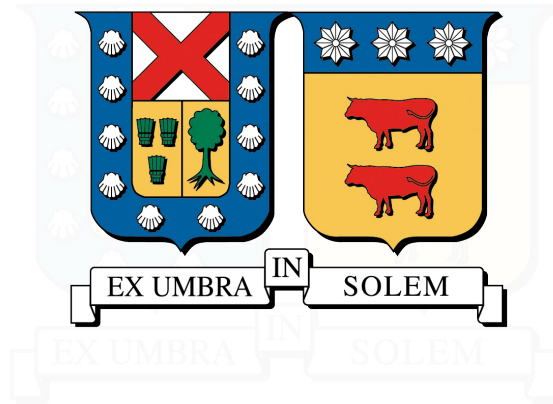


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INGENIERÍA MECÁNICA
SANTIAGO - CHILE



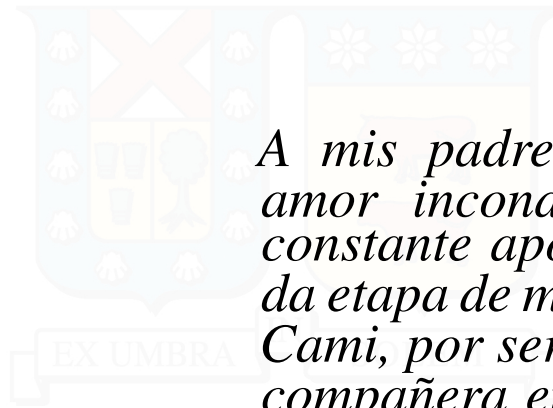
ESTIMACIÓN Y MEDICIÓN DE LA EROSIÓN EN PALAS DE UNA TURBINA HIDRÁULICA MEDIANTE SIMULACIONES COMPUTACIONALES

GONZALO FELIPE BOZO NALLI

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MECÁNICO

PROFESOR GUÍA : DR. RODRIGO CASSINELI PALLHARINI.
PROFESOR CORREFERENTE : DR. JAIME NÚÑEZ S.

JUNIO 2023



A mis padres, por su amor incondicional y constante apoyo en cada etapa de mi vida; y a Cami, por ser mi faro y compañera en este viaje, ayudándome siempre a ser mejor. ...

"Tener la capacidad de esforzarte al máximo por ser algo es un talento en sí mismo."

AGRADECIMIENTOS

En primer lugar, quiero expresar mi más profundo agradecimiento a mis padres por su apoyo incondicional, tanto emocional como económico, durante todo este proceso. A mi hermana, por estar presente y ser parte de mi vida universitaria.

Agradezco a Cami, por ser mi compañera de estudios y amiga, por las risas, el apoyo y las horas de trabajo juntos. Sin ella, no hubiese sido posible lograr este objetivo.

A continuación, deseo expresar mi profundo agradecimiento a mi asesor de tesis, Rodrigo Cassineli, por su dedicación, apoyo y orientación en el desarrollo de este trabajo. Su experiencia y conocimientos han sido fundamentales para llevar a cabo mi investigación.

También quiero agradecer a mis profesores y compañeros de la universidad, con quienes he compartido ramos y experiencias tanto dentro como fuera del ámbito académico. Su presencia, apoyo y camaradería han sido fundamentales en mi formación y crecimiento personal.

Un agradecimiento especial a Francisco Ramirez, por su colaboración y apoyo en la realización de la memoria, compartiendo conocimientos y experiencias en este proyecto conjunto.

A mis amigos, que han sido una fuente constante de apoyo emocional y momentos de distracción, especialmente Benjamin, Sebastián, Dani y Samanta.

Por último, quiero expresar mi gratitud a todas aquellas personas que, de una forma u otra, han contribuido al éxito de mi carrera universitaria. Aprendí la importancia de la colaboración y el trabajo en equipo, y estoy profundamente agradecido por cada uno de ustedes.

RESUMEN

El objetivo principal de esta memoria es realizar una estimación de la erosión en una turbina hidráulica del tipo Pelton mediante métodos computacionales. Para lograrlo, se estudió la teoría aplicada a la erosión en métodos computacionales y se consideraron los parámetros más importantes. Además, se analiza la herramienta para realizar estas simulaciones, OpenFoam 8, y sus respectivas librerías asociadas, principalmente (DPMFoam).

Para ejecutar las simulaciones, se utiliza el método computacional de volúmenes finitos a través del software OpenFoam y con una discretización basada en el esquema lineal de Gauss. Por otro lado, el seguimiento de las partículas se logra mediante el método de descripción Euler-Lagrange con un acoplamiento del tipo one-way, y para obtener la erosión, se considera el modelo de Finnie. La selección de los métodos señalados fue aplicada en la validación y en la simulación principal.

El primer paso que se llevó a cabo fue la validación, utilizando como referencia el estudio realizado por la University of Glasgow, que consiste en un experimento del tipo jet impingement, como el realizado por Gnanavelu, con un objetivo a distancias de 5 mm y 25 mm. El propósito de esta validación fue obtener la posición de las partículas al momento del impacto, la velocidad de impacto y el ángulo de impacto de las partículas. Los resultados de la validación fueron positivos, salvo algunas diferencias que se deben principalmente a actualizaciones de la librería DPMFoam.

Después de lograr una validación exitosa, se llevó a cabo la simulación para encontrar la erosión en una turbina Pelton. Con el fin de poder comparar los resultados con una situación real, se utilizaron los parámetros de una experiencia realizada en India, donde se replicó a escala una turbina Pelton de una planta de energía ubicada en el Himalaya.

Los resultados finales en términos de zonas de desgaste fueron positivos, ya que la simulación coincide con el experimento. Sin embargo, las mediciones realizadas en el experimento no se pueden replicar a la perfección debido a que son de una alta complejidad técnica y requieren de mayores recursos computacionales que los utilizados.

Finalmente, se recomienda para futuros estudios considerar la deformación de la pala de la turbina, ya que influye significativamente en la erosión abrasiva que ocurre en los distintos materiales, y también realizar un estudio del fenómeno cuando se incorpora un revestimiento en la pala hidráulica.

ABSTRACT

The main objective of this work is to estimate erosion in a Pelton-type hydraulic turbine using computational methods. To achieve this, the theory applied to erosion in computational methods was studied, and the most important parameters were considered. In addition, the tools to carry out these simulations, such as OpenFoam 8 with its respective associated libraries (mainly DPMFoam), were also studied.

To run the simulations, the computational method of finite volumes is used through the OpenFoam software, and a discretization based on the linear Gaussian scheme is applied. On the other hand, particle tracking is achieved using the Euler-Lagrange description method with one-way coupling. To obtain the erosion, the Finnie model is considered. The selection of the appropriate methods was applied in the validation and the main simulation.

The first step that was executed was validation, using as a reference the study carried out by the University of Glasgow, which consists of a jet impingement-type experiment, similar to the one conducted by Gnanavelu, with targets at distances of 5 mm and 25 mm. The objective of this validation was to obtain the particle impact location, the particle velocity at impingement, and the impingement angles. The validation results were positive, except for some differences that are mainly due to updates in the DPMFoam library.

After achieving successful validation, the simulation was carried out to find the erosion in a Pelton turbine. To compare the results with a real situation, the parameters of an experiment performed in India were used, where a Pelton turbine from a power plant located in the Himalayas was replicated to scale.

The final results in terms of wear zones were positive, as the simulation matches the experiment. However, the measurements taken in the experiment cannot be replicated perfectly due to their high technical complexity and requiring greater computational resources than those used.

Finally, it is recommended for future studies to consider the deformation of the turbine blade, as it greatly influences the abrasive erosion that occurs in different materials, and also to conduct a study of the phenomenon when a coating is incorporated into the hydraulic blade.

Índice de Contenidos

1. Introducción	1
1.1. Objetivo General	2
1.2. Objetivos Específicos	2
2. Marco Teórico	3
2.1. Turbina Pelton	3
2.2. Erosión	4
2.2.1. Erosión en la turbina Pelton	5
2.3. Modelos de predicción de abrasión hidráulica	6
2.3.1. Modelo de Finnie	6
2.3.2. Modelo de Nandakumar	7
2.3.3. Modelo de Grant y Tabakoff	8
2.4. Acoplamientos	8
2.5. Condiciones iniciales	9
2.6. Jet Impigement Test (JIT)	9
2.7. Antecedentes y estudios relevantes en erosión hidráulica	10
3. Método Computacional	12
3.1. Computational Fluid Dynamics (CFD)	12
3.1.1. Ecuaciones Gobernantes	13
3.1.2. Mallado	13
3.1.3. Criterio de Courant-Friedrichs-Levy	15
3.1.4. Discretización	15
3.1.5. Soluciones iterativas	16
3.1.6. Modelos de turbulencia	16
3.2. OpenFOAM (Open Field Operation And Manipulation)	17
3.3. Métodos de descripción de partícula	18
3.4. Modelamiento de la fase discreta	21
3.5. Enfoque de la solución	21
3.6. Estructura de OpenFOAM	22
3.6.1. System	23
3.6.1.1. blockMesh	23
3.6.1.2. snappyHexMesh	24
3.6.1.3. controlDict	24
3.6.1.4. decomposeParDict	24
3.6.1.5. fvSchemes	24
3.6.1.6. fvSolution	24
3.6.1.7. setFieldsDict	24
3.6.2. Constant	24
3.6.2.1. kinematicCloudProperties	24
3.6.3. Directorios de tiempo	26

3.6.3.1.	U.water	26
3.6.3.2.	p	26
3.6.3.3.	k.water	26
3.6.3.4.	nut.water	26
3.6.3.5.	epsilon.water	26
3.7.	Postprocesamiento	27
4.	Validación y Verificación	28
4.1.	Caso científico utilizado para la validación	29
4.2.	Validación	37
4.2.1.	Geometría y mallado	37
4.2.2.	Resultados	39
4.2.2.1.	Localización de impacto	39
4.3.	Resultados de velocidad de impacto	42
4.3.0.1.	Ángulos de Impacto	44
4.4.	Análisis de Validación	45
5.	Simulación de Erosión en Pala de Turbina Pelton	47
5.1.	Resultados experimentales y Resultados Numéricos	47
5.2.	Consideraciones	49
5.2.1.	Selección modelo de erosión de Finnie	51
5.3.	Resultados	52
5.3.1.	Resultados de la simulación en OpenFOAM	52
5.3.1.1.	Intensidad de la erosión en la cuchara de la turbina Pelton	52
5.3.1.2.	Cálculo de la erosión normalizada	53
5.3.1.3.	Comparación de los resultados de erosión normalizada	53
5.3.1.4.	Comparación con el estudio experimental de Kumar	53
5.3.1.5.	Discusión de resultados y limitaciones	55
6.	Análisis y Conclusiones	57
	Bibliografía	59
A.	Código OpenFoam validación 5 mm	63
A.1.	blockMeshDict	63
A.2.	snappyHexMesh	65
A.3.	controlDict	71
A.4.	fvSchemes	72
A.5.	fvSolution	73
A.6.	mapFieldsDict	74
A.7.	meshQualityDict	74
A.8.	setFieldsDict	74
A.9.	surfaceFeaturesDict	75
A.10.	kinematicCloudProperties	75
A.11.	g	79
A.12.	momentumTransport.water	80
A.13.	transportProperties	80
A.14.	U.water	81
A.15.	p	82
A.16.	k.water	83
A.17.	epsilon.water	84
A.18.	nut.water	85
B.	Código OpenFoam validación 25 mm	88
B.1.	blockMeshDict	88

B.2. snappyHexMesh	91
B.3. controlDict	97
B.4. fvSchemes	98
B.5. fvSolutions	99
B.6. mapFieldsDict	99
B.7. meshQualityDict	100
B.8. setFieldsDict	100
B.9. surfaceFeaturesDict	101
B.10. kinematicCloudProperties	101
B.11. g	106
B.12. momentumTransport.water	106
B.13. transportProperties	107
B.14. U.water	107
B.15. p	108
B.16. k.water	109
B.17. epsilon.water	110
B.18. nut.water	111
C. Código OpenFoam erosión pala Pelton	113
C.1. blockMeshDict	113
C.2. snappyHexMesh	116
C.3. controlDict	123
C.4. fvSchemes	123
C.5. fvSolutions	124
C.6. mapFieldsDict	125
C.7. meshQualityDict	125
C.8. setFieldsDict	125
C.9. surfaceFeaturesDict	126
C.10. kinematicCloudProperties	126
C.11. g	131
C.12. momentumTransport.water	131
C.13. transportProperties	131
C.14. U.water	132
C.15. p	133
C.16. k.water	134
C.17. epsilon.water	134
C.18. nut.water	135
D. Código python postprocesamiento	137
D.1. Código extracción de datos	137
D.2. Código orden de datos	142
E. Tiempo exposición de la pala	147

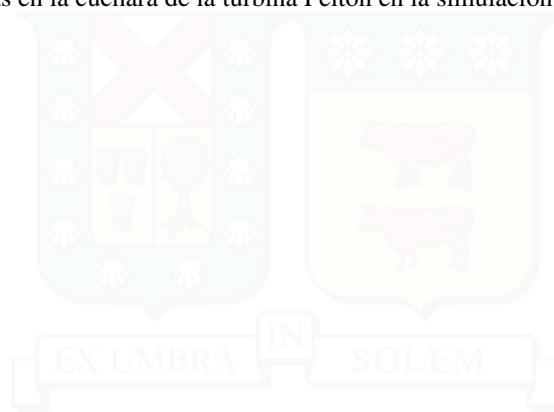
Índice de Tablas

3.1. Modelos de turbulencia	17
4.1. Parámetros de simulación de Lopéz	30
4.2. Análisis de localización de impacto de partículas a 5 mm de distancia de la boquilla al objetivo	40
4.3. Análisis de localización de impacto de partículas a 25 mm de distancia de la boquilla al objetivo	41
5.1. Detalles de los parámetros utilizados en el estudio.	48
5.2. Detalles de los parámetros seleccionados para la simulación.	49
5.3. Comparación de los resultados de erosión normalizada obtenidos en la simulación y en el estudio experimental de Kumar [1].	53

Índice de Figuras

1.1. Efecto de la erosión en palas de Turbina Pelton	1
2.1. Turbina Pelton de seis boquillas.	3
2.2. Transporte del agua a la turbina Pelton.	4
2.3. Configuración de una cuchara Pelton.	4
2.4. Descomposición de velocidades de una partícula antes de impactar una superficie.	5
2.5. Formas en que se produce la abrasión estudiada por Finnie	6
2.6. Se describen los distintos tipos de acoplamientos entre partícula y fluido (a) Una dirección (b) Dos direcciones (c) Cuatro direcciones	8
2.7. Configuración del Jet Impingement Test	10
3.1. Ortogonalidad de las celdas	14
3.2. Asimetría entre dos celdas adyacentes	14
3.3. Variación de velocidad en flujos turbulentos	16
3.4. Describe el rastreo de partícula mediante el método de Euler	19
3.5. Describe el rastreo de partícula mediante el método de Lagrange	19
3.6. Estructura de OpenFOAM	23
4.1. Configuración Jet Impingement Test.	29
4.2. Condiciones de borde y Figura JIT.	30
4.3. Modelos de inyección OpenFOAM	31
4.4. Localización del impacto a 5mm.	32
4.5. Localización del impacto a 25mm.	32
4.6. Cilindro de 25mm subdividido en regiones de 1 mm.	33
4.7. Velocidad de impacto a 5mm.	34
4.8. Velocidad de impacto a 25mm.	34
4.9. Ángulo de impacto a 5mm.	35
4.10. Ángulo de impacto a 25mm.	36
4.11. Geometría cubica para distancia de 5mm.	37
4.12. Geometría cilíndrica para una distancia de 5 mm.	38
4.13. Geometría cilíndrica para distancia de 25 mm	39
4.14. Localización de impacto a 5mm.	40
4.15. Localización de impacto a 25mm.	41
4.16. Velocidad de impacto a 5mm.	43
4.17. Velocidad de impacto a 25mm.	43
4.18. Ángulo de impacto a 5mm.	44
4.19. Ángulo de impacto a 25mm.	45
5.1. Configuración Experimento.	48
5.2. Configuración Experimento.	49
5.3. Geometría y malla de la pala de la turbina Pelton.	50

5.4. Intensidad de la erosión en la cuchara de la turbina Pelton, representada en términos de volumen extraído.	52
5.5. Progresión de la erosión y eliminación del recubrimiento en la cuchara de la turbina Pelton a lo largo del experimento realizado por Kumar et al. [1].	53
5.6. Comparación de la erosión normalizada obtenida en la simulación y en el estudio experimental de Kumar [1].	54
5.7. Partículas pegadas en la cuchara de la turbina Pelton en la simulación.	55



1 | Introducción

En la actualidad, la erosión se ha convertido en un problema bastante relevante en el desgaste de las palas de las turbinas hidráulicas (Ver Figura 1.1), lo que lleva a severos inconvenientes operacionales y de mantenimiento [2]. Esto se debe a que el fluido utilizado para el funcionamiento de la turbina contiene sedimento compuesto por partículas abrasivas. Debido a esto, se han desarrollado gran número de modelos computacionales considerando los distintos parámetros según el enfoque utilizado [3]. Antes de tomar decisiones debido a las distintas situaciones en las que ocurre la erosión, ya sea con distintos tipos, tamaños y material del sedimento, distintas velocidades y caudales de flujo, sumado al material y forma que puede tener la maquinaria objeto de la erosión, la erosión producida en distintos casos también es diferente. Puede primar la deformación dúctil por sobre la frágil [4], viceversa o en partes relativamente iguales. Por lo tanto, realizar prototipos para cada situación puede ser altamente costoso, y es por ello que optar por simulaciones es la opción económicamente conveniente [5]. Aunque la simulación con métodos computacionales puede ser la opción más económica, se requiere experiencia en el software que se utilizará como herramienta, así como conocimiento teórico del fenómeno a estudiar [6], en este caso la erosión. Es necesario conocer la teoría que se aplicará en diferentes situaciones. Esto se debe al poco acuerdo que existe entre los distintos autores para definir un modelo de erosión, según un estudio realizado por Meng y Ludema [7], existen 182 ecuaciones distintas, cada una considera distintos parámetros y variables de entre un total de 100.



(a) Pala de turbina Pelton con la superficie erosionada en Khimti, Nepal.

(b) Efecto de la erosión en la superficie y splitter de una turbina Pelton Rangjung, Bhutan

Figura 1.1: Efecto de la erosión en palas de Turbina Pelton

(Fuente: Sediment Erosion in Hydraulic Turbines [8])

Durante la última década, las técnicas basadas en CFD han sido ampliamente utilizadas para predecir el desgaste erosivo de componentes de maquinaria hidráulica debido al impacto de partículas sólidas [9]. Esto apunta a ser una herramienta bastante útil tanto para predecir la erosión en las maquinarias al momento del diseño como al momento de decidir si implementar alguna solución, como el tipo de revestimiento en una máquina hidráulica. La plataforma OpenFoam ha demostrado ser flexible al momento de adaptarse a los avances teóricos y los distintos contextos en los que puede ser utilizado [10]. Sin embargo, para poder aprovechar de la mejor manera esta herramienta, es necesario tener un gran conocimiento del software en

general, de las librerías que se seleccionan para cada caso y los conocimientos científicos que se requieran para cada situación. Como se mencionó anteriormente, los avances realizados en erosión han sido bastantes, pero las simulaciones computacionales de erosión en turbina Pelton son escasas para la cantidad de casos reales existentes, y aún más difícil encontrar que se realicen en OpenFoam. Esto denota la importancia de realizar avances en estos tópicos.

1.1. Objetivo General

Estimar la erosión en las palas de la turbina hidráulica tipo Pelton adaptando un modelo computacional en OpenFOAM basándose en el estudio realizado por la University of Glasgow [11].

1.2. Objetivos Específicos

Para cumplir con el objetivo general, se establecen los siguientes objetivos específicos.

- Estudiar bibliografía relacionada con los métodos computacionales de erosión desde el punto de vista teórico.
- Realizar un estudio sobre la teoría de la erosión para seleccionar los parámetros más importantes a considerar y que afectarán de mayor manera la erosión de las palas.
- Interiorizarse en la plataforma OpenFOAM y sus herramientas para validar el modelo computacional de erosión.
- Selección del modelo matemático adecuado para simular de mejor manera la erosión de las palas.
- Calcular computacionalmente la erosión causada por un chorro de agua y partículas sólidas en una pala de turbina Pelton.

2 | Marco Teórico

2.1. Turbina Pelton

La turbina Pelton es un tipo de turbina de impulso utilizada en las centrales eléctricas. Este tipo de máquinas entrega una excelente manera de producir energía si el depósito de agua es alto y caudales relativamente bajos [12]. El flujo de agua fluye a través de una tubería aguas abajo hasta llegar a un conjunto de boquillas que están ubicadas frente a las palas que están unidas a una rueda. Generalmente las turbinas Pelton se utilizan cuando la altura total es superior a 300 m [12]. Existe una instalación en Suiza en la que la columna de agua es 1700m [12].

Una ventaja importante de las turbinas Pelton es que puede funcionar con agua cargada de sedimentos, ya que el daño por erosión es fácil de reparar [12].

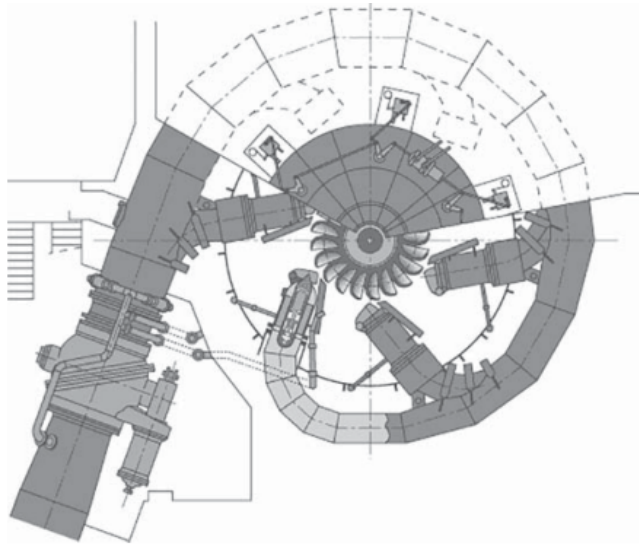


Figura 2.1: Turbina Pelton de seis boquillas. (Fuente: Principios de las turbomaquinas [12])

Las turbinas de impulso se caracterizan por ser impulsadas por uno o más chorros libres de alta velocidad (Figura 2.1). La turbina de impulso recibe suministro de agua desde una altura elevada a través de un conducto largo llamado penstock [13] (Figura 2.2). La energía potencial del agua debido a la altura se convierte en energía cinética al llegar a las boquillas [13]. El agua se acelera a través de la boquilla y se descarga como un chorro libre a alta velocidad a presión atmosférica. El chorro golpea las palas deflectoras unidas a la rueda, convirtiendo la energía cinética en trabajo [13].

En la cuchara de la turbina Pelton, el flujo se divide en dos partes debido a la forma que poseen, y cambia de dirección en casi 180° en el plano horizontal (ver Figura 2.3). Si se ignoran la gravedad y la

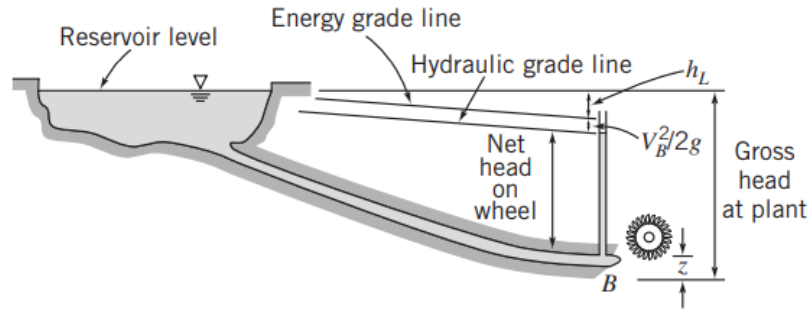


Figura 2.2: Transporte del agua a la turbina Pelton. (Fuente: Fox McDonalds [13])

fricción, la velocidad del agua en relación a la pala no cambia durante la descarga. Es importante notar que el flujo de agua no es permanente, debido a que la orientación de cada pala cambia continuamente en relación al chorro incidente [13]. Además, hay un número distinto de cucharas en grados de contacto diferentes con el chorro que varía en cada instante [13].

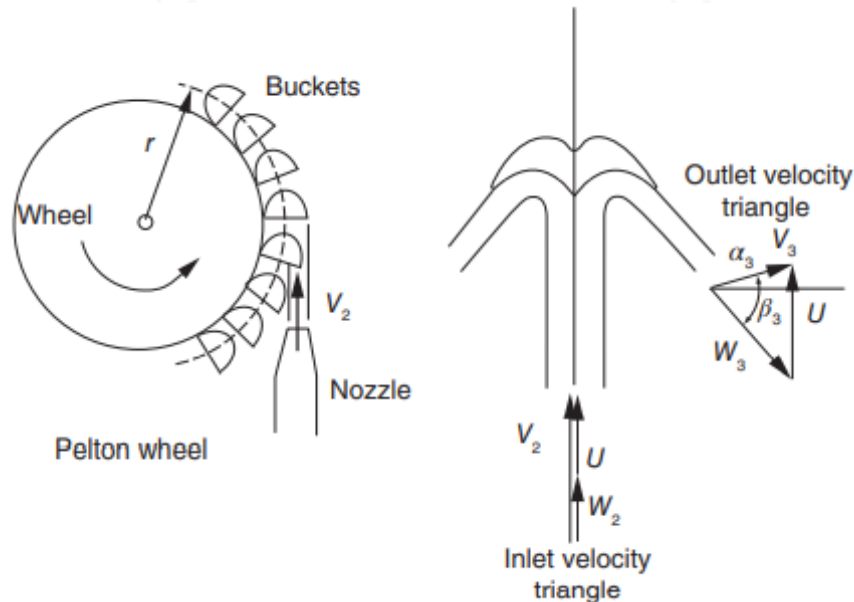


Figura 2.3: Configuración de una cuchara Pelton. (Fuente: Principios de las turbomaquinas Seppo Kortela [12])

2.2. Erosión

La abrasión hidráulica en componentes de máquinas hidráulicas como turbinas, bombas, etc. puede ser interpretada como el proceso de alteración gradual del estado y la forma que toma lugar en la superficie de los componentes. El proceso se desarrolla en respuesta a la acción de partículas sólidas abrasivas suspendidas en el líquido de trabajo y también por la influencia del fluido. Mientras que las partículas presentes en el fluido actúan mecánicamente sobre la superficie, el efecto del agua pura en las superficies es mecánica y química (acción corrosiva). Por lo tanto, la abrasión hidráulica puede ser considerada como un proceso mecánico-abrasivo.

Se ha demostrado que la pérdida de material por erosión-corrosión no es simplemente la superposición

de la pérdida por erosión y corrosión, sino que existe una sinergia entre ambos fenómenos que aumenta la pérdida de material. De esta manera los casos de erosión-corrosión se pueden expresar según la siguiente ecuación:

$$\Delta W_T = \Delta W_E + \Delta W_C + \Delta W_S \quad (2.1)$$

Donde ΔW_T es la pérdida total de material, ΔW_E es la pérdida por erosión, ΔW_C la pérdida por corrosión pura y ΔW_S es la pérdida por la sinergia. En este trabajo se enfoca en las pérdidas por erosión.

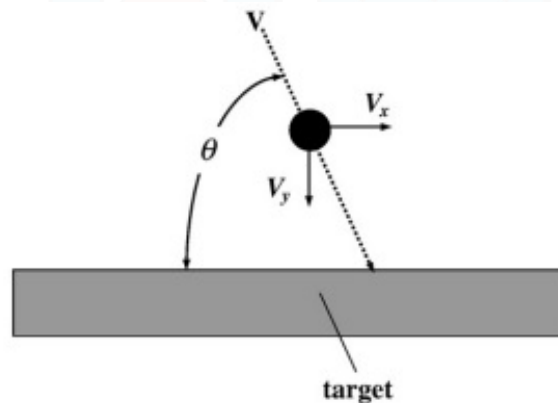


Figura 2.4: Descomposición de velocidades de una partícula antes de impactar una superficie.

(Fuente: Nandakumar [14])

La Figura 2.4 muestra una partícula impactando en el objetivo. La velocidad de impacto V puede ser dividida en dos componentes: velocidad tangencial V_x y velocidad normal V_y . Cuando una partícula impacta en el objetivo, ambos componentes producen una fuerza tangencial F_x y una fuerza normal F_y : Cada una de esas fuerzas juega roles distintos en el proceso de erosión. Para la fuerza normal F_y , la partícula golpea el objetivo como si fuese un martillo provocando la deformación de la superficie. Si la deformación es repetitiva eventualmente causara la pérdida de material se llama pérdida por daño de deformación o erosión por impacto. En cambio, en el caso de la fuerza tangencial F_x funciona como un arado que corta la superficie arrebatando material, lo que se llama remoción por corte o erosión por deslizamiento.

Son muchos los factores que afectan en la pérdida por erosión pura. Los principales factores asociados con las partículas son, impacto, velocidad, ángulo, tamaño, forma, densidad, rigidez, dureza y carga. Los principales factores asociados a la superficie de impacto son su dureza, ductilidad, entre otros.

2.2.1. Erosión en la turbina Pelton

Según Duan [4], la erosión en las turbomaquinas se divide en tres categorías:

1. Micro erosión que se puede encontrar en superficies expuestas a velocidades muy altas donde las partículas de una magnitud de 60 micrones y más pequeñas que se encuentran a altas velocidades de rotación debido a la turbulencia en las capas límites y por lo tanto, provocan una erosión abrasiva relativamente fuerte.
2. Erosión de vórtice de flujo secundario, que se puede encontrar en lo que se conoce como vórtice de herradura en las esquinas de los conductos u otros lugares donde el vórtice y el flujo secundario son causados por el efecto combinado de las capas límites y/o flujo desacelerado.
3. Las aceleraciones perpendiculares a las líneas de corriente provocan la separación de los granos de arena, los cuales siguen una trayectoria diferente al flujo de agua y colisionan con las paredes de los conductos. En los flujos con aceleraciones normales a las líneas de corriente, los granos de arena cuyo tamaño sea de 0.5 mm o más pueden causar daños severos.

Las máquinas hidráulicas que trabajan con números de Reynolds entre 10^6 a 10^8 serán sometidas a las tres categorías de erosión antes mencionadas.

El daño dependerá de las partículas, la forma y tamaño del grano, así como de la velocidad absoluta y el radio hidráulico del conducto considerado. También, se debe considerar que las turbinas más pequeñas sufrirán un daño más severo por la arena que las turbinas más grandes de la misma velocidad específica, debido a los radios hidráulicos y de curvatura más pequeños, lo que genera una cantidad mayor de partículas en contacto con la superficie, causando mayor aceleración normal de las líneas de corriente.

Debido a la altura para la que están diseñadas las turbinas Pelton la velocidad que alcanza el chorro supera los 100 [m/s] y la aceleración máxima de partículas será normalmente mayor a 5000 [m/s²] dependiendo del tamaño de la cuchara y la altura de la planta [4]. Por lo que serán bastante afectadas por la tercera categoría de erosión.

2.3. Modelos de predicción de abrasión hidráulica

La abrasión hidráulica es un fenómeno complejo que ocurre en componentes de turbinas hidráulicas, como las palas, debido al impacto de partículas sólidas en suspensión en el fluido. Comprender y predecir este fenómeno es crucial para mejorar la eficiencia, vida útil y reducir los costos de mantenimiento en las turbinas hidráulicas. En esta sección, se exploran tres de los modelos de predicción de abrasión hidráulica que han sido desarrollados y utilizados en la literatura para estudiar y analizar el fenómeno de erosión en turbinas hidráulicas.

2.3.1. Modelo de Finnie

Es uno de los modelos más usados en la literatura, consiste en dos ecuaciones que dependen del ángulo de impacto y que considera el volumen de material removido por una sola partícula. Considera dos constantes, la primera relaciona la profundidad del contacto con la profundidad de corte, mientras que la segunda constante, relaciona las fuerzas verticales con las horizontales. Además, se considera la deformación plástica y el ángulo de impacto. Este modelo obtiene resultados precisos para ángulos de impacto bajos y/o en materiales dúctiles [15].

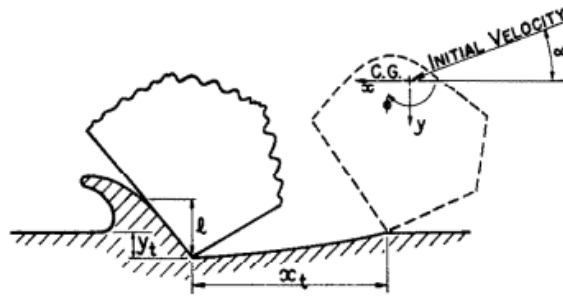


Figura 2.5: Formas en que se produce la abrasión estudiada por Finnie.

(Fuente: Finnie. [15])

En la Figura 2.5 se observa una situación ideal en que una partícula golpea la superficie y remueve material. Inicialmente el vector de velocidad del centro de gravedad de la partícula conforma un ángulo alfa con la superficie.

$$Q = \frac{mV^2}{\rho\psi K} \left(\sin 2\alpha - \frac{6}{K} \sin^2 \alpha \right) \quad (2.2)$$

$$\begin{aligned}
 &\text{si } \tan\alpha \leq \frac{K}{6} \\
 Q &= \frac{mV^2}{p\psi K} \left(\frac{K\cos^2\alpha}{6} \right) \\
 &\text{si } \tan\alpha \geq \frac{K}{6}
 \end{aligned} \tag{2.3}$$

Donde Q es el volumen de material removido por una partícula abrasiva de masa m y velocidad V, ψ , corresponde a la profundidad del corte α al ángulo de incidencia y p la tensión de flujo plástico. K es la relación de la componente vertical de fuerza sobre la partícula con la componente horizontal [16].

La ecuación 2.2, que aplica a ángulos pequeños, corresponde al caso en el que la partícula deja la superficie mientras sigue cortando. La ecuación 2.3, que aplica a ángulos más grandes, corresponde al caso en que el movimiento horizontal de la partícula cesa mientras esta sigue cortando [16].

El valor de K no puede ser fácilmente predicho es por ello que el autor, basándose en mediciones de procesos similares, utiliza un valor de $K=2$ [16].

Al comparar con resultados experimentales, el modelo utilizado por Finnie logra ser bastante similar para ángulos bajos ($\alpha < 30^\circ$). Sin embargo, para grandes ángulos el modelo dista de la realidad. La diferencia con el experimento para ángulos elevados radica en que el impacto más frontal deforma mayormente la superficie, lo que influye en el ángulo de incidencia de las partículas que impactaran más adelante en el tiempo [16].

Luego realizando correcciones empíricas teniendo en cuenta la masa de todas las partículas M en vez de una sola m y que además no todas las partículas remueven material de la superficie (el autor asume arbitrariamente un 50 % de ellas lo hace) se establece la siguiente ecuación [16].

$$\begin{aligned}
 Q &\approx \frac{MV^2}{8p} (\sin 2\alpha - 3\sin^2\alpha) \\
 \alpha &\leq 18,5^\circ
 \end{aligned} \tag{2.4}$$

$$\begin{aligned}
 Q &\approx \frac{MV^2}{24p} \cos^2\alpha \\
 \alpha &\geq 18,5^\circ
 \end{aligned} \tag{2.5}$$

El modelo de erosión de Finnie queda limitado para ángulos sobre 45° en los que se subestima enormemente la pérdida de material.

En particular, para ángulos cercanos a la máxima erosión ($15-20^\circ$) la remoción de volumen viene dado por:

$$Q \approx 0,075 \left(\frac{MV^2}{2} \right) \frac{1}{p} \tag{2.6}$$

Ese 7,5 % de energía cinética de la partícula dividido en el flujo de presión entrega un estimado de volumen removido [16].

2.3.2. Modelo de Nandakumar

Nandakumar se enfoca en desarrollar un modelo que incluya los efectos de partículas abrasivas y el material objetivo, además de capturar varios de los fenómenos observados en otros modelos. El modelo incluye mecanismos basados en la remoción por deformación y por corte [14].

$$\Delta Q = C m \rho_p^{0,15} (V_0 \sin\Theta)^{2,3} D m^{1,1875} d_p^{-0,0625} V_0^{2,375} (\cos\Theta)^2 (\sin\Theta)^{0,375} \tag{2.7}$$

Donde C y D son constantes empíricas, m es la masa de la partícula, ρ es la densidad de la partícula, Θ es el ángulo de impacto, V_0 es la velocidad de impacto y d_p el diámetro de la partícula.

2.3.3. Modelo de Grant y Tabakoff

El modelo de erosión de Grant y Tabakoff [17] se basa en la suposición de que la erosión depende de dos mecanismos diferentes que ocurren a ángulos bajos y altos de impacto, respectivamente. Para calcular la erosión en ángulos intermedios de impacto, se utiliza una combinación de estos dos mecanismos.

La ecuación estudiada en el documento de López [18], basada en el modelo de Grant y Tabakoff, se presenta de la siguiente manera:

$$\epsilon = K1f(\beta_1)(V_{1T}^2 - V_{2T}^2) + f(V_{1N}) \quad (2.8)$$

Donde:

- ϵ representa el daño por erosión por unidad de masa de las partículas de impacto
- $K1$ es el contacto del material
- $f(\beta_1)$ es una función empírica del ángulo de impacto de las partículas
- V_{1T} es la componente tangencial de la velocidad de las partículas entrantes
- V_{2T} es la componente tangencial de la velocidad de las partículas que rebotan
- $f(V_{1N})$ es la componente de la erosión debido a la componente normal de la velocidad

2.4. Acoplamiento

El intercambio de momento entre las fases de partículas discreta se modela según Qui Long [19] considerando un acoplamiento unidireccional, bidireccional o de cuatro direcciones (Ver Figura 2.6).

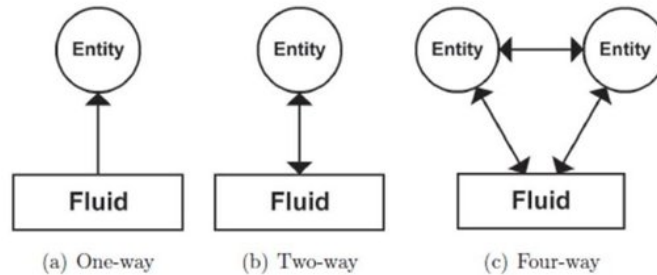


Figura 2.6: Se describen los distintos tipos de acoplamiento entre partícula y fluido (a) Una dirección (b) Dos direcciones (c) Cuatro direcciones

(Fuente: Qui Long [19])

En la modelación de partículas en un flujo, el acoplamiento entre las partículas y el fluido es un aspecto fundamental para la precisión y exactitud de las simulaciones. Existen diferentes niveles de acoplamiento que consideran la interacción entre las partículas y el fluido en distintos grados, y la elección del tipo de acoplamiento es crucial para obtener resultados precisos en función de la aplicación y las condiciones del problema en estudio. A continuación, se presentan y explican brevemente los diferentes tipos de acoplamiento: one-way, two-way y four-way coupling.

- Acoplamiento unidireccional (One-way coupling): Este nivel de acoplamiento considera la influencia del fluido sobre las partículas, pero no tiene en cuenta el efecto de las partículas sobre el fluido (Ver Figura 2.6 a). Es decir, el movimiento de las partículas es influenciado por el fluido, pero las partículas no afectan el flujo del fluido en la simulación. Este enfoque es adecuado para casos en los que la concentración de partículas es baja y su efecto sobre el fluido es despreciable.
- Acoplamiento bidireccional (Two-way coupling): El acoplamiento bidireccional considera la interacción entre las partículas y el fluido en ambas direcciones (Ver Figura 2.6 b). Es decir, el fluido influye en el movimiento de las partículas y, a su vez, las partículas afectan el flujo del fluido. Este enfoque es más preciso y se utiliza en casos en los que la concentración de partículas es moderada y su impacto en el fluido no puede ser ignorado.
- Acoplamiento en cuatro direcciones (Four-way coupling): Este nivel de acoplamiento es el más avanzado y considera no solo la interacción entre las partículas y el fluido, sino también la interacción entre las propias partículas (Ver Figura 2.6 c). Esto incluye el efecto de las colisiones entre partículas y el efecto que tienen estas colisiones en el movimiento de las partículas y el flujo del fluido. El acoplamiento four-way es utilizado en casos en los que las partículas están muy concentradas y las interacciones entre ellas son significativas.

2.5. Condiciones iniciales

En primer lugar, se calcula el campo de flujo de la fase continua del fluido, considerando un flujo de una sola fase, sin tener en cuenta la existencia de partículas dispersas. Posteriormente, se rastrea el movimiento de las partículas de acuerdo con el flujo calculado, tomando en consideración las posiciones y velocidades iniciales.

Para un flujo estacionario, el rastreo de las partículas se lleva a cabo una vez que se ha obtenido la solución convergente del flujo de la fase continua mediante la resolución de las ecuaciones de continuidad y de Navier-Stokes. Por otro lado, en el caso de una simulación de flujo transitorio, el seguimiento de los movimientos de las partículas se realiza al final de cada paso de tiempo en la simulación del flujo.

Se busca mejorar la precisión en la representación del flujo y el movimiento de las partículas, lo cual es esencial para obtener resultados confiables en las simulaciones. La correcta elección de las condiciones iniciales y la adecuada implementación del seguimiento de las partículas son aspectos fundamentales en este proceso.

2.6. Jet Impingement Test (JIT)

Es una metodología basada en las investigaciones de Hattori y Gnanavelu[20, 21]. Consiste en una tobera sumergida en una mezcla continua de agua y de partículas sólidas (fase dispersa) como se muestra en la Figura 2.7, cuya función es concentrar el flujo tanto en la fase continua como en la dispersa en una dirección hacia un material objetivo con forma circular.

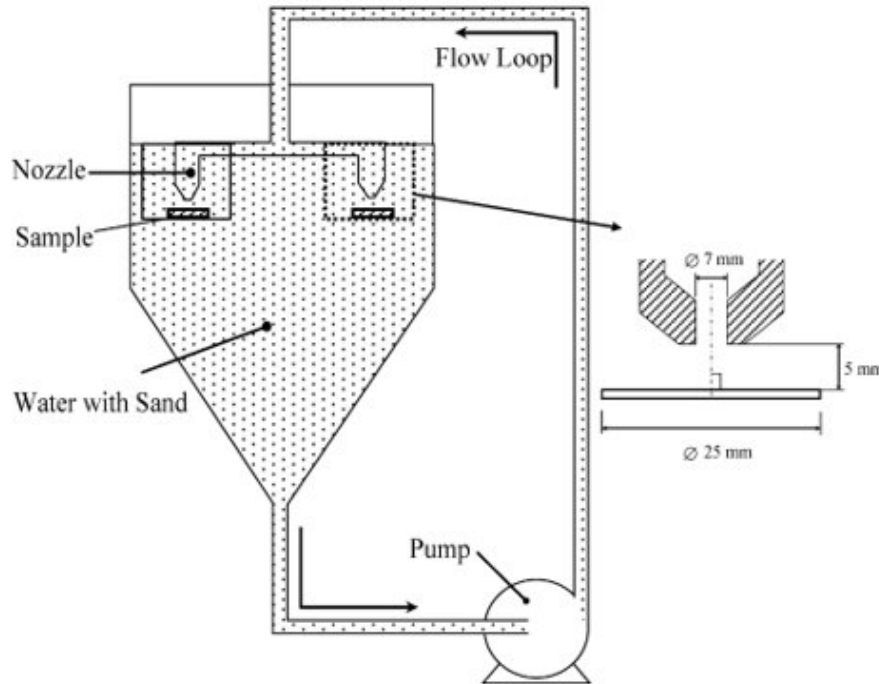


Figura 2.7: Configuración del Jet Impingement Test
(Fuente: Gnanavelu [21])

2.7. Antecedentes y estudios relevantes en erosión hidráulica

En el ámbito de la erosión hidráulica, diversas investigaciones han sido llevadas a cabo con el propósito de entender y predecir el desgaste provocado por las partículas sólidas arrastradas por un fluido. Estos estudios han analizado distintos aspectos, como la comparación de herramientas de simulación, la evaluación de modelos de erosión y la predicción de daños en diferentes materiales. A continuación, se presentan algunos de los trabajos más relevantes en este campo.

Dentro de las investigaciones previas realizadas en la materia, se destacan los trabajos realizados por Alejandro López. El primero de ellos [11], en el que se basa este trabajo y se profundiza más adelante, consiste en comparar OpenFoam y Ansys Fluent para poder verificar las capacidades de erosión de la biblioteca Lagrangiana de OpenFoam. Para esto, se realizan simulaciones de pruebas de impacto de chorro como un método efectivo para estudiar la erosión creada por partículas sólidas arrastradas por un líquido. Se analizan los parámetros de velocidad, ángulo y posición de impacto, obteniendo resultados similares en ambos software.

En otro estudio de López [18], se modela la erosión en uno de los canales de una bomba centrífuga utilizando OpenFoam, con el objetivo de comparar alguno de los modelos de erosión más utilizados en el marco Euleriano-Lagrangiano junto con un estudio comparativo de las tasas de erosión obtenidas con cada modelo. Los resultados arrojaron disparidades entre los modelos debido a los diferentes factores tomados en consideración.

En 2018, López perfecciona su modelo utilizando por primera vez un algoritmo de deformación de malla tridimensional para evaluar los cambios de flujo de los fluidos con la erosión. Al igual que en el modelo del 2015 [11], el caso de validación elegido es el Jet Impingement Test analizado por Hattori [20] y Gnanavelu [21].

Gnanavelu [21] desarrolló un método para predecir el daño por erosión, utilizando una combinación de experimentos de laboratorio estándar y simulaciones CFD. Para lograr su propósito, se generó un mapa de

desgaste universal para el material en cuestión, utilizando una prueba de Jet Impingement para generar el desgaste y la interpretación se realiza mediante una simulación CFD para generar un mapa que proporcione el desgaste en función de la velocidad y el ángulo, con el objetivo de utilizar las simulaciones para predecir desgaste. Los resultados obtenidos por el software fueron similares a los experimentales a pesar de algunos errores señalados por el autor. Finalmente, se concluye que una prueba de impacto de chorro (JIT) proporciona suficientes datos para realizar predicciones de tasas de desgaste dentro de diferentes geometrías.

Hattori [20] realizó pruebas de impacto de chorro (JIT) con partículas erosivas sobre distintos materiales, como aluminio puro, cobre puro, acero inoxidable martensítico fundido, acero inoxidable austenítico y acero fundido. Además, el comportamiento de las partículas se calculó utilizando un código comercial para análisis de flujo, STAR-CD. Los factores considerados fueron la velocidad de impacto, el ángulo de impacto, tamaño, material y frecuencia de las partículas. Los datos obtenidos experimentalmente permitieron a Hattori definir los diferentes métodos de cálculo numérico para predecir la erosión en los distintos casos.

Estos estudios previos han sentado las bases para el análisis y la predicción de la erosión hidráulica en diferentes contextos y materiales. La comprensión de los mecanismos involucrados y la evaluación de distintos modelos y herramientas de simulación han sido fundamentales para avanzar en el conocimiento de este fenómeno y en la implementación de estrategias para mitigar sus efectos en distintos sistemas y componentes.



3 | Método Computacional

En esta memoria se utiliza la versión 8 de OpenFOAM para desarrollar las simulaciones que se lleva a cabo para el análisis. Los archivos utilizados de los directorios en cada una de las simulaciones se encuentran en los Apéndices A, B y C. Las geometrías fueron desarrolladas en Autodesk Inventor, a excepción de la pala de la turbina que debido a la complejidad de su geometría fue obtenida desde GrabCad [22] algunas modificaciones necesarias fueron realizadas en Paraview.

3.1. Computational Fluid Dynamics (CFD)

La Dinámica de Fluidos Computacional (CFD, por sus siglas en inglés) es una rama de la ingeniería que se ocupa de resolver numéricamente las ecuaciones que describen el comportamiento de los fluidos, tales como el movimiento, la transferencia de calor, la reacción química, la mezcla y la dispersión de sustancias, entre otros procesos. A través de la simulación computacional, CFD permite analizar el comportamiento de los fluidos en diferentes situaciones, lo que permite optimizar el diseño y el funcionamiento de sistemas que involucren fluidos, como aviones, automóviles, turbinas, equipos de procesamiento químico, entre otros [23] [24].

La Dinámica de Fluidos Computacional (CFD) es una herramienta poderosa que se utiliza en una amplia variedad de aplicaciones en la industria, la academia y la investigación. Algunas de las aplicaciones más comunes de CFD incluyen la optimización del diseño de aerodinámica de vehículos como automóviles, aviones y cohetes; la simulación de procesos químicos y de combustión en sistemas de energía y propulsión; la predicción de la circulación sanguínea en el cuerpo humano y la simulación de procesos de mezcla en la industria alimentaria y de procesamiento químico. También se utiliza en la simulación de tsunamis, inundaciones y otros desastres naturales para predecir su impacto y desarrollar estrategias de mitigación de riesgos. En resumen, el CFD es una herramienta versátil y valiosa que se utiliza en una amplia variedad de campos para comprender y mejorar el comportamiento de los fluidos en sistemas complejos [25] [26].

El libro “An Introduction to Computational Fluid Dynamics” de Versteeg y Malalasekera [27] describe de manera detallada el funcionamiento del CFD. En esencia, el CFD utiliza métodos numéricos para resolver las ecuaciones fundamentales de la dinámica de fluidos. Estas ecuaciones describen el comportamiento de un fluido, como la velocidad, la presión y la densidad, y se resuelven numéricamente utilizando una malla de puntos discretos que cubren el dominio de interés.

Una vez que se ha generado la malla, el dominio se divide en pequeños elementos para los cuales se establecen las ecuaciones de la dinámica de fluidos. Estas ecuaciones se resuelven para cada elemento, utilizando los valores de las variables en los puntos nodales. Los valores de las variables en cada punto nodal se calculan utilizando un método de interpolación.

Existen varios métodos numéricos utilizados en el CFD, incluyendo el método de diferencias finitas, el método de volúmenes finitos y el método de elementos finitos. Cada método tiene sus propias ventajas y desventajas y se utiliza según la aplicación.

Una vez que se han resuelto las ecuaciones de la dinámica de fluidos para todos los elementos de la malla, se pueden calcular las propiedades del fluido en cualquier punto dentro del dominio. Estas propiedades

pueden incluir la velocidad, la presión y la densidad, así como otros parámetros de interés, como el flujo de calor y la concentración de especies químicas.

En resumen, el CFD es una herramienta poderosa que permite a los ingenieros y científicos simular el comportamiento de los fluidos en sistemas complejos. A través de la generación de una malla de puntos discretos y la resolución numérica de las ecuaciones de la dinámica de fluidos, el CFD puede proporcionar información valiosa sobre el flujo de fluidos en una amplia variedad de aplicaciones, desde la optimización del diseño de vehículos hasta la simulación de procesos químicos y biológicos.

3.1.1. Ecuaciones Gobernantes

Las ecuaciones gobernantes en CFD están representadas por las leyes de conservación de la física [13].

1. Conservación de la masa - Ecuación de Continuidad

$$\frac{\partial \rho}{\partial t} + \text{div}(\rho V) = 0 \quad (3.1)$$

2. Conservación del momentum - La ecuación de Navier-Stokes

$$\rho \frac{DV}{Dt} = \rho g + \nabla \tau'_{ij} - \nabla p \quad (3.2)$$

3. Conservación de la energía - Primera ley de la termodinámica

$$\rho \frac{Dh}{Dt} = \frac{Dp}{Dt} + \nabla(k \nabla T) - \Phi \quad (3.3)$$

Donde $\frac{DV}{Dt} = \frac{\delta V}{\delta t} + (V \cdot \nabla)V$, el termino Φ en la ecuación 3.3 es la función disipación y el tensor de esfuerzos viscosos τ'_{ij} es

$$\tau'_{ij} = \mu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) + \delta_{ij} \lambda \cdot \text{div} V \quad (3.4)$$

Asumiendo un fluido Newtoniano ($\tau'_{ij}=0$) y flujo incompresible, la ecuación de Navier-Stokes se simplifica a:

$$\rho \frac{DV}{Dt} = \rho g - \nabla p + \mu \nabla^2 V \quad (3.5)$$

En la ecuación de continuidad para un fluido estacionario y densidad uniforme.

$$\nabla \cdot V = 0 \quad (3.6)$$

3.1.2. Mallado

En OpenFOAM la geometría esta dividida en pequeños volúmenes de control llamados celdas. El conjunto de estas celdas a lo largo de todo el dominio es lo que se llama malla. Cada celda tiene un centro y caras que rodean la celda. Las caras internas conectan las celdas en una geometría, mientras que los bordes de las caras, también llamados parches, son parte de los bordes del dominio computacional [28].

Una malla valida debe satisfacer las siguientes propiedades

- Todo el dominio computacional necesita estar cubierto de celdas
- Toda las celdas deben ser convexas y el centro de celda debe estar dentro de ella

- Todas las celdas deben estar cerradas

No existe una definición exacta de lo que es una “buena” malla, pero la calidad de ésta es generalmente medida por: no-ortogonalidad, asimetría, relación de aspecto, suavidad y número de celdas.

- No-ortogonalidad: Está definido como el ángulo entre el vector normal a la cara común entre dos celdas y el vector entre los centros de dos celdas adyacentes (ver Figura 3.1). Errores numéricos significantes pueden ocurrir si las celdas son no-ortogonales entre si. OpenFOAM permite un valor predeterminado máximo para la ortogonalidad de 70° grados, pero lo deseable es mantener el ángulo lo más pequeño posible.

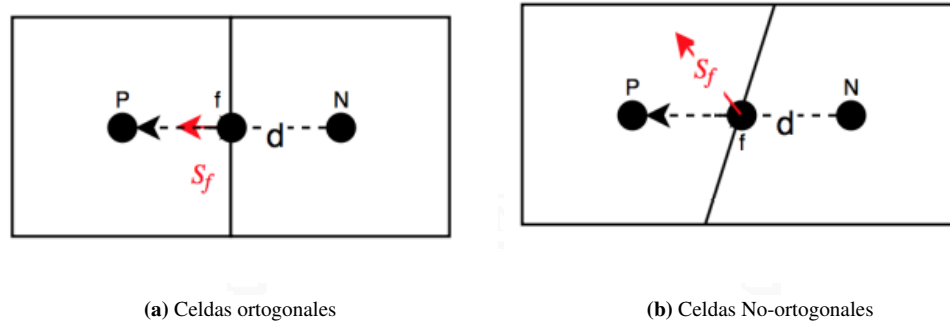


Figura 3.1: Ortogonalidad de las celdas

(Fuente: CFD in OF Gullberg [28])

- Asimetría: La asimetría surge cuando hay una discrepancia entre la ubicación del centro de la cara y donde el vector de centro a centro se encuentra con la cara (Figura 3.2). El valor ideal para la asimetría es cero y el umbral de asimetría en OpenFOAM es cuatro.

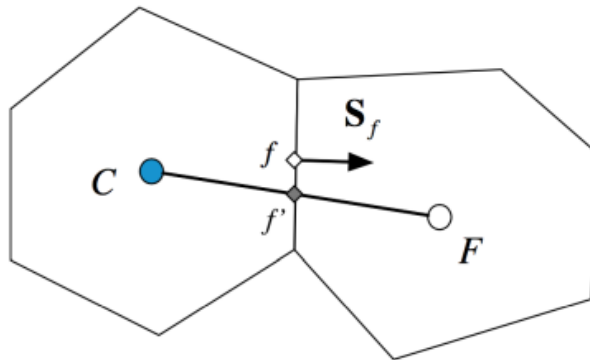


Figura 3.2: Asimetría celdas

(Fuente: CFD in OF Gullberg [28])

- Relación de aspecto: La relación de aspecto es la relación entre el ancho máximo y mínimo de una celda. Las relaciones de aspecto grandes disminuirán la eficiencia y la precisión de los solucionadores lineales en OpenFOAM. Además, las relaciones de aspecto grandes introducen errores de interpolación al difuminar el gradiente, lo que provoca la difusión numérica. Las relaciones de aspecto grandes deben evitarse especialmente en áreas donde el flujo cambia rápidamente y el gradiente es grande. La relación de aspecto debe ser lo más cercana posible a 1, pero el umbral estándar en OpenFOAM es 1000.
- Suavidad: También es importante evitar saltos repentinos en el tamaño de las celdas manteniendo la malla uniforme. Los saltos repentinos en el tamaño causarán un error de interpolación, agregando difusión a la solución. Para garantizar que la malla sea suave, la diferencia entre el volumen de celda mínimo y máximo debe ser lo más pequeña posible.

- Número de celdas: El último paso en un procedimiento de mallado es un estudio de refinamiento, donde la densidad de la malla aumenta al agregar más celdas. La malla se hace cada vez más fina hasta que se alcanza la independencia de la cuadrícula, lo que significa que no hay cambios significativos en la solución cuando se refina la cuadrícula. Los procesos de refinamiento también pueden incluir agregar celdas localmente dentro de las capas límite para capturar características de flujo significativas. Sin embargo, es importante no "sobremallar", ya que esto puede hacer que el modelo se vuelva ineficiente e irresoluble.

3.1.3. Criterio de Courant-Friedrichs-Levy

El número de Courant es un número adimensional que es usado en simulaciones de fluidos computacionales dinámicos (CFD) para evaluar el paso de tiempo requerido por una simulación transiente, para un tamaño de malla y un flujo de velocidad dado. El número de Courant consiste en el cociente entre el paso de tiempo y el intervalo de espacial [29]

$$C = \frac{\Delta t}{\frac{\Delta x}{u}} = \frac{u \Delta t}{\Delta x} \quad (3.7)$$

Donde u es la velocidad del fluido, Δt es el paso de tiempo y Δx es el paso espacial de la malla.

Para un caso tridimensional esto se generaliza a una escala teniendo en cuenta la dimensión del volumen de control. El criterio CFL significa que si el número de Courant es menor o igual a 1,0, el fluido no viaja más de una celda por cada paso de tiempo. Para análisis explícitos dependientes del tiempo, este criterio es necesario para la estabilidad y el número de Courant debe limitarse por debajo de 1,0.

3.1.4. Discretización

Las ecuaciones gobernantes tienen que ser discretizadas en tiempo y espacio para ser resueltas numéricamente. La integración de volumen finito gaussiano se basa en la suma de valores en las caras de las celdas, que deben interpolarse desde los centros de las celdas, y es la opción más común para la discretización en OpenFOAM. Los esquemas de discretización pueden ser de cualquier orden, pero los esquemas de orden superior serán más complejos y pueden dar problemas de estabilidad. OpenFOAM proporciona una gran flexibilidad en cuanto a la elección de esquemas de discretización e interpolación entre puntos. Comúnmente se utiliza un esquema de diferenciación central basado en los dos puntos vecinos más cercanos a cada lado del centro de la celda, el esquema lineal de Gauss. Esto da un esquema preciso de segundo orden como se ilustra a continuación.

$$\frac{\partial u}{\partial x} = \frac{u_{i+1} - u_{i-1}}{2\delta x} + O(\delta x^2) \quad (3.8)$$

Para la discretización en el tiempo se utiliza el esquema de Forward Euler, dando una precisión de primer orden:

$$\frac{\partial u}{\partial t} = \frac{u_i^{n+1} - u_i^n}{\delta t} + O(\delta t) \quad (3.9)$$

En conjunto, esto se conoce como el esquema de espacio central de avance de tiempo, que proporciona una convergencia de primer orden en el tiempo y una convergencia de segundo orden en el espacio. El esquema de discretización Upwind que se muestra en la ecuación 3.10 se usa comúnmente para la discretización en el espacio cuando hay problemas de estabilidad, pero puede dar una precisión que no es satisfactoria.

$$\frac{\partial u}{\partial t} = \frac{u_i^n - u_{i-1}^n}{\delta x} + O(\delta x) \quad (3.10)$$

La opción vanLeer es una variante de la disminución de la variación total. Puede, por ejemplo, usarse junto con el esquema de Gauss para resolver mejor los campos que son propensos a tener áreas con fuertes choques o grandes discontinuidades. OpenFOAM ofrece versiones mejoradas de algunos de los esquemas para escalares que deben estar estrictamente delimitados. Puede encontrar una descripción detallada de estas opciones en la Guía del usuario de OpenFOAM. [30].

3.1.5. Soluciones iterativas

Después de discretizar las ecuaciones gobernantes lo que procede es resolverlas mediante métodos numéricos iterativos. Los solucionadores de matrices de ecuaciones funcionan mediante iteraciones, esto significa que producirán un residuo como medida de error en la solución. Para que la solución sea más precisa el error debe ser mas pequeño. Se debe especificar una tolerancia del solucionador para controlar la precisión del solucionador lineal. Las soluciones delimitadas por tolerancias muy bajas puede llevar mucho tiempo y, por lo tanto, es importante considerar un equilibrio adecuado entre precisión y la cantidad de ciclos de iteración necesarios para lograr la convergencia.

3.1.6. Modelos de turbulencia

Todos los flujos relevantes para los problemas de ingeniería, desde los más simples hasta los más avanzados, se vuelven inestables y turbulentos por encima de cierto número de Reynolds. Por lo tanto, existe la necesidad de herramientas que puedan representar los efectos de la turbulencia. Hay varias formas de abordar la turbulencia numéricamente, que van desde la simulación numérica directa (DNS), a través de la simulación de grandes remolinos (LES) más simple, hasta el promedio de Reynolds Navier-Stokes (RANS). Este último es el más utilizado para aplicaciones prácticas e industriales[31].

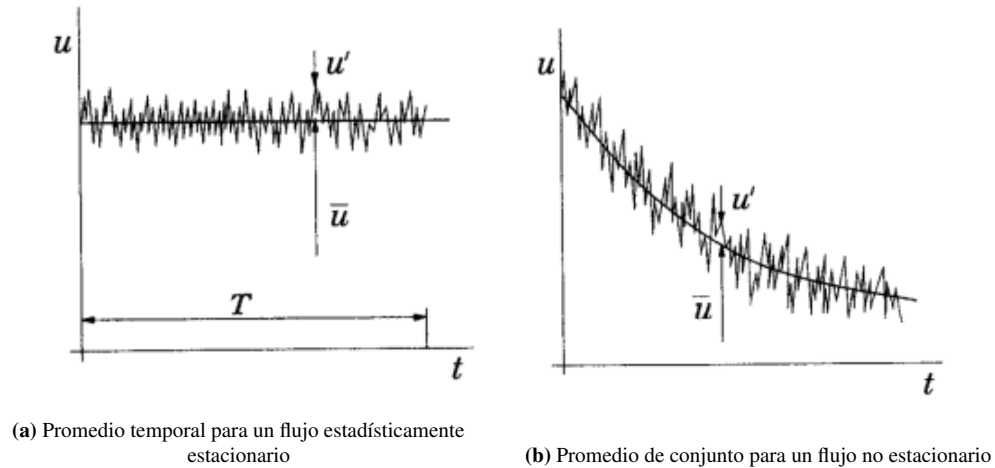


Figura 3.3: Variación de velocidad en flujos turbulentos

(Fuente: Computational methods for fluid dynamics [32])

RANS supone que para ventanas de tiempo grandes, en relación con el período de las fluctuaciones, la velocidad media $\bar{U}$ varía lentamente con el tiempo, como se ilustra en la Figura 3.3. El mismo método se utiliza para la variable de presión. La variable U se puede representar como la media $\bar{U}$ más un valor fluctuante U' , de modo que[31].:

$$U = \bar{U} + U' \quad (3.11)$$

Donde

$$\bar{U} = \frac{1}{T} \int_{t_0}^{t_0+T} U dt = \bar{U} + U' \quad (3.12)$$

Las velocidades y la presión promedio se pueden insertar en la ecuación de Navier-Stokes.

$$\rho \frac{D\bar{V}}{Dt} = \rho g + \nabla \tau_{ij} - \nabla \bar{p} \quad (3.13)$$

$$\tau_{ij} = \mu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) + \rho v_i' v_j' \quad (3.14)$$

Promediar las ecuaciones de momento produce seis incógnitas adicionales, que requieren ecuaciones adicionales para ser resueltas. Por lo tanto, se han desarrollado modelos de turbulencia para predecir las tensiones de Reynolds y cerrar el sistema de ecuaciones de flujo medio (Ecuaciones 3.5 3.13). Estos se clasifican por el número de ecuaciones de transporte extra que necesitan ser resueltas [31].

No. de ecuaciones de transporte extra	Modelo de Turbulencia
Cero	Modelo longitud de mezcla
Uno	Modelo Spalart-Allmaras
Dos	Modelo k- ϵ
Dos	Modelo k- ω
Dos	Modelo de estrés algebraico
Siete	Modelo de estrés de Reynolds

Tabla 3.1: Modelos de turbulencia [27]

El modelo k- ϵ estándar está ampliamente probado y bien establecido, y se sabe que tiene un rendimiento excelente para muchos flujos relevantes para la industria. Sin embargo, tiene un rendimiento deficiente cuando se trata de ciertos flujos no confinados, así como flujos con grandes tensiones adicionales (por ejemplo, capas límite curvas, flujos en remolino) y flujos giratorios [31]. De acuerdo con Perrig [33], el modelo k- ϵ ha mostrado un rendimiento deficiente en las regiones de la pala donde se sospecha que el flujo presenta grandes esfuerzos de corte.

El modelo k- ω de Wilcox no requiere funciones de amortiguamiento de pared en aplicaciones con números de Reynolds bajos, por lo que es más preciso y robusto en áreas cercanas a la pared. Sin embargo, es muy sensible al valor de flujo libre en ω , ya que la viscosidad de remolino se vuelve indeterminada o infinita cuando $k \rightarrow 0$ y $\omega \rightarrow 0$ [27].

El modelo k- ϵ depende mucho menos que el modelo k- de los valores asumidos en el flujo libre. Menter [34] propuso un modelo híbrido, mezclando los modelos k- ϵ y k- ω en regiones cercanas a la pared, y usando el modelo estándar k- ϵ en áreas de alta turbulencia lejos de la pared. Esto se conoce como el modelo de transporte de esfuerzo cortante (SST).

3.2. OpenFOAM (Open Field Operation And Manipulation)

OpenFOAM es un software de simulación numérica de dinámica de fluidos (CFD) de código abierto y gratuito. Fue desarrollado por OpenCFD Ltd. y se utiliza ampliamente en la industria, la academia y la investigación [35].

OpenFOAM utiliza el método de volúmenes finitos para resolver las ecuaciones de la dinámica de fluidos y proporciona una amplia gama de solvers para una variedad de aplicaciones, desde flujo laminar hasta turbulento, incluyendo flujo multifásico y transferencia de calor. Además, el software cuenta con una amplia gama de herramientas de preprocesamiento y posprocesamiento para facilitar la generación de mallas, la definición de condiciones de contorno y la visualización de los resultados[27].

Lo que hace que OpenFOAM sea único es su arquitectura modular, que permite a los usuarios agregar, modificar y combinar componentes del software según sea necesario. Esto significa que se pueden crear solvers y herramientas personalizadas para abordar problemas específicos, lo que hace que OpenFOAM sea una herramienta muy flexible y adaptable [36].

Otra ventaja de OpenFOAM es su comunidad de usuarios activa y comprometida, que comparte solvers, herramientas y técnicas para mejorar continuamente el software. Además, OpenFOAM tiene una licencia de código abierto, lo que significa que los usuarios tienen acceso completo al código fuente y pueden modificarlo para sus propias necesidades [37].

En resumen, OpenFOAM es un software de CFD de código abierto y gratuito que utiliza el método de volúmenes finitos para resolver las ecuaciones de la dinámica de fluidos. Con su arquitectura modular y su comunidad activa de usuarios, es una herramienta flexible y adaptable para una amplia gama de aplicaciones en la industria, la academia y la investigación [38].

Para facilitar el uso y la extensibilidad de OpenFOAM, existen varias herramientas y utilidades que se proporcionan con el software. Estas herramientas permiten al usuario definir y configurar diferentes parámetros de simulación, como las condiciones de contorno y los modelos de turbulencia, entre otros. Además, OpenFOAM cuenta con una gran variedad de solvers, preconditionadores y métodos numéricos que permiten resolver problemas de flujo de manera eficiente y precisa[39].

La versatilidad de OpenFOAM también se refleja en la amplia gama de formatos de archivos que admite. OpenFOAM puede leer y escribir datos de entrada y salida en varios formatos comunes de CFD, como ANSYS CFX, Fluent y STAR-CD, entre otros. También puede leer y escribir archivos de geometría en varios formatos, como STL, IGES y STEP, lo que facilita la importación y exportación de modelos [36].

Además de las herramientas y utilidades proporcionadas con el software, OpenFOAM también cuenta con una amplia documentación y recursos en línea que permiten a los usuarios aprender y utilizar el software de manera efectiva. Esto incluye manuales de usuario detallados, tutoriales, cursos en línea y una comunidad en línea activa que proporciona soporte y ayuda a los usuarios [40].

En otras palabras, OpenFOAM es un software de CFD altamente modular y adaptable que utiliza el método de volúmenes finitos para resolver ecuaciones de flujo. Es de código abierto, lo que permite a los usuarios personalizar y extender sus funcionalidades según sus necesidades. Además, cuenta con una gran variedad de herramientas y utilidades, así como con una amplia documentación y recursos en línea, lo que lo hace una opción atractiva para la simulación de problemas de flujo en diferentes áreas de aplicación [41].

OpenFOAM también cuenta con herramientas de visualización, como Paraview, que permiten visualizar los resultados de la simulación en 2D o 3D, y proporcionan herramientas para manipular y analizar los datos generados. Además, cuenta con un amplio conjunto de utilidades que facilitan el manejo y la gestión de las simulaciones, como por ejemplo el control de versiones, el monitoreo y la visualización de los procesos en ejecución, y el manejo de la memoria y del almacenamiento de datos.

3.3. Métodos de descripción de partícula

Los métodos de descripción de partícula Euler y Lagrange son dos enfoques diferentes para el análisis de fluidos. En la descripción Euler, se sigue el movimiento del fluido en un espacio fijo (Ver Figura 3.4), donde las propiedades del fluido (como la densidad y la velocidad) se miden en cada punto del espacio en

un instante dado. Por lo tanto, en la descripción Euler, el fluido se considera como un campo continuo de propiedades en el espacio [27].

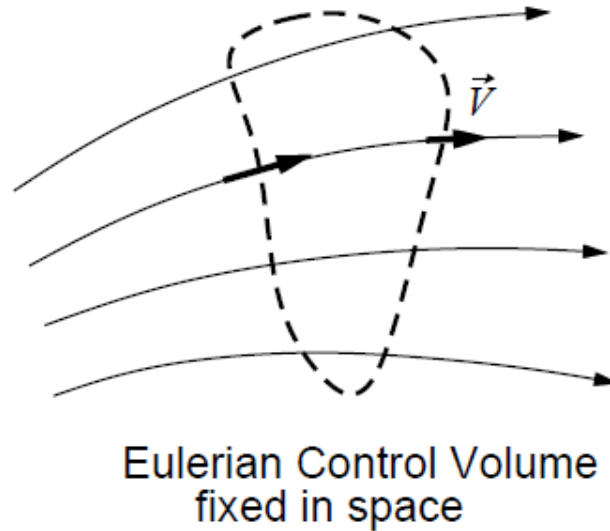


Figura 3.4: Describe el rastreo de partícula mediante el método de Euler
(Fuente:physicsstackexchange [42])

Por otro lado, en la descripción Lagrange, se sigue el movimiento de cada partícula de fluido individual en el espacio (Ver Figura 3.5), y se mide cómo cambian las propiedades del fluido a medida que se mueve cada partícula. En la descripción Lagrange, el fluido se considera como un conjunto de partículas individuales que se mueven a través del espacio [26].

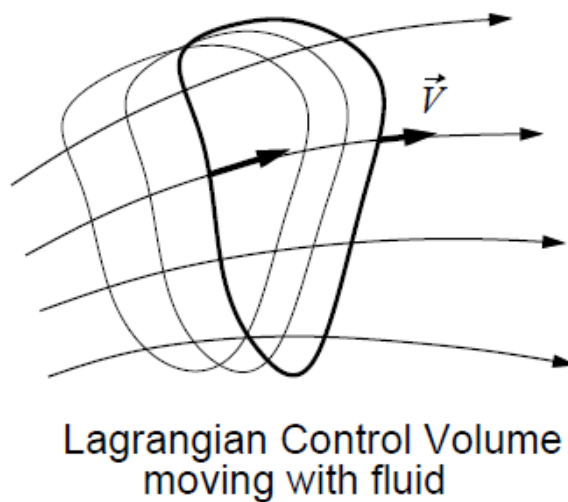


Figura 3.5: Describe el rastreo de partícula mediante el método de Lagrange
(Fuente:physicsstackexchange [42])

Cada enfoque tiene sus ventajas y desventajas. En la descripción Euler, es más fácil analizar el flujo a nivel global y obtener una vista completa de todo el campo de flujo. En la descripción Lagrange, se puede analizar la evolución del fluido en función del tiempo y de las posiciones de las partículas individuales, lo que es útil para estudiar los efectos de las fuerzas y la interacción de partículas en el fluido [23].

El método Euler-Lagrange es una técnica utilizada en dinámica de fluidos para describir el movimiento de partículas suspendidas en un fluido. Este método combina las descripciones Euleriana y Lagrangiana del movimiento de partículas en un fluido, permitiendo un análisis más completo del comportamiento del fluido [43].

El método Euler-Lagrange combina estas dos descripciones mediante la inclusión de partículas en el análisis Euleriano del fluido. Las partículas se consideran como elementos discretos que se mueven en respuesta al campo de velocidades del fluido. El movimiento de cada partícula se describe Lagrangianamente y se resuelve junto con las ecuaciones Eulerianas del fluido [44].

El método Euler-Lagrange es particularmente útil para estudiar flujos complejos en los que las partículas interactúan con el fluido, como en la combustión, la sedimentación, la dispersión de partículas y la formación de burbujas. Además, puede ser utilizado para estudiar la dinámica de partículas en medios porosos, como en la filtración de aire y agua a través de materiales porosos [27].

Aunque el método Euler-Lagrange es más complejo que la descripción Euleriana o Lagrangiana por separado, su combinación permite un análisis más completo del comportamiento de fluidos y partículas en una variedad de aplicaciones prácticas [45].

Existen dos fases de dispersión en las que se encuentra un líquido con material particulado, una es la fase dispersa en la que las partículas se encuentran de forma heterogénea con el líquido; en cambio la fase continua las partículas están mezcladas principalmente de manera homogénea. De este modo la fase continua es modelada usando la aproximación de Euler y la dispersa la de Lagrange, es por ello que se denomina como una aproximación Euler-Lagrange en este tipo de casos [11].

Para la fase continua de fluido, el campo de flujo se calcula como un flujo de una sola fase sin la existencia de una fase de partícula dispersa. Lo cual se archiva y se utiliza como condición inicial después [11].

Las principales ecuaciones N-S son de continuidad y conservación de momento. (Por lo general se considera un fluido incompresible).

$$\vec{\nabla} \cdot \vec{v} = 0 \quad (3.15)$$

$$\rho \frac{\delta \vec{v}}{\delta t} + \rho(\vec{v} \cdot \vec{\nabla}) \vec{v} = -\vec{\nabla} p + \mu \Delta \vec{v} + \rho \vec{f}_e \quad (3.16)$$

OpenFOAM tiene solucionadores para flujos con régimen estacionario (ej.: simpleFoam) y para régimen transitorio (ej.: pimpleFoam) [11].

En el enfoque Lagrange, el movimiento de partícula viene determinado por el equilibrio de fuerzas de la partícula y las condiciones en las que se libera la partícula (condiciones iniciales). Para modelar la fase de partículas discretas, se forman primero las ecuaciones de movimiento de las partículas, en base a la Ley de Newton [11].

$$m_p \frac{du_p}{dt} = F_p \quad (3.17)$$

$$\frac{dx_p}{dt} = u_p \quad (3.18)$$

Donde u_p es la velocidad y x_p son la velocidad y posición respectivamente de la partícula. Mientras que m_p es la masa y F_p la suma de fuerzas que actúan en la partícula [11].

3.4. Modelamiento de la fase discreta

Para llevar a cabo la simulación computacional es necesario representar las partículas dentro de un fluido, esto se logra usando la librería lagrangiana de OpenFOAM 8, en específico el solver DPMFoam que consiste en un solucionador transiente de partículas en un medio continuo [11]. Para obtener los datos tanto de las partículas erosivas como del fluido en el que se mueve se utiliza el enfoque Euler-Lagrange mencionado en la sección 3.3.

El acoplamiento utilizado es unidireccional, por lo tanto sólo se considera la influencia del fluido sobre las partículas, en este caso se verán afectadas sólo por la fuerza de arrastre (se desprecia la fuerza de gravedad y todo el resto de las fuerzas) por lo que el balance de fuerzas sobre la partícula de acuerdo a la ecuación 3.19 [11].

$$F_p = m_p \frac{du_p}{dt} = F_D \quad (3.19)$$

Donde F_D es la fuerza de arrastre sobre la partícula que es obtenida mediante las ecuaciones 3.20 y 3.21

$$F_D = m_p \frac{18\mu}{\rho_p d_p^2} \frac{C_D Re_p}{24} (u - u_p) \quad (3.20)$$

$$C_D = \begin{cases} \frac{24}{Re_p} & : Re_p < 1 \\ \frac{24}{Re_p} (1 + 0.15 Re_p^{0.687}) & : 1 \leq Re_p \leq 1000 \\ 0.44 & : Re_p > 1000 \end{cases} \quad (3.21)$$

Donde el coeficiente de arrastre se compone de tres partes que desde arriba hacia abajo corresponden a la ley de Stokes, la región de transición, y la ley de Newton [11].

3.5. Enfoque de la solución

Para lograr una representación precisa de la erosión, es esencial enfocarse en la zona de impacto del chorro, que es la zona más compleja del flujo y requiere un mallado adecuado y complejo. Dado que también se deben simular las partículas, el costo computacional es significativo, especialmente en la pala de la turbina, debido a la complejidad de su geometría. Por lo tanto, se decidió simular solo la mitad de la pala para reducir los tiempos de simulación, que ya son bastante largos.

Para reducir los tiempos de cálculo y la complejidad técnica, se optó por modelar solo una pala sin movimiento. Mantener la pala estática es una opción viable que ayuda a disminuir la complejidad técnica y a mejorar la eficiencia de la simulación.

Para garantizar la precisión de la simulación, se realiza una validación replicando el trabajo de López [11]. Se llevarán a cabo dos simulaciones, con la boquilla a 5 mm y 25 mm del objetivo, y se compararán los resultados con los obtenidos por López. Para visualizar los resultados, se utiliza la plataforma Paraview y un programa creado en Python (Anexo D).

Una vez validada la simulación, se procede a simular la erosión de la pala de una turbina Pelton utilizando el experimento realizado por Kumar [1] como guía. La malla se ajustará en función de las dimensiones proporcionadas por la experiencia.

3.6. Estructura de OpenFOAM

OpenFOAM es un software de dinámica de fluidos computacional de código abierto que utiliza el método de volúmenes finitos para resolver las ecuaciones de conservación de masa, momentum y energía de los fluidos. La estructura de OpenFOAM se divide en una serie de directorios y archivos que contienen los componentes necesarios para llevar a cabo la simulación de flujo [46] como se puede ver en la Figura 3.6.

En la carpeta “applications”, se encuentran los diferentes casos de simulación y las herramientas necesarias para ejecutarlos. Cada caso de simulación tiene su propia carpeta, que contiene un archivo de configuración y todos los archivos necesarios para la simulación [47].

La carpeta “src” contiene los archivos fuente de OpenFOAM, incluyendo las bibliotecas, utilidades y aplicaciones. En la carpeta “solver”, se encuentran los solucionadores numéricos utilizados para resolver las ecuaciones de conservación en los casos de simulación. Cada solucionador se puede personalizar para adaptarse a diferentes situaciones de flujo [40].

El directorio “system” contiene los archivos de configuración del sistema, incluyendo el archivo “controlDict”, que especifica los parámetros de la simulación, como el tiempo total de simulación, la frecuencia de muestreo, la precisión numérica y otros detalles [40].

La carpeta “constant” contiene archivos de configuración que se mantienen constantes durante toda la simulación, como las condiciones de contorno y las propiedades del fluido [40].

Finalmente, el directorio “polyMesh” contiene los archivos de malla para la simulación. La malla se genera a partir de un archivo de descripción de geometría y se divide en celdas finitas para permitir la resolución numérica de las ecuaciones de flujo [40].

Se puede decir que la estructura de OpenFOAM se divide en directorios y archivos que contienen todos los componentes necesarios para llevar a cabo la simulación de flujo, incluyendo las herramientas de simulación, solucionadores numéricos, archivos de configuración, propiedades del fluido, condiciones de contorno y malla. Cada carpeta y archivo cumple un papel importante en la configuración y ejecución de la simulación de flujo.

La estructura general de estos archivos y carpetas se encuentra en la Figura 3.6. Es posible que para ciertos casos se requiera de herramientas adicionales para algunos procedimientos en los que OpenFOAM no tenga lo requerido para realizar análisis, estas serán señaladas mas adelante.

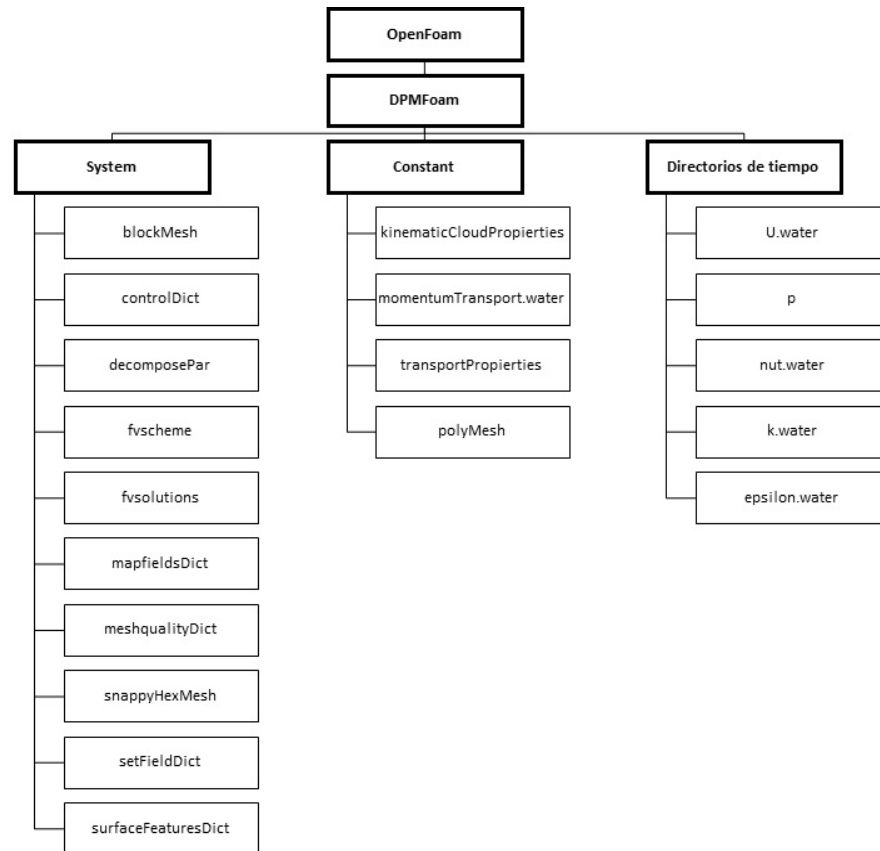


Figura 3.6: Estructura de OpenFOAM [40]

En general y para todos los solvers de OpenFOAM existen tres directorios principales que conforman la estructura principal de OpenFOAM, estos son:

1. **Directorio System** : El directorio system contiene archivos asociados a la resolución y ejecución de la simulación
2. **Directorio Constant** : Este directorio contiene archivos que están relacionados con la física del problema, incluida la malla y las propiedades físicas que requiere el solver. En este directorio se encuentran las geometrías adicionales que se quieran incluir de forma externa y el archivo relacionado con la física de las partículas.
3. **Directorios de tiempo ("0")** : Contiene las condiciones iniciales para la simulación, ya sea presión, velocidad, gravedad, viscosidad, según se requiera.

3.6.1. System

3.6.1.1. blockMesh

Es una herramienta que permite delimitar el dominio espacial en el que se va a trabajar, además de la creación de geometrías simples.

3.6.1.2. snappyHexMesh

Es una herramienta utilizada para la incorporación de geometrías mas complejas a la malla, las geometrías son diseñadas en software de modelamiento 3D y se pueden incorporar mediante este directorio en los formatos OBJ y STL. El directorio snappyHexMesh incorpora la geometría y su mallado a la ya generada malla por blockMesh, este último comando siempre debe ejecutarse antes.

3.6.1.3. controlDict

El diccionario controlDict se utiliza para especificar los controles principales del caso. Esto incluye, por ejemplo, información de temporización, formato de escritura y bibliotecas opcionales que se pueden cargar en tiempo de ejecución.

3.6.1.4. decomposePardict

Es el diccionario asociado el método de computo paralelo utilizado por OpenFOAM, este consiste en ejecutar la simulación en procesadores paralelos y así disminuir el tiempo de los cálculos.

Los procesadores utilizados son cuatro y el tipo de descomposición que se usara es la scotch.

La descomposición del tipo scotch no requiere entradas geométricas proporcionadas por el usuario e intenta minimizar el numero de limites del procesador.

3.6.1.5. fvSchemes

En este diccionario se especifican los métodos numéricos para resolver las distintas instancias como gradientes, derivadas, interpolaciones.

3.6.1.6. fvSolution

En este diccionario se controlan las ecuaciones de los solvers, tolerancias y algoritmos.

3.6.1.7. setFieldsDict

Permite establecer las condiciones del entorno, para este caso un dominio inicial lleno de aire.

3.6.2. Constant

3.6.2.1. kinematicCloudPropierties

Este directorio define las propiedades y comportamiento de las nubes de partículas utilizadas en la simulación. Al ser uno de los directorios más importantes se describen las distintas entradas que se utilizaron en esta simulación a continuación:

1. **Solution:** Configuración de la solución de partículas

- a) **Active:** Activa o desactiva la inyección de nubes de partículas, en este caso el valor asignado sera true.
- b) **Coupled:** Define el acoplamiento de la partícula con el fluido, false para unidireccional y true para bidireccional. True será el seleccionado.
- c) **transient:** Establece si la simulación sera transiente o en estado estacionario. (false)

- d) **cellValueSourceCorrection** correccion de los valores de celda dentro del intervalo de tiempo lagrangiano. (on)
 - e) **maxCo**: máximo numero de Courant (1)
- 2. **interpolationSchemes**: interpolación de esquemas para campos continuos
- 3. **averagingMethod**: promedio lagrangiano de las celdas y volúmenes duales (tetraedricos) para doble malla
- 4. **integrationSchemes**: define los esquemas de integración del campo de velocidad U
- 5. **constantProperties**: propiedades de las partículas.
 - a) **rho0**: densidad de la partícula
 - b) **alphaMax**: máxima fracción de fase dispersa
 - c) **poissonsRatio**: relación de la deformación transversal y axial de las partículas
 - d) **youngsModulus**: valor del modulo de Young
- 6. **subModels**: propiedades particulares de las partículas
 - a) **particleForces**: se definen las fuerzas que actuaran en la partícula.
 - b) **injectionModels**: Se definen las condiciones de inyección de partículas.
 - 1) **type**: tipo de inyección de partículas
 - 2) **parcelBasisType**: opciones de como enumeración de las partículas ya sea un valor fijo, mediante cantidad o masa.
 - 3) **patchName**: nombre del parche desde donde se inyectaran las partículas.
 - 4) **U0** Campo de velocidad con que las partículas serán inyectadas en m/s
 - 5) **parcelsPerSecond** numero de partículas inyectadas por segundo.
 - 6) **meanParticleDiameter**: Denomina el diámetro promedio de las partículas.
 - 7) **nParticle**: Numero de partículas inyectadas.
 - 8) **sizeDistribution**: Sirve para establecer la distribución de tamaños por partículas, pueden ser valores fijos o una distribucion normal.
 - 9) **flowRateProfile**: Distribución de las partículas inyectadas respecto a la duración, 1 significa uniforme.
 - 10) **massTotal**: masa total de partículas inyectadas.
 - 11) **SOI**: tiempo en el que parte la inyección de partículas.
 - 12) **duration**: Tiempo que dura la inyección de partículas.
- 7. **dispersionModel**: "none" debido a que no se implementan modelos de dispersión.
- 8. **patchInteractionModel**: localInteraction para interacciones parche-parche.
- 9. **localInteractionCoeffs** Define la interacción de las fronteras con las partículas. se nombran los parches y se le asignan las condiciones que se muestran a continuación:
 - a) **type**: representa el tipo de interacción que tendrá la partícula con el parche, para este caso rebound para rebote simple.
 - b) **e**: Corresponde al coeficiente de elasticidad que define el porcentaje de disipación en la dirección normal al parche.
 - c) **mu**: coeficiente de restitución que define el porcentaje de disipación en la dirección tangencial.

10. **heatTransferModel**: para modelos de transferencia de calor, se define none debido a que no será considerado.
11. **surfaceFilmModel**: el modelo de película de superficie en caso de goteo o interacción con la película, en este caso no se considera por lo tanto el valor es none.
12. **collisionModel**: Se utiliza para definir interacciones partícula-partícula. En este caso se desprecia este tipo de interacciones por lo tanto el valor es none.
13. **packingModel**: Modelo implícito para aplicar en una interacción inter-partícula, también se desprecia por lo tanto el valor es none.
14. **dampingModel**: Clase base para modelos de amortiguamiento de colisión. El seleccionado es Relaxation, en el que las partículas se relajan hacia la media local en una escala de tiempo.
15. **isotropyModel**: Es un término de colisión que hace que las distribuciones de velocidad se relajen a distribuciones gaussianas isotrópicas, se utiliza el modelo estocástico. donde las velocidades de las partículas se modifican muestreando una distribución gaussiana más delta, que depende de una escala de tiempo. Esto aleatoriza algunas velocidades de partículas mientras deja otras sin cambios. Cuanto menor sea el valor de la escala de tiempo, mayor será la proporción de velocidades de partículas afectadas.
16. **stochasticCollisionModel**: No se considera modelo estocástico para la colisión de partículas.
17. **radation**: En este caso no aplica considerar radiación.
18. **cloudFunctions**: Se utiliza para hacer seguimiento a partículas o calcular erosión.
 - a) **particleErosion**: Se ingresan los valores necesarios para calcular la erosión y la geometría a la que se quiere calcular la erosión.

3.6.3. Directorios de tiempo

3.6.3.1. U.water

Establece las condiciones del agua como su velocidad e interacción en cada uno de los parches, es decir corresponde a las condiciones de borde del agua.

3.6.3.2. p

Corresponde a las condiciones de borde en de presión en cada parche.

3.6.3.3. k.water

Condiciones de borde de la energía cinética turbulenta del agua en cada parche.

3.6.3.4. nut.water

Se definen las condiciones de borde para la viscosidad cinemática del agua.

3.6.3.5. epsilon.water

Se define la condición de borde de la tasa de disipación de energía cinética en cada parche.

3.7. Postprocesamiento

Una vez realizada la simulación se debe recabar la información y analizarla, para la validación se requiere la ubicación de las partículas al impactar el objetivo, la velocidad y el ángulo de impacto. Existe un método de post procesamiento en OpenFOAM para obtener todos los datos de las partículas en distinta posición y tiempo. Cuando Alejandro López [11] y su equipo realizaron el estudio el postprocesamiento se entregaban los resultados en coordenadas cartesianas, que eran las que se usaban para hacer el seguimiento de las partículas. Con el objetivo de mejorar el seguimiento de las partículas en Abril de 2017 OpenFOAM implementa el seguimiento con coordenadas baricéntricas [48], al hacerlo en el post procesamiento también se entregan estas coordenadas, sin embargo no entregan las coordenadas del tetraedro asociado para llevar esos datos al sistema cartesiano.

Debido a la situación expuesta anteriormente se opta por una opción alternativa, esta consiste que desde Paraview se pueden obtener las coordenadas y velocidades de las partículas en el tiempo representado.

Para obtener una buena representación de los datos se requiere guardar los datos cada 0,0001 segundos de intervalo de tiempo, generando una hoja de datos por cada paso de tiempo.

Para poder obtener los datos de manera mas rápida se crea un programa en python (Apéndice D) con el objetivo de extraer, filtrar y procesar la información de las miles de hojas de datos obtenidas. En detalle se consideran las partículas a 0.1 mm de distancia del objetivo mientras que el resto queda descartada, se guardan y almacenan las coordenadas para posteriormente representarlas gráficamente. Se clasifican en grupos las partículas según su distancia al centro del objetivo cada 0,5 mm y se promedian las velocidades y ángulos de cada grupo. Los ángulos se calculan vectorialmente con las velocidades en cada dirección.

4 | Validación y Verificación

Con el avance de la tecnología sumado al desarrollo de nuevos software y métodos computacionales ha aumentado el uso de las simulaciones. El aumento de este recurso sucede debido a que entregan información bastante importante a un costo menor del que podría ser el experimental [49] [27].

A pesar de ser un recurso que presenta mayor flexibilidad que el trabajo experimental [50], es muy importante que los datos entregados sean acorde a la realidad. Es por ello que muchas entidades y reconocidas asociaciones como el American National Standard Institute (ANSI) [51], American Institute of Aeronautics and Astronautics (AIAA) [52] y American Society of Mechanical Engineers (ASME) [53], han establecido parámetros o guías para el procedimiento de verificación y validación (V&V).

Adoptando las definiciones establecidas por la ASME [54] sobre V&V:

1. **Verificación:** El proceso de determinar que un modelo computacional representa con precisión el modelo matemático subyacente y su solución.
2. **Validación:** El proceso de determinar el grado en que un modelo es una representación precisa del mundo real desde la perspectiva de los usos previstos del modelo.

La verificación y validación son dos procesos importantes en la simulación de fluidos computacional para garantizar que los resultados de la simulación sean precisos y confiables [55].

La verificación se refiere a la verificación de que el modelo matemático y la implementación numérica son correctos y precisos. Este proceso implica la comparación de los resultados de la simulación con soluciones analíticas o soluciones de referencia que se consideran exactas [32]. La verificación se puede realizar en diferentes niveles, desde la verificación de la discretización de las ecuaciones de flujo hasta la verificación de la implementación numérica del modelo de turbulencia [32].

La validación, por otro lado, se refiere a la comparación de los resultados de la simulación con los datos experimentales o las simulaciones previas. La validación es importante para garantizar que el modelo matemático y las condiciones de contorno utilizadas en la simulación sean precisas y representativas del problema físico real [56].

Para llevar a cabo la verificación y validación, se pueden utilizar diferentes métricas de error, como la raíz del error cuadrático medio (RMSE) y el coeficiente de correlación (R-squared). También se pueden realizar pruebas de sensibilidad para evaluar cómo varían los resultados de la simulación en función de las diferentes condiciones de entrada y parámetros de modelado [57].

Es importante tener en cuenta que la verificación y validación son procesos continuos que se deben realizar a lo largo de todo el ciclo de vida de la simulación, desde la planificación hasta la implementación y la interpretación de los resultados. La verificación y validación adecuadas garantizan la precisión y confiabilidad de los resultados de la simulación, lo que es esencial para la toma de decisiones informadas en la industria y la investigación científica [58].

La estrategia de V&V para este estudio consiste en comparar resultados con otro análisis de la literatura y adicionalmente compararlo con un procedimiento experimental de la literatura.

4.1. Caso científico utilizado para la validación

Este estudio consistió en la simulación del impacto de un chorro de agua en un objetivo mediante el uso de la herramienta de simulación numérica Ansys Fluent y OpenFOAM. La simulación se realizó con el objetivo de estudiar el fenómeno de erosión y desgaste en la superficie del objetivo debido al impacto del chorro de agua (Ver Figura 4.1), de la misma manera en que lo planteó Gnanavelu [21]. Se realizaron dos configuraciones experimentales, una con la boquilla de agua a 5 mm del objetivo y otra con la boquilla a 25 mm del objetivo. El objetivo utilizado en el experimento tenía una superficie plana y circular, y estaba hecho de acero inoxidable 316L [21]. El chorro de agua utilizado tenía una velocidad de entrada de 6 m/s y un diámetro de 10 mm [11].

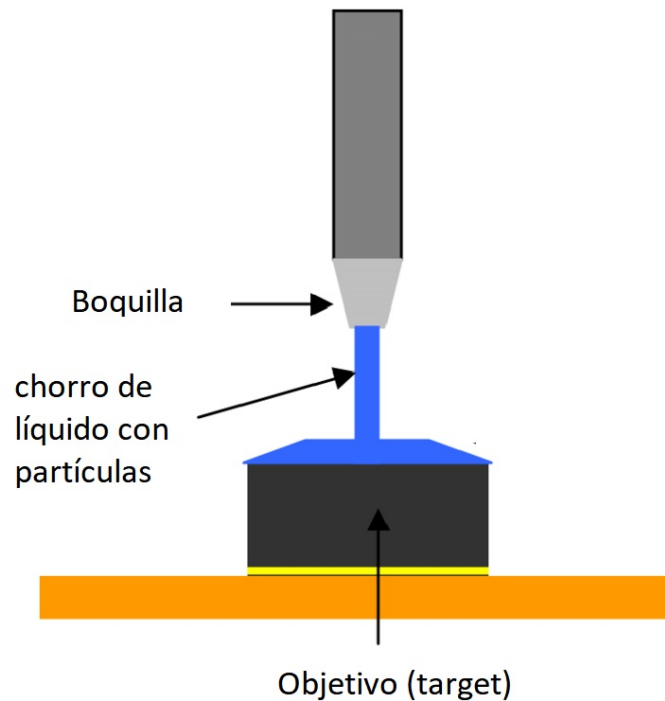


Figura 4.1: Configuración Jet Impingement Test.

(Fuente: Review of Two-phase Electronics Cooling for Army Vehicle Applications [59])

De este modo el objetivo de la validación es replicar lo realizado por López [11], sin embargo no se utiliza la misma versión de OpenFOAM debido a dos razones principales; la primera es que la versión estaba descontinuada para el sistema operativo actual de Ubuntu y no fue posible instalarlo junto a la versión antigua de Ubuntu, la segunda razón consiste en que en aquella versión no simulaba la interacción partículas-fluido, López tuvo que modificar el código de OpenFOAM para incluir el modelo de inyección de partículas Euler-Lagrange [11]. Este modelo permite simular la trayectoria de las partículas y su interacción con el fluido en un flujo multifase. No fue posible obtener acceso a la modificación realizada y para poder realizarla se requieren conocimientos avanzados de C++ lo que está fuera del estudio de esta memoria.

Para la simulación, se utilizó un modelo de turbulencia k-epsilon estándar y un modelo de seguimiento de partículas para representar el comportamiento de las partículas de agua en el chorro de la misma manera que lo hizo López [11]. La malla utilizada en la simulación consistió en una malla estructurada bidimensional, generada mediante el software Autodesk Inventor a diferencia de lo que realiza López que utilizó Gambit [11]. El tamaño de la celda en la dirección del chorro se redujo para capturar con precisión el comportamiento de las partículas en la zona de impacto.

Se define la mitad de la geometría con una condición de borde en simetryplane, para lograr reducir los costos computacionales. La entrada de agua y partículas ocurre en el parche inlet, mientras que top outlet

y bottom outlet son los parches cuyas condiciones de borde permiten la salida del agua y las partículas del dominio. Finalmente wall es el parche que representa los bordes sólidos de la boquilla e impide el paso de agua y partículas. Las condiciones de borde utilizadas se pueden ver con mayor claridad en la Figura 4.2

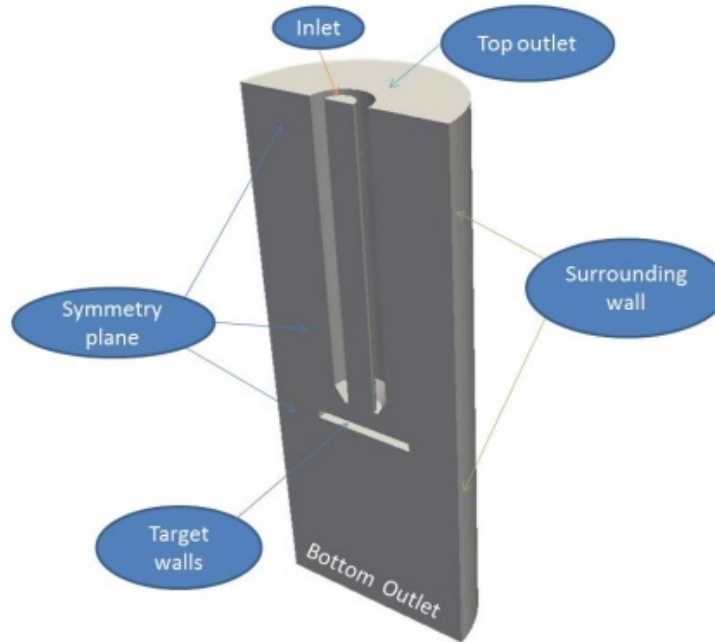


Figura 4.2: Condiciones de borde y Figura JIT.

(Fuente: CFD study of Jet Impingement Test erosion using Ansys Fluent and OpenFOAM [11])

Por otra parte los parámetros asignados a la simulación transiente se muestran en la Tabla 4.1

Tabla 4.1: Parámetros de simulación de López.[11]

Variable	Unidad	Valor
Paso de tiempo	s	1.8759e-05
Número de pasos de tiempo	-	53308
Material de la partícula	-	Carbón
Distribución de diámetro de partícula	-	Uniforme
Diámetro de la partícula	m	2.5e-04
Acoplamiento entre fases	-	Unidireccional
Fuerzas	-	Drag
Coefficiente de arrastre	-	Arrastre en esferas
Densidad de partículas	$\frac{Kg}{m^3}$	2206

Se considera una fracción del volumen de sólidos correspondiente al 1 % , así el flujo másico de las partículas fue $0,000962 \frac{kg}{s}$ lo que resulta en 53308 partículas por segundo. Además la cantidad de celdas fueron 984960 para el caso de la boquilla a 5 mm del objetivo y 670349 celdas para 25 mm de distancia entre la boquilla y el objetivo (target) [11].

Las partículas elegidas eran esferas con una distribución uniforme de radios de 250 μm , una densidad de 2206 Kg m³ en el caso de 5 mm y 2300 Kg m³ en el de 25 mm, junto con una velocidad inicial igual a la

velocidad de entrada del fluido (10 m/s). Un estudio exhaustivo de las trayectorias de partículas reveló que la modificación de la velocidad de entrada de partículas (poniéndola a 0 m/s, por ejemplo) no se percibió aguas abajo debido a la corta distancia que las partículas recorrieron para acelerar cerca de la velocidad del fluido y la comparativamente larga distancia a la superficie del objetivo [11].

Según Lopez [11] el modelo de inyección superficial en OpenFoam calcula el número de partículas a inyectar por paso de tiempo y, utilizando una función aleatoria, elige diferentes ubicaciones en la superficie seleccionada para las diferentes inyecciones. Diferencia importante que tenía con el software Fluent 15 y la versión 2.2 de OpenFOAM.

Para la versión 2.3 de OpenFOAM se implementó un cambio en el modelo de inyección de parches [60], en el que previamente las partículas se inyectaban en ubicaciones entre los centros de la cara límite y las celdas cercanas a la pared, como se muestra en la Figura 4.3.

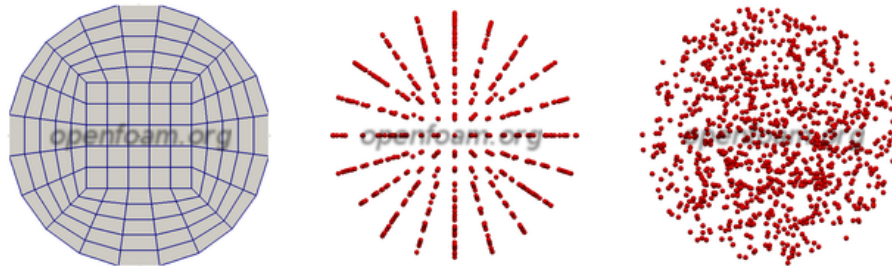


Figura 4.3: Modelos de inyección OpenFOAM (a la izquierda esta la malla, al centro el modelo antiguo a la derecha el modelo nuevo).

(Fuente: OpenFOAM.org [60])

En esta nueva versión, los modelos de inyección de parches distribuyen las partículas de manera aleatoria en las caras de los parches, reduciendo considerablemente la estructura ordenada en la distribución de partículas. Se menciona también que el objeto de función de nube ha sido revisado para permitir múltiples objetos del mismo tipo y ser consistente con la especificación principal de objetos de función en el archivo controlDict [60].

El modelo de inyección aleatoria implementado por OpenFOAM sumado a la función de dispersión estocástica del diccionario kinematicCloudProperties, permitieron que las ubicaciones de impacto en el objetivo (target) no carecieran de aleatoriedad [11].

Los primeros resultados analizados son las ubicaciones de los impactos de cada partícula tanto para distancias de la boquilla al objetivo (target) de 5 mm y 25 mm, como se muestra respectivamente en las Figuras 4.4 y 4.5.

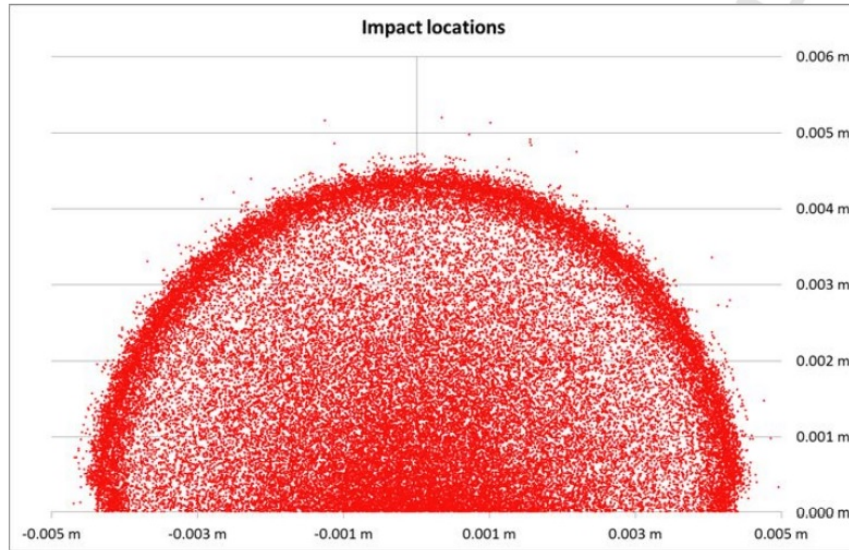


Figura 4.4: Localización del impacto a 5mm.

(Fuente: CFD study of Jet Impingement Test erosion using Ansys Fluent and OpenFOAM [11])

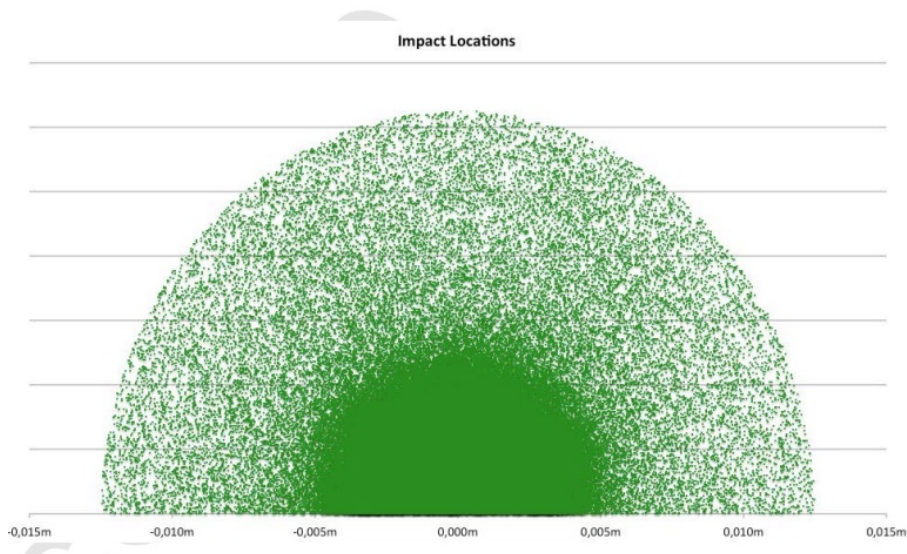


Figura 4.5: Localización del impacto a 25mm.

(Fuente: CFD study of Jet Impingement Test erosion using Ansys Fluent and OpenFOAM) [11]

López observó que la mayoría de los impactos estaban ubicados en la región interna de 0 a 4 mm del objetivo cuando la separación entre la boquilla y el objetivo era de 5 mm. Como el número de impactos en la región externa no fue lo suficientemente alto, los valores promedio en esas regiones no se consideraron debido a la baja densidad de impactos en comparación con la densidad de impactos en la región interna [11]. Este análisis solo lo realiza respecto a la configuración de 5 mm de distancia (Figura 4.4)

Se puede ver que en ambas situaciones hay una concentración de partículas en el centro del objetivo para la primera configuración (Figura 4.4) entre los 0 y 2 mm de radio, mientras que en la segunda (Figura 4.5) entre los 0 y 5 mm. A 5 mm de distancia se nota también una mayor concentración en la periferia comparado con 25 mm de distancia. Además, el área abarcada por la distancia de 25 mm tiene un radio mayor de impacto 12 mm comparado con el radio a los 5 mm de separación del objetivo con un valor de 4 mm, algo

que es esperable debido a que la velocidad de las partículas disminuye con la distancia, lo que significa que las partículas tienen más tiempo para dispersarse y difundirse antes de alcanzar el objetivo [61].

Una vez que se recopilaban todas las variables de impacto de más de 50.000 partículas, se postprocesaron para obtener su promedio en comparación con la distancia desde el centro del objetivo circular. Para hacer esto, el círculo de 25 mm de diámetro se dividió en regiones concéntricas más pequeñas separadas por 1 mm entre sí, como se representa en la Figura 4.6 [11].

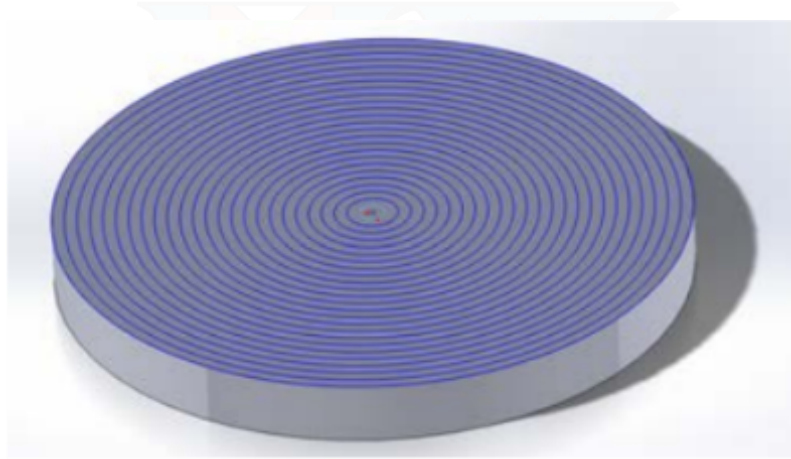


Figura 4.6: Cilindro de 25mm subdividido en regiones de 1 mm.

(Fuente: CFD study of Jet Impingement Test erosion using Ansys Fluent and OpenFOAM [11])

De esta forma primero se recopilaban los datos de la velocidad de impacto de las partículas y se muestra un promedio según la distancia radial anterior respecto al centro del objetivo de la forma que se explica en el párrafo anterior y cuyos resultados son representados en las Figuras 4.7 y 4.8 para 5 mm y 25 mm de distancia de la boquilla al objetivo respectivamente.

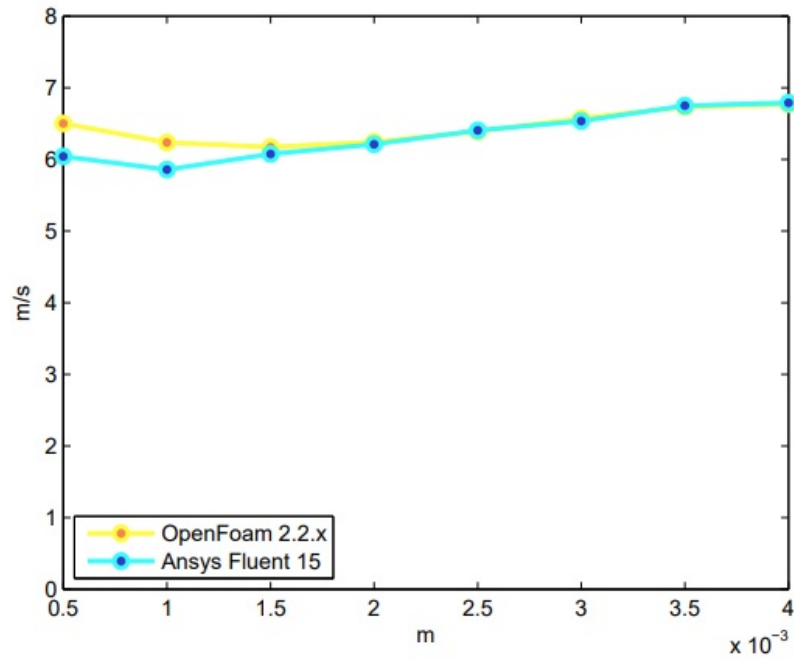


Figura 4.7: Velocidad de impacto a 5mm.

(Fuente: CFD study of Jet Impingement Test erosion using Ansys Fluent and OpenFOAM [11])

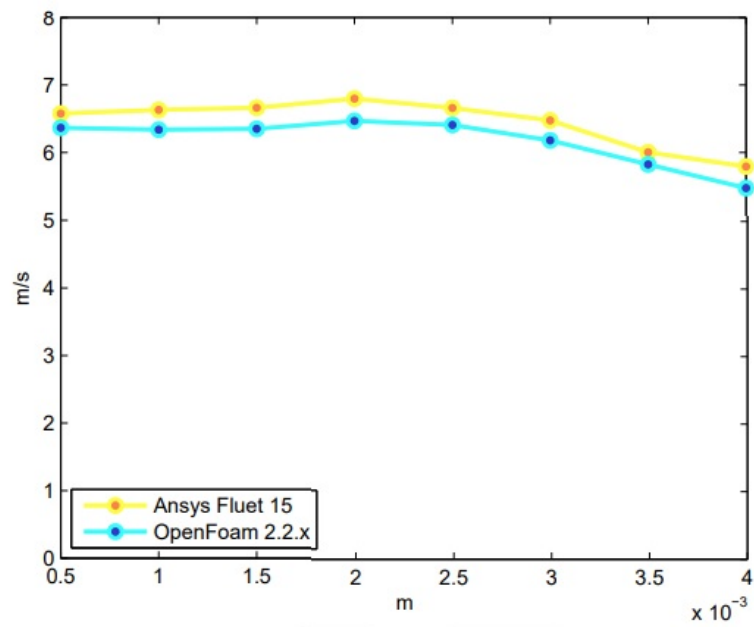


Figura 4.8: Velocidad de impacto a 25mm.

(Fuente: CFD study of Jet Impingement Test erosion using Ansys Fluent and OpenFOAM [11])

En su estudio López [11] observó que como se muestra en las Figuras 4.7 y 4.8, se pueden observar pequeñas diferencias entre los perfiles de la velocidad de las partículas a lo largo del radio para OpenFoam y Fluent 15, con una diferencia relativa media del 2 %. También destacó que, en el caso de 5 mm, las velocidades de impacto en el centro del objetivo tienen valores más bajos, siendo estos valores más altos cuanto más alejados del centro. Cuando la separación entre la boquilla y el objetivo es mayor, ocurre lo contrario, siendo las velocidades en el centro ligeramente más altas que más alejadas de él. Esto se lo adjudicó a que probablemente a la disparidad en las aceleraciones que tanto el fluido como las partículas experimentan en las dos brechas diferentes entre la salida de la boquilla y el objetivo.

La principal diferencia entre una configuración y otra, es que la velocidad a una distancia de 5 mm del objetivo aumenta a medida que las partículas se alejan del centro. A diferencia de la configuración a 25 mm de distancia que la velocidad primero aumenta levemente, para luego disminuir a medida que las partículas se alejan del centro.

Finalmente se obtiene el promedio de los ángulos de impactos respecto a su ubicación radial desde el centro, para los tramos señalados en la Figura 4.6, para ambas configuraciones y cuyos resultados se representan en las Figuras 4.9 y 4.10.

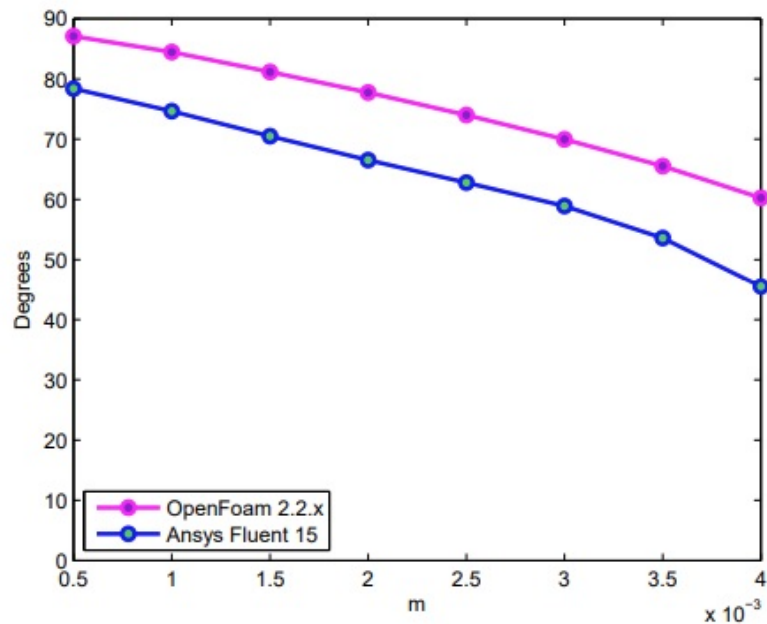


Figura 4.9: Ángulo de impacto a 5mm.

(Fuente: CFD study of Jet Impingement Test erosion using Ansys Fluent and OpenFOAM [11])

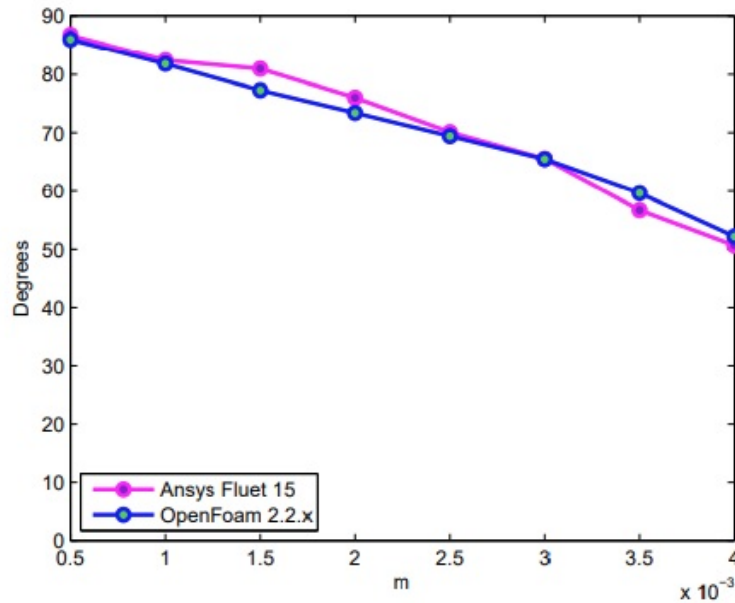


Figura 4.10: Ángulo del impacto a 25mm.

(Fuente: CFD study of Jet Impingement Test erosion using Ansys Fluent and OpenFOAM [11])

En su estudio López [11] obtiene una diferencia relativa promedio del 15,3 %. López señala que las pequeñas diferencias podrían atribuirse a los resultados ligeramente diferentes en estado estacionario, en los que las diferencias en la velocidad cerca de la pared podrían explicar las diferencias en los ángulos de impacto. Sin embargo, ambos softwares capturan la misma pendiente para el comportamiento del ángulo de impacto a lo largo del radio del objetivo. En este caso, la diferencia en la pendiente de las líneas formadas por los ángulos de impacto promedio es menor en comparación con la de las velocidades. Los resultados indican que esta pendiente es ligeramente más empinada para el JIT de 25 mm de distancia.

En ambos casos se puede ver que el ángulo de impacto pasa de ser de un valor cercano a los 90 grados en el centro a inclinarse y reducir el ángulo de impacto a medida que las partículas se alejan del centro. Esto se explica por la trayectoria que traen las partículas antes de impactarse, las del centro impactan con una velocidad principalmente en un eje en dirección perpendicular al objetivo (target), mientras que para alejarse del centro debe aumentar la componente de la velocidad perpendicular al objetivo (target) disminuyendo, en consecuencia, el ángulo de impacto.

El objetivo de la validación es replicar esta experiencia obteniendo los mismos resultados para luego poder aplicarlo a una situación similar, lo que en este caso sería la erosión en una pala de turbina Pelton.

La validación se realiza en OpenFOAM 8, por lo tanto, se debe considerar que han habido importantes actualizaciones en el software hasta el presente. No fue posible utilizar la misma versión debido a que está obsoleta.

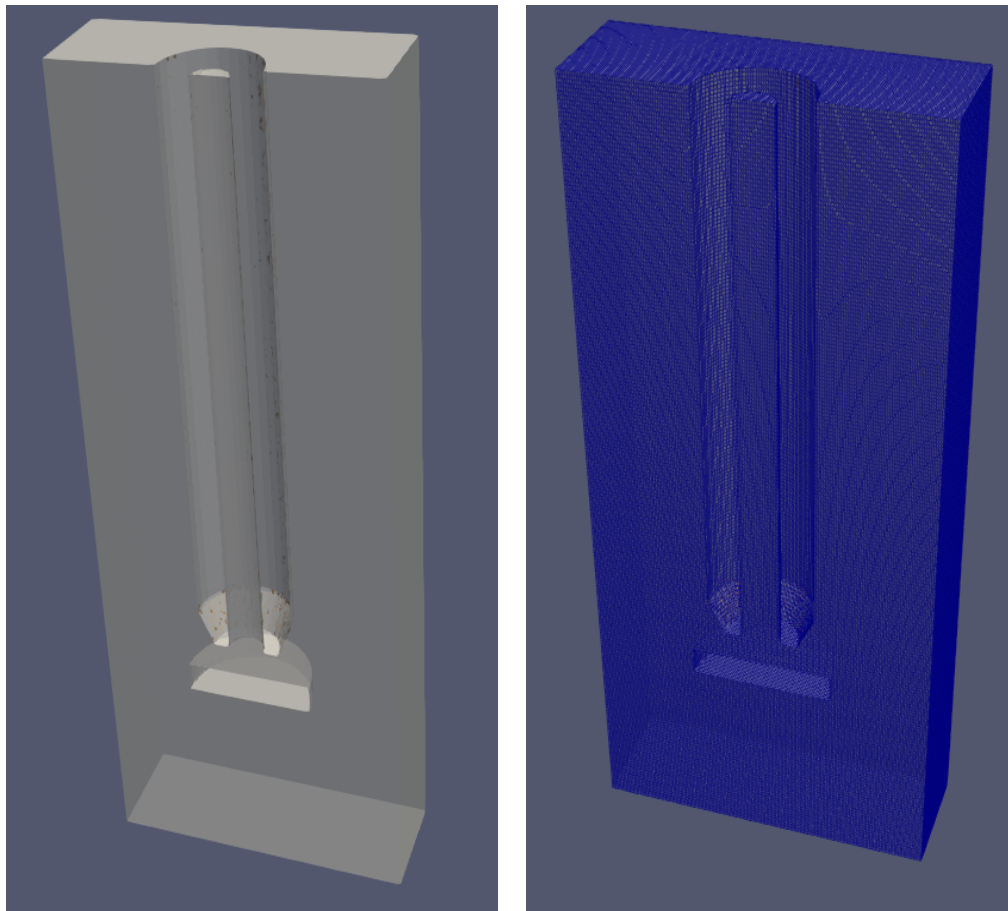
4.2. Validación

En este capítulo, se abordará el proceso de validación de los modelos y métodos previamente descritos en la memoria, enfocándose en la comparación con el caso de estudio presentado por Alejandro López [11]. A través de la comparación entre los resultados obtenidos mediante simulaciones y valores de referencia, se puede determinar si el modelo es adecuado para el análisis y predicción de la erosión hidráulica en diferentes contextos y condiciones.

Para llevar a cabo la validación, se analizará el caso de estudio propuesto por López [11], en el que se compararán los resultados generados por los modelos implementados con los datos de la simulación realizada por él. Se prestará especial atención a la localización de impacto, la velocidad de impacto y el ángulo de impacto. Estas comparaciones permitirán identificar las fortalezas y debilidades de los modelos, así como las áreas en las que podrían requerirse mejoras o ajustes.

4.2.1. Geometría y mallado

Como se mencionó en la sección 3.6.1.2 para realizar el mallado se requiere de un dominio generado por blockMesh y en ese bloque generado es donde snappyHexMesh extrae el volumen para dar forma a la geometría. En un principio se prueba con un dominio con forma de paralelepípedo como se ve en la Figura 4.11.



(a) Geometría en un dominio en base a un paralelepípedo

(b) Mallado de geometría en base a un paralelepípedo

Figura 4.11: Geometría cubica para distancia de 5mm.

Como se ve en la Figura 4.11b la malla requiere ser muy fina para lograr una mejor adaptación a la geometría, esto significa un alto consumo de recursos computacionales. Una forma de reducir considerablemente cantidad de celdas de la malla es optando por un dominio cilíndrico como se ve en la Figura 4.13.

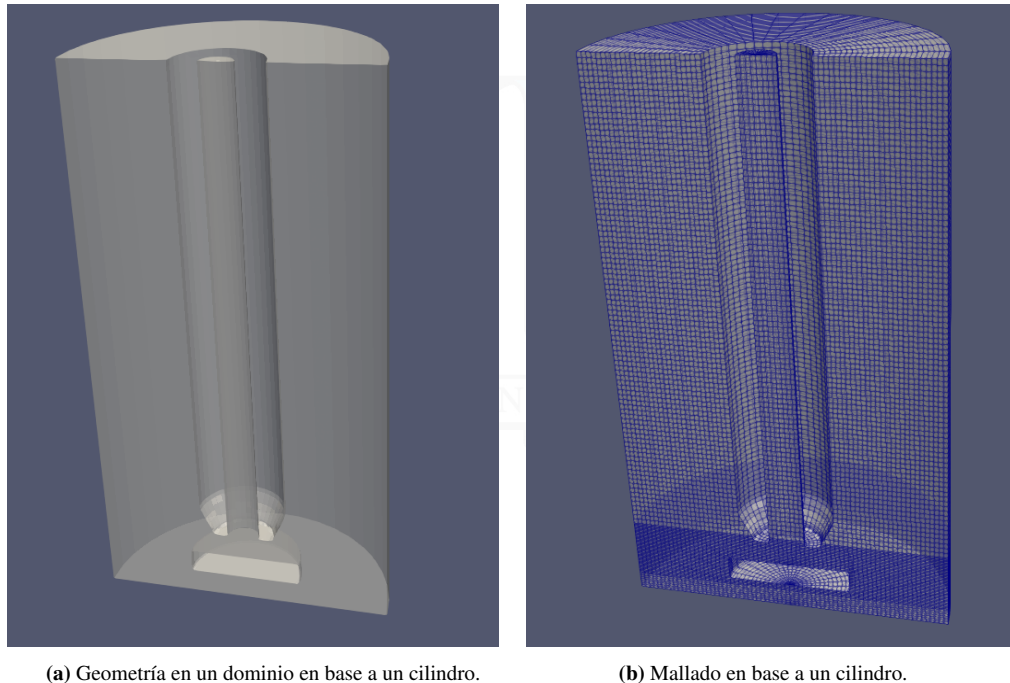


Figura 4.12: Geometría cilíndrica para una distancia de 5 mm.

Usar un cilindro como forma de base para generar una malla tiene varias ventajas. En primer lugar, la forma cilíndrica es ideal para flujos de fluidos que tienen una simetría cilíndrica, ya que permite una malla más uniforme y suave en la dirección axial del cilindro. Además, la generación de mallas a partir de un cilindro se realiza con mayor facilidad, ya que los algoritmos de malla cilíndrica son más simples y eficientes que los algoritmos de malla basados en un bloque paralelepípedo.

Por otro lado, el uso de un bloque paralelepípedo como forma de base puede ser una opción cuando se trabaja con geometrías más complejas o no cilíndricas. Sin embargo, esto requiere un mayor esfuerzo en la creación de mallas ya que los algoritmos son más complejos y puede ser necesario crear múltiples parches para adaptarse a la geometría.

El uso de un cilindro como forma de base para la generación de mallas ofrece varias ventajas, como una mejor adaptación a flujos con simetría cilíndrica y una mayor facilidad en la generación de la malla. Sin embargo, en casos de geometrías complejas no cilíndricas, el uso de un bloque paralelepípedo puede ser necesario, pero a costa de un mayor esfuerzo en la creación de la malla.

Un cilindro es una forma más natural para simular geometrías cilíndricas, como las boquillas de los fluidos o los objetivos en los que impactan los chorros. Al tener una sección transversal circular, el mallado generado a partir del cilindro puede adaptarse mejor a la forma de la geometría cilíndrica que se está simulando, lo que se traduce en una mayor precisión en la representación del flujo.

Además, el uso de un cilindro como base para la generación del mallado puede ser más eficiente computacionalmente que el uso de un bloque paralelepípedo. Esto se debe a que un cilindro tiene menos caras y vértices que un bloque paralelepípedo, lo que significa que se necesitan menos nodos en la malla para representar la misma geometría. Como resultado, se puede lograr una buena calidad de malla con menos nodos, lo que se traduce en un menor costo computacional y un tiempo de simulación más corto.

En resumen, el uso de un cilindro como base para la generación del mallado presenta varias ventajas

en la simulación de flujos, especialmente en la simulación de geometrías cilíndricas como las boquillas y objetivos. Además de ser más eficiente computacionalmente, el mallado generado a partir del cilindro puede adaptarse mejor a la geometría de la simulación, lo que se traduce en una mayor precisión en la representación del flujo.

La forma de las celdas toma una forma radial y se adapta perfectamente a la geometría con una cantidad de celdas considerablemente mas bajas y que se traduce a una mayor velocidad de simulación.

De la misma manera la geometría y el mallado de la boquilla a 25 mm de distancia se realiza con una malla cilíndrica, como se muestra en la figura (4.13b)

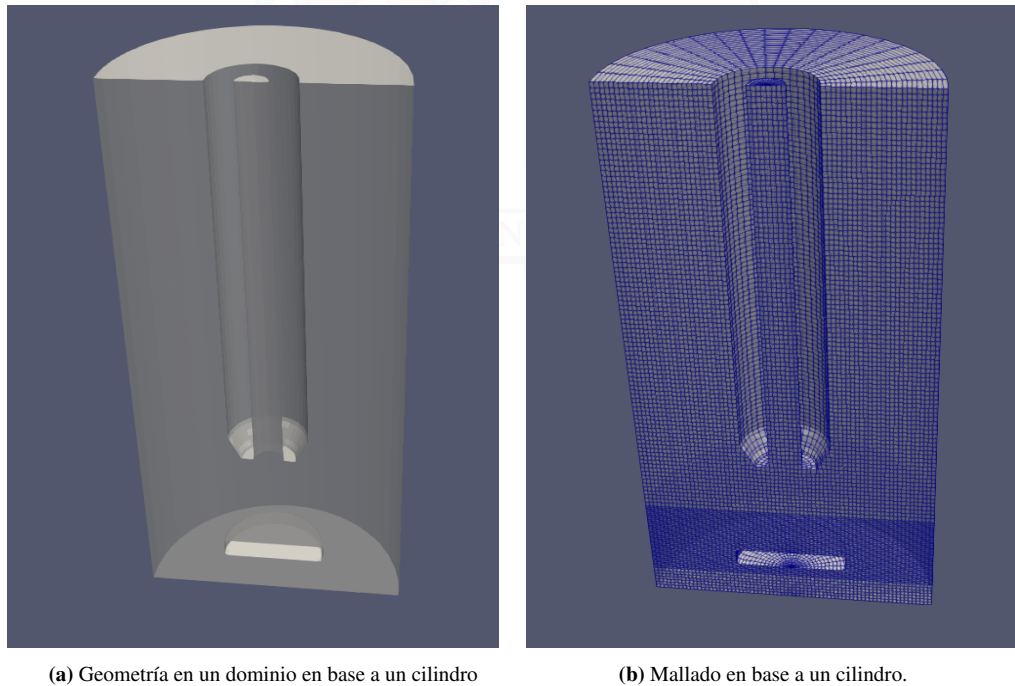


Figura 4.13: Geometría cilíndrica para distancia de 25 mm

En ambos casos se refino la malla cerca del objetivo porque es donde existen los mayores cambios de velocidad y presión, como también ocurre el impacto de las partículas en el objetivo.

4.2.2. Resultados

4.2.2.1. Localización de impacto

Para la obtención de los valores requeridos se sigue el procedimiento establecido en la sección 3.7. OpenFOAM entrega las coordenadas de la partícula en las tres dimensiones, por lo que simplemente se seleccionan las coordenadas correspondientes y se representan en un gráfico, una en función de otra según corresponda. De esta forma, los resultados de la validación para la inyección a 5 mm del objetivo para la localización del impacto de las partículas se encuentran en la Figura 4.14.

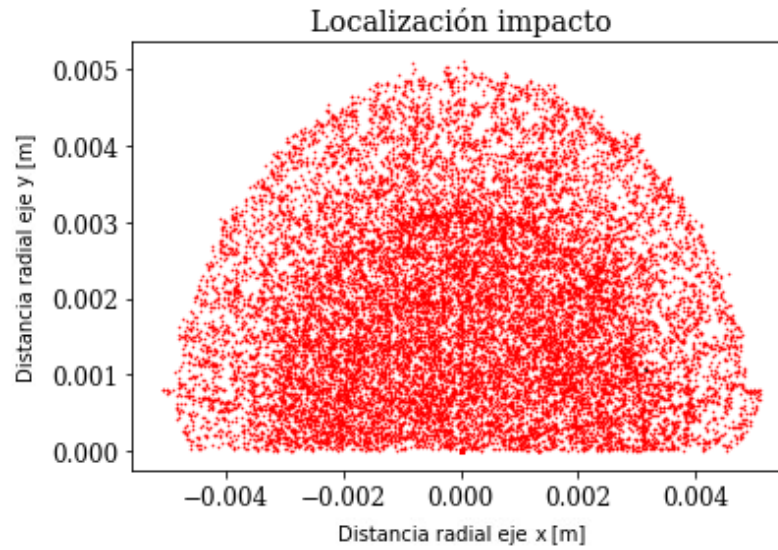


Figura 4.14: Localización de impacto a 5mm.

Los puntos de impacto se concentran mayormente en un radio de 3 mm desde el centro correspondiendo al 70,2 % del total de partículas. En la Tabla 4.2 se detalla la cantidad de partículas ubicadas según la ubicación radial del objetivo distribuidos por rango.

Tabla 4.2: Análisis de localización de impacto de partículas a 5 mm de distancia de la boquilla al objetivo

Distancia radial en mm	Cantidad de partículas	Cantidad acumulada de partículas	Porcentaje de partículas	Porcentaje acumulado de partículas	Concentración (partículas/mm ²)
0-0.5	732	732	3 %	1 %	932
0.5-1	2280	3013	10 %	6 %	968
1-1.5	4063	7076	19 %	13 %	1035
1.5-2	5033	12109	23 %	23 %	915
2-2.5	8002	20111	37 %	38 %	1132
2.5-3	8619	28730	40 %	54 %	998
3-3.5	8616	37346	40 %	70 %	844
3.5-4	6165	43511	28 %	82 %	523
4-4.5	4850	48361	22 %	91 %	363
4.5-5	4947	53308	23 %	100 %	332

Los resultados de la validación para la inyección a 25 mm del objetivo para la localización del impacto de las partículas están representados en la Figura 4.15.

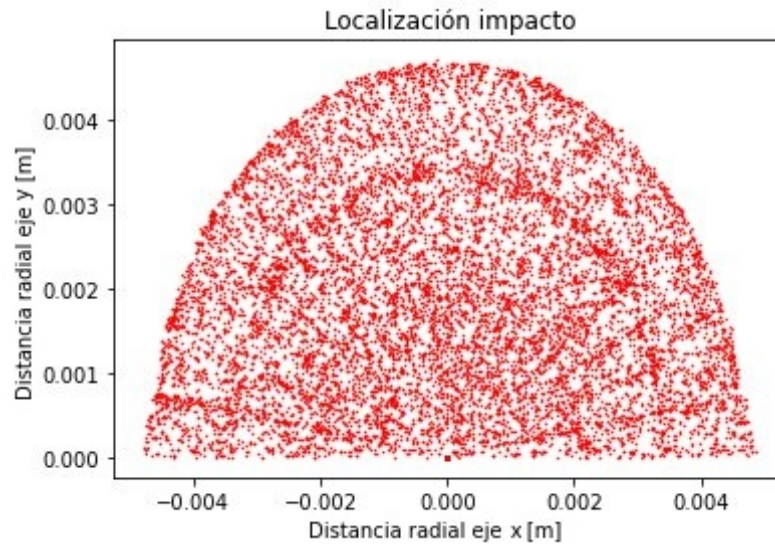


Figura 4.15: Localización de impacto a 25mm.

Y de una forma más detallada el análisis de los resultados se puede ver en la Tabla 4.3.

Tabla 4.3: Análisis de localización de impacto de partículas a 25 mm de distancia de la boquilla al objetivo

Distancia radial en mm	Total partículas	Acumulado	% del total	% acumulado	Concentración (partículas/mm ²)
0,0 – 0,5	622	622	1 %	1 %	792
0,5 – 1,0	2000	2622	3 %	5 %	849
1,0 – 1,5	3512	6134	6 %	12 %	894
1,5 – 2,0	5040	11174	8 %	21 %	917
2,0-2,5	6476	17650	11 %	33 %	916
2,5-3,0	6420	24070	11 %	45 %	743
3,0-3,5	9149	33219	15 %	62 %	896
3,5-4,0	7601	40819	13 %	77 %	645
4,0-4,5	9636	50456	16 %	95 %	722
4,5-5,0	2852	53308	5 %	100 %	191

Basándose en las tablas proporcionadas (Tablas 4.2 y 4.3), se pueden obtener las siguientes conclusiones sobre la localización del impacto de las partículas en el objetivo circular a diferentes distancias (5 mm y 25 mm) en un Jet Impingement test para estudiar la erosión:

Distribución de partículas: En las Tablas 4.2 y 4.3 obtenidas a partir de la simulación realizada, se puede observar que la mayoría de las partículas se concentran a una distancia radial del centro del objetivo de entre 1 mm a 4 mm en la posición radial del objetivo. En la Tabla 4.2 (5 mm de distancia), la concentración más alta de partículas (40 %) se encuentra en un rango de distancia radial del objetivo de 2.5-3 mm, mientras que en la Tabla 4.3 (25 mm de distancia), el rango de 3.0-3.5 mm tiene la mayor concentración (15 %).

Porcentaje acumulado de partículas: A medida que las partículas se alejan del centro del objetivo, el porcentaje acumulado de partículas aumenta en ambos casos. En la Tabla 4.2, se alcanza el 100 % en el rango de 4.5-5 mm, mientras que en la Tabla 4.3, se alcanza el 100 % en el mismo rango.

Cambio en la concentración de partículas: Comparando las Tablas 4.2 y 4.3, se puede observar que la concentración de partículas (partículas/mm²) disminuye a medida que aumenta la distancia entre la boquilla y el objetivo. Por ejemplo, en el rango de 0-0.5 mm, la concentración en la Tabla 4.2 es de 932 partículas/mm², mientras que en la Tabla 4.3 es de 792 partículas/mm². Esto sugiere que las partículas se dispersan más a medida que se alejan de la fuente.

Erosión y desgaste: La distribución de partículas y su concentración en la posición radial del objetivo proporciona información valiosa sobre las áreas de erosión y desgaste en el objetivo. En este caso, parece que las áreas con mayor concentración de partículas están entre 1 mm y 4 mm desde el centro del objetivo en ambas distancias (5 mm y 25 mm).

Estas conclusiones ayudan a comprender cómo el impacto de las partículas en el objetivo varía en función de la distancia y cómo afecta la erosión y el desgaste en diferentes áreas del objetivo circular.

Para comparar los resultados obtenidos por López en términos de localización de impacto, esto solo se puede hacer de manera visual ya que en su estudio no presenta los datos de forma numérica ya que su enfoque principalmente estaba en obtener la velocidad y el ángulo de impacto que son los valores necesarios para en estudios posteriores calcular la erosión. Para una distancia de 5 mm al objetivo como se muestra en la Figura 4.4 existe una mayor concentración de partículas tanto en el centro como en los bordes, algo similar sucede a una distancia de 25 mm (Ver Figura 4.5) pero en este caso la mayor concentración solo ocurre en el centro, esto difiere de los resultados obtenidos en la simulación donde las mayores concentraciones no ocurren en el centro (Ver Tablas 4.2 y 4.3). Esto también difiere con el estudio combinado numérico y experimental realizado por Nguyen [62] donde concluye que la trayectoria de las partículas sigue en su mayoría la dirección del flujo de agua y que dado que la velocidad de las partículas depende del gradiente de presión en la región de alta presión estática, la trayectoria de las partículas se dirige hacia la dirección exterior; por lo tanto, la cantidad de partículas de arena que golpean en la parte central de la superficie de la muestra de prueba se reduce, lo que resulta en que más partículas golpean la región próxima a la parte central. Es posible que el estudio de López en términos de localización de impacto se haya visto afectado por el comportamiento de las partículas ilustrado en la Figura 4.3 donde el orden de distribución de las partículas era basado en la estructura de la malla. Respecto a la localización de impacto López [11] solo comenta que a la distancia de la boquilla de 5 mm sólo utilizará las partículas en la región entre 0 y 4 mm (Donde esta la mayor concentración) debido a que el impacto en la región externa no es suficientemente alto.

4.3. Resultados de velocidad de impacto

En esta sección, se presentan los resultados obtenidos para las velocidades de impacto de las partículas en el objetivo en el Jet Impingement Test. La velocidad de impacto juega un papel importante en la erosión y el desgaste del material, y su análisis permite comprender cómo varía la erosión en diferentes condiciones de operación y configuraciones.

A continuación, en la Figura 4.16 se muestran las velocidades de impacto promedio para cada distancia radial al centro en el objetivo:

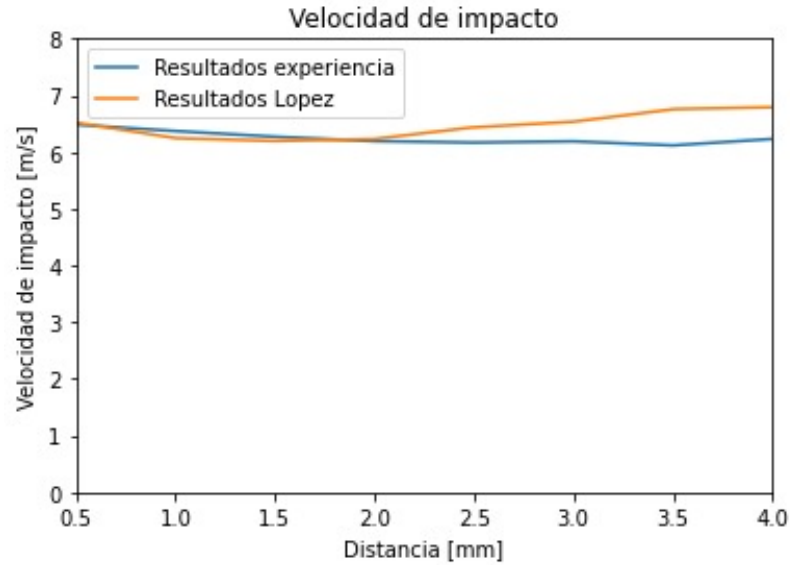


Figura 4.16: Velocidad de impacto a 5mm.

La velocidad de impacto a 25 mm de distancia obtenida, en comparación con la simulación de López se puede ver en la Figura 4.17

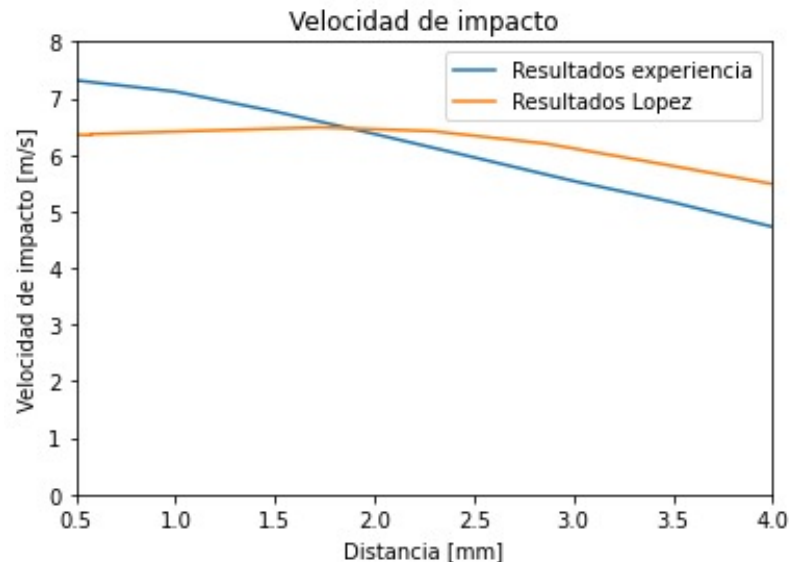


Figura 4.17: Velocidad de impacto a 25mm.

A 5 mm de distancia, las velocidades de impacto tienden a disminuir a medida que las partículas se alejan del centro del objetivo. Esto sugiere que las partículas pierden velocidad a medida que se dispersan radialmente desde el punto de estancamiento (centro del objetivo). La erosión podría ser mayor cerca del punto de estancamiento debido a las mayores velocidades de impacto y disminuir hacia la periferia del objetivo. La diferencia porcentual relativa con los resultados de López corresponde a un 3,14 %,

A 25 mm de distancia, las velocidades de impacto también disminuyen en general a medida que las partículas se alejan del centro del objetivo, siguiendo una tendencia similar a la observada a 5 mm de distancia. Esto sugiere que el patrón de erosión podría ser similar en ambas distancias, aunque las magnitudes

de las velocidades de impacto pueden variar. La diferencia porcentual relativa con los resultados de López corresponde a un 1,51 %

Comparando las velocidades de impacto a 5 mm y 25 mm de distancia, se puede observar que las velocidades de impacto son en general más altas a 25 mm que a 5 mm. Esto indica que, a medida que aumenta la distancia al objetivo, las partículas impactan con mayor velocidad, lo que podría resultar en una erosión más severa en áreas cercanas al punto de estancamiento a mayor distancia del objetivo.

Es importante tener en cuenta que las velocidades de impacto en sí mismas no son el único factor que determina la erosión. También hay que considerar los ángulos de impacto, la concentración de partículas y las propiedades de las partículas y del material del objetivo. Sin embargo, las velocidades de impacto proporcionan información valiosa sobre cómo se distribuye la energía de las partículas en el objetivo y cómo podría variar la erosión en función de la distancia radial al punto de estancamiento.

4.3.0.1. Ángulos de Impacto

Es importante señalar que cuando se habla de ángulo de impacto se hace referencia al ángulo que forma la línea de trayectoria de la partícula con el objetivo al momento del impacto como se muestra en la Figura 2.4 de la Sección 2.2

Los ángulos de impactos de cada partícula se obtienen mediante el complemento de la pendiente respecto al objetivo del vector director correspondiente a la velocidad de la partícula representado por la Ecuación 4.1 y cuyo modulo se obtiene con la Ecuación 4.2.

$$\theta = \arccos\left(\frac{x}{\|\mathbf{u}\|}\right) \quad (4.1)$$

$$\|\mathbf{u}\| = \sqrt{x^2 + y^2 + z^2} \quad (4.2)$$

Como se tienen los componentes de velocidad de cada los resultados obtenidos para una distancia de 5 mm se muestran en la Figura 4.18.

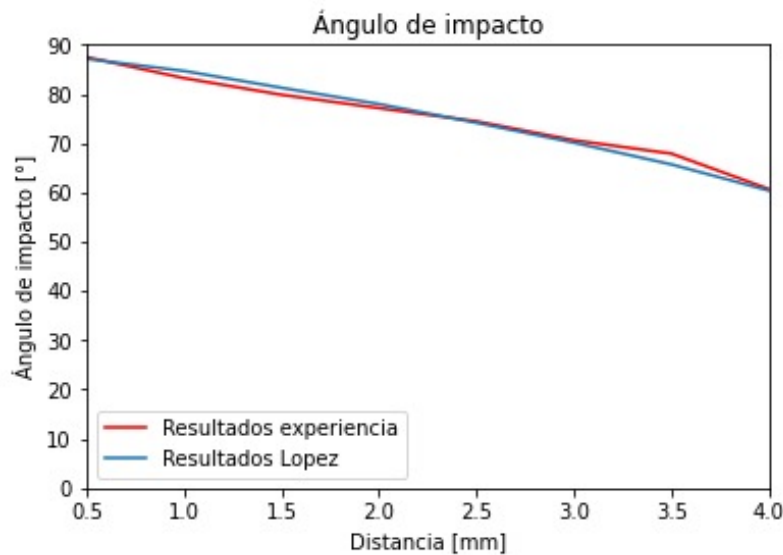


Figura 4.18: Ángulo del impacto a 5mm.

De la misma manera los resultados para la distancia de 25 mm se destacan en la Figura 4.19

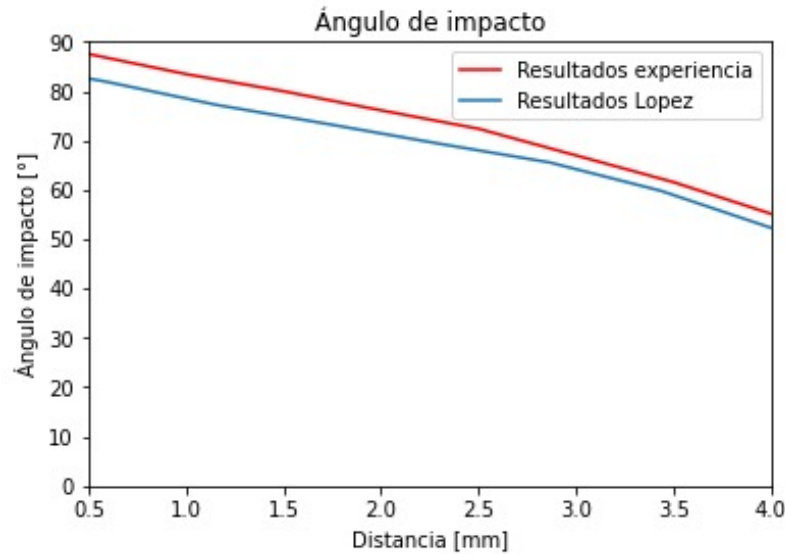


Figura 4.19: Ángulo del impacto a 25mm.

A 5 mm de distancia, los ángulos de impacto disminuyen en general a medida que las partículas se alejan del centro del objetivo. Esto indica que las partículas impactan en ángulos más inclinados lejos del punto de estancamiento (centro del objetivo), lo que puede afectar la erosión en esas áreas. La erosión podría ser menor en áreas donde los ángulos de impacto son más inclinados en comparación con áreas cercanas al punto de estancamiento, donde los ángulos de impacto son más cercanos a la perpendicular. La diferencia porcentual relativa con los resultados de López corresponde a un 0,15 %

A 25 mm de distancia, los ángulos de impacto también disminuyen en general a medida que las partículas se alejan del centro del objetivo, siguiendo una tendencia similar a la observada a 5 mm de distancia. Esto sugiere que el patrón de erosión podría ser similar en ambas distancias, aunque las magnitudes de los ángulos de impacto pueden variar. La diferencia porcentual relativa con los resultados de López corresponde a un 3,25 %

Comparando los ángulos de impacto a 5 mm y 25 mm de distancia, se puede observar que los ángulos de impacto son en general mayores a 25 mm que a 5 mm. Esto sugiere que, a medida que aumenta la distancia al objetivo, las partículas impactan en ángulos más cercanos a la perpendicular, lo que podría resultar en una erosión más severa en áreas cercanas al punto de estancamiento a mayor distancia del objetivo.

4.4. Análisis de Validación

Las tendencias generales en los datos de velocidad y ángulo de impacto son similares entre los resultados obtenidos en este trabajo y los de López. Esto sugiere que ambos conjuntos de datos reflejan patrones consistentes en la erosión en el Jet Impingement Test.

Hay discrepancias numéricas entre los resultados de este trabajo y los de López, lo que podría deberse a diferencias en las versiones de OpenFOAM utilizadas (2.x frente a 8) y las mejoras en el seguimiento de partículas y modelos de inyección en las versiones más recientes. La actualización implementada en 2017, que introdujo un seguimiento baricéntrico en OpenFOAM-dev, mejoró significativamente el algoritmo de seguimiento, permitiendo un seguimiento más preciso y robusto en mallas de mala calidad y facilitando la simulación de partículas en casos con geometría rotativa, entre otras mejoras. Esto puede contribuir a las discrepancias entre los resultados obtenidos en OpenFOAM 2.x y 8.

A pesar de las discrepancias numéricas, las tendencias generales similares indican que OpenFOAM

8 sigue siendo una herramienta válida y precisa para estudiar la erosión en el Jet Impingement Test. La actualización del seguimiento baricéntrico en OpenFOAM 8 podría haber contribuido a una mayor precisión y robustez en la simulación en comparación con OpenFOAM 2.x.



5 | Simulación de Erosión en Pala de Turbina Pelton

En la siguiente fase del estudio, se busca estimar la erosión en la pala de una turbina Pelton mediante una simulación computacional en OpenFOAM. Sin embargo, para garantizar que los resultados sean precisos y se ajusten a la realidad, es fundamental compararlos con un caso experimental previamente establecido. De esta forma, se podrá validar la precisión y fiabilidad de las simulaciones computacionales.

Primero, se describirá el modelo de turbina Pelton que se utilizará en la simulación y las condiciones de funcionamiento bajo las cuales se llevará a cabo el análisis. Luego, se presentarán los resultados de la simulación computacional y se analizará la distribución de la erosión en diferentes componentes de la turbina.

Una vez obtenidos los resultados de la simulación, se procederá a compararlos con los datos experimentales disponibles en la literatura. Esta comparación permitirá evaluar el grado de concordancia entre la simulación computacional y el caso experimental, proporcionando una medida de la validez del modelo y enfoque.

En base a los resultados de esta validación, se discutirán las implicaciones del estudio en el diseño y mantenimiento de turbinas Pelton, así como las posibles mejoras y ajustes necesarios en el modelo de simulación para lograr una mayor precisión en la predicción de la erosión.

5.1. Resultados experimentales y Resultados Numéricos

En este estudio, se utilizará como parámetro experimental para la comparación con la simulación el artículo "Effect of concentration and size of sediments on hydro-abrasive erosion of Pelton turbine" de Anant Kumar Rai, Arun Kumar y Thomas Staubli[1]. El objetivo de dicho trabajo es analizar el comportamiento de la erosión en diferentes materiales bajo las mismas condiciones erosivas e hidráulicas en turbinas Pelton ubicadas en regiones geológicamente jóvenes como el Himalaya.

El primer paso en este análisis consiste en describir el modelo de la turbina Pelton utilizada en el estudio experimental. Para ello, se realizaron experimentos con diferentes rangos de velocidad, duración de exposición, tamaño de sedimentos y concentraciones en palas de turbina Pelton a escala 1:8, replicando las condiciones de una planta hidroeléctrica (HPP) ubicada en el Himalaya. Se emplearon seis materiales distintos en la fabricación de las palas de la turbina, incluyendo tres tipos de acero, dos tipos de recubrimientos y bronce, para alturas de hasta 200 m. El montaje de la experiencia realizada se puede ver en la Figura 5.1 mientras que los detalles de los parámetros utilizados en el estudio se presentan en la Tabla 5.1.

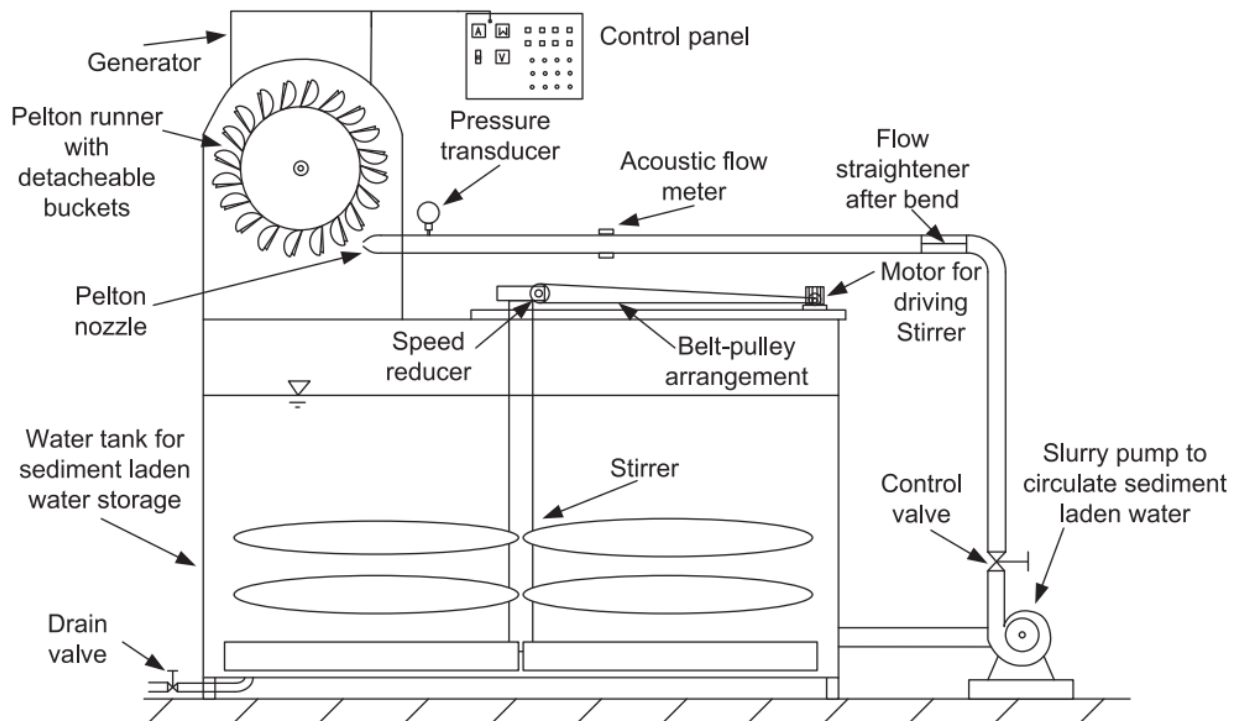


Figura 5.1: Configuración Experimento.

(Fuente: Effect of concentration and size of sediments on hydro-abrasive erosion of Pelton turbine [1])

Tabla 5.1: Detalles de los parámetros utilizados en el estudio.

Parámetros	Número y rango de valores
Velocidad del chorro	4 valores - (51.36, 54.91, 58.24 y 61.39) m/s
Velocidad del flujo	4 valores - (22.05, 25.62, 28.97 y 32.11) m/s
Concentración del sedimento	3 valores - (750, 1500 y 3000) ppm
Distribución de tamaño de las partículas	4 valores - (0-90), (90-180), (180-250) y (250-355) μm
Tiempo de duración	4 valores - (3, 6, 9 y 12) Hrs
Material	Bronce, 3 tipos de acero martensítico y 2 tipos de revestimiento
Diámetro primitivo de la pala	1 valor - 192 mm
Diámetro de la boquilla	1 valor - 12 mm

Considerando los valores que se mantuvieron constantes en para los distintos análisis en el estudio y que facilitan la comparación se seleccionan los valores de la Tabla 5.2.

Tabla 5.2: Detalles de los parámetros seleccionados para la simulación.

Parámetros	Número y rango de valores
Velocidad del chorro	61.39 m/s
Velocidad del flujo	32.11 m/s
Concentración del sedimento	1500 ppm
Distribución de tamaño de las partículas	(90-180) μm
Tiempo de duración	4 valores - 3 h
Material	acero martensítico
Diámetro primitivo de la pala	1 valor - 192 mm
Diámetro de la boquilla	1 valor - 12 mm

5.2. Consideraciones

En esta etapa del estudio, se realiza una simulación computacional en OpenFOAM para estimar la erosión en las palas de la turbina Pelton, considerando las condiciones iniciales de la validación. La simulación emplea un método de descripción Euler-Lagrange para modelar las interacciones entre el agua y las partículas, y sólo se tiene en cuenta la fuerza de arrastre del agua sobre las partículas (one-way coupling).

La configuración de la simulación en este caso se realiza como se muestra en la Figura 5.2 donde se ve la boquilla y la cuchara en una vista de perfil. La geometría del modelo incluye un parche por el cual ingresan el agua y las partículas, y se considera que las partículas desaparecen al impactar con la pala o cuchara de la turbina Pelton. Debido a la complejidad y los recursos computacionales necesarios, no se considera el movimiento de la pala en la simulación, por lo que se mantiene estática y se posiciona de tal manera que pueda emular de mejor forma el impacto de las partículas en la pala.

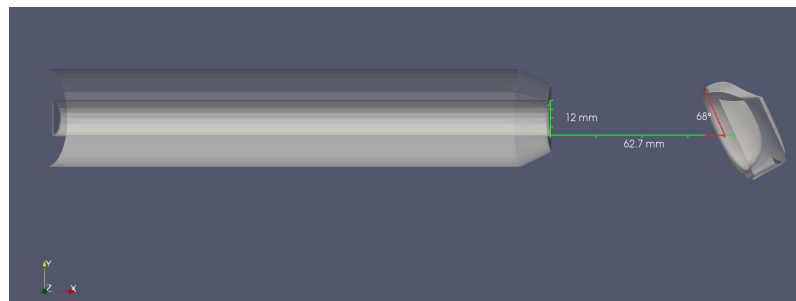
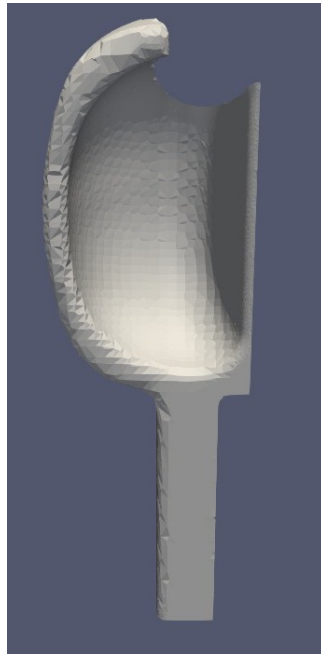


Figura 5.2: Configuración de la simulación.

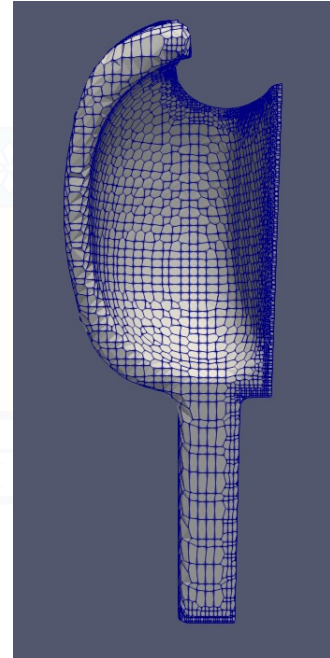
Para mejorar el rendimiento computacional, se considera solo la mitad de la pala en la simulación, aprovechando su simetría y asumiendo que los resultados serán simétricos (ver Figura 5.3a y 5.3c).

Para lograr una mayor precisión en la simulación, se utiliza un mallado más refinado en los alrededores de la pala y en la zona previa al impacto. Las Figuras 5.3b y 5.3d muestran la geometría de la pala con la malla aplicada. Aunque en la validación se empleó una malla cilíndrica por su eficiencia computacional, en este caso se opta por un mallado cúbico o cuadrado debido a que la forma de la pala de la turbina no se adapta bien al mallado cilíndrico. La diferencia de tipos de malla en la validación no debería afectar mayormente el procedimiento, lo más importante es asegurarse de que ambas mallas (la utilizada en la

simulación y la utilizada en la validación) sean de buena calidad y se ajusten adecuadamente a la geometría y las características del flujo.



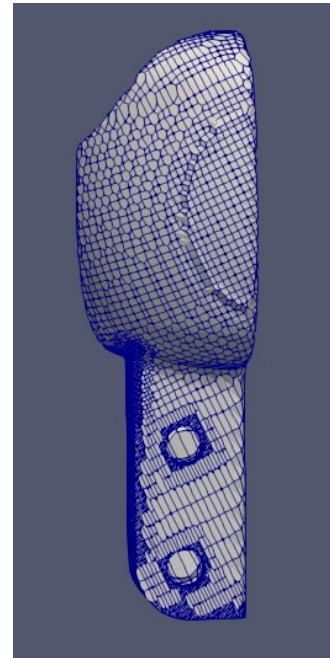
(a) Geometría de la mitad de la pala.



(b) Malla de la geometría de la pala.



(c) Geometría del perfil de la mitad de la pala.



(d) Malla del perfil de la geometría de la pala.

Figura 5.3: Geometría y malla de la pala de la turbina Pelton.

Las ecuaciones y parámetros numéricos utilizados en la simulación son los mismos que en la validación, y se encuentran en el marco teórico del estudio. Al implementar estos aspectos en la simulación computacional en OpenFOAM, se espera obtener resultados que permitan analizar y comparar la erosión en

las palas de la turbina Pelton, en función de las condiciones y parámetros del estudio experimental.

En el desarrollo de la simulación, se decidió no incluir ciertos aspectos, como el efecto de turbulencia en la interacción entre las partículas y el flujo, o el efecto de rebote de las partículas al impactar con la pala, debido a su complejidad técnica y a los recursos computacionales requeridos. La exclusión de estos factores se realizó para mantener un enfoque más simplificado y reducir la demanda de recursos computacionales, aunque podría tener un impacto en la precisión de la simulación.

Es importante mencionar que, debido a un bug conocido en OpenFOAM [63] [64], algunas partículas pueden quedar pegadas a la geometría de la pala a pesar de que se les asignó la condición de desaparecer al impactar. Este comportamiento inesperado será discutido y considerado en la sección de resultados al analizar la erosión en las palas de la turbina Pelton.

Además, debido al tiempo y recursos computacionales necesarios para realizar la simulación, se decidió simular solo 450 segundos de exposición a la erosión. A pesar de esta limitación, se pudo observar en los resultados preliminares que la erosión normalizada es lineal en función del tiempo, lo cual es consistente con los resultados obtenidos en el caso experimental de Kumar [1]. Por lo tanto, se optó por obtener una ecuación lineal a partir de los datos de erosión obtenidos en los primeros 450 segundos y proyectar los resultados para compararlos con los casos experimentales de 3, 6, 9 y 12 horas de exposición.

5.2.1. Selección modelo de erosión de Finnie

El modelo de Finnie es un enfoque clásico y ampliamente utilizado para predecir la erosión en turbina Pelton y otros materiales sometidos a partículas abrasivas en alta velocidad. La selección de este modelo para el trabajo de tesis presenta varias ventajas en comparación con otros modelos, como los de Nandakumar y Grant y Tabakoff [18] [15] [16].

En primer lugar, el modelo de Finnie es más simple y fácil de implementar que los otros dos modelos mencionados. La simplicidad del modelo permite un enfoque más directo en la evaluación de la erosión y facilita su implementación en simulaciones numéricas y análisis de datos [18]. Además, debido a su larga historia y uso generalizado en la literatura, el modelo de Finnie proporciona una base sólida para comparar resultados con estudios previos y casos reales, lo que facilita la validación y discusión de los resultados obtenidos [18].

Por otro lado, el modelo de Nandakumar es más complejo y requiere considerar varios parámetros adicionales relacionados con la interacción entre las partículas y el material, lo que puede complicar su implementación y análisis. Aunque este modelo puede ofrecer una descripción más detallada de la erosión y la interacción entre las partículas y el material [18], también puede resultar en un mayor tiempo de cómputo y esfuerzo de modelado. Por lo tanto, si el objetivo de este trabajo es obtener una comprensión general de la erosión y cómo afecta a las palas de la turbina Pelton, el modelo de Finnie es el más adecuado.

En cuanto al modelo de Grant y Tabakoff, también es más complejo que el modelo de Finnie, ya que considera el proceso de erosión como una combinación de dos mecanismos: erosión por corte y erosión por deformación plástica [17]. Aunque este enfoque puede proporcionar una descripción más precisa de la erosión en ciertos casos, también implica una mayor dificultad en su implementación y análisis en comparación con el modelo de Finnie [18]. Además, este modelo ha sido aplicado principalmente en el contexto de la erosión en componentes de turbinas de gas, lo que podría limitar su aplicabilidad y comparabilidad con otros estudios en el ámbito de las turbinas hidráulicas como la Pelton [17].

En un estudio realizado por López en 2015 [18], se compararon los tres modelos de erosión mencionados previamente: el modelo de Finnie, el modelo de Nandakumar y el modelo de Grant y Tabakoff. Los resultados del estudio mostraron que los tres modelos predecían la ubicación de la erosión máxima en la misma posición. Sin embargo, existían discrepancias al cuantificar la erosión y la erosión relativa alrededor de la ubicación de la erosión máxima. Si bien los modelos de erosión de Finnie y Grant y Tabakoff arrojaron resultados similares para el canal de bombeo considerado, el modelo de Nandakumar predijo desgaste en lugares donde los otros dos no lo hicieron. Dado que los tres modelos proporcionaron resultados similares para la ubicación de la erosión máxima, López concluyó que era razonable elegir el modelo de Finnie como

un primer enfoque, debido a la falta de constantes experimentales en las fórmulas, lo que hace que no sea necesario realizar pruebas experimentales para obtenerlas [18].

5.3. Resultados

En esta sección, se presentan y analizan los resultados obtenidos de la simulación realizada en OpenFOAM. Los objetivos principales de esta sección son mostrar la distribución de la erosión en la cuchara de la turbina Pelton, calcular la erosión normalizada y comparar los resultados de la simulación con los datos experimentales obtenidos por Kumar [1]. Se discutirán las imágenes generadas en ParaView que ilustran la intensidad de la erosión, así como las fotografías del experimento de Kumar que muestran la progresión de la erosión y la eliminación del recubrimiento. Además, se presentarán los resultados de erosión normalizada en forma de tabla y gráfico, comparándolos con los datos experimentales de Kumar. Finalmente, se abordarán las limitaciones y desafíos de la simulación, incluido el bug en OpenFOAM que causa que algunas partículas se peguen a la cuchara y cómo se proyectaron los resultados a partir de los datos preliminares y experimentales.

5.3.1. Resultados de la simulación en OpenFOAM

5.3.1.1. Intensidad de la erosión en la cuchara de la turbina Pelton

La simulación en OpenFOAM proporcionó información valiosa sobre la distribución de la erosión en la cuchara de la turbina Pelton. La Figura 5.4 muestra la intensidad de la erosión en la cuchara, representada en términos de volumen extraído. Se puede observar que la erosión es más intensa en la zona del splitter, lo que indica que estas regiones experimentan un mayor impacto de las partículas.

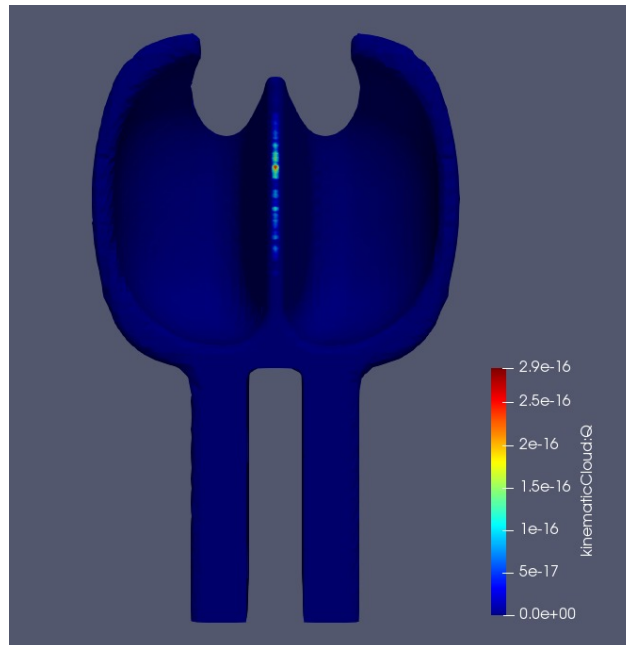


Figura 5.4: Intensidad de la erosión en la cuchara de la turbina Pelton, representada en términos de volumen extraído.

5.3.1.2. Cálculo de la erosión normalizada

La simulación permitió calcular la erosión normalizada a partir de los resultados obtenidos en términos de volumen extraído. Se utilizó el modelo de predicción de erosión de Finnie, descrito en la sección 2.3.1. OpenFOAM cuenta con una herramienta de cálculo de erosión basada en el modelo de Finnie que permite obtener el volumen extraído en cada punto de la malla en formato CSV. Sumando los valores de volumen extraído de cada punto de la malla, se obtuvo la erosión total en la cuchara. Para comparar los resultados con los obtenidos por Kumar et al. [1], fue necesario normalizar los valores de la erosión. En el experimento, esta normalización se realizó debido a que se estudiaron distintas cucharas con dimensiones ligeramente variables, lo cual podría influir notablemente en los resultados aunque las variaciones fueran pequeñas.

La erosión normalizada se calculó utilizando la siguiente ecuación:

$$E_n = \frac{V_e}{V_t} \quad (5.1)$$

donde E_n representa la erosión normalizada, V_e es el volumen erosionado total y V_t es el volumen total de la cuchara. A partir de los datos de volumen extraído en cada punto de la malla, se calculó el volumen erosionado total y se dividió por el volumen total de la cuchara para obtener la erosión normalizada.

5.3.1.3. Comparación de los resultados de erosión normalizada

La Tabla 5.3 presenta una comparación entre los resultados de erosión normalizada obtenidos en la simulación y los resultados experimentales de Kumar [1]. Además, se incluyen las diferencias porcentuales relativas en base a los resultados de Kumar.

Tabla 5.3: Comparación de los resultados de erosión normalizada obtenidos en la simulación y en el estudio experimental de Kumar [1].

Hora	Resultados del estudio	Resultados de Kumar	Dif %
3	0.0004315	0.0007	38.36
6	0.0008635	0.0012	28.04
9	0.0012955	0.0017	23.79
12	0.0017275	0.0022	21.48

5.3.1.4. Comparación con el estudio experimental de Kumar

Fotografías del experimento de Kumar Se presenta una serie de fotografías de la cuchara de la turbina Pelton tomadas a lo largo del experimento realizado por Kumar et al. [1], que ilustran la erosión progresiva y la eliminación del recubrimiento. La imagen se titula “progressive erosion and coating removal” y se puede ver en la Figura 5.5.

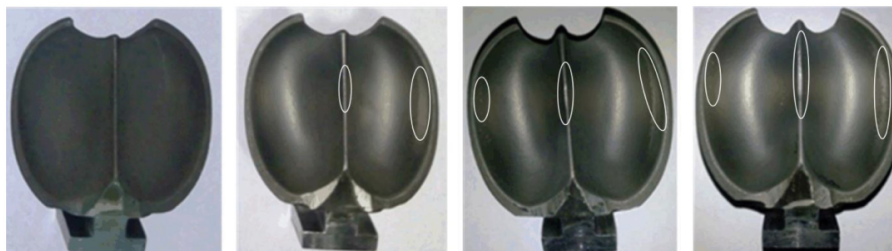


Figura 5.5: Progresión de la erosión y eliminación del recubrimiento en la cuchara de la turbina Pelton a lo largo del experimento realizado por Kumar et al. [1].

Las fotografías muestran claramente cómo la erosión y el desgaste del recubrimiento aumentan a medida que avanza el experimento. En las imágenes, se puede observar que los primeros indicios de erosión aparecen en el centro (en el splitter) y en el borde derecho de la cuchara (segunda imagen). Posteriormente, también aparece erosión en el borde izquierdo de la cuchara (tercera imagen), mientras que las otras zonas de erosión aumentan de tamaño. En la cuarta imagen, las tres zonas incrementan su tamaño aún más. Estos resultados experimentales proporcionan una base para comparar y validar los resultados obtenidos en la simulación realizada en este trabajo.

A diferencia de las imágenes experimentales, la simulación realizada en este trabajo muestra erosión solo en la zona del splitter. Esta diferencia puede deberse a que no se consideró el rebote de las partículas en la simulación por razones de eficiencia computacional. Esta simplificación podría explicar la ausencia de erosión en los bordes de la cuchara y las diferencias en la cantidad total de erosión observada. Es posible mencionar esta diferencia aquí y profundizar más en ella en la sección de conclusiones.

Comparación de resultados de erosión normalizada A continuación, se presenta un gráfico que compara los resultados de erosión normalizada obtenidos en la simulación realizada en este trabajo con los resultados experimentales de Kumar et al. [1]. El gráfico se muestra en la Figura 5.6.

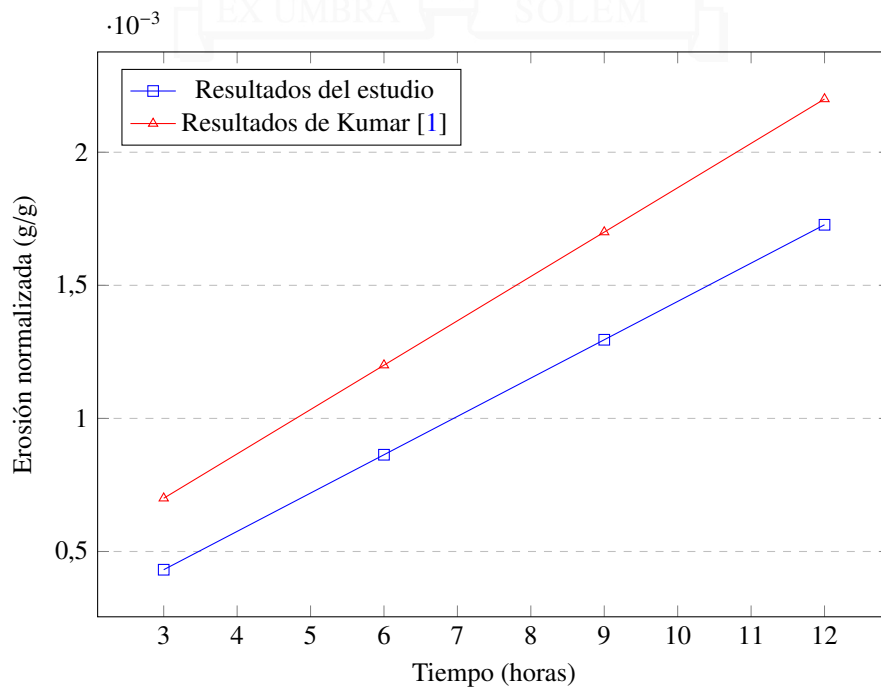


Figura 5.6: Comparación de la erosión normalizada obtenida en la simulación y en el estudio experimental de Kumar [1].

Analizando el gráfico de la Figura 5.6, se pueden observar similitudes y diferencias entre los dos conjuntos de datos. Si bien hay discrepancias en los valores absolutos de la erosión normalizada, la tendencia general de los resultados de la simulación sigue un patrón similar al de los resultados experimentales. Esto sugiere que, a pesar de las simplificaciones realizadas en la simulación, como la omisión del rebote de partículas, la simulación es capaz de capturar el comportamiento general de la erosión en la cuchara de la turbina Pelton.

Las diferencias observadas entre los resultados de la simulación y los datos experimentales pueden deberse a varias razones, incluida la simplificación mencionada anteriormente y las diferencias en las condiciones de operación, como la velocidad y el ángulo de impacto de las partículas en la simulación y el experimento. A pesar de estas diferencias, la simulación proporciona una herramienta útil para estudiar la erosión en las cucharas de las turbinas Pelton y puede ser mejorada en futuros trabajos, incluyendo factores como el rebote de partículas y las condiciones de operación más realistas.

En resumen, aunque existen discrepancias entre los resultados de la simulación y los datos experimentales de Kumar et al. [1], la simulación muestra una tendencia general similar y proporciona información valiosa sobre el proceso de erosión en la cuchara de la turbina Pelton. Estos resultados pueden ser útiles para mejorar el diseño de las cucharas y desarrollar estrategias de mantenimiento para prolongar la vida útil de las turbinas Pelton.

5.3.1.5. Discusión de resultados y limitaciones

Bug en OpenFOAM Un aspecto importante a considerar en la simulación es la presencia de un bug en OpenFOAM que provoca que algunas partículas se peguen a la cuchara en lugar de desaparecer al impactar, como se esperaba [63, 64]. Esto afecta los resultados de la simulación, ya que puede conducir a una sobreestimación de la erosión en ciertas áreas de la cuchara. Para mitigar el impacto de este bug en los resultados presentados, se examinaron cuidadosamente las áreas afectadas y se compararon con las áreas de erosión observadas en el estudio experimental de Kumar et al. [1].

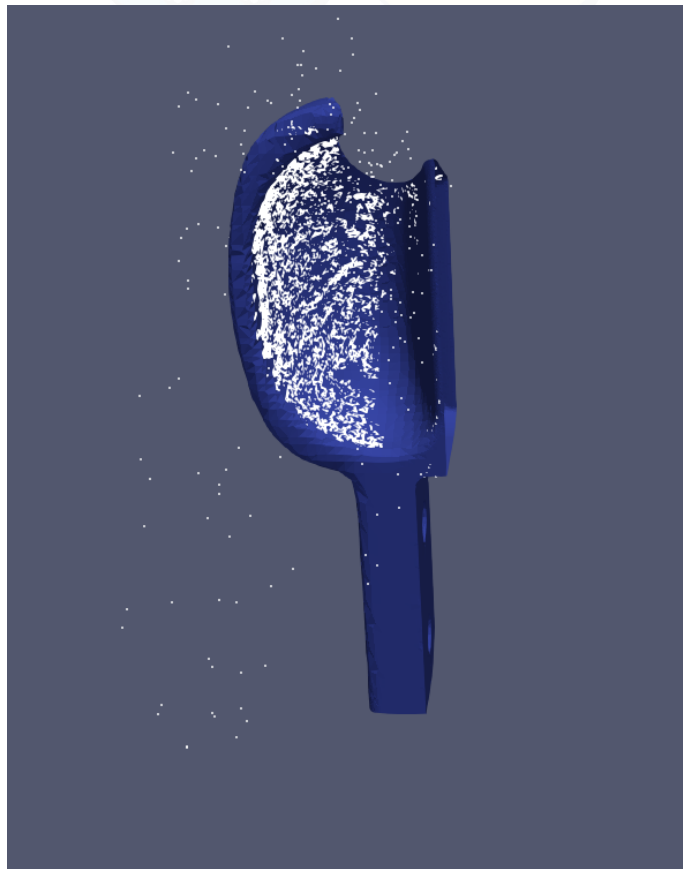


Figura 5.7: Partículas pegadas en la cuchara de la turbina Pelton en la simulación.

La Figura 5.7 muestra las partículas pegadas en la cuchara de la turbina Pelton en la simulación. Estas partículas están relacionadas con el bug mencionado y podrían estar contribuyendo a una sobreestimación de la erosión en estas áreas. A pesar de este problema, los resultados generales de la simulación aún proporcionan información valiosa sobre la erosión en la cuchara de la turbina Pelton y cómo se comparan con el estudio experimental.

Limitaciones de la simulación La simulación también tiene algunas limitaciones que deben ser consideradas al interpretar los resultados. Una limitación importante es el tiempo de simulación, que fue de solo 450

segundos debido a las restricciones computacionales y al tiempo necesario para obtener resultados de erosión. Aunque este tiempo de simulación es más corto que los tiempos de exposición del estudio experimental de Kumar et al. [1], se observó que la erosión normalizada era lineal en los resultados preliminares y experimentales. Por lo tanto, se decidió proyectar los resultados de la simulación utilizando una ecuación lineal obtenida a partir de estos datos.

A pesar de estas limitaciones, la simulación proporciona información útil sobre la erosión en la cuchara de la turbina Pelton y cómo se compara con los resultados experimentales. La simulación también puede ser mejorada en futuros trabajos abordando estas limitaciones y considerando factores adicionales que puedan afectar la erosión, como el rebote de partículas y condiciones de operación más realistas.

Ausencia del movimiento de la turbina Pelton Otra limitación importante de la simulación es que no se consideró el movimiento de la turbina Pelton durante el proceso de erosión. En condiciones reales, la turbina estaría girando a alta velocidad, lo que podría afectar la dinámica de las partículas y, en consecuencia, la distribución y la intensidad de la erosión en la cuchara.

El hecho de que el movimiento de la turbina no se haya tenido en cuenta en la simulación podría haber resultado en diferencias entre los resultados obtenidos en la simulación y los observados en el estudio experimental de Kumar et al. [1]. Estas diferencias podrían manifestarse en la localización y magnitud de la erosión en distintas áreas de la cuchara.

6 | Análisis y Conclusiones

Este trabajo se centró en estimar la erosión en una turbina hidráulica del tipo Pelton mediante métodos computacionales, utilizando la herramienta OpenFOAM 8 y la librería DPMFoam. Se estudiaron los parámetros más relevantes para llevar a cabo las simulaciones, incluyendo la discretización, los esquemas numéricos y los modelos de erosión. Además, se validó el modelo comparándolo con un experimento del tipo jet impingement y se realizó una simulación para estimar la erosión en una turbina Pelton utilizando parámetros de un experimento real en India.

La validación del modelo mediante la comparación con el experimento jet impingement demostró la capacidad del modelo para predecir la posición, velocidad y ángulo de impacto de las partículas en la superficie del objetivo. A pesar de algunas diferencias entre los resultados de la simulación y los datos experimentales, atribuidas principalmente a las actualizaciones de la librería DPMFoam, la validación mostró una buena correspondencia en términos generales.

En cuanto a la simulación de la erosión en la turbina Pelton, los resultados obtenidos mostraron zonas de desgaste y erosión en la superficie de las palas de la turbina que coincidieron con las observadas en el experimento real llevado a cabo por Kumar. Se observó una buena correspondencia entre la zona de erosión en la simulación y la del experimento, especialmente en la parte central donde se concentra el mayor impacto. Sin embargo, en la zona lateral de la pala, donde el experimento de Kumar muestra una mayor erosión, no se observa un impacto similar en la simulación. Esta discrepancia se debe a que el modelo de simulación no considera el rebote de partículas, que es lo que provoca principalmente el impacto en esa zona.

Además, se discutió la diferencia en la erosión total de la cuchara entre la simulación y el experimento de Kumar, atribuyéndola a la falta de consideración del rebote de partículas en la simulación. La erosión observada en la parte baja de la pala en la simulación se presenta debido a que la pala no presenta movimiento, ni las palas que le siguen y le preceden como sucedería en una situación normal. Se reconoce que el diseño de la pala en la simulación no es el mismo que el de la experiencia, aunque se adaptó para tener dimensiones similares.

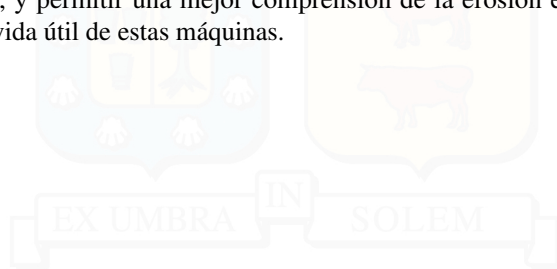
En cuanto a las perspectivas de investigación, hay varios trabajos que podrían complementar y mejorar este estudio. Uno de los temas que podrían mejorarse bastante es la predicción de la erosión mediante la incorporación de la deformación de la pala a medida que se desgasta. Como se menciona en la Sección 2.2, a medida que el material se desgasta, la erosión va siendo más efectiva, y uno de estos factores es el ángulo de impacto. Incorporar esta deformación en la simulación podría mejorar la precisión de los resultados.

Otro tema importante es la implementación de la erosión en la turbina Pelton con todas las palas y con movimiento. En la NTNU (Norwegian University of Science and Technology), se simuló de esta manera la turbina Pelton [65], aunque el objetivo del trabajo no incluye el estudio de la erosión. Realizar simulaciones de erosión en estas condiciones podría permitir una mejor comprensión del fenómeno en situaciones más reales.

Finalmente, sería de gran utilidad realizar simulaciones con distintos revestimientos para poder compararlos entre sí en distintos casos, como con distintas palas o materiales de las partículas. Esto podría ser de gran utilidad antes de decidir invertir en protecciones para las turbinas, y permitiría una mejor comprensión de cómo los revestimientos pueden influir en la erosión y en la vida útil de las turbinas.

En conclusión, este estudio permitió estimar la erosión en una turbina hidráulica tipo Pelton mediante métodos computacionales, utilizando la herramienta OpenFoam 8 y considerando los parámetros más importantes. La validación del modelo mediante la comparación con un experimento real demostró la capacidad del modelo para predecir la posición, velocidad y ángulo de impacto de las partículas en la superficie del objetivo. La simulación final permitió encontrar zonas de desgaste y erosión en la superficie de las palas de la turbina que coincidieron con las observadas en el experimento real.

A partir de los resultados obtenidos, se discutieron algunas recomendaciones y sugerencias para futuros estudios, incluyendo la incorporación de la deformación de la pala a medida que se desgasta, la implementación de la erosión en la turbina Pelton con todas las palas y con movimiento, y la realización de simulaciones con distintos revestimientos. Estos trabajos podrían contribuir a ampliar y profundizar nuestro conocimiento en esta área, y permitir una mejor comprensión de la erosión en turbinas hidráulicas y su impacto en la eficiencia y vida útil de estas máquinas.



Bibliografía

- [1] Anant Kumar Rai, Arun Kumar, and Thomas Staubli. Effect of concentration and size of sediments on hydro-abrasive erosion of pelton turbine. *Renewable Energy*, 145:893–902, 2020. (document), 3.5, 5.1, 5.1, 5.2, 5.3, 5.3.1.2, 5.3.1.3, 5.3, 5.3.1.4, 5.5, 5.3.1.4, ??, 5.6, 5.3.1.4, 5.3.1.5, 5.3.1.5, 5.3.1.5
- [2] Sailesh Chitrakar, Hari Prasad Neopane, and Ole Gunnar Dahlhaug. A review on sediment erosion challenges in hydraulic turbines. *Sedimentation engineering*, page 9, 2018. 1
- [3] Alejandro López, Matthew T Stickland, and William M Dempster. Cfd study of fluid flow changes with erosion. *Computer Physics Communications*, 227:27–41, 2018. 1
- [4] Changguo Duan and Vladimir Y Karelin. *Abrasive erosion and corrosion of hydraulic machinery*, volume 2. World Scientific, 2003. 1, 2.2.1, 2.2.1
- [5] Reetuparna Das. When to Prototype? When to Simulate?, 2017. 1
- [6] Alice Hager, Christoph Kloss, and Christoph Goniva. Combining open source and easy access in the field of dem and coupled cfd-dem: Liggghts®, cfdem® coupling and cfdem® workbench. In *Computer Aided Chemical Engineering*, volume 43, pages 1699–1704. Elsevier, 2018. 1
- [7] HC Meng and KC Ludema. Wear models and predictive equations: their form and content. *Wear*, 181:443–457, 1995. 1
- [8] Hari Prasad Neopane, O Gunnar Dahlhaug, and Michel Cervantes. Sediment erosion in hydraulic turbines. *Global Journal of Researches in Engineering Mechanical and Mechanics Engineering*, 11(6):17–26, 2011. 1.1
- [9] Rahul Tarodiya, Subodh Khullar, and Avi Levy. Particulate flow and erosion modeling of a pelton turbine injector using cfd-dem simulations. *Powder Technology*, 399:117168, 2022. 1
- [10] Hrvoje Jasak. Openfoam: open source cfd in research and industry. *International Journal of Naval Architecture and Ocean Engineering*, 1(2):89–94, 2009. 1
- [11] Alejandro López, William Nicholls, Matthew T Stickland, and William M Dempster. Cfd study of jet impingement test erosion using ansys fluent® and openfoam®. *Computer Physics Communications*, 197:88–95, 2015. 1.1, 2.7, 3.3, 3.3, 3.3, 3.4, 3.4, 3.5, 3.7, 4.1, 4.1, 4.2, 4.1, 4.1, 4.4, 4.5, 4.1, 4.6, 4.7, 4.8, 4.1, 4.9, 4.10, 4.1, 4.2, 4.2.2.1
- [12] A Korpela Seppo. Steam turbines. principles of turbomachinery. 2.1, 2.1, 2.3
- [13] Robert W Fox, Alan T McDonald, and John W Mitchell. *Fox and McDonald's introduction to fluid mechanics*. John Wiley & Sons, 2020. 2.1, 2.2, 2.1, 3.1.1
- [14] Cunkui Huang, S Chiovelli, P Minev, Jingli Luo, and K Nandakumar. A comprehensive phenomenological model for erosion of materials in jet flow. *Powder Technology*, 187(3):273–279, 2008. 2.4, 2.3.2
- [15] Iain Finnie. Erosion of surfaces by solid particles. *wear*, 3(2):87–103, 1960. 2.3.1, 2.5, 5.2.1

- [16] Iain Finnie. Some observations on the erosion of ductile metals. *wear*, 19(1):81–90, 1972. 2.3.1, 2.3.1, 5.2.1
- [17] G Grant and W Tabakoff. An experimental investigation of the erosive characteristics of 2024 aluminum alloy. Technical report, Cincinnati Univ oh Dept of Aerospace Engineering, 1973. 2.3.3, 5.2.1
- [18] Alejandro Lopez, Matthew Stickland, and William Dempster. Modeling erosion in a centrifugal pump in an eulerian-lagrangian frame using openfoam®. *Open Engineering*, 5(1), 2015. 2.3.3, 2.7, 5.2.1
- [19] Qilong Bi. Numerical modelling of near-bottom sediment transport: Turbulence modulation, new process models and application to the scheldt & belgian coast. 2015. 2.4, 2.6
- [20] Kenichi Sugiyama, Kenji Harada, and Shuji Hattori. Influence of impact angle of solid particles on erosion by slurry jet. *Wear*, 265(5-6):713–720, 2008. 2.6, 2.7
- [21] ANKANJF Gnanavelu, N Kapur, A Neville, and JF Flores. An integrated methodology for predicting material wear rates due to erosion. *Wear*, 267(11):1935–1944, 2009. 2.6, 2.7, 2.7, 4.1
- [22] Files 3D Models | The GrabCAD Community Library Free CAD Designs. OpenFOAM Barycentric Tracking, 2019. 3
- [23] Suhas V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Taylor & Francis, New York, 1st edition, 1980. 3.1, 3.3
- [24] Richard H. Pletcher, John C. Tannehill, and Dale A. Anderson. *Computational Fluid Mechanics and Heat Transfer*. Taylor & Francis, New York, 4th edition, 2017. 3.1
- [25] E.L. Houghton, P.W. Carpenter, S.H. Collicott, B. Goldman, and D.T. Valentine. *Introduction to Computational Fluid Dynamics*. Cambridge University Press, New York, 1st edition, 2012. 3.1
- [26] John D. Anderson. *Computational Fluid Dynamics: The Basics with Applications*. McGraw-Hill, New York, 3rd edition, 2017. 3.1, 3.3
- [27] HK Versteeg and W Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson Education, 2007. 3.1, 3.1, 3.1.6, 3.2, 3.3, 3.3, 4
- [28] Rebecca Gullberg. Computational fluid dynamics in openfoam. *Report TKP*, 4555, 2017. 3.1.2, 3.1, 3.2
- [29] Richard Courant, Kurt Friedrichs, and Hans Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische annalen*, 100(1):32–74, 1928. 3.1.3
- [30] Christopher Greenshields. *OpenFOAM v10 User Guide*. The OpenFOAM Foundation, London, UK, 2022. 3.1.4
- [31] W Malalasekera and HK Versteeg. An introduction to computational fluid dynamics. *The finite volume method, Harlow: Prentice Hall*, page 1995, 2007. 3.1.6, 3.1.6, 3.1.6, 3.1.6
- [32] Joel H Ferziger and Mirsad Perić. Computational methods for fluid dynamics. In *Springer*, 2002. 3.3, 4
- [33] Alexandre Perrig. Hydrodynamics of the free surface flow in pelton turbine buckets. Technical report, EPFL, 2007. 3.1.6
- [34] Florian R Menter. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA journal*, 32(8):1598–1605, 1994. 3.1.6
- [35] OpenCFD Ltd. OpenFOAM. <https://www.openfoam.com/>, 2021. Accedido el 28 de febrero de 2023. 3.2
- [36] Henry G. Weller, Gavin Tabor, Hrvoje Jasak, and Christer Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12(6):620–631, 1998. 3.2

- [37] OpenFOAM Foundation. About OpenFOAM. <https://www.openfoam.com/about.php>, 2021. Accedido el 28 de febrero de 2023. 3.2
- [38] OpenFOAM. What is openfoam? <https://www.openfoam.com/about.php>, 2021. Accessed: 2023-02-26. 3.2
- [39] Hrvoje Jasak. Error analysis and estimation for the finite volume method with applications to fluid flows. *Imperial College London*, 1996. 3.2
- [40] OpenFOAM. User guide. <https://www.openfoam.com/documentation/user-guide/>, 2021. Accessed: 2023-02-26. 3.2, 3.6, 3.6
- [41] F. Moukalled, L. Mangani, and M. Darwish. *The finite volume method in computational fluid dynamics: An advanced introduction with OpenFOAM and Matlab*. Springer, 2016. 3.2
- [42] Physics Stack Exchange. Eulerian description for a deforming control volume. <https://physics.stackexchange.com/questions/537111/eulerian-description-for-a-deforming-control-volume>. Accessed: 2023-03-01. 3.4, 3.5
- [43] F. M. White. *Viscous fluid flow*. John Wiley & Sons, 2006. 3.3
- [44] P. A. Cundall and O. D. Strack. A discrete numerical model for granular assemblies. *Géotechnique*, 29(1):47–65, 1979. 3.3
- [45] J. J. Monaghan. Simulating free surface flows with sph. *Journal of computational physics*, 110(2):399–406, 1994. 3.3
- [46] J. D. Anderson Jr. *Fundamentals of Aerodynamics*. McGraw-Hill Education, 2010. 3.6
- [47] H. K. Versteeg and W. Malalasekera. An introduction to computational fluid dynamics. In *The finite volume method*. Prentice Hall, 2001. 3.6
- [48] W Bainbridge. OpenFOAM Barycentric Tracking, 2017. 3.7
- [49] Wei Shyy, Bo Cheng, and Jae Wan Lee. A review of computational fluid dynamics applications in renewable energy. *Renewable and Sustainable Energy Reviews*, 109:357–374, 2019. 4
- [50] Yujue Wang, Menglan Duan, and Jinghai Li. Advances in computational fluid dynamics (cfD) in the oil and gas industry. *Journal of Natural Gas Science and Engineering*, 24:25–42, 2015. 4
- [51] ANSI. Software engineering—guidelines for the application of ansi/iso/iec 12207:1995 in the acquisition of software, 1990. ANSI/IEEE Std 12207.0-1990. 4
- [52] AIAA. Guide for the verification and validation of computational fluid dynamics simulations. Technical Report G-077-2009, American Institute of Aeronautics and Astronautics, 2009. 4
- [53] ASME. *Standard for Verification and Validation in Computational Fluid Dynamics and Heat Transfer*. ASME V&V 20-2006, 2006. 4
- [54] Klaus-Jürgen Bathe and Ravi Chaudhary. An overview of the ptc 60/v&v 10: guide for verification and validation in computational solid mechanics. *Journal of Applied Mechanics*, 77(2):020802, 2010. 4
- [55] PJ Roache. Verification and validation in computational science and engineering. *Hermosa Publishers*, 2009. 4
- [56] Christopher J Roy and William L Oberkampf. Verification and validation of computational fluid dynamics simulations. *Progress in Aerospace Sciences*, 41(1):1–14, 2005. 4
- [57] Vernon J Rossow and Paul J Schuele. Assessment of error measures for comparison of cfd and experiment. In *International Symposium on Transport Phenomena and Dynamics of Rotating Machinery*, pages 341–350. Springer, 2017. 4

- [58] William L Oberkampf, Timothy G Trucano, and Charles Hirsch. Verification, validation, and predictive capability in computational engineering and physics. *Applied mechanics reviews*, 63(1):010801, 2010. 4
- [59] Nicholas Robert Jankowski. Review of two-phase electronics cooling for army vehicle applications. *SAE International Journal of Aerospace*, 10(1):59–66, 2017. 4.1
- [60] Chris Greenshields. 2.3.0: Physical Modelling, 2014. 4.1, 4.3, 4.1
- [61] A G Gnanavelu, SD Pathak, and T Sundararajan. Effect of jet-to-target distance on erosion behaviour of hvof sprayed wc-17co coatings. *Surface and Coatings Technology*, 235:540–547, 2013. 4.1
- [62] VB Nguyen, QB Nguyen, ZG Liu, S Wan, CYH Lim, and YW Zhang. A combined numerical–experimental study on the effect of surface evolution on the water–sand multiphase flow characteristics and the material erosion behavior. *Wear*, 319(1-2):96–109, 2014. 4.2.2.1
- [63] strange stuck particles on the rebound wall in MPPICFoam – CFD Online Discussion Forums, 2020. 5.2, 5.3.1.5
- [64] Particle Fate in DPMFoam or MPPICFoam – CFD Online Discussion Forums, 2019. 5.2, 5.3.1.5
- [65] Jone Rivrud Rygg. Cfd analysis of a pelton turbine in openfoam. Master’s thesis, Institutt for energi-og prosessteknikk, 2013. 6

A | Código OpenFoam validación 5 mm

A.1. blockMeshDict

```

convertToMeters 0.01;

vertices
(
    (0 0 0)
    (0 -0.35 0)
    (0 0 -0.35)
    (0 0.35 0)

    (0 -3.15 0)
    (0 0 -3.15)
    (0 3.15 0)

    (11.5 0 0)
    (11.5 -0.35 0)
    (11.5 0 -0.35)
    (11.5 0.35 0)
    (11.5 -3.15 0)
    (11.5 0 -3.15)
    (11.5 3.15 0)

);

blocks
(
    hex (1 2 0 0 8 9 7 7) (10 4 140) simpleGrading (1 1 ((0.86 0.65 1) (0.11 0.3 1) (0.03 0.05 1)))
    hex (2 3 0 0 9 10 7 7) (10 4 140) simpleGrading (1 1 ((0.86 0.65 1) (0.11 0.3 1) (0.03 0.05 1)))
    hex (4 5 2 1 11 12 9 8) (10 32 140) simpleGrading (1 1 ((0.86 0.65 1) (0.11 0.3 1) (0.03 0.05
1)))
    hex (5 6 3 2 12 13 10 9) (10 32 140) simpleGrading (1 1 ((0.86 0.65 1) (0.11 0.3 1) (0.03 0.05
1)))

);

edges
(
    arc 1 2 (0 -0.2474873734 -0.2474873734)
    arc 2 3 (0 0.2474873734 -0.2474873734)
    arc 8 9 (11.5 -0.2474873734 -0.2474873734)

```

```

arc 9 10 (11.5 0.2474873734 -0.2474873734)
arc 4 5 ( 0 -2.2273863607 -2.2273863607)
arc 11 12 (11.5 -2.2273863607 -2.2273863607)
arc 5 6 ( 0 2.2273863607 -2.2273863607)
arc 12 13 (11.5 2.2273863607 -2.2273863607)
);

```

```

    boundary
(
/*
frontAndBack

```

```

type wall;
faces
(
(0 3 2 1)
);
*/

```

```

    inlet

```

```

type patch;
faces
(
(2 0 0 1)
(3 0 0 2)
);

```

```

front

```

```

type symmetryPlane;
faces
(
(0 1 8 7)
(3 0 7 10)
(1 4 11 8)
(6 3 10 13)
);

```

```

    wall1

```

```

type wall;
faces
(
(1 2 5 4)
(3 6 5 2)
);

```

```

wall2

```

```

type wall;
faces

```



```
(
(10 7 7 9)
(9 7 7 8)
(8 11 12 9)
(10 9 12 13)
);

lowerWall

type wall;
faces
(
(4 5 12 11)
);

upperWall

type wall;
faces
(
(12 5 6 13)
);

);

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

A.2. snappyHexMesh

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

// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry

target type triSurfaceMesh; file "target.stl";
wall type triSurfaceMesh; file "wall.stl";

;
```

```

    // Settings for the castellatedMesh generation.
    castellatedMeshControls

    // Refinement parameters
    //

    // If local number of cells is >= maxLocalCells on any processor
    // switches from from refinement followed by balancing
    // (current method) to (weighted) balancing before refinement.
    maxLocalCells 2500000;//100000

    // Overall cell limit (approximately). Refinement will stop immediately
    // upon reaching this number so a refinement level might not complete.
    // Note that this is the number of cells before removing the part which
    // is not 'visible' from the keepPoint. The final number of cells might
    // actually be a lot less.
    maxGlobalCells 14000000;

    // The surface refinement loop might spend lots of iterations refining just a
    // few cells. This setting will cause refinement to stop if <= minimumRefine
    // are selected for refinement. Note: it will at least do one iteration
    // (unless the number of cells to refine is 0)
    minRefinementCells 0;

    // Allow a certain level of imbalance during refining
    // (since balancing is quite expensive)
    // Expressed as fraction of perfect balance (= overall number of cells /
    // nProcs). 0=balance always.
    maxLoadUnbalance 0.10;

    // Number of buffer layers between different levels.
    // 1 means normal 2:1 refinement restriction, larger means slower
    // refinement.
    nCellsBetweenLevels 3;

    // Explicit feature edge refinement
    //

    // Specifies a level for any cell intersected by its edges.
    // This is a featureEdgeMesh, read from constant/triSurface for now.
    features
    (

    );

    // Surface based refinement
    //

```

```
// Specifies two levels for every surface. The first is the minimum level,
// every cell intersecting a surface gets refined up to the minimum level.
// The second level is the maximum level. Cells that 'see' multiple
// intersections where the intersections make an
// angle > resolveFeatureAngle get refined up to the maximum level.
```

```
refinementSurfaces
```

```
wall
```

```
// Surface-wise min and max refinement level
level (0 0);
```

```
// Optional specification of patch type (default is wall). No
// constraint types (cyclic, symmetry) etc. are allowed.
patchInfo
```

```
type wall;
//inGroups (motorBikeGroup);
```

```
target
// Surface-wise min and max refinement level
level (0 0);
```

```
// Optional specification of patch type (default is wall). No
// constraint types (cyclic, symmetry) etc. are allowed.
patchInfo
```

```
type wall;
//inGroups (motorBikeGroup);
```

```
// Resolve sharp angles
resolveFeatureAngle 180;
```

```
// Region-wise refinement
//
```

```
// Specifies refinement level for cells in relation to a surface. One of
// three modes
// - distance. 'levels' specifies per distance to the surface the
// wanted refinement level. The distances need to be specified in
// descending order.
// - inside. 'levels' is only one entry and only the level is used. All
// cells inside the surface get refined up to the level. The surface
// needs to be closed for this to be possible.
```

// - outside. Same but cells outside.

refinementRegions

refinementBox

mode inside;
levels ((1E15 1)); // levels ((1E15 1))
gapLevel (6 6 6)
gapMode inside;

// Mesh selection

//

// After refinement patches get added for all refinementSurfaces and
// all cells intersecting the surfaces get put into these patches. The
// section reachable from the locationInMesh is kept.
// NOTE: This point should never be on a face, always inside a cell, even
// after refinement.
locationInMesh (0.003 0.0019 0);

// Whether any faceZones (as specified in the refinementSurfaces)
// are only on the boundary of corresponding cellZones or also allow
// free-standing zone faces. Not used if there are no faceZones.
allowFreeStandingZoneFaces true;

// Settings for the snapping.
snapControls

//- Number of patch smoothing iterations before finding correspondence
// to surface
nSmoothPatch 3;

//- Relative distance for points to be attracted by surface feature point
// or edge. True distance is this factor times local
// maximum edge length.
tolerance 2.0;

//- Number of mesh displacement relaxation iterations.
nSolveIter 30;

//- Maximum number of snapping relaxation iterations. Should stop
// before upon reaching a correct mesh.
nRelaxIter 5;

```
// Feature snapping

// Number of feature edge snapping iterations.
// Leave out altogether to disable.
nFeatureSnapIter 10;

// Detect (geometric only) features by sampling the surface
// (default=false).
implicitFeatureSnap false;

// Use castellatedMeshControls::features (default = true)
explicitFeatureSnap true;

// Detect points on multiple surfaces (only for explicitFeatureSnap)
multiRegionFeatureSnap false;

// Settings for the layer addition.
addLayersControls

// Are the thickness parameters below relative to the undistorted
// size of the refined cell outside layer (true) or absolute sizes (false).
relativeSizes true;

// Per final patch (so not geometry!) the layer information
layers

"(lowerWall|wall).*"

nSurfaceLayers 1;

// Expansion factor for layer mesh
expansionRatio 1.0;

// Wanted thickness of final added cell layer. If multiple layers
// is the thickness of the layer furthest away from the wall.
// Relative to undistorted size of cell outside layer.
// See relativeSizes parameter.
finalLayerThickness 0.3;

// Minimum thickness of cell layer. If for any reason layer
// cannot be above minThickness do not add layer.
// Relative to undistorted size of cell outside layer.
minThickness 0.1;

// If points get not extruded do nGrow layers of connected faces that are
// also not grown. This helps convergence of the layer addition process
```

```
// close to features.
// Note: changed(corrected) w.r.t 17x! (didn't do anything in 17x)
nGrow 0;

// Advanced settings

// When not to extrude surface. 0 is flat surface, 90 is when two faces
// are perpendicular
featureAngle 60;

// At non-patched sides allow mesh to slip if extrusion direction makes
// angle larger than slipFeatureAngle.
slipFeatureAngle 30;

// Maximum number of snapping relaxation iterations. Should stop
// before upon reaching a correct mesh.
nRelaxIter 3;

// Number of smoothing iterations of surface normals
nSmoothSurfaceNormals 1;

// Number of smoothing iterations of interior mesh movement direction
nSmoothNormals 3;

// Smooth layer thickness over surface patches
nSmoothThickness 10;

// Stop layer growth on highly warped cells
maxFaceThicknessRatio 0.5;

// Reduce layer growth where ratio thickness to medial
// distance is large
maxThicknessToMedialRatio 0.3;

// Angle used to pick up medial axis points
// Note: changed(corrected) w.r.t 17x! 90 degrees corresponds to 130 in 17x.
minMedianAxisAngle 90;

// Create buffer region for new layer terminations
nBufferCellsNoExtrude 0;

// Overall max number of layer addition iterations. The mesher will exit
// if it reaches this number of iterations; possibly with an illegal
// mesh.
nLayerIter 50;
```

```

// Generic mesh quality settings. At any undoable phase these determine
// where to undo.
meshQualityControls

include "meshQualityDict"

// Advanced

// Write flags
writeFlags
(
  scalarLevels
  layerSets
  layerFields // write volScalarField for layer coverage
);

// Merge tolerance. Is fraction of overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1e-6;

// ***** //

```

A.3. controlDict

```

application DPMFoam;

startFrom latestTime;

startTime 0;

stopAt endTime;

endTime 1;

deltaT 0.1;

writeControl runTime;
writeInterval 0.1;

purgeWrite 0;

writeFormat binary;

```

```

writePrecision 6;

writeCompression off;

timeFormat general;

timePrecision 6;

runTimeModifiable yes;

adjustTimeStep yes;

maxCo 1;

maxDeltaT 0.1;

// *****

```

A.4. fvSchemes

```

ddtSchemes
default Euler;

```

```

gradSchemes

default Gauss linear;

```

```

divSchemes

default none;
div(alphaPhi.water,k.water) bounded Gauss upwind;
div(alphaPhi.water,epsilon.water) bounded Gauss upwind;
//div(phi,omega) bounded Gauss upwind;
div(alphaPhi.water,U.water) Gauss linearUpwindV unlimited;
div(((alpha.water*nuEff.water)*dev2(T(grad(U.water))))) Gauss linear;

```

```

laplacianSchemes

default Gauss linear corrected;

```

```

interpolationSchemes

```

```
default linear;
```

```
snGradSchemes
```

A.5. fvSolution

```
solvers
```

```
"(p|kinematicCloud:theta)"
```

```
solver GAMG;
tolerance 1e-06;
relTol 0.01;
smoother GaussSeidel;
```

```
"(p|kinematicCloud:theta)Final"
```

```
solver GAMG;
tolerance 1e-04;
relTol 0;
smoother GaussSeidel;
```

```
"(U.water|k.wather|epsilon.wather)"
```

```
solver smoothSolver;
smoother symGaussSeidel;
tolerance 1e-04;
relTol 0.1;
```

```
"(U.water|k.water|epsilon.water)Final"
```

```
solver smoothSolver;
smoother symGaussSeidel;
tolerance 1e-04;
relTol 0;
```

```
PIMPLE
```

```
nOuterCorrectors 1;
nCorrectors 2;
momentumPredictor yes;
nNonOrthogonalCorrectors 0;
pRefCell 0;
pRefValue 0;
```

relaxationFactors

// ***** //

A.6. mapFieldsDict

```
// List of pairs of source/target patches for mapping
patchMap
(
);
```

```
// List of target patches cutting the source domain (these need to be
// handled specially e.g. interpolated from internal values)
cuttingPatches
(
);
```

// ***** //

A.7. meshQualityDict

```
// Include defaults parameters from master dictionary
includeEtc caseDicts/mesh/generation/meshQualityDict"
```

```
//- minFaceWeight (0 -> 0.5)
minFaceWeight 0.05;
```

```
minVol 1e-13;
maxBoundarySkewness 4;
maxInternalSkewness 4;
maxConcave 80;
minVolRatio 0.01;
minTringleTwist -1;
//minVolCollapseRatio -0.000001;
minDeterminant 0.001;
// ***** //
```

A.8. setFieldsDict

```
defaultFieldValues
(
```

```
volScalarFieldValue alpha.air 1
volScalarFieldValue alpha.water 0
);
```

```
regions
(
```

```
);
```

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

A.9. surfaceFeaturesDict

```
surfaces ("target.stl");
```

```
// Identify a feature when angle between faces < includedAngle
includedAngle 150;
```

```
subsetFeatures
```

```
// Keep nonManifold edges (edges with >2 connected faces)
nonManifoldEdges no;//no
```

```
// Keep open edges (edges with 1 connected face)
openEdges no;//yes
```

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

A.10. kinematicCloudProperties

```
solution//particle solution settings
active true;//true for particle adding-Activate or deactivate the particle cloud
coupled false;//true for the influence of parcels to the continuous phase-Enable or disable phase
coupling
transient yes;//true for transient simulation-Transient or steady-state solution
cellValueSourceCorrection on;//correction of cell values within the Lagrangian time step -//Enable/disable
correction of momentum transferred to the Eulerian phase
maxCo 1; //maximum courant number
calcFrequency 0.001;
maxTrackTime 100;
```

```
interpolationSchemes//interpolation schemes for continuous fields
```

```
rho.water cell;
U.water cellPoint;
```

```
mu.water cell;
alpha.water cell;
```

```
averagingMethod basic;//dual for dual-mesh Lagrangian averaging procedure - averaging over the
cells and dual
(tetrahedral) volumes
```

```
integrationSchemes//definition of integration schemes of velocity field U
```

```
U Euler;
```

```
sourceTerms//scheme for source term in momentum equation of continuous phase, relevant if coupled
is activated
```

```
schemes
```

```
U semiImplicit 1;//1//explicit or semiImplicit, relaxCoeff for each of the fields
```

```
constantProperties//density and max fraction of particles
```

```
rho0 2206;//densidad de la particula kg/m3
```

```
alphaMax 0.9;//
```

```
poissonsRatio 0.35;//ratio of traverse to axial strain for your particles (dimensionless)
```

```
youngsModulus 1e8;// 1e8 constant of proportionality relating the deformation to the applied stress
for your
particles (N/m2)
```

```
subModels //particular particle properties definition-Specify applicable forces on the particles
particleForces//particle drag modelling
/*WenYuDrag
```

```
alphac alpha.water;
*/
sphereDrag;
```

```
//gravity;
```

```
injectionModels
```

```

    model1//method of particle injection (model1 - user name of model defined below)
type patchInjection;//patchInjection for injection at the patch
parcelBasisType fixed;//parcel basis enumeration/options of how to represent your particles. besides
'fixed' they can be represented by 'number' or 'mass'
patchName inlet;//name of the patch for particle injection
U0 (10 0 0);
parcelsPerSecond 53308;//number of particle per second
meanParticleDiameter 0.00025;
nParticle 1;//number of particles to be injected
//massFlowRate 0.0001;
//meanParticleDiameter 0.00025

    sizeDistribution //size of particles constant (fixedValue), or e.g. normal distribution (normal)
type fixedValue;//type of size distribution
fixedValueDistribution
value 0.00025;//

    flowRateProfile constant 1;//distribution of injected particle with respect to the duration, constant 1
means uniform
massTotal 0;//total mass to inject
SOI 0;//start time of injection
duration 2000;//

    dispersionModel none;//no dispersion model is implemented yet

    patchInteractionModel localInteraction;//localInteraction for patch-by-patch interaction specification

    localInteractionCoeffs//patch interaction definition

patches
(

    front

type none;//symmetryPlane o none

    "wall|lowerWall|upperWall"

type rebound;
e 1;
mu 0;

    target

type escape;//rebound
e 0;

```

mu 0;

inlet|wall1|wall2"

type escape;//escape for outlet patches

);

heatTransferModel none;//none for no heat transfer modelling

surfaceFilmModel none;//none for no surfaceFilmModel-Surface film model for dripping and film interaction
//absorb, bounce, splash

collisionModel none;//pairCollision

packingModel none;//implicit model for applying an inter-particle stress to the particles

dampingModel relaxation;//no model for collision damping time of particle velocity fluctuation

relaxationCoeffs

timeScaleModel

type nonEquilibrium;
alphaPacked 0.67;
e 0.9;

isotropyModel stochastic;//stochastic return-to-isotropy model - adding collision term that causes velocity distributions to relax to isotropic Gaussian distributions

stochasticCoeffs

timeScaleModel

type isotropic;//isotropic model for the time scale over which the velocity field of a dispersed phase tends towards an isotropic distribution
alphaPacked 0.67;
e 0.9;

stochasticCollisionModel none;//Particle collision: spring, slider+dash-pot

```
radiation off;//no radiation model
```

```
// Use cloudFunctions to record particle tracks, calculate particle erosion
cloudFunctions
```

```
patchPostProcessing
```

```
type patchPostProcessing;
maxStoredParcels 1000000;
patches
(
target
);
```

```
particleErosion
```

```
type particleErosion;
patches
(
target
);
p 20000000000;
psi 2;
K 2;
```

```
particleTracks1
```

```
type particleTracks;
trackInterval 1;
maxSamples 10000000;
resetOnWrite yes;
log yes;
```

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

A.11. g

```
FoamFile
```

```
version 2.0;
format ascii;
class uniformDimensionedVectorField;
location constant";
object g;
```

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

```
dimensions [0 1 -2 0 0 0];
value (0 0 -9.81);
```

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

A.12. momentumTransport.water

```
simulationType RAS;
```

```
RAS
```

```
// Mandatory entries
RASModel kEpsilon;
```

```
// Optional entries
```

```
turbulence on;
printCoeffs on;
```

```
// Optional model coefficients
```

```
Cmu 0.09;
C1 1.44;
C2 1.92;
C3 0.0;
sigmak 1.0;
sigmaEps 1.3;
```

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

A.13. transportProperties

```
continuousPhaseName water;
```

```
rho.water 997;
```

```
transportModel Newtonian;
nu 1e-06;
```

```
continuousPhaseName air;
```

```
rho.air 1;
```

```
transportModel Newtonian;
nu 1.48e-05;
```

```
sigma 0.07
// ***** //
```

A.14. U.water

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

```
internalField uniform (0 0 0);
```

```
boundaryField
```

```
wall
```

```
type noSlip;
```

```
inlet
```

```
type fixedValue;
value uniform (10 0 0);
```

```
target
```

```
type fixedValue;
value uniform (0 0 0);
```

```
wall1
```

```
type inletOutlet;
phi phi.water;
inletValue uniform (0 0 0);
value uniform (0 0 0);
```

```
wall2
```

```
type noSlip;
```

```
lowerWall
```

```
type noSlip;
```

```

    upperWall

type noSlip;

    front

type symmetryPlane;

```

```

// *****

```

A.15. p

```

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

internalField uniform 0;

boundaryField

wall

type fixedFluxPressure;
value internalField;

    inlet

type fixedFluxPressure;
value internalField;

    target

type fixedFluxPressure;
value internalField;

    frontAndBack

type fixedFluxPressure;
value internalField;

```

```

    wall1

    type fixedValue;
    value uniform 0;

    wall2

    type fixedValue;
    value internalField;

    lowerWall

    type fixedFluxPressure;
    value internalField;

    upperWall
    type fixedFluxPressure; value internalField;

    front

    type symmetryPlane;

    // ***** //

```

A.16. k.water

```

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

    internalField uniform 0.236;

    boundaryField

    lowerWall

    type kqRWallFunction;
    value uniform 0.236;

    upperWall

    type kqRWallFunction;
    value uniform 0.236;

    frontAndBack

```

```
type kqRWallFunction;  
value uniform 0.236;
```

wall1

```
type zeroGradient;
```

wall2

```
type zeroGradient;
```

wall

```
type kqRWallFunction;  
value uniform 0.236;
```

target

```
type kqRWallFunction;  
value uniform 0.236;
```

inlet

```
type fixedValue;  
value uniform 0.236;
```

front

```
type symmetryPlane;
```



A.17. epsilon.water

```
dimensions [0 2 -3 0 0 0 0];  
internalField uniform 38.37;  
boundaryField  
lowerWall  
type epsilonWallFunction;  
value uniform 38.37;
```

```

upperWall
type epsilonWallFunction;
value uniform 38.37;
frontAndBack
type epsilonWallFunction;
value uniform 38.37;
wall1
type zeroGradient;
wall2
type zeroGradient;
wall
type epsilonWallFunction;
value uniform 38.37;
target
type epsilonWallFunction;
value uniform 38.37;
inlet
type fixedValue;
value uniform 38.37;
front
type symmetryPlane;
// ***** //

```

A.18. nut.water

```

dimensions [0 2 -1 0 0 0 0];
internalField uniform 0;
boundaryField
lowerWall
type nutkWallFunction;
value uniform 0;
upperWall
type nutkWallFunction;
value uniform 0;
frontAndBack
type nutkWallFunction; dimensions [0 2 -1 0 0 0 0];

```

```
internalField uniform 0;
boundaryField
lowerWall
type nutkWallFunction;
value uniform 0;
upperWall
type nutkWallFunction;
value uniform 0;
frontAndBack
type nutkWallFunction;
value uniform 0;
wall1
type calculated;
value uniform 0;
wall2
type calculated;
value uniform 0;
wall
type nutkWallFunction;
value uniform 0;
target
type nutkWallFunction;
value uniform 0;
inlet
type calculated;
value uniform 0;
front
type symmetryPlane;
value uniform 0;
wall1
type calculated;
value uniform 0;
wall2
type calculated;
value uniform 0;
wall
```



```
type nutkWallFunction;  
value uniform 0;  
target  
type nutkWallFunction;  
value uniform 0;  
inlet  
type calculated;  
value uniform 0;  
front  
type symmetryPlane;
```



B | Código OpenFoam validación 25 mm

B.1. blockMeshDict

```

convertToMeters 0.01;
vertices
(
//izquierda
(0 0 0) //0
(0 -0.35 0) //1
(0 0 -0.35) //2
(0 0.35 0) //3
(0 -3.15 0) //4
(0 0 -3.15) //5
(0 3.15 0) //6
(14 0 0) //7
(14 -0.35 0) //8
(14 0 -0.35) //9
(14 0.35 0) //10
(14 -3.15 0) //11
(14 0 -3.15) //12
(14 3.15 0) //13
);
blocks
(
hex (1 2 0 0 8 9 7 7) (10 4 140) simpleGrading (1 1 ((0.8 0.65 1) (0.15 0.3 1) (0.05 0.05 1))) //0
hex (2 3 0 0 9 10 7 7) (10 4 140) simpleGrading (1 1 ((0.8 0.65 1) (0.15 0.3 1) (0.05 0.05 1)))
//1
hex (4 5 2 1 11 12 9 8) (10 32 140) simpleGrading (1 1 ((0.8 0.65 1) (0.15 0.3 1) (0.05 0.05 1)))
//2

```

```

hex (5 6 3 2 12 13 10 9) (10 32 140) simpleGrading (1 1 ((0.8 0.65 1) (0.15 0.3 1) (0.05 0.05 1)))
//3
);
edges
(
arc 1 2 (0 -0.2474873734 -0.2474873734)
arc 2 3 (0 0.2474873734 -0.2474873734)
arc 8 9 (14 -0.2474873734 -0.2474873734)
arc 9 10 (14 0.2474873734 -0.2474873734)
arc 4 5 ( 0 -2.2273863607 -2.2273863607)
arc 11 12 ( 14 -2.2273863607 -2.2273863607)
arc 5 6 ( 0 2.2273863607 -2.2273863607)
arc 12 13 ( 14 2.2273863607 -2.2273863607)
);
boundary
(
/*
frontAndBack
type wall;
faces
(
(0 3 2 1)
);
*/
inlet
type patch;
faces
(
(2 0 0 1)
(3 0 0 2)
);
front
type symmetryPlane;
faces
(
(0 1 8 7)
(3 0 7 10)

```

```
(1 4 11 8)
(6 3 10 13)
);
wall1
type wall;
faces
(
(1 2 5 4)
(3 6 5 2)
);
wall2
type wall;
faces
(
(10 7 7 9)
(9 7 7 8)
(8 11 12 9)
(10 9 12 13)
);
lowerWall
type wall;
faces
(
(4 5 12 11)
);
upperWall
type wall;
faces
(
(12 5 6 13)
);
);
```



B.2. snappyHexMesh

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

snap true;

addLayers false;

// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry
//inlet type triSurfaceMesh; file "inlet.stl";
target type triSurfaceMesh; file "target.stl";
wall type triSurfaceMesh; file "wall.stl";

//motorBike
//
//type triSurfaceMesh;
//file "inlet.stl";
//type triSurfaceMesh;
//file "target.stl";
//type triSurfaceMesh;
//file "wall.stl";
//
/*refinementBox
/*type searchableBox;
min (0.0 -0.001 -0.001);
max ( 0.0105 0.001 0.0);
/*type searchableBox;
min (0.0 0.0003 -0.0009);
max ( 0.0097 0.001 0.0);
*/
;

// Settings for the castellatedMesh generation.
castellatedMeshControls
```

```
// Refinement parameters
//
// If local number of cells is >= maxLocalCells on any processor
// switches from from refinement followed by balancing
// (current method) to (weighted) balancing before refinement.
maxLocalCells 2500000;//100000
// Overall cell limit (approximately). Refinement will stop immediately
// upon reaching this number so a refinement level might not complete.
// Note that this is the number of cells before removing the part which
// is not 'visible' from the keepPoint. The final number of cells might
// actually be a lot less.
maxGlobalCells 14000000;//2000000
// The surface refinement loop might spend lots of iterations refining just a
// few cells. This setting will cause refinement to stop if <= minimumRefine
// are selected for refinement. Note: it will at least do one iteration
// (unless the number of cells to refine is 0)
minRefinementCells 0;//10
// Allow a certain level of imbalance during refining
// (since balancing is quite expensive)
// Expressed as fraction of perfect balance (= overall number of cells /
// nProcs). 0=balance always.
maxLoadUnbalance 0.10;
// Number of buffer layers between different levels.
// 1 means normal 2:1 refinement restriction, larger means slower
// refinement.
nCellsBetweenLevels 3;
// Explicit feature edge refinement
//
// Specifies a level for any cell intersected by its edges.
// This is a featureEdgeMesh, read from constant/triSurface for now.
features
(
//
// file "motorBike.eMesh";
//level 6;
//
```

```

file "wall.eMesh";
level 0;//4 probar con otros niveles
/*
file "target.eMesh";
level 0;
file "inlet.eMesh";
level 0;
*/
);
// Surface based refinement
//
// Specifies two levels for every surface. The first is the minimum level,
// every cell intersecting a surface gets refined up to the minimum level.
// The second level is the maximum level. Cells that 'see' multiple
// intersections where the intersections make an
// angle > resolveFeatureAngle get refined up to the maximum level.
refinementSurfaces
/*
inlet
// Surface-wise min and max refinement level
level (0 0);//(2 2) probar con mas
// Optional specification of patch type (default is wall). No
// constraint types (cyclic, symmetry) etc. are allowed.
patchInfo
type patch;
//inGroups (motorBikeGroup);
*/
wall
// Surface-wise min and max refinement level
level (0 0);
// Optional specification of patch type (default is wall). No
// constraint types (cyclic, symmetry) etc. are allowed.
patchInfo
type wall;
//inGroups (motorBikeGroup);
target

```

```

// Surface-wise min and max refinement level
level (0 0);/(3 3)

// Optional specification of patch type (default is wall). No
// constraint types (cyclic, symmetry) etc. are allowed.
patchInfo
type wall;
//inGroups (motorBikeGroup);
// Resolve sharp angles
resolveFeatureAngle 30;
// Region-wise refinement
//
// Specifies refinement level for cells in relation to a surface. One of
// three modes
// - distance. 'levels' specifies per distance to the surface the
// wanted refinement level. The distances need to be specified in
// descending order.
// - inside. 'levels' is only one entry and only the level is used. All
// cells inside the surface get refined up to the level. The surface
// needs to be closed for this to be possible.
// - outside. Same but cells outside.
refinementRegions
/*refinementBox
mode inside;
levels ((1E15 1));/levels ((1E15 1))
gapLevel (6 6 6)
gapMode inside;
*/
refinementBox
mode inside;
levels ((1E15 1));/levels ((1E15 1))
gapLevel (6 6 6)
gapMode inside;
// Mesh selection
//
// After refinement patches get added for all refinementSurfaces and
// all cells intersecting the surfaces get put into these patches. The

```

```

// section reachable from the locationInMesh is kept.
// NOTE: This point should never be on a face, always inside a cell, even
// after refinement.
locationInMesh (0.003 0.0019 0);
// Whether any faceZones (as specified in the refinementSurfaces)
// are only on the boundary of corresponding cellZones or also allow
// free-standing zone faces. Not used if there are no faceZones.
allowFreeStandingZoneFaces true;
// Settings for the snapping.
snapControls
// - Number of patch smoothing iterations before finding correspondence
// to surface
nSmoothPatch 3;//3
// - Relative distance for points to be attracted by surface feature point
// or edge. True distance is this factor times local
// maximum edge length.
tolerance 2.0;//0.25
// - Number of mesh displacement relaxation iterations.
nSolveIter 30;//30
// - Maximum number of snapping relaxation iterations. Should stop
// before upon reaching a correct mesh.
nRelaxIter 5;//5
// Feature snapping
// - Number of feature edge snapping iterations.
// Leave out altogether to disable.
nFeatureSnapIter 10;
// - Detect (geometric only) features by sampling the surface
// (default=false).
implicitFeatureSnap false;
// - Use castellatedMeshControls::features (default = true)
explicitFeatureSnap true;
// - Detect points on multiple surfaces (only for explicitFeatureSnap)
multiRegionFeatureSnap false;
// Settings for the layer addition.
addLayersControls
// Are the thickness parameters below relative to the undistorted

```

```
// size of the refined cell outside layer (true) or absolute sizes (false).
relativeSizes true;
// Per final patch (so not geometry!) the layer information
layers
"(lowerWall|wall).*"
nSurfaceLayers 1;//1
// Expansion factor for layer mesh
expansionRatio 1.0;//1.0
// Wanted thickness of final added cell layer. If multiple layers
// is the thickness of the layer furthest away from the wall.
// Relative to undistorted size of cell outside layer.
// See relativeSizes parameter.
finalLayerThickness 0.3;//0.3
// Minimum thickness of cell layer. If for any reason layer
// cannot be above minThickness do not add layer.
// Relative to undistorted size of cell outside layer.
minThickness 0.1;//0.1
// If points get not extruded do nGrow layers of connected faces that are
// also not grown. This helps convergence of the layer addition process
// close to features.
// Note: changed(corrected) w.r.t 17x! (didn't do anything in 17x)
nGrow 0;
// Advanced settings
// When not to extrude surface. 0 is flat surface, 90 is when two faces
// are perpendicular
featureAngle 60;
// At non-patched sides allow mesh to slip if extrusion direction makes
// angle larger than slipFeatureAngle.
slipFeatureAngle 30;
// Maximum number of snapping relaxation iterations. Should stop
// before upon reaching a correct mesh.
nRelaxIter 3;
// Number of smoothing iterations of surface normals
nSmoothSurfaceNormals 1;
// Number of smoothing iterations of interior mesh movement direction
nSmoothNormals 3;
```

```

// Smooth layer thickness over surface patches
nSmoothThickness 10;

// Stop layer growth on highly warped cells
maxFaceThicknessRatio 0.5;

// Reduce layer growth where ratio thickness to medial
// distance is large
maxThicknessToMedialRatio 0.3;

// Angle used to pick up medial axis points
// Note: changed(corrected) w.r.t 17x! 90 degrees corresponds to 130 in 17x.
minMedianAxisAngle 90;

// Create buffer region for new layer terminations
nBufferCellsNoExtrude 0;

// Overall max number of layer addition iterations. The mesher will exit
// if it reaches this number of iterations; possibly with an illegal
// mesh.
nLayerIter 50;

// Generic mesh quality settings. At any undoable phase these determine
// where to undo.
meshQualityControls
include "meshQualityDict"

// Advanced
// Write flags
writeFlags
(
    scalarLevels
    layerSets
    layerFields // write volScalarField for layer coverage
);

// Merge tolerance. Is fraction of overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1e-6;

```

B.3. controlDict

```

application DPMFoam;
startFrom latestTime;

```

```

startTime 0;
stopAt endTime;
endTime 1;//0.5
deltaT 0.1;//0.00001
writeControl runTime;
writeInterval 0.1;//0.00001
purgeWrite 10;//2000
writeFormat binary;
writePrecision 6;
writeCompression off;
timeFormat general;
timePrecision 6;
runTimeModifiable yes;
adjustTimeStep yes;
maxCo 1;
maxDeltaT 0.1;

```

B.4. fvSchemes

```

ddtSchemes
default Euler;
gradSchemes
default Gauss linear;
divSchemes
default none;
div(alphaPhi.water,k.water) bounded Gauss upwind;
div(alphaPhi.water,epsilon.water) bounded Gauss upwind;
//div(phi,omega) bounded Gauss upwind;
div(alphaPhi.water,U.water) Gauss linearUpwindV unlimited;
div((((alpha.water*nuEff.water)*dev2(T(grad(U.water)))))) Gauss linear;
laplacianSchemes
default Gauss linear corrected;
interpolationSchemes
default linear;
snGradSchemes
default corrected;

```

B.5. fvSolutions

```

solvers
"(p|kinematicCloud:theta)"
solver GAMG;
tolerance 1e-06;//1e-04
relTol 0.01;
smoother GaussSeidel;
"(p|kinematicCloud:theta)Final"
solver GAMG;
tolerance 1e-04;//1e-06
relTol 0;
smoother GaussSeidel;
"(U.water|k.wather|epsilon.wather)"
solver smoothSolver;
smoother symGaussSeidel;
tolerance 1e-04;//1e-05
relTol 0.1;
"(U.water|k.wather|epsilon.wather)Final"
solver smoothSolver;
smoother symGaussSeidel;
tolerance 1e-04;//1e-05
relTol 0;
PIMPLE
nOuterCorrectors 1;
nCorrectors 2;
momentumPredictor yes;
nNonOrthogonalCorrectors 0;
pRefCell 0;
pRefValue 0;
relaxationFactors

```

B.6. mapFieldsDict

```

// List of pairs of source/target patches for mapping
patchMap
(

```

```

);
// List of target patches cutting the source domain (these need to be
// handled specially e.g. interpolated from internal values)
cuttingPatches
(
);

```

B.7. meshQualityDict

```

// Include defaults parameters from master dictionary
includeEtc caseDicts/mesh/generation/meshQualityDict"
//- minFaceWeight (0 -> 0.5)
minFaceWeight 0.05;//0.02
minVol 1e-13;//1e-13
maxBoundarySkewness 4;//20-4
maxInternalSkewness 4;
maxConcave 80;
minVolRatio 0.01;
minTringleTwist -1;
//minVolCollapseRatio -0.000001;
minDeterminant 0.001;

```

B.8. setFieldsDict

```

defaultFieldValues
(
volScalarFieldValue alpha.air 1
//volScalarFieldValue alpha.water 0
);
regions
(
);

```

B.9. surfaceFeaturesDict

```
surfaces ("wall.stl");
// Identify a feature when angle between faces < includedAngle
includedAngle 150;//150
subsetFeatures
// Keep nonManifold edges (edges with >2 connected faces)
nonManifoldEdges no;//no
// Keep open edges (edges with 1 connected face)
openEdges no;//yes
```

B.10. kinematicCloudProperties

```
solution//particle solution settings
active true;//true for particle adding-Activate or deactivate the particle cloud
coupled false;//true for the influence of parcels to the continuous phase-Enable or disable phase coupling
transient yes;//true for transient simulation-Transient or steady-state solution
cellValueSourceCorrection on;//correction of cell values within the Lagrangian time step -//Enable/disable correction of momentum transferred to the Eulerian phase
maxCo 1; //maximum courant number
calcFrequency 0.001;
maxTrackTime 100;
interpolationSchemes//interpolation schemes for continuous fields
rho.water cell;
U.water cellPoint;
mu.water cell;
alpha.water cell;
averagingMethod basic;//dual for dual-mesh Lagrangian averaging procedure - averaging over the cells and dual (tetrahedral) volumes
integrationSchemes//definition of integration schemes of velocity field U
U Euler;
sourceTerms//scheme for source term in momentum equation of continuous phase, relevant if coupled is activated
schemes
U semiImplicit 1;//1//explicit or semiImplicit, relaxCoeff for each of the fields
constantProperties//density and max fraction of particles
rho0 2206;//densidad de la particula kg/m3
```

```

alphaMax 0.9;//0.99
poissonsRatio 0.35;//ratio of traverse to axial strain for your particles (dimensionless)
youngsModulus 1e8;// 1e8 constant of proportionality relating the deformation to the applied stress
for your particles (N/m2)

subModels //particular particle properties definition-Specify applicable forces on the particles
particleForces//particle drag modelling
/*WenYuDrag
alphac alpha.water;
*/
sphereDrag;
//gravity;
injectionModels
/*
alpha=0.1
U=20
A=0.012
Q=U*A
d=650e-6
v=(4/3)*pi*(d/2)3
n=1
rate=alpha*Q/(v*n)
*/
model1//method of particle injection (model1 - user name of model defined below)
type patchInjection;//patchInjection for injection at the patch
parcelBasisType fixed;//parcel basis enumeration/options of how to represent your particles. besides
'fixed' they can be represented by 'number' or 'mass'
patchName inlet;//name of the patch for particle injection
U0 (0 0 0);
parcelsPerSecond 53308;//1390885 //number of particle per second
meanParticleDiameter 0.00025;//0.00025
nParticle 1;//number of particles to be injected
//massFlowRate 0.0001;
//meanParticleDiameter 0.00025
/*sizeDistribution
type normal;
normalDistribution
expectation 650e-6;

```

```

    variance 25e-6;
    minValue 500e-6;
    maxValue 800e-6;
    */
    sizeDistribution //size of particles constant (fixedValue), or e.g. normal distribution (normal)
    type fixedValue;//type of size distribution
    fixedValueDistribution
    value 0.00025;//0.00025
    flowRateProfile constant 1;//distribution of injected particle with respect to the duration, constant 1
means uniform
    massTotal 0;//total mass to inject
    SOI 0;//start time of injection
    duration 2000;//60
    dispersionModel none;//no dispersion model is implemented yet
    patchInteractionModel localInteraction;//localInteraction for patch-by-patch interaction specifica-
tion
    localInteractionCoeffs//patch interaction definition
    patches
    (
    /*
    frontAndBack
    type escape;//rebound for simple rebound, e - elasticity coefficient ( "percentage" dissipation in normal
direction) , mu - restitution coefficient (dissipation in tangential direction)
    e 1;
    mu 0;
    */
    front
    type none;//symmetryPlane o none
    "wall|lowerWall|upperWall"
    type rebound;
    e 1;
    mu 0;
    target
    type escape;//rebound
    e 0;
    mu 0;
    inlet|wall1|wall2"

```

```

type escape;//escape for outlet patches
);
heatTransferModel none;//none for no heat transfer modelling
surfaceFilmModel none;//none for no surfaceFilmModel-Surface film model for dripping and film
interaction //absorb, bounce, splash
collisionModel none;//pairCollision
/*pairCollisionCoeffs
maxInteractionDistance 60e-6;//0.0025
writeReferredParticleCloud no;
pairModel pairSpringSliderDashpot;
pairSpringSliderDashpotCoeffs
useEquivalentSize no;
alpha 0.02;//function of coefficient of restitution
b 1.5;//Spring power: b = 1 (linear) or b = 3/2 (Hertzian theory)
mu 0.10; //Coefficient of friction
cohesionEnergyDensity 0;
collisionResolutionSteps 12;
;
wallModel wallSpringSliderDashpot;
wallSpringSliderDashpotCoeffs
useEquivalentSize no;
collisionResolutionSteps 12;
youngsModulus 1e8;
poissonsRatio 0.23;
alpha 0.01;
b 1.5;
mu 0.09;
cohesionEnergyDensity 0;
;
U U.water;
*/
packingModel none;//implicit model for applying an inter-particle stress to the particles
dampingModel relaxation;//no model for collision damping time of particle velocity fluctuation
relaxationCoeffs
timeScaleModel
type nonEquilibrium;
alphaPacked 0.67;//0.58

```

```

    e 0.9;
    isotropyModel stochastic;//stochastic return-to-isotropy model - adding collision term that causes
velocity distributions to relax to isotropic Gaussian distributions
    stochasticCoeffs
    timeScaleModel
    type isotropic;//isotropic model for the time scale over which the velocity field of a dispersed phase
tends towards an isotropic distribution
    alphaPacked 0.67;//0.58
    e 0.9;
    stochasticCollisionModel none;//Particle collision: spring, slider+dash-pot
    radiation off;//no radiation model
    // Use cloudFunctions to record particle tracks, calculate particle erosion
    cloudFunctions
    patchPostProcessing
    type patchPostProcessing;
    maxStoredParcels 1000000;
    patches
    (
    target
    );
    particleErosion
    type particleErosion;
    patches
    (
    target
    );
    p 2000000000;
    psi 2;
    K 2;
    particleTracks1
    type particleTracks;
    trackInterval 1;
    maxSamples 10000000;
    resetOnWrite yes;
    log yes;
    /*particleErosion
    functionObjectLibs ("libcloudFunctionObjects.so");

```

```

enabled true;
outputControl outputTime;
log true;
valueOutput true;
//yield stress for aluminium = 11000000 Pa or 11 MPa
p 11000000;
//Ratio of the depth of contact to the depth of cut (default value = 2 )
psi 2;
//Ratio of vertical to horizontal force components (2 for angular abrasive grains)
K 2;
patches
(
moving-wall
);
*/

```

B.11. g

```

FoamFile
version 2.0;
format ascii;
class uniformDimensionedVectorField;
location constant";
object g;
// ****
dimensions [0 1 -2 0 0 0 0];
value (0 0 -9.81);

```

B.12. momentumTransport.water

```

simulationType RAS;
RAS
// Mandatory entries
RASModel kEpsilon;
// Optional entries
turbulence on;
printCoeffs on;

```

```
// Optional model coefficients
/* Cmu 0.09;
C1 1.44;
C2 1.92;
C3 0.0;
sigmak 1.0;
sigmaEps 1.3;*/
```

B.13. transportProperties

```
continuousPhaseName water;
rho.water 997;//997
transportModel Newtonian;
nu 1e-06;//1e-06
continuousPhaseName air;
rho.air 1.225;//1.225
transportModel Newtonian;
nu 1.48e-05;//1.48e-05
sigma 0.07//0.07
```

B.14. U.water

```
dimensions [0 1 -1 0 0 0 0];
internalField uniform (0 0 0);
boundaryField
{
    wall
    {
        type noSlip;
    }
    inlet
    {
        type fixedValue;
        value uniform (3.8 0 0);
    }
    target
    {
        type fixedValue;
        value uniform (0 0 0);
    }
    /*
    frontAndBack
    {
        type fixedValue;
        value uniform (0 0 0);
    }
}
```

```

*/
wall1
type inletOutlet;
phi phi.water;
inletValue uniform (0 0 0);
value uniform (0 0 0);
/*
wall2
type inletOutlet;
phi phi.water;
inletValue uniform (0 0 0);
value uniform (0 0 0);
*/
wall2
type noSlip;
lowerWall
type noSlip;
upperWall
type noSlip;
front
type symmetryPlane;

```

B.15. p

```

dimensions [0 2 -2 0 0 0 0];
internalField uniform 100;
boundaryField
wall
type fixedFluxPressure;
value internalField;
inlet
type fixedValue;
value uniform 140;
target
type fixedFluxPressure;
value internalField;

```

```
frontAndBack
type fixedFluxPressure;
value internalField;
wall1
type fixedValue;
value uniform 100;
wall2
type fixedValue;
value internalField;
lowerWall
type fixedFluxPressure;
value internalField;
upperWall
type fixedFluxPressure;
value internalField;
front
type symmetryPlane;
```

B.16. k.water

```
dimensions [0 2 -2 0 0 0 0];
internalField uniform 0.236;
boundaryField
lowerWall
type kqRWallFunction;
value uniform 0.236;
upperWall
type kqRWallFunction;
value uniform 0.236;
frontAndBack
type kqRWallFunction;
value uniform 0.236;
wall1
type zeroGradient;
wall2
type zeroGradient;
```

```
wall
type kqRWallFunction;
value uniform 0.236;
target
type kqRWallFunction;
value uniform 0.236;
inlet
type fixedValue;
value uniform 0.236;
front
type symmetryPlane;
```

B.17. epsilon.water

```
dimensions [0 2 -3 0 0 0 0];
internalField uniform 38.37;//38.37
boundaryField
lowerWall
type epsilonWallFunction;
value uniform 38.37;
upperWall
type epsilonWallFunction;
value uniform 38.37;
frontAndBack
type epsilonWallFunction;
value uniform 38.37;
wall1
type zeroGradient;
wall2
type zeroGradient;
wall
type epsilonWallFunction;
value uniform 38.37;
target
type epsilonWallFunction;
value uniform 38.37;
```

```
inlet
type fixedValue;
value uniform 38.37;
front
type symmetryPlane;
```

B.18. nut.water

```
dimensions [0 2 -1 0 0 0 0];
internalField uniform 0;
boundaryField
lowerWall
type nutkWallFunction;
value uniform 0;
upperWall
type nutkWallFunction;
value uniform 0;
frontAndBack
type nutkWallFunction;
value uniform 0;
wall1
type calculated;
value uniform 0;
wall2
type calculated;
value uniform 0;
wall
type nutkWallFunction;
value uniform 0;
target
type nutkWallFunction;
value uniform 0;
inlet
type calculated;
value uniform 0;
front
```

type symmetryPlane;



C | Código OpenFoam erosión pala Pelton

C.1. blockMeshDict

```

convertToMeters 3;
vertices
(
//izquierda
(-2 0 0) //0
(-2 -0.35 0) //1
(-2 0 -0.35) //2
(-2 0.35 0) //3
(-2 -3.15 0) //4
(-2 0 -3.15) //5
(-2 3.15 0) //6
(13 0 0) //7
(13 -0.35 0) //8
(13 0 -0.35) //9
(13 0.35 0) //10
(13 -3.15 0) //11
(13 0 -3.15) //12
(13 3.15 0) //13
);
blocks
(
hex (1 2 0 0 8 9 7 7) (10 4 130) simpleGrading (1 1 1) //0
hex (2 3 0 0 9 10 7 7) (10 4 130) simpleGrading (1 1 1) //1
hex (4 5 2 1 11 12 9 8) (10 32 130) simpleGrading (1 1 1) //2

```

```

hex (5 6 3 2 12 13 10 9) (10 32 130) simpleGrading (1 1 1) //3
/*hex (1 2 0 0 8 9 7 7) (16 12 200) simpleGrading (1 1 2) //0
hex (2 3 0 0 9 10 7 7) (16 12 200) simpleGrading (1 1 2) //1
hex (4 5 2 1 11 12 9 8) (16 80 200) simpleGrading (1 0.5 2) //2
hex (5 6 3 2 12 13 10 9) (16 80 200) simpleGrading (1 0.5 2) //3*/
);
edges
(
arc 1 2 (-2 -0.2474873734 -0.2474873734)
arc 2 3 (-2 0.2474873734 -0.2474873734)
arc 8 9 (13 -0.2474873734 -0.2474873734)
arc 9 10 (13 0.2474873734 -0.2474873734)
arc 4 5 (-2 -2.2273863607 -2.2273863607)
arc 11 12 (13 -2.2273863607 -2.2273863607)
arc 5 6 (-2 2.2273863607 -2.2273863607)
arc 12 13 (13 2.2273863607 -2.2273863607)
);
boundary
(
/*
frontAndBack
type wall;
faces
(
(0 3 2 1)
);
*/
inlet
type patch;
faces
(
(2 0 0 1)
(3 0 0 2)
);
front
type symmetryPlane;

```

```
faces
(
(0 1 8 7)
(3 0 7 10)
(1 4 11 8)
(6 3 10 13)
);
wall1
type wall;
faces
(
(1 2 5 4)
(3 6 5 2)
);
wall2
type wall;
faces
(
(10 7 7 9)
(9 7 7 8)
(8 11 12 9)
(10 9 12 13)
);
lowerWall
type wall;
faces
(
(4 5 12 11)
);
upperWall
type wall;
faces
(
(12 5 6 13)
);
);
```



C.2. snappyHexMesh

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

snap true;

addLayers false;

// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry

//inlet type triSurfaceMesh; file "inlet.stl";
target type triSurfaceMesh; file "target.stl";
wall type triSurfaceMesh; file "wall.stl";

//motorBike

//

//type triSurfaceMesh;
//file "inlet.stl";

//type triSurfaceMesh;
//file "target.stl";

//type triSurfaceMesh;
//file "wall.stl";

//

/*refinementBox
/*type searchableBox;
min (0.0 -0.001 -0.001);
max ( 0.0105 0.001 0.0);
/*type searchableBox;
min (0.0 0.0003 -0.0009);
max ( 0.0097 0.001 0.0);
*/
;

// Settings for the castellatedMesh generation.
castellatedMeshControls
```

```

// Refinement parameters
//
// If local number of cells is >= maxLocalCells on any processor
// switches from from refinement followed by balancing
// (current method) to (weighted) balancing before refinement.
maxLocalCells 2500000;//100000
// Overall cell limit (approximately). Refinement will stop immediately
// upon reaching this number so a refinement level might not complete.
// Note that this is the number of cells before removing the part which
// is not 'visible' from the keepPoint. The final number of cells might
// actually be a lot less.
maxGlobalCells 14000000;//2000000
// The surface refinement loop might spend lots of iterations refining just a
// few cells. This setting will cause refinement to stop if <= minimumRefine
// are selected for refinement. Note: it will at least do one iteration
// (unless the number of cells to refine is 0)
minRefinementCells 0;//10
// Allow a certain level of imbalance during refining
// (since balancing is quite expensive)
// Expressed as fraction of perfect balance (= overall number of cells /
// nProcs). 0=balance always.
maxLoadUnbalance 0.10;
// Number of buffer layers between different levels.
// 1 means normal 2:1 refinement restriction, larger means slower
// refinement.
nCellsBetweenLevels 3;
// Explicit feature edge refinement
//
// Specifies a level for any cell intersected by its edges.
// This is a featureEdgeMesh, read from constant/triSurface for now.
features
(
//
// file "motorBike.eMesh";
//level 6;
//

```

```

/*
file "wall.eMesh";
level 0;//4 probar con otros niveles
*/
/*
file "target.eMesh";
level 0;
file "inlet.eMesh";
level 0;
*/
);
// Surface based refinement
//
// Specifies two levels for every surface. The first is the minimum level,
// every cell intersecting a surface gets refined up to the minimum level.
// The second level is the maximum level. Cells that 'see' multiple
// intersections where the intersections make an
// angle > resolveFeatureAngle get refined up to the maximum level.
refinementSurfaces
/*
inlet
// Surface-wise min and max refinement level
level (0 0);//(2 2) probar con mas
// Optional specification of patch type (default is wall). No
// constraint types (cyclic, symmetry) etc. are allowed.
patchInfo
type patch;
//inGroups (motorBikeGroup);
*/
wall
// Surface-wise min and max refinement level
level (0 0);
// Optional specification of patch type (default is wall). No
// constraint types (cyclic, symmetry) etc. are allowed.
patchInfo
type wall;

```

```

//inGroups (motorBikeGroup);
target
// Surface-wise min and max refinement level
level (0 0);/(3 3)
// Optional specification of patch type (default is wall). No
// constraint types (cyclic, symmetry) etc. are allowed.
patchInfo
type wall;
//inGroups (motorBikeGroup);
// Resolve sharp angles
resolveFeatureAngle 30;
// Region-wise refinement
//
// Specifies refinement level for cells in relation to a surface. One of
// three modes
// - distance. 'levels' specifies per distance to the surface the
// wanted refinement level. The distances need to be specified in
// descending order.
// - inside. 'levels' is only one entry and only the level is used. All
// cells inside the surface get refined up to the level. The surface
// needs to be closed for this to be possible.
// - outside. Same but cells outside.
refinementRegions
/*refinementBox
mode inside;
levels ((1E15 1));/levels ((1E15 1))
gapLevel (6 6 6)
gapMode inside;
*/
refinementBox
mode inside;
levels ((1E15 1));/levels ((1E15 1))
gapLevel (6 6 6)
gapMode inside;
// Mesh selection
//

```

```

// After refinement patches get added for all refinementSurfaces and
// all cells intersecting the surfaces get put into these patches. The
// section reachable from the locationInMesh is kept.
// NOTE: This point should never be on a face, always inside a cell, even
// after refinement.
locationInMesh (0.003 0 -0.1);
// Whether any faceZones (as specified in the refinementSurfaces)
// are only on the boundary of corresponding cellZones or also allow
// free-standing zone faces. Not used if there are no faceZones.
allowFreeStandingZoneFaces true;
// Settings for the snapping.
snapControls
//- Number of patch smoothing iterations before finding correspondence
// to surface
nSmoothPatch 3;//3
//- Relative distance for points to be attracted by surface feature point
// or edge. True distance is this factor times local
// maximum edge length.
tolerance 2.0;//0.25
//- Number of mesh displacement relaxation iterations.
nSolveIter 30;//30
//- Maximum number of snapping relaxation iterations. Should stop
// before upon reaching a correct mesh.
nRelaxIter 5;//5
// Feature snapping
//- Number of feature edge snapping iterations.
// Leave out altogether to disable.
nFeatureSnapIter 10;
//- Detect (geometric only) features by sampling the surface
// (default=false).
implicitFeatureSnap false;
//- Use castellatedMeshControls::features (default = true)
explicitFeatureSnap true;
//- Detect points on multiple surfaces (only for explicitFeatureSnap)
multiRegionFeatureSnap false;
// Settings for the layer addition.

```

```

addLayersControls
// Are the thickness parameters below relative to the undistorted
// size of the refined cell outside layer (true) or absolute sizes (false).
relativeSizes true;
// Per final patch (so not geometry!) the layer information
layers
"(lowerWall|wall).*"
nSurfaceLayers 1;//1
// Expansion factor for layer mesh
expansionRatio 1.0;//1.0
// Wanted thickness of final added cell layer. If multiple layers
// is the thickness of the layer furthest away from the wall.
// Relative to undistorted size of cell outside layer.
// See relativeSizes parameter.
finalLayerThickness 0.3;//0.3
// Minimum thickness of cell layer. If for any reason layer
// cannot be above minThickness do not add layer.
// Relative to undistorted size of cell outside layer.
minThickness 0.1;//0.1
// If points get not extruded do nGrow layers of connected faces that are
// also not grown. This helps convergence of the layer addition process
// close to features.
// Note: changed(corrected) w.r.t 17x! (didn't do anything in 17x)
nGrow 0;
// Advanced settings
// When not to extrude surface. 0 is flat surface, 90 is when two faces
// are perpendicular
featureAngle 60;
// At non-patched sides allow mesh to slip if extrusion direction makes
// angle larger than slipFeatureAngle.
slipFeatureAngle 30;
// Maximum number of snapping relaxation iterations. Should stop
// before upon reaching a correct mesh.
nRelaxIter 3;
// Number of smoothing iterations of surface normals
nSmoothSurfaceNormals 1;

```

```

// Number of smoothing iterations of interior mesh movement direction
nSmoothNormals 3;

// Smooth layer thickness over surface patches
nSmoothThickness 10;

// Stop layer growth on highly warped cells
maxFaceThicknessRatio 0.5;

// Reduce layer growth where ratio thickness to medial
// distance is large
maxThicknessToMedialRatio 0.3;

// Angle used to pick up medial axis points
// Note: changed(corrected) w.r.t 17x! 90 degrees corresponds to 130 in 17x.
minMedianAxisAngle 90;

// Create buffer region for new layer terminations
nBufferCellsNoExtrude 0;

// Overall max number of layer addition iterations. The mesher will exit
// if it reaches this number of iterations; possibly with an illegal
// mesh.
nLayerIter 50;

// Generic mesh quality settings. At any undoable phase these determine
// where to undo.
meshQualityControls
include "meshQualityDict"

// Advanced
// Write flags
writeFlags
(
    scalarLevels
    layerSets
    layerFields // write volScalarField for layer coverage
);

// Merge tolerance. Is fraction of overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1e-6;

```

C.3. controlDict

```

application DPMFoam;
startFrom latestTime;
startTime 0;
stopAt endTime;
endTime 10800;//5
deltaT 1;//1.8e-5
writeControl runTime;
writeInterval 1;
purgeWrite 5;
writeFormat binary;
writePrecision 6;
writeCompression off;
timeFormat general;
timePrecision 6;
runTimeModifiable yes;
adjustTimeStep yes;
maxCo 1;
maxDeltaT 0.1;

```

C.4. fvSchemes

```

ddtSchemes
default Euler;

gradSchemes
default Gauss linear;

divSchemes
default none;
div(alphaPhi.water,k.water) bounded Gauss upwind;
div(alphaPhi.water,epsilon.water) bounded Gauss upwind;
//div(phi,omega) bounded Gauss upwind;
div(alphaPhi.water,U.water) Gauss linearUpwindV unlimited;
div((((alpha.water*nuEff.water)*dev2(T(grad(U.water)))))) Gauss linear;

laplacianSchemes
default Gauss linear corrected;

interpolationSchemes

```

```

default linear;
snGradSchemes
default corrected;

```

C.5. fvSolutions

```

solvers
"(p|kinematicCloud:theta)"
solver GAMG;
tolerance 1e-06;//1e-04
relTol 0.01;
smoother GaussSeidel;
"(p|kinematicCloud:theta)Final"
solver GAMG;
tolerance 1e-04;//1e-06
relTol 0;
smoother GaussSeidel;
"(U.water|k.wather|epsilon.wather)"
solver smoothSolver;
smoother symGaussSeidel;
tolerance 1e-04;//1e-05
relTol 0.1;
"(U.water|k.wather|epsilon.wather)Final"
solver smoothSolver;
smoother symGaussSeidel;
tolerance 1e-04;//1e-05
relTol 0;
PIMPLE
nOuterCorrectors 1;
nCorrectors 2;
momentumPredictor yes;
nNonOrthogonalCorrectors 0;
pRefCell 0;
pRefValue 0;
relaxationFactors

```

C.6. mapFieldsDict

```
// List of pairs of source/target patches for mapping
patchMap
(
);
// List of target patches cutting the source domain (these need to be
// handled specially e.g. interpolated from internal values)
cuttingPatches
(
);
```

C.7. meshQualityDict

```
// Include defaults parameters from master dictionary
includeEtc caseDicts/mesh/generation/meshQualityDict"
//- minFaceWeight (0 -> 0.5)
minFaceWeight 0.05;//0.02
minVol 1e-13;//1e-13
maxBoundarySkewness 4;//20-4
maxInternalSkewness 4;
maxConcave 80;
minVolRatio 0.01;
minTringleTwist -1;
//minVolCollapseRatio -0.000001;
minDeterminant 0.001;
```

C.8. setFieldsDict

```
defaultFieldValues
(
volScalarFieldValue alpha.air 1
volScalarFieldValue alpha.water 0
);
regions
(
/*boxToCell
```

```

box (0 0 -0.1) (0.15 0.701 0.1);
fieldValues
(
volScalarFieldValue alpha.air 0
volScalarFieldValue alpha.water 1
);
*/
);

```

C.9. surfaceFeaturesDict

```

surfaces ("wall.stl");
// Identify a feature when angle between faces < includedAngle
includedAngle 150;//150
subsetFeatures
// Keep nonManifold edges (edges with >2 connected faces)
nonManifoldEdges no;//no
// Keep open edges (edges with 1 connected face)
openEdges no;//yes

```

C.10. kinematicCloudProperties

```

solution//particle solution settings
active true;//true for particle adding-Activate or deactivate the particle cloud
coupled false;//true for the influence of parcels to the continuous phase-Enable or disable phase
coupling
transient yes;//true for transient simulation-Transient or steady-state solution
cellValueSourceCorrection on;//correction of cell values within the Lagrangian time step -//Enable/disable
correction of momentum transferred to the Eulerian phase
maxCo 1; //maximum courant number
calcFrequency 0.001;
maxTrackTime 100;
interpolationSchemes//interpolation schemes for continuous fields
rho.water cell;
U.water cellPoint;
mu.water cell;
alpha.water cell;

```

```

    averagingMethod basic;//dual for dual-mesh Lagrangian averaging procedure - averaging over the
    cells and dual (tetrahedral) volumes

    integrationSchemes//definition of integration schemes of velocity field U
    U Euler;

    sourceTerms//scheme for source term in momentum equation of continuous phase, relevant if coupled
    is activated

    schemes
    U semiImplicit 1;//1//explicit or semiImplicit, relaxCoeff for each of the fields
    constantProperties//density and max fraction of particles
    rho0 1442;//densidad de la particula kg/m3
    alphaMax 0.9;//0.99
    poissonsRatio 0.35;//ratio of traverse to axial strain for your particles (dimensionless)
    youngsModulus 1e8;// 1e8 constant of proportionality relating the deformation to the applied stress
    for your particles (N/m2)

    subModels //particular particle properties definition-Specify applicable forces on the particles
    particleForces//particle drag modelling
    /*WenYuDrag
    alphac alpha.water;
    */
    sphereDrag;
    gravity;
    injectionModels
    /*
    alpha=0.1
    U=20
    A=0.012
    Q=U*A
    d=650e-6
    v=(4/3)*pi*(d/2)3
    n=1
    rate=alpha*Q/(v*n)
    */
    model1//method of particle injection (model1 - user name of model defined below)
    type patchInjection;//patchInjection for injection at the patch
    parcelBasisType fixed;//parcel basis enumeration/options of how to represent your particles. besides
    'fixed' they can be represented by 'number' or 'mass'
    patchName inlet;//name of the patch for particle injection
    U0 (32.11 0 0);

```

```

parcelsPerSecond 233;//1390885 //number of particle per second
meanParticleDiameter 0.00025;//0.00025
nParticle 1;//number of particles to be injected
//massFlowRate 0.0001;
//meanParticleDiameter 0.00025
/*sizeDistribution
type normal;
normalDistribution
expectation 650e-6;
variance 25e-6;
minValue 500e-6;
maxValue 800e-6;
*/
sizeDistribution //size of particles constant (fixedValue), or e.g. normal distribution (normal)
type fixedValue;//type of size distribution
fixedValueDistribution
value 0.00025;//0.00025
flowRateProfile constant 1;//distribution of injected particle with respect to the duration, constant 1
means uniform
massTotal 0;//total mass to inject
SOI 0;//start time of injection
duration 2000;//60
dispersionModel none;//no dispersion model is implemented yet
patchInteractionModel localInteraction;//localInteraction for patch-by-patch interaction specifica-
tion
localInteractionCoeffs//patch interaction definition
patches
(
/*
frontAndBack
type escape;//rebound for simple rebound, e - elasticity coefficient ( "percentage" dissipation in normal
direction) , mu - restitution coefficient (dissipation in tangential direction)
e 1;
mu 0;
*/
front
type none;//symmetryPlane o none

```

```

wall
type rebound;
e 1;
mu 0;
/*target
type rebound;//rebound
e 0;
mu 0;
*/
inlet|wall1|wall2|lowerWall|upperWall|target"
type escape;//escape for outlet patches
);
heatTransferModel none;//none for no heat transfer modelling
surfaceFilmModel none;//none for no surfaceFilmModel-Surface film model for dripping and film
interaction //absorb, bounce, splash
collisionModel none;//pairCollision
/*pairCollisionCoeffs
maxInteractionDistance 60e-6;//0.0025
writeReferredParticleCloud no;
pairModel pairSpringSliderDashpot;
pairSpringSliderDashpotCoeffs
useEquivalentSize no;
alpha 0.02;//function of coefficient of restitution
b 1.5;//Spring power: b = 1 (linear) or b = 3/2 (Hertzian theory)
mu 0.10; //Coefficient of friction
cohesionEnergyDensity 0;
collisionResolutionSteps 12;
;
wallModel wallSpringSliderDashpot;
wallSpringSliderDashpotCoeffs
useEquivalentSize no;
collisionResolutionSteps 12;
youngsModulus 1e8;
poissonsRatio 0.23;
alpha 0.01;
b 1.5;
mu 0.09;

```

```

    cohesionEnergyDensity 0;
    ;
    U U.water;
    */
    packingModel none;//implicit model for applying an inter-particle stress to the particles
    dampingModel relaxation;//no model for collision damping time of particle velocity fluctuation
    relaxationCoeffs
    timeScaleModel
    type nonEquilibrium;
    alphaPacked 0.67;//0.58
    e 0.9;
    isotropyModel stochastic;//stochastic return-to-isotropy model - adding collision term that causes
    velocity distributions to relax to isotropic Gaussian distributions
    stochasticCoeffs
    timeScaleModel
    type isotropic;//isotropic model for the time scale over which the velocity field of a dispersed phase
    tends towards an isotropic distribution
    alphaPacked 0.67;//0.58
    e 0.9;
    stochasticCollisionModel none;//Particle collision: spring, slider+dash-pot
    radiation off;//no radiation model
    // Use cloudFunctions to record particle tracks, calculate particle erosion
    cloudFunctions
    patchPostProcessing
    type patchPostProcessing;
    maxStoredParcels 1000000;
    patches
    (
    target
    );
    particleErosion
    type particleErosion;
    patches
    (
    target
    );
    p 450000000;

```

```
psi 2;
K 2;
```

C.11. g

```
dimensions [0 1 -2 0 0 0 0];
value (0 0 -9.81);
```

C.12. momentumTransport.water

```
simulationType RAS;
RAS
// Mandatory entries
RASModel kEpsilon;
// Optional entries
turbulence on;
printCoeffs on;
// Optional model coefficients
Cmu 0.09;
C1 1.44;
C2 1.92;
C3 0.0;
sigmak 1.0;
sigmaEps 1.3;
```

C.13. transportProperties

```
continuousPhaseName water;
rho.water 997;
transportModel Newtonian;
nu 1e-06;
continuousPhaseName air;
rho.air 1;
transportModel Newtonian;
nu 1.48e-05;
sigma 0.07
```

C.14. U.water

```

dimensions [0 1 -1 0 0 0];
internalField uniform (0 0 0);
boundaryField
{
    wall
    {
        type noSlip;
    }
    inlet
    {
        type fixedValue;
        value uniform (32.11 0 0);
        target
        {
            type fixedValue;
            value uniform (0 0 0);
        }
    }
    frontAndBack
    {
        type fixedValue;
        value uniform (0 0 0);
    }
    wall1
    {
        type inletOutlet;
        phi phi.water;
        inletValue uniform (0 0 0);
        value uniform (0 0 0);
    }
    wall2
    {
        type inletOutlet;
        phi phi.water;
        inletValue uniform (0 0 0);
        value uniform (0 0 0);
    }
    wall2
    {
        type noSlip;
    }
    lowerWall
    {
        type noSlip;
    }
    upperWall
    {
        type noSlip;
    }
}

```

front
type symmetryPlane;

C.15. p

dimensions [0 2 -2 0 0 0 0];
internalField uniform 0;
boundaryField
wall
type fixedFluxPressure;
value *internalField*;
inlet
type fixedFluxPressure;
value *internalField*;
target
type fixedFluxPressure;
value *internalField*;
frontAndBack
type fixedFluxPressure;
value *internalField*;
wall1
type fixedValue;
value uniform 0;
wall2
type fixedValue;
value *internalField*;
lowerWall
type fixedFluxPressure;
value *internalField*;
upperWall
type fixedFluxPressure;
value *internalField*;
front
type symmetryPlane;

C.16. k.water

```
dimensions [0 2 -2 0 0 0 0];
internalField uniform 0.236;
boundaryField
{
    lowerWall
    {
        type kqRWallFunction;
        value uniform 0.236;
    }
    upperWall
    {
        type kqRWallFunction;
        value uniform 0.236;
    }
    frontAndBack
    {
        type kqRWallFunction;
        value uniform 0.236;
    }
    wall1
    {
        type zeroGradient;
    }
    wall2
    {
        type zeroGradient;
    }
    wall
    {
        type kqRWallFunction;
        value uniform 0.236;
    }
    target
    {
        type kqRWallFunction;
        value uniform 0.236;
    }
    inlet
    {
        type fixedValue;
        value uniform 0.236;
    }
    front
    {
        type symmetryPlane;
    }
}
```

C.17. epsilon.water

```
dimensions [0 2 -3 0 0 0 0];
internalField uniform 38.37;
boundaryField
{
    lowerWall
    {
        type epsilonWallFunction;
    }
}
```

```
value uniform 38.37;
upperWall
type epsilonWallFunction;
value uniform 38.37;
frontAndBack
type epsilonWallFunction;
value uniform 38.37;
wall1
type zeroGradient;
wall2
type zeroGradient;
wall
type epsilonWallFunction;
value uniform 38.37;
target
type epsilonWallFunction;
value uniform 38.37;
inlet
type fixedValue;
value uniform 38.37;
front
type symmetryPlane;
```

C.18. nut.water

```
dimensions [0 2 -1 0 0 0 0];
internalField uniform 0;
boundaryField
lowerWall
type nutkWallFunction;
value uniform 0;
upperWall
type nutkWallFunction;
value uniform 0;
frontAndBack
type nutkWallFunction;
```

```
value uniform 0;
wall1
type calculated;
value uniform 0;
wall2
type calculated;
value uniform 0;
wall
type nutkWallFunction;
value uniform 0;
target
type nutkWallFunction;
value uniform 0;
inlet
type calculated;
value uniform 0;
front
type symmetryPlane;
```



D | Código python postprocesamiento

D.1. Codigo extraccion de datos

```
import math
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from scipy import linalg
import numpy
import pandas as pd
from untitled0 import *
import csv
nparticulas=5300053000
t=[]
d=open(r'/home/gonzalo/data/datos.csv','a')
writer=csv.writer(d)
origid=numpy.zeros(nparticulas)
x=numpy.zeros(nparticulas)
y=numpy.zeros(nparticulas)
z=numpy.zeros(nparticulas)
t1=numpy.zeros(nparticulas)
t2=numpy.zeros(nparticulas)
t3=numpy.zeros(nparticulas)
t4=numpy.zeros(nparticulas)
ux=numpy.zeros(nparticulas)
uy=numpy.zeros(nparticulas)
uz=numpy.zeros(nparticulas)
u=numpy.zeros(nparticulas)
r=numpy.zeros(nparticulas)
celdai=numpy.zeros(nparticulas)
```

```

celda=[]
bari=[]
numc=3
cont=0
tt=[]
contador=0
target=pd.read_csv(r'/home/gonzalo/data/target.csv')
ua=numpy.zeros(nparticulas)
ra=numpy.zeros(nparticulas)
ya=numpy.zeros(nparticulas)
za=numpy.zeros(nparticulas)
ang=numpy.zeros(nparticulas)
uprom=numpy.zeros(8)
angprom=numpy.zeros(8)
rad2deg = 180.0 / math.pi
ulopez=pd.read_csv(r'/home/gonzalo/data/U.csv')
uylopez=ulopez.iloc[:,1]
uxlopez=ulopez.iloc[:,0]
anglelopez=pd.read_csv(r'/home/gonzalo/data/angle.csv')
anglelylopez=anglelopez.iloc[:,1]
anglexlopez=anglelopez.iloc[:,0]
row=[]
for i in range(1,1):
df = pd.read_csv(r'/home/gonzalo/data/datos.csv')
df = pd.read_csv(r'/home/gonzalo/data/data0.' + str(i) + '.csv')
t.append(df)
print (df)
a=t[5].iloc[0][0]
b=t[5].iloc[1,5]row,column
print(type(a))
for k in t:
for i in range(len(k.iloc[:,0])):
if k.iloc[i,4]>=0.1245:
cont=cont+1
if int( origid[int(k.iloc[i,0])])==0:
contador=contador+1

```

```

origid[int(k.iloc[i,0])]=k.iloc[i,0]
row=[k.iloc[i,0],k.iloc[i,1],k.iloc[i,2],k.iloc[i,3],k.iloc[i,4],k.iloc[i,5],k.iloc[i,6]]
writer.writerow(row)
x[int(k.iloc[i,0])]=k.iloc[i,4]
y[int(k.iloc[i,0])]=k.iloc[i,5]
z[int(k.iloc[i,0])]=k.iloc[i,6]
r[int(k.iloc[i,0])]=(k.iloc[i,5]**2 + k.iloc[i,6]**2)**0.5
ra[int(k.iloc[i,0])]=(k.iloc[i,5]**2 + k.iloc[i,6]**2)**0.5
ya[int(k.iloc[i,0])]=k.iloc[i,5]
za[int(k.iloc[i,0])]=k.iloc[i,6]*-1
ua[int(k.iloc[i,0])]=(k.iloc[i,1]**2 + k.iloc[i,2]**2 + k.iloc[i,3]**2)**0.5
ang[int(k.iloc[i,0])]=90-math.acos(k.iloc[i,1]/ua[int(k.iloc[i,0])])*rad2deg
d.close()

for i in range(len(target.iloc[:,0])):
    t1[int(target.iloc[i,10])]=target.iloc[i,1]
    t2[int(target.iloc[i,10])]=target.iloc[i,2]
    t3[int(target.iloc[i,10])]=target.iloc[i,3]
    t4[int(target.iloc[i,10])]=target.iloc[i,4]
    ux[int(target.iloc[i,10])]=target.iloc[i,11]
    uy[int(target.iloc[i,10])]=target.iloc[i,12]
    uz[int(target.iloc[i,10])]=target.iloc[i,13]
    u[int(target.iloc[i,10])]=(target.iloc[i,11]**2 + target.iloc[i,12]**2 + target.iloc[i,13]**2)**0.5
    celdai[int(target.iloc[i,10])]=target.iloc[i,5]
    if target.iloc[i,5] not in celda:
        celda.append(target.iloc[i,5])
    plt.plot(ya, za, '.', color='red', markersize=1)
    plt.title("Localización impacto")
    plt.xlabel("m")
    plt.ylabel("m")
    plt.show()

c1=0
c2=0
c3=0
c4=0
c5=0
c6=0

```

```
c7=0
c8=0
for i in range(len(ua)):
    if ra[i]>0.1:0.004
    asdf=0
    elif ra[i]>0.00375:
    uprom[7]=uprom[7]+ua[i]
    angprom[7]=angprom[7]+ang[i]
    c1+=1
    elif ra[i]>0.00325:
    uprom[6]=uprom[6]+ua[i]
    angprom[6]=angprom[6]+ang[i]
    c2+=1
    elif ra[i]>0.00275:
    uprom[5]=uprom[5]+ua[i]
    angprom[5]=angprom[5]+ang[i]
    c3+=1
    elif ra[i]>0.00225:
    uprom[4]=uprom[4]+ua[i]
    angprom[4]=angprom[4]+ang[i]
    c4+=1
    elif ra[i]>0.00175:
    uprom[3]=uprom[3]+ua[i]
    angprom[3]=angprom[3]+ang[i]
    c5+=1
    elif ra[i]>0.00125:
    uprom[2]=uprom[2]+ua[i]
    angprom[2]=angprom[2]+ang[i]
    c6+=1
    elif ra[i]>0.00075:
    uprom[1]=uprom[1]+ua[i]
    angprom[1]=angprom[1]+ang[i]
    c7+=1
    elif ra[i]>0:
    uprom[0]=uprom[0]+ua[i]
    angprom[0]=angprom[0]+ang[i]
```

```

c8+=1
uprom[7]=uprom[7]/c1
uprom[6]=uprom[6]/c2
uprom[5]=uprom[5]/c3
uprom[4]=uprom[4]/c4
uprom[3]=uprom[3]/c5
uprom[2]=uprom[2]/c6
uprom[1]=uprom[1]/c7
uprom[0]=uprom[0]/c8
angprom[7]=angprom[7]/c1
angprom[6]=angprom[6]/c2
angprom[5]=angprom[5]/c3
angprom[4]=angprom[4]/c4
angprom[3]=angprom[3]/c5
angprom[2]=angprom[2]/c6
angprom[1]=angprom[1]/c7
angprom[0]=angprom[0]/c8
rprom=numpy.linspace(0.5,4.0,num=8)
plt.plot(rprom,uprom,label='Resultados experiencia')
plt.plot(uxlopez,uylopez,label='Resultados Lopez')
plt.xlim(0.5,4)
plt.ylim(0,8)
plt.title("Velocidad de impacto")
plt.xlabel("Distancia [m]")
plt.ylabel("Velocidad de impacto [m/s]")
plt.legend()
plt.show()
plt.plot(rprom,angprom,color='red',label='Resultados experiencia')
plt.plot(anglexlopez,angleylopez,label='Resultados Lopez')
plt.xlim(0.5,4)
plt.ylim(0,90)
plt.title("Ángulo de impacto")
plt.xlabel("Distancia [m]")
plt.ylabel("Ángulo de impacto [°]")
plt.legend()
plt.show()

```

D.2. Codigo orden de datos

```
import math
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from scipy import linalg
import numpy
import pandas as pd
from untitled0 import *
import csv
nparticulas=5300053000
t=[]
d=open(r'/home/gonzalo/data/datos.csv','a')
writer=csv.writer(d)
origid=numpy.zeros(nparticulas)
x=numpy.zeros(nparticulas)
y=numpy.zeros(nparticulas)
z=numpy.zeros(nparticulas)
t1=numpy.zeros(nparticulas)
t2=numpy.zeros(nparticulas)
t3=numpy.zeros(nparticulas)
t4=numpy.zeros(nparticulas)
ux=numpy.zeros(nparticulas)
uy=numpy.zeros(nparticulas)
uz=numpy.zeros(nparticulas)
u=numpy.zeros(nparticulas)
r=numpy.zeros(nparticulas)
celdai=numpy.zeros(nparticulas)
celda=[]
bari=[]
numc=3
cont=0
tt=[]
contador=0
target=pd.read_csv(r'/home/gonzalo/data/target.csv')
ua=numpy.zeros(nparticulas)
```

```

ra=numpy.zeros(nparticulas)
ya=numpy.zeros(nparticulas)
za=numpy.zeros(nparticulas)
ang=numpy.zeros(nparticulas)
uprom=numpy.zeros(8)
angprom=numpy.zeros(8)
rad2deg = 180.0 / math.pi
ulopez=pd.read_csv(r'/home/gonzalo/data/U.csv')
uylopez=ulopez.iloc[:,1]
uxlopez=ulopez.iloc[:,0]
anglelopez=pd.read_csv(r'/home/gonzalo/data/angle.csv')
anglelylopez=anglelopez.iloc[:,1]
anglexlopez=anglelopez.iloc[:,0]
row=[]
for i in range(1,1313):
df = pd.read_csv(r'/home/gonzalo/data/datos1.csv')
df = pd.read_csv(r'/home/gonzalo/data/data0.' + str(i) + '.csv')
t.append(df)
print (df)
a=t[5].iloc[0][0]
b=t[5].iloc[1,5]row,column
print(type(a))
for k in t:
for i in range(len(k.iloc[:,0])):
if k.iloc[i,4]>=0.1245:
cont=cont+1
if int( origid[int(k.iloc[i,0])])==0:
contador=contador+1
origid[int(k.iloc[i,0])]=k.iloc[i,0]
row=[k.iloc[i,0],k.iloc[i,1],k.iloc[i,2],k.iloc[i,3],k.iloc[i,4],k.iloc[i,5],k.iloc[i,6]]
writer.writerow(row)
x[int(k.iloc[i,0])]=k.iloc[i,4]
y[int(k.iloc[i,0])]=k.iloc[i,5]
z[int(k.iloc[i,0])]=k.iloc[i,6]
r[int(k.iloc[i,0])]=(k.iloc[i,5]**2 + k.iloc[i,6]**2)**0.5
ra[int(k.iloc[i,0])]=(k.iloc[i,5]**2 + k.iloc[i,6]**2)**0.5

```

```

ya[int(k.iloc[i,0])]=k.iloc[i,5]
za[int(k.iloc[i,0])]=k.iloc[i,6]*-1
ua[int(k.iloc[i,0])]=(k.iloc[i,1]**2 + k.iloc[i,2]**2 + k.iloc[i,3]**2)**0.5
ang[int(k.iloc[i,0])]=90-math.acos(k.iloc[i,1]/ua[int(k.iloc[i,0])))*rad2deg
d.close()

for i in range(len(target.iloc[:,0])):
    t1[int(target.iloc[i,10])]=target.iloc[i,1]
    t2[int(target.iloc[i,10])]=target.iloc[i,2]
    t3[int(target.iloc[i,10])]=target.iloc[i,3]
    t4[int(target.iloc[i,10])]=target.iloc[i,4]
    ux[int(target.iloc[i,10])]=target.iloc[i,11]
    uy[int(target.iloc[i,10])]=target.iloc[i,12]
    uz[int(target.iloc[i,10])]=target.iloc[i,13]
    u[int(target.iloc[i,10])]=(target.iloc[i,11]**2 + target.iloc[i,12]**2 + target.iloc[i,13]**2)**0.5
    celdai[int(target.iloc[i,10])]=target.iloc[i,5]
    if target.iloc[i,5] not in celda:
        celda.append(target.iloc[i,5])
    plt.plot(ya, za, '.', color='red', markersize=1)
    plt.title("Localización impacto")
    plt.xlabel("m")
    plt.ylabel("m")
    plt.show()
    c1=0
    c2=0
    c3=0
    c4=0
    c5=0
    c6=0
    c7=0
    c8=0
    for i in range(len(ua)):
        if ra[i]>0.1:0.004
            asdf=0
        elif ra[i]>0.00375:
            uprom[7]=uprom[7]+ua[i]
            angprom[7]=angprom[7]+ang[i]

```

```
c1+=1
elif ra[i]>0.00325:
    uprom[6]=uprom[6]+ua[i]
    angprom[6]=angprom[6]+ang[i]
c2+=1
elif ra[i]>0.00275:
    uprom[5]=uprom[5]+ua[i]
    angprom[5]=angprom[5]+ang[i]
c3+=1
elif ra[i]>0.00225:
    uprom[4]=uprom[4]+ua[i]
    angprom[4]=angprom[4]+ang[i]
c4+=1
elif ra[i]>0.00175:
    uprom[3]=uprom[3]+ua[i]
    angprom[3]=angprom[3]+ang[i]
c5+=1
elif ra[i]>0.00125:
    uprom[2]=uprom[2]+ua[i]
    angprom[2]=angprom[2]+ang[i]
c6+=1
elif ra[i]>0.00075:
    uprom[1]=uprom[1]+ua[i]
    angprom[1]=angprom[1]+ang[i]
c7+=1
elif ra[i]>0:
    uprom[0]=uprom[0]+ua[i]
    angprom[0]=angprom[0]+ang[i]
c8+=1
uprom[7]=uprom[7]/c1
uprom[6]=uprom[6]/c2
uprom[5]=uprom[5]/c3
uprom[4]=uprom[4]/c4
uprom[3]=uprom[3]/c5
uprom[2]=uprom[2]/c6
uprom[1]=uprom[1]/c7
```

```
uprom[0]=uprom[0]/c8
angprom[7]=angprom[7]/c1
angprom[6]=angprom[6]/c2
angprom[5]=angprom[5]/c3
angprom[4]=angprom[4]/c4
angprom[3]=angprom[3]/c5
angprom[2]=angprom[2]/c6
angprom[1]=angprom[1]/c7
angprom[0]=angprom[0]/c8
rprom=numpy.linspace(0.5,4.0,num=8)
plt.plot(rprom,uprom,label='Resultados experiencia')
plt.plot(uxlopez,uylopez,label='Resultados Lopez')
plt.xlim(0.5,4)
plt.ylim(0,8)
plt.title("Velocidad de impacto")
plt.xlabel("Distancia [m]")
plt.ylabel("Velocidad de impacto [m/s]")
plt.legend()
plt.show()

plt.plot(rprom,angprom,color='red',label='Resultados experiencia')
plt.plot(anglexlopez,angleylopez,label='Resultados Lopez')
plt.xlim(0.5,4)
plt.ylim(0,90)
plt.title("Ángulo de impacto")
plt.xlabel("Distancia [m]")
plt.ylabel("Ángulo de impacto [°]")
plt.legend()
plt.show()
```

E | Tiempo exposición de la pala

Si se considera que la velocidad tangencial de la turbina es igual a la velocidad del chorro, es decir $V=32.11$ m/s, se busca la velocidad angular con la siguiente relación:

$$\omega = \frac{V}{r} \quad (\text{E.1})$$

Así con un radio de 0.096 m, ω tiene un valor de 334,5 [Rad/s]. Con ello se puede obtener el periodo con la siguiente ecuación.

$$T = \frac{2\pi}{\omega} \quad (\text{E.2})$$

el periodo es de 0.019 segundos, como son 24 palas el tiempo de exposición al impacto de las partículas de cada una de ellas es de 0.0008 y en un total de 3 horas en las que hay 568421 revoluciones el tiempo de exposición total es de 450 segundos.