UNIVERSIDAD NACIONAL DE CÓRDOBA FACULTAD DE CIENCIAS EXACTAS, FISICAS Y NATURALES

ESCUELA I.M.E – I.M. – T.M.

Análisis de incertezas en el cómputo de flujos turbulentos empleando OpenFOAM®

NOMBRE DEL AUTOR: Giuliano Adrian Carrillo

CARRERA: Ingeniería Mecánica

NOMBRE DEL DIRECTOR DE TESIS: Dr. Ing. Juan Pablo Saldía

INDICE

1. INTRODUCCIÓN 3
1.1 Prefacio 3
2 Modelo Físico 5
2.1 Leyes de conservación de Fluidos en movimiento. 5
2.2 Ecuación de Navier-Stokes para un fluido Newtoniano. 7
2.3 Forma diferencial e integral de la ecuación general de transporte 8
2.4 Modelado de la turbulencia 10
2.5 Efecto de las fluctuaciones turbulentas sobre las propiedades del flujo 11
2.6 Ecuaciones para flujo incompresible de Navier-Stokes Promediadas según Reynolds. 12
3 Modelo Numérico 13
3.1 Método de volúmenes finitos para problemas de difusión-convección. 13
3.2 Algoritmos de Solución para problemas de acoplamiento presión-velocidad en flujos estacionarios 14
3.3 Algoritmo SIMPLE 16
3.4 Cálculo de flujos turbulentos 18
3.5 Modelo k-ε 20
3.6 Modelo k-ω 21
3.7 Otros modelos 21
4 Error e Incertezas en la dinámica de los fluidos computacional 22
4.1 Validación y Verificación 22
4.2 Representación en serie de potencias del error de discretización. 23
4.3 Procedimiento propuesto para la estimación del error. 29
4.4 Estimación de incerteza. 30
5. RESULTADOS 32
5.1 Metodología 32
5.2 OpenFoam 33
5.3 Caso de análisis - *Backward Facing Step* 34
5.4 Configuración numérica 36
5.4.1 Mallado de la geometría 36
5.4.2 Configuración de condiciones iniciales y condiciones de borde 42
5.4.3 Configuración numérica restante 47
5.5 Resultados de simulaciones numéricas. 47
5.6 Discusión de los resultados 69
6.CONCLUSIÓN 71
7. APENDICES 73

Apéndice A – BlockMeshDict –Backward Facing Step 73
APENDICE B – Condiciones iniciales y condiciones de borde..... 75
APENDICE C – Archivos restante de configuración numérica 82
APENDICE D- Código de verificación de la solución 86
8. REFERENCIAS 93

1. INTRODUCCIÓN

1.1 Prefacio

La mecánica de fluidos computacional (CFD, por sus siglas en inglés) constituye una rama de la mecánica de fluidos, que presenta un rol en continuo crecimiento como complemento de ensayos experimentales y/o enfoques analíticos en el estudio de problemas que involucran procesos de tipo fluidodinámico. De particular importancia es el rol que cumple en problemas de diseño aerodinámico, turbomáquinas, transferencia de calor, aerodinámica de automóviles, combustión, bioingeniería, entre otros, en dónde provee una alternativa que reduce el costo en comparación a ensayos experimentales, o que incluso puede permitir el estudio de condiciones difíciles, sino inviables, de reproducir en forma experimental.

OpenFOAM® es un software que provee utilidades y herramientas que abarcan todas las fases de un estudio de CFD, tanto en lo que concierne a la definición geométrica y de mallado, la visualización y manipulación de los resultados, y la implementación de un amplio espectro de modelos físicos y numéricos aplicados a la dinámica de fluidos, principalmente orientado al método de los volúmenes finitos (Ferizger et. al., 2001).

En la actualidad existe un creciente interés en la comunidad de CFD, en la importancia de la verificación y cuantificación de los errores e incertezas inherentes a toda solución numérica provista por un código de cálculo fluidodinámico (Slotnick, 2014). No obstante, su amplia difusión, pocos estudios han sido realizados sobre la cuantificación de estos errores utilizando el software OpenFOAM®*, entre ellos (Deng, 2008).*

En este trabajo, el análisis de incertezas es aplicado al caso *Backward Facing step*, hace referencia a un flujo de fluido incompresible que circula por un canal rectangular, el cual posee un escalón, a partir del cual la sección por donde circula el fluido se ensancha abruptamente, produciéndose un flujo recirculante debido al desprendimiento de la capa límite, producido por el gradiente adverso de presión. La elección de este caso para este trabajo final, se debe a su extensa reproducción por distintos investigadores y entidades reconocidas como N.A.S.A (National Aeronautics and Space Administration) y E.R.C.OF.T.A.C (European Research Community on Flow, Turbulence and Combustión), lo que facilita la obtención de datos para la comparación de simulaciones numéricas, como de datos obtenidos de experimentos de laboratorio.

Para el análisis de incertezas sobre simulaciones CFD, un conjunto total de 30 simulaciones es realizado, para el cual, son empleados distintos tamaños de celdas, esquemas de discretización y modelos de turbulencia. Los rangos de incertezas, son obtenidos, mediante un código, realizado en Python, el cual contiene embebido un procedimiento de verificación de la solución que consta del modelado del error de discretización, mediante series de potencias, del cual se obtienen las variables de los respectivos ajustes utilizando el método de mínimos cuadrados, para luego calcular el valor de incerteza, a través de multiplicar el error por un factor de seguridad, dependiente del rango de datos disponibles, y las desviaciones estándar de los ajustes, según la metodología introducida en (M.Hoekstra, 2014). Los resultados obtenidos son presentados en gráficas, a través de las cuales se observa el comportamiento de convergencia de incertezas al aplicar el método de refinamiento de grilla.

El presente trabajo se estructura partiendo del capítulo número 1 en el cual se encuentra la introducción de este trabajo y un breve resumen de la mecánica de los fluidos computacional. En el capítulo 2 se presenta el modelado físico, donde las ecuaciones que gobiernan a la mecánica de los fluidos y el modelado de la turbulencia son mencionadas. En el capítulo 3, se hace referencia a los conceptos relacionados con el modelado numérico de las ecuaciones mencionadas en el capítulo anterior, particularmente, mediante el método de volúmenes finitos. El capítulo 4, desarrolla conceptos teóricos acerca de la validación, verificación y análisis de incertezas en simulaciones de mecánica de los fluidos computacional. Luego, es encontrado el capítulo 5 en donde se presenta el desarrollo del trabajo junto con los resultados obtenidos, los cuales son discutidos y evaluados para obtener las conclusiones correspondientes que se describen en el capítulo número 6.

2 Modelo Físico

2.1 Leyes de conservación de Fluidos en movimiento

Toda simulación CFD, se conforma de un modelo físico, y el modelo numérico propiamente dicho. Las leyes de conservación, que rigen la mecánica de los fluidos, son la base principal de todo modelo físico de una simulación de dinámica de los fluidos computacional. Estas ecuaciones expresan el principio de conservación de masa, principio de conservación de momento lineal y angular, y el principio de conservación de energía. Las ecuaciones que rigen un flujo de fluido incompresible, newtoniano y laminar, son descritas a continuación.

La primera ecuación a analizar es la ecuación de **conservación de masa**, que surge de escribir el balance de masa para un volumen de fluido, el cual, en palabras, equivale a decir, que la variación de masa dentro de un volumen o elemento fluido, es igual al flujo de masa a través de los límites del mismo. Realizando dicho balance a través de un elemento de fluido, se llega a la expresión:

$$\frac{\partial \rho}{\partial t} + div(\rho . \bar{V}) = 0 \tag{2.1}$$

La expresión anterior equivale al principio de conservación de masa tridimensional, valido para flujo inestacionario, y flujo compresible, también conocida como ecuación de continuidad, donde ρ es la densidad, t es el tiempo, y $\bar{V}$ el vector velocidad . El primer término describe la variación temporal de masa por unidad de volumen, mientras que el segundo describe el flujo neto de masa, por unidad de volumen, que atraviesa los límites del volumen de un elemento de fluido. Suponiendo flujo incompresible, la ecuación de continuidad, puede ser expresada como:

$$div(\bar{V}) = 0 \tag{2.2}$$

El segundo principio a describir, es el **principio de conservación de momento** aplicado a un volumen de control fijo en el espacio, el cual enuncia que la variación de cantidad de movimiento por unidad de volumen, de un elemento fluido, es igual a la suma de fuerzas exteriores que actúan sobre el volumen de dicho elemento. La velocidad de incremento de momento en las tres direcciones del espacio, por unidad de volumen de una partícula de fluido se representan por:

$$\rho \frac{Du}{Dt} \quad ; \quad \rho \frac{Dv}{Dt} \quad ; \quad \rho \frac{Dw}{Dt} \tag{2.3}$$

Donde u, v, y w se corresponden con las componentes cartesianas de la velocidad y $\frac{DA}{Dt}$, es la derivada convectiva. Ahora bien, las fuerzas que actúan sobre una partícula fluida, pueden dividirse en **fuerzas de superficies** (fuerzas de presión, fuerzas viscosas) y **fuerzas másicas** (fuerzas electromagnéticas, fuerzas gravitacionales). La presión, una tensión normal se denota por P, mientras que las tensiones viscosas son denotadas por τ. La notación τ_{ij} es aplicada para indicar la dirección de las tensiones viscosas, los sub índices i y j indican que la componente de tensión actúa en la dirección j sobre una superficie normal a la dirección i. Si se consideran las fuerzas másicas por separado, contenidas en un término fuente y se realiza la correspondiente suma de fuerzas en el espacio para igualarla a la la variación de momento por unidad de volumen en cada una de las tres direcciones, se obtienen las ecuaciones de conservación de momento para un elemento de fluido. Estas ecuaciones, particularizadas, para flujo incompresible, se muestran a continuación:

$$\rho \frac{Du}{Dt} = \rho . \frac{\partial u}{\partial t} + \rho . div(u . \bar{V}) = \frac{\partial(-P + \tau_{xx})}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + S_x \tag{2.4}$$

$$\rho \frac{Dv}{Dt} = \frac{\partial(-P + \tau_{yy})}{\partial y} + \frac{\partial \tau_{yx}}{\partial x} + \frac{\partial \tau_{zy}}{\partial z} + S_y \tag{2.5}$$

$$\rho \frac{Dw}{Dt} = \frac{\partial(-P + \tau_{zz})}{\partial z} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{xz}}{\partial x} + S_z \tag{2.6}$$

La ecuación de conservación restante, es la **ecuación de conservación de energía**, derivada de la primera ley de la termodinámica, en donde se establece que, el calor suministrado a un sistema, es igual al incremento de energía total del sistema, más el trabajo mecánico realizado por el mismo. Realizando un análisis sobre el trabajo producido por las fuerzas de superficie sobre el volumen de un elemento fluido, la adición de calor, y operando algebraicamente, se llega a la ecuación de conservación de la energía, aplicada para un flujo incompresible:

$$\rho \frac{DE}{Dt} = -div(P.\bar{V}) + \left[\frac{\partial(u.\tau_{xx})}{\partial x} + \frac{\partial(u.\tau_{yx})}{\partial y} + \frac{\partial(u.\tau_{zx})}{\partial z} + \frac{\partial(v.\tau_{xy})}{\partial x} + \frac{\partial(v.\tau_{yy})}{\partial y} + \frac{\partial(v.\tau_{zy})}{\partial z} + \frac{\partial(w.\tau_{xz})}{\partial x} + \frac{\partial(w.\tau_{yz})}{\partial y} + \frac{\partial(w.\tau_{zz})}{\partial z}\right] + div\big(k.grad(T)\big) + S_E \quad (2.7)$$

Donde E denota la energía total, T la temperatura y k, la conductividad térmica.

2.2 Ecuación de Navier-Stokes para un fluido Newtoniano

Las ecuaciones de Navier-Stokes se obtienen a partir de la ecuación de conservación de momento, y son aquellas que describen el movimiento de un fluido newtoniano. Estas ecuaciones, en conjunto con la ecuación de continuidad, determinan la presión y las componentes de la velocidad en todo el campo de movimiento. Las ecuaciones de Navier-Stokes (W.Malalasekera, 2007), conforman la estructura principal, del modelo físico de toda simulación de CFD.

Una de las principales incógnitas en la ecuación de conservación de momento aplicada a un elemento de fluido, son las tensiones viscosas. La ecuación de Navier-Stokes, surge de expresar a estas tensiones, en función de la tasa de deformación lineal y volumétrica del elemento fluido. Entonces, considerando un fluido newtoniano, en donde la variación de las tensiones viscosas con la tasa de deformación es proporcional a través de la viscosidad dinámica, aplicando la hipótesis de Stokes, considerando un fluido isotérmico e isotropía, y operando algebraicamente a través de la ecuación de conservación de momento, se obtienen las ecuaciones de Navier-Stokes, para un fluido newtoniano e incompresible:

$$\rho \frac{Du}{Dt} = -\frac{\partial P}{\partial x} + div(\mu.grad(u)) + \left[\mu \frac{1}{3} \frac{\partial}{\partial x}(div(\bar{V}))\right] + S_x \quad (2.8)$$

$$\rho \frac{Dv}{Dt} = -\frac{\partial P}{\partial y} + div(\mu.\mathrm{grad}(v)) + \left[\mu \frac{1}{3} \frac{\partial}{\partial y}(div(\bar{V}))\right] + S_y \quad (2.9)$$

$$\rho \frac{Dw}{Dt} = -\frac{\partial P}{\partial z} + div(\mu.\mathrm{grad}(w)) + \left[\mu \frac{1}{3} \frac{\partial}{\partial z}(div(\bar{V}))\right] + S_z \quad (2.10)$$

Donde μ denota la viscosidad dinámica. Estas expresiones equivalen a las componentes escalares de la ecuación de Navier-Stokes, que en forma vectorial se puede escribir como:

$$\rho \frac{D\bar{V}}{Dt} = -\text{grad}(\bar{P}) + \mu.\bar{\nabla}^2(\bar{V}) + \mu \frac{1}{3}\frac{\partial}{\partial z}(\text{grad}(div(\bar{V}))) + S_z \tag{2.11}$$

El primer término de la derecha de 2.11 corresponde al término de las fuerzas de presiones, el segundo a las fuerzas viscosas tangenciales, el tercero fuerzas viscosas volumétricas y el ultimo es el término fuente que contiene las fuerzas másicas. Incorporando el término de fuerzas viscosas por deformación volumétrica, en el término fuente, se encuentra la forma más útil de aplicar las ecuaciones de Navier Stokes, a códigos CFD.

$$\rho \frac{Du}{Dt} = -\frac{\partial P}{\partial x} + div(\mu.\text{grad}(u)) + S'_x \tag{2.12}$$

$$\rho \frac{Dv}{Dt} = -\frac{\partial P}{\partial y} + div(\mu.\text{grad}(v)) + S'_y \tag{1.13}$$

$$\rho \frac{Dw}{Dt} = -\frac{\partial P}{\partial z} + div(\mu.\text{grad}(w)) + S'_z \tag{2.14}$$

2.3 Forma diferencial e integral de la ecuación general de transporte

El modelo físico, para un flujo considerado incompresible, queda definido matemáticamente por el siguiente conjunto de ecuaciones diferenciales parciales:

Continuidad $$div(\bar{V}) = 0 \tag{2.15}$$

Momento – x $$\rho \frac{\partial u}{\partial t} + \rho.div(u.\bar{V}) = -\frac{\partial P}{\partial x} + div(\mu.\text{grad}(u)) + S'_x \tag{2.16}$$

Momento – y $$\rho \frac{\partial v}{\partial t} + \rho.div(v.\bar{V}) = -\frac{\partial P}{\partial y} + div(\mu.\text{grad}(v)) + S'_y \tag{2.17}$$

Momento – z $$\rho \frac{\partial w}{\partial t} + \rho.div(w.\bar{V}) = -\frac{\partial P}{\partial z} + div(\mu.\text{grad}(w)) + S'_z \tag{2.18}$$

Energía $$\rho\frac{\partial i}{\partial t}+\rho.div(i.\bar{V})=-p.div(\bar{V})+div\big(k.grad(T)\big)+\emptyset+S_i \quad (2.19)$$

Si en las ecuaciones de momento, 2.16-2.18, el gradiente de presiones, es colocado dentro del término fuente, y en la ecuación de la energía 2.19, la misma tarea es realizada para el termino de trabajo por parte de las presiones, una similitud en la forma matemática del sistema de ecuaciones anterior, es encontrada. Esta similitud, permite, generalizar al sistema anterior, con una sola ecuación, mediante el remplazo de cualquiera de las variables de transporte, por una variable genérica Φ. La ecuación resultante, es denominada **ecuación general de transporte**, y representa los principales procesos de transporte de un flujo de fluidos. Esta ecuación, expresa que la variación temporal de la variable de transporte, más el transporte debido a la parte convectiva, es igual al transporte por difusión de la variable genérica Φ más un término fuente.

$$\rho\frac{\partial \Phi}{\partial t}+\rho.div(\Phi.\bar{V})=div(\Gamma.\mathrm{grad}(\Phi))+S_\Phi \quad (2.20)$$

El punto de partida de todo procedimiento computacional que utilice el método de volúmenes finitos, se basa en la integración de la ecuación anterior (W.Malalasekera, 2007). Entonces, integrando a través del volumen del dominio, y aplicando el teorema de la divergencia de Gauss a los términos convectivos y difusivos, se llega a la expresión:

$$\int_{CV}\rho\frac{\partial \Phi}{\partial t}dV+\int_A n.\rho.(\Phi.\bar{V})dA=\int_A n.(\Gamma.\mathrm{grad}(\Phi))dA+\int_{CV}S_\Phi dV \quad (2.21)$$

La expresión 2.21, expresa que el incremento, de la variable de transporte debido a la variación temporal por unidad de volumen, más el incremento debido al flujo convectivo a través de la superficie del volumen, es igual al incremento producido por el flujo difusivo a través de los límites del volumen, más el incremento de la variable transportada debido al término fuente.

2.4 Modelado de la turbulencia

El presente trabajo, busca realizar un análisis de incertezas sobre flujos turbulentos incompresibles, por lo cual, el modelo físico debe contener los aspectos relacionados con el modelado de la turbulencia.

El número de Reynolds (UL/v donde U y L son la velocidad y la longitud característica y v es la viscosidad cinemática), indica la relación entre las fuerzas inerciales y las fuerzas viscosas de un flujo de fluidos. Para valores pequeños de este parámetro adimensional, el flujo se muestra ordenado, con capas identificables y sin transferencia transversal de partículas en la dirección transversal a la dirección principal del flujo, lo cual se conoce como flujo laminar. A medida que el número de Reynolds aumenta, hasta un valor conocido como Reynolds crítico, el flujo se vuelve inestable, es decir, que, ante cualquier tipo de perturbación, la dinámica del flujo cambia de régimen, pasando a un estado caótico y aleatorio, en donde las propiedades del flujo varían localmente de manera inestacionaria y tridimensional, aun cuando se tienen flujos bidimensionales, lo cual se conoce como flujo turbulento. La turbulencia, que se genera para valores de Reynolds superior al Reynolds crítico, nace debido a un estado de inestabilidad, producido por la incapacidad de las fuerzas viscosas de contener el movimiento aleatorio, impulsado por las fuerzas inerciales sobre las partículas fluidos, obteniendo como consecuencia, estructuras de remolinos, de diferentes escalas de longitud y de tiempo, que aumentan la transferencia de masa, en todas las direcciones del flujo, y a su vez, de las demás variables de transporte.

La ecuación de Navier-Stokes 2.11, es válida para flujo laminar, debido a que solo tiene en consideración, los efectos de las tensiones viscosas debido a la difusión molecular. Para flujo turbulento, la ecuación de Navier-Stokes, sufre modificaciones, por la aparición, de nuevas tensiones asociada al transporte adicional de momento, producido por los remolinos turbulentos, en la dirección transversal al flujo.

La velocidad inestacionaria se descompone en un valor medio estacionario U con una componente de fluctuación u'(t), de modo que se puede escribir a la velocidad como u(t)=U+ u'(t). esta descomposición se denomina **descomposición de Reynolds.** Un flujo turbulento puede ahora ser caracterizado en términos de valores medio de las propiedades del flujo (u,v,w,P,etc) y algunas propiedades estadísticas de sus fluctuaciones (u',v',w'p',etc).

2.5 Efecto de las fluctuaciones turbulentas sobre las propiedades del flujo

Al considerar un volumen de control, el cual es atravesado por un flujo turbulento con un gradiente de velocidad media en la dirección y, la presencia de vórtices o remolinos crea una fuerte mezcla entre corrientes aleatorias que se asocia con el pasaje de porciones de fluido atreves de los límites del volumen de control. Este movimiento de fluido, si bien no puede crear o destruir masa, genera un transporte adicional de momento y energía en el volumen de control. Debido a la existencia de un gradiente de velocidad, el transporte macroscópico de masa en la dirección vertical, mezclara partes del flujo con distintas velocidades, generando un mecanismo adicional, al mecanismo de transporte entre laminas visto para el flujo laminar, que acelerara las regiones más lentas del flujo, y frenara la más rápida, adquiriendo un perfil de velocidad más uniforme. Este cambio de momento adicional, como es de esperarse, genera tensiones de corte adicionales debido a los remolinos de la turbulencia, y dichas tensiones son conocidas como **tensiones de Reynolds**. Estas tensiones, como se indican en el apéndice C, se expresan:

$$\frac{dF}{dA} = \tau_{reynolds} = \rho\overline{v'u'} \tag{2.22}$$

El momento segundo de las componentes fluctuantes de la velocidad (W.Malalasekera, 2007), es distinto de cero, aunque la media individual de cada componente sea nula, debido a que existe una correlación entre los valores de las componentes de la velocidad, debido a la estructura de vórtices que posee la turbulencia.

Estos esfuerzos, suelen modelarse en términos del gradiente de velocidad promedio, haciendo una analogía con la ley de newton de la viscosidad, e introduciendo una variable matemática, conocida como **viscosidad de remolino** o **viscosidad turbulenta**, que es una propiedad del modelo, y no del fluido:

$$\tau_{reynolds} = \rho\overline{v'u'} = \mu_t \frac{\partial \bar{u}}{\partial y} \tag{2.23}$$

2.6 Ecuaciones para flujo incompresible de Navier-Stokes promediadas según Reynolds.

La turbulencia, como fue mencionado anteriormente, obliga a realizar modificaciones sobre la ecuación de Navier-Stokes, debido al incremento adicional de transporte producido por los remolinos turbulentos. Este incremento adicional se refleja en el sistema de ecuaciones al utilizar la descomposición de Reynolds, en donde las componentes fluctuantes de la velocidad, agregan un término en el miembro derecho, que representa el efecto de transporte producido por las tensiones de Reynolds. Operando, como se indica en el apéndice D, se obtienen las ecuaciones de Reynolds-Navier Stokes:

$$\frac{\partial U}{\partial t} + div(U\bar{U}) = -\frac{1}{\rho}\frac{\partial P}{\partial x} + \boldsymbol{\nu}.div\big(grad(U)\big) + \frac{1}{\rho}\left[\frac{\partial(-\rho\overline{u'^2})}{\partial x} + \frac{\partial(-\rho\overline{u'v'})}{\partial y} + \frac{\partial(-\rho\overline{u'w'})}{\partial z}\right] \qquad (2.24)$$

$$\frac{\partial V}{\partial t} + div(V\bar{U}) = -\frac{1}{\rho}\frac{\partial P}{\partial y} + \boldsymbol{\nu}.div\big(grad(V)\big) + \frac{1}{\rho}\left[\frac{\partial(-\rho\overline{u'v'})}{\partial x} + \frac{\partial(-\rho\overline{v'^2})}{\partial y} + \frac{\partial(-\rho\overline{v'w'})}{\partial z}\right] \qquad (2.25)$$

$$\frac{\partial W}{\partial t} + div(W\bar{U}) = -\frac{1}{\rho}\frac{\partial P}{\partial z} + \boldsymbol{\nu}.div\big(grad(W)\big) + \frac{1}{\rho}\left[\frac{\partial(-\rho\overline{u'w'})}{\partial x} + \frac{\partial(-\rho\overline{w'v'})}{\partial y} + \frac{\partial(-\rho\overline{w'^2})}{\partial z}\right] \qquad (2.26)$$

El resultado, deriva en las tensiones de Reynolds, que constan de tres tensiones normales adicionales, y seis tensiones tangenciales adicionales, producidas por los remolinos turbulentos. Estas tensiones suelen ser grandes en comparación con las tensiones viscosas que dominan el flujo laminar. Estas ecuaciones, conforman un sistema de ecuaciones denominadas Ecuaciones de **Navier -Stokes- promediados según Reynolds**, o mejor conocidas como **R.A.N.S** (por sus siglas en ingles *Reynolds – averaged – Navier – Stokes equations*).

3 Modelo Numérico

3.1 Método de volúmenes finitos para problemas de difusión-convección

El modelado numérico, es una técnica en donde se aplican distintos métodos que permiten acondicionar las ecuaciones del modelo físico (por lo general ecuaciones diferenciales parciales), para conformar un sistema de ecuaciones algebraicas cuya solución, pueda ser obtenida mediante procesos iterativos, con un nivel adecuado, de equilibrio entre precisión de la solución, el costo temporal y el costo computacional de los resultados (W.Malalasekera, 2007).

El método de volúmenes finitos, es una formulación de diferencia finitas, que conforma una de las técnicas de modelado numérico más utilizadas en la actualidad. La aplicación del método de volúmenes finitos, se fundamenta a partir de la integración de la ecuación general de transporte Eq 2.20. Esta integración, se realiza en un dominio discreto en el espacio, por lo cual, el primer paso a realizar es la discretización del dominio, o bien, la **generación de la malla**, que consta de dividir el dominio en volúmenes de control discretos. Cada volumen de control, que conforma el dominio, contiene un punto nodal en su interior, en donde se definen las variables, y posee límites definidos, donde se valúan los flujos convectivos y difusivos de la ecuación de transporte.

El siguiente paso a la generación de grilla, es la **integración de la ecuación de transporte** sobre cada uno de los volúmenes de control obtenidos de la discretización. Para cada volumen de control, se obtiene una ecuación discretizada, por lo cual, aplicada a todo el dominio, se obtiene un sistema de ecuaciones algebraicas, cuya cantidad de términos, depende de la dimensión del flujo. En estas ecuaciones, las incógnitas están dadas por el valor de la variable, en el nodo de cada volumen de control y sus nodos vecinos adyacentes.

Distintos esquemas de discretización, son utilizados, para estimar el valor de los coeficientes que acompañan a los términos difusivos y convectivos, ya que deben ser valuados en las caras de los volúmenes de control para representar los flujos de transporte, y como las variables se almacenan, en los nodos centrados en los volúmenes, los valores estimados, quedan escritos en función, de los valores de las variables de transporte almacenados en los nodos que comparten el área donde debe ser realizada la estimación. Existen distintos esquemas de discretización, con distintos órdenes de precisión en función del truncamiento de la serie de Taylor, donde

algunos de los más conocidos en los softwares de mecánica de los fluidos computacional, son los esquemas de diferenciación central, *upwind* o *QUICK*. La utilización de distintos esquemas de discretización, depende de la fuerza relativa entre la convección y la difusión, ya que, la convección, afecta al transporte de las propiedades del flujo en la dirección del mismo, a diferencia de la difusión, en donde su gradiente se distribuye de la misma manera en todas las direcciones.

A modo de ejemplificación, una de las ecuaciones que forman parte del sistema de ecuaciones algebraico obtenido luego de la discretización, para un nodo genérico P, en un dominio unidimensional, rodeado por nodos E y W, adquiere la siguiente forma:

$$a_P \Phi_P = a_W \Phi_W + a_E \Phi_E + S_\Phi . \Delta V \qquad (3.1)$$

Donde los coeficientes a_W, a_P , y a_E son función del espaciamiento entre puntos nodales, coeficientes de transporte, y el área que limita a cada uno de los volúmenes de control.

El último paso en el método de volúmenes finitos, consta de la **solución del sistema de ecuaciones algebraicas lineales** planteada en el punto anterior, para obtener la distribución de la propiedad de transporte Φ en todo el dominio discretizado. Esta resolución, suele ser llevada a cabo por medio de métodos iterativos, tales como el método de Gauss-Seidel.

3.2 Algoritmos de Solución para problemas de acoplamiento presión-velocidad en flujos estacionarios

Si se conoce el gradiente de presión, el proceso de obtención de ecuaciones discretizadas para las velocidades a partir de las ecuaciones de momento es exactamente el mismo que para cualquier otro escalar. Este es el caso, en el cual se propuso la integración de los términos de presión dentro del término fuente, sin embargo, en la mayoría de los problemas, la distribución de presiones debe surgir como parte de la solución, y es aquí, donde la metodología anterior debe sufrir ciertas modificaciones, debido al acoplamiento que se produce en las ecuaciones de momento entre la presión y la velocidad. El problema de acoplamiento entre estas variables, puede resolverse con la aplicación de estrategias iterativas como el método **SIMPLE**. En este algoritmo, los flujos convectivos por unidad de masa a través de las caras de las celdas se

evalúan a partir de los denominados componentes de velocidad estimados. Además, un campo de presión estimado se utiliza para resolver las ecuaciones de momento, y una ecuación de corrección de presión, deducida de la ecuación de continuidad, se resuelve para obtener un campo de corrección de presión, que a su vez se utiliza para actualizar los campos de velocidad y presión. Para iniciar el proceso de iteración se utilizan suposiciones iniciales para los campos de velocidad y presión. A medida que el algoritmo avanza, el objetivo debe ser progresivamente mejorar estos campos estimados. El proceso se itera hasta la convergencia de los campos de velocidad y presión, al obtener un valor de residuo entre cada iteración definido antes del comienzo del proceso.

El método del volumen finito para este tipo de problemas comienza, como siempre, con la discretización del dominio y de las ecuaciones de transporte relevantes. Primero hay que decidir dónde almacenar las velocidades. Parece lógico definirlas en los mismos lugares que las variables escalares tales como presión, temperatura, etc. Sin embargo, si las velocidades y las presiones son ambas definidas en los nodos de un volumen de control ordinario, un campo de presión altamente no uniforme puede actuar como un campo uniforme en las ecuaciones discretizadas de momento. Es decir, el hecho de almacenar las velocidades en los mismos puntos nodales que la presión, deriva en que la discretización del término de gradientes de presiones, no permita introducir los efectos de un campo altamente no uniforme, en la ecuación de momento. Es por esto, que se utiliza lo que se conoce como grilla escalonada para las componentes de la velocidad, en donde se almacenan todas las variables escalares en puntos nodales ordinarios, mientras que la velocidad es calculada en grilla escalonadas centradas alrededor de las caras de los volúmenes de control. Como consecuencia de esta nueva disposición de grilla, se utiliza una nueva notación, en donde las líneas que interceptan los nodos de presión se definen mediante las letras mayúsculas I,J (caso bidimensional) y las líneas que interceptan los nodos de velocidad, mediante las letras minúsculas i,j. Con esta nueva notación, la ecuación de momento en x discretizada para la velocidad en la posición i,J queda dado por

$$a_{i,J}u_{i,J} = \sum a_{nb}\,u_{nb} - \frac{p_{I,J} - p_{I-1,J}}{\delta x}\Delta V + S.\Delta V \tag{3.2}$$

O

$$a_{i,J}u_{i,J} = \sum a_{nb}\, u_{nb} + (p_{I-1,J} - p_{I,J})A_{i,J} + b_{i,J} \tag{3.3}$$

Donde ΔV es el volumen de cada volumen de control. $b_{i,J} = S.\Delta V$= es el término fuente de la ecuación de momento, $A_{i,J}$ es el área de la cara (este u oeste) del volumen u-control. El término fuente del gradiente de presión, ha sido discretizado por medio de una interpolación lineal entre los nodos de presión en los límites de volumen de control de la velocidad. En el nuevo sistema de numeración los nodos vecinos E, W, N y S implicados en la suma $\Sigma a_{nb}u_{nb}$ son (i-1, J), (i +1, J), (i, J-1) y (i, J + 1).

3.3 Algoritmo SIMPLE

Este algoritmo para la solución de problemas de acoplamiento presión-velocidad en flujos estacionarios, es esencialmente un procedimiento de estimar y corregir la presión sobre la disposición de grilla escalonada presentada anteriormente.

Para iniciar el proceso de cálculo SIMPLE se supone un campo de presión p*. Las ecuaciones de momento discretizadas se resuelven usando el campo de presión supuesto, para obtener componentes de velocidad u* y v*, dando lugar a la expresión (3.3) (para la dirección x).

$$a_{i,J}u_{i,J}{}^{*} = \sum a_{nb}\, u_{nb}{}^{*} + (p_{I-1,J}{}^{*} - p_{I,J}{}^{*})A_{i,J} + b_{i,J} \tag{3.4}$$

Luego, la corrección de presión p′, es definida como la diferencia entre el campo de presión real p, y el campo de presión supuesto p*:

$$p = p^{*} + p' \tag{3.5}$$

De manera similar, las correcciones de las componentes de la velocidad u′ y v′ son definidas:

$$u = u^{*} + u' \tag{3.6}$$

$$v = v^{*} + v \tag{3.7}$$

La sustitución del campo de presión real p, en las ecuaciones de momento, produce el campo de velocidad real (u, v). Restando las ecuaciones de momento discretizadas, de los campos de momento y velocidad reales, por las ecuaciones de momento con las variables estimadas, se obtiene la expresión (3.34).

$$a_{i,J}\left(u_{i,J} - u_{i,J}^{*}\right) = \sum a_{nb}\left(u_{nb} - u_{nb}^{*}\right) + \left[\left(p_{I-1,J} - p_{I-1,J}^{*}\right) - \left(p_{I,J} - p_{I,J}^{*}\right)\right]A_{i,J} + b_{i,J} \qquad (3.8)$$

Remplazando las fórmulas de corrección:

$$a_{i,J}u_{i,J}' = \sum a_{nb}\,u_{nb}' + \left(p_{I-1,J}' - p_{I,J}'\right)A_{i,J} + b_{i,J} \qquad (3.9)$$

En este punto se introduce una aproximación: se eliminan los términos $\Sigma a_{nb}u'_{nb}$ y $\Sigma a_{nb}v'_{nb}$ para simplificar las ecuaciones de las correcciones de velocidad. La omisión de estos términos es la principal aproximación del algoritmo SIMPLE. Se obtiene:

$$u_{i,J}' = \left(p_{I-1,J}' - p_{I,J}'\right)d_{i,J} \qquad (3.10)$$

Con $d_{i,J} = A_{i,J}/a_{i,J}$. Estas ecuaciones describen las correcciones que debe realizarse a cada componente de la velocidad durante cada iteración, por lo que se puede escribir:

$$u_{i,J} = u_{i,J}^{*} + \left(p_{I-1,J}' - p_{I,J}'\right)d_{i,J} \qquad (3.11)$$

La continuidad se satisface en forma discretizada para un volumen de control escalar mediante la siguiente expresión:

$$\left[(\rho uA)_{i+1,j} - (\rho uA)_{i,J}\right] + \left[(\rho uA)_{I,j+1} - (\rho uA)_{I,J}\right] = 0 \qquad (3.12)$$

Sustituyendo las ecuaciones obtenidas para las velocidades, y reordenando, se obtiene una ecuación, que permite obtener las ecuaciones corregidas, y se expresa como:

$$a_{I,J}p'_{I,J} = a_{I+1,J}p'_{I+1,J} + a_{I-1,J}p'_{I-1,J} + a_{I,J+1}p'_{J+1} + a_{I,J-1}p'_{I,J-1} + b'_{I,J} \qquad (3.13)$$

Donde los coeficientes son función de la velocidad, el área y la densidad. La ecuación anterior representa la ecuación de continuidad discretizada, como una ecuación para la corrección de presión p'. Resolviendo esta ecuación, el campo de corrección de presión p' puede obtenerse en todos los nodos escalares. Una vez que se conoce el campo de corrección de presión, se puede obtener el campo de presión real utilizando la fórmula $p = p^* + p'$. La omisión de términos tales como $\Sigma a_{nb} u'_{nb}$ en la derivación no afecta a la solución final porque las correcciones de presión y de velocidad serán todas cero en una solución convergente, dando p *=p, u*=u y v*=v.

El procedimiento comienza resolviendo las ecuaciones discretizadas de momento, remplazando los valores de presión estimada, y obteniendo los valores de velocidad estimados. Estos son remplazados en la ecuación de continuidad discretizada para obtener los valores de corrección de presión, y obtener los campos corregidos de presión y velocidad. Por ultimo estos valores son remplazados en la ecuación general de transporte para obtener el resto de las variables transportadas. Si la convergencia es alcanzada, el procedimiento se da por terminado, de otra manera, los valores calculados para la presión y velocidad, son asignados como nuevos valores estimados, para volver a repetir el proceso, hasta alcanzar la convergencia.

Existen otros algoritmos de resolución reconocidos, tal como el SIMPLER, o el PISO, los cuales poseen una lógica de resolución general, incorporando pasos predictores y correctores adicionales para la presión, durante una misma iteración.

3.4 Cálculo de flujos turbulentos

Para muchas aplicaciones de ingeniería, no es necesario resolver en detalle las fluctuaciones turbulentas, sino, que basta con obtener información acerca de las propiedades del flujo promediadas en el tiempo. Por lo tanto, gran cantidad de cálculos de flujos turbulentos, incluyendo este trabajo, son aplicados mediante cálculos basados en el método **Reynolds-averaged-Navier-Stokes (R.A.N.S)**. Los modelos RA.N.S, resuelven las características del

flujo a través de las propiedades medias del mismo, utilizando distintos métodos para modelar las tensiones de Reynolds, que surgen en la ecuación de Navier-Stokes a causa de la turbulencia.

Con el fin de ser capaz, de calcular flujos turbulentos con las ecuaciones RANS, es necesario desarrollar **modelos turbulentos**, para predecir las tensiones de Reynolds y los términos escalares de transporte. Esto permite cerrar el sistema de ecuaciones con las incógnitas adicionales. Los modelos RANS de turbulencia, se clasifican en base al número adicional de ecuaciones de transporte que son necesarias para cerrar el sistema, por ejemplo, dos de los modelos utilizados en este trabajo, como son el modelo k-ε y el modelo k-ω, adicionan dos ecuaciones de transporte al sistema general.

La mayoría de los modelos turbulentos RANS, se basan en suponer que existe una analogía entre la acción de las tensiones viscosas y las tensiones de Reynolds. Es sabido que la turbulencia disminuye, a menos que exista una cierta tasa de deformación en el flujo, por lo tanto, las tensiones turbulentas, se incrementan a medida que se incrementa la tasa de deformación del flujo medio ya que aumenta el estiramiento de vórtices. En base a esto, las tensiones de Reynolds pueden ser aproximadas como proporcionales a la tasa de deformación media, dando lugar a la siguiente expresión:

$$\tau_{ij} = -\rho\overline{u_i' v_j'} = \mu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) - \frac{2}{3}\, \rho k \delta_{ij} \tag{3.14}$$

Donde $k = \frac{1}{2}(\overline{u'^2} + \overline{v'^2} + \overline{w'^2})$ es la energía cinética turbulenta por unidad de masa. El primer término del miembro derecho, es una analogía con la ley de Newton de la viscosidad, en donde aparece la viscosidad de remolino μ_t.

Cabe destacar, que existen otros métodos para resolver problemas con flujos turbulentos, como L.E.S (Large Eddy Simulation), o D.N.S (Direct numerical simulation), los cuales, si bien, pueden entregar resultados mucho más aproximados a la realidad para geometrías complejas, tienen un costo computacional mucho más elevado, que en muchos casos hace inviable su uso industrial en la actualidad.

3.5 Modelo k-ε

El modelo k- ε se enfoca en los mecanismos que afectan a la energía cinética turbulenta k. la energía cinética instantánea k(t) de un flujo turbulento, es la suma de la energía cinética media $K = \frac{1}{2}(U^2 + V^2 + W^2)$, y la energía cinética turbulenta $k = \frac{1}{2}(\overline{u'^2} + \overline{v'^2} + \overline{w'^2})$, de modo tal que:

$$k(t) = K + k \tag{3.15}$$

Operando algebraicamente las ecuaciones de Reynolds-Navier Stokes (Ver An introduction to Computational Fluid Dynamics – Trubulence and its modelling (W.Malalasekera, 2007)), pueden obtenerse ecuaciones de conservación, para la energía cinética media, como también para la energía cinética instantánea del flujo turbulento. En estas expresiones, es posible observar, que uno de los términos es igual en magnitud, pero de signo opuesto, y hace referencia a la conversión de energía cinética del flujo medio en energía cinética turbulenta. Este término, deriva en la tasa de disipación por unidad de volumen de energía cinética turbulenta ε, que es causada por el trabajo realizado por los remolinos más pequeños contra las tensiones viscosas.

Para que los cálculos sean estables, ε, debe ser del mismo orden de magnitud que la energía cinética de la turbulencia k, ya que si no, estaría entregándose más energía a los remolinos turbulentos de la que se disipa, y crecería indefinidamente. De aquí es que, aplicando análisis dimensional, es posible especificar la viscosidad de remolino para modelar las tensiones de Reynolds como:

$$\mu_t = C\rho v\ell = \rho C_\mu \frac{k^2}{\varepsilon} \tag{3.16}$$

Mediante la introducción de una ecuación de conservación para k y ε, puede obtenerse la viscosidad de remolino, para modelar las tensiones de Reynolds, y así, resolver el sistema de ecuaciones RANS para flujo turbulento.

3.6 Modelo k-ω

En el modelo k-ε, la viscosidad turbulenta, es expresada como el producto de la escala de velocidad y la escala de longitud. La tasa de disipación de energía cinética turbulenta, no es la única variable posible para determinar la escala de longitud, en efecto, muchos otros modelos de dos ecuaciones han sido desarrollados. La alternativa más utilizada, es el modelo k-ω, el cual utiliza la frecuencia de la turbulencia ω=ε/k como una segunda variable. Utilizando esta variable la longitud de escala se escribe como $\ell = \sqrt{k}/\omega$, por lo cual, la viscosidad de la turbulencia queda dada por:

$$\mu_t = \rho k/\omega \tag{3.17}$$

Al igual que para el modelo k-ω, dos ecuaciones de transporte para k y ω, son utilizadas para derivar en μ_t.

3.7 Otros modelos

Se pueden mencionar una gran cantidad de modelos, y variantes de los anteriores para obtener la viscosidad de la turbulencia, como por ejemplo la variante SST k-ω, que busca mejorar la precisión en la cercanía de la pared del dominio ante gradientes adversos de presión combinando los modelos k-ω y k-ε, según donde funcione mejor cada uno. Una lista de los modelos restantes más conocidos se menciona a continuación:

- Mixing length model
- Spalart – Allmaras model
- Non-linear k-ε models

4 Error e Incertezas en la dinámica de los fluidos computacional

4.1 Validación y Verificación

La evaluación de la calidad de una simulación, es comúnmente conocida como **Verificación** y **Validación**, (P.J.Roche, Verification and Validation in Computational Science and Engeenering, 1998) donde la verificación se encarga de demostrar que se están resolviendo las ecuaciones de forma correcta, desde el punto de vista del modelado numérico, ya sea evaluando la forma en la que el código resuelve las ecuaciones a través del error, en comparación con una solución conocida, o bien, evaluando los resultados a través de incertezas cuando la solución es desconocida. Por otro lado, la validación, se enfoca en demostrar que las ecuaciones que se están resolviendo, son las correctas, es decir, que hace referencia al modelado físico.

A la hora de hablar de verificación, es importante diferenciar entre el concepto de error, e incerteza, ya que el primero hace referencia a un valor exacto, que surge de la comparación, del resultado obtenido de una simulación, con una solución exacta (verificación del código), obtenida, por ejemplo, de datos experimentales, mientras que el concepto de incerteza, hace referencia a un valor, que define un rango, dentro del cual se encuentra la solución exacta, que por lo general es desconocida (verificación de la solución), con un cierto grado de confianza dado por el procedimiento utilizado para la estimación de dicha incerteza.

En cuanto al error en una simulación numérica, puede diferenciarse entre tres tipos de errores. El primero se denomina error de redondeo, que se produce cuando se utilizan números que tienen un límite de cifras significativas para representar números exactos, y en la actualidad suele ser insignificante en comparación debido a la doble precisión de las computadoras. Luego tenemos el error iterativo, que se da en las simulaciones numéricas debido a la utilización de métodos iterativos para resolver sistemas de ecuaciones algebraicas, y va disminuyendo a medida que aumenta el número de iteraciones, por ende, puede reducirse a órdenes de magnitud muy pequeños, dependiendo la complejidad del problema a resolver. Por último, el error de discretización, o mejor conocido como error de truncamiento, que resulta del empleo de aproximaciones como un procedimiento matemático exacto, al convertir las ecuaciones diferenciales parciales que rigen para un sistema continuo, en un sistema de ecuaciones algebraicas aplicado a un dominio espacial discretizado. Un claro ejemplo, surge en el uso de esquema de discretización, para aproximar los flujos convectivos y difusivos, en los límites de

los volúmenes de control, mediante el uso de las variables adyacentes almacenadas en los nodos. El error de discretización, disminuye con el tamaño de grilla, ya que, al utilizar un número finito de términos en la serie de Taylor, para el empleo de esquemas de discretización, a medida que disminuye la distancia entre puntos nodales, disminuye este tipo de error. En la mayoría de las simulaciones CFD, el error de discretización, es el de mayor orden de magnitud, y es a quien se prestara atención en este trabajo, por considerar a las otras fuentes de error, insignificantes en comparación.

4.2 Representación en serie de potencias del error de discretización.

La estimación del error de discretización es realizada, mediante la expansión en serie de potencias (truncada). La ecuación básica para estimar el error de discretización, (P.J.Roche, Fundamentals of Verification and Validation , 2009) es:

$$\in_{\emptyset} \cong \delta_{RE} = \emptyset_i - \emptyset_0 = \alpha h_i^p \tag{4.1}$$

$\emptyset_i$ representa una cantidad integral o local del flujo, $\emptyset_0$ es la estimación de la solución exacta, α es una constante a ser determinada, h_i es el tamaño típico de celda o volumen y p es el orden de convergencia de grilla observado. La estimación de $\in_{\emptyset}$ requiere la determinación de $\emptyset_0$, α y p.

Como puede apreciarse, la estimación del error, posee tres incógnitas, por lo cual, para cerrar el sistema se necesitan como mínimo tres ecuaciones. Esto quiere decir, que deben ser realizadas como mínimo tres simulaciones, con un espaciamiento de grilla diferente entre sí. A su vez, estas grillas deben encontrarse dentro de lo que se conoce como rango asintótico, es decir, que el error, producto de los tamaños de grillas seleccionados, tiene un comportamiento asintótico, lo cual garantiza, que el término principal de la expansión en serie de potencias es suficiente para estimar el error. Otro aspecto importante para garantizar la correcta aplicación del procedimiento, es que las mallas sean geométricamente similares y con igual relación de refinamiento entre las mallas, para poder captar realmente el comportamiento del error de discretización, y errores debido a las diferencias en las mallas, se adicionen a los resultados.

Dado que no siempre las condiciones anteriores pueden darse la práctica, existen otros estimadores a utilizar, que, en determinados casos, pueden representar mejor al comportamiento del error.

$$\in_{\emptyset} \cong \delta_1 = \emptyset_i - \emptyset_0 = \alpha h_i \tag{4.2}$$

$$\in_{\emptyset} \cong \delta_2 = \emptyset_i - \emptyset_0 = \alpha h_i^2 \tag{4.3}$$

$$\in_{\emptyset} \cong \delta_{12} = \emptyset_i - \emptyset_0 = \alpha_1 h_i + \alpha_2 h_i^2 \tag{4.4}$$

Estas tres alternativas son únicamente usadas si la estimación de la primera ecuación es imposible o no es redituable, es decir, si el orden de convergencia de grilla es demasiado pequeño o demasiado grande. Como la selección del orden de convergencia es importante para conocer si nuestras simulaciones se encuentran o no en el rango asintótico, entonces, en la práctica conviene utilizar más de tres grillas para la estimación del error, ya que, mientras menor sea la cantidad, mayor será la sensibilidad de los datos (L.Eca M. , 2002). Es muy recomendable usar al menos cuatro grillas cuando se espera cierta dispersión en los datos, es decir, para la mayoría de los problemas de flujo de ingeniería.

La obtención las variables que presentan a los ajustes, pueden ser obtenidos mediante el uso de la estimación de error por mínimos cuadrados. Es decir, las variables incógnitas, son obtenidas, a través de una curva, que se ajuste a los datos, de tal manera, que el error por mínimos cuadrados, entre esta curva, y los datos obtenidos, sea le menor posible. Luego, la mejor estimación del error, resulta como aquella que posee el menor error estándar estimado, o desviación estándar. El error por mínimos cuadrados, para cada una de las expresiones se presenta a continuación:

$$S_{RE}(\emptyset_0, \alpha, p) = \sqrt{\sum_{i=1}^{n_g} (\emptyset_i - (\emptyset_0 + \alpha h_i^p))^2} \tag{4.5}$$

$$S_1(\emptyset_0, \alpha) = \sqrt{\sum_{i=1}^{n_g} (\emptyset_i - (\emptyset_0 + \alpha h_i))^2} \tag{4.6}$$

$$S_2(\emptyset_0, \alpha) = \sqrt{\sum_{i=1}^{n_g} (\emptyset_i - (\emptyset_0 + \alpha h_i^2))^2} \tag{4.7}$$

$$S_{12}(\emptyset_0, \alpha_1, \alpha_2) = \sqrt{\sum_{i=1}^{n_g} (\emptyset_i - (\emptyset_0 + \alpha_1 h_i + \alpha_2 h_i^2))^2} \tag{4.8}$$

Por lo general, las grillas más finas, son más confiables en cuanto a los resultados, ya que la estimación por serie de potencias, se aproxima de una mejor manera, a la función a representar, por ende, puede darse más peso a las grillas más finas, y derivar en un enfoque ponderado como se muestra a continuación.

$$S_{RE}^{w}(\emptyset_0, \alpha, p) = \sqrt{\sum_{i=1}^{n_g} w_i (\emptyset_i - (\emptyset_0 + \alpha h_i^p))^2} \tag{4.9}$$

$$S_1^w(\emptyset_0, \alpha) = \sqrt{\sum_{i=1}^{n_g} w_i (\emptyset_i - (\emptyset_0 + \alpha h_i))^2} \tag{4.10}$$

$$S_2^w(\emptyset_0, \alpha) = \sqrt{\sum_{i=1}^{n_g} w_i (\emptyset_i - (\emptyset_0 + \alpha h_i^2))^2} \tag{4.11}$$

$$S_{12}^w(\emptyset_0, \alpha_1, \alpha_2) = \sqrt{\sum_{i=1}^{n_g} w_i (\emptyset_i - (\emptyset_0 + \alpha_1 h_i + \alpha_2 h_i^2))^2} \tag{4.12}$$

Donde w_i se basa en el tamaño típico de celda:

$$w_i = \frac{\frac{1}{h_i}}{\sum_{i=1}^{n_g} \frac{1}{h_i}} \tag{4.13}$$

Garantizando que

$$\sum_{i=1}^{n_g} w_i = 1 \tag{4.14}$$

La minimización de mínimos cuadrados de las ecuaciones vistas hasta aquí se presentan en el a continuación, el cual además incluye la definición de la desviación estándar de los ajustes que serán usados como una medida de la calidad del ajuste (L.Eca M. , 2002).

Para la expansión de un solo término con el orden de convergencia de grilla incógnita, $\emptyset_0, \alpha \; y \; p$ son determinadas del mínimo de la función:

$$S_{RE}(\emptyset_0, \alpha, p) = \sqrt{\sum_{i=1}^{n_g} (\emptyset_i - (\emptyset_0 + \alpha h_i^p))^2} \tag{4.15}$$

Que se obtiene haciendo:

$$\frac{\partial S_{RE}}{\partial \emptyset_0} = 0 \; , \quad \frac{\partial S_{RE}}{\partial \alpha} = 0 \quad , \frac{\partial S_{RE}}{\partial p} = 0 \; , \tag{4.16}$$

Lo cual deriva en el siguiente sistema de ecuaciones no lineales:

$$\emptyset_0 = \sum_{i=1}^{n_g} w_i \emptyset_i - \alpha \sum_{i=1}^{n_g} w_i h_i^p \tag{4.17}$$

$$\alpha = \frac{\sum_{i=1}^{n_g} w_i \emptyset_i h_i^p - (\sum_{i=1}^{n_g} w_i \emptyset_i)(\sum_{i=1}^{n_g} w_i h_i^p)}{\sum_{i=1}^{n_g} w_i h_i^{2p} - (\sum_{i=1}^{n_g} w_i h_i^p)(\sum_{i=1}^{n_g} w_i h_i^p)} \tag{4.18}$$

$$\sum_{i=1}^{n_g} w_i \emptyset_i h_i^p \log(h_i) - \emptyset_0 \sum_{i=1}^{n_g} w_i h_i^p \log(h_i) - \alpha \sum_{i=1}^{n_g} w_i h_i^{2p} \log(h_i) = 0 \tag{4.19}$$

La desviación estándar del ajuste queda dada por:

$$\sigma_{RE} = \sqrt{\frac{\sum_{i=1}^{n_g} n.w_i(\emptyset_i - (\emptyset_0 + \alpha h_i^p))^2}{(n_g - 3)}} \tag{4.20}$$

Para La expansión de un solo término de primer orden, $\emptyset_0$ y α, son determinados del mínimo de la función:

$$S_1(\emptyset_0, \alpha) = \sqrt{\sum_{i=1}^{n_g} (\emptyset_i - (\emptyset_0 + \alpha h_i))^2} \tag{4.21}$$

Que es obtenido de igualar:

$$\frac{\partial S_1}{\partial \emptyset_0} = 0 \ , \ \ \frac{\partial S_1}{\partial \alpha} = 0 \tag{4.22}$$

lo cual deja el siguiente sistema de ecuaciones lineales:

$$\begin{bmatrix} 1 & \sum_{i=1}^{n_g} w_i h_i \\ \sum_{i=1}^{n_g} w_i h_i & \sum_{i=1}^{n_g} w_i h_i^2 \end{bmatrix} \begin{bmatrix} \emptyset_0 \\ \alpha \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n_g} w_i \emptyset_i \\ \sum_{i=1}^{n_g} w_i \emptyset_i h_i \end{bmatrix} \tag{4.23}$$

Cuya desviación estándar está dada por:

$$\sigma_{RE} = \sqrt{\frac{\sum_{i=1}^{n_g} n.w_i(\emptyset_i - (\emptyset_0 + \alpha h_i))^2}{(n_g - 2)}} \tag{4.24}$$

Para expansión de un solo término de segundo orden, $\emptyset_0$ y α, son determinados del mínimo de la función:

$$S_2(\emptyset_0, \alpha) = \sqrt{\sum_{i=1}^{n_g} (\emptyset_i - (\emptyset_0 + \alpha h_i^2))^2} \tag{4.25}$$

Que es obtenido de igualar:

$$\frac{\partial S_2}{\partial \emptyset_0} = 0 \ , \ \ \frac{\partial S_2}{\partial \alpha} = 0 \tag{4.26}$$

Derivando en el siguiente sistema de ecuaciones lineales:

$$\begin{bmatrix} 1 & \sum_{i=1}^{n_g} w_i h_i^2 \\ \sum_{i=1}^{n_g} w_i h_i^2 & \sum_{i=1}^{n_g} w_i h_i^4 \end{bmatrix} \begin{bmatrix} \emptyset_0 \\ \alpha \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n_g} w_i \emptyset_i \\ \sum_{i=1}^{n_g} w_i \emptyset_i h_i^2 \end{bmatrix} \tag{4.27}$$

Cuya desviación estándar está dada por:

$$\sigma_{RE} = \sqrt{\frac{\sum_{i=1}^{n_g} n. w_i (\emptyset_i - (\emptyset_0 + \alpha h_i^2))^2}{(n_g - 2)}} \tag{4.28}$$

Para expansión de dos términos, con términos de primer y segundo orden, $\emptyset_0$ y α, son determinados del mínimo de la función:

$$S_{12}(\emptyset_0, \alpha_1, \alpha_2) = \sqrt{\sum_{i=1}^{n_g} (\emptyset_i - (\emptyset_0 + \alpha_1 h_i + \alpha_2 h_i^2))^2} \tag{4.29}$$

Que es obtenido de igualar:

$$\frac{\partial S_{12}}{\partial \emptyset_0} = 0 \ , \ \ \frac{\partial S_{12}}{\partial \alpha_1} = 0 \ \ , \frac{\partial S_{12}}{\partial \alpha_2} = 0 \ , \tag{4.30}$$

Dando lugar al siguiente sistema de ecuaciones lineales:

$$\begin{bmatrix} 1 & \sum_{i=1}^{n_g} w_i h_i & \sum_{i=1}^{n_g} w_i h_i^2 \\ \sum_{i=1}^{n_g} w_i h_i & \sum_{i=1}^{n_g} w_i h_i^2 & \sum_{i=1}^{n_g} w_i h_i^3 \\ \sum_{i=1}^{n_g} w_i h_i^2 & \sum_{i=1}^{n_g} w_i h_i^3 & \sum_{i=1}^{n_g} w_i h_i^4 \end{bmatrix} \begin{bmatrix} \emptyset_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n_g} w_i \emptyset_i \\ \sum_{i=1}^{n_g} w_i \emptyset_i h_i \\ \sum_{i=1}^{n_g} w_i \emptyset_i h_i^2 \end{bmatrix} \tag{4.31}$$

Dicho ajuste, posee una desviación estándar dada por:

$$\sigma_{12} = \sqrt{\frac{\sum_{i=1}^{n_g} n. w_i (\emptyset_i - (\emptyset_0 + \alpha_1 h_i + \alpha_2 h_i^2))^2}{(n_g - 3)}} \tag{4.32}$$

4.3 Procedimiento propuesto para la estimación del error

Habiendo descrito los ingredientes principales, es posible proponer un procedimiento para la estimación del error de discretización (M.Hoekstra, 2014). El procedimiento depende la disponibilidad de al menos cuatro puntos de datos, en la elección correcta de un tamaño de celda típico y en la suposición de que los errores de redondeo e iterativos son insignificantes con respecto al error de discretización.

El procedimiento comienza determinando el orden de convergencia de grilla p, desde el mínimo de S_{RE} y S_{RE}^{w}, de modo tal que, si ambas soluciones contienen un 0.5<p<2 , entonces, el mejor ajuste es seleccionado como aquel, que posea la menor desviación estándar. En el caso que una de las dos soluciones de un p fuera de este rango, esa solución es descartada, y el mejor ajuste es seleccionado como el restante. Si ambas soluciones dan un p fuera de este rango, se procede con el uso de los demás estimadores de error. Si p>2, las estimaciones de error pueden ser demasiado pequeñas para los dos estimadores anteriores, por ende, se selecciona entre los estimadores 4.2 o 4.3, con y sin ponderar, al de menor desviación estándar. Si p<0.5, el error

estimado también puede ser demasiado conservativo, por ende, se procede de manera similar a cuando p>2, solo que se agrega la opción de seleccionar el estimador 4.4, en caso de que este sea el de menor desviación estándar, de nuevo, con y sin ponderación.

4.4 Estimación de incerteza.

Una vez obtenida el mejor estimador del error (M.Hoekstra, 2014), el error estimado es convertido en una incerteza con un factor de seguridad que depende del orden de convergencia observado y de la desviación estándar del ajuste La desviación estándar del ajuste σ, el orden de convergencia de grilla p, y la estimación del error, son ahora utilizadas para el cálculo de la incerteza $U_\emptyset$,de cada uno de los datos, es decir, un rango en el cual, la solución exacta, se encuentra con un 95% de probabilidad.

$$\emptyset_i - U_\emptyset \leq \emptyset_{exact} \leq \emptyset_i + U_\emptyset \tag{4.33}$$

Antes del cálculo de la incerteza para cada variable obtenida, se calcula una variable denominada rango de datos, que define en parte, el valor del factor de seguridad a utilizar para multiplicar al error, y obtener el valor de la incerteza. Este valor se calcula como:

$$\Delta_\emptyset = \frac{(\emptyset_i)_{max} - (\emptyset_i)_{min}}{n_g - 1} \tag{4.35}$$

La estimación del error es considerada de confianza si la solución es monótonamente convergente con 0.5<p<2. Y si $\sigma < \Delta_\emptyset$. siguiendo el procedimiento de indice de convergencia de grilla, el factor de seguridad es elgido como Fs=1.25 si la estimación del error es considerado confiable, sino Fs=3. Entonces, en base a la relación entre la desviación estándar y el rango de datos, se utilizan diferentes expresiones para el cálculo de la incerteza, $\sigma < \Delta_\emptyset$ y $\sigma > \Delta_\emptyset$.

- Para $\sigma < \Delta_\emptyset$

$$U_\emptyset = Fs. \in_\emptyset (\emptyset_i) + \sigma + \left|\emptyset_i - \emptyset_{ajuste}\right| \tag{4.35}$$

- Para $\sigma > \Delta_\emptyset$

$$U_{\emptyset} = 3\frac{\sigma}{\Delta_{\emptyset}}\left(\epsilon_{\emptyset}\ (\emptyset_i) + \sigma + \left|\emptyset_i - \emptyset_{ajuste}\right|\right) \tag{4.36}$$

La incerteza estimada contiene por lo tanto tres componentes, el valor absoluto del error de discretización estimado multiplicado por un factor de seguridad, la desviación estándar del ajuste, la diferencia entre punto de datos reales y el valor obtenido del ajuste para la misma densidad de grilla.

5. RESULTADOS

5.1 Metodología

El presente trabajo intenta reproducir un análisis de incertezas sobre un caso ampliamente estudiado denominado Backward Facing Step, el cual se especifica en detalle en el apartado 3.2. El desarrollo de las simulaciones numéricas, es realizado usando OpenFOAM®, por medio del sistema operativo Linux, en un computador laptop BANGHO®, con un procesador Intel® Core(TM) i7-4710MQ 2.5GHz y 8GB RAM.

Un total de 30 simulaciones son realizadas para el desarrollo del trabajo, con el fin obtener una cantidad necesaria de datos para realizar un análisis de incertezas, y poder estudiar la variación de las mismas, al cambiar distintos parámetros de las configuraciones numéricas. Las 30 simulaciones son realizadas con el solver simpleFoam que se corresponde con el método SIMPLE visto en la sección de modelado numérico de este trabajo. A su vez, las simulaciones se dividen en tres grupos, de modo tal que, 10 son realizadas utilizando el modelo de turbulencia k-épsilon, 10 con el modelo k-w y las 10 restante con el modelo k-wSST. Para cada modelo de turbulencia, las 10 simulaciones se dividen en dos casos particulares, en los cuales se evalúan 5 mallas por separado. Cada grupo de 5 mallas equivalentes geométricamente, con un refinamiento constante entre malla y malla, son utilizados para realizar un análisis de incertezas. La diferencia entre los dos grupos, de cinco mallas, correspondientes a cada modelo de turbulencia, es que en ambos grupos se tiene un refinamiento del tamaño de celda de 1.2, se utiliza un esquema de discretización upwind, pero se encuentran en un rango diferente de refinamiento de celda.

Una vez realizada todas las simulaciones, los datos son procesados atreves de un código de Python, que contiene embebido el procedimiento para análisis de incertezas mencionado 4.4. Este código, devuelve como salida parámetros de los ajustes realizados para la regresión de cada conjunto de datos, y una gráfica que muestra el valor de la incerteza numérica para cada resultado de la simulación, en función del refinamiento de grilla. Posteriormente, se presentan gráficos acerca del comportamiento de los residuos de las simulaciones, y conclusiones son realizadas acerca de los resultados obtenidos.

5.2 OpenFoam®

Un gran abanico de softwares, han sido desarrollados para realizar simulaciones de CFD, y trabajar sobre las fases de pre-procesamiento, resolución de ecuaciones y post-procesamiento. Existen softwares comerciales, tales como ANSYS FLUENT®, STAR-CMM+® y FLOW3D®, dentro de los más utilizados, como también existen códigos libres de gran utilidad, como es el caso de OpenFOAM®.

OpenFOAM®, es un software de uso gratuito y de código abierto, que comenzó a ser distribuido en el año 2004, desarrollado por el Imperial College de Londres, institución muy reconocida por su dedicación al estudio de la mecánica de los fluidos computacional desde 1960. El código es aplicable a un amplio rango de campos de la ingeniería como, turbulencia, transferencia de calor, reacciones químicas, electromagnetismo, y demás, donde cada caso es trabajado en un conjunto de archivos de texto, que luego son procesados, y se encuentran clasificados en función del solucionador, el tipo de flujo, y la simulación que se desea realizar. Esta herramienta a su vez posee módulos especiales que permiten realizar analizar de turbo maquinaria, hasta la posibilidad de realizar mallas dinámicas para las simulaciones de cámaras de combustión de motor de combustión interna.

OPEN FOAM, compatible con el sistema operativo Linux, posee una herramienta que permite realizar geometrías y mallas, a través de archivos de texto, denominada BlockMesh, como también posee una herramienta de post-procesamiento que permite la visualización y análisis de datos denominada paraView.

Los casos de OpenFOAM®, se configuran editando archivos de casos (archivos de texto). La edición de archivos es posible en OpenFOAM® porque la interface de inputs y outpus, usa un formato de diccionario con palabras clave que transmiten significado suficiente para ser entendido por el usuario, por lo que el usuario debe seleccionar un editor (en función del solver y el tipo de flujo) de su elección, y modificar los archivos de texto para configurar lo que se quiere simular.

El primer paso a la hora de trabajar con OpenFOAM®, es tomar la carpeta de caso adecuada en base al tipo de flujo y el solucionador a utilizar. Esta carpeta se compone de tres campos, denominados:

- 0: contiene los archivos de textos utilizados para la configuración de las condiciones iniciales (velocidad, presión y variables del modelo turbulento) y de contorno del problema.
- Constant: se encuentran los archivos de texto, necesarios para la configuración de los coeficientes de transporte y modelo de turbulencia.
- System: contiene los archivos correspondientes a: BlockMeshDict (necesario para la realización del dominio computacional y el mallado correspondiente). fvschemes, en el cual se configuran los esquemas de discretización a utilizar para los términos de las ecuaciones diferenciales a resolver. Controldict, archivo en el cual se especifican el número de iteraciones a realizar, tipo de solución (ej: estado estacionario), como también, la edición de archivos de salida. Por último, se tiene un archivo denominado fvsolution, en el cual se configuran los parámetros del solucionador y las condiciones de tolerancia y convergencia.

La etapa de pre-procesamiento, es llevada a cabo a partir de la configuración de los archivos anteriormente mencionados, de acuerdo a el caso específico que se desea simular. Una vez realizada dicha tarea, la carpeta es ejecutada a través de la terminal de Linux, donde el código resuelve las ecuaciones mediante el solucionador correspondiente empleando como parámetros de entrada, los datos proporcionados en la etapa anterior. Posteriormente los datos son visualizados y procesados en paraView, que posee una interface amigable para el usuario y varias herramientas que permiten estudiar variables locales y globales del flujo, en todo punto del domino.

5.3 Caso de análisis - Backward Facing Step

En el presente trabajo, un análisis de incertezas sobre simulaciones de CFD es realizado, utilizando OpenFOAM®, reproduciendo el caso ampliamente estudiado conocido como Backward Facing Step (Spalart ,1986) (Driver, 1991).

El caso Backward Facing step, hace referencia a un flujo de fluido incompresible que circula por un canal rectangular, el cual posee un escalón, a partir del cual la sección por donde circula el fluido se ensancha abruptamente, produciéndose un flujo recirculante debido al desprendimiento de la capa límite, producido por el gradiente adverso de presión, como se muestra en la figura 3.1.

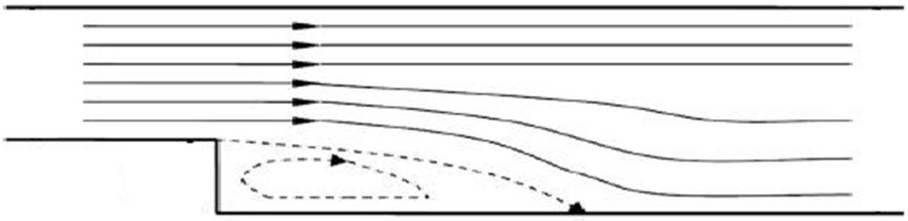

Figura 5.1 – Backward Facing Step

La elección de este caso para este trabajo final, se debe a su extensa reproducción por distintos investigadores y entidades reconocidas como N.A.S.A (National Aeronautics and Space Administration) y E.R.C.O.F.T.A.C (European Research Community on Flow, Turbulence and Combustión), lo que facilita la obtención de datos para la comparación de simulaciones numéricas, como de datos obtenidos de experimentos de laboratorio. Particularmente la N.A.S.A, dispone en su página web de datos experimentales, de distintas variables, del caso Backward Facing Step, cuyo fin es la validación de distintos modelos de turbulencia.

Los parámetros físicos y geométricos, son reproducidos desde los casos de estudios mencionados en el párrafo anterior, con el fin de poder realizar comparaciones. La geometría de estudio, entonces, se analiza desde una perspectiva 2d, y está dada por una relación *altura de entrada/altura de escalón=8/1*, con el fin de reducir el gradiente adverso de presiones y disminuir el costo computacional a la hora de realizar las simulaciones numéricas. La altura del escalón h, se corresponde a media pulgada, lo que es lo mismo h=0.0127m. El escalón se encuentra ubicado 4h aguas arriba de la entrada del flujo y el dominio termina 40h aguas abajo del escalón con una altura de 9h, como se muestra en la figura 3.2.

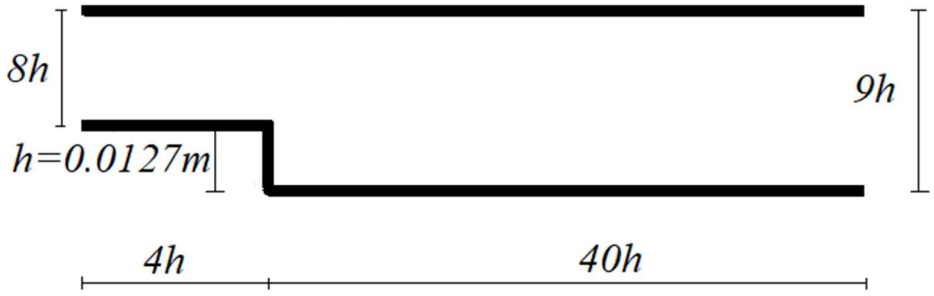

Figura 5.2 – Dominio geométrico

En la entrada se utiliza un perfil parabólico de velocidad, donde la velocidad de referencia es de U_{ref}=44.2m/s, y el número de Reynolds basado en la altura del escalón y la velocidad de

referencia es de Re=50.000. Este perfil de velocidad parabólico, se obtiene de datos experimentales, provistos por la N.A.S.A, el cual busca reproducir correctamente la formación de la capa límite y los efectos que se producen en la zona aledaña a la pared. Sin embargo, la entidad mencionada, proporciona los datos del perfil de velocidad, en forma adimensionalizada por la velocidad de referencia, es decir *u/Uref*, al igual que los demás datos experimentales. Es por esto, que, en el presente trabajo, se trabaja con una velocidad adimensionalizada, como así también, con un dominio adimensionalizado por la altura del escalón h. En la figura 3.3, se observa la coordenada vertical adimensionalizada en función de la componente horizontal de la velocidad adimensionalizada.

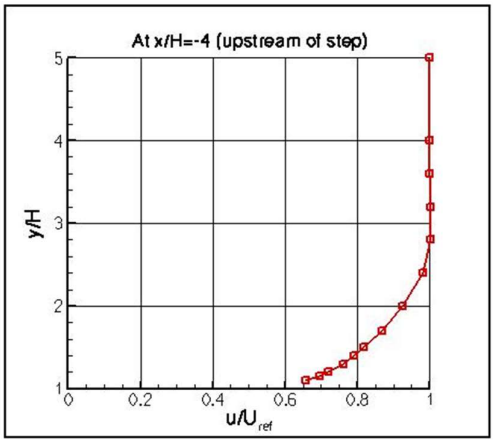

Figura 5.3 – Perfil de velocidad de entrada - Reproducida de https://turbmodels.larc.nasa.gov.

5.4 Configuración Numérica

La configuración numérica en OpenFoam®, se estructura alrededor de la carpeta de casos incompresible/simpleFoam/pitzDaily. Dicha estructura contiene los archivos mencionados en el apartado 5.2, los cuales deben ser configurados para la adaptación al caso de estudio Backward Facing Step.

5.4.1 Mallado de la geometría

La etapa de pre-procesamiento, como se mencionó en apartados anteriores, es la primera etapa de toda simulación CFD, donde el primer paso de esta etapa es la creación de la geometría computacional y el mallado de la misma.

La geometría computacional, está dado por las dimensiones descritas en el apartado 5.3 para Backward Facing Step, las cuales son ingresadas a OpenFOAM®, a través del archivo

BlockMeshDict (Ver APENDICE A). BlockMesh, el módulo de mallado de OpenFoam®, crea la geometría y la malla mediante la lectura de un mismo archivo, de modo tal que el primer paso, consiste en definir los vértices que definen a la geometría computacional. Ahora bien, estos vértices, por lo general, son más que los que definen a la geometría real del dominio, ya que, a la hora de realizar una malla para una simulación CFD, el dominio se subdivide en bloques, con el fin de poder dar distintas relaciones de distancia entre celdas. En las zonas donde se requiere mayor precisión en los cálculos (pj: cerca de la pared, para captar mejor los efectos de la capa límite), la malla debe ser más fina, o lo que es lo mismo, la distancia entre celdas debe ser menor. En zonas donde el flujo no requiere tanta precisión (flujo medio), conviene tener una distancia de celdas mayor, para ahorrar recursos computacionales, y no realizar una simulación con todo el dominio con una distancia entre celdas muy pequeña, lo que equivale a un número mayor de nodos a calcular. Ahora bien, otro echo importante, es que la variación del tamaño de celdas, debe ser gradual de una región del dominio a otra, y cada cara de una celda o volumen, debe coincidir con una única cara de una celda o volumen vecino. De no cumplirse estas condiciones, la simulación puede incurrir en problemas de divergencia numérica a la hora de calcular derivadas como así también en problemas de conservación.

Es por estos motivos, que el dominio computacional se divide en bloques, que, para mejor entendimiento, pueden imaginarse como un cubo, en donde a cada eje, se le asigna una dirección, y ley de mayado, es decir, una ley con la cual el tamaño de la celda va disminuyendo a lo largo de un eje, de manera gradual, para evitar sobre saltos bruscos en el tamaño de celda. Esta ley de mayado, puede ser una relación lineal, es decir, que el tamaño de celda puede ir variando linealmente, o bien puede tener cualquier ley matemática aplicable. El gradualismo de esta variación, puede afectar en los cálculos de flujos turbulentos. El Dr. Daniel Horna Muñoz, comenta durante una charla de investigación numérica de dinámica fluviales, en la Facultad de Ciencias Exactas Físicas y Naturales, como, trabajando para el instituto IOWA FLOOD CENTER, debieron dedicar gran cantidad de tiempo, para descubrir un problema de divergencia en simulaciones de inundaciones, el cual finalmente derivo en un factor demasiado alto de la ley lineal de mallado (mayor a 0.1), al aplicar el método L.E.S. los métodos R.A.N.S sin embargo, son más estables a mayores factores de reducción del tamaño de celda.

En la figura 5.4, puede observarse el mapa de los vértices necesarios para formar los bloques en los cuales se ha subdividido el dominio.

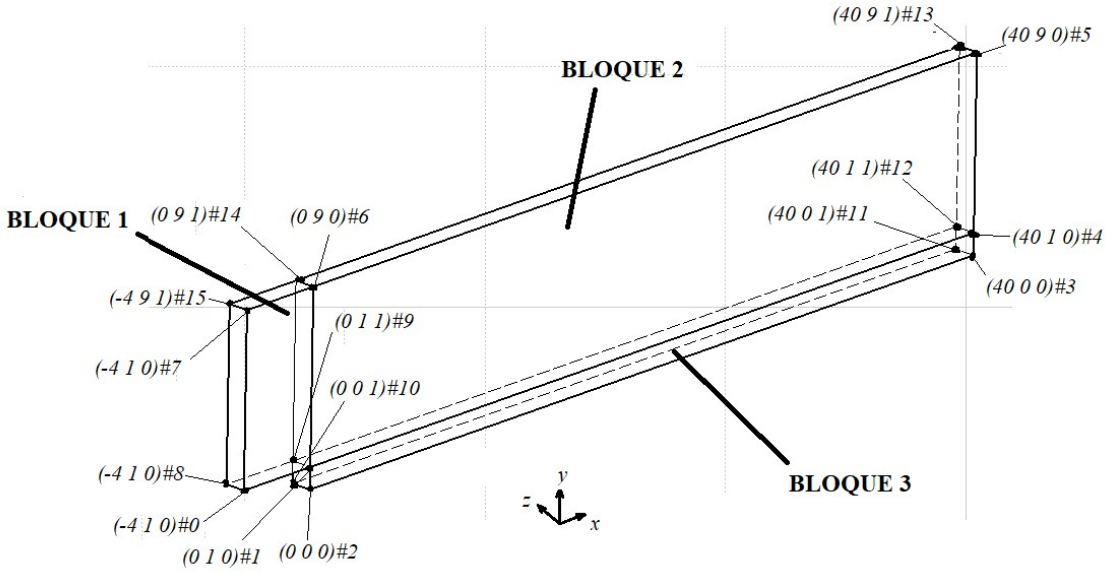

Figura 5.4 – Mapa de vértices-

Estos vértices, son ingresados, en forma de coordenadas espaciales y con el orden especificado en el archivo BlockMeshDict, bajo la keyword vértices. Posteriormente, los bloques son creados, ingresando una lista de los vértices, que componen cada bloque (Ver APENDICE A), donde cada vértice es ingresado con el numero con el que este, fue colocado en la lista de vértices, y en el orden que dispone OpenFoam®, que sigue la disposición observada en la figura 5.5.

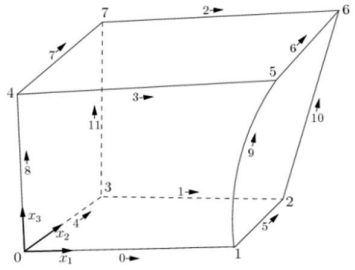

Figura 5.5 – convención de vértices según OpenFoam- Reproducido de https://cfd.direct/openfoam/user-guide/v6-blockmesh

Debajo de los vértices que conforman cada bloque, se ingresa el número de celdas, que contendrá el bloque en cada dirección, y luego son ingresados los factores de refinamiento que tendrán estas celdas en la dirección dada. En el APENDICE A, puede observarse que, para la dirección vertical, se hace uso de la opción que dispone BlockMesh, de dividir la longitud de un eje perteneciente a un bloque, asignarle a cada parte una cantidad de celdas y una dirección de mallado. Esto permite, refinar la malla en dos direcciones, utilizando un solo bloque,

mientras que, en otros mayadores, la misma tarea debe ser realizada con el agregado de otro bloque.

La última configuración a realizar en el archivo BlockMeshDict, es la definición de los límites de dominio en los cuales se especifican las condiciones de borde. Estos límites están conformados por las superficies fronteras del dominio, indicados por los vértices correspondientes, y que para el caso de Backward Facing Step, son configurados según se indica en la figura 5.6.

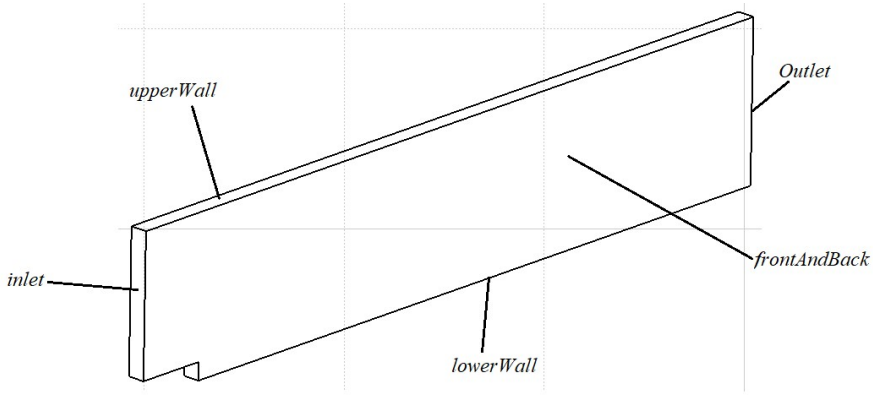

Figura 5.6 – Superficies límites

Una vez configurado el archivo BlockMeshDict, a través de la terminal de Linux, tipiando BlockMesh en la carpeta correspondiente al caso de Backward Facing Step, la malla se crea automáticamente. En total, 10 mallas distintas son creadas para las 30 simulaciones. Todas las mallas son geométricamente equivalentes, y poseen un factor de refinamiento constante de 1.2, con distinto rango de espaciamiento entre celdas. Cada grupo de cinco mallas, son utilizadas para realizar un único análisis de incertezas. Dado que el factor de refinamiento, afecta al número de celdas en cada dirección de un bloque, por ser bidimensional, el número de celdas se ve afectado por un factor de 1.44 respectivamente. A continuación, se presentan las propiedades de las mallas.

Mallas – Grupo 1

	Número de celdas	**Ancho de celdas**
Malla 1	63.600	0.16

Malla 2	91.584	0.14
Malla 3	131.754	0.11
Malla 4	190.165	0.09
Malla 5	273.733	0.08

Mallas – Grupo 2

	Número de celdas	**Ancho de celdas**
Malla 1	41.200	0.0250
Malla 2	59.400	0.0300
Malla 3	849.000	0.0361
Malla 4	122.000	0.0434
Malla 5	175.020	0.0521

El factor de refinamiento de las mallas, como así también, el tamaño de la celda aledaña a la pared, queda definido por el parámetro adimensional conocido como distancia adimensional a la pared y+. Este parámetro, es de fundamental importancia en el cálculo de flujos turbulentos, ya que limita el tamaño de la celda adyacente a la pared, con el fin de captar correctamente los efectos viscosos de la capa límite. Este parámetro se define como el producto de la velocidad de fricción, y la distancia dimensional a la pared, sobre la viscosidad dinámica. Cada zona de un perfil de velocidad turbulento, queda definido entre distintos parámetros de y+, como, por ejemplo, la sub-capa viscosas se encuentra en regiones dadas para $y+<5$. Si la distancia del primer punto nodal (tamaño de celda adyacente a la pared), hacia el limite dado por la pared, hace que el valor de $y+>5$, los efectos viscosos de la capa límite, no serán considerados correctamente por el cálculo, por lo que se obtendrán valores erróneos de las tensiones en los límites. Sin embargo, en casos, en los cuales el número de Reynolds es muy alto, el tamaño de celda adyacente a la pared, puede resultar muy pequeño, por lo cual, se utilizan condiciones de

borde que serán mencionadas más adelante, conocidas como funciones de pared, que simulan los efectos de la capa límite en la zona adyacente a la pared.

Para el correcto funcionamiento, de las funciones de pared, el parámetro y+, debe encontrarse en la región logarítmica del perfil turbulento y+>20. Además, como en este informe se pretende realizar un análisis de incertezas, utilizando como uno de los modelos de turbulencia, al modelo k-épsilon estándar, debe ser tenido en cuenta el hecho de que dicho modelo, falla para bajos números de Reynolds, por lo cual, para casos en los cuales se pretende captar la sub-capa viscosa, mediante la ley de la pared (y+<5), debe optarse por utilizar variantes de este modelo de turbulencia que se adapten a bajos números de Reynolds. Es por esto, que el parámetro y+, queda limitado, como una cota inferior, en nuestro caso, para el tamaño de celda, para un correcto funcionamiento del modelo k-épsilon estándar (300>y+>30). El hecho de elegir dos grupos de mallas ubicados en rangos diferentes de refinamientos, nos permite apreciar, como varia la incertidumbre de los distintos modelos, para diversos valores de y+.

Con la cota inferior de y+>30, un Reynolds 50.000, una longitud referencia L=1 y velocidad de referencia unitaria, se utilizó un calculador del tamaño mínimo de celda, provisto por la N.A.S.A y conocido como Viscous Grid Spacing Calculator, para lo cual se obtuvo una altura de celda mínima dada por 10^{-2}. Luego, utilizando la fórmula de mallado de OpenFoam, y este valor mínimo de celda, una expresión fue obtenida quedando en función de la cantidad de elementos y el factor de refinamiento para un bloque dado, por lo cual, se fue variando el número de elementos, para luego verificar el valor de y+, mediante la utilidad yPlus de OpenFoam, aplicado a la malla más fina, y asegurarse que el resto de las mallas no excedieran esta condición para el Grupo número 2, en el cual se obtuvo un valor mínimo de $Yplus_{min} \approx 32$, mientras que para la malla mas final del grupo 1 se obtiene un valor de $Yplus_{min} \approx 20$.

En la figura 3.7, puede observarse la visualización desde paraView, de la estructura típica de las mallas anteriores, en donde es posible apreciar, como se produce un refinamiento gradual hacia las paredes, como también hacia la zona del vértice del escalón, donde se producen efectos del flujo relevantes.

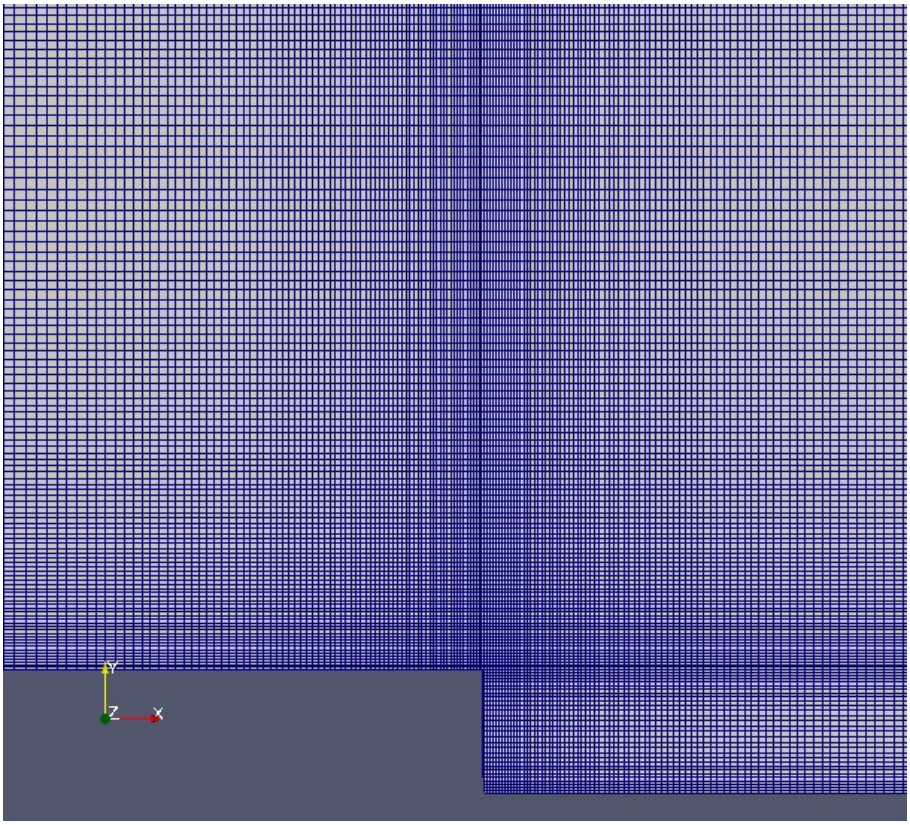

Figura 5.7 –Visualización de malla m5

5.4.2 Configuración de condiciones iniciales y condiciones de borde

La configuración de las condiciones iniciales y las condiciones de borde, necesaria para la resolución del sistema de ecuaciones, es realizada a partir de los archivos ubicados en el directorio 0, mencionado en el apartado 5.2. Estos archivos definen los valores en las caras límites (de borde), y en el dominio interno en el tiempo cero (iniciales) de la presión y velocidad, como también de las respectivas variables de los distintos modelos turbulentos.

Cinco condiciones de borde, más un valor para el campo interno, deben ser asignadas en total. Según la configuración de la malla, tendremos 5 superficies límites, denominadas inlet (entrada de flujo), frontAndBack (límites en la dirección z del dominio), outlet (salida de flujo), upperWall (pared superior) y lowerWall (pared inferior). Para cada variable (presión, velocidad, k, épsilon, omega), debe asignarse un valor o una función a cada uno de estos patch o superficies límites.

OpenFOAM®, dispone de distintas condiciones de borde estándar, para aplicar según el patch o superficie límite, ya sea una pared, en donde es necesario aplicar la condición de no

deslizamiento para la velocidad, o bien, una condición de borde del tipo empty, cuya función es restarle una dimensión al análisis del flujo.

A modo de ejemplo, en el APENDICE B, se muestran las condiciones de borde e iniciales, aplicadas para Backward Facing Step, utilizando el modelo de turbulencia k-wSST.

En toda superficie límite o punto del dominio, la presión o la velocidad debe ser especificada debido a su relación de acoplamiento para la correcta resolución de las ecuaciones que rigen el movimiento del fluido. En base a esto, las condiciones de borde se especifican para Backward Facing Step de la siguiente manera.

Velocidad

- **Inlet:** Como se mencionó en el apartado 5.3, la velocidad de entrada está dado por un perfil obtenido por datos experimentales para recrear de la manera más eficiente las condiciones reales del flujo. Dado que OpenFoam, no cuenta con una condición de borde estándar que pueda recrear las zonas características de un perfil turbulento (sub capa laminar, capa de amortiguamiento, capa de transición y capa turbulenta), se recurrió a otra alternativa. Los datos experimentales del perfil de velocidades turbulento adimensionalizado obtenido por la N.A.S.A, para el estudio de Backward Facing Step, y la posterior validación de distintos modelos turbulentos (NASA-Jhon.W.Slater – Backward Facing Step) han sido incorporados numéricamente en funciones programadas en Fortran 90, por el instituto M.A.R.A.T.EC (M.A.R.A.T.EC – Workshops on CFD Uncertainty Analysis), que pueden ser encontradas para su uso libre en http://maretec.ist.utl.pt/html_files/CFD_workshops. Estas funciones, de inlet, para la velocidad, como también para las demás variables de los modelos turbulentos, fueron utilizadas y recompiladas en una librería de OpenFoam y luego utilizados en los archivos de las condiciones de borde. Esta nueva librería es denominada `libSATurbulentInletVelocity.so`, y contiene la condición de borde de velocidad para la entrada, la cual se denomina `SATurbulentInletVelocity`.
- **Outlet:** la condición de borde para la velocidad, correspondiente a la salida del flujo, se designó como la condición estándar de OpenFoam, conocida como zeroGradient, la cual indica que el gradiente, o lo que es lo mismo, la derivada de la variable a la cual se aplica esta condición, con respecto al vector normal a la superficie libre, es nulo.

- **UpperWall y lowerWall:** Para ambos casos se utilizó una condición estándar de noSlip, la cual garantiza la condición de no deslizamiento para el fluido en las paredes físicas del dominio.
- **FrontAndBack:** los laterales del dominio computacional, son configurados con una condición de borde estándar de OpenFoam, denominada empty. Esta condición de borde reduce una dimensión del dominio computacional, y evita resolver los cálculos en la dirección correspondiente a la normal a dicha superficie, haciendo que el problema pase a ser bidimensional.
- **Condición inicial para el campo interno:** La velocidad se especifica para la iteración cero en 0m/s.

Presión

- **Inlet:** la condición de borde para la presión especificada en el patch de entrada, es especificada como zeroGradient, ya que en dicha superficie se especifican valores de velocidad
- **Outlet:** Un valor de presión es especificado en el patch de salida de flujo, por no estar especificada la velocidad. Una presión uniforme para toda la superficie de 0Pa es configurada.
- **UpperWall y lowerWall:** Para ambos casos se utilizó una condición estándar de zeroGradient.
- **FrontAndBack:** los laterales del dominio computacional, son configurados con una condición de borde estándar de OpenFoam, denominada empty. Esta condición de borde reduce una dimensión del dominio computacional, y evita resolver los cálculos en la dirección correspondiente a la normal a dicha superficie, haciendo que el problema pase a ser bidimensional.
- **Condición inicial para el campo interno:** La presión se especifica para la iteración cero en 0Pa.

Energía cinética de la turbulencia k.

- **Inlet:** De forma similar a la condición de borde de entrada para la velocidad, funciones derivadas de datos experimentales, son compiladas en la librería libSATurbulentInletVelocity.so, y luego utilizadas para la simulación numérica. Para este caso, la condición de borde es denominada `TurbulentInletVelocityk`.

- **Outlet:** se utiliza una condición estándar de zeroGradient.
- **UpperWall y lowerWall:** Para las paredes, una condición de borde estándar de OpenFoam, conocida como wallFunction es utilizada. Esta condición de borde, genera en la cercanía de la pared una variación logarítmica de la variable, desde el valor cero que se tiene en la pared hasta el valor del campo interno de flujo. Esta variación, intenta recrear la realidad y dar estabilidad a los cálculos, dando una variación del estilo del perfil de velocidades turbulento, ya que, estas variables, toman su valor en base a la cantidad de turbulencia que exista en el flujo y las relaciona con la velocidad media del flujo y la tensión de corte en la pared.
- **FrontAndBack:** Nuevamente se utiliza la condición estándar empty.
- **Condición inicial para el campo interno:** Para garantizar la estabilidad de los cálculos, la condición inicial de la energía cinética turbulenta para el campo interno del flujo, se adopta en base a las fórmulas de condiciones iniciales para flujo interno, descritas en Introduction to Computational Fluid Dinamycs – W. Malalasekera – pág. 77. De aquí, la energía cinética turbulenta puede ser calculada según la fórmula 3.1.

$$k = \frac{3}{2}(U_{ref}.T_i)^2 \quad (5.1)$$

donde T_i, es la intensidad de energía cinética turbulenta, la cual equivale a la media cuadrática, o desviación estándar, de la suma de las fluctuaciones de la velocidad, adimensionalizado por la velocidad de referencia, como indica la fórmula 3.2.

$$T_i = \frac{u'}{U_{ref}} \quad (5.2)$$

Asumiendo isotropía de modo tal que $u'_x = u'_y = u'_z = 0.05.U_{ref}$:

$$k = \frac{3}{2}(1x0.05)^2 = 0.00375J$$

Disipación de energía de la turbulencia epsilon.

- **Inlet:** De forma similar a la condición de borde de entrada para la velocidad, funciones derivadas de datos experimentales, son compiladas en la librería libSATurbulentInletVelocity.so, y luego utilizadas para la simulación numérica. Para este caso, la condición de borde es denominada `EpsilonTurbulentInlet`.
- **Outlet:** se utiliza una condición estándar de zeroGradient.

- **UpperWall y lowerWall:** la wallFunction específica para épsilon, es utilizada de forma similar que para la energía cinética turbulenta k.
- **FrontAndBack:** Nuevamente se utiliza la condición estándar empty.
- **Condición inicial para el campo interno:** De forma similar, a la energía cinética turbulenta k, se calcula el valor de épsilon para el campo interno, utilizando la fórmula 3.3.

$$\varepsilon = C_{\mu}^{3/4} \frac{k^{3/2}}{l} \tag{5.3}$$

Donde C_{μ} es una constante empírica cuyo valor es 0.09 y l es 0.07 veces la longitud característica, que para este caso equivale a la altura del escalón adimensionalizado, entonces:

$$\varepsilon = 0.09^{3/4} \frac{0.00375^{3/2}}{0.07} = 0.000539j$$

Frecuencia de la turbulencia omega.

- **Inlet:** De forma similar a la condición de borde de entrada para la velocidad, funciones derivadas de datos experimentales, son compiladas en la librería libSATurbulentInletVelocity.so, y luego utilizadas para la simulación numérica. Para este caso, la condición de borde es denominada `omegaTurbulentInlet`.
- **Outlet:** se utiliza una condición estándar de zeroGradient.
- **UpperWall y lowerWall:** la wallFunction específica para épsilon, es utilizada de forma similar que para la energía cinética turbulenta k.
- **FrontAndBack:** Nuevamente se utiliza la condición estándar empty.
- **Condición inicial para el campo interno:** el valor de la frecuencia de la turbulencia para el campo interno es calculado en base a la fórmula 3.4.

$$\omega = \varepsilon / k \tag{5.4}$$

Entonces:

$$\omega = \frac{0.000539j}{0.00375J} = 0.4437$$

5.4.3 Configuración numérica restante

Una vez configurados los archivos de mallado y condiciones de borde, el archivo controlDict debe ser configurado. Este archivo administra la configuración de la cantidad de iteraciones a realizar, el paso de tiempo mediante el cual se resuelven las ecuaciones, y los tiempos para los cuales, archivos de salida son generados.

Para las simulaciones a realizar, se configura la cantidad de 2000 iteraciones, esperando que la convergencia del resultado se produzca antes de llegar a dicha cifra, con un paso de tiempo igual a la unidad. En este archivo, el cual puede encontrarse en APENDICE C, también se especifica el nombre de la librería compilada para la utilización de las condiciones de borde en la entrada.

En el archivo fvsolution, se especifican las magnitudes de los residuos que definen la convergencia. Para la presión se utiliza control residual de 10^{-2} y 10^{-3} para la velocidad.

Por último, los esquemas de discretización a utilizar (en las simulaciones upwind o QUICK para los términos convectivos de la ecuación de transporte), son especificados en el archivo fvschemes. En este archivo, también se configura cada simulación para obtener un resultado de flujo estacionario, y que los pasos de tiempos, sirvan únicamente como iteraciones para la resolución de las ecuaciones numéricas.

5.5 Resultados de simulaciones numéricas

Una vez configurados los parámetros numéricos, las simulaciones son realizadas, mediante el solver simpleFoam para los 30 casos descritos. Dentro de los resultados, son obtenidos simulaciones en su mayoría resultados convergentes. Una vez que una simulación es finalizada, archivos con los resultados numéricos de la presión y velocidad son generados, los cuales pueden ser visualizados en paraView. En la figura 5.8, 5.9, y 5.10, pueden observarse tres resultados convergentes para los tres modelos de turbulencia distintos, en donde pueden apreciarse diferencias significativas, variando únicamente el modelo de turbulencia, sobre todo en la transferencia de momento de la capa límite inferior.

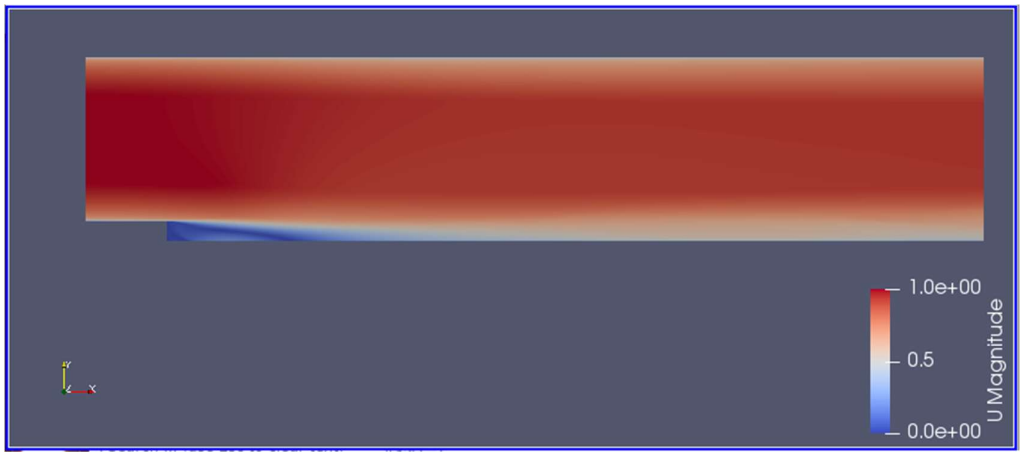

Figura 5.8 – Visualización en paraView de simulación convergente obtenida con modelo kwSST , en la malla m4, con un refinamiento de 1.2 y sistema de discretización upwind

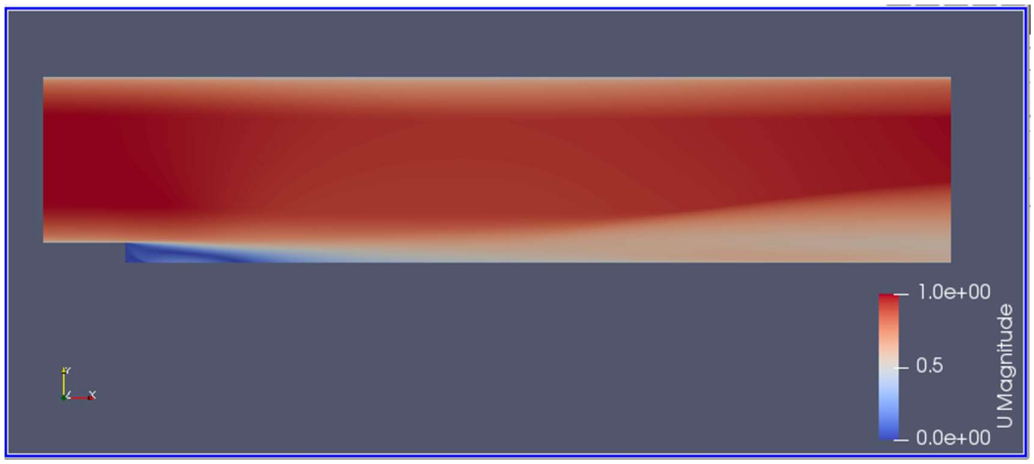

Figura 5.9 – Visualización en paraView de simulación convergente obtenida con modelo kw, en la malla m4, con un refinamiento de 1.2 y sistema de discretización upwind

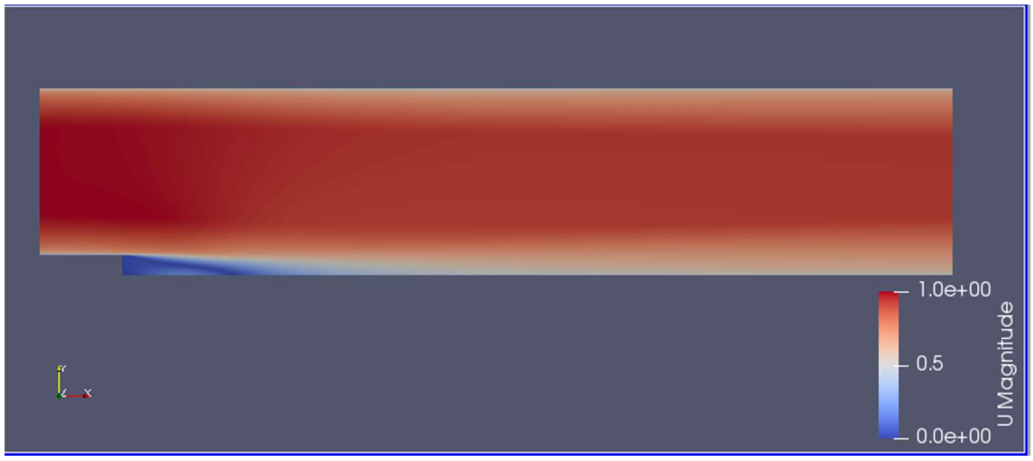

Figura 5.10 – Visualización en paraView de simulación convergente obtenida con modelo kepsilón, en la malla m4, con un refinamiento de 1.2 y sistema de discretización upwind

En la figura 5.11, puede apreciarse la traza de líneas de corrientes obtenido de una de las simulaciones, en donde puede observarse el flujo recirculante que se produce debido al desprendimiento del flujo al atravesar la sección que corresponde al escalón.

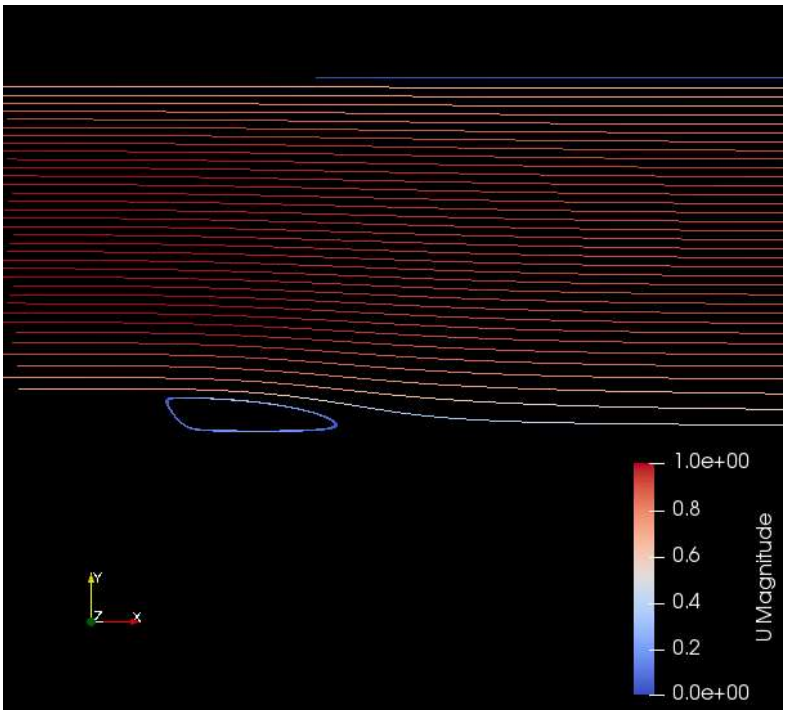

Figura 5.11 – Visualización en paraView de las líneas de corriente del flujo, mediante una simulación convergente obtenida con modelo kw, en la malla m4, con un refinamiento de 1.2 y sistema de discretización upwind.

Luego de concluir con las 30 simulaciones, para cada una de ellas, los coeficientes de fricción C_f, en la pared superior e inferior, el coeficiente de presión C_p, en la pared inferior, y el punto de reinserción del flujo son calculados, mediante el post – procesador de datos paraView, que permite calcular los valores de presiones y tensiones viscosas, para luego procesarlas e integrarlas sobre una superficie para obtener los valores de dichos coeficientes, o bien para encontrar el punto en el cual las tensiones viscosas en la pared inferior cambian de signo, el cual se identifica como punto de reinserción del flujo mejor conocido como reattachment length. Este último valor, equivale a la coordenada x posterior al escalón, para el cual el flujo vuelve a apegarse a la pared. Detrás de este punto, las velocidades en la cercanía de la pared serán negativas debido a la recirculación, y posterior a este positivas, por ende, las tensiones viscosas cambian de signo, como se muestra en la figura 5.12, esta variable fue calculada únicamente para grupo de mallas número 2.

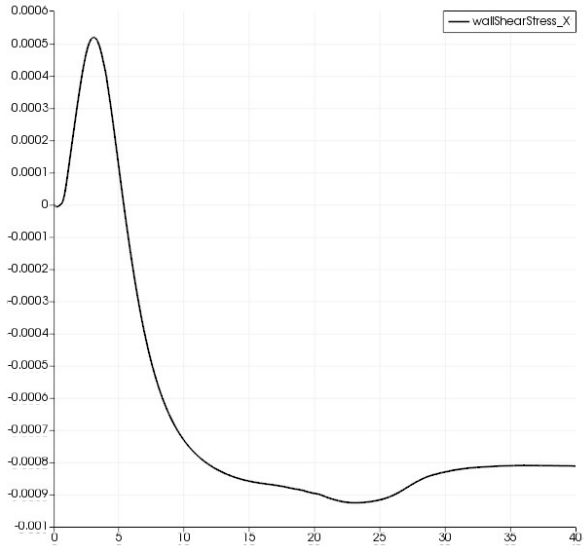

Figura 5.12 – Visualización en paraView de las tensiones en la pared inferior en función de la coordenada, mediante una simulación convergente obtenida con modelo kw, en la malla m4, con un refinamiento de 1.2 y sistema de discretización upwind.

A continuación, se presentan los resultados obtenidos para cada una de las simulaciones en función del modelo de turbulencia, en las tablas 5.1, 5.2 y 5.3. A su vez, en la tabla 5.4 se presentan algunos resultados obtenidos por otros autores para comparación.

Modelo kwSST						
	Malla	**Convergencia**	**Cp - lowerWall**	**Cf - upperWall**	**Cf - lowerWall**	**Reattachment Length**
Grupo 1 **Factor de refinamiento** **1.2** **Esquema de discretización upwind para términos convectivos**	Malla 1	309 iteraciones	0.093521	0.040608	0.021269	---
	Malla 2	404 iteraciones	0.0927464	0.0404896	0.0221782	---
	Malla 3	532 iteraciones	0.0934712	0.0404578	0.0230962	---
	Malla 4	685 iteraciones	0.0921054	0.0404584	0.0236293	---

	Malla 5	886 iteraciones	0.0921603	0.0404584	0.023852	---
Grupo 2 **Factor de refinamiento** **1.2** **Esquema de discretización upwind para términos convectivos**	Malla 1	290 iteraciones	0.101252	0.0462155	0.0280159	5.9332
	Malla 2	320 iteraciones	0.101202	0.0462208	0.0278704	5.9924
	Malla 3	349 iteraciones	0.101079	0.0462602	0.0276941	6.0476
	Malla 4	384 iteraciones	0.100949	0.0463157	0.0274538	6.0876
	Malla 5	429 iteraciones	0.101093	0.0461664	0.0268923	6.1028

Tabla 5.1 – Resultados de simulaciones obtenidas con modelo de turbulencia kwSST

Modelo k-épsilon						
	Malla	**Convergencia**	**Cp - lowerWall**	**Cf - upperWall**	**Cf - lowerWall**	**Reattachment Length**
Grupo 1 **Factor de refinamiento** **1.2** **Esquema de discretización upwind para términos convectivos**	Malla 1	268 iteraciones	0.104835	0.0433916	0.0304532	---
	Malla 2	353 iteraciones	0.103687	0.0430197	0.0304295	---
	Malla 3	470 iteraciones	0.102051	0.0429384	0.0341082	---
	Malla 4	No - 2000	0.101773	0.0429164	0.0391538	---

	Malla 5	751 iteraciones	0.091064	0.0428445	0.0402436	---
Grupo 2 **Factor de refinamiento** **1.2** **Esquema de discretización upwind para términos convectivos**	Malla 1	190 iteraciones	0.103772	0.0467277	0.0315899	5.3368
	Malla 2	207 iteraciones	0.103602	0.0442546	0.0309850	5.3820
	Malla 3	219 iteraciones	0.102320	0.0435615	0.0304623	5.4228
	Malla 4	274 iteraciones	0.102274	0.0429454	0.0301935	5.4560
	Malla 5	345 iteraciones	0.102105	0.0421398	0.0300627	5.4644

Tabla 5.2– Resultados de simulaciones obtenidas con modelo de turbulencia k-épsilon.

Modelo kw						
		Convergencia	**Cp - lowerWall**	**Cf - upperWall**	**Cf - lowerWall**	**Reattachment Length**
Grupo 1 **Factor de refinamiento** **1.2** **Esquema de discretización upwind para términos convectivos**	Malla 1	411 iteraciones	0.0894148	0.0415237	0.0261189	---
	Malla 2	510 iteraciones	0.0774614	0.0420725	0.0264492	---
	Malla 3	635 iteraciones	0.0629274	0.0428683	0.0282535	---
	Malla 4	803 iteraciones	0.0514656	0.0436349	0.0298008	---

	Malla 5	1007 iteraciones	0.0409204	0.0445117	0.0310418	---
Grupo 2 **Factor de refinamiento** **1.2** **Esquema de discretización upwind para términos convectivos**	Malla 1	339 iteraciones	0.107154	0.0418290	0.0276698	5.5996
	Malla 2	370 iteraciones	0.107560	0.0418825	0.0278425	5.6476
	Malla 3	405 iteraciones	0.108026	0.0419928	0.0280812	5.6856
	Malla 4	454 iteraciones	0.108188	0.0420921	0.0283236	5.7156
	Malla 5	540 iteraciones	0.108672	0.0420527	0.0282304	5.7348

Tabla 5.3 – Resultados de simulaciones obtenidas con modelo de turbulencia kw.

	Resultados para comparación			
	Cp - lowerWall	**Cf - upperWall**	**Cf - lowerWall**	**Reattachment Length**
L.Eca y M.Hoekstra mediante la utilización del modelo turbulento Spalart Allmaras	0.125	0.0474	0.0262	6.010
L.Eca y M.Hoekstra mediante la utilización del modelo turbulento kwSST	0.0957	0.0487	0.030	5.092

Tabla 5.4 – Resultados de simulaciones obtenidos por otros autores

Los datos obtenidos para cada grupo de cinco grillas, se procesan atreves de un código de Python, el cual contiene embebido el procedimiento de verificación de la solución, cuyo fin es estimar el valor de incerteza de cada valor obtenido en las simulaciones. Este código tiene como entrada, los valores obtenidos de una variable, para cada una de las cinco grillas, y tiene como salida, un valor estimado para la variable, una desviación estándar, y un gráfico que indica el grado de incerteza para cada uno de los valores obtenidos en las simulaciones. El código puede observarse en el apéndice D.

El código toma los valores de entrada de las variables a analizar, y los valores de ancho de celda correspondiente, con los cuales resuelve los sistemas de ecuaciones 2.105, 2.106 y 2.107 para el caso ponderado y sin ponderar. De la resolución de estas ecuaciones, se obtiene un valor estimado de la variable de análisis, y las variables para la función de regresión del error correspondiente a la fórmula 2.86. Si ambos valores de convergencia de grilla p, se encuentran entre 0.5 y 2, entonces se toma el ajuste con menor desviación estándar. En caso de que una de los dos ajustes tenga el valor de p entre estos valores, ese ajuste es tomado y de caso contrario, se recurre a las otras fórmulas de ajuste por regresión para el error. Es decir, si $p<0.5$ o $p>2$, se resuelven los sistemas de ecuaciones, que derivan de la regresión por mínimo cuadrado, para cada caso de las formulas 2.90, 2.91 y 2.92. Luego, se calcula la desviación estándar para cada ajuste, y la menor de estas es adoptado como función error.

Una vez obtenida la función error, el código calcula un factor de seguridad en base a la magnitud de la desviación estándar, y el valor de convergencia de grilla p. Luego, en base al factor de seguridad obtenido, se utiliza una fórmula para calcular la incerteza, en base al rango de dispersión de los resultados, la desviación estándar, el error obtenido para un tamaño de celda dado, y los valores obtenidos de la simulación de la variable y el estimado, con los cuales el código devuelve el correspondiente grafico de incerteza.

En las figuras 5.13 a 5.33, se presentan las medidas de incertezas obtenidas para cada una de las simulaciones.

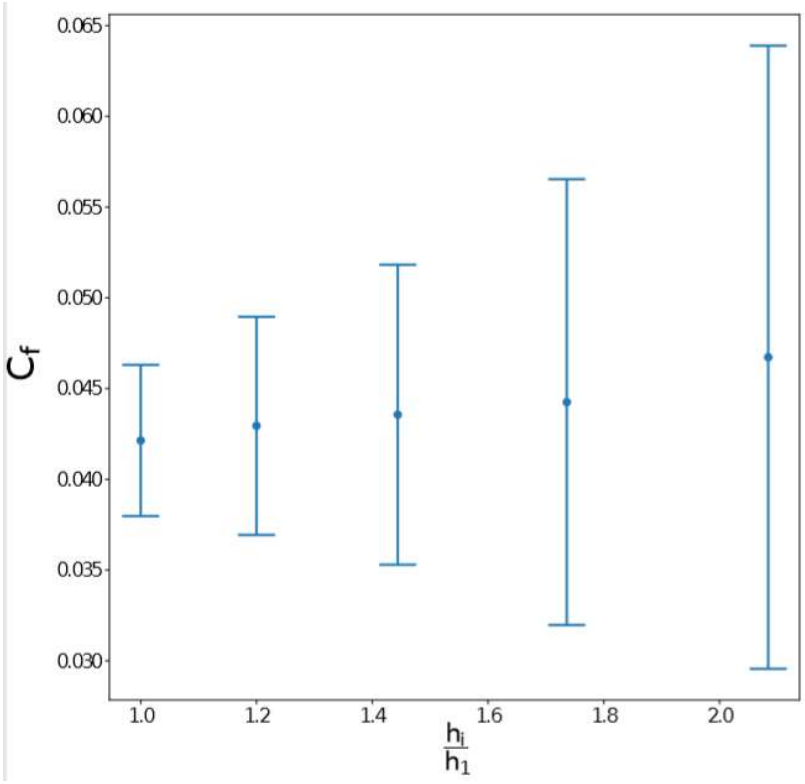

Figura 5.13 – Convergencia de coeficiente de resistencia en la pared superior – modelo kepsilón – malla con factor de refinamiento 1.2 – Grupo2

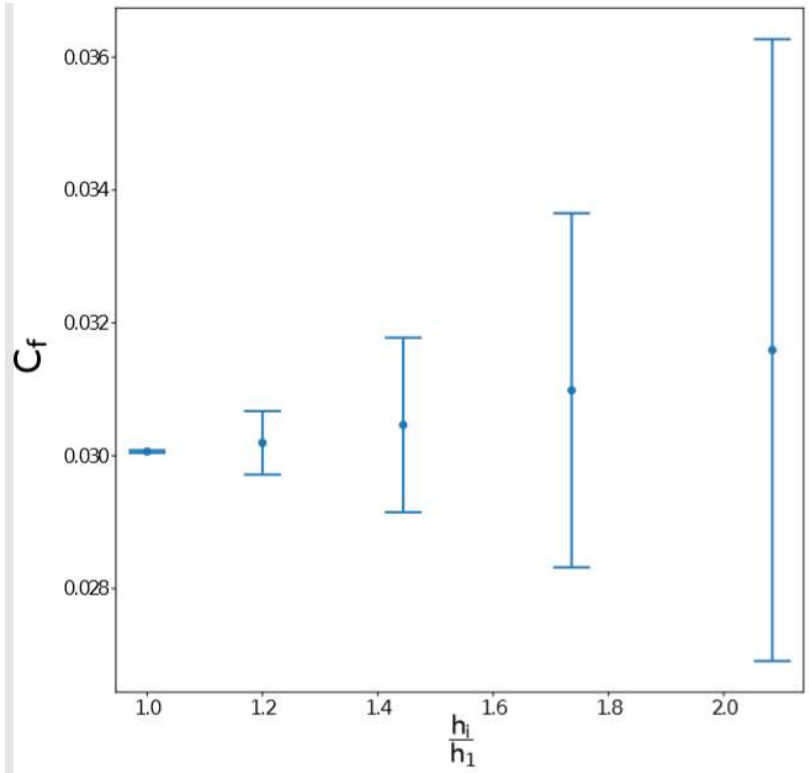

Figura 5.14 – Convergencia de coeficiente de resistencia en la pared inferior – modelo kepsilón – malla con factor de refinamiento 1.2 – Grupo 2

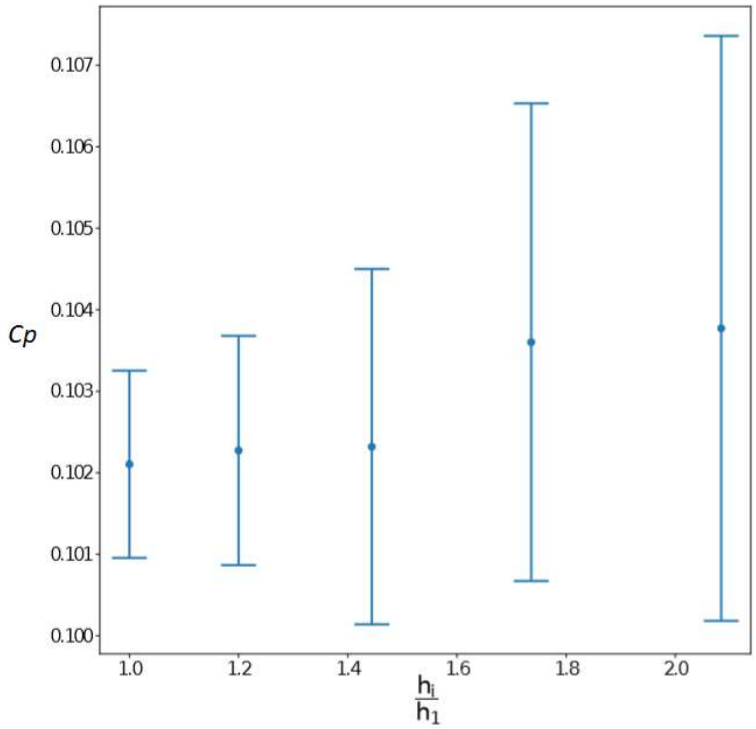

Figura 5.15 – Convergencia de coeficiente de presión en la pared inferior – modelo kepsilón – malla con factor de refinamiento 1.2-grupo 2

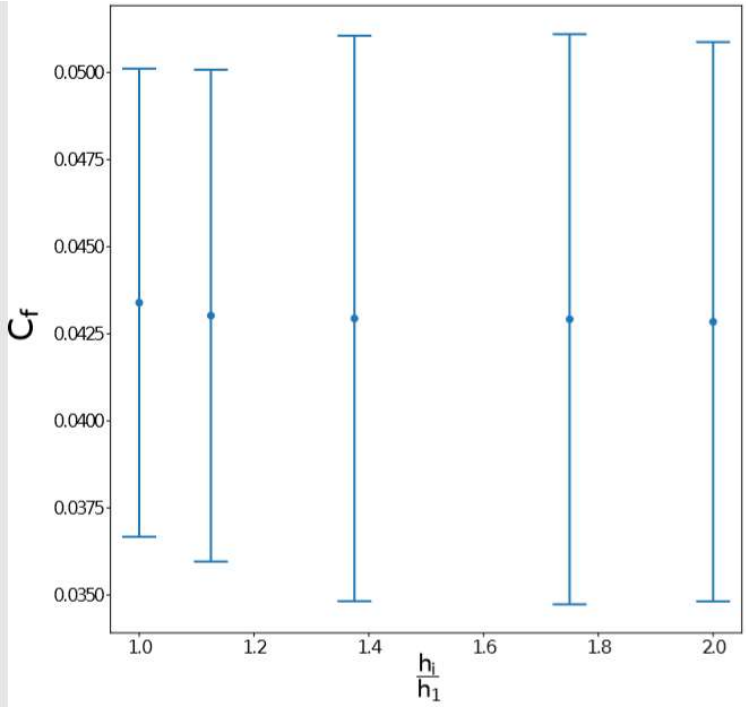

Figura 5.16 – Convergencia de coeficiente de fricción en la pared superior – modelo kepsilón – malla con factor de refinamiento 1.2- Grupo 1

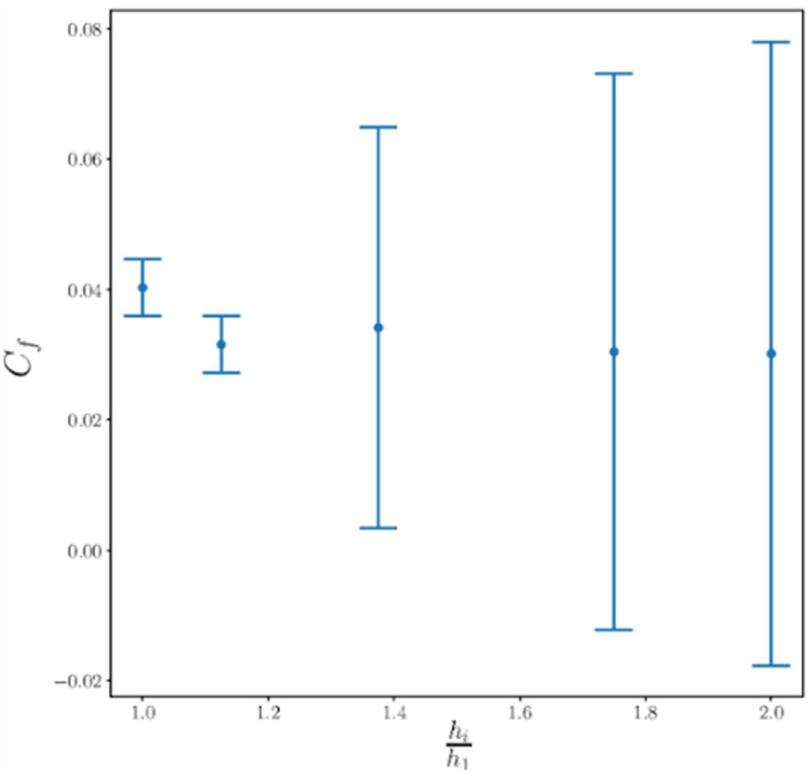

Figura 5.17 – Convergencia de coeficiente de fricción en la pared inferior – modelo kepsilón – malla con factor de refinamiento 1.2- Grupo 1

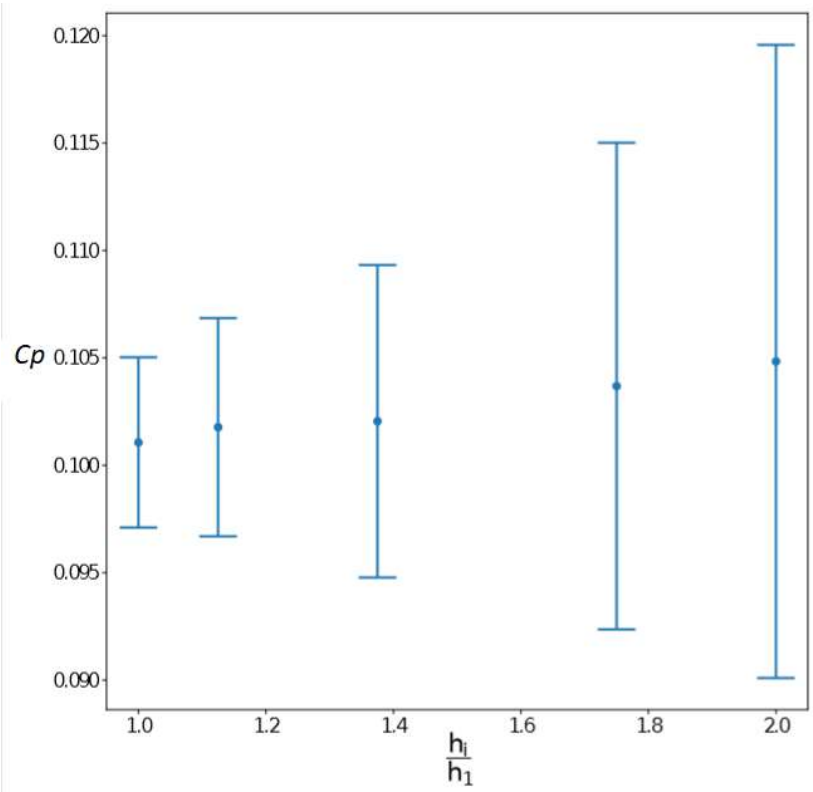

Figura 5.18 – Convergencia de coeficiente de presión en la pared inferior – modelo kepsilón – malla con factor de refinamiento 1.2- Grupo 1

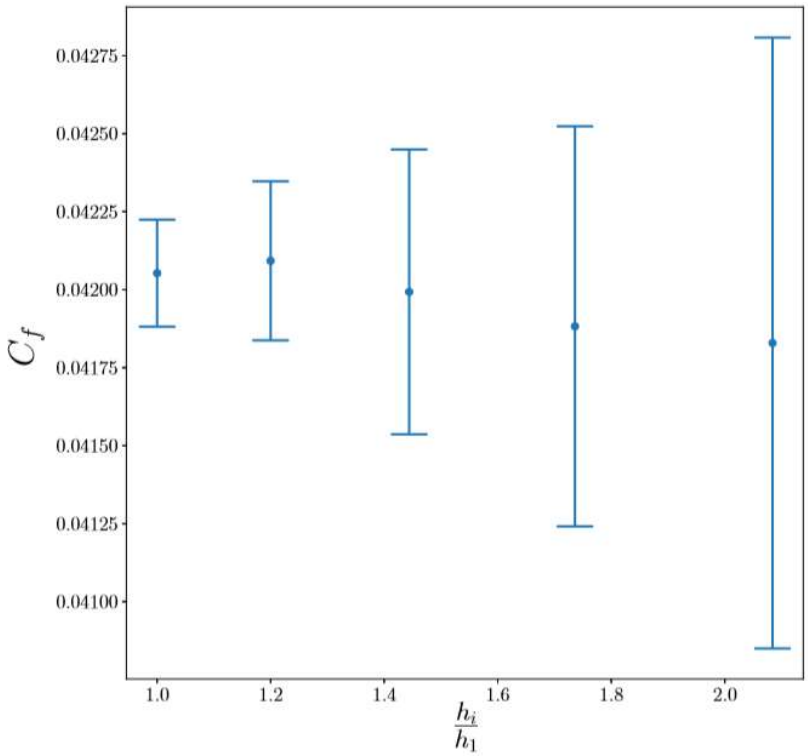

Figura 5.19 – Convergencia de coeficiente de fricción en la pared superior – modelo kw – malla con factor de refinamiento 1.2- Grupo 2

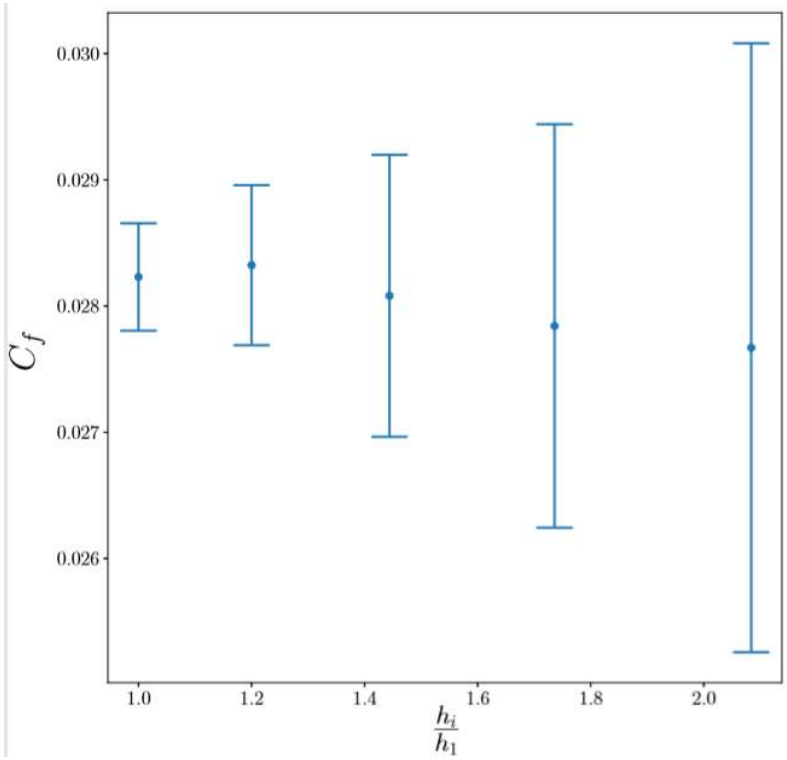

Figura 5.20 – Convergencia de coeficiente de fricción en la pared inferior – modelo kw – malla con factor de refinamiento 1.2 - Grupo 2.

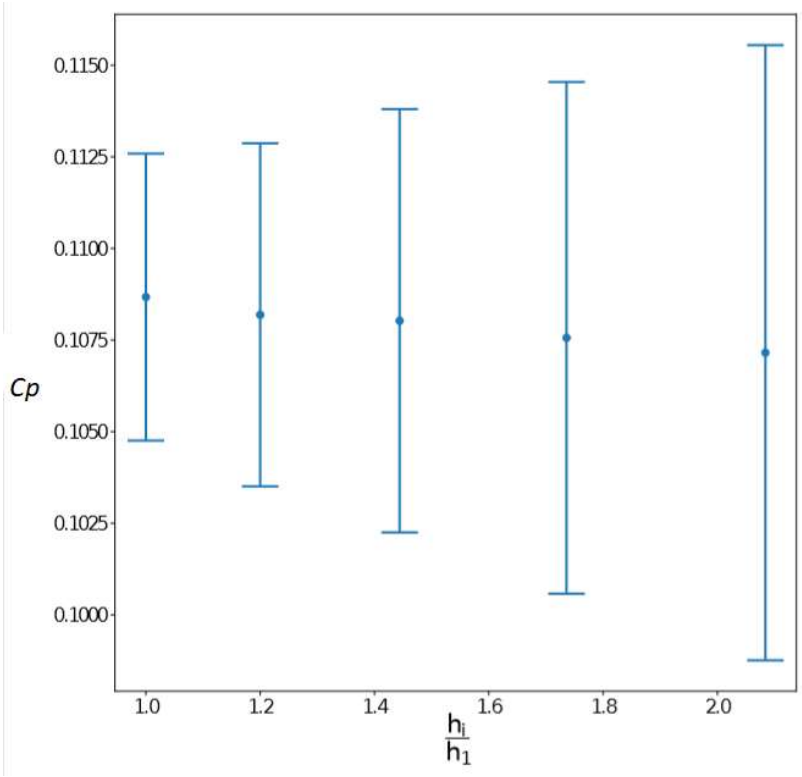

Figura 5.21 – Convergencia de coeficiente de presión en la pared inferior – modelo kw – malla con factor de refinamiento 1.2- Grupo 2

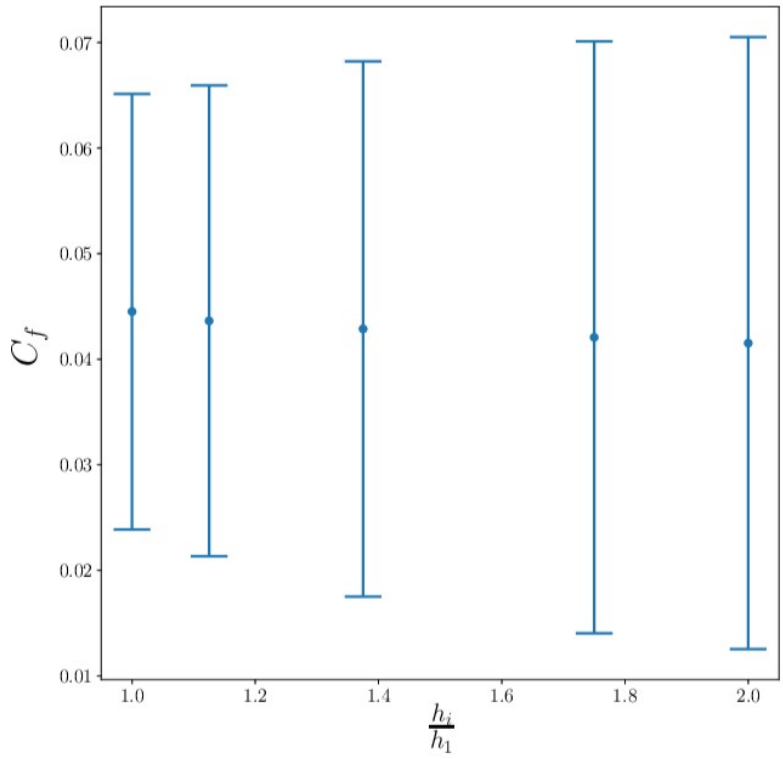

Figura 5.22 – Convergencia de coeficiente de resistencia en la pared superior – modelo kw – malla con factor de refinamiento 1.2 – Grupo 1

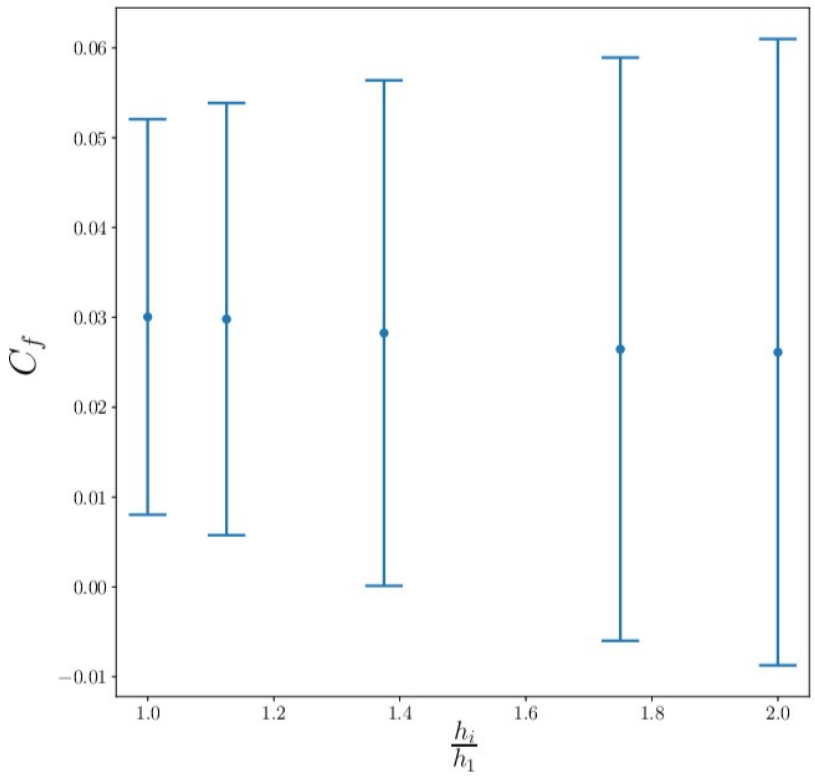

Figura 5.23 – Convergencia de coeficiente de resistencia en la pared inferior – modelo kw – malla con factor de refinamiento 1.2 – Grupo1

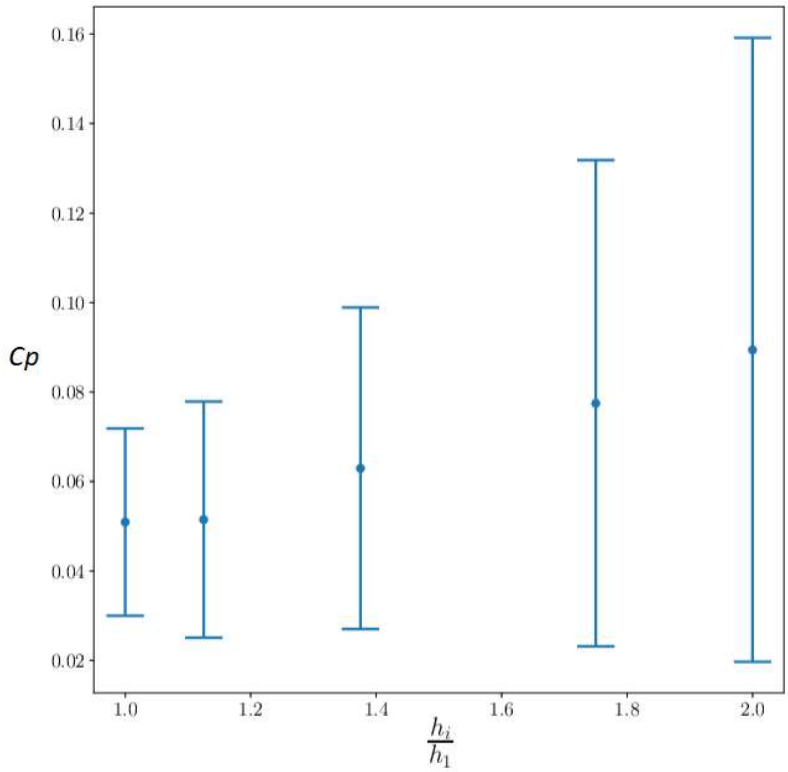

Figura 5.24 – Convergencia de coeficiente de presión en la pared inferior – modelo kw – malla con factor de refinamiento 1.2 – Grupo 1

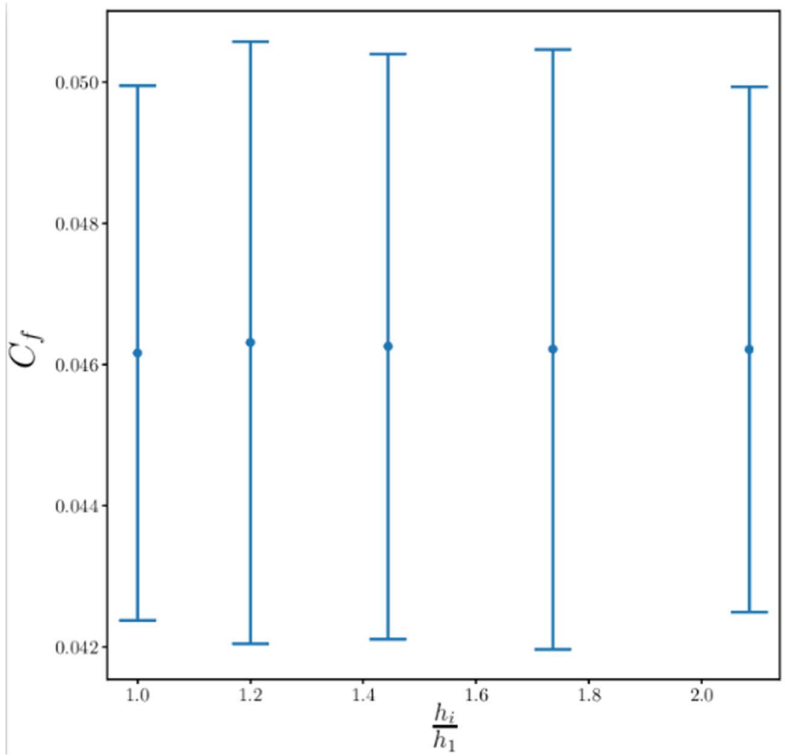

Figura 5.25 – Convergencia de coeficiente de resistencia en la pared superior – modelo kwSST – malla con factor de refinamiento1.2 – Grupo 2

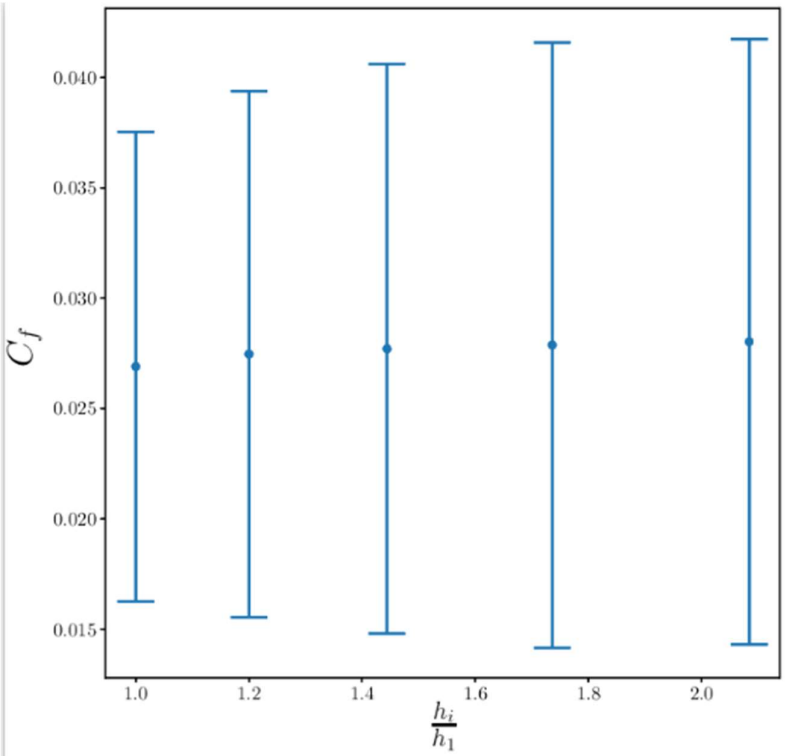

Figura 5.26 – Convergencia de coeficiente de resistencia en la pared inferior – modelo kwSST – malla con factor de refinamiento 1.2 – Grupo 2

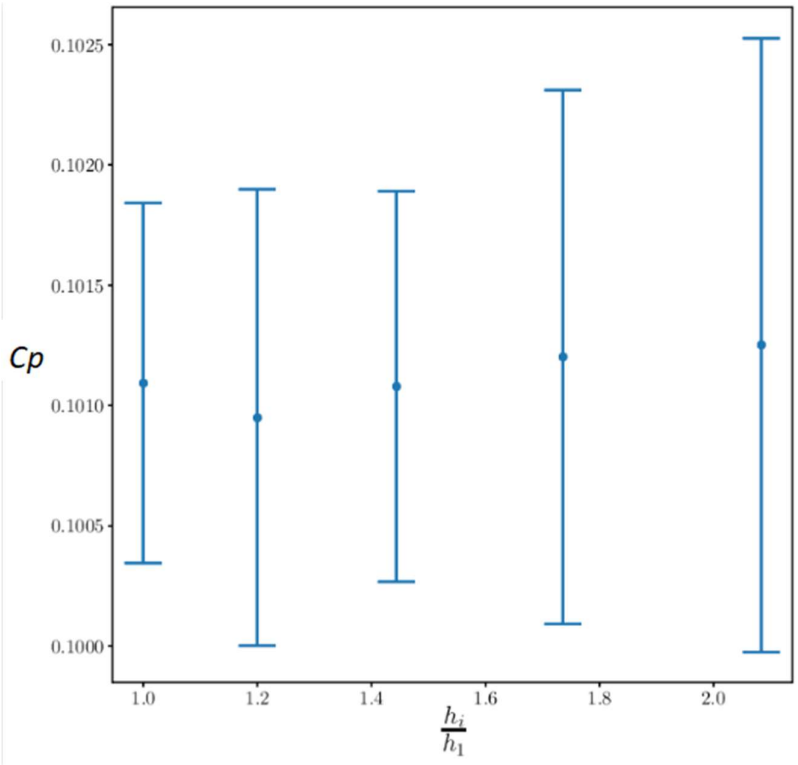

Figura 5.27 – Convergencia de coeficiente de presión en la pared inferior – modelo kwSST – malla con factor de refinamiento 1.2 – Grupo 2

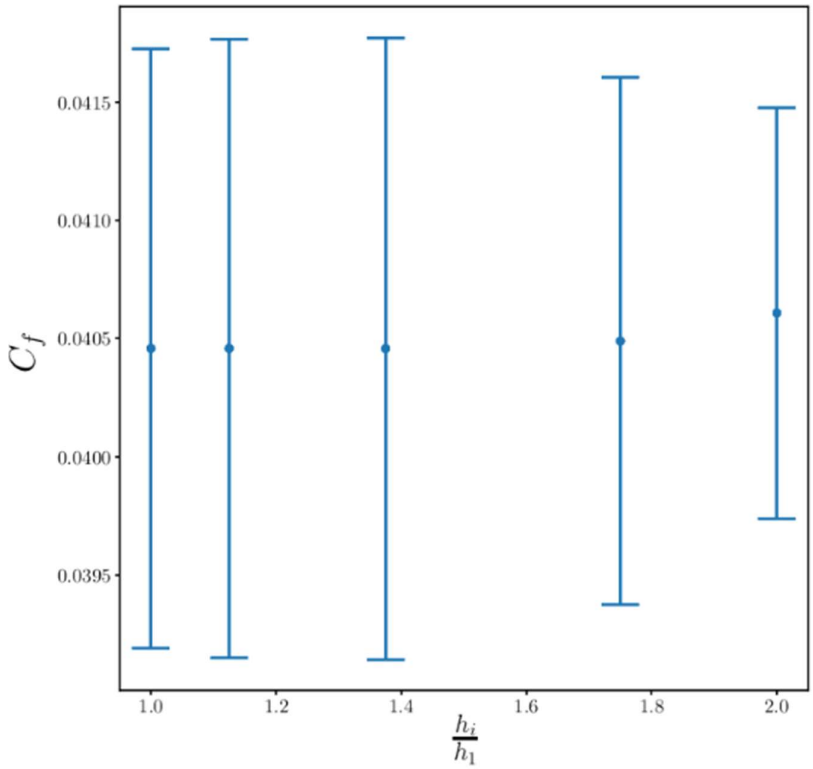

Figura 5.28 – Convergencia de coeficiente de resistencia en la pared superior – modelo kwSST – malla con factor de refinamiento 1.2 – Grupo 1

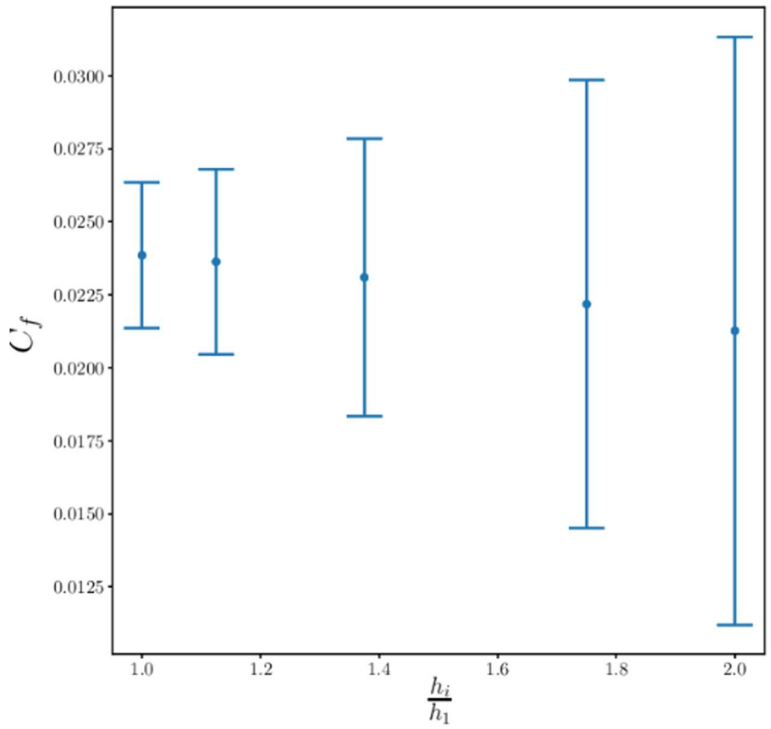

Figura 5.29 – Convergencia de coeficiente de resistencia en la pared inferior – modelo kwSST – malla con factor de refinamiento 1.2 – Grupo 1

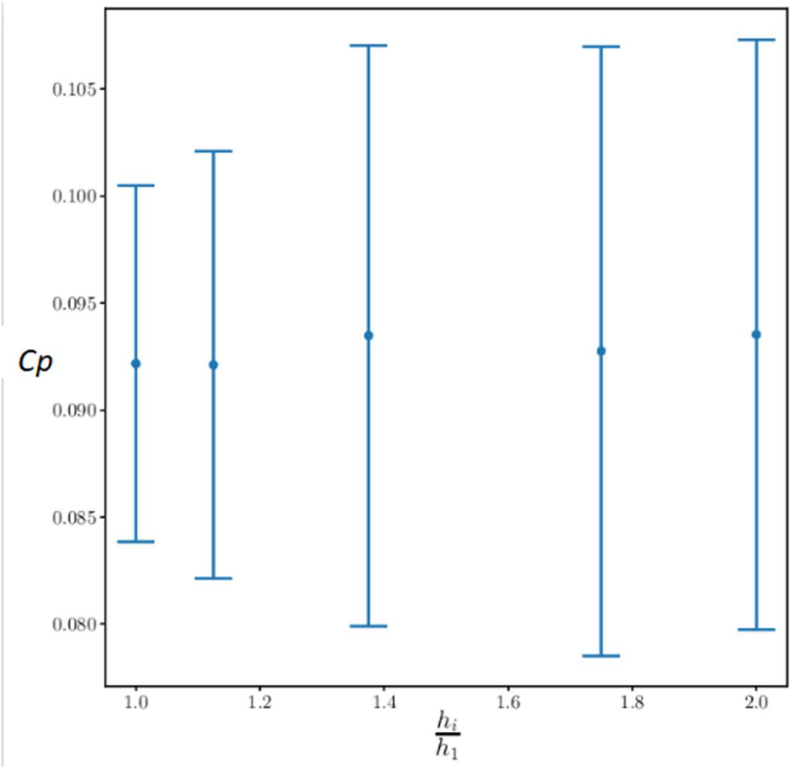

Figura 5.30 – Convergencia de coeficiente de presión en la pared inferior – modelo kwSST – malla con factor de refinamiento 1.2 – Grupo 1

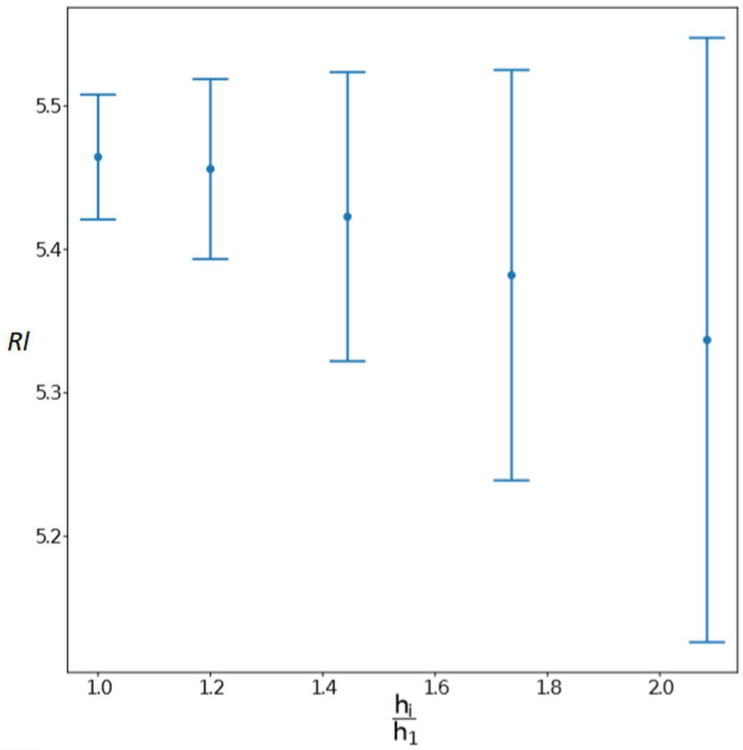

Figura 5.31 – Convergencia de coeficiente de punto de reinserción de flujo en la pared inferior– modelo kepsilón – malla con factor de refinamiento 1.2 – Grupo 2

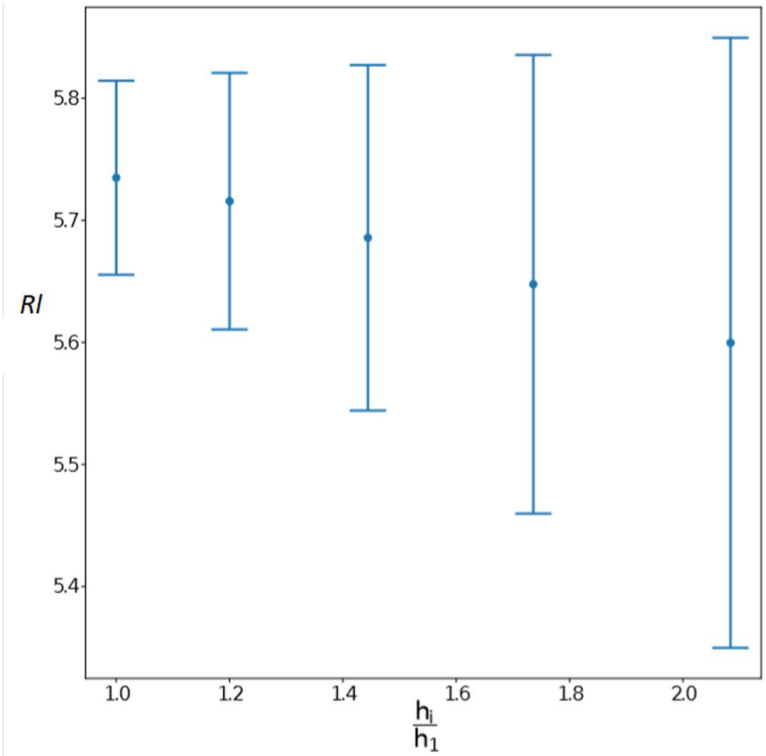

Figura 5.32 – Convergencia de coeficiente de punto de reinserción de flujo en la pared inferior– modelo kw – malla con factor de refinamiento 1.2 – Grupo 2.

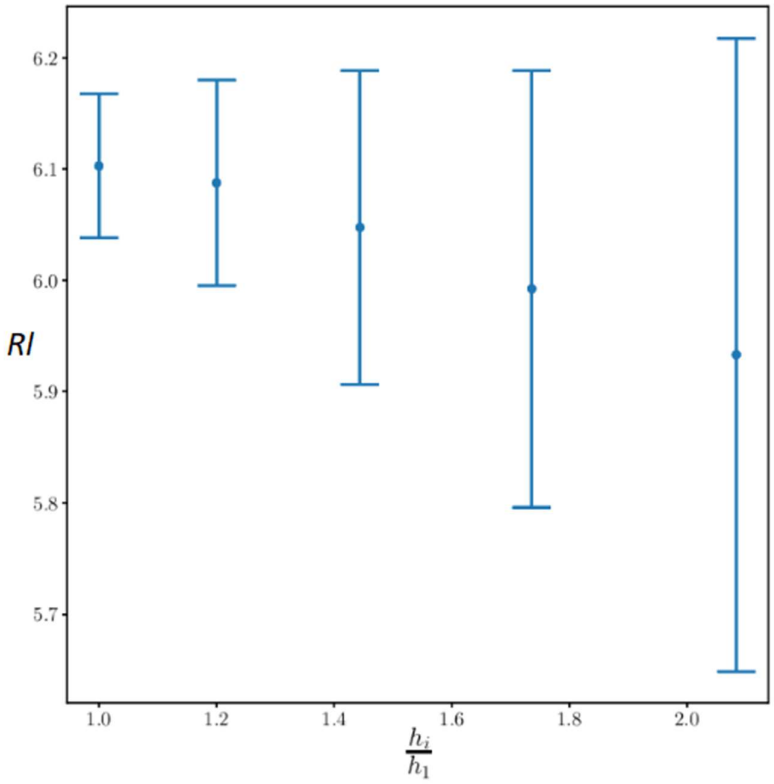

Figura 5.33 – Convergencia de coeficiente de punto de reinserción de flujo en la pared inferior– modelo kwSST – malla con factor de refinamiento 1.2 – Grupo 2

Como dato complementario a las gráficas de incertezas obtenidos para cada grupo de mallas, graficas de residuos de las simulaciones, son adjuntados a continuación en las figuras 5.34 a 5.39, con el fin de mostrar el comportamiento del error numérico en función del número de iteraciones. El corte de las iteraciones, se da en base a la configuración del archivo de OpenFoam fvschemes, en donde se especifica un valor residual de 10^{-2} para la presión y 10^{-3}, para el resto de las variables.

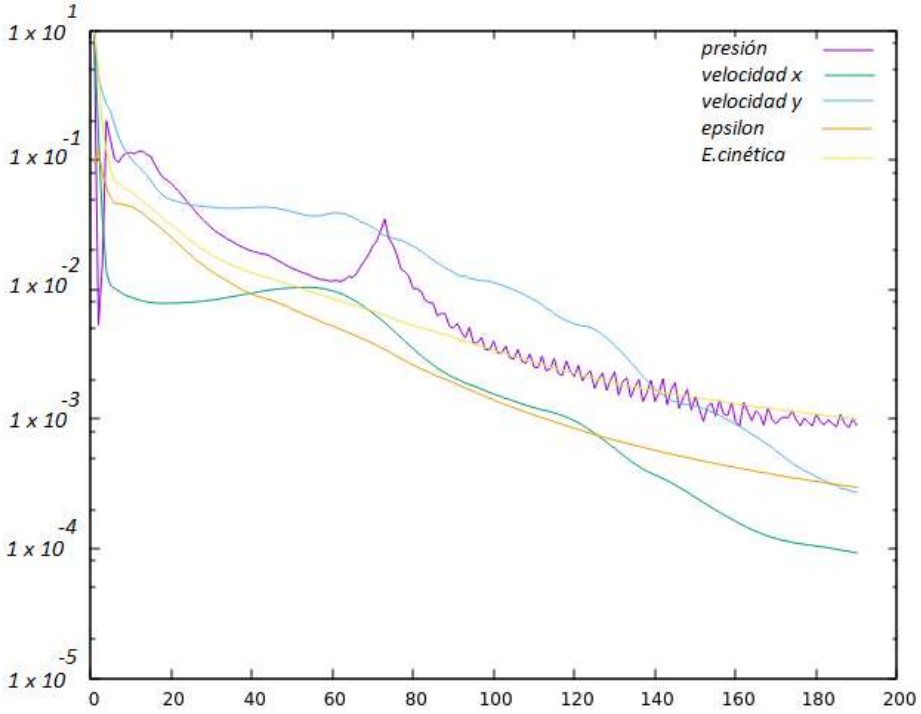

Figura 5.34 – Convergencia de la norma de los residuos en función del número de iteraciones – modelo kepsilón – malla 1 (más gruesa) – Grupo 2

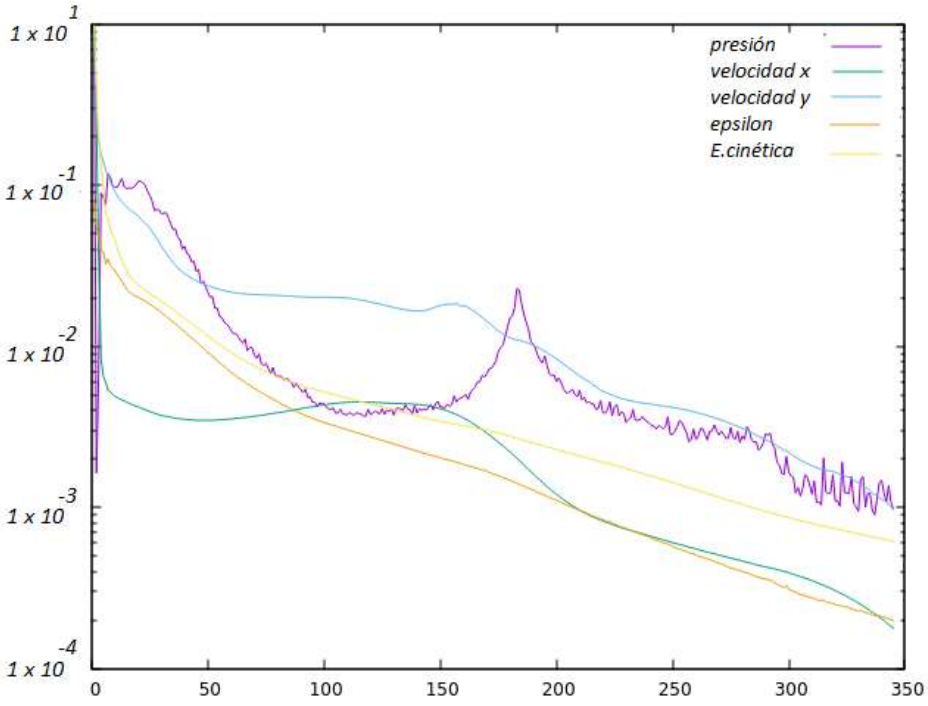

Figura 5.35 – Convergencia de la norma de los residuos en función del número de iteraciones– modelo kepsilón – malla 5 (más fina) – Grupo 2

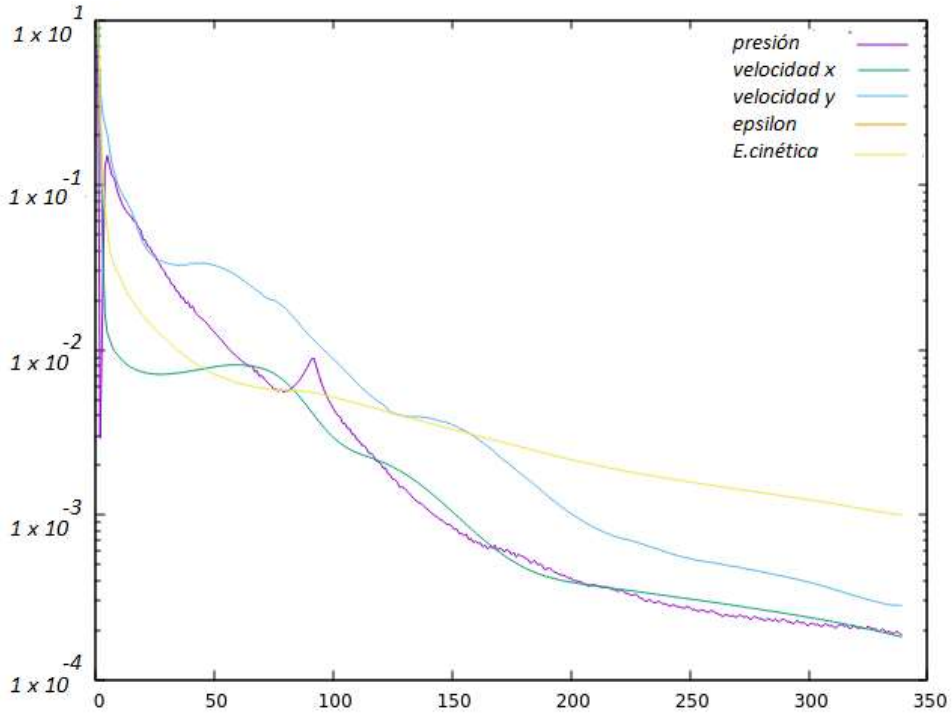

Figura 5.36 – – Convergencia de la norma de los residuos en función del número de iteraciones – modelo kw – malla 1 (más gruesa) – Grupo 2

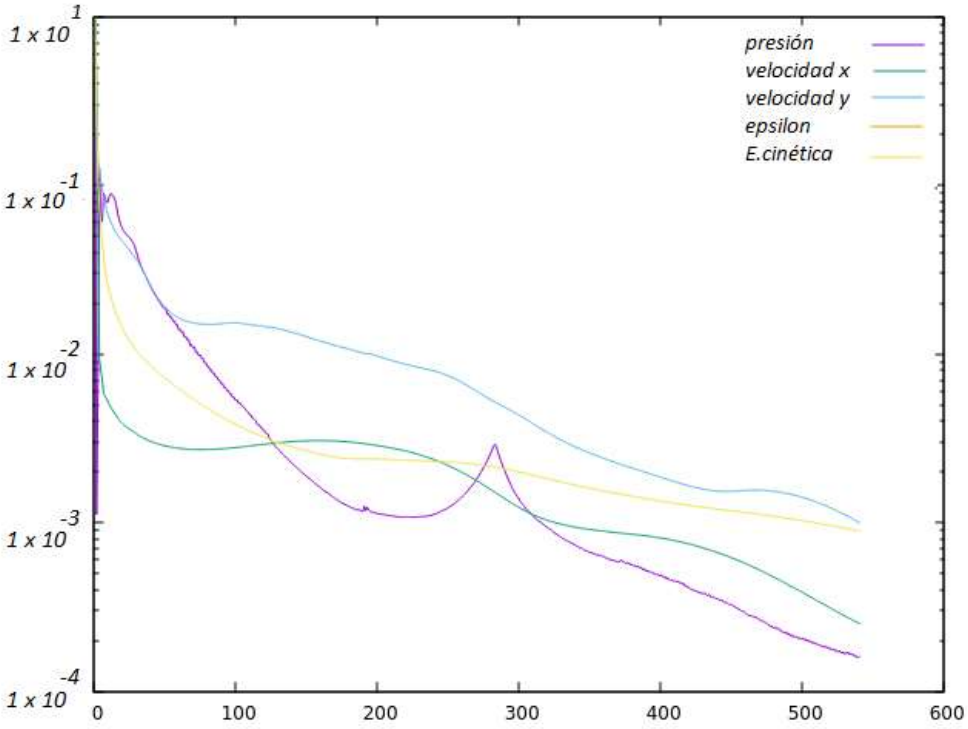

Figura 5.37 – – Convergencia de la norma de los residuos en función del número de iteraciones – modelo kw – malla 5 (más fina) – Grupo 2

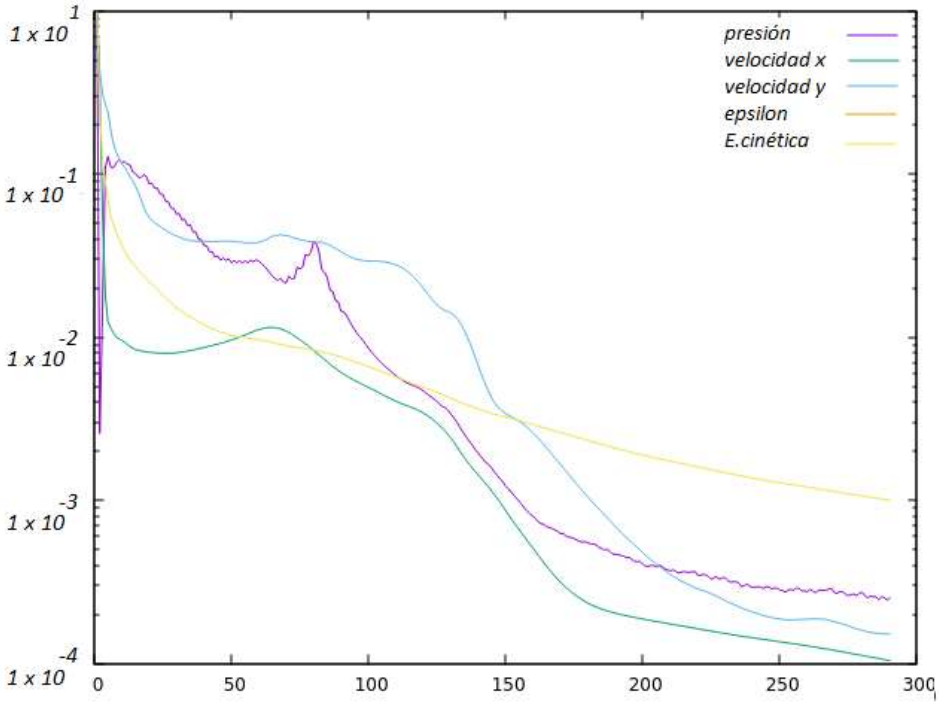

Figura 5.38 – – Convergencia de la norma de los residuos en función del número de iteraciones – modelo kwSST – malla 1 (más gruesa) – Grupo 2

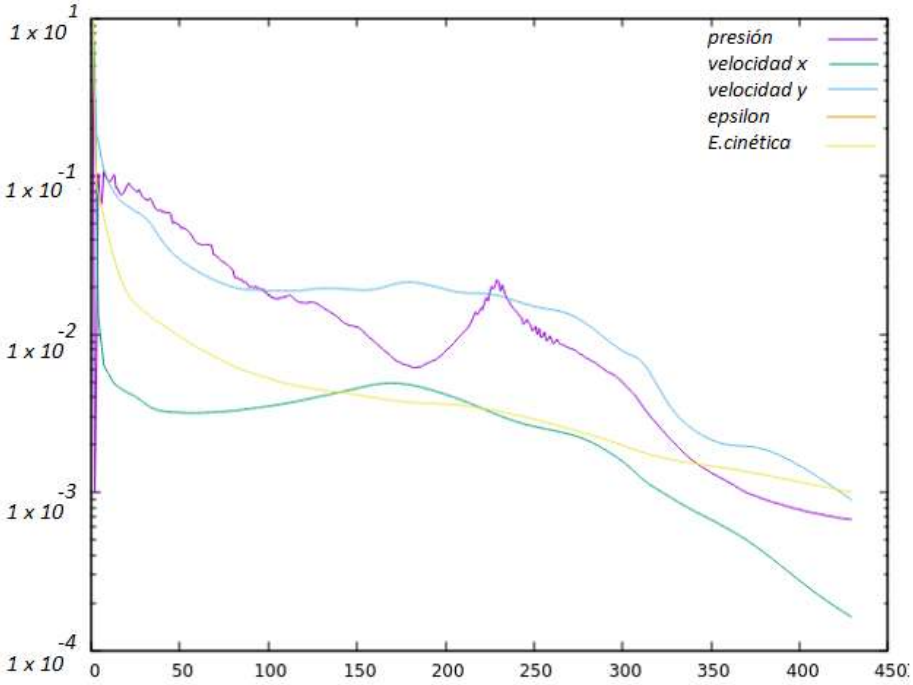

Figura 5.39 – – Convergencia de la norma de los residuos en función del número de iteraciones – modelo kepsilón – malla 5 (más fina) – Grupo 2

5.6 Discusión de los resultados

Los resultados obtenidos del análisis de incertezas, muestran una clara dependencia del modelo de turbulencia y del tamaño de celda. El modelo de turbulencia kw, para el conjunto de mallas ortogonales del grupo 2, muestra los mejores resultados, obteniendo valores de incertezas del orden de 3% para el coeficiente de resistencia de la pared superior, y del orden del 7 y 4% para el coeficiente de presión y de resistencia en la pared inferior. Para el Grupo de mallas 1, este modelo, presenta, por el contrario, los peores resultados, obteniendo valores del orden del 90% para ambos coeficientes de resistencia y del 60% para el coeficiente de presión. El modelo de turbulencia kepsilón, para el conjunto de mallas del grupo 2, muestra órdenes de magnitudes de incertezas del 20% para el coeficiente de resistencia de la pared superior, 2% en el coeficiente de resistencia de la pared inferior y ordenes de magnitud del 1% para el coeficiente de presión. El modelo kepsilón, para el grupo de mallas 1, presenta ordenes de magnitud del orden del 30% para ambos coeficientes de resistencia y del 7.5% para el coeficiente de presión. Para modelo de turbulencia kwSST, para el conjunto de mallas del grupo 2, puede observarse una incerteza del orden del 15% para el coeficiente de resistencia de la pared superior, 70% para el coeficiente de resistencia de la pared inferior y un 5% para el coeficiente de presión de la pared inferior. Mientras que para el grupo de mallas número 1, los resultados de este modelo de turbulencia mejoran, obteniendo alores del 6% para el coeficiente de resistencia de la pared superior, 20% para el coeficiente de resistencia de la pared inferior y 15% para el coeficiente de presión en la pared inferior.

Por otro lado, el punto de inserción del flujo, utilizando el grupo de malla número dos, entrega un valor de incertezas del orden del 2% para todos los modelos de turbulencia, aunque con diferentes valores estimados para dicha variable.

Los resultados evidencian, como algunos modelos de turbulencia, se adaptan mejor, en cuanto a los valores obtenidos de incertezas, a un conjunto de mallas u a otra. Un posible motivo de este hecho, se puede deber a la mejora de precisión, para los modelos de bajos número de Reynolds, como los kw, a la hora de realizar mallas cada vez más refinadas, pero una baja en la calidad de los resultados por parte del modelo Kepsilón, al refinar la malla en demasía, debido

a que esto hace descender el valor y+, lo cual, como se mencionó anteriormente, produce falla en dicho modelo. Por otro lado, la deficiencia de los modelos kw y kwSST, en el conjunto de mallas más grueso, puede asociarse al echo, de que el rango de valores obtenidos, no se encuentre dentro del rango de convergencia asintótico del error de discretización, haciendo que el procedimiento de estimación de incertezas, pierda validez para estos resultados, y puedan considerarse anómalos.

Por último, observando las gráficas de las convergencias de los residuos, es posible identificar como, el modelo Kepsilón, es el que muestra una mayor velocidad de convergencia, seguida por el modelo kwSST y luego el modelo kw.

6.CONCLUSIÓN

En este trabajo, un análisis de incertezas sobre simulaciones CFD, obtenidas mediante el software OpenFoam, fue realizado, con el fin de verificar las soluciones obtenidas, ante la variación de parámetros numéricos. Para esto, 30 simulaciones fueron realizadas, para las cuales, distintos tamaños de celdas, y modelos de turbulencia fueron utilizadas. La obtención de los rangos de incertezas, fueron obtenidas, modelando el error de discretización, mediante series de potencias, y obteniendo las variables de los respectivos ajustes utilizando el método de mínimos cuadrados, para luego calcular el valor de incerteza, a través de multiplicar el error por un factor de seguridad, dependiente del rango de datos disponibles, y las desviaciones estándar de los ajustes. Los resultados obtenidos fueron presentados en gráficas, a través de las cuales se puede observar el comportamiento de convergencia de las incertezas al aplicar el método de refinamiento de grilla.

Los resultados, indican como la elección, del tamaño de grilla, y el modelo de turbulencia, pueden afectar a los resultados obtenidos, por lo que su selección, es de gran importancia a la hora de intentar obtener resultados realistas por medio de simulaciones CFD. Es aquí, donde puede apreciarse, la importancia de la experiencia y conocimientos, del operador, de un código de CFD, como OpenFoam, haciendo que este pase a ser más valioso que el código que se encuentra embebido. Como pueden observarse en los resultados, una mala selección del modelo de turbulencia, o una malla inadecuada, pueden provocar resultados que difieran bastante de la realidad.

Por otro lado, se pone en evidencia, el valor de la verificación de la solución, mediante el análisis de incerteza, ya que es, este método, el que permite inferir, cuando, una solución, es realmente confiable, e incluso, permite, cuantificar, el rango de confiabilidad de dicha solución. Este aspecto, el cual permitió, derivar, en que muchas de las simulaciones realizadas, no puedan ser utilizados como datos confiables, para luego pasar a ser un parámetro de diseño, utilizado en el desarrollo de un proyecto, también, son de importante aplicación, como argumento de venta o presentación. Esto último, se basa en el hecho, de que, en la actualidad, la mayoría del software de diseño, pueden ser vistos como una caja negra, en donde, datos de entrada son colocados, para luego obtener datos de salida, que no poseen respaldo alguno. La verificación de la solución, nace debido a esta deficiencia de las simulaciones numéricas, y su aplicación,

crece en conjunto con el desarrollo del CFD, a tal punto, de que su uso, puede llegar a normalizar, la obtención de resultados, en diversas aplicaciones de ingeniería.

Por último, se propone dirigir este trabajo a futuro, hacia al análisis de incertezas, utilizando OpenFoam, de variables locales, dentro de una cierta región del flujo, como pueden ser valores obtenidos en las simulaciones de presión y velocidad. Para dicha tarea es necesario el desarrollo de un método de interpolación de segundo orden, que permita interpolar datos de una malla a otra, para poder realizar dicho análisis. En la actualidad, OpenFoam, cuenta con métodos de interpolación de primer orden, los cuales son ineficientes para realizar un análisis de este tipo, por ende, su desarrollo, puede colaborar como una herramienta muy útil para la verificación de las soluciones obtenidas.

7. APENDICES

Apéndice A – BlockMeshDict –Backward Facing Step

Un ejemplo del archivo BlockMeshDict para la creación de la malla para el caso de Backward Facing Step, se muestra a continuación:

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  5
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

//convertToMeters 0.001;

vertices
(
(-4 1 0)
(0 1 0)
(0 0 0)
(40 0 0)
(40 1 0)
(40 9 0)
(0 9 0)
(-4 9 0)
(-4 1 -1)
(0 1 -1)
(0 0 -1)
(40 0 -1)
(40 1 -1)
(40 9 -1)
(0 9 -1)
(-4 9 -1)
);


blocks
(
    hex (8 9 14 15 0 1 6 7)
```

```
    (40 80 1)
    simpleGrading
(
      0.25
      (
        (0.5 0.5 4)
        (0.5 0.5 0.25)
      )
      1
    )
    hex (9 12 13 14 1 4 5 6)
    (400 80 1)
    simpleGrading
((
        (0.1 0.1 4)
        (0.9 0.9 1)
      )
      (
        (0.5 0.5 4)
        (0.5 0.5 0.25)
      )
      1
    )
    hex (10 11 12 9 2 3 4 1)
    (400 15 1)
    simpleGrading
((
        (0.1 0.1 4)
        (0.9 0.9 1)
      )
      (
        (0.5 0.5 2)
        (0.5 0.5 0.5)
      )
      1
    )
);

edges
(
);

boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 8 7 15)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (4 12 5 13)
```

```
            (3 11 4 12)
        );
    }
    upperWall
    {
        type wall;
        faces
        (
            (7 6 14 15)
            (6 5 13 14)
        );
    }
    lowerWall
    {
        type wall;
        faces
        (
            (0 1 9 8)
            (2 10 1 9)
            (2 3 11 10)

        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 1 7 6)
            (1 4 6 5)
            (2 3 1 4)
            (8 9 15 14)
            (9 12 14 13)
            (10 11 9 12)
        );
    }
);

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

APENDICE B – Condiciones iniciales y condiciones de borde.

Archivo de condiciones iniciales y condiciones de borde para la velocidad

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  5
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
```

```
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

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

internalField   uniform (0 0 0);

boundaryField
{
    inlet
    {
        type            SATurbulentInletVelocity;
        value           uniform (1 0 0);
    }

    outlet
    {
        type            zeroGradient;
    }

    upperWall
    {
        //type            fixedValue;
        //value           uniform (0 0 0);
        type            noSlip;
    }

    lowerWall
    {
      type              noSlip;
        //type            fixedValue;
        //value           uniform (0 0 0);
    }

    frontAndBack
    {
        type            empty;
    }
}

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

Archivo de condiciones iniciales y condiciones de borde para la presión

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  5
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }

    upperWall
    {
        type            zeroGradient;
    }

    lowerWall
    {
        type            zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}
```

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

Archivo de condiciones iniciales y condiciones de borde para la energía cinética turbulenta k.

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  5
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0.00375;

boundaryField
{
    inlet
    {
        type            TurbulentInletVelocityk;
        value           uniform 0.00375;
    }
    outlet
    {
        type            zeroGradient;
    }
    upperWall
    {
        type            kqRWallFunction;
        value           uniform 0.00375;
    }
    lowerWall
    {
        type            kqRWallFunction;
        value           uniform 0.00375;
    }
```

```
    frontAndBack
    {
        type            empty;
    }
}


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

Archivo de condiciones iniciales y condiciones de borde para frecuencia de la turbulencia w.

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  5
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      omega;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

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

internalField   uniform 0.4437;

boundaryField
{
    inlet
    {
        type            omegaTurbulentInlet;
        value           $internalField;
    }
    outlet
    {
        type            zeroGradient;
    }
    upperWall
    {
        type            omegaWallFunction;
        value           $internalField;
    }
```

```
    lowerWall
    {
        type            omegaWallFunction;
        value           $internalField;
    }
    frontAndBack
    {
        type            empty;
    }
}


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

Archivo de condiciones iniciales y condiciones de borde para la viscosidad de la turbulencia.

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  5
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
```

```
        value           uniform 0;
    }
    upperWall
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
    lowerWall
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
    frontAndBack
    {
        type            empty;
    }
}


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

Archivo de condiciones iniciales y condiciones de borde para energía de disipación turbulenta Épsilon.

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  5
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 0.000539;
```

```
boundaryField
{
    inlet
    {
        type            EpsilonTurbulentInlet;
        value           uniform 0.000539;
    }
    outlet
    {
        type            zeroGradient;
    }
    upperWall
    {
        type            epsilonWallFunction;
        value           uniform 0.000539;
    }
    lowerWall
    {
        type            epsilonWallFunction;
        value           uniform 0.000539;
    }
    frontAndBack
    {
        type            empty;
    }
}


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

APENDICE C – Archivos restante de configuración numérica

El siguiente archivo, es denominado controlDict, ubicado en la carpeta System.

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  5
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
```

```
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

application     simpleFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         2000;

deltaT          1;

writeControl    timeStep;

writeInterval   2000;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

libs      ("libSATurbulentInletVelocity.so");

***********************************************************************
**
```

A continuación, se presenta el archivo fvsolution

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  5
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
```

```
\*---------------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
    }

    "(U|k|epsilon|omega|f|v2)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-05;
        relTol          0.1;
    }
}

SIMPLE
{
    nNonOrthogonalCorrectors 0;
    consistent      yes;

    residualControl
    {
        p               1e-2;
        U               1e-3;
        "(k|epsilon|omega|f|v2)" 1e-3;
    }
}

relaxationFactors
{
    equations
    {
        U               0.9; // 0.9 is more stable but 0.95 more
convergent
        ".*"            0.9; // 0.9 is more stable but 0.95 more
convergent
    }
}
```

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

Fvschemes

```
/*--------------------------------*- C++ -*----------------------------
------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  5
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------
------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * //

ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      bounded Gauss linearUpwind grad(U);
    div(phi,k)      bounded Gauss limitedLinear 1;
    div(phi,epsilon) bounded Gauss limitedLinear 1;
    div(phi,omega)  bounded Gauss limitedLinear 1;
    div(phi,v2)     bounded Gauss limitedLinear 1;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
    div(nonlinearStress) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}
```

```
interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}


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

APENDICE D- Código de verificación de la solución

Código para verificación de la solución, y estimación de incerteza, para el caso resuelto mediante el modelo turbulento kwSST, malla m3, con refinamiento de 1.2.

```
import pandas
import numpy
import matplotlib.pyplot
from pylab import *
from scipy.optimize import fsolve
import scipy
import sys
import math

input_file  =  sys.argv[1]
output_figure = sys.argv[2]
input_data = pandas.read_csv(input_file,sep=',')
oro = input_data['Solucion'].values
h = input_data['Paso malla'].values
invh = 1.0/h
weights = invh/numpy.sum(invh)
ones = numpy.ones(numpy.size(h))


args_weighted = (h, oro, weights)
args_unweighted = (h, oro, ones)
valor_inicial= numpy.array((1,1,1))


def equations(a):
    oro0, alfa, p = a
    return (-oro0+sum(oro)-alfa*sum(h**p), -alfa+(sum(oro*(h**p))-
sum(oro)*sum(h**p))/(sum(h**(2*p))-sum(h**p)*sum(h**p)),
sum(oro*(h**p)*numpy.log(h))-oro0*sum((h**p)*numpy.log(h))-
alfa*sum((h**(2*p))*numpy.log(h)))
def nonlinearsystem(x, args):
```

```
  step, phi, weights = args
  phi0, alpha, p = x
  new_alpha = (numpy.sum(weights*phi*step**p) - numpy.dot(phi, weights)
* numpy.dot(weights, step**p)) / \
    (numpy.dot(weights,step**(2*p)) - numpy.dot(weights, step**p) *
numpy.dot(weights,step**p))
  new_phi0 = numpy.dot(phi,weights) - new_alpha * numpy.dot(weights,
step**p)
  logstep = numpy.log(step)
  f = lambda y :numpy.sum(weights*phi*logstep*step**y)-
new_phi0*numpy.sum(weights*step**y*logstep)- \
    new_alpha*numpy.sum(weights*step**(2*y)*logstep)
  new_p, info_dict, ier, msg = fsolve(f, p, full_output=True)
  converged = (ier==1)
  return (numpy.array((new_phi0, new_alpha, new_p)), converged)

def solve_nonlinear_system(initial, args):
  sol = initial
  residual = 10.0
  iteration = 0
  max_iteration = 100000
  tol = 1.0e-6
  successful = True
  while (residual > tol):
    new_sol, converged = nonlinearsystem(sol, args)
    if (not converged ) :
      successful = False
      break
    residual = numpy.linalg.norm(new_sol - sol)/numpy.linalg.norm(sol)

    sol = new_sol
    iteration += 1
    if(iteration > max_iteration):
      successful = False
      break
  return sol, successful

def desviacion_B2(x, args):
  step, phi, weights = args
  phi0, alpha, p = x
  ng = numpy.size(step)
  return numpy.sqrt( numpy.sum ( ng*weights*(phi-
(phi0+alpha*step**p))**2) / (ng-3))

def desviacion_B3(x,args):
  step, phi, weights = args
  phi0, alpha = x
  ng = numpy.size(step)
  return numpy.sqrt( numpy.sum ( ng*weights*(phi-(phi0+alpha*step))**2)
/ (ng-2))

def desviacion_B4(x,args):
  step, phi, weights = args
  phi0, alpha = x
  ng = numpy.size(step)
  return numpy.sqrt( numpy.sum ( ng*weights*(phi-
(phi0+alpha*step**2))**2) / (ng-2))
```

```
def desviacion_B5(x,args):
  step, phi, weights = args
  phi0, alpha1, alpha2 = x
  ng = numpy.size(step)
  return numpy.sqrt( numpy.sum ( ng*weights*(phi-
(phi0+alpha1*step+alpha2*step**2))**2) / (ng-3))



def matrix_coeff(p, args):
  step, phi, weights = args
  return numpy.sum(weights*step**p)
def rhs_coeff(p, args):
  step, phi, weights = args
  return numpy.sum(weights*phi*step**p)
matrix_factor = lambda step, weights, exp: numpy.sum(weights*step**exp)




B3_unweighted_matrix = numpy.matrix([[1, matrix_coeff(1,
args_unweighted)],  \
  [matrix_coeff(1, args_unweighted), matrix_coeff(2,
args_unweighted)]])
B3_unweighted_rhs = numpy.array([rhs_coeff(0,args_unweighted),
rhs_coeff(1,args_unweighted)])

B3_weighted_matrix = numpy.matrix([[1, matrix_coeff(1, args_weighted)],
\
  [matrix_coeff(1, args_weighted), matrix_coeff(2, args_weighted)]])
B3_weighted_rhs = numpy.array([rhs_coeff(0,args_weighted),
rhs_coeff(1,args_weighted)])


B4_unweighted_matrix = numpy.matrix([[1, matrix_coeff(2,
args_unweighted)],  \
  [matrix_coeff(2, args_unweighted), matrix_coeff(4,
args_unweighted)]])
B4_unweighted_rhs = numpy.array([rhs_coeff(0,args_unweighted),
rhs_coeff(2,args_unweighted)])




B4_weighted_matrix = numpy.matrix([[1, matrix_coeff(2, args_weighted)],
\
  [matrix_coeff(2, args_weighted), matrix_coeff(4, args_weighted)]])
B4_weighted_rhs = numpy.array([rhs_coeff(0,args_weighted),
rhs_coeff(2,args_weighted)])

B5_unweighted_matrix = numpy.matrix([[1, matrix_coeff(1,
args_unweighted), matrix_coeff(2, args_unweighted)],  \
  [matrix_coeff(1, args_unweighted), matrix_coeff(2, args_unweighted),
matrix_coeff(3, args_unweighted)], \
  [matrix_coeff(2, args_unweighted), matrix_coeff(3, args_unweighted),
matrix_coeff(4, args_unweighted)]])
```

```
B5_unweighted_rhs = numpy.array([rhs_coeff(0,args_unweighted),
rhs_coeff(1,args_unweighted), \
rhs_coeff(2, args_unweighted)])

B5_weighted_matrix = numpy.matrix([[1, matrix_coeff(1, args_weighted),
matrix_coeff(2, args_weighted)],  \
  [matrix_coeff(1, args_weighted), matrix_coeff(2, args_weighted),
matrix_coeff(3, args_weighted)], \
  [matrix_coeff(2, args_weighted), matrix_coeff(3, args_weighted),
matrix_coeff(4, args_weighted)]])

B5_weighted_rhs = numpy.array([rhs_coeff(0,args_weighted),
rhs_coeff(1,args_weighted), \
rhs_coeff(2, args_weighted)])


valor_inicial= numpy.array((0.04,0.05,2))
ng_square = numpy.sqrt(numpy.size(h))


B2_solution_weighted, weighted_succeed =
solve_nonlinear_system(valor_inicial, args_weighted)
B2_sigma_weighted = desviacion_B2(B2_solution_weighted, args_weighted)
B2_solution_unweighted, unweighted_succeed =
solve_nonlinear_system(B2_solution_weighted, args_unweighted)
B2_sigma_unweighted = desviacion_B2(B2_solution_unweighted,
args_unweighted)/ng_square


B3_solution_weighted = numpy.linalg.solve(B3_weighted_matrix,
B3_weighted_rhs)
B3_sigma_weighted = desviacion_B3(B3_solution_weighted, args_weighted)
B3_solution_unweighted = numpy.linalg.solve(B3_unweighted_matrix,
B3_unweighted_rhs)
B3_sigma_unweighted = desviacion_B3(B3_solution_unweighted,
args_unweighted)/ng_square

B4_solution_weighted = numpy.linalg.solve(B4_weighted_matrix,
B4_weighted_rhs)
B4_sigma_weighted = desviacion_B4(B4_solution_weighted, args_weighted)
B4_solution_unweighted = numpy.linalg.solve(B4_unweighted_matrix,
B4_unweighted_rhs)
B4_sigma_unweighted = desviacion_B4(B4_solution_unweighted,
args_unweighted)/ng_square




B5_solution_weighted = numpy.linalg.solve(B5_weighted_matrix,
B5_weighted_rhs)
B5_sigma_weighted = desviacion_B5(B5_solution_weighted, args_weighted)
B5_solution_unweighted = numpy.linalg.solve(B5_unweighted_matrix,
B5_unweighted_rhs)
B5_sigma_unweighted = desviacion_B5(B5_solution_unweighted,
args_unweighted)/ng_square

B3_delta_weighted = B3_solution_weighted[1] * h
B3_delta_unweighted = B3_solution_unweighted[1] * h
```

```
B4_delta_weighted = B4_solution_weighted[1] * h ** 2
B4_delta_unweighted = B4_solution_unweighted[1] * h ** 2
B5_delta_weighted = B5_solution_weighted[1] * h +
B5_solution_weighted[2] * h ** 2
B5_delta_unweighted = B5_solution_unweighted[1] * h +
B5_solution_unweighted[2] * h ** 2




epsilon_phi = 0
p = 0
standard_deviation = 0
p_weighted = B2_solution_weighted[2]
p_unweighted = B2_solution_unweighted[2]
if (p_weighted > 0.5 and p_weighted <= 2 and p_unweighted > 0.5 and
p_unweighted <= 2 and
weighted_succeed and unweighted_succeed):
  index = numpy.argmin([B2_sigma_unweighted, B2_sigma_weighted])
  standard_deviation = numpy.amin([B2_sigma_unweighted,
B2_sigma_weighted])
  if(index == 0):
    epsilon_phi = B2_solution_unweighted[1] * h ** p_unweighted
    p = p_unweighted
    phi_fit = B2_solution_unweighted[0] + B2_solution_unweighted[1] * h
** p_unweighted
  else:
    epsilon_phi = B2_solution_weighted[1] * h ** p_weighted
    p = p_weighted
    phi_fit = B2_solution_weighted[0] + B2_solution_weighted[1] * h **
p_weighted
elif (p_weighted > 0.5 and p_weighted <= 2 and weighted_succeed) :
  epsilon_phi = B2_solution_weighted[1] * h ** p_weighted
  standard_deviation = B2_sigma_weighted
  p = p_weighted
  phi_fit = B2_solution_weighted[0] + B2_solution_weighted[1] * h **
p_weighted
elif (p_unweighted > 0.5 and p_unweighted <= 2 and unweighted_succeed):
  epsilon_phi = B2_solution_unweighted[1] * h **p_unweighted
  standard_deviation = B2_sigma_unweighted
  p = p_unweighted
  phi_fit = B2_solution_unweighted[0] + B2_solution_unweighted[1] * h
** p_unweighted
elif (p_weighted > 2 and p_unweighted > 2 and weighted_succeed and
unweighted_succeed):
  delta = numpy.array([B3_delta_unweighted, B3_delta_weighted,
B4_delta_unweighted, B4_delta_weighted])
  sigma = numpy.array([B3_sigma_unweighted, B3_sigma_weighted,
B4_sigma_unweighted, B4_sigma_weighted])
  epsilon_phi = delta[numpy.argmin(sigma)]
  standard_deviation = numpy.amin(sigma)

  index = numpy.argmin(sigma)

  if index == 0:
    phi_fit = B3_solution_unweighted[0] + B3_solution_unweighted[1] * h
  elif index == 1:
```

```
    phi_fit = B3_solution_weighted[0] + B3_solution_weighted[1] * h
  elif index == 2:
    phi_fit = B4_solution_unweighted[0] + B4_solution_unweighted[1] *
h**2
  elif index == 3:
    phi_fit = B4_solution_weighted[0] + B4_solution_weighted[1] * h**2
else:
  delta = numpy.array([B3_delta_unweighted, B3_delta_weighted,
B4_delta_unweighted, B4_delta_weighted, \
    B5_delta_unweighted, B5_delta_weighted])
  sigma = numpy.array([B3_sigma_unweighted, B3_sigma_weighted,
B4_sigma_unweighted, B4_sigma_weighted, \
    B5_sigma_unweighted, B5_sigma_weighted])

  epsilon_phi = delta[numpy.argmin(sigma)]




  index = numpy.argmin(sigma)
  standard_deviation = numpy.amin(sigma)
  p = p_weighted

  if index == 0:
    phi_fit = B3_solution_unweighted[0] + B3_solution_unweighted[1] * h
  elif index == 1:
    phi_fit = B3_solution_weighted[0] + B3_solution_weighted[1] * h

  elif index == 2:
    phi_fit = B4_solution_unweighted[0] + B4_solution_unweighted[1] *
h**2
  elif index == 3:
    phi_fit = B4_solution_weighted[0] + B4_solution_weighted[1] * h**2
  elif index == 4:
    phi_fit = B5_solution_unweighted[0] + B5_solution_unweighted[1] * h
+ \
      B5_solution_unweighted[2] * h ** 2
  elif index == 5:
    phi_fit = B5_solution_weighted[0] + B5_solution_weighted[1] * h + \
      B5_solution_weighted[2] * h  ** 2



delta_phi = (numpy.amax(oro) - numpy.amin(oro))  / (numpy.size(h) - 1)

safe_factor = 3
if(p >= 0.5 and p < 2.1 and standard_deviation < delta_phi and
weighted_succeed and unweighted_succeed):
  safe_factor = 1.25

#APPENDIX A.1.4
if (standard_deviation < delta_phi):
  incerteza = safe_factor * epsilon_phi + standard_deviation +
numpy.abs(oro - phi_fit)
else:
  incerteza = 3 * standard_deviation / delta_phi * \
```

```
    ( epsilon_phi + standard_deviation + numpy.abs(oro - phi_fit) )

from matplotlib import rc, pyplot



rc('text',usetex=True)
fig = pyplot.figure(figsize=(10,10))
ax = fig.add_subplot(111)
ax.set_xlabel(r'$\frac{h_i}{h_1}$',fontsize=30)
ax.set_ylabel(r'$C_f$',fontsize=30)
ax.errorbar(h/h[0],oro , yerr=incerteza, fmt='o',
capthick=2,elinewidth=2,capsize=15)
ax.tick_params(labelsize=16)
fig.savefig(output_figure,bbox_inches='tight')
```

8. REFERENCIAS

- L.Eca, M. (2002). *Evaluation of verification procedures for CFD applications ins: Symposium of Naval Hydrodinamics.* Japon: Fukuoka.

- L.Eca, M. H. (2009). *Error estimation based on refinement of grid: a challenge for grid generation: Conferencia de métodos númericos ingenieria* . Barcelona España: SEMNI.

- M.Hoekstra, L. y. (2014). *A procedure for the estimation of the numerical uncertainty of CFD calculations base on grid refinament studies.* Lisbon, Porturgal: Crosck Marcks.

- P.J.Roche. (1998). *Verification and Validation in Computational Science and Engeenering.* Albuquerque, Nuevo Mexico: Hermosa Publishers.

- P.J.Roche. (2009). *Fundamentals of Verification and Validation* . Albuquerque, Nuevo Mexico: Hermosa Publishers.

- W.Malalasekera, H. y. (2007). *An Introduction to Computational Fluid Dynamics.* Edinburgh, England: Pearson Education Limited.

- OpenCFD Ltd (ESI Group) (©2004-2018), OpenFOAM. https://www.openfoam.com/

- **NASA Official John W. Slater. Tuesday, 08-Jul-2008. Backward-Facing Step .https://www.grc.nasa.gov**

- MARINE ENVIRONMENT & TECHNOLOGY CENTER MARATEC. (©2004-2008). Workshops on CFD Uncertainty Analysis. http://maretec.ist.utl.pt/html_files/CFD_Workshops.htm