Advanced CFD: Project Report

Implementing a New Turbulence Model in OpenFOAM

1. Introduction

This project was focussed on the modelling of turbulence in computational fluid dynamics (CFD). Specifically, it aimed to implement a new turbulence model in OpenFOAM; then to look at the process of validating the model against experimental data or data from a direct numerical simulation (DNS) and to compare it against the existing implemented models. The model to be implemented was the $\zeta - f$ model by Hanjalik et al (2004) [1]. This was originally proposed as a more stable and better performing version of the $\overline{v^2} - f$ model proposed by Durbin (1991) [2]. Alongside the implementation in OpenFOAM, the model was validated against the Channel flow DNS case by Kim et al (1987) [3] with comparisons to the $\overline{v^2} - f$ model and assessing the effects of mesh wall refinement. Furthermore the project encountered a number of other important aspects of CFD, this included stability considerations, advanced meshing techniques, automated post-processing and balancing the refinement level of meshes for RANS modelling.

2. Background and Theory

2.1. Turbulence Modelling

The full behaviour of incompressible fluids, including turbulence, is described by the incompressible Navier-Stokes (NS) equations, shown in equations 1 and 2. However the NS equations are non-linear and closely coupled making them extremely difficult to solve. One technique for solving these is DNS; this is very computationally expensive with the cost proportional to $\mathrm{Re}^{9/4}$. DNS is therefore only used for simple cases when producing data for validation of models.

$$\nabla \cdot \underline{u} = 0 \qquad \textbf{(1)}$$

$$\frac{\partial \underline{u}}{\partial t} + \nabla \cdot \underline{u}\underline{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \underline{u} \qquad \textbf{(2)}$$

Models are used to reduce the computational cost by replacing some terms of the equations, this reduces the effect of the coupling making them easier to solve. Turbulence models apply an averaging operation to the NS equations. After some rearranging, this collects a number of terms into the Reynolds Stress Tensor, $\underline{R}$, as shown in equation 3.

$$\frac{\partial \bar{\underline{u}}}{\partial t} + \nabla \cdot \left(\bar{\underline{u}}\bar{\underline{u}}\right) + \nabla \cdot \underline{R} = \frac{1}{\rho}\nabla \bar{p} + \nu \nabla^2 \bar{\underline{u}} \qquad \textbf{(3)}$$

There are two main types of turbulence models; Reynolds Averaged Navier-Stokes (RANS) and the Volumetric averaged. The volumetric averaged models consist of Large Eddy Simulation

(LES) and Detached Eddy Simulation (DES); these simulate the larger eddies in the flow and apply other models for the small eddies dominated primarily by viscous effects. Although volumetric averaged models are more accurate they are also more computationally expensive, instead RANS models will be the primary focus of this report.

RANS focuses on modelling the Reynolds Stress Tensor; one of these methods is the Turbulent Viscosity approach which represents the tensor as simpler variables. The other method is the Reynolds Stress approach which directly models the components of the tensor.

2.2. $\zeta - f$ Model Theory

The $\zeta - f$ model is part of the turbulent viscosity RANS models. This family of turbulence models relate the anisotropic part of the Reynolds Stress Tensor to a turbulent viscosity, ν_t. This produces the averaged NS equations as in equation 4, these are simplified but lose the anisotropy that occurs in the flow in near-wall regions. The $\overline{v^2} - f$ and $\zeta - f$ models introduce a streamline-normal stress term which operates alongside the k-ε equations and the elliptic dampening function f. This attempts to introduce anisotropy into the flow when normal to the streamlines to counter the assumptions, and consequent inaccuracies, made by the turbulent viscosity models. The isotropy assumption effectively removes directional dependence, this breaks down in particular as the flow approaches walls. Models such as the standard k-ε require wall functions to mitigate the inaccuracies of this assumption, conversely the $\overline{v^2} - f$ and $\zeta - f$ models are valid up to the wall so require a boundary layer mesh to fully resolve the flow. The main advantage of the $\overline{v^2} - f$ and $\zeta - f$ models is therefore in wall-bounded flows.

The $\overline{v^2} - f$ and $\zeta - f$ models are an extension to the k-ε model, whereby two additional equations are solved. The $\zeta - f$ is a variant of $\overline{v^2} - f$, where $\zeta = \overline{v^2}/k$. The main problem with the models is that the additional equations make them closely coupled and highly unstable. The $\zeta - f$ was proposed as an improvement on the $\overline{v^2} - f$, with better numerical stability and a reduced sensitivity to grid non-uniformities and mesh refinement at the walls. The $\zeta - f$ model equations are as follows in equations 4-12:

$$\frac{Dk}{Dt} = P_k - \varepsilon + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] \qquad \textbf{(4)}$$

$$\frac{D\varepsilon}{Dt} = \frac{C^*_{\varepsilon 1} P_k - C_{\varepsilon 2}\varepsilon}{T} + \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] \qquad \textbf{(5)}$$

$$\frac{D\zeta}{Dt} = f - \frac{\zeta}{k}P_k + \frac{\partial}{\partial x_k}\left[\left(\nu + \frac{\nu_t}{\sigma_\zeta}\right)\frac{\partial \zeta}{\partial x_k}\right] \qquad \textbf{(6)}$$

$$L^2\nabla^2 f - f = \frac{1}{T}\left(C_1 + C'_2\frac{P_k}{\varepsilon}\right)\left(\zeta - \frac{2}{3}\right) \quad \textbf{(7)}$$

Where the turbulent time and length scales and the turbulent viscosity is are calculated as:

$$T = max\left[min\left(\frac{k}{\varepsilon},\ \frac{a}{\sqrt{6}C_\mu|S_{ij}|\zeta}\right),\ C_T\left(\frac{\nu}{\varepsilon}\right)^{\frac{1}{2}}\right] \quad \textbf{(8)}$$

$$L = C_L max\left[min\left(\frac{k^{3/2}}{\varepsilon},\ \frac{k^{1/2}}{\sqrt{6}C_\mu|S_{ij}|\zeta}\right),\ C_\eta\left(\frac{\nu^3}{\varepsilon}\right)^{\frac{1}{4}}\right] \quad \textbf{(9)}$$

$$\nu_t = C_\mu \zeta k T \quad \textbf{(10)}$$

The mean strain rate tensor, S_{ij}, and the kinetic energy production term, P_k, are:

$$S_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) \quad \textbf{(11)}$$

$$P_k = 2\nu_t S_{ij} S_{ij} \quad \textbf{(12)}$$

The constants from these equations are defined as:

$$C_\mu = 0.22;\quad C_\eta = 85.0;\quad C_1 = 0.4;\quad C_2 = 0.65;\quad C_L = 0.36;\quad C_T = 6.0;$$
$$C_{\varepsilon 1} = 1.4(1 + 0.012/\zeta);\quad C_{\varepsilon 2} = 1.9;\quad \alpha = 0.6;\quad \sigma_k = 1.0;\quad \sigma_\varepsilon = 1.3;\quad \sigma_{zeta} = 1.2;$$

3. Implementation of the Model

3.1. Initial implementation

The turbulence models to be implemented in OpenFOAM, which uses the C++ programming language, are in the name-space RASModels within the name-space incompressible. The turbulence model itself is defined as a class. This class defines the equation constants, bounding values and the fields for each of the equations under the protected status. This prevents the values from being changed outside of the class, such as when called by another function; although not strictly necessary for operation this is best coding practice. The public status contains the constructor and destructor of the class and various functions which return variables. This allows other functions to access this information as well as containing parts of the calculation to tidy the code. It is worth noting that the public status does not contain any variable definitions, thus meaning that no functions calling the class can change the manner in which it operates. The class and function definitions are contained in the header file, whereas the contents of the functions are within the .C file; this keeps the code well organised and allows it to be easily understood.

The implementation of the models in OpenFOAM is best started by duplicating existing models and replacing the references to the variables and model names with those of the new model. In this case the $\overline{v^2}-f$ model had already been implemented and so this was duplicated and replaced with the $\zeta - f$ model name and variables. OpenFOAM has been designed to make the implementation of new equations very readable, it contains functions for each operator allowing the equation to be constructed in a similar manner to the mathematical notation. This is then passed to the 'solve' function which applies the numerical schemes to solve the equations. The original equation representation of the $\zeta - f$ model in OpenFOAM is as follows:

```
tmp<fvScalarMatrix> epsEqn
(
    fvm::ddt(epsilon_)
  + fvm::div(phi_, epsilon_)
  - fvm::laplacian(DepsilonEff(), epsilon_)
  ==
    Ceps1*G/T
  - fvm::Sp(Ceps2_/T, epsilon_)
);
```

```
tmp<fvScalarMatrix> kEqn
(
    fvm::ddt(k_)
  + fvm::div(phi_, k_)
  - fvm::laplacian(DkEff(), k_)
  ==
    G
  - fvm::Sp(epsilon_/k_, k_)
);
```

```
tmp<fvScalarMatrix> zetaEqn
(
    fvm::ddt(zeta_)
  + fvm::div(phi_, zeta_)
  - fvm::laplacian(DzetaEff(), zeta_)
  ==
  - fvm::Sp(G/k_, zeta_)
  + f_
);
```

```
tmp<fvScalarMatrix> fEqn
(
  - L2*fvm::laplacian(f_)
  ==
  - fvm::Sp(1, f_)
  - 1.0/T*
    (C1+C2*G/epsilon_)*
    (zeta_-2/3)
);
```

3.2. Preliminary Tests

In order to check the implementation of the equations a very cheap backward facing step case was prepared. This aimed to simply test whether the code worked as expected and to detect any errors that were missed in compiling before running the more expensive validation cases. The case required the RASProperties, fvSchemes and fvSolutions dictionaries to be set up. The RASProperties file had the entry of turbulence model set to zetaF. The fvSchemes dictionary had default schemes set up, this allowed the turbulence models to be switched over quickly without requiring the changing of all the files. The fvSolutions dictionary was also modified to include entries for each new variable to specify the solver and the tolerances; relaxation factors and residual controls were also entered. The initial conditions for ζ were set up with an inlet value of 2/3, note that ζ is dimensionless; this is calculated as $\overline{v^2} = 2k/3;\ \zeta = \overline{v^2}/k \ \Rightarrow\ \zeta = 2/3$ at the inlet. At the walls $\overline{v^2} = 0;\quad k = 0 \quad \Rightarrow\quad \zeta = \mathit{undefined}$ as a result ζ is often set to be close to zero, in this case 1e-10; since this is only an initial condition the exact value is not critical but the simulations are thought to react better to low wall values. The initial conditions of f are set to be zero-gradient at the inlet and close to zero at the walls, set as 1e-10s^{-1}. The initial conditions of k and ε are calculated in the same manner as for the standard k-ε model.

3.3. Stability Issues

After the original $\zeta - f$ had been implemented it was tested in the backward facing step case; it was found to run very quickly however it was also found to be highly unstable. It was first thought that this might be due to the sensitivity to the mesh. As a result the model was run in a full resolution validation case but it still diverged rapidly. In order to better understand the stability issues the implementation of the $\overline{v^2} - f$ model in OpenFOAM was inspected. It was found that this model was not the original model proposed by Durbin (1991) [2] instead this was a variation on the model proposed by Lien and Kalitzin [4], which had a few minor alterations to improve the stability. In particular these alterations are as follows in equations 13-15:

$$\frac{D\overline{v^2}}{Dt} = min(kf,\ -\alpha + C_2 P_k) - \frac{N\varepsilon\overline{v^2}}{k} - \frac{\partial}{\partial x}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial\overline{v^2}}{\partial x}\right] \qquad \textbf{(13)}$$

$$L^2\nabla^2 f = f + \frac{1}{k}(\alpha - C_2 P_k) \qquad \textbf{(14)}$$

$$\alpha = \frac{1}{T}\left((C_1 - N)\overline{v^2} - \frac{2}{3}k(C_1 - 1)\right) \qquad \textbf{(15)}$$

These alterations add in a term to the $\overline{v^2}$ equation which is able to break the positive feedback loop between the $\overline{v^2}$ and f equations. This loop means that essentially an increase in the value of $\overline{v^2}$ causes and increase in f which then increases $\overline{v^2}$. The term $min(kf,\ -\alpha + C_2 P_k)$ means that if f increases by too great an amount then another value is used in the $\overline{v^2}$ equation, thus breaking the feedback loop. This results in a far stabler model, with comparable accuracy. The consequences of this is that the $\zeta - f$ model, which was proposed as a stabler version of the original $\overline{v^2} - f$, is actually less stable than the implemented $\overline{v^2} - f$ version. Upon inspection it is found that the $\zeta - f$ equations still maintain the positive feedback loop which was the primary source of instability in the $\overline{v^2} - f$ model. To solve this issue it is necessary to apply a similar procedure to Lien and Kalitzin [4] to stabilise the model. This was attempted a number of separate ways, each of these was tested in the backward facing step case to assess the ease with which they can be run. The first of these implementations replaced the f-term in the ζ equation as follows in equation 16; the principle of this is to break the feedback loop. The new f-term was the same as from the stabilised $\overline{v^2} - f$ model but recalculated for $\zeta = \overline{\nu^2}/k$.

$$\frac{D\zeta}{Dt} = min\left(f,\ \frac{-\alpha + C_2 P_k}{k}\right) - \frac{\zeta}{k}P_k + \frac{\partial}{\partial x_k}\left[\left(\nu + \frac{\nu_t}{\sigma_\zeta}\right)\frac{\partial\zeta}{\partial x_k}\right] \qquad \textbf{(16)}$$

This variation also had severe stability issues which caused the solution to diverge rapidly. A further alternative was proposed which added in a modification to the $\zeta P_k / k$ term. Incorporating the term from the stabilised $\overline{v^2} - f$ model, recalculated for $\zeta = \overline{\nu^2}/k$, as shown in equation 17.

$$\frac{D\zeta}{Dt} = min\left(f, \frac{-\alpha + C_2 P_k}{k}\right) - min\left(\frac{\zeta}{k}P_k, \frac{N\varepsilon}{k}\zeta\right) + \frac{\partial}{\partial x_k}\left[\left(\nu + \frac{\nu_t}{\sigma_\zeta}\right)\frac{\partial\zeta}{\partial x_k}\right] \quad \textbf{(17)}$$

This implementation also had severe stability issues despite applying the smallest scaling factor to ζ from the original and stabilised models. A third variation to the $\zeta - f$ was then implemented entirely based on the stabilised $\overline{v^2} - f$ model; this adjusted the equations by setting $\zeta = \overline{v^2}/k$, as shown in equation 18.

$$\frac{D\zeta}{Dt} = min\left(f, \frac{-\alpha + C_2 P_k}{k}\right) - \frac{N\varepsilon\zeta}{k} + \frac{\partial}{\partial x_k}\left[\left(\nu + \frac{\nu_t}{\sigma_\zeta}\right)\frac{\partial\zeta}{\partial x_k}\right] \quad \textbf{(18)}$$

This model successfully ran with no stability issues, the constants used were the same as those used for the $\overline{v^2} - f$ model. Although, clearly were the model to be used to a greater extent, the constants should be evaluated for accuracy. The f-equation used for the implementation was also from the stabilised $\overline{v^2} - f$ model, adjusted with $\zeta = \overline{v^2}/k$, as in equation 19.

$$L^2\nabla^2 f - f = -\frac{C_2 P_k}{k} + \frac{1}{T}\left(\left(C_1 - N\right)\zeta - \frac{2}{3}\left(C_1 - 1\right)\right) \quad \textbf{(19)}$$

The code implementation of the stabilised $\zeta - f$ model is as shown below. The run time of the implemented $\zeta - f$ model was comparable to that of the implemented $\overline{v^2} - f$ model.

```
tmp<fvScalarMatrix> epsEqn
(
    fvm::ddt(epsilon_)
  + fvm::div(phi_, epsilon_)
  - fvm::laplacian(DepsilonEff(), epsilon_)
  ==
    Ceps1*G/T
  - fvm::Sp(Ceps2_/T, epsilon_)
);
```

```
tmp<fvScalarMatrix> kEqn
(
    fvm::ddt(k_)
  + fvm::div(phi_, k_)
  - fvm::laplacian(DkEff(), k_)
  ==
    G
  - fvm::Sp(epsilon_/k_, k_)
);
```

```
tmp<fvScalarMatrix> zetaEqn
(
    fvm::ddt(zeta_)
  + fvm::div(phi_, zeta_)
  - fvm::laplacian(DzetaEff(), zeta_)
  ==
  - fvm::Sp(N*epsilon_/k, zeta_)
  + min(f_, (-alpha + C2_*G)/k_)
);
```

```
tmp<fvScalarMatrix> fEqn
(
  - L2*fvm::laplacian(f_)
  ==
  - fvm::Sp(1, f_)
  - 1.0/k_*(alpha - C2_*G)
);
```

4. Meshing

Meshing can also be subdivided into structured and unstructured meshes. In structured meshing, cells are all of regular shape – importantly the cells are all hexes and not tetrahedra. In unstructured meshing the cells are often all tetrahedra or prisms, they are of different shapes and sizes and are optimised for quality. Unstructured meshes are quick to prepare and can provide good mesh for complex geometries. It is possible to put in refinement regions, which reduce mesh size in specific locations, and to create an inflation layer at the walls, which creates prism layer of cells providing the necessary number of layers to full resolve the flow in the boundary layers. However due to the complex nature of the algorithms a lot of control over the mesh is lost, in particular it is very difficult to get a high quality mesh with a good refinement level and surface resolution.

Conversely, structured meshes take a large amount of preparation but can provide a far superior mesh quality. The domain is divided into regions by connectors, each of these can have different numbers of cells along them and growth rates from the ends. These regions must be hexahedral with the same number of cells along each of the opposite sides. Boundary layer refinement is achieved by producing a region which follows the boundary layer and has first cell heights and the total number of cells set to the required resolution for the boundary layer. A few difficulties can occur when a boundary suddenly steps away from the flow, as the resolution on the step can lead to very thin cells in the domain. The level of control offered by a structured mesh is very high, particularly for simple geometries; it enables the meshes from specific validation and DNS cases to be replicated exactly. The Pointwise meshing software will be used for this project, this software can produce a very high quality mesh with structured, unstructured and prism cells. It allows full control over the mesh by placing connectors and setting dimensions and spacing constraints. The meshes can be exported to OpenFOAM, however they require renumberMesh to be run to set the mesh in the manner that OpenFOAM better recognises.

It is important when running CFD cases to consider correct meshing practices to ensure that results can be trusted. One important practice is a mesh sensitivity study since the refinement level of the computational mesh can effect the result. In areas of a high change in the value of flow quantities enough points must be evaluated in order to fully capture the true value. Equally, in areas with sharp changes in geometry the cells must be of a sufficient quantity to fully represent the surface. A further consideration is that with RANS models the mesh can be over-refined whereby the mesh is attempting to resolve flow scales less than the Kolmogorov length-scale, in which the viscous effects are dominating. The result of this over-refinement is that the equations often diverge, consequently the mesh should be refined down to the point where the resolution no longer effects the monitored outputs. For the validation cases the mesh resolution is often specified, particularly in DNS cases. In such an instance the mesh should be to the specifications described so that the mesh resolution does not cause differences between the simulation results, thus a mesh sensitivity study would not be necessary for a comparison to the data.

Another meshing practice is to fully resolve flows in the boundary layers; this is achieved via the use of an inflation layer mesh. The mesh is defined by the first layer height and number of layers; the first layer cell height should be such that the y+ value equals 1 and the number of layers should be sufficient to fully enclose the boundary layer. The first layer cell height is calculated as:

$$y = \frac{y^+ \nu}{U_\tau} \qquad where: \quad U_\tau = \sqrt{\frac{\tau_w}{\rho}} \; ; \quad \tau_w = \frac{C_f \rho U_\infty^2}{2} \; ; \quad C_f = \frac{0.026}{Re_x^{1/7}} \; ; \quad Re_x = \frac{U_\infty L}{\nu} \qquad (\mathbf{20})$$

5. Post-processing

The post-processing of CFD cases is a very important procedure, it covers the whole process of extracting relevant data from the simulations and representing the data in an appropriate format. Ordinarily the data might be extracted to analyse the drag forces and stagnation points caused by specific designs or used to assess the effects of a specific design on the flow or vice versa. Conversely in some applications of CFD, primarily model or software development, the main use of extracted data is in comparisons to experimental or DNS data to assess the accuracy of the predictions – this will be the primary use in this project. The OpenFOAM installation is accompanied by Paraview, this is another open-source software package that is widely used in industry. Paraview is built as a Python wrapper around a C++ architecture, this makes it highly programmable. When examined in full it is found that paraview actually consists of a number of packages, one of these is the graphical user interface (GUI) which allows the user to interact with the python wrapper with relative ease. Another is the two python installations, pvpython and pvbatch, which are originally written in python 2 but have some of the python 3 functionality and further modules can be imported via the command:

```
from __future__ import <required_module>
```

The pvbatch installation is specifically designed for parallel processing across multiple hosts – a vital feature for large cases. The parallel features of pvbatch are based on domain decomposition, whereby the main domain is split into smaller domains with interfaces between each other. This is a technique that favours CPU parallelisation rather than GPU parallelisation as it reaches a limit whereby the number of interfaces slows the processing down. For this project the parallel processing features were not used as the cases were just small enough to be processed without them.

The pvpython installation is an application, primarily with serial processing, which allows python scripts to be written which interact with the paraview python functions and classes to process the data. This essentially consists of a python 2 installation with all the required modules pre-installed and specific functions incorporated into the package as a module called paraview. Within this paraview module there are two main sub-modules, simple and server. The simple module handles almost all of the paraview post-processing capabilities, whereas the server module handles the management of the client and host machines when running in parallel or remotely; for this project only the simple module was used. The pvpython installation can be used as a python shell, whereby the commands are typed in and evaluated immediately, or it can be used to run scripts. These scripts can be prepared either by writing them in full or by recording a python trace of all the necessary actions via the GUI – the latter being more common in industry and the method used in this project.

The pvpython trace, once recorded, was edited to allow a preliminary declaration of the case names. Since the file structure for each case was identical, the input and output files could then be automatically calculated by the script. The input file format was originally attempted as the standard OpenFOAM format, however, this posed some difficulty in that the paraFoam application that is called also prepares the file in a format only readable through the GUI. Consequently the input file was first prepared using the foamToVTK command; this had another benefit in that the command can also reduce the data size to be processed. The foamToVTK command allows the latest time and specific fields to be included in the file, this is vital for large cases and particularly if near the processing limits of the computers, as it dramatically reduces the processing requirements. The command was called from the terminal as:

```
foamToVTK -latestTime -fields '(p U)'
```

It is also possible to call the command through python as:

```
import subprocess as sp
sp.call([“foamToVTK”, ”-latestTime”, ”-fields”, ”(p U)”])
```

Note: all importing of modules is done at the start of the script. Due to the large number of files to be post-processed it was important to attempt to automate as much of the process as possible to save time. This was attempted in pvpython via a for loop; a dictionary was created with the case names and corresponding end times, this could then be looped over. The script then changes into the case directory, in order to run the foamToVTK command, via the os module. Once the foamToVTK command is run, the rest of the script from the python trace is run – the script then moves onto the next loop.

Although the first looping worked without issue, the second iteration of the loop caused errors in the pvpython functions. This could not be rectified as the errors had no apparent source but were instead due to pvpython maintaining the objects and their properties from the previous loop. This was attempted to be rectified by defining the post-processing as a function, the loop then simply calls the function with the caseName and endTime variables as inputs. This aimed to clear the memory at the end of the function thus allowing the looping to continue. Unfortunately, this also did not succeed and the same errors occurred on the second loop. It is believed that the pvpython module mimics the GUI, in that an instance of pvpython is equivalent to maintaining an open GUI session. Thus all settings, function calls, objects and their attributes are remembered within the session and cannot be overwritten with a loop. The issue then arises in that the pvpython script cannot take arguments on being called outside of pvpython. The only way to loop the case would therefore be to write a shell script that uses a command such as 'sed' to edit the python files before running, it could then be included in a loop and run the cases automatically. This was considered, however, the level of automation already achieved within python was deemed suitable to proceed without automatic looping; as the time which would be saved was equal to the time cost of writing the additional shell scripts.

One of the primary reasons for the automated post-processing, alongside time saving, was to reduce the loading on the computers. The more powerful computers which were available and which could have been used did not have the ability to have macros scripts run; however this was coupled to the need to remotely access the computers rather than in situ. Although it is possible to access graphics via a ssh connection, by use of X11 forwarding, the rate of data-transfer was insufficient to make this plausible – particularly for the full resolution cases. The alternative was to process the cases locally on a less powerful machine. This was attempted using the GUI however the data to be processed overloaded the computer and forced it heavily into swap, eventually crashing it altogether. Consequently, a less demanding means of processing was required. By using the foamToVTK command, the amount of data input could be dramatically reduced. Further reductions were achieved by avoiding the use of the GUI and leaving the processing to be done in the background. These measures reduced the computational loading such that the cases could be run successfully on a far less powerful machine.

6. Channel Flow Validation

The purpose of validating models is to ensure that they produce reliable and accurate representations of the true flow conditions. This is of vital importance, without validation the accuracy of the models cannot be established and therefore their use in CFD would be very limited. Many canonical test cases exist for almost every main flow behaviour normally originating from experiments or DNS, indeed in many of the cases have multiple sets of results from separate publications. For this project it was originally intended to validate the implemented $\zeta - f$ against a number of these test cases. However, due to the severe instability issues experienced this was not possible; the other test cases that were attempted are a backward facing step and a 3D diffuser. The validation case that was successful was the channel flow DNS case by Kim et al (1987) [3].

6.1. Case Set-up

The mesh set-up for the channel flow DNS case of Kim et al (1987) [3] is defined relative to the geometry and has a Reynolds number of 3300 based on the channel half width, δ. The domain size and number of cells is defined as follows in Table 1, note that due to the nature of fluid dynamics the specifications by Kim et al are all given relative to the flow conditions. The validation case for this project set $\delta = 0.1m,$ giving the actual domain sizes as shown in Table 1. This also took a kinematic viscosity of 1.5e-5m^2s^{-1}, giving an inlet velocity of 0.495m/s. Kim et al proposed that the mesh used a y^+ at the first cell height of 0.05, such a resolution would be inappropriate for a RANS model so a y^+ of 1 was taken instead. The mesh had a total of 3962880 cells.

Another three meshes were produced for the channel flow case, one of these had no wall refinement but the same number of cells. The other two meshes had half the resolution in each direction, shown in Table 1, one without wall refinement the other with a y^+ of 1, both had 499200 cells. These meshes allow the effect of the mesh resolution to be assessed; specifically the effect of the global resolution on the accuracy of the model as well as the effect of the wall refinement. The meshes with half the resolution in each direction will be hereafter referred to simply as being of half resolution, note this does not refer to the cell count which has approximately eight times fewer cells.

Table 1: Domain and mesh specifications relative to the channel half width (δ)

Direction	**Domain Size**		**Number of Cells**	
	Relative	**Actual (m)**	**Full Resolution**	**Half Resolution**
x (stream-wise)	$4\pi\delta$	1.2566	192	96
y	2δ	0.2	129	65
z (span-wise)	$2\pi\delta$	0.6283	160	80

The inlet conditions had k, ε and $\overline{v^2}$ calculated using equations 21 and 22; ζ has an inlet value of 0.6667 (2/3) and f has a zero gradient condition, as explained in section 3.2. The wall conditions used values close to zero for k, ε, $\overline{v^2}$, ζ and f. It is worth noting that, although k and ε are often evaluated with wall functions, wall functions could not be used in this case due to the necessity to evaluate ζ and f up to the wall. The two sides of the domain in the span-wise direction were set separately to be symmetry planes, it is important in OpenFOAM that these two sides are in separate patches.

$$k = \frac{3}{2}\left(U_{ref} T_i\right)^2 \;;\quad \varepsilon = C_\mu^{3/4}\frac{k^{3/2}}{l} \;;\quad \overline{v^2} = \frac{2}{3}k \qquad \textbf{(21)}$$

Where:
$$T_i = 0.16\,\mathrm{Re}^{-1/8} \;;\quad l = 0.07\,L \;;\quad C_\mu = 0.09 \qquad \textbf{(22)}$$

The full resolution case with wall refinement were run for the $\overline{v^2}-f$ and $\zeta-f$ models whereas the full resolution case without wall refinement also included the k-ε model. The half resolution cases were both only run for the $\overline{v^2}-f$ and $\zeta-f$ models.

6.2. Results

6.2.1. Kim et al (1987) DNS data

Much of the DNS data presented by Kim et al [3] for the channel flow looks at the transient properties of the flow, since turbulence is an inherently transient process. RANS simulations, however, originate from a time-averaged operation on the flow; thus making transient simulations with RANS self-contradictory. This limits some of the data comparisons that could have been made, were another form of turbulence modelling used – such as LES. Kim et al [3] do present some time-averaged data regarding the velocity profiles, shown in Figure 1, this is compared against the law of the wall, a theoretical rule which is breaks down in the transition region of the boundary layer. Figure 1 (a) also compares to the experimental data of Eckelmann [5], which Kim et al found to have a systematic error so corrected the data to better fit the theoretical and DNS results. The U^+ value is calculated as: $U^+ = U/U_\tau$

Another data set that can be used is the pressure plot across the channel, this uses a normalised pressure and takes the root mean square (RMS) over time, thus the P_{rms} shown in Figure 1 (b) is calculated as P_{norm} in equation 23. This pressure profile is taken at a location such that the channel flow is fully developed. Kim et al noted that this pressure is particularly low compared to measurements from other cases, specifically Willmarth (1975) [6] had a pressure which varied between 2 and 3.

$$P_{norm} = \frac{P_{rms}}{\rho U_\tau^2} \; ; \quad P_{rms} = \sqrt{\frac{1}{T_2 - T_1} \int_{T_1}^{T_2} [P(t)]^2 dt} \qquad \textbf{(23)}$$

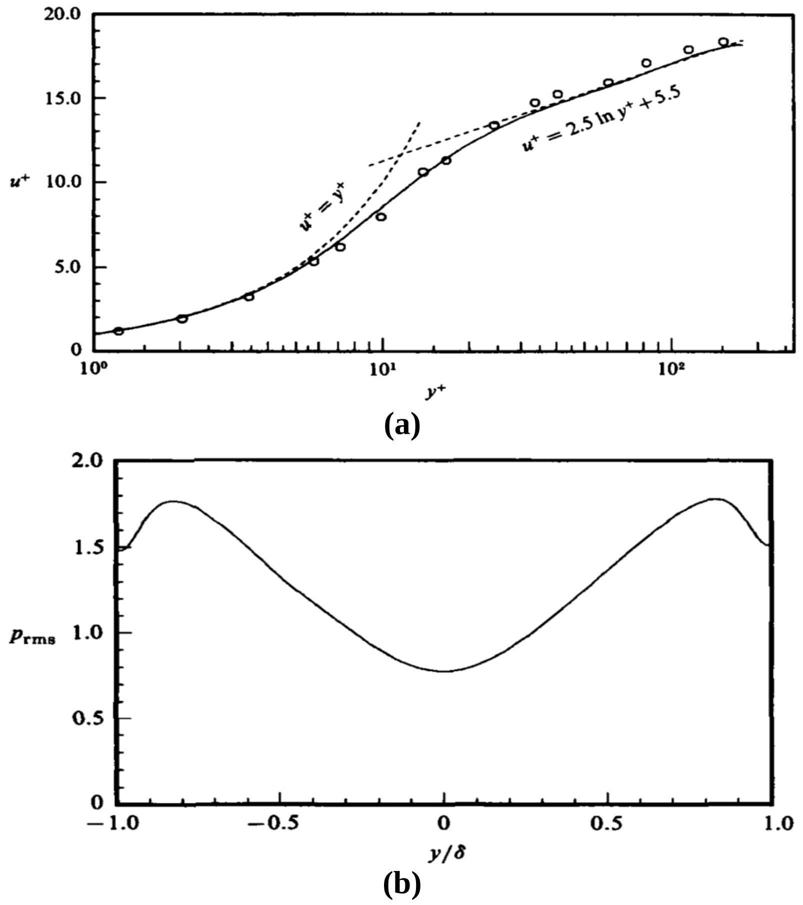


Figure 1: Plots of the normalised velocity (U^+) against the distance from the walls (y^+) from Kim et al (solid), Eckelmann (dots) and the law of the wall (dashed) (a). Normalised pressure against the distance from the centreline (y/δ) (b) from the DNS by Kim et al (1987) [3]

6.2.2. Full Resolution with Wall Refinement

The results of the full resolution case with wall refinement have been overlaid onto the data from Kim et al [3], with the data from Eckelmann [5] removed for clarity, in Figure 2. The $\overline{v^2}-f$ model, in blue, shows good agreement with the shape of the DNS data by Kim et al [3] but consistently over-predicts the velocity. This over-prediction is less when near to the wall and increases to its maximum around the transition point; this suggests that the near-wall inaccuracy may be velocity-dependent as it increases with the velocity. However the log-law region has a far lower gradient than the DNS data, although this corrects the inaccuracy of the velocity prediction, it also means that the shape of the velocity profile less accurate. The $\zeta-f$ model, in red, shows near-perfect agreement with the DNS data close to the wall ($y^+ < 10$). However the model becomes far less accurate after the transition region, indeed beyond $y^+ = 25$ the $\overline{v^2}-f$ model provides a better prediction of the velocity than the $\zeta-f$. Despite the later inaccuracies, the predictions of the $\zeta-f$ model for near-wall flows shows that, when using a full resolution mesh with wall refinement, the model is superior to the $\overline{v^2}-f$ in terms of achieving the correct velocity profile.

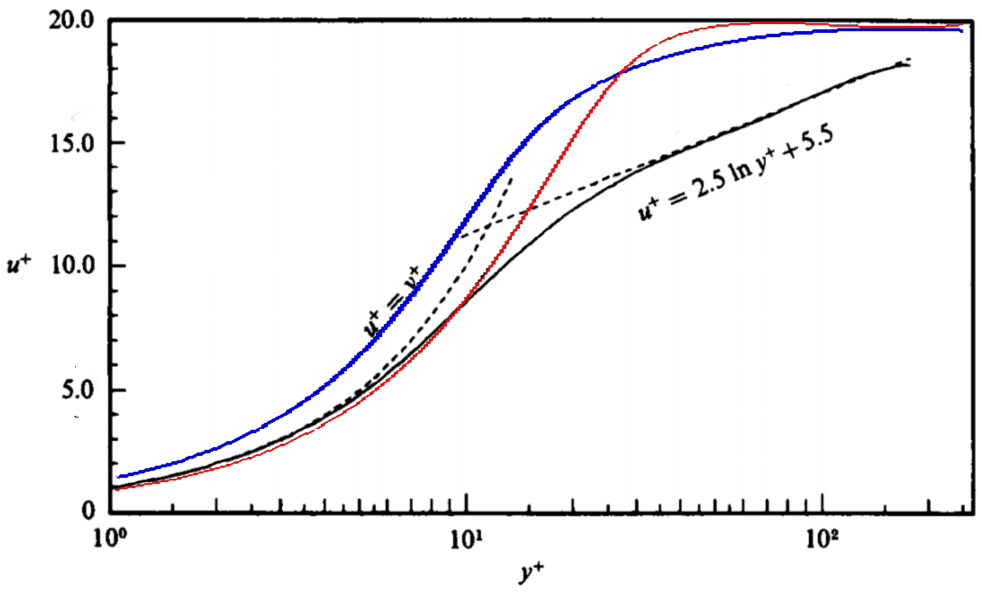
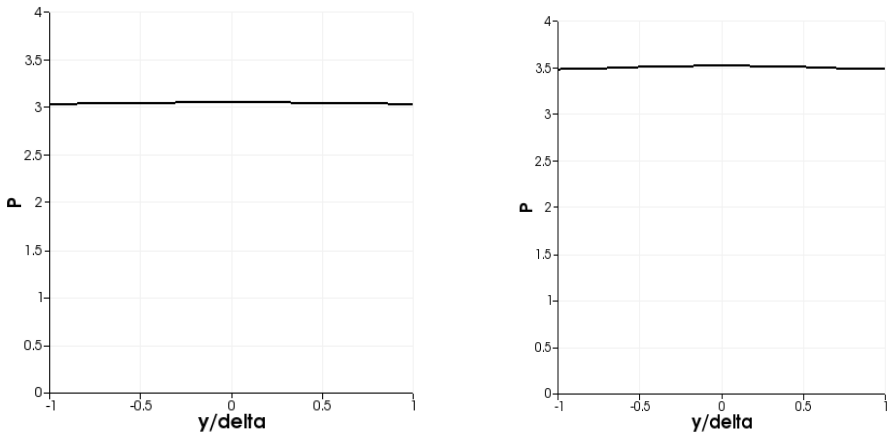


Figure 2: Plots of the normalised velocity (U^+) against the distance from the walls (y^+) for the $\overline{v^2}-f$ (blue) and $\zeta-f$ (red) models, with a full mesh resolution and wall refinement, against the DNS data of Kim et al [4] (black) and the law of the wall (dashed).



(a) **(b)**

Figure 3: Normalised Pressure against the distance from the centreline (y/δ) for the $\overline{v^2}-f$ (a) and $\zeta-f$ (b) models, with a full mesh resolution and wall refinement

The plots of pressure variation for the $\overline{v^2}-f$ and $\zeta-f$ models are shown in Figure 3 – these have not been overlaid with the DNS data of Kim due to the difference in scales. It is clear that both models over-predict the pressure and fail to produce the full variation in pressure found by Kim et al [3].

6.2.3. Full Resolution without Wall Refinement

In the cases without wall refinement the k-ε model was also tested, this model uses wall functions to predict the near wall behaviour, the results for the velocity profiles are shown in Figure 4. The performance of the $\overline{v^2}-f$ and $\zeta-f$ models are very similar to the results with wall refinement, however the mesh only refines down to a y^+ of 2.5. It is worth noting that the mesh is too such a fine resolution, since it is acceptable for use in DNS, that even without specific wall refinement the cells are almost at the correct level of refinement in the boundary layer. The performance of the k-ε model is very interesting, it takes a logarithmic increase in the velocity from the first cell height to a y^+ of 70. This is almost certainly a result of the wall functions, an inspection of the k and epsilon wall functions reveals that they assume the first cell height is $y^+ = 11.6$ from which point onwards a logarithmic function is applied. This somewhat limits the flexibility of the model to adapt to flow conditions such as separation of the boundary layer, this is a common problem with wall functions. Industrial practice is often to fully resolve the boundary layer as this gets a better representation of the actual flow. The performance of the k-ε when compared to the $\overline{v^2}-f$ and $\zeta-f$ suggests that it is able to represent the transition region better in terms of the actual velocity. However it is outperformed by the $\overline{v^2}-f$ and $\zeta-f$ for both near-wall prediction and the profile shape.

The prediction of the pressure variations across the channel are shown in Figure 5 for the k-ε, $\overline{v^2}-f$ and $\zeta-f$. The k-ε heavily over-predicts the pressure magnitude, however none of the models successfully predict the pressure variations. It is worth noting that the $\zeta-f$ pressure magnitude has decreased dramatically from the full wall resolution case, whereas the $\overline{v^2}-f$ has remained at a similar level, suggesting that the $\zeta-f$ may have a higher near-wall mesh sensitivity.

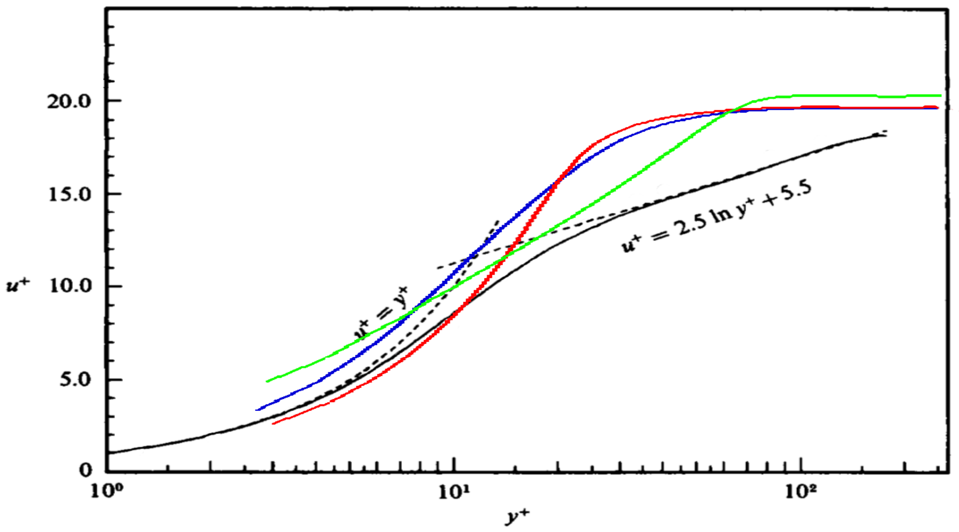
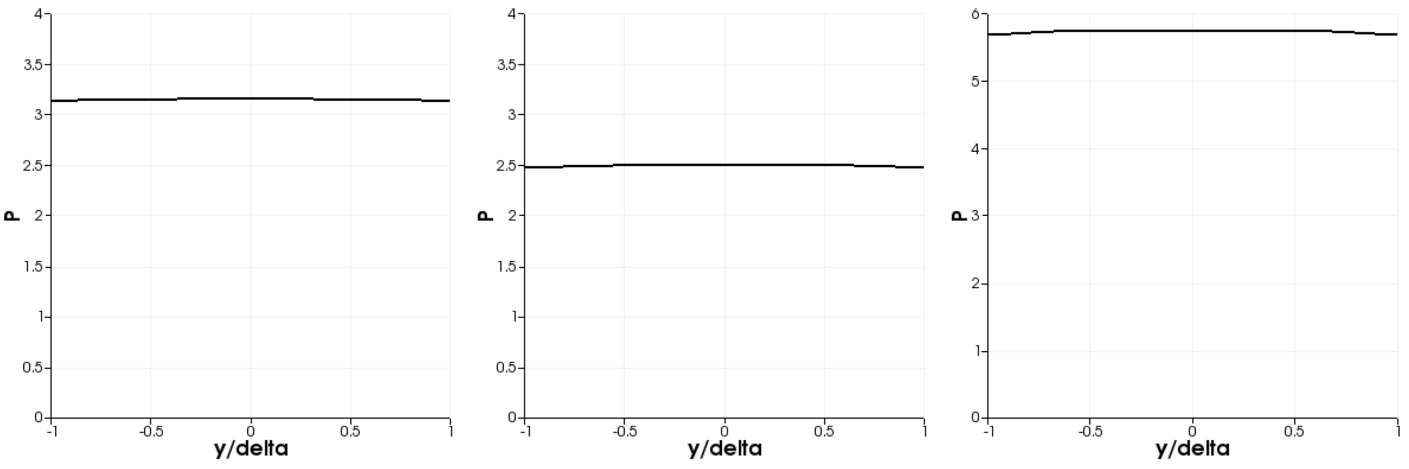


Figure 4: Plots of the normalised velocity (U^+) against the distance from the walls (y^+) for the $\overline{v^2}-f$ (blue), $\zeta-f$ (red) and k-ε (green) models, with a full mesh resolution but no wall refinement, against the DNS data of Kim et al [4] (black) and the law of the wall (dashed).

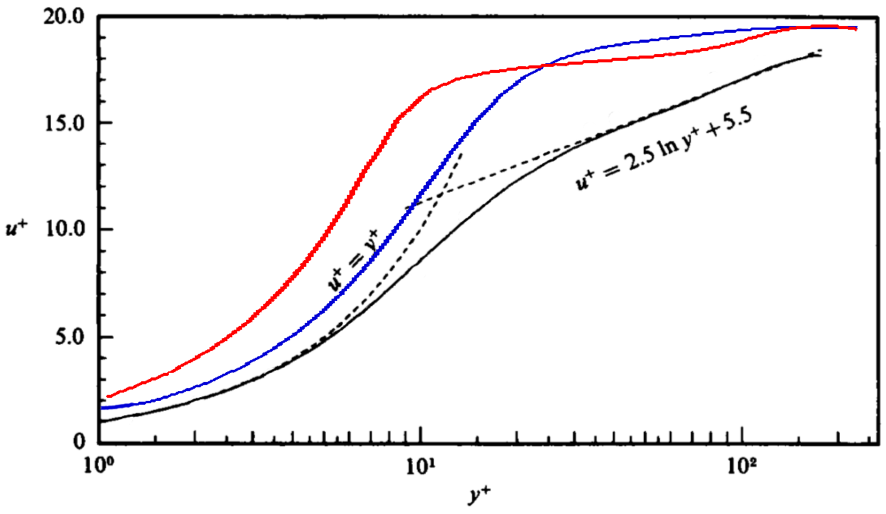


(a) **(b)** **(c)**

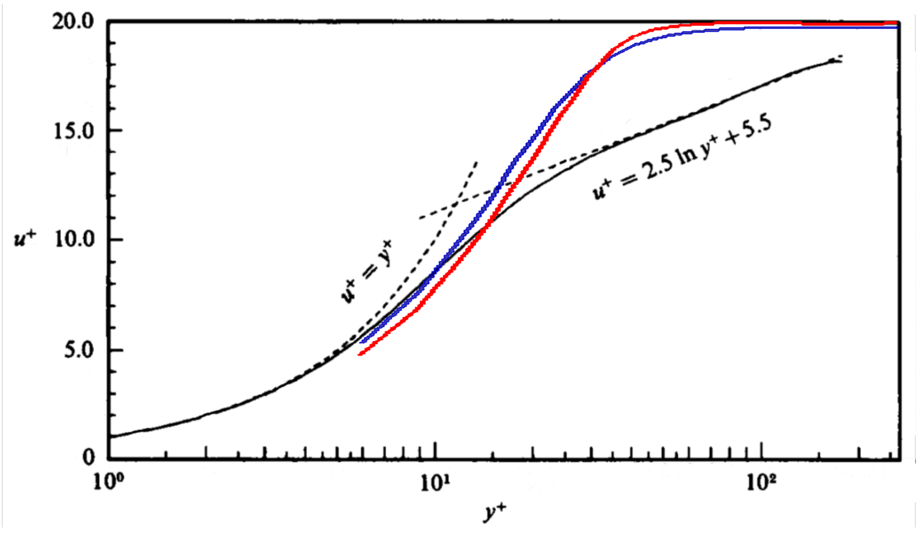
Figure 5: Normalised Pressure against the distance from the centreline (y/δ) for the $\overline{v^2}-f$ (a), $\zeta-f$ (b) and k-ε (c) models, with a full mesh resolution but no wall refinement

6.2.4. Half Resolution

The results for the velocity profiles of the half resolution cases are shown in Figure 6; (a) shows the wall refined cases, (b) shows the unrefined cases. The $\overline{v^2}-f$ model shows a remarkable indifference to the mesh resolution, actually achieving a better prediction for the near-wall velocity in the refined case. Both the $\overline{v^2}-f$ and $\zeta-f$ models have a very good performance on the unrefined case. Although it is worth noting that, if the mesh has high wall refinement, the cells further from the wall are larger and so achieve a worse resolution than without refinement, if the total number of cells remains the same. This might give some explanation to the better performance around the far-wall and transition regions of the $\zeta - f$ model on the unrefined case. It would appear that the $\zeta - f$ model achieves a better representation of the the transition and log-law regions when comparing the full and half resolution cases with wall refinement. It is therefore vital to understand that the primary purpose of validation is to establish the repeatable results of the models. Consequently the half resolution cases are merely to show the sensitivity to the mesh and not to suggest running partially resolved meshes in an attempt to achieve a more accurate prediction; were this to be done without comparisons to data, the result would be entirely unreliable.



(a)



(b)

Figure 6: Plots of the normalised velocity (U$^+$) against the distance from the walls (y$^+$) for the $\overline{v^2} - f$ (blue) and $\zeta - f$ (red) models, at a full mesh resolution with wall refinement (a) and without wall refinement (b), against the DNS data of Kim et al [4] (black) and the law of the wall (dashed).

The results of the pressure distribution for the half resolution cases are shown in Figure 7, (a) and (b) both have wall resolution. It is clear that the $\zeta-f$ pressure distribution drops dramatically in magnitude without wall resolution, whereas the $\overline{v^2}-f$ has only a minor drop. This suggests that the $\zeta-f$ is far more sensitive to the mesh resolution than the $\overline{v^2}-f$. The reason for the drop in pressure is that the velocity drop, and consequent pressure increase, in the boundary layer is less well captured without wall refinement.

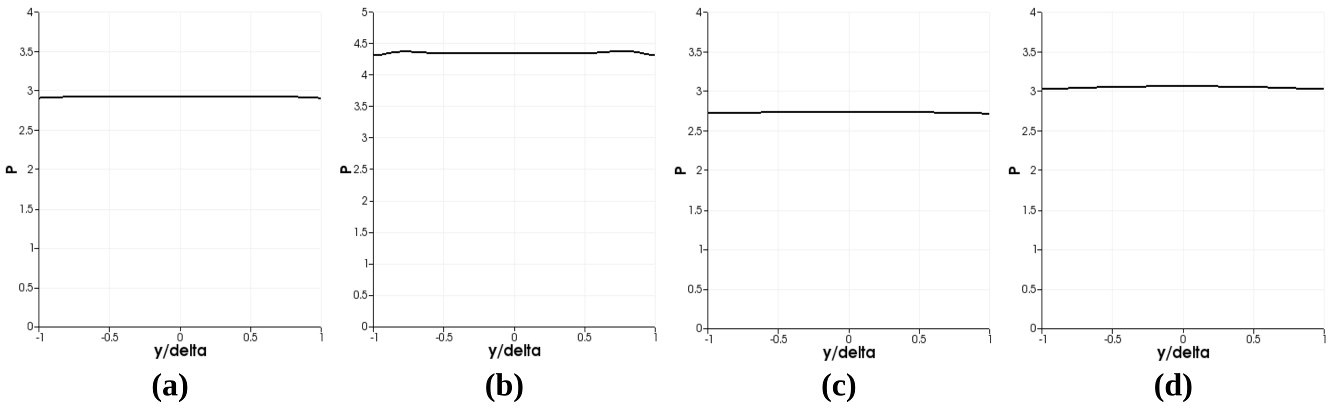
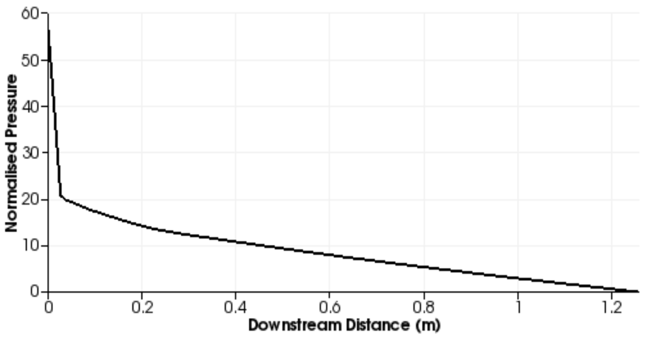


Figure 7: Normalised Pressure against the distance from the centreline (y/δ) for the $\overline{v^2}-f$ (a & c) and $\zeta-f$ (b & d) models with half mesh resolution; (a) & (b) have wall refinement, (c) & (d) do not

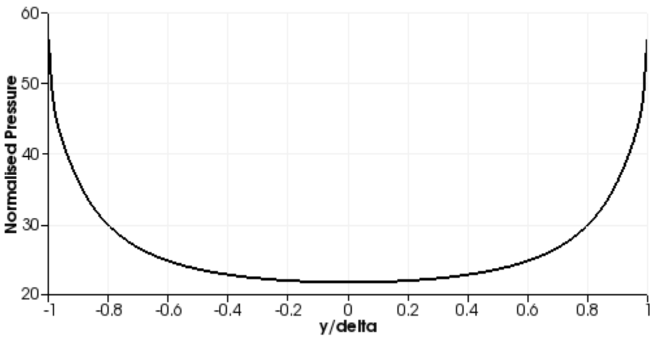
7. Discussion

The results in section six show that the $\overline{v^2}-f$ and $\zeta-f$ have comparable performance; the $\zeta-f$ has a greater mesh sensitivity but a better performance in near-wall regions; the $\overline{v^2}-f$ has a better performance in the log-law region and a lower sensitivity to the mesh. It is interesting that the $\zeta-f$ was originally found by Hanjalik et al [1] to have improvements over the $\overline{v^2}-f$ in mesh sensitivity, yet this implementation is more sensitive than the $\overline{v^2}-f$.

The lack of variation in the pressure profile is surprising, since the velocity fluctuations so closely match the data. From conservation of energy it would naturally suggest that in the low velocity regions the pressure would be higher, however this is not the case. Kim et al [3] use a fully developed profile at the inlet of the channel, it is believed that this may cause some differences in the result. Where the channel develops from the laminar region initially there is a very high pressure which may cause the latter near-wall pressures to drop as they would behind a stagnation point. The initial friction halts the near-wall flow and pushes the main flow towards the centre of the channel, this means that the flow close to the wall has a lower energy and consequently a relatively lower pressure. Analysing the pressure distribution along the wall from the inlet to the outlet, in Figure 8 (a), shows a spike in pressure near the inlet in the laminar region of the boundary layer. A further analysis of the P_{rms} distribution across the channel at the inlet, in Figure 8 (b), also shows a better agreement with the shape of the data from Kim et al [3], although the pressure magnitude is clearly far higher. This further suggests that the inlet conditions or measuring point may be the cause of the discrepancy between the results from this project and the results of Kim et al [3].



(a)



(b)

Figure 8: Pressure distribution against the stream-wise distance from the inlet, along the wall (a). Pressure distribution across the channel at the inlet (b)

The ability of the $\zeta - f$ model to predict the near-wall velocities is truly impressive, with a full mesh resolution and wall refinement the results perfectly match the DNS results. Unfortunately this performance deteriorates after the transition region and the performance of the $\overline{v^2} - f$ is better in the log-law region. Further investigations should investigate the performance of the $\zeta - f$ with a sustained high resolution up to a y^+ of 200, this should be done by decreasing the growth rate of the cells. From the sensitivity to the mesh of $\zeta - f$ that has been found from this project, it is expected that the deviation from the DNS data may be compounded by the reducing resolution of the mesh. Indeed, were this the case then the performance of the $\zeta - f$ model may be able to be improved through the transition and log-law regions.

One other major benefit of the $\zeta - f$ model is that it is valid up to the wall, this avoids the use of wall functions. Although wall functions can offer a cheap and relatively accurate evaluation of the boundary layer flow, they fail to predict complex flow structures which make them unsuitable for use in a large number of cases.

The implemented variation of the $\zeta - f$ model is as stable as the already implemented $\overline{v^2} - f$, it successfully ran without any difficulty on a number of cases with different mesh sizes. However it is worth noting that a backward facing step case and 3D diffuser were attempted, but the models were unable to run successfully – including the k-ε. The reason for this failure on the other validation cases is that the meshes that were used were too high a resolution. The 3D diffuser case of Ryon had a specified mesh resolution that was used for testing the $\overline{v^2} - f$ and $\zeta - f$ on the NASA Glenn-HT compressible CFD code. However when this mesh was replicated and used in OpenFOAM none of the models could run, this could be due to differences in the code or due to over-refinement in some areas of the mesh. Examining the Kolmogorov length-scale for the diffuser case using showed that the wall spacing at the expansion section of the diffuser caused the resolution to drop to smaller that the Kolmogorov length scale. This would cause the RANS simulation to diverge as the code will be attempting to resolve flows smaller than the smallest length-scale. Clearly next to the wall the value of epsilon is different from the rest of the flow, consequently Ryon [7] may have specified conditions near the walls better and been able to successfully run the case. The time constraints on this project resulted in both the diffuser and backward facing step cases being abandoned.

8. Conclusion

The implementation of a new turbulence model in OpenFOAM uses many of the very advanced features of the code; however the mathematical representation which the code allows makes it intuitive to read. After experiencing severe stability issues with the original $\zeta - f$ equation, the same concept that was used to derive the original equation was applied to the implemented $\overline{v^2} - f$ model. This produced a set of very stable and efficient equations that, when validated against the channel flow case by Kim et al, showed better near-wall performance than the $\overline{v^2} - f$ model. The downside to the implemented $\zeta - f$ model was its mesh sensitivity, which is far higher than the $\overline{v^2} - f$ model. Future expansions to this work should be to further validate the model and to tune the constants to account for the difference in the equations. Furthermore, as discussed in section 7, the model should be tested with a higher resolution mesh extending further from the walls, produced with a lower growth rate and more layers. The tuning of the constants and the mesh should see the model improve in performance, certainly its current performance is above the $\overline{v^2} - f$ in some aspects and so improving it would be of particular interest.

The project encountered many aspects of CFD which are common to industry. One such issue was in the remote access and data-transfer rates as well as the need to reduce the computational cost, this led to an investigation into producing low cost input files which had only the required data present. A further result of this was to investigate automating the post-processing and avoiding the use of the GUI to save computational and time costs.

References

[1] Hanjalić, K. Popovac, M. and Hadžiabdić, M. A robust near-wall elliptic-relaxation eddy-viscosity turbulence model for CFD. *International Journal of Heat and Fluid Flow*. 2004, **25**, pp 1047-1051.

[2] Durbin, P. A. Near-wall turbulence closure modelling without damping functions. *Theoretical and Computational Fluid Dynamics*. 1991, **3**, pp 1-13.

[3] Kim, J. Moin, P. and Moser, R. Turbulence Statistics in Fully Developed Channel Flow at Low Reynolds Number. *Journal of Fluid Mechanics*. 1987, **177**, pp 133-166.

[4] Lien F. S. and Kalitzin G. Computations of transonic flow with the v2-f turbulence model. *Int. J. Heat Fluid Flow*. 2001, **22**(53), pp 53-61.

[5] Eckelmann, H. The structure of the viscous sublayer and the adjacent wall region in a turbulent channel flow. *Journal of Fluid Mechanics.* 1974, **65**, pp 439-459.

[6] Willmarth, W W. Pressure fluctuations beneath turbulent boundary layers. *Annual Review of Fluid Mechanics.* 1975, **7**, pp 13-36.

[7] Ryon, J. A. *Computational Simulation of Separated Flow in a Three-Dimensional Diffuser Using* $\overline{v^2}-f$ *and* $\zeta-f$ *models*. Master of Science, Iowa State University, 2008.