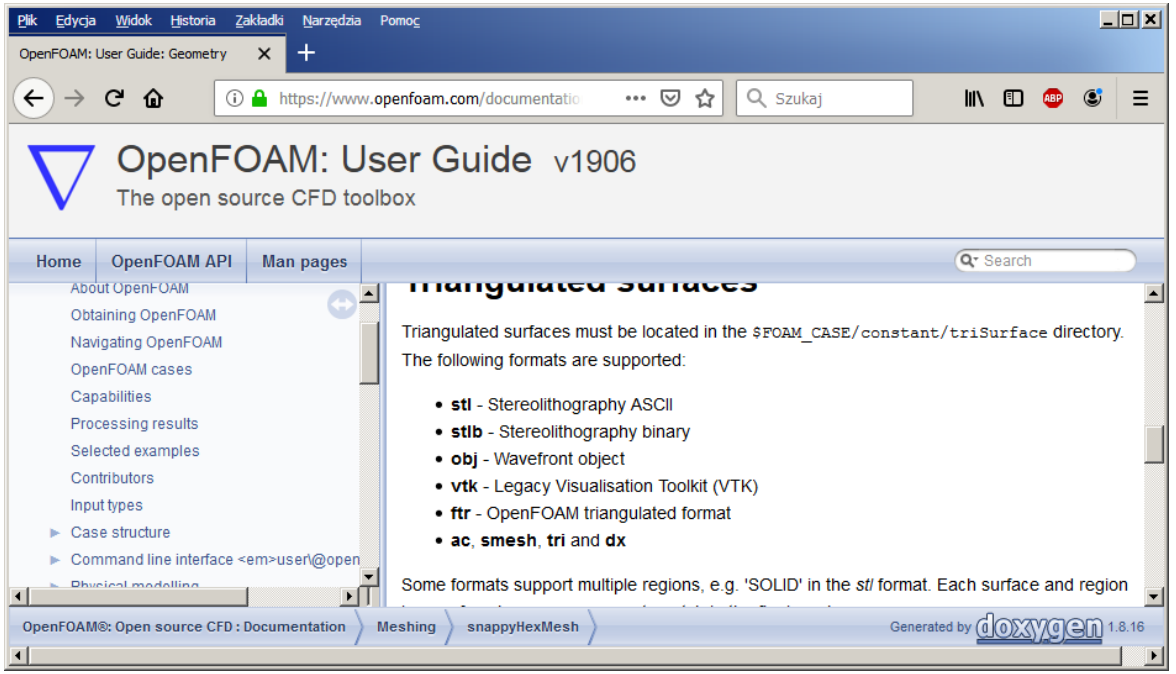
e)



Zadanie 2 – słownik snappyHexMeshDict

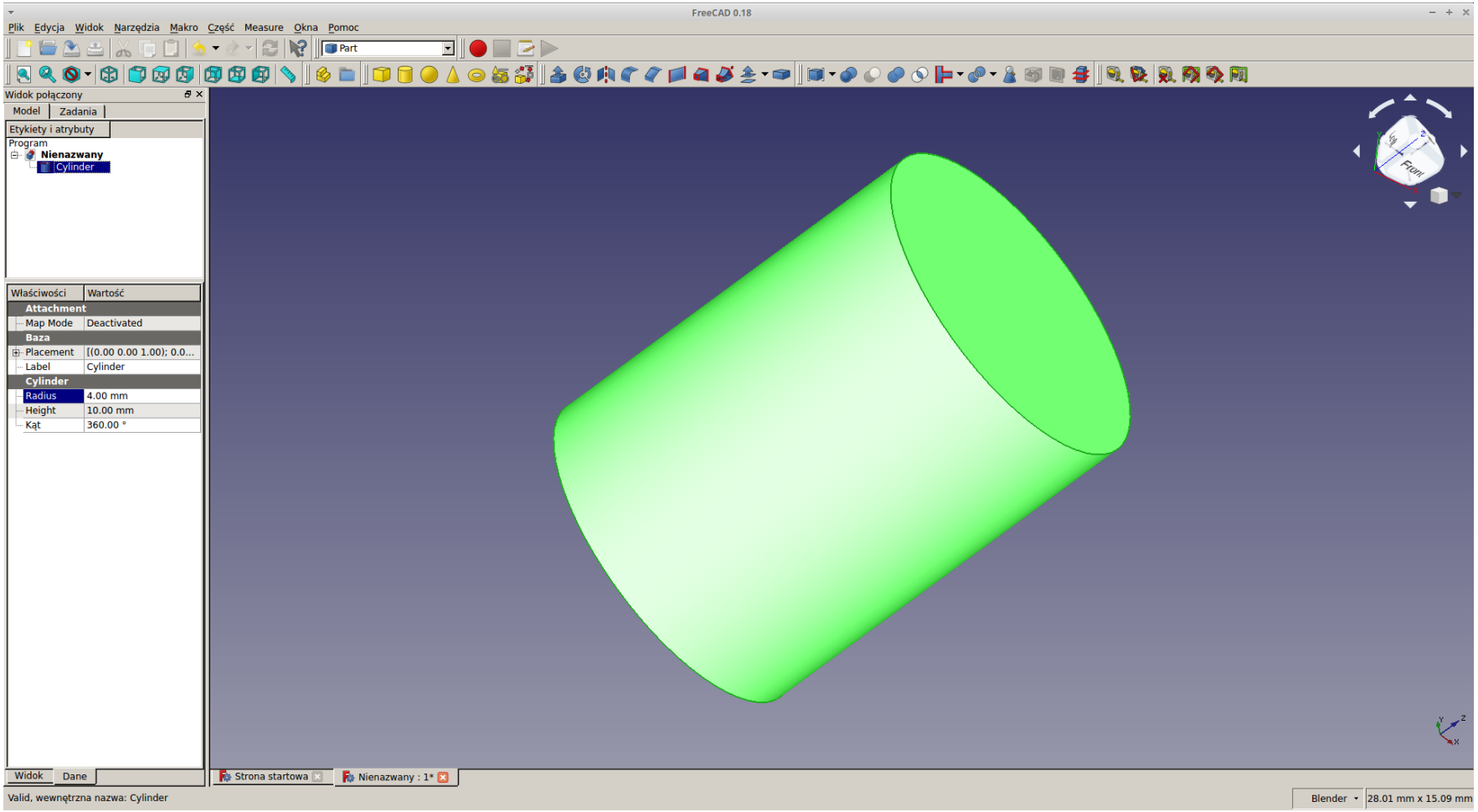
Dopuszczalne formaty plików geometrii to:

- stl – format STL ASCII
- stlb – format STL binarny
- obj – Wavefront object
- vtk – format VTK
- ftr – format siatek OpenFOAM
- ac, smesh, tri oraz dx



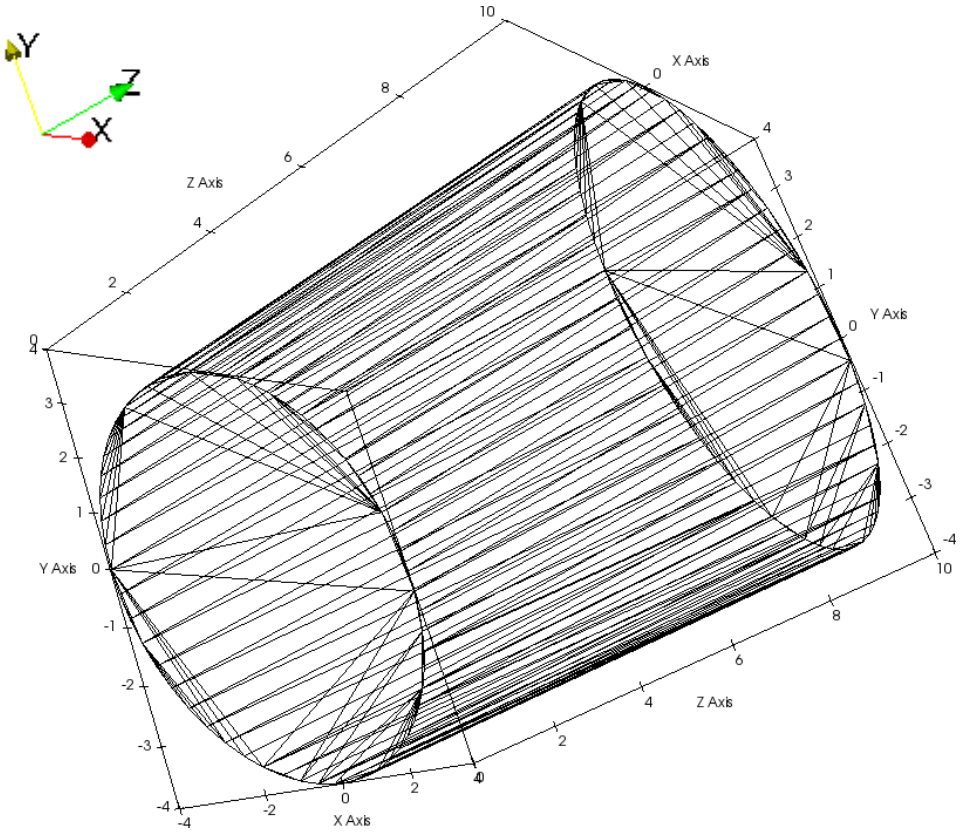
Informacje wg:
https://www.openfoam.com/documentation/guides/latest/doc/guide-meshing-snappyhexmesh-geometry.html

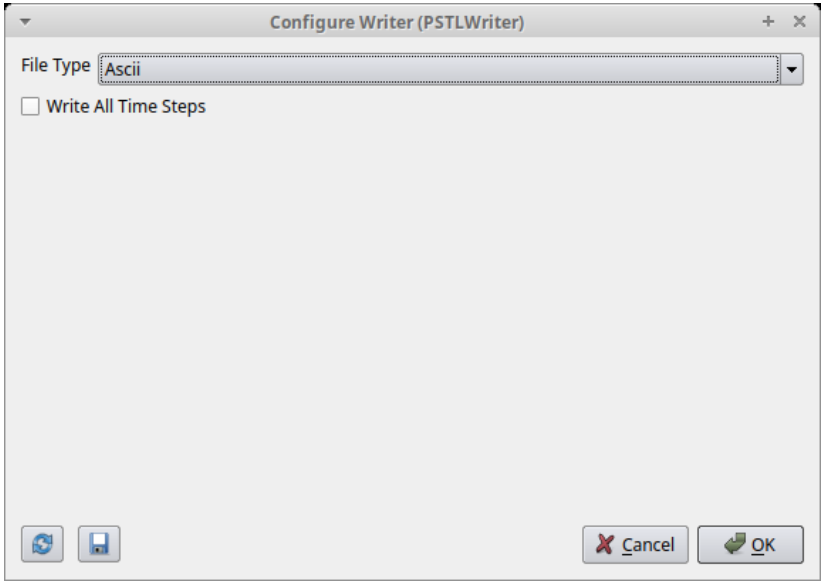
Zadanie 2 – tworzenie geometrii STL



W przykładzie cylinder narysowano w programie FreeCAD, a następnie eksportowano do formatu *.stl.

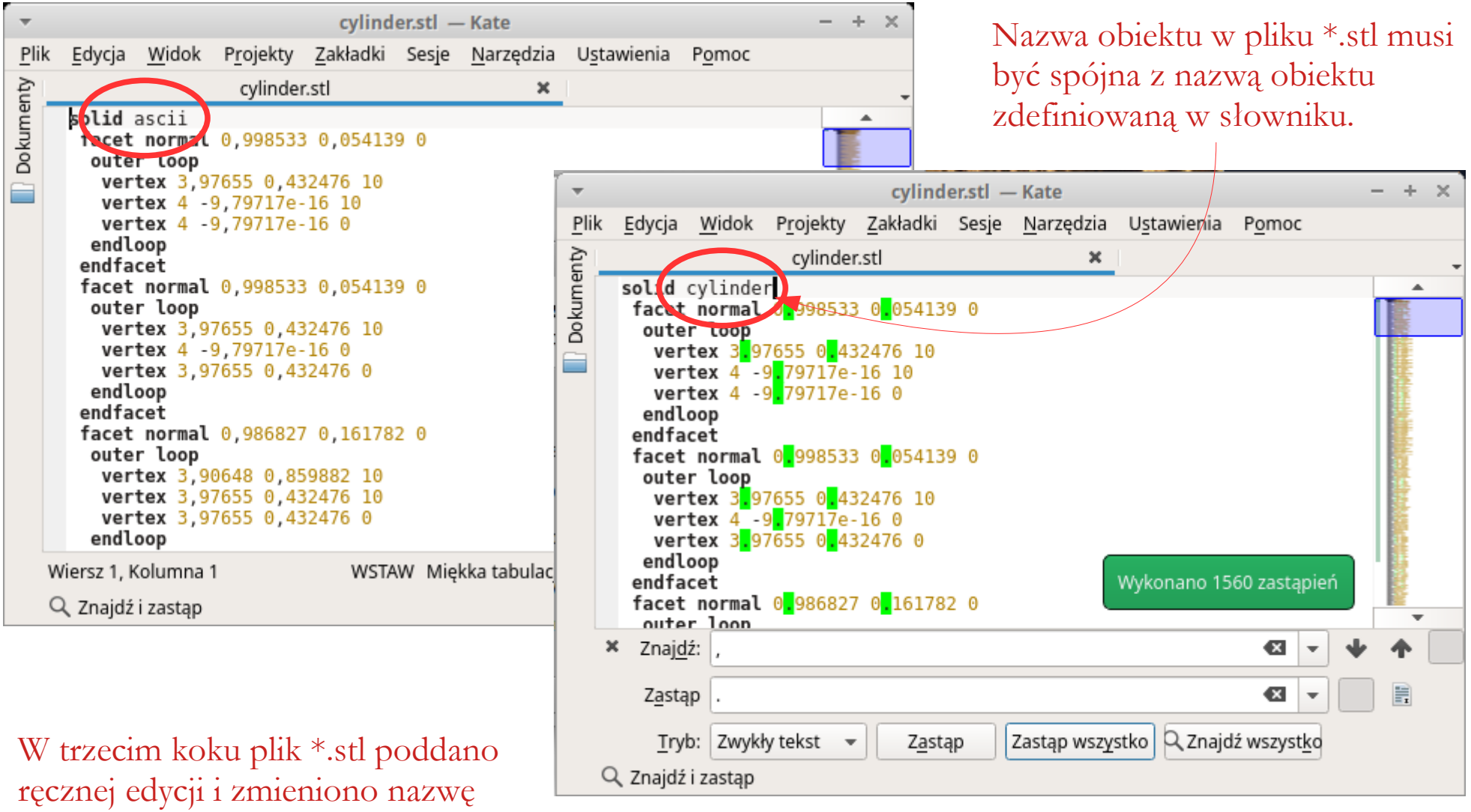
Zadanie 2 – tworzenie geometrii STL





Ponieważ FreeCad zapisał plik *.stl w postaci binarnej, został on otwarty w programie ParaView, a następnie ponownie zapisany pod inną nazwą – podczas tej operacji można wybrać postać zapisu: binarną lub ASCII.

Zadanie 2 – tworzenie geometrii STL



Nazwa obiektu w pliku *.stl musi być spójna z nazwą obiektu zdefiniowaną w słowniku.

W trzecim koku plik *.stl poddano ręcznej edycji i zmieniono nazwę obiektu STL (domyślnie było ascii) oraz znak separatora dziesiętnego.

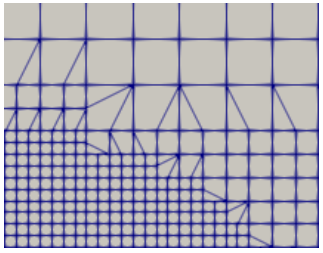
Zadanie 2 – słownik snappyHexMeshDict

```
refinementSurfaces
{
    cylinder
    {
        level (0 0);
    }

}
```

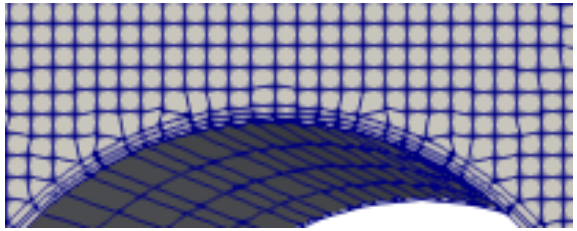
```
layers
{
    cylinder
    {
        nSurfaceLayers 3;
    }

}
```

Ważne jest, że nazwa obiektu powinna pojawić się w słowniku **snappyHexMeshDict** jeszcze dwukrotnie:

- w sekcji **refinementSurfaces**, w której definiuje się minimalny i maksymalny poziom podziału komórek początkowych na mniejsze (tu 0, czyli bez podziału);

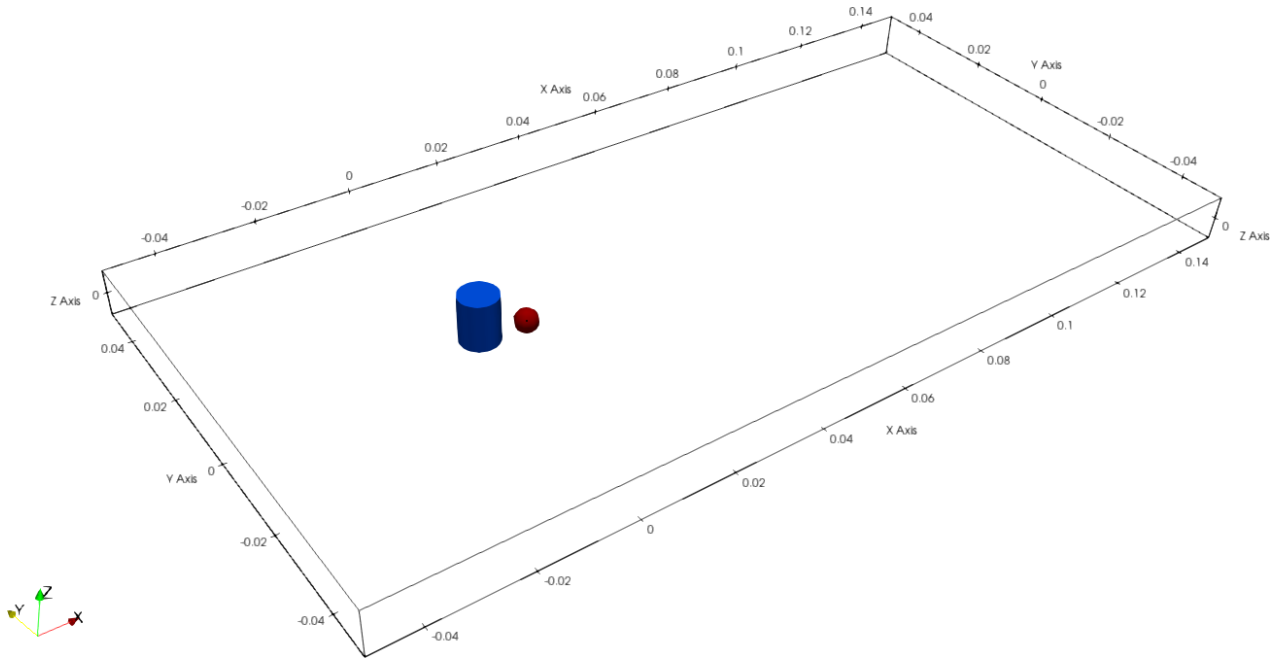


- w sekcji **layers**, w której definiuje się, dla jakich obiektów mają być utworzone siatki przyścienne (liczbę warstw siatki określa parametr **nSurfaceLayers**).



Zadanie 2 – słownik snappyHexMeshDict

```
// Mesh selection
// ~~~~~~~~~~~~~~~
locationInMesh (0.01 0.0 0.0);
```



Bardzo ważnym elementem konfigurowania słownika **snappyHexMeshDict** jest wskazanie punktu znajdującego się w obszarze, w którym ma być wygenerowana siatka.

Gdyby wskazać punkt znajdujący się wewnątrz cylindra, siatka utworzyłaby się wewnątrz cylindra, a nie wokół niego.

Dobrym pomysłem jest wczytanie siatki głównej (tła) po użyciu narzędzia **blockMesh**, a następnie wczytanie pliku *.stl (tu wczytaną geometrię – niebieski cylinder – trzeba było jeszcze w programie ParaView przeskalować!) – teraz można narysować punkt w dowolnym miejscu i sprawdzać, czy jest poprawnie ulokowany (tu punkt zdefiniowany w słowniku ma kolor czerwony).

Zadanie 2 – słowniki p i U

```
boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    fixedWalls
    {
        type            zeroGradient;
    }
    cylinder
    {
        type            zeroGradient;
    }
    frontAndBack
    {
        type            empty;
    }
}
```

```
boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0.0245 0 0);
    }
    outlet
    {
        type            zeroGradient;
    }
    fixedWalls
    {
        type            noSlip;
    }
    cylinder
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    frontAndBack
    {
        type            empty;
    }
}
```

Nie wolno zapomnieć o tym, aby nowy brzeg opisać w słownikach **p** i **U**.
Warunek ciśnieniowy taki sam jak na ściance, warunek prędkościowy – zerowanie wszystkich składowych.

Zadanie 2 – słownik controlDict

```
application     icoFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         1.0;
deltaT          0.0001;
writeControl    timeStep;
writeInterval   1000;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
```

W słowniku **controlDict** warto wydłużyć czas symulacji, zmienić częstotliwość zapisu oraz zwiększyć krok czasowy.

Słowniki **fvSchemes** oraz **fvSolution** pozostają bez zmian.

```
application     icoFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         25;
deltaT          0.005;
writeControl    timeStep;
writeInterval   100;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
```

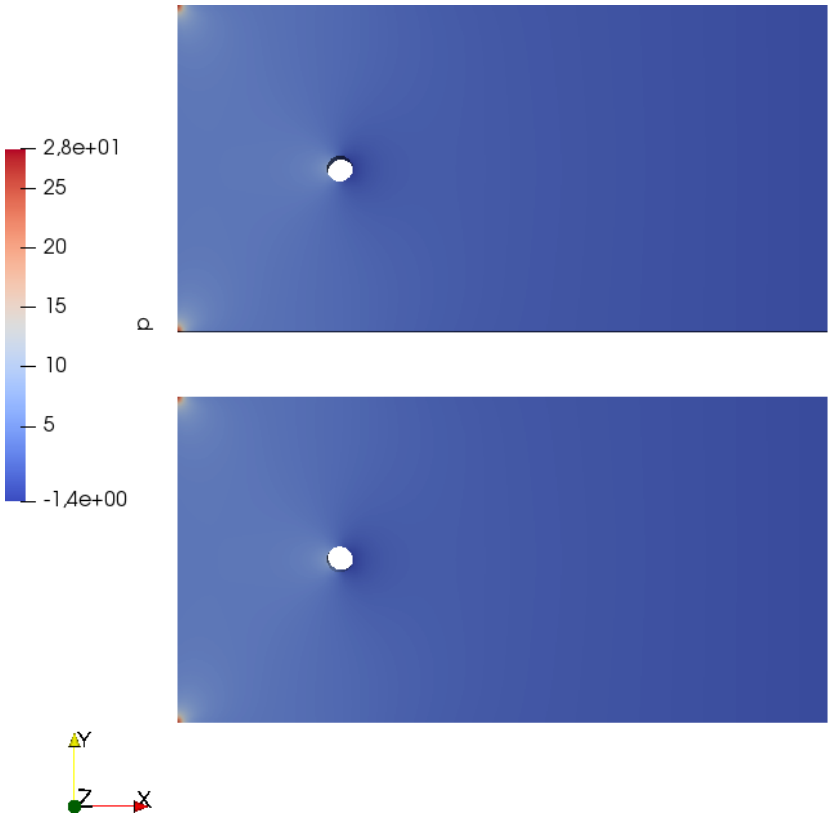
Zadanie 2 – wyniki

Uruchomienie przykładu:

- wygenerować siatkę tła: **blockMesh**
- wygenerować siatkę końcową: **snappyHexMesh -overwrite**
- uruchomić program ParaView: **paraFoam**

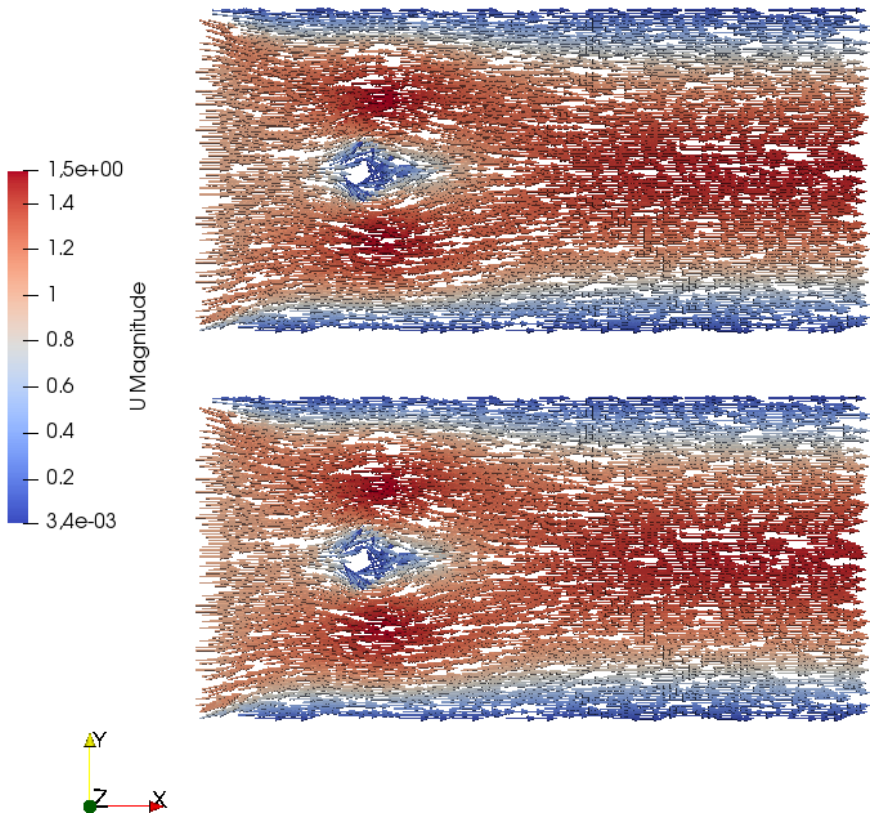
Tu program snappyHexMesh został wywołany z parametrem „overwrite”, co oznacza, że kolejne fazy tworzenia siatki nie będą przechowywane w różnych katalogach.

Zadanie 2 – wyniki



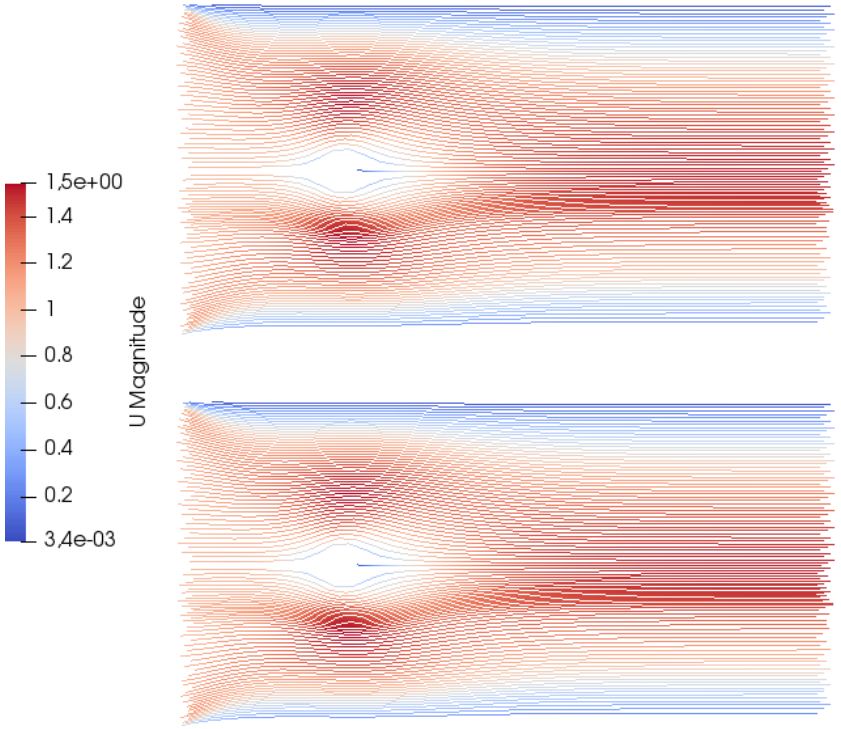
Pole quasi-ciśnień

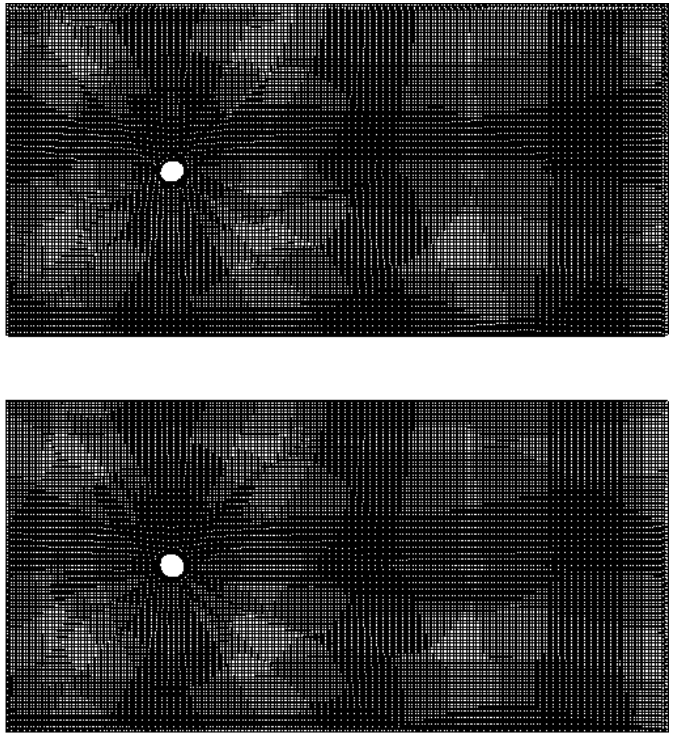
Dla obu sposobów definiowania cylindra wyniki są identyczne.



Pole prędkości

Zadanie 2 – wyniki



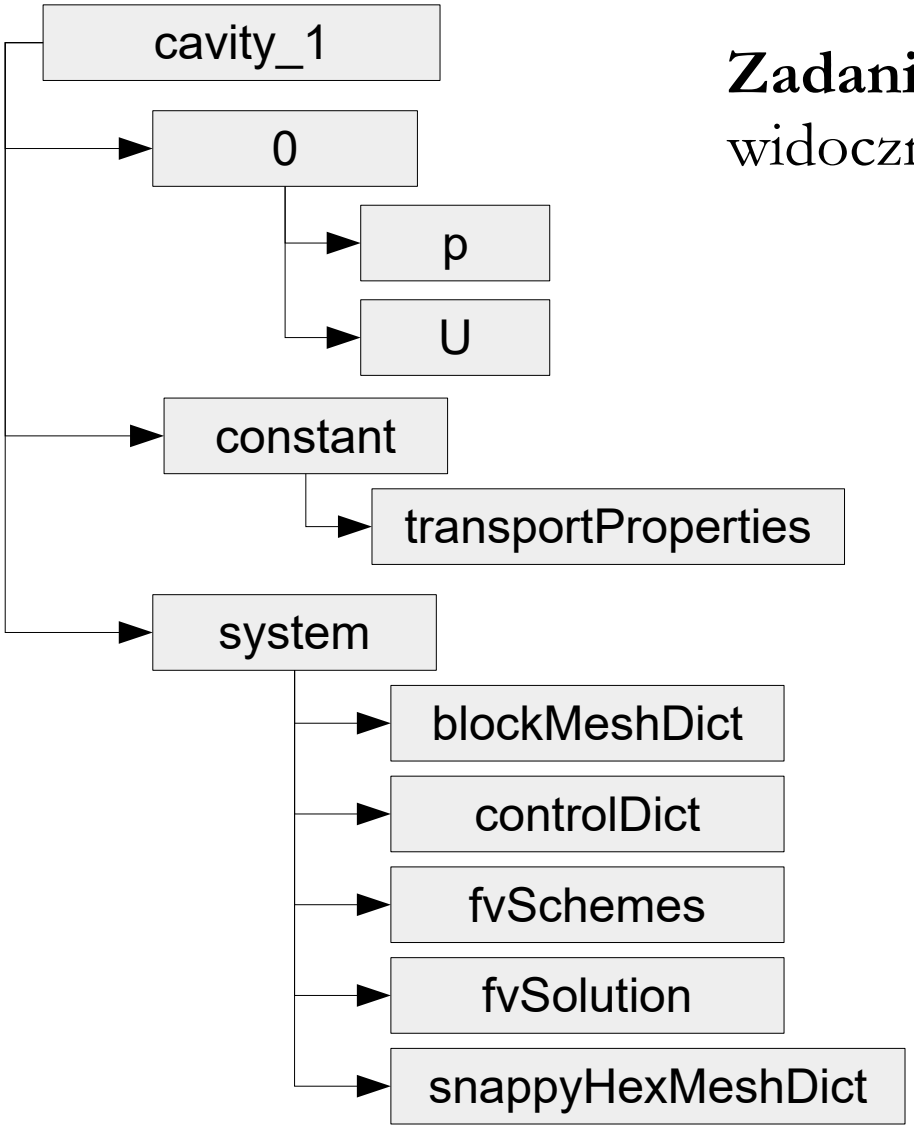


Linie prądu

Dla obu sposobów definiowania cylindra wyniki są identyczne.

Siatka obliczeniowa

Zadanie 3

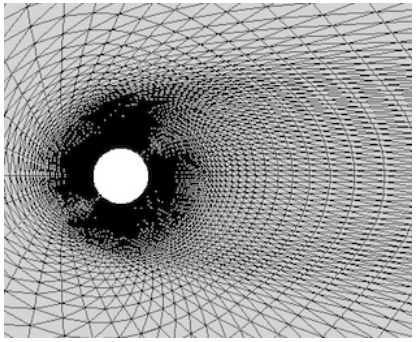
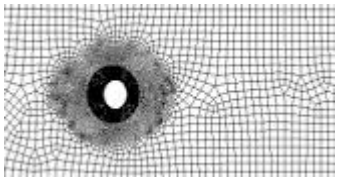
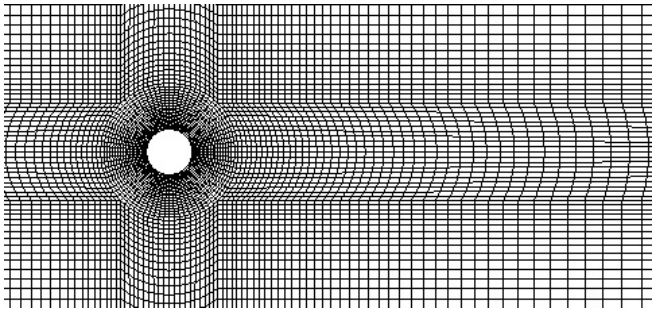
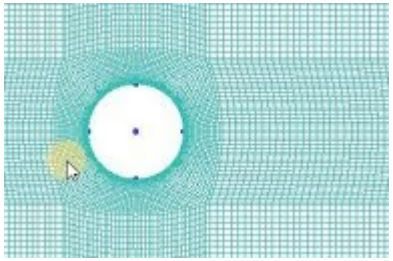
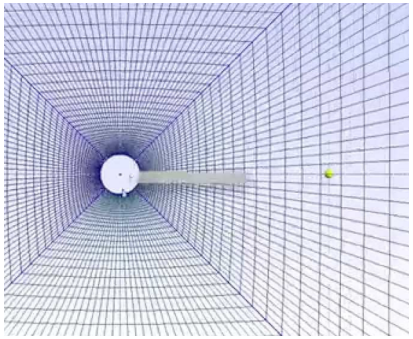
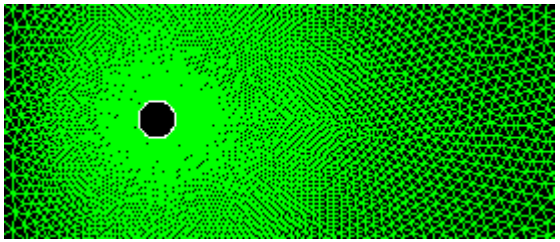
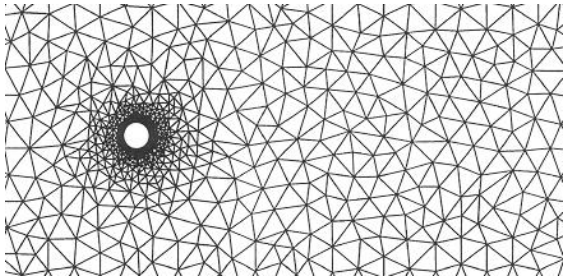


Zadanie 3 – wykonać symulację, w której widoczne będą wiry spływowe von Karmana.

Przykład wirów von Karmana w atmosferze ziemskiej – zdjęcie zrobione w roku 1999 przez satelitę LANDSAT 7 nad wyspą Selkirka (Pacyfik).

Zadanie 3 – lokalne zagęszczenie siatki

- nawet pobieżny przegląd zagadnienia ujawnia, że trzeba zagęścić siatkę wokół opływanego obiektu – **jak to zrobić**?

Zadanie 3 – lokalne zagęszczenie siatki

```
geometry
{
    cylinder
    {
        type searchableCylinder;
        point1 (  0   0  -0.005);
        point2 (  0   0   0.005);
        radius     0.004;
    }

    refCylinder
    {
        type searchableCylinder;
        point1 (  0   0  -0.005);
        point2 (  0   0   0.005);
        radius    0.008;
    }

    refBox
    {
        type searchableBox;
        min ( 0    -0.008 -0.005);
        max ( 0.15  0.008  0.005);
    }
}
```

Siatkę można zagęszczać w dowolnych blokach 3D (zasady są takie same jak podczas definiowania zwykłej geometrii).

Bloki definiuje się w słowniku **snappyHexMeshDict** w sekcji **geometry** – liczba bloków może być dowolna, ważne jedynie, aby każdy blok miał niepowtarzalną nazwę.

W tym przypadku dodano dwa bloki geometryczne:

- cylinder o środku i orientacji takiej samej jak przeszkoda, ale o dwa razy większej średnicy;
- prostopadłościan o wysokości (w kierunku Y) równej średnicy cylindra zdefiniowanego wyżej oraz długości równej prawej części obszaru obliczeniowego.

Aby odróżnić obiekty przeszkód od obiektów służących do zagęszczania siatki, nazwy tych ostatnich poprzedzono tu słowem „ref”.

Zadanie 3 – lokalne zagęszczenie siatki

```
refinementRegions
{
    refCylinder
    {
        mode inside;
        levels ((0 2));
    }
    refBox
    {
        mode inside;
        levels ((0 1));
    }
}
```

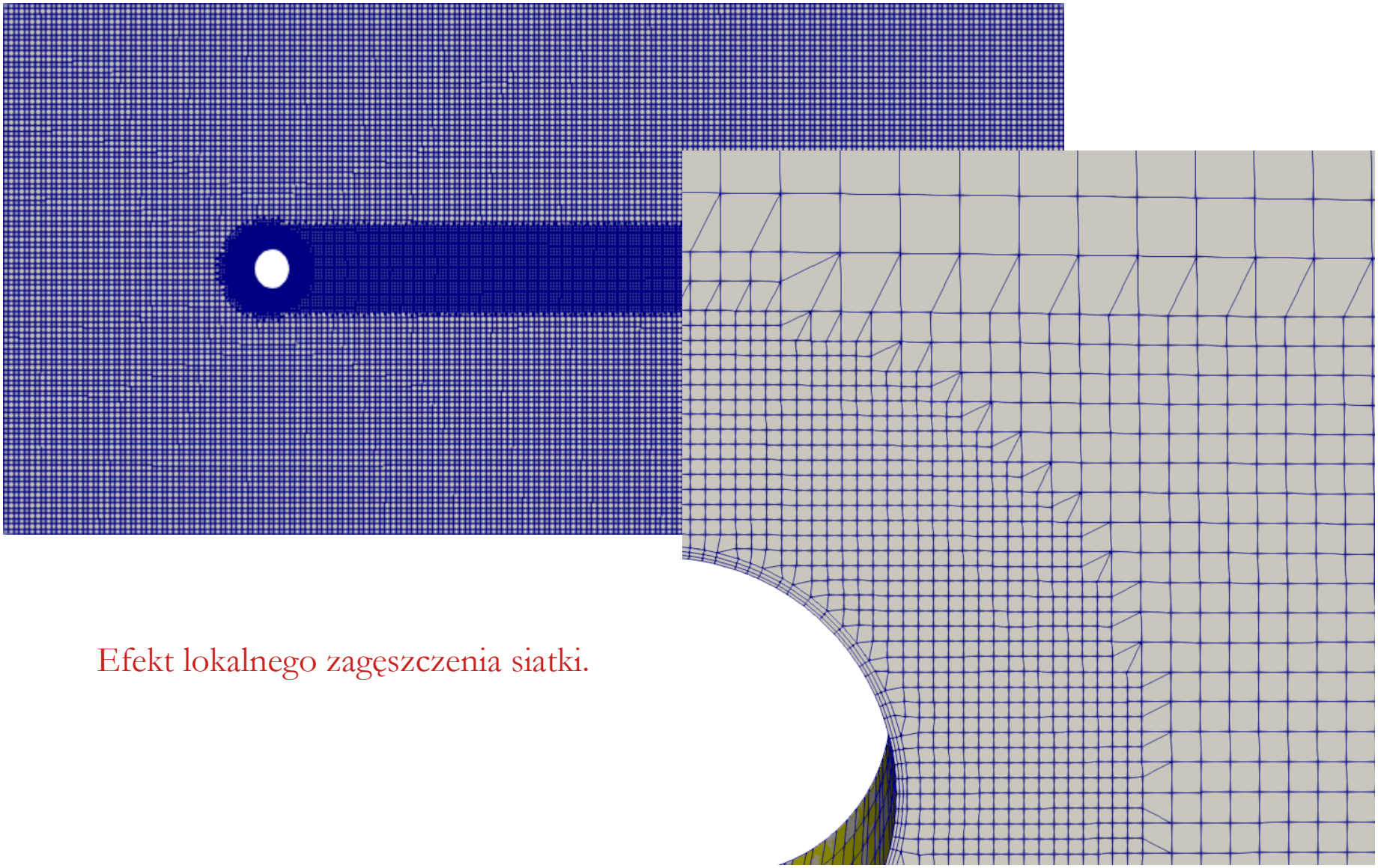
Bloki, w których siatka ma być zagęszczona należy dodać do sekcji **refinementRegions** znajdującej się w słowniku **snappyHexMeshDict.**

Zmienna **levels** określa minimalny i maksymalny stopień zagęszczenia siatki:

- w przypadku **refCylinder** każda dotychczasowa komórka zostanie „przecięta” dwukrotnie, w celu stworzenia komórek mniejszych;
- w przypadku **refBox** każda dotychczasowa komórka zostanie „przecięta” jeden raz.

Aby sprawdzić działanie nowego słownika należy uruchomić program **blockMesh**, a następnie program **snappyHexMesh.**

Zadanie 3 – lokalne zagęszczenie siatki



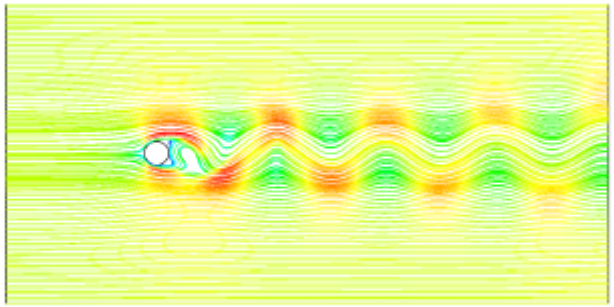
Efekt lokalnego zagęszczenia siatki.

Zadanie 3 – parametry płynu

- podczas definiowania warunków początkowych, brzegowych oraz parametrów medium dobrze jest oprzeć się na konkretnym przykładzie – tu wykorzystany zostanie ponownie artykuł Sato i Kobayashi.

3. Analysis model

The analysis model in this study is shown in Figure 2. The fluid was assumed to be water at 293 K where the density was $\rho = 998.204$ (kg/m^3) and the coefficient of viscosity was $\mu = 1.002 \times 10^{-3}$ (Pa sec). The diameter of the cylinder was assumed to be $d = 8$ mm. Naturally, an analysis of the von Karman vortex should begin with two-dimensional behaviors of the flow around the circular cylinder. However, as, Abaqus/CFD is designed to deal with fluids with three-dimensional behaviors, the fluid was modeled using the three-dimensional element (FC3D8), which has one division and a thickness of 1 mm.



(h) CASE8 *Re* = 195

	Re (-)	*U* (mm/sec)		*Re* (-)	*U* (mm/sec)
CASE1	0.038	4.77×10^{-3}	CASE5	26	3.26
CASE2	0.1	1.25×10^{-2}	CASE6	50	6.27
CASE3	1.1	0.138	CASE7	100	12.5
CASE4	20	2.510	CASE8	195	24.5

Uwaga na jednostki!

Sato M., Kobayashi T.: A fundamental study of the flow past a circular cylinder using Abaqus/CFD. 2012 SIMULIA Community Conference.

Zadanie 3 – parametry płynu

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0.0245 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0.0245 0 0);
    }
    outlet
    {
        type            zeroGradient;
    }
    fixedWalls
    {
        type            noSlip;
    }
    cylinder
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    frontAndBack
    {
        type            empty;
    }
}
```

W słowniku **U** podajemy wartość z artykułu (dla CASE8).

Słownik **p** pozostaje bez zmian.

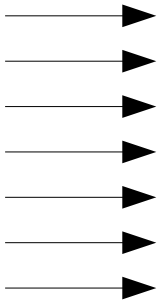
W słowniku **transportProperties** zmieniamy wartość lepkości kinematycznej – można ją policzyć na podstawie danych zawartych w artykule.

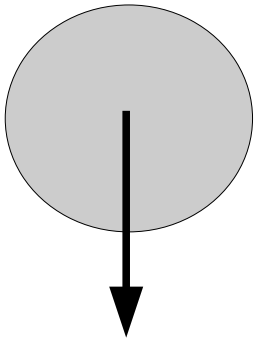
$$\nu = \frac{\mu}{\rho} = \frac{0.001002}{998.204} = 1.0038\text{E-}06 \left[\frac{m^2}{s}\right]$$

```
nu              [0 2 -1 0 0 0 0] 1.0038E-06;
```

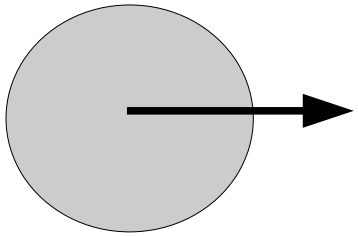
Zadanie 3 – monitorowanie sił

- jak monitorować siłę oporu i siłę unoszenia na cylindrze?





$$F_l = \frac{1}{2} c_l \rho u^2 A_{xz}$$



$$F_d = \frac{1}{2} c_d \rho u^2 A_{yz}$$

F_l - siła unoszenia (lift force) [N]

c_l - współczynnik unoszenia [-]

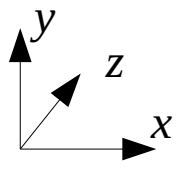
ρ - gęstość płynu [kg/m^3]

u - prędkość płynu [m/s]

$A_{xz,yz}$ - pole powierzchni ścianki w płaszczyźnie XZ lub YZ [m^2]

F_d - siła oporu (drag force) [N]

c_d - współczynnik oporu [-]



Zadanie 3 – monitorowanie sił

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

forceCoeffs1
{
    type            forceCoeffs;

    libs            ("libforces.so");

    writeControl    timeStep;
    timeInterval    1;

    log             yes;

    patches         (cylinder);
    rho             rhoInf;
    rhoInf          998.204;
    liftDir         (0 1 0);
    dragDir         (1 0 0);
    CofR            (0 0 0);
    pitchAxis       (0 0 1);
    magUInf         0.0245;
    lRef            8E-05;
    Aref            8E-05;
}

// ************************************************************************* //
```

W katalogu **system** tworzymy nowy plik
- nazwa jest dowolna, ale nie należy stosować znaków narodowych ani spacji.

Zadanie 3 – monitorowanie sił

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

forceCoeffs1
{
    type            forceCoeffs;

    libs            ("libforces.so");

    writeControl    timeStep;
    timeInterval    1;

    log             yes;

    patches         (cylinder);
    rho             rhoInf;
    rhoInf          998.204;
    liftDir         (0 1 0);
    dragDir         (1 0 0);
    CofR            (0 0 0);
    pitchAxis       (0 0 1);
    magUInf         0.0245;
    lRef            8E-05;
    Aref            8E-05;
}

// ************************************************************************* //
```

Definicja typu monitora.

Zadanie 3 – monitorowanie sił

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

forceCoeffs1
{
    type            forceCoeffs;

    libs            ("libforces.so");

    writeControl    timeStep;
    timeInterval    1;

    log             yes;

    patches         (cylinder);
    rho             rhoInf;
    rhoInf          998.204;
    liftDir         (0 1 0);
    dragDir         (1 0 0);
    CofR            (0 0 0);
    pitchAxis       (0 0 1);
    magUInf         0.0245;
    lRef            8E-05;
    Aref            8E-05;
}

// ************************************************************************* //
```

Nazwa biblioteki, w której znajduje się narzędzie do monitorowania sił.

Zadanie 3 – monitorowanie sił

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

forceCoeffs1
{
    type            forceCoeffs;

    libs            ("libforces.so");

    writeControl    timeStep;
    timeInterval    1;

    log             yes;

    patches         (cylinder);
    rho             rhoInf;
    rhoInf          998.204;
    liftDir         (0 1 0);
    dragDir         (1 0 0);
    CofR            (0 0 0);
    pitchAxis       (0 0 1);
    magUInf         0.0245;
    lRef            8E-05;
    Aref            8E-05;
}

// ************************************************************************* //
```

Informacja, że monitorowanie ma się odbywać względem czasu, a zapis ma następować w każdym kroku czasowym.

Zadanie 3 – monitorowanie sił

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

forceCoeffs1
{
    type            forceCoeffs;

    libs            ("libforces.so");

    writeControl    timeStep;
    timeInterval    1;

    log             yes;

    patches         (cylinder);
    rho             rhoInf;
    rhoInf          998.204;
    liftDir         (0 1 0);
    dragDir         (1 0 0);
    CofR            (0 0 0);
    pitchAxis       (0 0 1);
    magUInf         0.0245;
    lRef            8E-05;
    Aref            8E-05;
}

// ************************************************************************* //
```

Informacja, czy zapisywać dane czy też nie.

Zadanie 3 – monitorowanie sił

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

forceCoeffs1
{
    type            forceCoeffs;

    libs            ("libforces.so");

    writeControl    timeStep;
    timeInterval    1;

    log             yes;

    patches         (cylinder);
    rho             rhoInf;
    rhoInf          998.204;
    liftDir         (0 1 0);
    dragDir         (1 0 0);
    CofR            (0 0 0);
    pitchAxis       (0 0 1);
    magUInf         0.0245;
    lRef            8E-05;
    Aref            8E-05;
}

// ************************************************************************* //
```

Nazwa powierzchni, która ma być monitorowana (może to być tylko obiekt typu **wall**).

Zadanie 3 – monitorowanie sił

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

forceCoeffs1
{
    type            forceCoeffs;

    libs            ("libforces.so");

    writeControl    timeStep;
    timeInterval    1;

    log             yes;

    patches         (cylinder);
    rho             rhoInf;
    rhoInf          998.204;
    liftDir         (0 1 0);
    dragDir         (1 0 0);
    CofR            (0 0 0);
    pitchAxis       (0 0 1);
    magUInf         0.0245;
    lRef            8E-05;
    Aref            8E-05;
}

// ************************************************************************* //
```

Definicja nazwy zmiennej, w której przechowywana jest wartość gęstości w przepływie niezaburzonym (z dala od obiektu) oraz jej wartość.

Zadanie 3 – monitorowanie sił

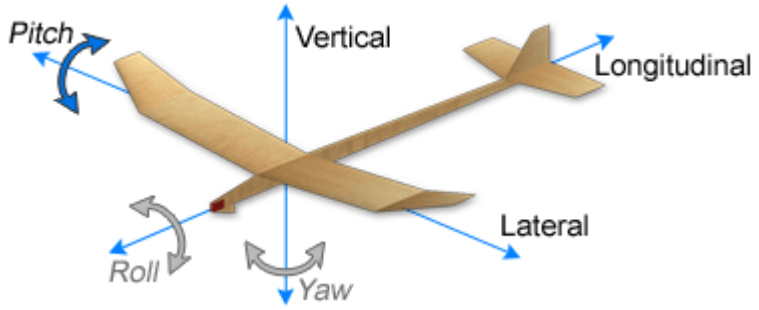
```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

forceCoeffs1
{
    type            forceCoeffs;

    libs            ("libforces.so");

    writeControl    timeStep;
    timeInterval    1;

    log             yes;

    patches         (cylinder);
    rho             rhoInf;
    rhoInf          998.204;
    liftDir         (0 1 0);
    dragDir         (1 0 0);
    CofR            (0 0 0);
    pitchAxis       (0 0 1);
    magUInf         0.0245;
    lRef            8E-05;
    Aref            8E-05;
}

// ************************************************************************* //
```

Określenie osi, względem których mają być obliczane siły, tu:

- siła unoszenia liczona jest względem osi Y;
- siła oporu liczona jest względem osi X.

Zadanie 3 – monitorowanie sił

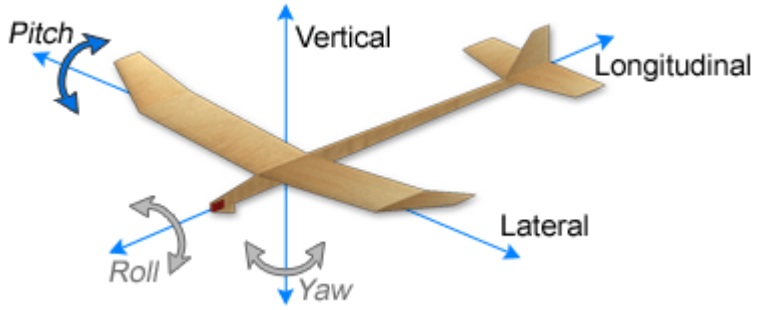
```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

forceCoeffs1
{
    type            forceCoeffs;

    libs            ("libforces.so");

    writeControl    timeStep;
    timeInterval    1;

    log             yes;

    patches         (cylinder);
    rho             rhoInf;
    rhoInf          998.204;
    liftDir         (0 1 0);
    dragDir         (1 0 0);
    CofR            (0 0 0);
    pitchAxis       (0 0 1);
    magUInf         0.0245;
    lRef            8E-05;
    Aref            8E-05;
}

// ************************************************************************* //
```



Współrzędne środka osi odpowiedzialnej za pochylenie obiektu (pitch).

Zadanie 3 – monitorowanie sił

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

forceCoeffs1
{
    type            forceCoeffs;

    libs            ("libforces.so");

    writeControl    timeStep;
    timeInterval    1;

    log             yes;

    patches         (cylinder);
    rho             rhoInf;
    rhoInf          998.204;
    liftDir         (0 1 0);
    dragDir         (1 0 0);
    CofR            (0 0 0);
    pitchAxis       (0 0 1);
    magUInf         0.0245;
    lRef            8E-05;
    Aref            8E-05;
}

// ************************************************************************* //
```



Wskazanie, która oś jest osią pochylenia.

Zadanie 3 – monitorowanie sił

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

forceCoeffs1
{
    type            forceCoeffs;

    libs            ("libforces.so");

    writeControl    timeStep;
    timeInterval    1;

    log             yes;

    patches         (cylinder);
    rho             rhoInf;
    rhoInf          998.204;
    liftDir         (0 1 0);
    dragDir         (1 0 0);
    CofR            (0 0 0);
    pitchAxis       (0 0 1);
    magUInf         0.0245;
    lRef            8E-05;
    Aref            8E-05;
}

// ************************************************************************* //
```

Wartość prędkości w przepływie niezaburzonym – tu jest to prędkość wlotowa.

Zadanie 3 – monitorowanie sił

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  dev
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/

forceCoeffs1
{
    type            forceCoeffs;

    libs            ("libforces.so");

    writeControl    timeStep;
    timeInterval    1;

    log             yes;

    patches         (cylinder);
    rho             rhoInf;
    rhoInf          998.204;
    liftDir         (0 1 0);
    dragDir         (1 0 0);
    CofR            (0 0 0);
    pitchAxis       (0 0 1);
    magUInf         0.0245;
    lRef            8E-05;
    Aref            8E-05;
}

// ************************************************************************* //
```

Pole powierzchni obiektu prostopadłe do kierunku wyznaczania danej siły – tu: oba pola są jednakowe.

Podczas obliczeń należy wziąć pod uwagę rzeczywistą szerokość komórki w kierunku osi Z.

Zadanie 3 – monitorowanie sił

```
application     icoFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         25;

deltaT          0.005;

writeControl    timeStep;

writeInterval   100;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

functions
{
    #includeFunc residuals
    #include "forceCoeffs"
}
```

Aby monitor sił zadziałał, należy w słowniku **controlDict** dodać na końcu następujący wpis:

```
functions
{
    #include "forceCoeffs"
}
```

Podczas działania programu powstanie nowy plik (w podkatalogu **/postProcessing/forceCoeffs1/0**) o nazwie **forceCoeffs.dat**, w którym wartości wszystkich parametrów zapisywane są oddzielnie dla każdego kroku czasowego – dane to można później łatwo wizualizować np. za pomocą programu Gnuplot.

Zadanie 3 – monitorowanie rezyduów

```
application     icoFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         25;

deltaT          0.005;

writeControl    timeStep;

writeInterval   100;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

functions
{
    #includeFunc residuals
    #include "forceCoeffs"
}
```

W przykładzie w sekcji **functions** dodano jeszcze jeden wpis. Umożliwi on wizualizację rezyduów w trakcie obliczeń (lub później). Dane do wykresu zapisywane są w podkatalogu **postProcessing/residuals/0** w pliku **residuals.dat**.

Aby monitorować wartość rezyduów na bieżąco, należy uruchomić program **icoFoam** (lub inny dowolny solwer) i przekierować wyjście z terminala do pliku (loga):

icoFoam > log &

a następnie uruchomić polecenie monitorowania:

foamMonitor -l -r 1 postProcessing/residuals/0/residuals.dat

Zadanie 3 – monitorowanie parametrów w punkcie

```
functions
{
    #includeFunc residuals
    #include "forceCoeffs"

    probes1
    {
        type probes;
        functionObjectLibs ("libsampling.so");
        setFormat gnuplot;
        surfaceFormat raw;

        probeLocations
        (
            (0.05 0.005 0.0)
        );

        fields
        (
            p
            U
        );
    }
);
```

W przykładzie dodano jeszcze słownik służący do zapisywana wartości wybranych pól w określonym punkcie przestrzeni.

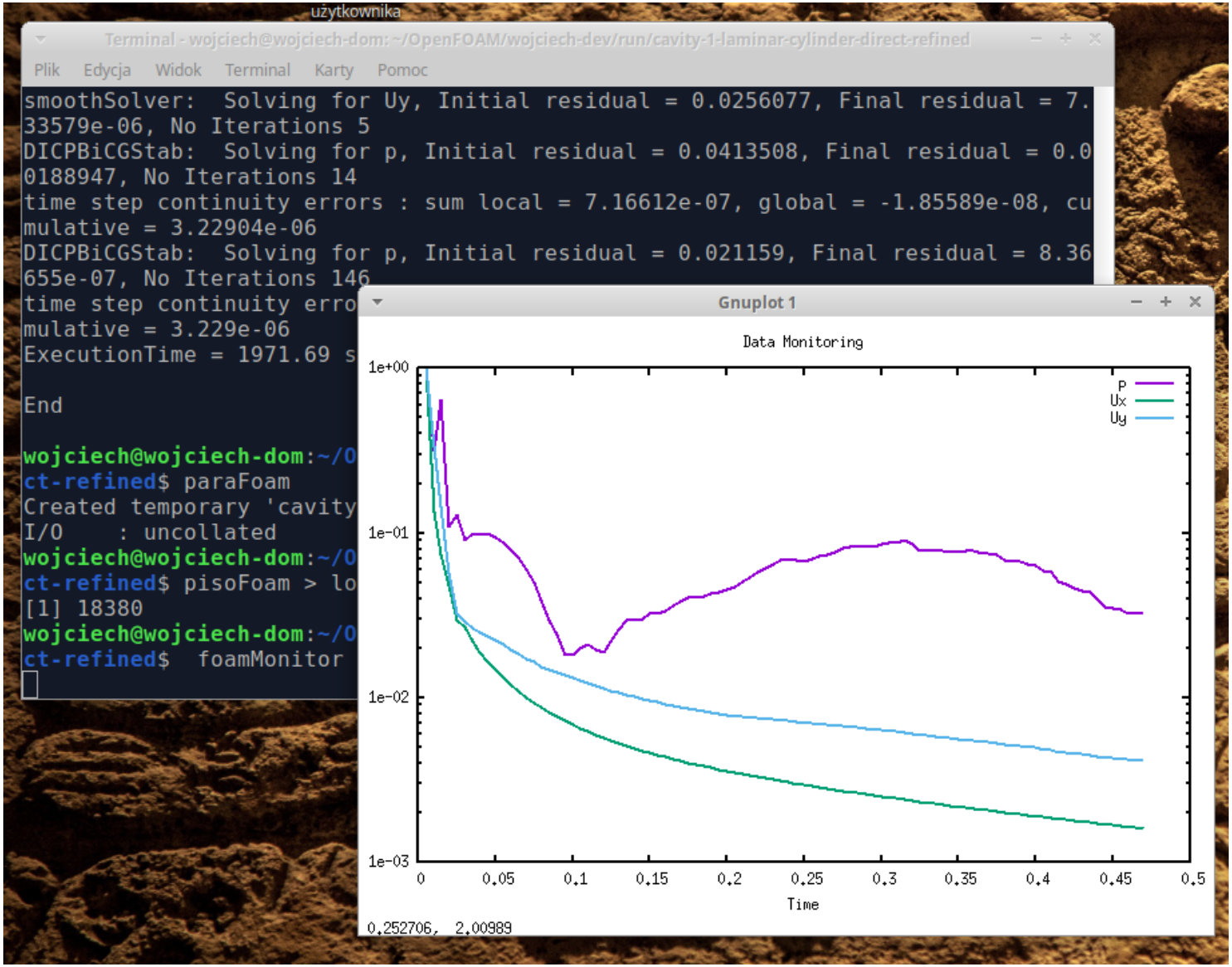
Położenie punktu określa sekcja **probeLocations**.

Nazwy pól określa sekcja **fields**.

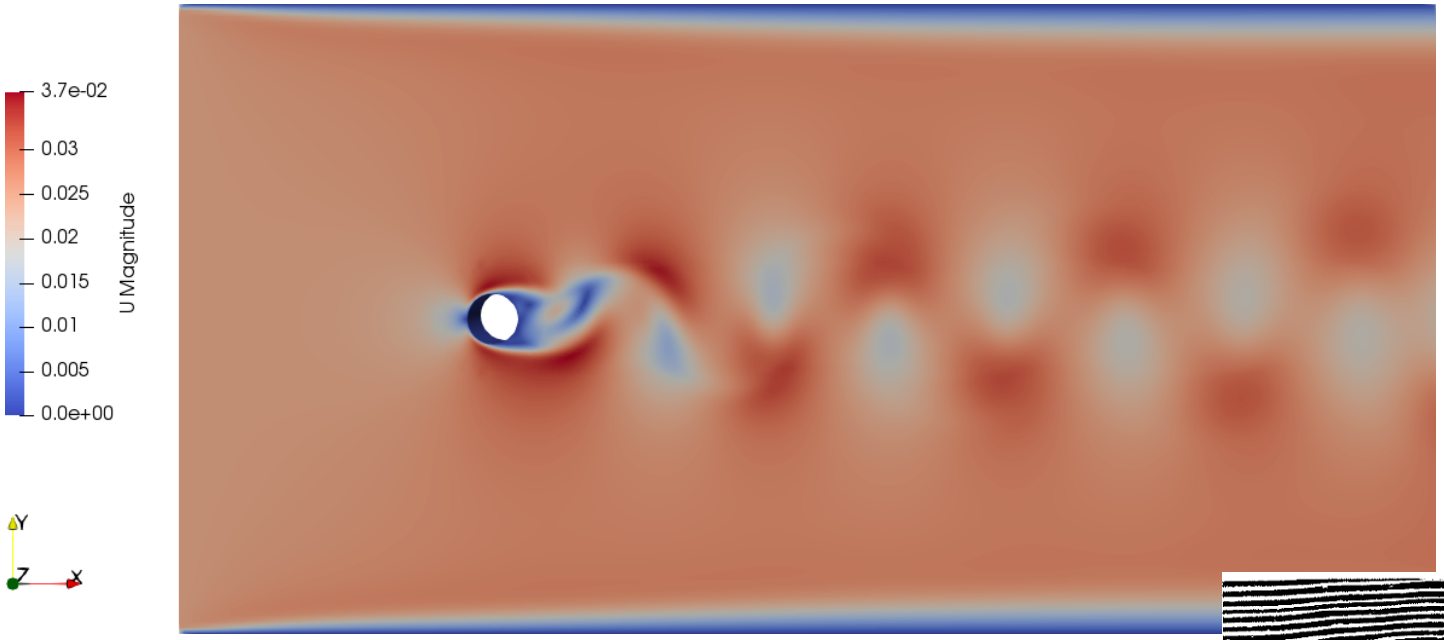
Wyniki zapisywane są w pliku **postProcessing/residuals/0** w pliku **probes1.dat**.

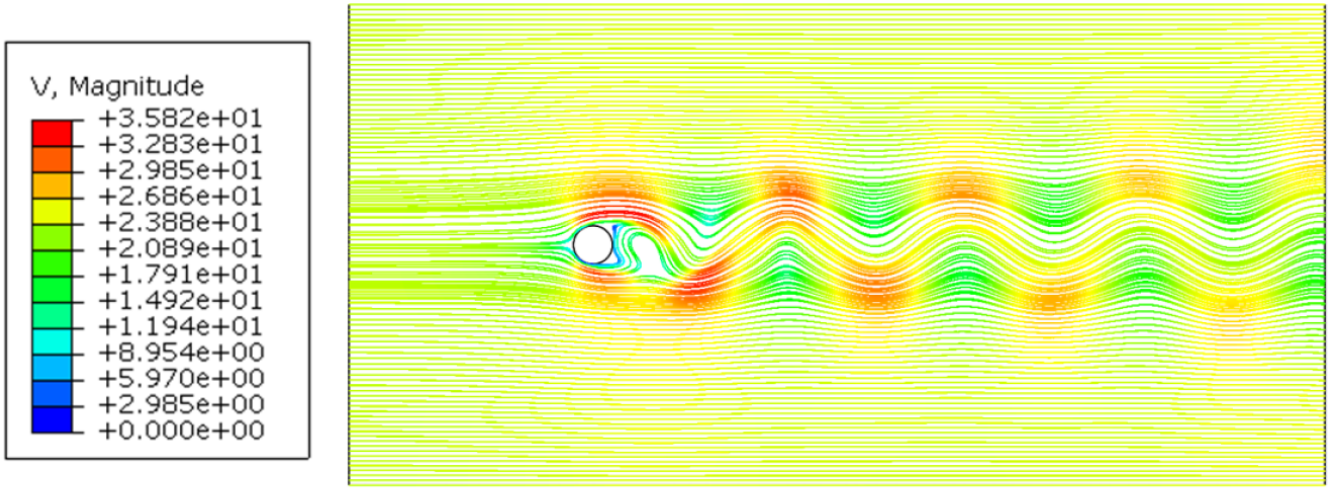
Zadanie 3 – wyniki

Wizualizacja rezyduów w trakcie obliczeń.

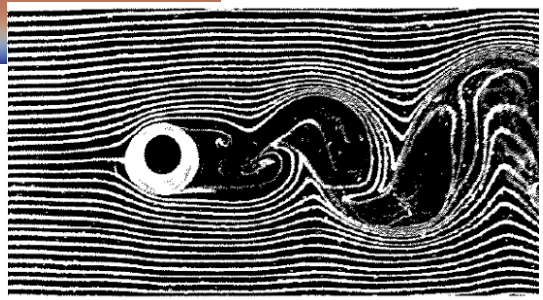


Zadanie 3 – wyniki



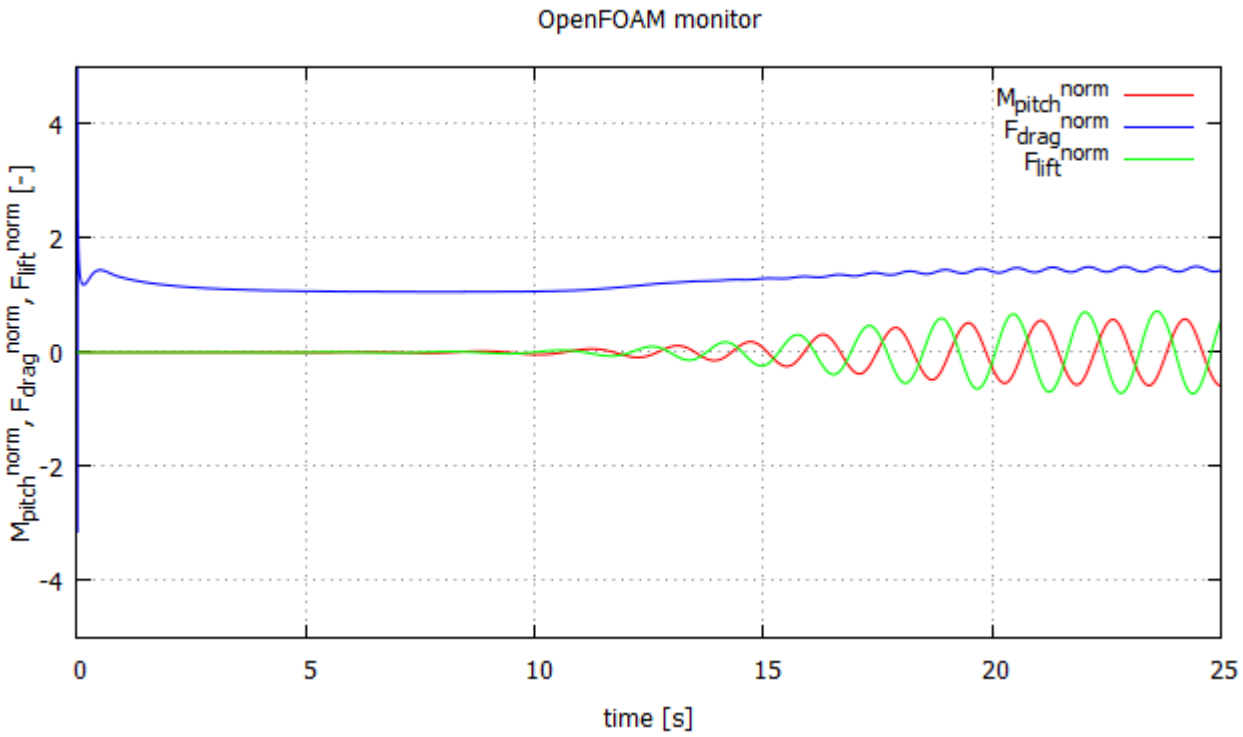


(h) CASE8 *Re* = 195



Porównanie pola prędkości z wynikami przedstawionymi w artykule Sato i Kobayashi.

Zadanie 3 – wyniki



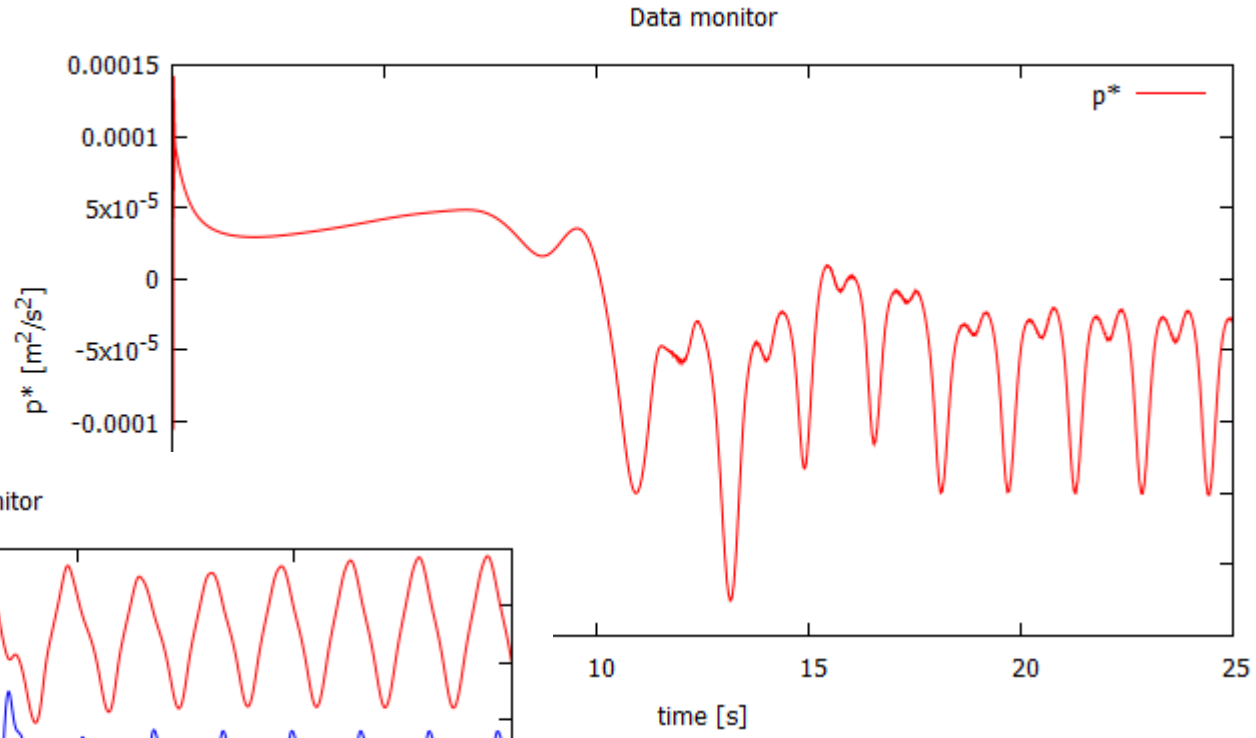
Wyniki działania monitora siły unoszenia i siły oporu.

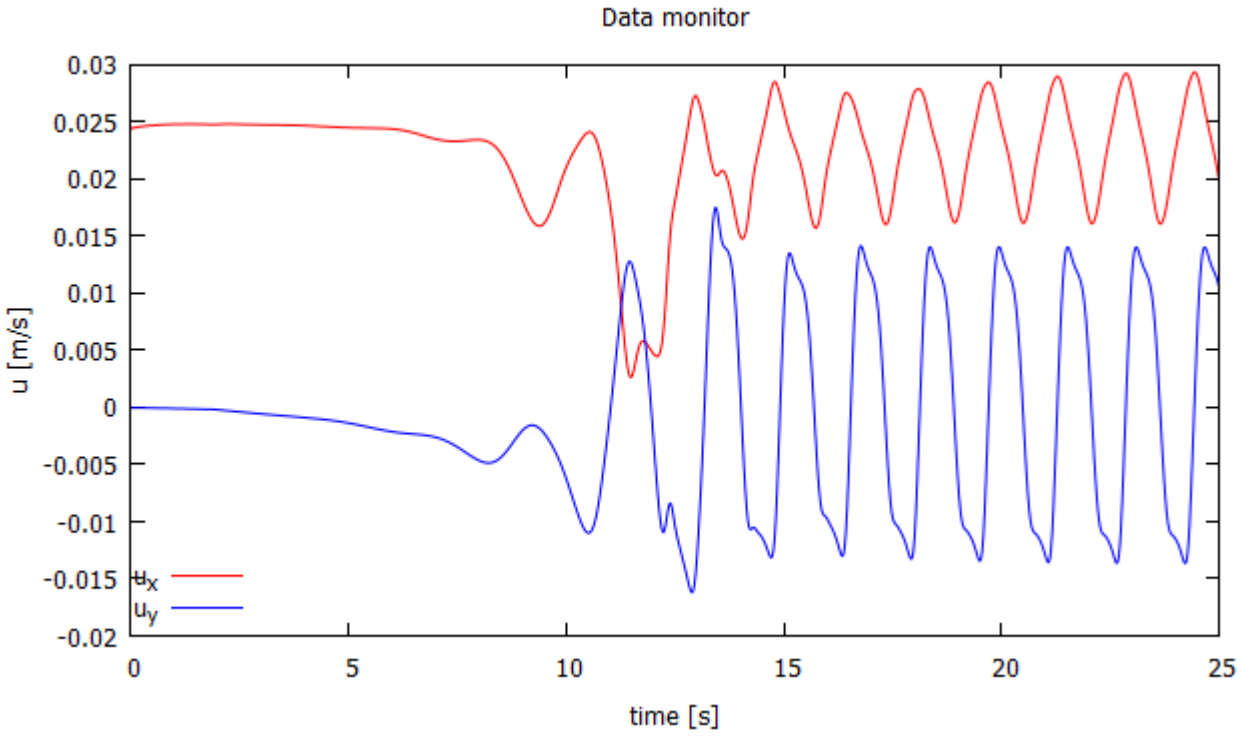
```
set title 'OpenFOAM monitor'
set xlabel 'time [s]'
set ylabel 'M_{pitch}^{norm}, F_{drag}^{norm}, F_{lift}^{norm} [-]'
set grid
set yrange [-5:5]

plot 'forceCoeffs.dat' u 1:2 title 'M_{pitch}^{norm}' with lines lt 1 lc 'red',\
     'forceCoeffs.dat' u 1:3 title 'F_{drag}^{norm}' with lines lt 1 lc 'blue',\
     'forceCoeffs.dat' u 1:4 title 'F_{lift}^{norm}' with lines lt 1 lc 'green'

pause mouse
```

Skrypt Gnuplota, za pomocą którego wykonano wykres.

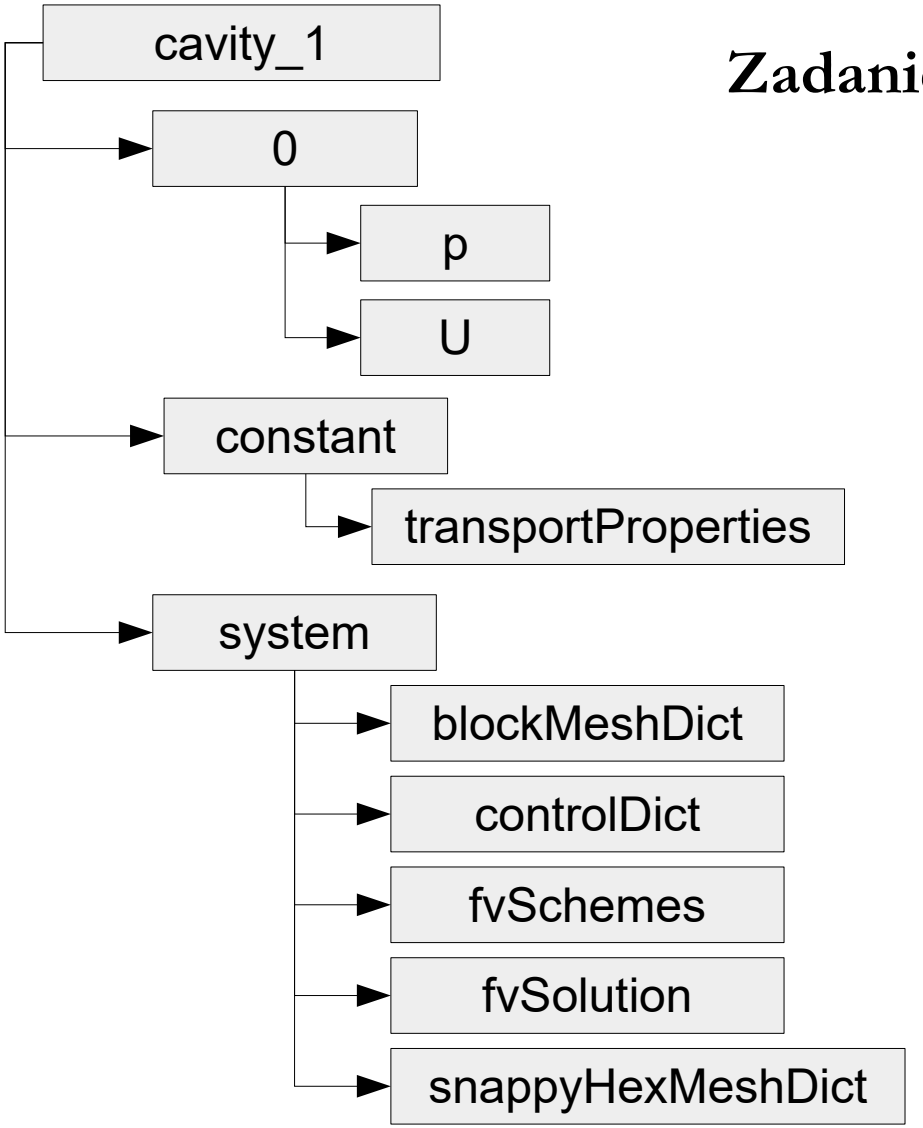
Zadanie 3 – wyniki

Wyniki działania monitora ciśnienia (podzielonego przez gęstość) oraz prędkości.





Zadanie 4



Zadanie 4 – zmienić solwer na **pisoFoam**.

Aby wykonać to zadanie należy znaleźć w przykładach pakietu OpenFOAM, jakiś model oparty na solwerze **pisoFoam**, np.

tutorials/incompressible/pisoFoam/laminar/porousBlockage

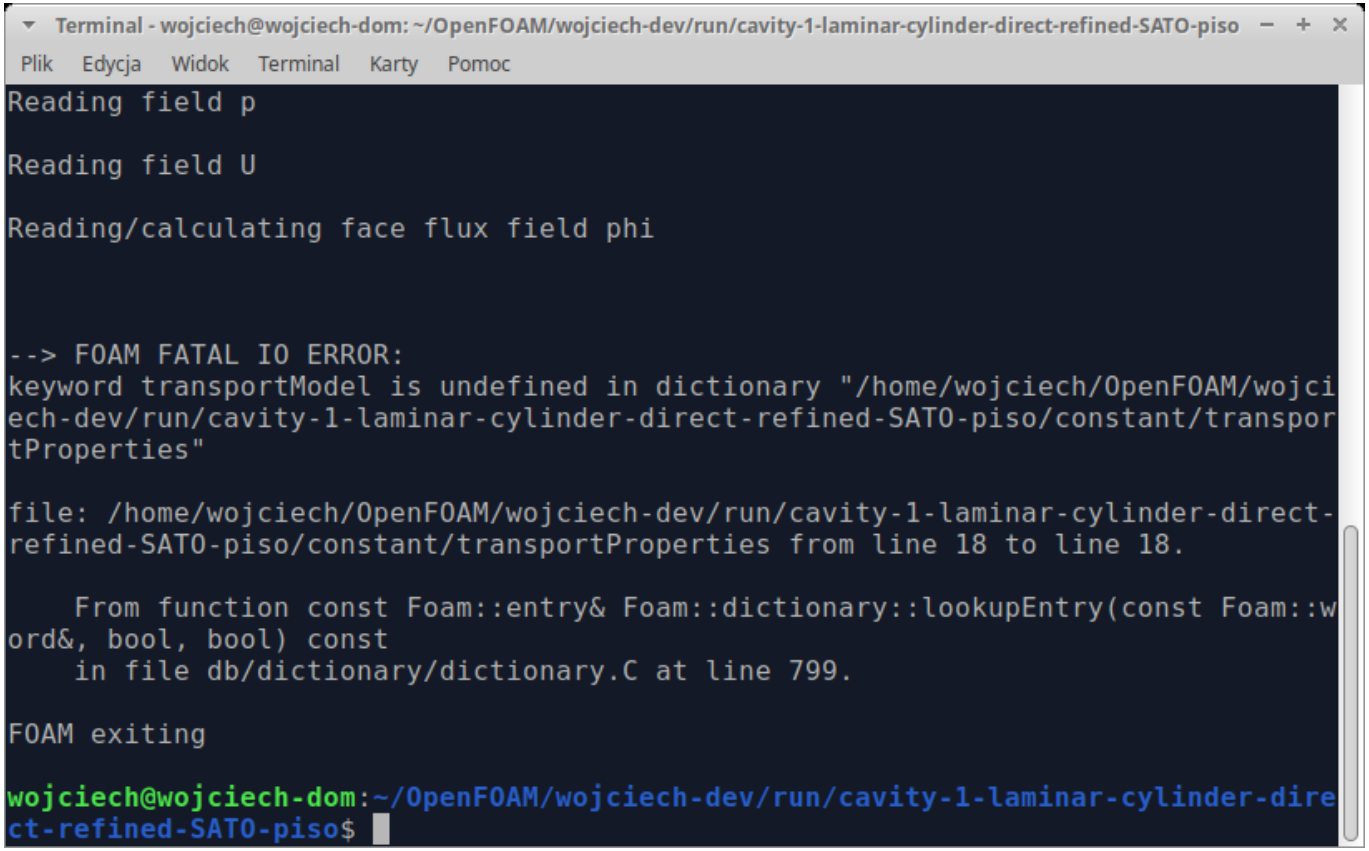
Tu postąpimy jednak nieco inaczej – zmienimy nazwę solwera i zobaczymy co się stanie (w razie błędów będziemy zaglądać do przykładu **porousBlockage**).

Zadanie 4

```
application     pisoFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         25;

deltaT          0.005;

writeControl    timeStep;

writeInterval   100;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

functions
{
    #includeFunc residuals
    #include "forceCoeffs"
}
```

Zmieniamy solwer na pisoFoam i uruchamiamy obliczenia poleceniem **pisoFoam** (na razie nie włączamy śledzenia rezyduów, bo spodziewamy się błędów).

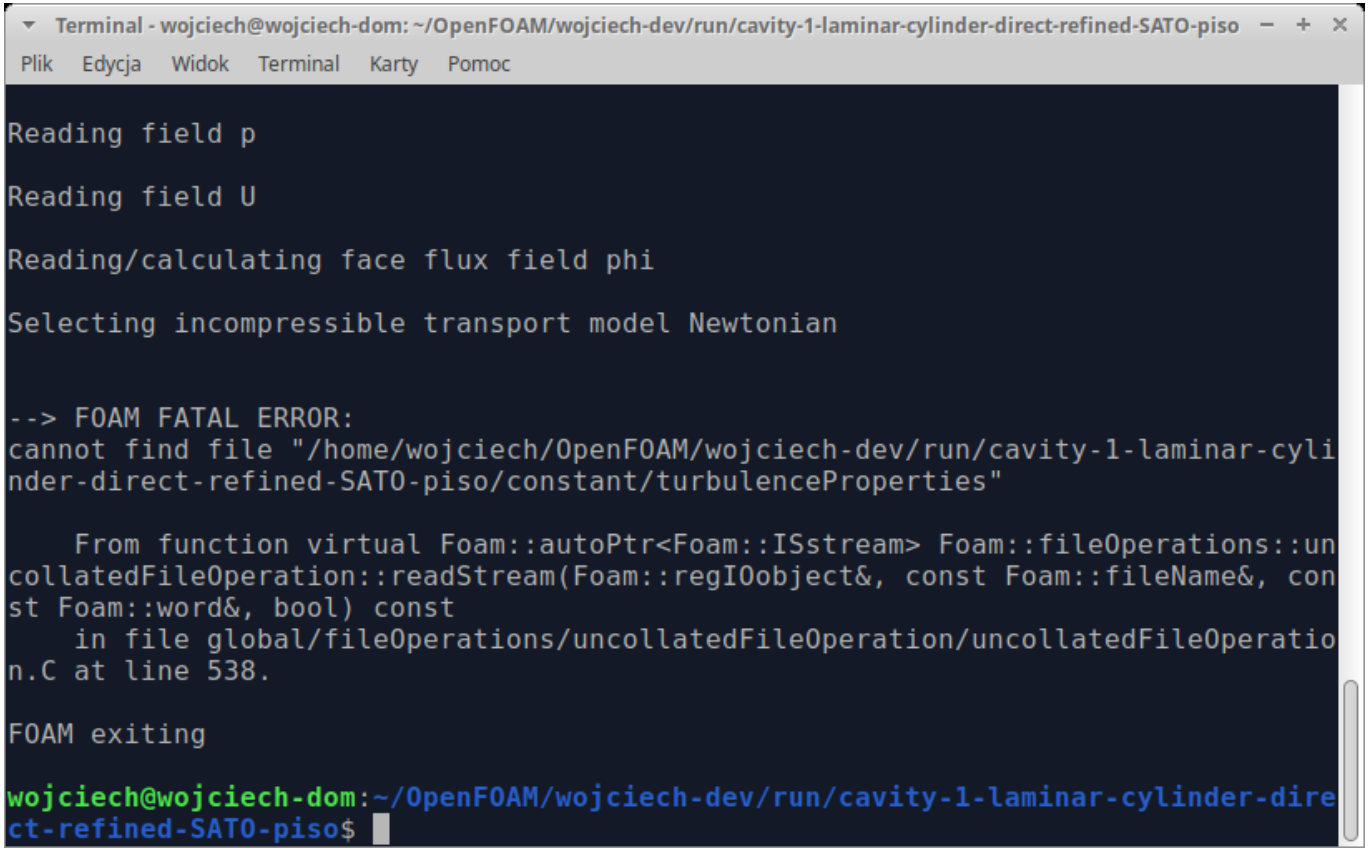
Zadanie 4 – zmiana solwera



Błąd 1 – w słowniku **transportProperties** brakuje wpisu **transportModel**. Wpis kopiujemy z tego samego słownika z przykładu **porousBlockage**:

transportModel Newtonian;

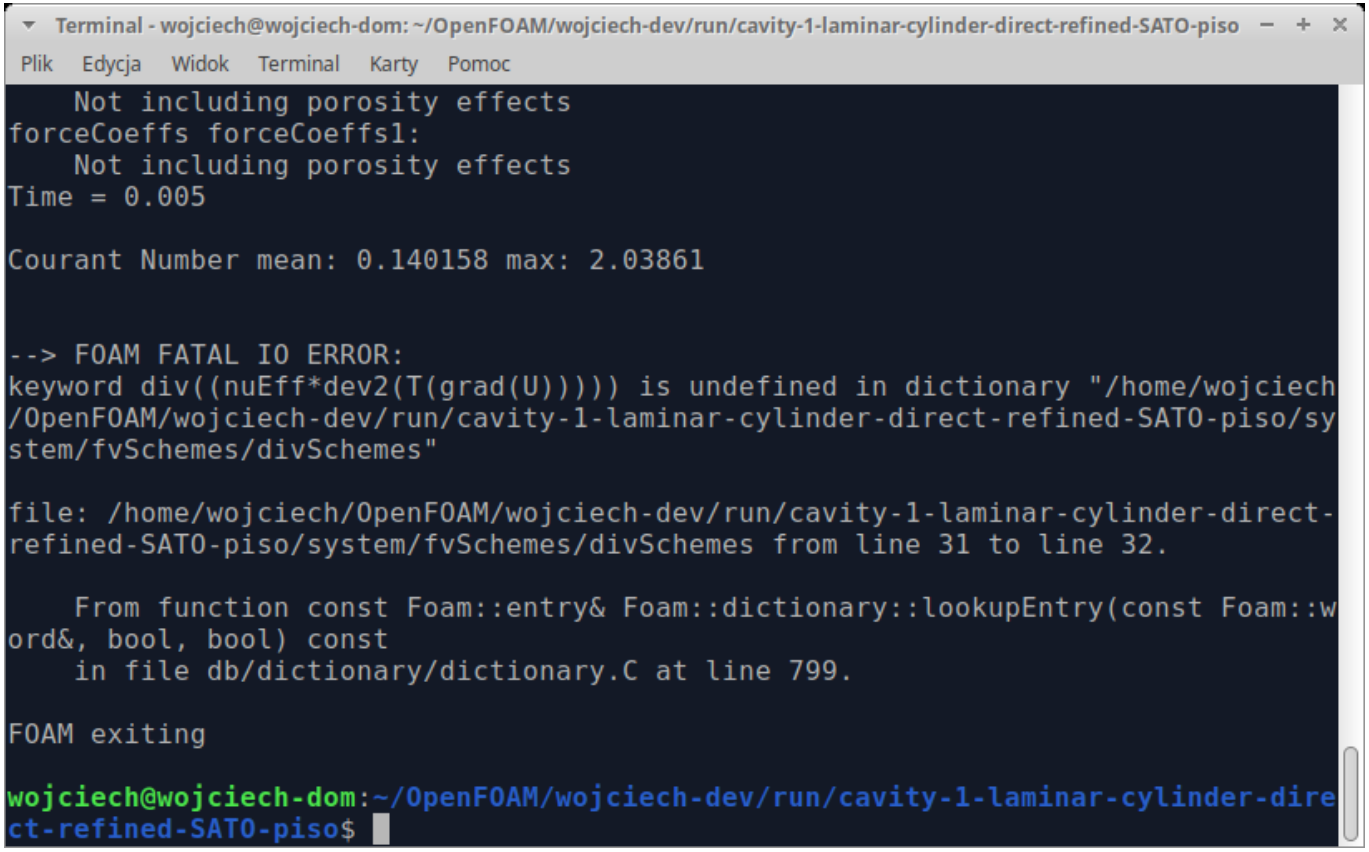
Zadanie 4 – zmiana solwera



Błąd 2 – w katalogu **constant** brakuje słownika **turbulenceProperties**. Kopiujemy go z przykładu **porousBlockage**. W słowniku jest tylko jeden wpis informujący, że przepływ jest laminarny:

simulationType laminar;

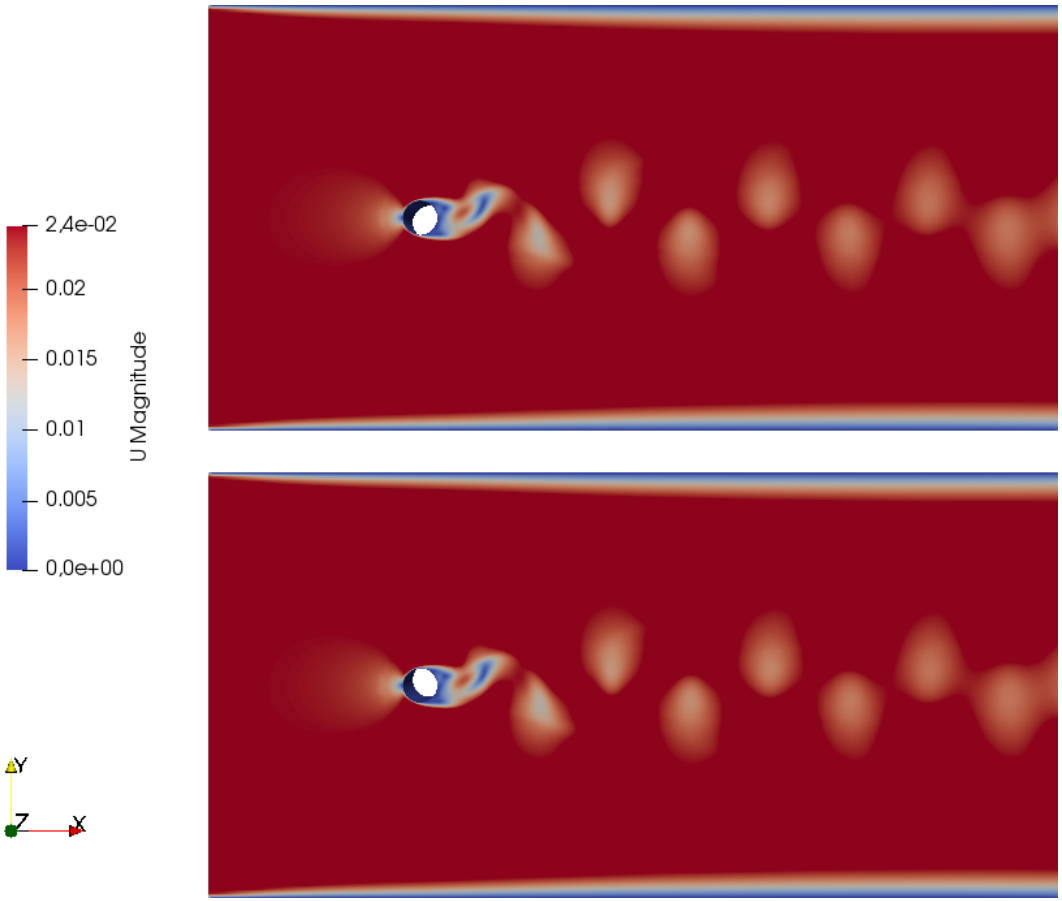
Zadanie 4 – zmiana solwera



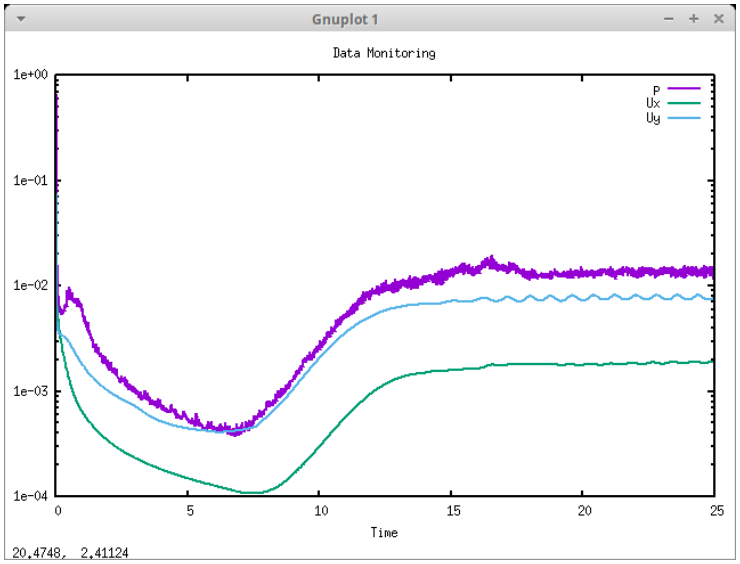
Błąd 3 – w słowniku **fvSchemes**, w sekcji **divSchemes** brakuje wpisu dla członu **div((nuEff*dev2(T(grad(U)))))**. Zaglądamy do przykładu **porousBlockage** i kopiujemy odpowiedni wpis:

div((nuEff*dev2(T(grad(U))))) Gauss linear;

Zadanie 4 – wyniki



Solwer **icoFoam**.



Solwer **pisoFoam** (u góry wykres rezyduów).

Po tych trzech zmianach solwer działa i uzyskuje się wyniki dokładnie takie same jak poprzednio.

Wydział Nauk Technicznych

UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN
The Faculty of Technical Sciences
POLAND, 10-957 Olsztyn, M. Oczapowskiego 11
tel.: (48)(89) 5-23-32-40, fax: (48)(89) 5-23-32-55
URL: http://www.uwm.edu.pl/edu/sobieski/ (in Polish)

Dziękuję za uwagę

Publikacja została napisana w wyniku odbywania przez autora stażu w Holandii, współfinansowanego przez Unię Europejską w ramach Europejskiego Funduszu Społecznego (Program Operacyjnego Wiedza Edukacja Rozwój), zrealizowanego w projekcie Program Rozwojowy Uniwersytetu Warmińsko-Mazurskiego w Olsztynie (POWR.03.05.00-00-Z310/17).

Wojciech Sobieski

Utrecht, 2019