Modèles de turbulence

TP OpenFOAM

Encadrant : Paola CINNELLA

Écoulement turbulent décollé : marche descendante

Etudiants :	*Numéros étudiants :*
Alexandre Richard	3700491
Valentin Duvivier	3700091

25 février 2022

Table des matières

1 Introduction

Durant ce travail pratique on s'intéresse à l'étude d'un écoulement dans un canal avec un élargissement brusque de section, de type « marche descendante ». L'écoulement incident est caractérisé par un nombre de Mach Ma=0.2 et un nombre de Reynolds $Re = 5.e^6$, soit un écoulement incompressible turbulent, on retrouve ce problème sur le site de la NASA [1].

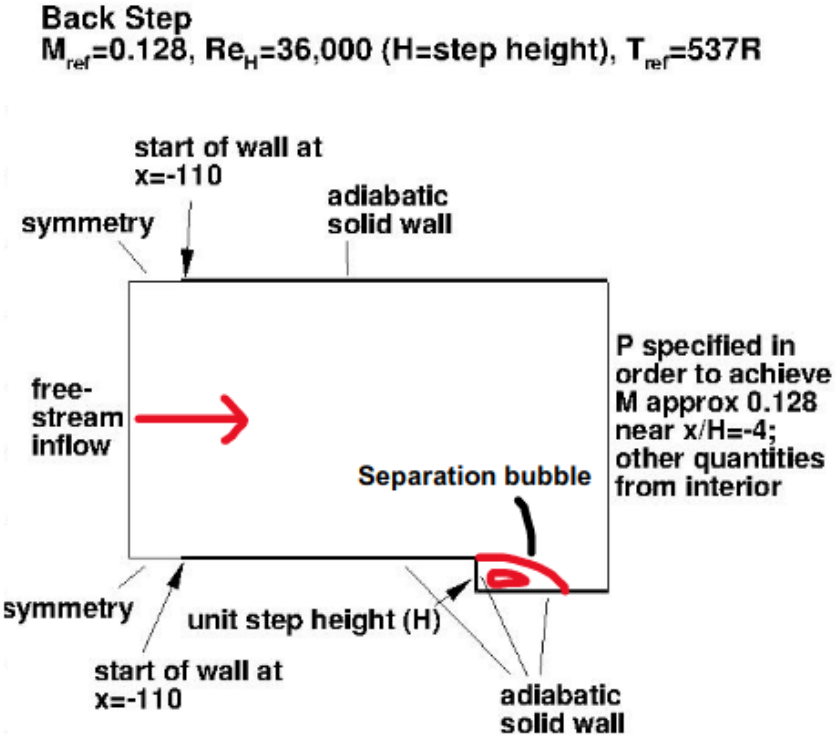


FIGURE 1 – Plaque plane - partie supérieure

Ce système est un cas de "décollement" dû à une variation brusque de géométrie ; à travers lequel on va étudier l'aspect général de différents modèles RANS. On va ainsi caractériser le coefficient de frottement $C_f = \frac{\tau_{wall}}{\frac{1}{2}\rho U_{ref}^2}$ ainsi que les profils de vitesse pour différentes distances à la marche. L'idée sera alors de voir les points forts des différents modèles ainsi que leurs différentes faiblesses.

Ci-dessous, on décrit brièvement chaque modèle utilisés durant le TP, à partir du cours du Professeur assistant L. Sciacovelli [2].

1.1 Le modèle komegaSST

Ce modèle est un modèle algébrique à 2 équations. Il est construit à partir des modèles $k_t - \epsilon$ et $k_t - \omega$:

$$\begin{cases} \dfrac{\delta k_t}{\delta t} + \dfrac{\delta \tilde{u_j} k_t}{\delta x_j} = \dfrac{1}{\bar{\rho}} \tau_{ij}^R \dfrac{\delta \tilde{u_i}}{\delta x_j} - \beta^* k\omega + \dfrac{\delta}{\delta x_j}\left[(\bar{\nu} + \sigma_k \nu_t)\dfrac{\delta k}{\delta x_j}\right] \\ \dfrac{\delta \omega}{\delta t} + \dfrac{\delta \tilde{u_j} \omega}{\delta x_j} = \dfrac{\alpha}{\rho \nu_t} \tau_{ij}^R \dfrac{\delta \tilde{u_i}}{\delta x_j} - \beta \omega^2 + \dfrac{\delta}{\delta x_j}\left[(\bar{\nu} + \sigma_\omega \nu_t)\dfrac{\delta \omega}{\delta x_j}\right] + 2(1 - F_1)\dfrac{\sigma_{\omega_2}}{\omega}\dfrac{\delta k_t}{\delta x_j}\dfrac{\delta \omega}{\delta x_j} \end{cases} \tag{1}$$

Avec ω le taux de de dissipation spécifique défini comme :

$$\omega = \frac{\epsilon}{C_{mu} k_t} \tag{2}$$

Ce modèle est dirigé par F_1 :

$$F_1 = \begin{cases} 1 & \text{(Région proche de la paroi) On retrouve alors le comportement du modèle } k_t\omega \\ 0 & \text{(Loin de la paroi) On retrouve alors le comportement du modèle } k_t - \epsilon \end{cases} \tag{3}$$

Ce modèle a donc les avantages des deux modèles. Il a un comportement correct dans la couche visqueuse et une indépendance aux valeurs au loin.

1.2 Le modèle kEspilonSST

Le modèle $k_t\epsilon$, là aussi un modèle à 2 équations de transport, est basé sur l'énergie cinétique turbulente et sur le taux de dissipation :

$$\begin{cases} \dfrac{\delta k_t}{\delta t} + \dfrac{\delta \tilde{u_j} k_t}{\delta x_j} = \dfrac{1}{\bar{\rho}} \tau_{ij}^R \dfrac{\delta \tilde{u_i}}{\delta x_j} - \epsilon + \dfrac{\delta}{\delta x_j}\left[\left(\bar{\nu} + \dfrac{\nu_t}{\sigma_k}\right)\dfrac{\delta k}{\delta x_j}\right] \\ \dfrac{\delta \epsilon}{\delta t} + \dfrac{\delta \tilde{u_j} \epsilon}{\delta x_j} = C_{\epsilon_1} \dfrac{\epsilon}{\bar{\rho}.k_t} \tau_{ij}^R \dfrac{\delta \tilde{u_i}}{\delta x_j} - C_{\epsilon_2} \dfrac{\epsilon^2}{k_t} + \dfrac{\delta}{\delta x_j}\left[\left(\bar{\nu} + \dfrac{\nu_t}{\sigma_\epsilon}\right)\dfrac{\delta \epsilon}{\delta x_j}\right] \end{cases} \tag{4}$$

Ce modèle possède une indépendance aux valeurs du flux libre et reproduit correctement la contrainte de cisaillement dans les écoulements à cisaillement libre.

1.3 Le modèle kEpsilonRNG

Ce modèle est une variante du modèle k-epsilon standard. Il consiste en une estimation au cours du calcul de la constante $C_{\epsilon 1}$, remplacée dans l'équation de dissipation par $C'_{\epsilon 1}$:

$$C'_{\epsilon 1} = C_{\epsilon 1} - \frac{\eta(1 - \frac{\eta}{\eta_0})}{1 + \beta\eta^3} \tag{5}$$

Avec $\eta = \frac{k}{\epsilon}\frac{p}{\eta_t}$.

Cette expression ajoute un terme fonction du taux de déformation η à l'équation du taux de dissipation, le rendant ainsi moins diffusif.

La différence principale entre la version standard et RNG est donc dans l'équation du taux de la dissipation turbulente d'énergie. Dans les écoulements à taux de contraintes élevés, le modèle RNG prévoit une plus faible viscosité turbulente (c'est-à-dire un taux de dissipation ϵ élevé et une production de turbulence k faible) que le modèle standard. Cette modification permet donc de corriger le modèle standard pour les écoulements avec une grande courbure des lignes de courant.

1.4 Le modèle LRR

Le modèle LRR résout directement les contraintes de Reynolds $\bar{\rho}\hat{R}_{ij}$, que l'on peut réécrire :

$$\bar{\rho}\hat{R}_{ij} = -\tau_{ij} = \overline{\rho u_i" u_j"} \tag{6}$$

avec la notation en moyenne de Favre. On obtient alors une équation de transport :

$$\frac{\delta\bar{\rho}\hat{R}_{ij}}{\delta t} + \frac{\bar{\rho}\hat{u}_k\hat{R}_{ij}}{\delta x_k} = \bar{\rho}P_{ij} + \bar{\rho}\Pi_{ij} - \bar{\rho}\epsilon_{ij} + \bar{\rho}D_{ij} + \bar{\rho}M_{ij} \tag{7}$$

Avec $\bar{\rho}M_{ij}$ le flux massique fluctuant qui est en fait négligé, $\bar{\rho}D_{ij}$ le terme de diffusion, $\bar{\rho}\epsilon_{ij}$ la dissipation, $\bar{\rho}\Pi_{ij}$ la corrélation pression-déformation et $\bar{\rho}P_{ij}$ le terme de production.

Ce modèle viendra conclure sur notre étude et sur l'utilisation de termes correctifs au profit de la robustesse de modèles.

2 Résultats numériques

2.1 Activité 1

La majorité des dictionnaires utilisé durant ce TP sont les mêmes que ce du *"TP1-Flat-plate"*, hormis ***Allrun*** qui ici permet la superpositions de courbes et modèles.

2.1.1 Etude de la structure des dictionnaires : 0, constant et system

Dans le dictionnaire 0, on retrouve des fonctions pour la viscosité turbulente ν_t, l'énergie cinétique turbulente k, la dissipation spécifique ω et la dissipation ϵ.

1. ν_t : ν_t en entrée, en sortie et en haut du domaine est fixé. Au niveau de la **plaque**, ν_t est calculé à partir d'une **fonction de proche paroi**.
2. ω : ω en entrée est fixé. A la **paroi**, ω est calculé à partir d'une **fonction de proche paroi** et une **correction** de **blended**, un traitement automatique pour rendre les résultats insensibles au raffinement du maillage des murs. En sortie et en haut de la plaque on a une condition de zéro gradient.
3. ϵ : ϵ en entrée est fixée à partir de la valeur calculée avec le modèle $k-\omega$: $\epsilon = \omega.k.C_\mu$. A la **plaque**, ϵ est calculé à partir d'une **fonction de proche paroi** et une **correction bas Reynolds** est activée. En sortie et en haut du domaine on a une condition de **zéro gradient** (i.e. valeur constante).
4. k : k en entrée est fixée. Au niveau de la **plaque**, k est calculé à partir d'une **fonction de proche paroi**. Aux autres faces du domaine on impose une condition de **zéro gradient**.

Dans le dictionnaire constant, on retrouve les propriétés de turbulence :

1. turbulenceProperties : C'est dans cette fonction que l'on définit le modèle de turbulence pour la simulation, le RAS "Reynolds Averaged Simulation" on peut également définir ou non si l'on traite la turbulence.

Dans le dictionnaire constant, on retrouve les fonctions liées à l'exploitation des résultats controlDict, la discrétisation des équations fvSchemes et la méthode de résolution fvSolution :

1. **controlDict** : dans cette fonction on choisit le **pas de temps** de la résolution ainsi que le **nombre d'itérations** ;
2. **fvSchemes** : dans cette fonction on discrétise les équations au travers des **opérateurs** gradient, divergence, Laplacien, etc. On caractérise par ailleurs la stationnarité de l'écoulement ;

3. **fvSolution** : contient les facteurs de relaxation mais surtout la méthode de résolution numérique et spécifie chaque solveur linéaire utilisé pour chaque équation discrétisée

2.1.2 Étude de la structure du script ALLRUN

Le script Allrun exécute blockMeshDict puis SimpleFoam, il affiche ensuite le coefficient de frottement pariétal en fonction de la position x normalisée avec la hauteur h de la marche : x/h. Il permet également de tracer les profils de vitesse à différentes positions x normalisées avec la hauteur h de la marche x/h.

Ce procédé d'automatisation nous permet une étude plus fluide et directe des modèles, et notamment de les comparer plus efficacement.

A nos courbes s'ajoute les données expérimentales de [3] qui servent de référence pour critiquer les résultats numériques des modèles utilisés. Pour finir, nous avons également ajouté une fonction traçant y+ en fonction de la position x/h.

2.2 Activité 2

On s'intéresse dans un premier temps au modèle komegaSST, et on chercher à en déduire certains comportements généraux et d'autres plus spécifiques à ce modèle.

2.2.1 Sensibilité de la solution au nombre d'itérations utilisées

Tout d'abord, on s'intéresse à caractériser la dépendance entre précision des résultats et nombre d'itérations pour converger. Contrairement au TP1, on cherche ici le nombre précis d'itérations. La précision des résultats est obtenue par comparaison avec la littérature [3].

Le tracé et la superposition des courbes ci-dessus mentionné amène que pour un temps de calcul de 1000 on a un résultat visuellement équivalent que pour un temps de calcul de 2000 ou de 4000. Par conséquent pour la suite on pourra se permettre d'utiliser un temps de calcul de 1000 afin d'assurer une bonne précision sur nos résultats tout en gardant un calcul rapide.

Pour le modèle kOmegaSST, il semble cependant que le taux imposé pour la convergence ne permette pas de converger vers une solution exacte (testé jusqu'à 10000 itérations). Ce point sera comparé avec les autres modèles pour voir si ces derniers convergent.

2.2.2 Distribution de yPlus le long de la paroi inférieure

Dans cette section on trace la longueur de première maille en fonction de la position le long de l'écoulement, afin de voir comment se comporte les données du modèle au cours de l'écoulement :

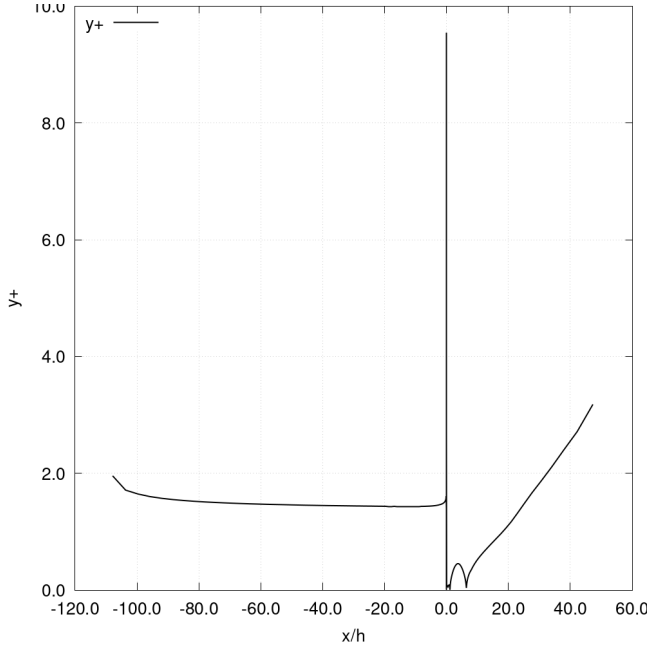


FIGURE 2 – yPlus pour le modèle kOmegaSST

On observe un comportement uniforme du modèle en amont du saut. On suit alors la forme de la marche, modélisant donc l'écoulement à la paroi ; puis un comportement en 2 phases :

— Au niveau du saut (i.e. $x/h = 0$), il y a un saut dans la courbe, qui est potentiellement dû à une erreur numérique car pour calculer y+ openFoam utilise la normale, mais ici puisque l'on a une marche, la normale n'est pas bien définie.

— Dans une seconde partie, on ne suit plus la topologie mais on augmente progressivement yPlus en aval du saut, afin de **simuler le recollement**.

Voyons comment se comporte le modèle comparé à la littérature [3].

2.2.3 Superposition des données de Driver&Seegmiller pour le Cf avec la courbe issue de la simulation

Dans la figure ci-dessous on regarde le comportement du modèle au niveau du saut, en le comparant notamment à une référence empirique [3] :

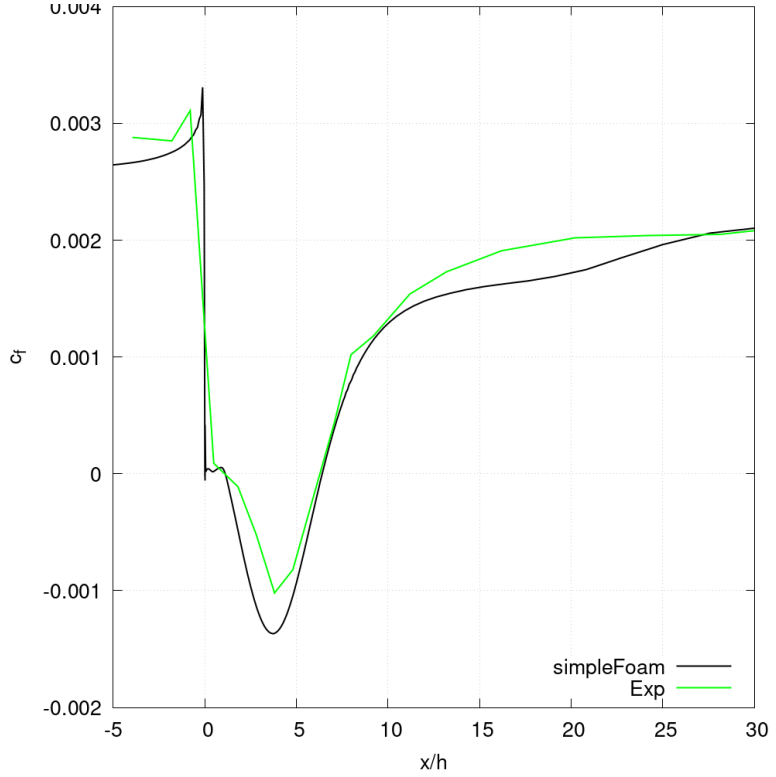


FIGURE 3 – Comparaison du Cf expérimentale et numérique

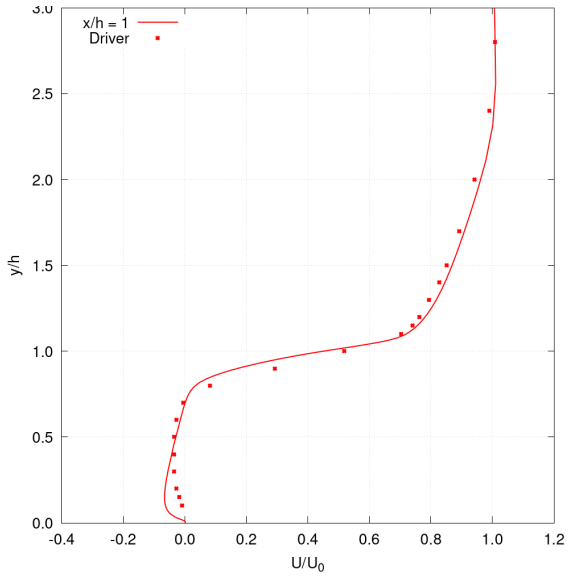
On voit que le Cf donné par le modèle kOmegaSST colle bien aux résultats expérimentaux en amont, en aval et surtout au niveau du saut, traduisant une **bonne adaptation du modèle à la géométrie**. En sommes, on **décroît** avec un taux **consistent** par rapport à la théorie. Cependant, l'amplitude n'est pas entièrement respecté, le **modèle sous-estimant légèrement** la valeur de Cf après le saut.

Enfin, le comportement global du modèle semble acceptable, excepté loin du saut où il y a une surestimation du taux de frottement Cf. Ainsi, le modèle reproduit bien les taux de croissances ou décroissance mais à tendance à sous ou sur évaluer les quantités à la paroi. Voyons si ce comportement est à associé au modèle, ou bien si l'étude d'une autre quantité avec le même modèle pourrait amener des conclusions différentes.

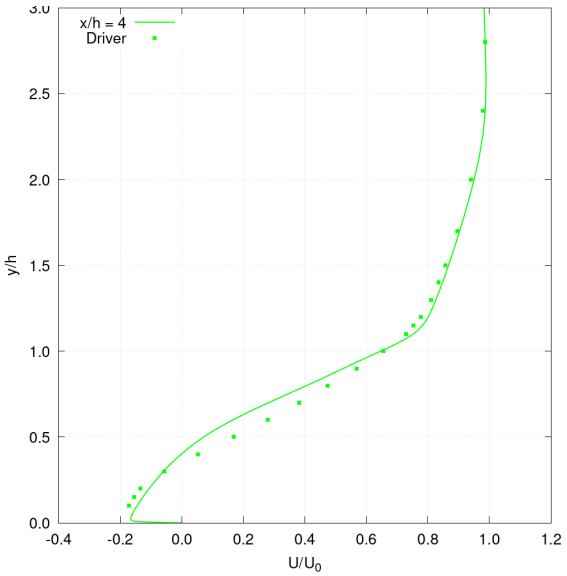
2.2.4 Profils de vitesse longitudinale

Dans cette section, on cherche à caractériser le comportement du modèle en fonction de la position dans l'écoulement. Pour cela on s'intéresse aux profils de vitesse en 4 points de l'écoulement :

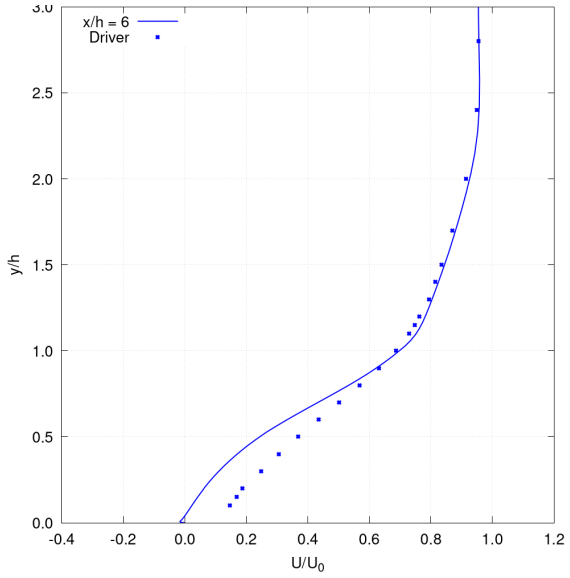
(a) x/h=1



(b) x/h=4



(c) x/h=6



(d) x/h=10

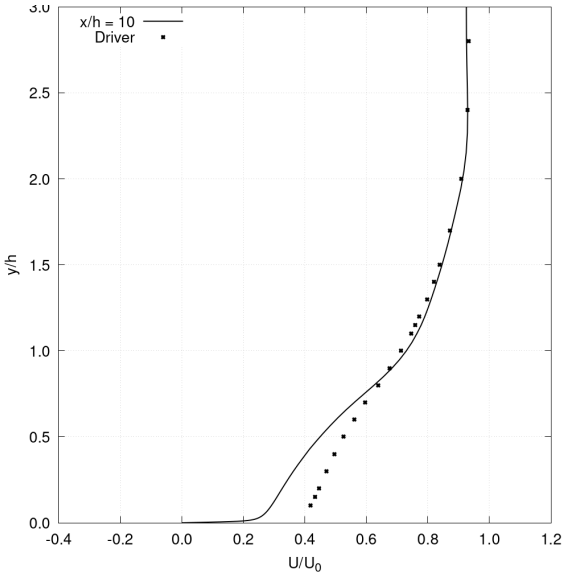


FIGURE 4 – Profils de vitesse - kOmegaSST

On peut remarquer que les profils de vitesse donnés par le modèle kOmegaSST collent globalement bien aux résultats expérimentaux de Driver & Seegmiller.

En détail :

— pour $y/h > 0.5$ les résultats numériques suivent très correctement les valeurs théoriques ainsi que les différents changement de profil (i.e. pente) ;

— pour $y/h < 0.5$, soit au niveau de la paroi, la condition d'adhérence ne permet pas au modèle de suivre la bonne pente. Il en ressort que le modèle à tendance à ne pas correctement reproduire les premières valeurs de vitesses.

Pour finir, le comportement à l'infini est très bien reproduit tandis que celui en proche paroi manque de précision, notamment au delà de $x/h = 0.5$, soit dans la zone de recollement. La méthode de simulation de recollement dans 2 nécessiterait peut-être plus d'attention, comme un choix plus précis de la taille de première maille.

Maintenant que nous avons vu comment se comporte le modèle kOmegaSSt, voyons comment d'autres modèles se comporte pour les mêmes champs d'intérêts.

2.3 Activité 3

On s'intéresse désormais au modèle kEpsilon standart décrit en introduction 1.2. Nous allons ainsi refaire les étapes de l'activité 2.2 à des fins de comparaisons.

2.3.1 Le schéma *"bounded Gauss limitedLinear 1"*

On cherche une nouvelle fois à voir le nombre d'itérations nécessaire au modèle pour converger :

```
Time = 11

smoothSolver:  Solving for Ux, Initial residual = 1, Final residual = 0.0910535, No Iterations 8
smoothSolver:  Solving for Uy, Initial residual = 1, Final residual = 0.0886209, No Iterations 8
GAMG:  Solving for p, Initial residual = 1, Final residual = 0.081852, No Iterations 2
time step continuity errors : sum local = 3.73162e+71, global = -2.71242e+71, cumulative = -2.71242e+71
smoothSolver:  Solving for epsilon, Initial residual = 1, Final residual = 0.0561327, No Iterations 2
bounding epsilon, min: -3.9458e+114 max: 2.88757e+115 average: 8.97223e+111
smoothSolver:  Solving for k, Initial residual = 1, Final residual = 0.0230551, No Iterations 1
bounding k, min: -9.13688e+117 max: 1.8178e+119 average: 1.27294e+115
ExecutionTime = 0.95 s  ClockTime = 1 s

Time = 12

smoothSolver:  Solving for Ux, Initial residual = 0.892431, Final residual = 0.0331025, No Iterations 1
smoothSolver:  Solving for Uy, Initial residual = 0.59258, Final residual = 0.0515929, No Iterations 5
#0  Foam::error::printStack(Foam::Ostream&) at ??:?
#1  Foam::sigFpe::sigHandler(int) at ??:?
#2  ? in /lib/x86_64-linux-gnu/libpthread.so.0
#3  Foam::scalarProduct<double, double>::type Foam::sumProd<double>(Foam::UList<double> const&,
Foam::UList<double> const&) at ??:?
#4  Foam::PCG::scalarSolve(Foam::Field<double>&, Foam::Field<double> const&, unsigned char) const at ??:?
#5  Foam::GAMGSolver::solveCoarsestLevel(Foam::Field<double>&, Foam::Field<double> const&) const at ??:?
#6  Foam::GAMGSolver::Vcycle(Foam::PtrList<Foam::lduMatrix::smoother> const&, Foam::Field<double>&,
Foam::Field<double> const&, Foam::Field<double>&, Foam::Field<double>&, Foam::Field<double>&,
Foam::Field<double>&, Foam::Field<double>&, Foam::PtrList<Foam::Field<double> >&,
Foam::PtrList<Foam::Field<double> >&, unsigned char) const at ??:?
#7  Foam::GAMGSolver::solve(Foam::Field<double>&, Foam::Field<double> const&, unsigned char) const at ??:?
#8  Foam::fvMatrix<double>::solveSegregated(Foam::dictionary const&) at ??:?
#9  Foam::fvMatrix<double>::solveSegregatedOrCoupled(Foam::dictionary const&) at ??:?
#10  Foam::fvMesh::solve(Foam::fvMatrix<double>&, Foam::dictionary const&) const at ??:?
#11  ? in /usr/lib/openfoam/openfoam2106/platforms/linux64GccDPInt32Opt/bin/simpleFoam
#12  __libc_start_main in /lib/x86_64-linux-gnu/libc.so.6
#13  ? in /usr/lib/openfoam/openfoam2106/platforms/linux64GccDPInt32Opt/bin/simpleFoam
```

FIGURE 5 – Convergence de la solution

On constate qu'avec le schéma *"bounded Gauss limitedLinear 1"* qui est un schéma avec reconstruction linéaire d'ordre 2, le modèle kEpsilon standard diverge après quelques itérations seulement. Il semble donc que le schéma utilisé ici ne soit pas assez stable pour l'étude d'un écoulement turbulent au dessus d'une géométrie complexe. Pour corriger cela on va changer le schéma de résolution pour les termes convectifs des équations de transport turbulentes.

2.3.2 Le schéma *"bounded Gauss upwind"*

On remplace le schéma précédent par le schéma *"bounded Gauss upwind"* :

```
Time = 600

smoothSolver:  Solving for Ux, Initial residual = 6.99756e-05, Final residual = 4.78669e-06, No Iterations 3
smoothSolver:  Solving for Uy, Initial residual = 0.000661548, Final residual = 3.50624e-05, No Iterations 3
GAMG:  Solving for p, Initial residual = 0.0215261, Final residual = 0.00147915, No Iterations 1
time step continuity errors : sum local = 0.0106678, global = -1.41755e-06, cumulative = -4.5993
smoothSolver:  Solving for omega, Initial residual = 9.37423e-07, Final residual = 3.28409e-08, No Iterations 2
smoothSolver:  Solving for k, Initial residual = 0.000148059, Final residual = 8.41997e-06, No Iterations 2
bounding k, min: -0.000659053 max: 65.7996 average: 10.3654
ExecutionTime = 28.42 s  ClockTime = 31 s

yPlus yPlus write:
    writing field yPlus
    patch upperWall y+ : min = 0.506262, max = 1.82393, average = 1.20091
    patch lowerWall y+ : min = 0.00408791, max = 8.78596, average = 1.50107
    functionObjects::turbulenceFields stressComponents writing field: turbulenceProperties:devReff
    functionObjects::pressure pressureCoefficient writing field: cp
writeCellCentres writeCellCentres write:
    writing cell-volumes field C to 600
    Writing the x component field of the cell-centres Cx to 600
    Writing the y component field of the cell-centres Cy to 600
    Writing the z component field of the cell-centres Cz to 600
wallShearStress wallShearStress write:
    writing field wallShearStress
    min/max(upperWall) = (-4.90115 -0.00298942 -2.49044e-20), (-2.11519 0.0117226 1.83185e-20)
    min/max(lowerWall) = (-5.18129 -1.9009 -1.58638e-18), (1.4916 0.221135 2.49441e-18)
```

FIGURE 6 – Convergence de la solution

En changeant de schéma on constate que le modèle compile. Plus précisément, on est capable de voir les différentes opérations faites par le modèle :

— la convergence est mesuré par le calcul du résidu à chaque itération ;

— on voit qu'il faut un certain nombre d'itérations pour chaque pas de temps pour que le résidu soit assez petit pour que le modèle soit considérer convergent ;

— le modèle semble être une combinaison de différents schémas avec termes de correction, qui doivent chacun être vérifié.

2.3.3 Convergence vers l'état stationnaire

Dans la continuité de notre étude, il est intéressant de vérifier si le modèle kEpsilon standard converge plus ou moins rapidement que le modèle kOmegaSST :

```
Time = 1077

smoothSolver:  Solving for Ux, Initial residual = 6.36777e-07, Final residual = 5.35249e-08, No Iterations 3
smoothSolver:  Solving for Uy, Initial residual = 9.89709e-07, Final residual = 6.38868e-08, No Iterations 3
GAMG:  Solving for p, Initial residual = 1.25169e-05, Final residual = 1.19765e-06, No Iterations 1
time step continuity errors : sum local = 7.60976e-06, global = 1.74005e-08, cumulative = 0.225134
smoothSolver:  Solving for epsilon, Initial residual = 3.77091e-07, Final residual = 2.08335e-08, No
Iterations 2
smoothSolver:  Solving for k, Initial residual = 1.83863e-06, Final residual = 1.42326e-07, No Iterations 3
ExecutionTime = 46.97 s  ClockTime = 49 s


SIMPLE solution converged in 1077 iterations

yPlus yPlus write:
    writing field yPlus
    patch upperWall y+ : min = 0.585071, max = 2.43296, average = 1.58951
    patch lowerWall y+ : min = 0.00533491, max = 8.90415, average = 1.7082
    functionObjects::turbulenceFields stressComponents writing field: turbulenceProperties:devReff
    functionObjects::pressure pressureCoefficient writing field: cp
writeCellCentres writeCellCentres write:
    writing cell-volumes field C to 1077
    Writing the x component field of the cell-centres Cx to 1077
    Writing the y component field of the cell-centres Cy to 1077
    Writing the z component field of the cell-centres Cz to 1077
wallShearStress wallShearStress write:
    writing field wallShearStress
    min/max(upperWall) = (-8.72074 -0.00328921 -2.7249e-20), (-3.66983 0.0175433 1.21207e-20)
    min/max(lowerWall) = (-8.69362 -0.464427 -9.89153e-19), (2.65715 0.0747178 1.08417e-18)
End
```

FIGURE 7 – Convergence de la solution

En conservant la même tolérance qu'avec le modèle kOmegaSST sur u et sur p, on constate que le modèle kEpsilon converge vers un état stationnaire pour un endtime de 1077. Cette fois ci le calcule converge, laissant penser à un gain de précision via le modèle kEpsilonSST par rapport au modèle kOmegaSST 2.2. Vérifions si cela se retranscrit dans les résultats numériques.

Superposition Driver&Seegmiller / simulation numérique

Pour vérifier ça, on trace de nouveau le coefficient de frottement Cf le long de l'écoulement :

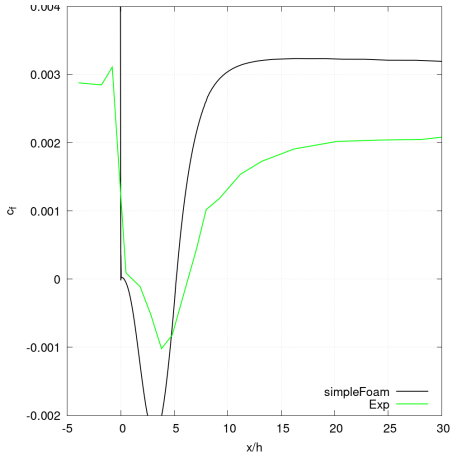


FIGURE 8 – Comparaison du Cf expérimentale et numérique

On constate que même si le comportement globale se retrouve dans le modèle kEpsilon, tout cela se fait à un coefficient multiplicateur prêt. Ainsi, le modèle kEpsilon standard reproduit correctement les étapes et notamment les changements de courbe, mais il surestime la valeur du Cf avant et après la marche ; et sous-estime sa valeur au niveau du saut (i.e. $x/c = 0$).

Ce modèle semble donc ne pas quantifier correctement les frottements à la paroi, ne reproduisant que grossièrement le comportement de saut. On pouvait s'y attendre, car le **modèle kEpsilon** est connu pour ses **qualités en champs libre** mais **pas en proche paroi**.

Distribution de yPlus le long de la paroi inférieure

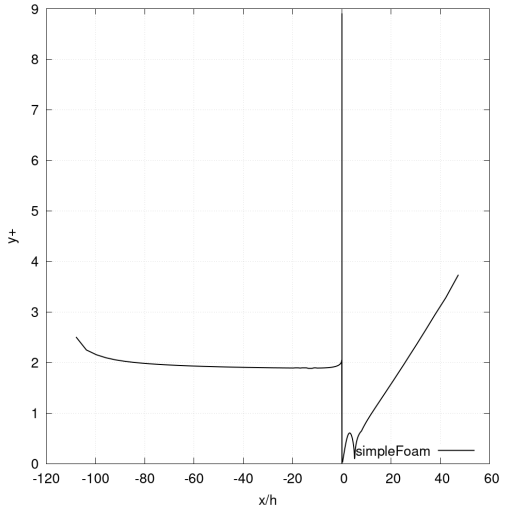


FIGURE 9 – Distribution de yplus

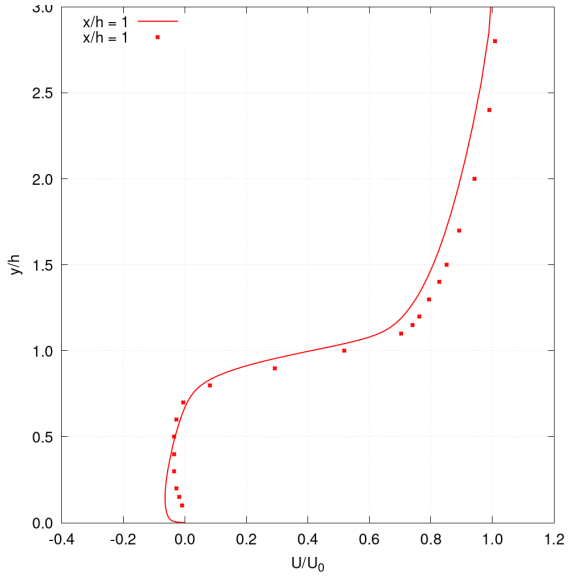
Sans plus de commentaire, le profil yPlus a le même comportement et donc le même "schéma de calcul" que le modèle précédent.

Afin de conclure sur la comparaison entre les 2 modèles, on va pour finir s'intéresser aux profils de vitesses le long de l'écoulement.

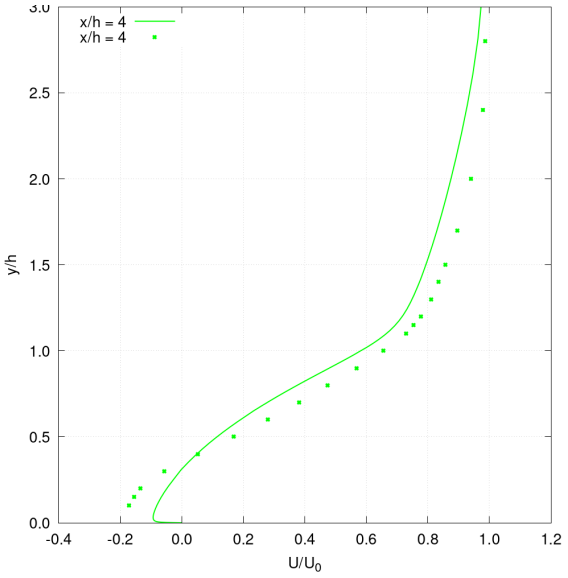
Profils de vitesse longitudinale

Les profils de vitesses sont tracés aux mêmes points de l'écoulement, pour une comparaison cohérente :

(a) x/h=1



(b) x/h=4



(c) x/h=6 (d) x/h=10

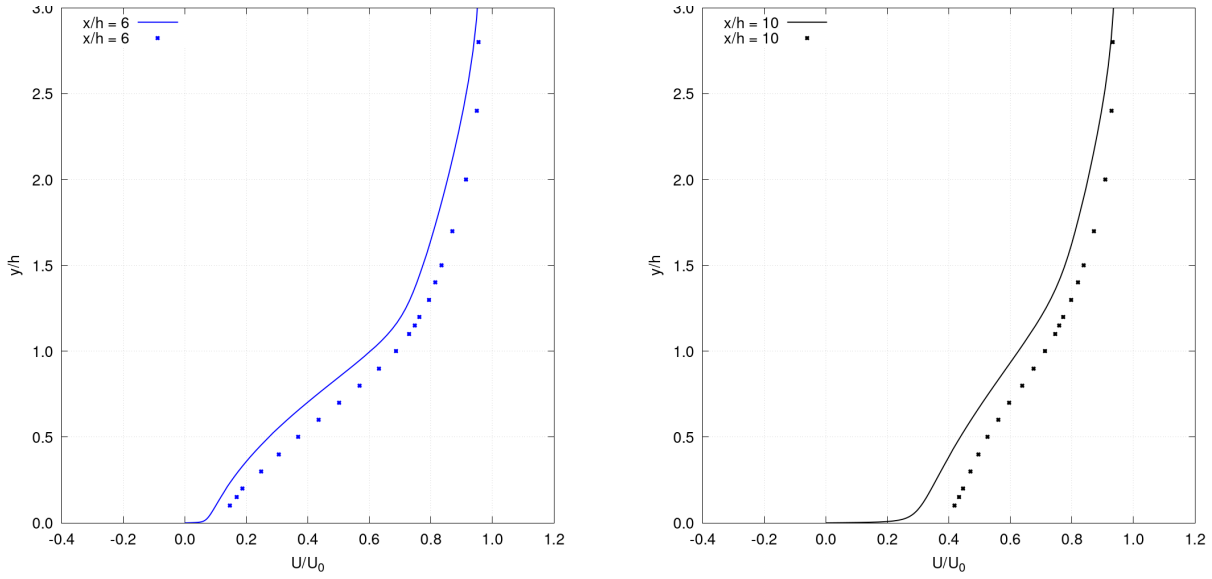


FIGURE 10 – Profils de vitesse - kEpsilon standard

On peut remarquer que les profils de vitesse donnés par le modèle kEpsilon collent moins bien aux résultats expérimentaux de Driver & Seegmiller que ceux donnés par le modèle kOmegaSST. En fait, là où ce modèle reproduit bien les variations et le comportement général du profil de vitesse, il a tendance à le faire apparaître plus tôt dans le graphe.

Conclusion

En sommes, le modèle ne reproduit ni les bonnes quantités ni ne les attribue à leur position physique. Le modèle kEpsilon n'est donc pas adapté à l'étude de turbulence sur géométrie complexe. Afin de palier à cela, des corrections existent, comme par exemple le modèle RNG.

2.3.4 Le modèle RNG

Le modèle RNG 1.3 conserve les mêmes bases que le modèle kEpsilon mais s'accompagne d'une correction via "une fonction d'amortissement non-linéaire" pour la formulation de la viscosité turbulente.

Ainsi, on viendrait corriger les écarts d'amplitude et de positionnement des différentes quantités, sans mettre à mal la stabilité du système.

Dans un premier temps, en conservant la même tolérance qu'avec le modèle kOmegaSST et donc que kEpsilon sur u et sur p, on constate que le modèle kEpsilonRNG converge vers un état stationnaire pour un endtime de 1168, ce qui est de l'ordre des temps du modèle 2.3.

Superposition des données de Driver&Seegmiller pour le Cf avec la courbe issue de la simulation

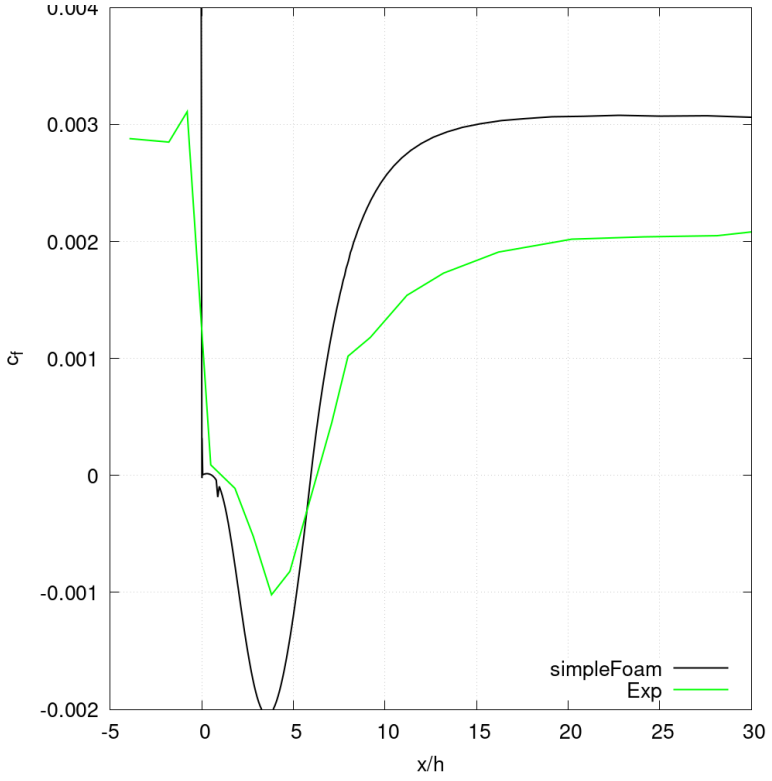


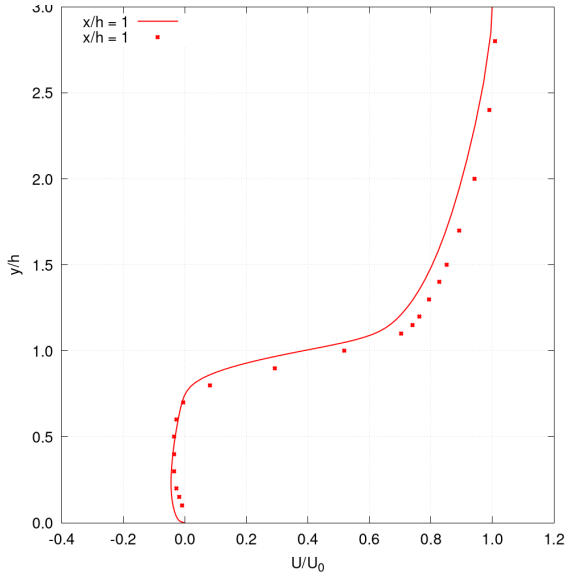
FIGURE 11 – Comparaison du Cf expérimentale et numérique

On constate que l'ajout du terme d'amortissement a corrigé les valeurs de Cf, voir un peu trop. Il semble ainsi que les écarts observés ci-dessus soit plutôt dûs à une erreur de calibrage, le comportement initial ayant été corrigé dans le bon sens. Il faudrait ainsi **retravailler le coefficient d'amortissement**, sont **comportement** générale étant celui **attendu**.

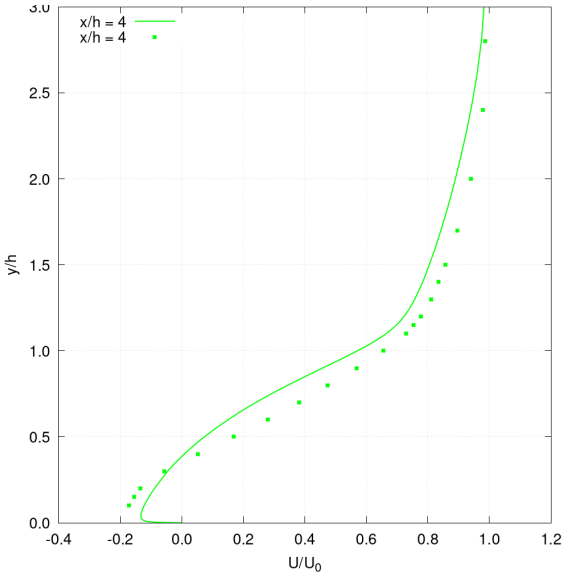
Tandis que le graphe de yPlus reste globalement inchangé pour ce modèle, par rapport aux précédents, on va conclure sur la pertinence de la correction du modèle kEpsilonRNG en regardant une nouvelle fois les profils de vitesses en 4 positions de l'écoulement.

Profils de vitesse longitudinale

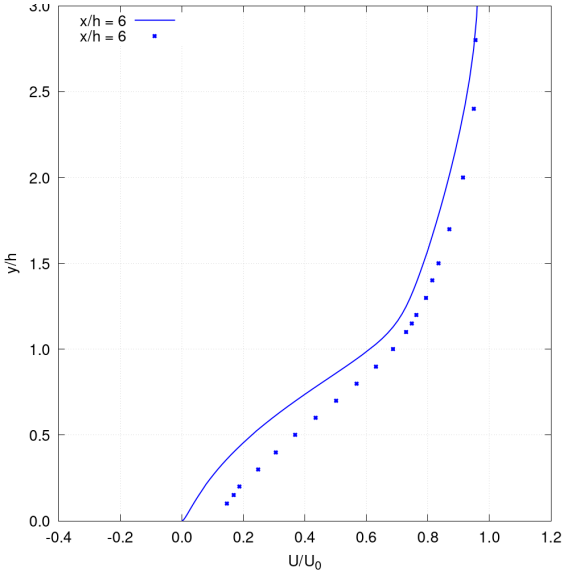
(a) x/h=1



(b) x/h=4



(c) x/h=6



(d) x/h=10

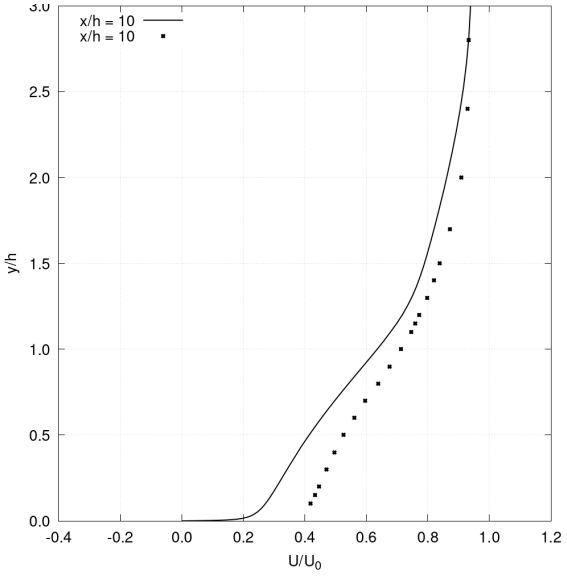


FIGURE 12 – Profils de vitesse - kEpsilonRNG

Des figures ci-dessus il vient que les profils de vitesses donnés par le modèle kEpsilonRNG ne colle pas beaucoup plus aux résultats expérimentaux de Driver & Seegmiller après application du terme d'amortissement. En effet, l'écoulement à l'infini est un peu mieux modélisé mais cela

n'est pas le cas du reste du domaine. Il y a ainsi toujours les erreurs d'adhérence à la paroi, et le modèle met toujours autant de temps pour recoller à la littérature.

Conclusion modèles

Les conclusions sur ces modèles sont que, de la présence d'une discontinuité géométrique, une attention particulière devrait être porté au comportement à la paroi, trop grossièrement modélisé ici. On constate également que le modèle kOmegaSST reste le plus précis pour ce type d'écoulement, alors que les modèles basés sur le modèle kEpsilon restent imprécis dans cette configuration, malgré la correction apportée par le modèle RNG au niveau de la marche descendante.

Nous allons maintenant utilisé un nouveau modèle, le modèle LRR 1.4, basé non plus sur l'énergie turbulente k comme les modèles étudiés jusqu'à maintenant, mais sur le transport des contraintes de Reynolds.

2.4 Activité 4

On termine notre étude de turbulence au dessus d'une marche par l'introduction d'un nouveau modèle, le modèle LRR 1.4.

2.4.1 Calcul en partant du temps 0

Avec le modèle LRR, si l'on part du temps 0, le calcul diverge et ne tend pas vers la solution souhaité, et de ce fait, le calcul ne se fait pas. Car ce modèle a besoin d'être initialisé.

2.4.2 Calcul comme une reprise du calcul kEpsilon

Si on lance cette fois, le modèle LRR à partir d'un calcul convergé du modèle kEpsilon, le calcul comme mentionné est déjà convergé et par conséquent, le calcul ne se lance pas non plus. On obtient alors le même résultat qu'avec le modèle kEpsilon. Un moyen de reprendre le calcul kEpsilon est par exemple de d'augmenter la tolérances pour faire d'avantages d'itérations ou de stopper le calcul de kEpsilon avant sa convergence et d'ensuite le continuer avec le modèle LRR, ce qui permet d'initialiser le modèle LRR avec les données du modèle kEpsilon standard.

C'est cette deuxième méthode qui sera effectuée de façon automatique via un ensemble de fonctions, afin de comparer la précision de ce modèle avec les mêmes critères que les modèles précédemment présentés.

En appliquant la première méthode, notamment avec un residualControl de $1e-10$ pour u et p, on observe très peu de diffèrence.

2.4.3 Robustesse du modèle LRR

Pour utiliser le modèle LRR nous devons avant tout ajouter des facteurs de relaxation dans le fichier fvSolution, pour améliorer la stabilité du calcul . On utilise un facteur de 0.2 pour U et pour R. De plus, on change plusieurs schémas dans fvschemes comme présenté ci dessous en prenant en considération les termes de convection et les termes de diffusion :

```
divSchemes
{
    default         none;

    div(phi,U)       bounded Gauss LUST grad(U);
    div(phi,k)       bounded Gauss upwind;
    div(phi,epsilon) bounded Gauss upwind ;
    div((nu*dev2(T(grad(U))))) Gauss linear;
    div(phi,omega)   bounded Gauss limitedLinear 1;
    div(R)           Gauss linear;
    div(phi,R)       bounded Gauss upwind;
    div(nonlinearStress)       bounded Gauss linear;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}
```

FIGURE 13 – Schémas numériques divergence

On peut dire que le modèle LRR n'est pas robuste, car on a pu voir que ses résultats et ses prévisions ne sont pas toujours exacts, lorsque une ou plusieurs des variables d'entrée ou des hypothèses sont radicalement modifiées.

2.4.4 Résultats numériques du modèle LRR

Superposition des données de Driver&Seegmiller pour le Cf avec la courbe issue de la simulation

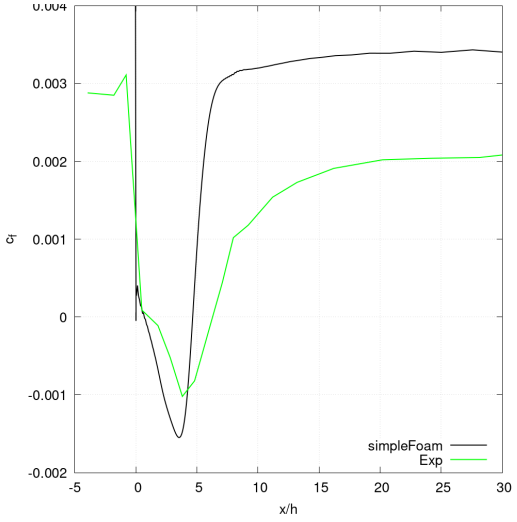


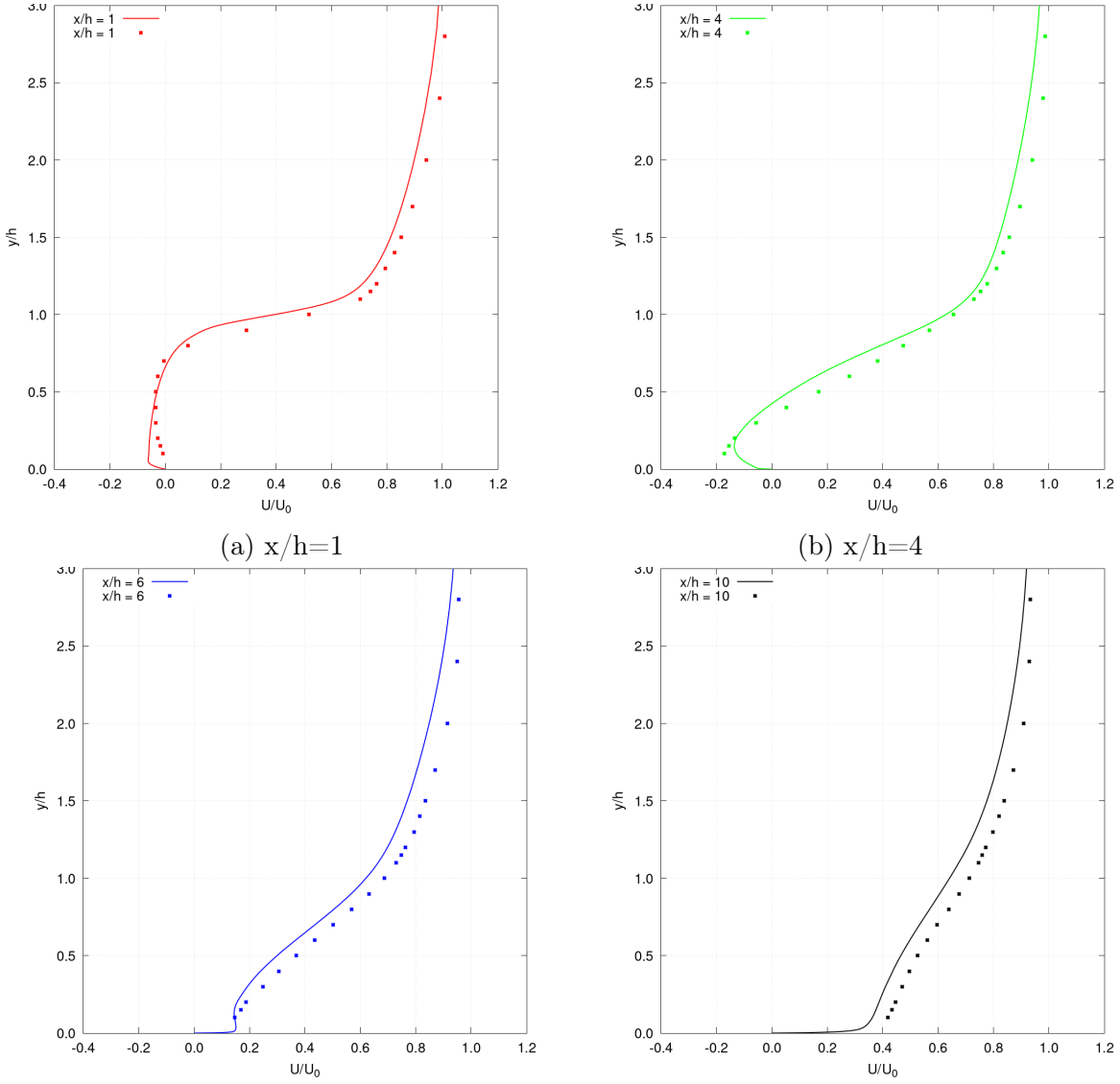
FIGURE 14 – Comparaison du Cf expérimentale et numérique

On constate que, tout comme le modèle kEpsilon standard, le modèle LRR surestime la valeur du Cf avant et après la marche. Cependant, au niveau de la marche descendante, l'amplitude de Cf est bien mieux respectée. Son comportement colle donc assez bien à l'expérience au niveau de la marche contrairement au modèle kEpsilon.

On voit donc ici l'importance d'adapter et appliquer des facteurs différents en fonctions de chaque étude, afin de calibrer au mieux les modèles et d'avoir un modèle le plus robuste possible.

Profils de vitesse longitudinale

Voyons comment le modèle LRR améliore par la même occasion les résultats du modèle kEpsilon standard :



(a) x/h=1

(b) x/h=4

(c) x/h=6

(d) x/h=10

FIGURE 15 – Profils de vitesse

On peut remarquer que les profils de vitesse donnés par le modèle LRR collent mieux aux résultats expérimentaux de Driver & Seegmiller que ceux donnés par le modèle kepsilon, sans pour autant palier à toutes les irrégularités citées précédemment.

On corrige ainsi les écarts de pente initiale et donc les première valeurs des profils de vitesse. Néanmoins, les corrections apportées par le modèle LRR ne permettent pas de combler totalement ces écarts de valeurs le long des y croissants.

3 Conclusion

En conclusion, à travers ce TP nous avons pu comparer plusieurs modèles de turbulence parmi les plus utilisés pour un cas avec gradient de pression et décollement, pour lequel la turbulence est loin d'un état d'équilibre.

Nous avons pu constater que la discrétisation numérique des équations de transport a une grande importance sur la robustesse des simulations RANS. On a pu le constater notamment avec le modèle kEpsilon.

De plus, ce TP nous aura permis de mettre en données plusieurs modèles de turbulence sous OpenFoam, dont le modèle aux tensions de Reynolds de Launder, Reece et Rodi (LRR). Il est intéressant de remarquer à quel point les paramètres initiaux ont une importance sur la convergence de la solution.

Du fait de ses qualités en proche paroi, le modèle kOmegaSST reste le meilleur ici pour un écoulement avec gradient de pression et décollement, son rapport côut/précision est bien meilleur que ceux des autres modèles que nous avons pu comparé. Cependant, nous avons également pu voir que des corrections permettaient d'améliorer le modèle kEpsilon dans cette configuration et cela en augmentant un peu le coup de calcul de celui-ci. Pour finir, ce fut intéressant d'utiliser un modèle complètement diffèrent, le modèle LRR, qui a montré des qualités pour modéliser l'écoulement au niveau de la discontinuité.

Références

[1] Christopher Rumsey. 2d backward facing step. https://turbmodels.larc.nasa.gov/backstep_val.html. Archive Nasa. Last updated 18-nov-2021.

[2] L. Sciacovelli. High-fidelity simulations for turbulent flows. Modeling handout, 2021-2022.

[3] H.Lee Seegmiller D.M.Driver. Features of reattaching turbulent shear layer in divergent channel flow. *AIAA Journal*, Vol. 23, No. 2 :163–171, Feb 1985.