Escuela Politécnica Superior de Jaén
Grado en Ingeniería Mecánica

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

ANÁLISIS DE LA INTERACCIÓN DE BURBUJAS EN UN FLUJO TURBULENTO

Alumno: Carlos Avilés Coca

Tutor: Prof. D. Carlos Martínez Bazán
Dpto: Ingeniería Mecánica y Minera

Junio, 2019

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Ingeniería Mecánica y Minera

Don Jesús Carlos Martínez Bazán , tutor del Proyecto Fin de Carrera titulado: **ANÁLISIS DE LA INTERACCIÓN DE BURBUJAS EN UN FLUJO TURBULENTO,** que presenta Carlos Avilés Coca, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2019

El alumno:

Carlos Avilés Coca

El tutor:

Jesú Carlos Martínez Bazán

ÍNDICE:

ÍNDICE DE FIGURAS:

ÍNDICE DE TABLAS:

NOMENCLATURA:

v ≡ Vector campo de velocidades.

w ≡ Vector vorticidad.

Γ ≡ Circulación.

D ≡ Diámetro del vórtice.

r_n ≡ Radio del núcleo.

y ≡ Desplazamiento en la dirección axial.

t ≡ Tiempo.

V_a ≡ Velocidad de avance del vórtice.

Γ^* ≡ Circulación adimensional.

E_c ≡ Energía cinética.

V_{at} ≡ Velocidad de avance del vórtice teórica.

Re ≡ Número de Reynolds.

F_v ≡ Fuerza de flotabilidad.

F_σ ≡ Fuerza realizada por la tensión superficial.

σ ≡ Tensión superficial.

V_b ≡ Volumen de una burbuja.

R_b ≡ Radio de una burbuja esférica.

a ≡ Radio de la aguja que genera una burbuja.

ρ ≡ Densidad del agua.

g ≡ Constante gravitacional.

V_F ≡ Volumen de Fritz.

R_F ≡ Radio de una burbuja según volumen de Fritz.

Q ≡ Caudal de aire.

Q_{cr} ≡ Caudal crítico de Fritz.

Ga ≡ Número de Galilei.

Bo ≡ Número de Bond.

We ≡ Número de Weber.

ν ≡ Viscosidad cinemática del agua.

Re ≡ Número de Reynolds.

Δy ≡ Paso de maya del eje de ordenadas.

D_b ≡ Diámetro de una burbuja esférica.

MOTIVACIÓN:

El estudio de flujos turbulentos dentro de un medio continuo es complejo ya que encontramos una gran irregularidad y caos, es decir son flujos estocásticos. No obstante, conocemos que son flujos rotacionales, muy difusivos, con números de Reynolds elevados.

Si a esta fase líquida le añadimos otra gaseosa, el comportamiento de ambas será complicado de predecir, de ahí la importancia de su estudio.

La interacción de flujos turbulentos con fases gaseosas, se pueden observar en numerosas aplicaciones del campo de la industria como en la naturaleza. Claros ejemplos son la rotura del oleaje, las nubes de burbujas generadas en la superficie del mar, el proceso de aireación y mezcla dentro de una planta de tratamiento de aguas residuales, la oxigenación de piscifactorías, la inyección de micro-burbujas dentro de un flujo turbulento para reducir la fuerza de arrastre, entre otras muchas más aplicaciones.

Para llegar a entender este tipo de flujos es necesario analizar problemas simplificados que nos permitan vislumbran la física subyacente y así avanzar en el conocimiento de su comportamiento. En este trabajo analizaremos experimentalmente sólo la interacción entre un anillo de vorticidad y una burbuja, para predecir el comportamiento entre ellos. Con este fin, es imprescindible conocer de manera detallada los parámetros que definirán a ambos elementos por separado antes de ponerlos en contacto.

Un anillo de vorticidad es una estructura vorticial de forma toroidal (ver **Figura 1**) que presenta dos regiones diferenciadas. Lejos del eje de simetría la velocidad disminuye proporcionalmente a 1/***r***, donde ***r*** es la distancia radial al eje. Sin embargo, el anillo presenta una región donde la velocidad presenta una dependencia lineal con ***r***.

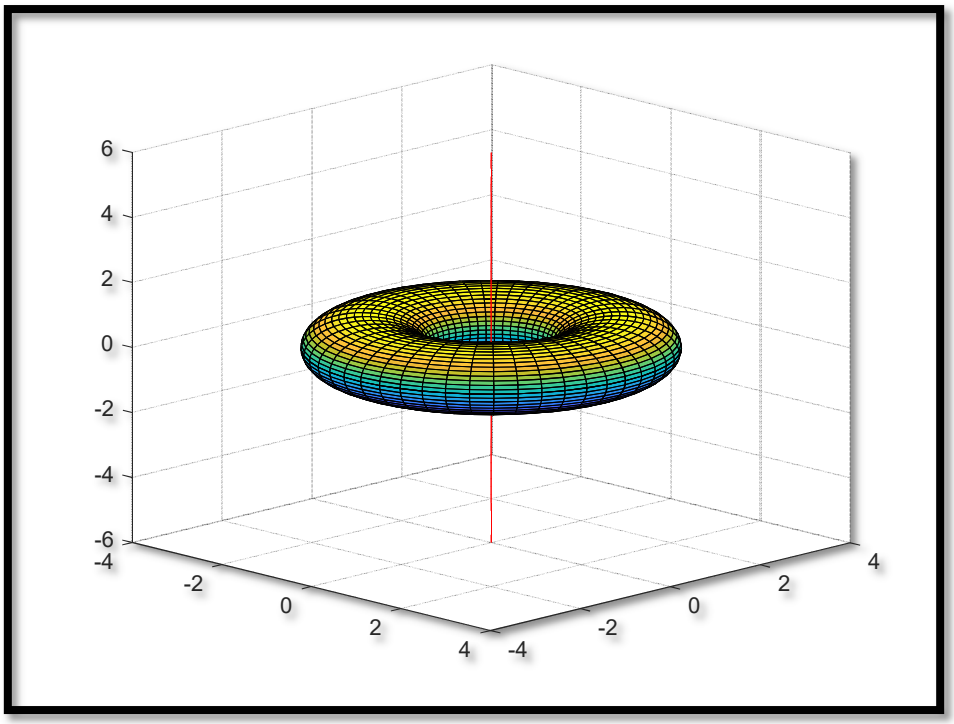

FIGURA 1: TOROIDE Y EJE DE AXIAL REALIZADO EN MATLAB.

La **Figura 2** muestra el campo de velocidades en un plano que pasa por el eje de un anillo de vorticidad. La primera región es una zona con vorticidad nula, mientras que en la segunda región la vorticidad es diferente de cero. Finalmente, cerca del eje de simetría, la velocidad es casi constante, y la vorticidad también es prácticamente nula. Esto nos muestra que el anillo tiene un núcleo con vorticidad aproximadamente constante y nos permite definir su diámetro como la distancia entre los centros del núcleo del anillo de vorticidad, como se muestra en la **Figura 3** y en la **Figura 5**.

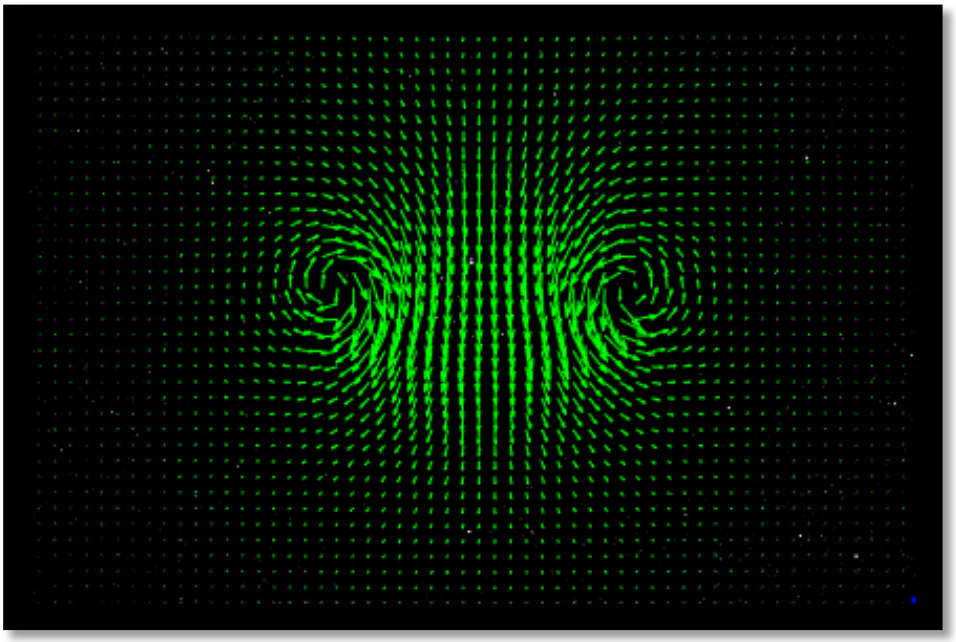

FIGURA 2: CAMPO DE VELOCIDAD DE UN VÓRTICE EN UN PLANO.

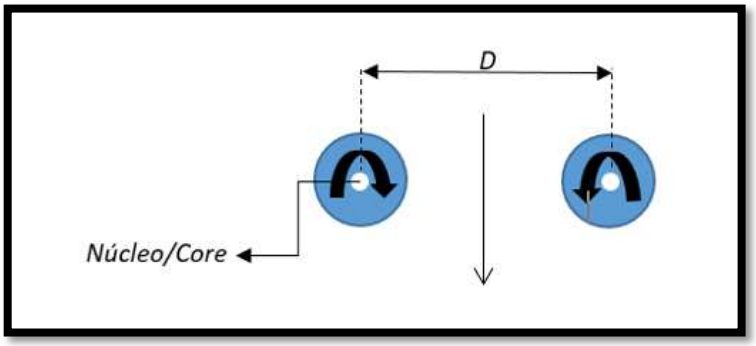

FIGURA 3: ESQUEMA GEOMÉTRICO DE UN VÓRTICE SECCIONADO.

Como se ha comentado anteriormente, el diámetro del vórtice, será la distancia que exista entre los dos núcleos de la sección. Por tanto, surge la necesidad de situar la posición de ambos núcleos, para esto recurriremos a la definición de vorticidad.

$$w = \nabla \wedge v \qquad [\text{Ec. } 1]$$

La vorticidad es el rotacional del campo de velocidad, este operador vectorial muestra la *tendencia de un campo a inducir rotación alrededor de un punto* **[1]**. Por ende, la vorticidad será máxima en valor absoluto en el núcleo del vórtice.

Si desarrollamos la vorticidad para un campo de velocidad cuyas componentes sean *u, v,* resultará **[2]**:

$$w = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \qquad [\text{Ec. } 2]$$

La dirección del vector resultante será la perpendicular al plano analizado.

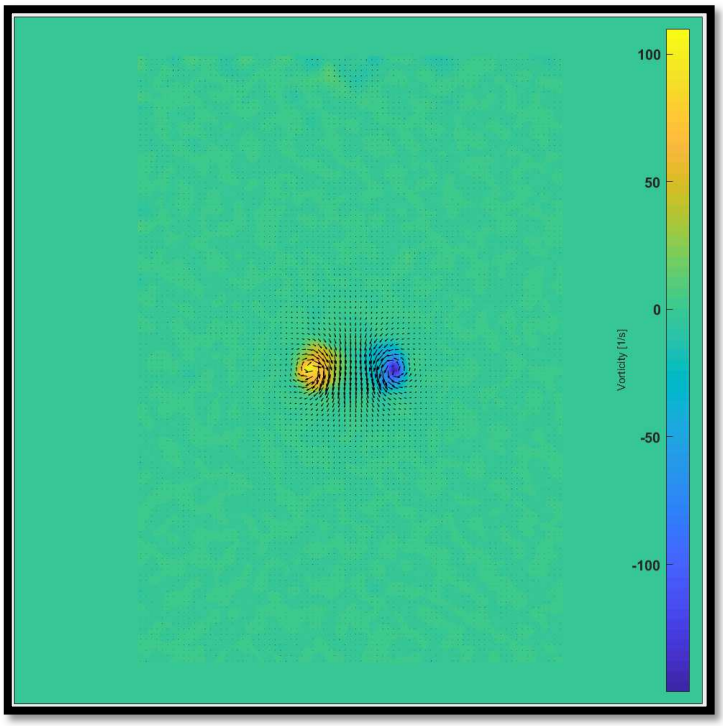

FIGURA 4: VORTICIDAD DE UN VÓRTICE EN EL PLANO.

Analizando la vorticidad a lo largo de un eje longitudinal que contenga a ambos núcleos, podremos ver de manera directa el diámetro del vórtice (**Figura 4**). El hecho de que en los núcleos se dé el mismo valor de vorticidad y sean de signo contrario, es debido a la simetría y al sentido de giro respectivamente (**Figura 5**).

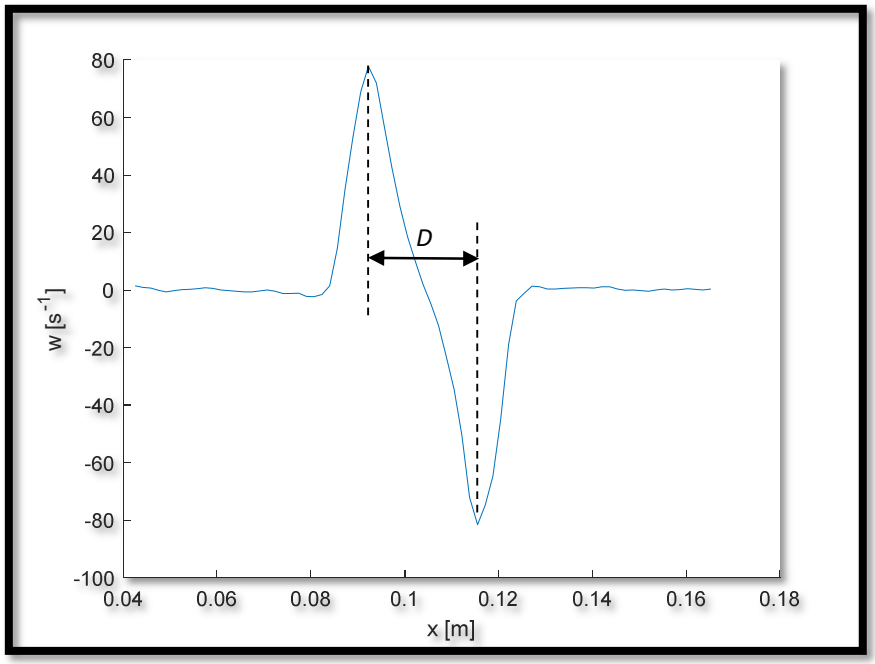

FIGURA 5: DEFINICIÓN DEL DIÁMETRO DEL VÓRTICE A TRAVÉS DE LA VORTICIDAD A LO LARGO DE UN EJE LONGITUDINAL QUE PASA POR LOS NÚCLEOS DE LA SECCIÓN.

Si analizamos la **Figura 5** junto con la velocidad axial a lo largo de dicho eje longitudinal que pase por los núcleos, podremos establecer el radio del núcleo o core, siendo éste la diferencia de la posición de máxima vorticidad en valor absoluto y la posición del mínimo de velocidad axial **[3]** (véase en la **Figura 6**), también se podría calcular para la otra mitad delimitada por el eje de simetría axial.

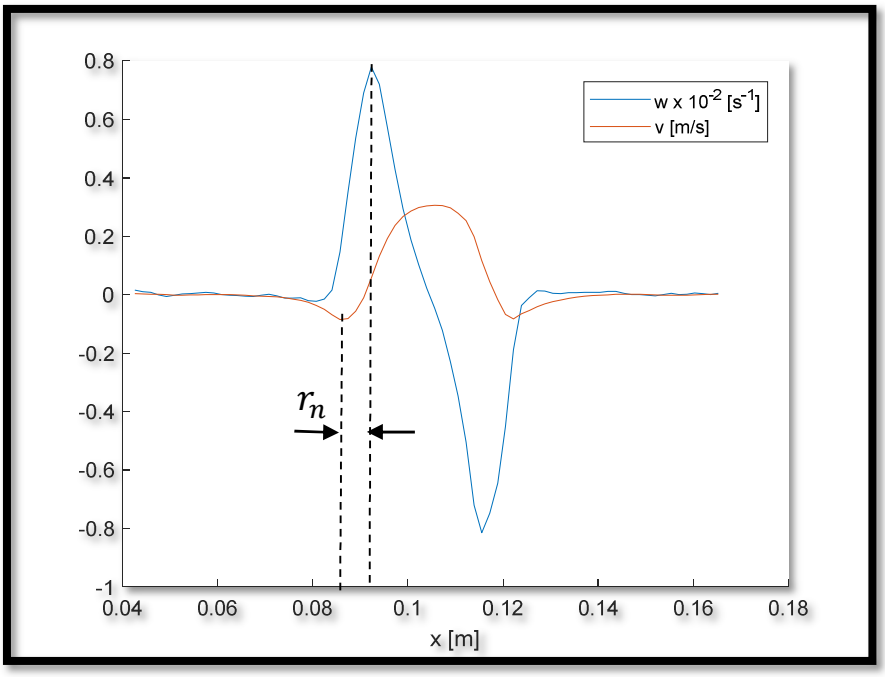

FIGURA 6: DEFINICIÓN DEL RADIO DEL NÚCLEO SEGÚN LA VORTICIDAD Y VELOCIDAD AXIAL A LO LARGO DE UN EJE LONGITUDINAL QUE PASA POR LOS NÚCLEOS DE LA SECCIÓN.

El vórtice es generado a partir de un gradiente de presiones generados en un conducto con un orificio de diámetro conocido en su extremo, una vez que este desaparece, lo único que justifica la dinámica del vórtice será su energía cinética, para poder cuantificarla recurriremos a la circulación. Entendemos como circulación a la integral a lo largo de una línea cerrada de un campo de velocidad.

$$\Gamma = \oint vdl \qquad [\mathrm{Ec.}\,3]$$

Para calcular la circulación, recurriremos a una línea cerrada en forma de rectángulo que albergue una de las mitades limitadas por eje axial de simetría en el plano (**Figura 7**).

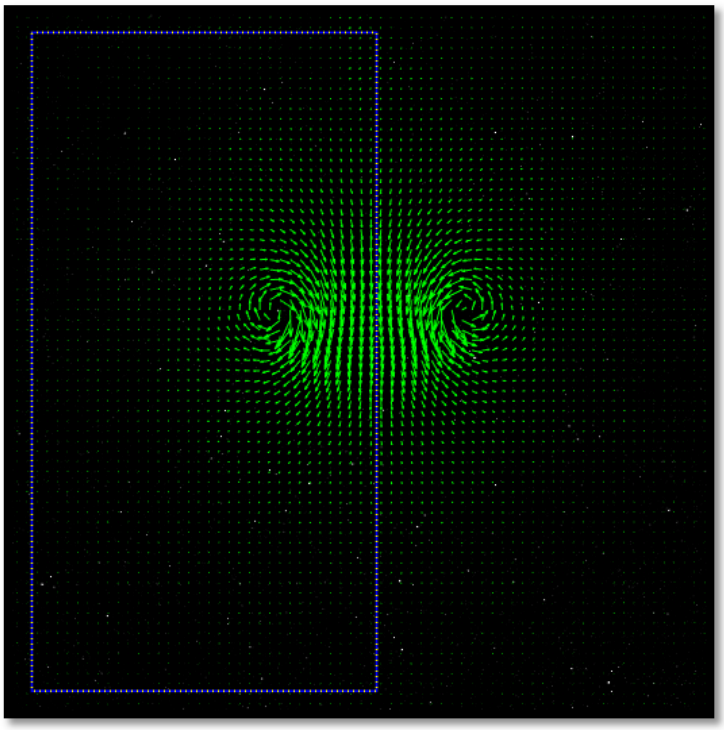

FIGURA 7: LÍNEA CERRADA UTILIZADA PARA EL CÁLCULO DE LA CIRCULACIÓN.

El Teorema de Thomson dice que *no puede aparecer ninguna circulación en el campo fluido cuando partimos del reposo* **[4]**, por lo que, si calculamos la circulación de la otra mitad restante, debe de salir el mismo valor de circulación, Γ, pero de signo contrario, para que la suma de ambas sea nula.

También podremos obtener la circulación en su formato adimensional, donde se relaciona con la velocidad de avance del vórtice y el diámetro del vórtice.

$$\Gamma^* = \frac{\Gamma}{DV_a} \qquad [Ec.4]$$

La velocidad de avance del vórtice, V_a, será la variación de la posición del centro de gravedad del vórtice a lo largo de la dirección axial en función del tiempo.

$$V_a = \frac{dy}{dt} \quad [Ec.5]$$

La energía cinética con la que avanza el vórtice será proporcional a la velocidad auto inducida al cuadrado, es decir la velocidad de avance al cuadrado, del mismo modo, la velocidad de avance será proporcional a la circulación **[5]**. Por lo que podemos establecer una relación directa entre la energía cinética o fuerzas inerciales con la circulación **[3]**.

$$E_c \propto V_a^2 \quad [Ec.6]$$

$$V_a \propto \frac{\Gamma}{\pi D} \quad [Ec.7]$$

$$E_c \propto \left(\frac{\Gamma}{\pi D}\right)^2 \quad [Ec.8]$$

Otra forma de poder calcular la velocidad autoinducida sería recurriendo a la siguiente expresión, la cual se ajusta bastante bien, siempre que conozcamos el radio del núcleo **[3]**.

$$V_{at} = \frac{\Gamma}{2\pi D}\left[\ln\left(\frac{4D}{r_n}\right) - \frac{1}{4}\right] \quad [Ec.9]$$

Como modo de comparación podemos despejar de esta ecuación el radio del núcleo que resultaría si $V_{at} = V_a$.

$$r_n = \frac{4D}{e^{\left(V_a \frac{2\pi D}{\Gamma} + \frac{1}{4}\right)}} \quad [Ec.10]$$

Si quisiéramos relacionar las fuerzas inerciales entre las fuerzas viscosas solamente tendríamos que relacionar la circulación con la viscosidad cinemática del fluido donde el vórtice se desarrolla, obteniendo el número de Reynolds **[6]**.

$$Re = \frac{\Gamma}{\nu} \quad [Ec.11]$$

Con el fin de buscar una relación directa de la circulación, aplicamos el *Teorema de Stokes* a la definición de circulación **[7]**.

$$\Gamma = \oint v dl = \iint (\nabla \wedge v)\, dS = \iint w\, dS \quad [Ec. 12]$$

No nos olvidemos del otro elemento fundamental, la burbuja. Buscamos una generación de burbujas cuasi estática con el fin de establecer sus características geométricas. En este caso, intervendrán fundamentalmente dos fuerzas que interaccionan en el desprendimiento de la burbuja, estas serán: la fuerza de flotabilidad, F_v, que se rige por el *Principio de Arquímedes* y la fuerza que realiza la tensión superficial, F_σ, que se opone al desprendimiento de la misma **[8]**.

$$\uparrow F_v = \rho g V_b \quad [Ec. 13]$$

$$\downarrow F_\sigma = 2\pi a \sigma \quad [Ec. 14]$$

Siendo *a* el radio de la aguja que genera la burbuja y σ la tensión superficial aire-agua.

La burbuja se desprenderá en el instante en el que $F_v > F_\sigma$, por lo que en el instante previo, donde $F_v = F_\sigma$, podremos calcular el volumen máximo que tendría la burbuja, este volumen es el conocido como Volumen de *Fritz*.

$$\sum F = 0; \quad F_v = F_\sigma \quad [Ec. 15]$$

$$V_F = \frac{2\pi a \sigma}{\rho g} \quad [Ec. 16]$$

En el caso de considerar una burbuja esférica **(Ec. 10)**, podremos obtener el Radio de la burbuja, R_b, según el Volumen de *Fritz* **(Ec. 11)**.

$$V_b = \frac{4}{3}\pi R_b^3 \quad [Ec. 17]$$

$$R_b = R_F = \left(\frac{3}{2}\frac{\sigma a}{\rho g}\right)^{1/3} \quad [Ec. 18]$$

Para garantizar que la generación cumple los requisitos para que sea cuasi estática, debemos comparar el caudal de aire inyectado con el caudal crítico de *Fritz*. Siempre que este último sea mayor que el caudal con el que generamos la burbuja, podremos asumir condiciones cuasi estáticas **[9]**.

$$Q_{cr} > Q \quad [Ec. 19]$$

$$Q_{cr} = \pi \left(\frac{16}{3g^2}\right)^{\frac{1}{6}} \left(\frac{\sigma a}{\rho}\right)^{\frac{5}{6}} \quad [Ec. 20]$$

Ya que el caudal del flujo de aire que es utilizado para generar la burbuja es complicado de cuantificar directamente compararemos el caudal de *Frizt* con el obtenido de manera numérica a partir del procesamiento de imágenes. En el caso de que el volumen de *Frizt* sea mayor del volumen de la burbuja generada, V_b, podremos afirmar que estamos en un estado cuasi estático.

$$V_F > V_b \quad [Ec. 21]$$

Para caracterizar el tipo de burbuja, tendremos que recurrir a los números adimensionales de *Bond* y *Galilei.*

El número adimensional *Galilei* representa la relación entre las fuerzas gravitacionales y las fuerzas viscosas. Mientras que el número adimensional *Bond* muestra la relación entre las fuerzas gravitacionales y las fuerzas generadas por la tensión superficial **[10]**.

$$Ga = \sqrt{\frac{g\ R_b^3}{\nu^2}} \quad [Ec. 22]$$

$$Bo = g\rho\frac{R_b^2}{\sigma} \quad [Ec. 23]$$

En lo referente a la interacción burbuja-torbellino, que al fin y al cabo es el objetivo de este trabajo, la cuantificaremos a través del numero adimensional de *Weber*, éste muestra la relación de las fuerzas inerciales del vórtice frente a las de tensión superficial de la burbuja, es decir, la fuerza que realiza el vórtice para romper la burbuja entre la fuerza de confinamiento de la tensión superficial **[3]**.

$$We = \rho\frac{\left(\frac{\Gamma}{\pi D}\right)^2}{\frac{\sigma}{R_b}} \quad [Ec. 24]$$

Siendo el valor de este número adimensional será fundamental a la hora de caracterizar la interacción. Otro número adimensional útil para la caracterización de la interacción será el número de *Reynolds* basado en el tamaño de la burbuja **[3]**, sabemos que este número adimensional refleja la interacción de las fueras inerciales entre las fuerzas viscosas, en este caso como longitud característica tendremos el radio de la burbuja y como velocidad, la auto inducida del vórtice, representada por una expresión proporcional a ella, reflejada en la **Ecuación 7**.

$$Re = \frac{\left(\frac{\Gamma}{\pi D}\right)R_b}{\nu} \quad [Ec. 25]$$

INSTALACIÓN EXPERIMENTAL:

FORMACIÓN DE ANILLOS DE VORTICIDAD Y BURBUJAS:

Para la formación de vórtices, forzaremos a un fluido presurizado, en nuestro caso agua, a pasar durante un tiempo determinado por un orificio, de diámetro interior conocido, que descargará a un depósito a presión atmosférica. Con el fin de presurizar el fluido, utilizaremos un depósito secundario, el cual llenaremos parcialmente de agua y presurizaremos al introducir en el mismo aire a una presión determinada. En cuanto a la generación de burbujas, recurriremos a una bomba de jeringa, con la cual podremos controlar el avance del émbolo y por tanto el caudal de aire que hacemos pasar a través de una aguja.

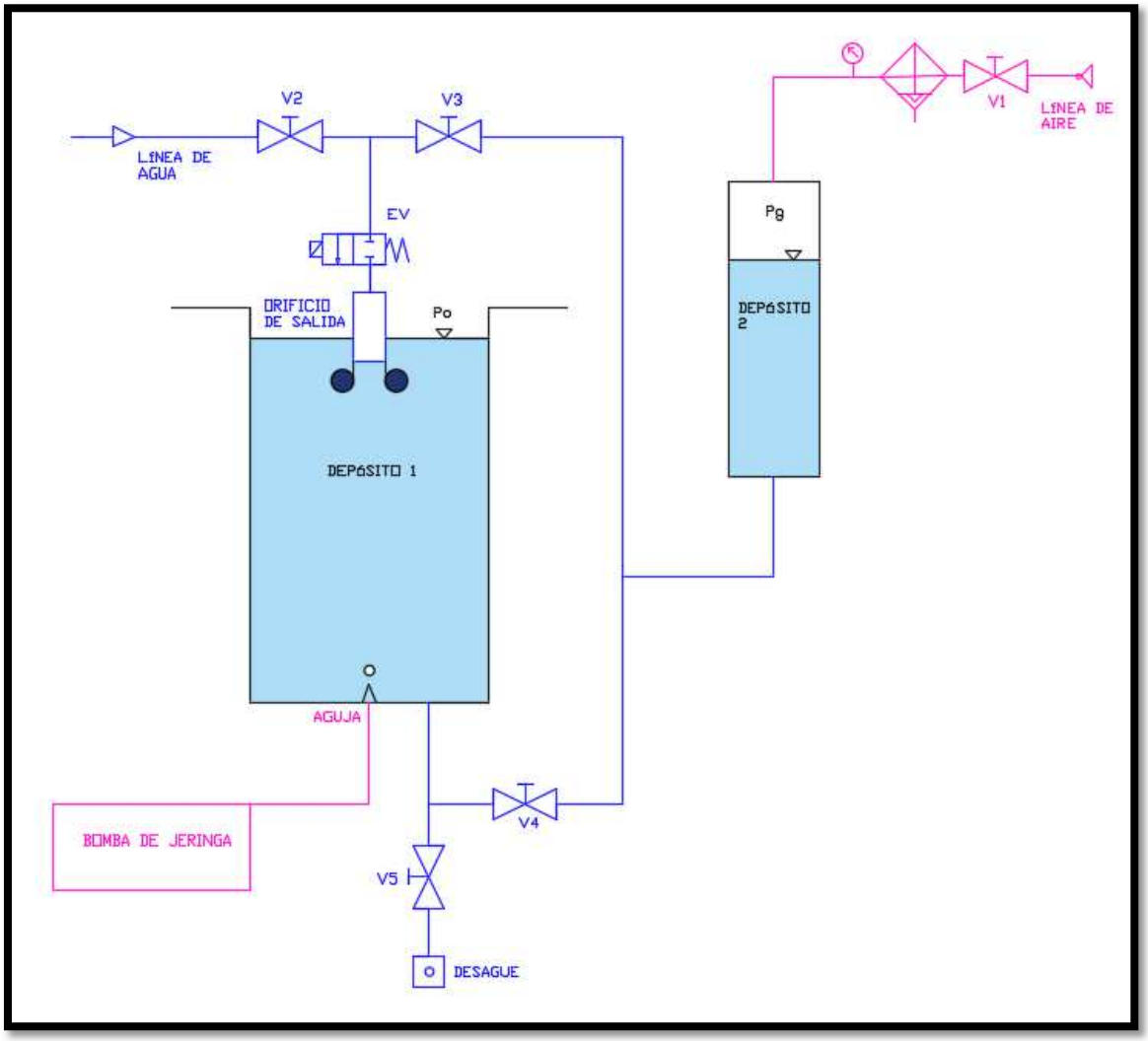

FIGURA 8: ESQUEMA DEL SISTEMA HIDRÁULICO Y NEUMÁTICO EN LA INSTALACIÓN.

En la instalación mostrada en la **Figura 8**, podemos apreciar válvulas de bola (V1, V2, V3, V4, V5) con estas podremos regular el camino del fluido según nos interese. La conexión entre las diferentes válvulas, electroválvula y depósitos se realizará a través de tubos de PVC de 8x5,5mm y la conexión entre la bomba de jeringa y la aguja con tubo de PVC de 4x6mm.

Para el llenado del depósito principal solamente tendríamos que abrir simultáneamente V2 y la electroválvula (EV), en el caso de querer llenar el depósito secundario con el fluido contenido en el principal, solamente tendríamos que abrir V4, en este caso el depósito secundario debe de estar a presión atmosférica. Una vez están llenos ambos depósitos, con todas las válvulas cerradas, podremos presurizar el depósito secundario abriendo V1 y regulando la presión a través del manómetro. Abriendo V3 tendríamos un flujo directo desde el depósito presurizado a la tobera, controlado a través de la apertura de la electroválvula. El último tramo encauzado del fluido presurizado, la tobera u orificio en nuestro caso, que tendrá un diámetro interior de salida variable, podrá ser modificado a través de una serie de a boquillas intercambiables.

La necesidad de utilizar una bomba de jeringa (*Standard Infuse/Withdraw Pump 11 Elite Programmable Syringe Pumps de Harvard Apparatus*) se debe a querer generar una única burbuja en un instante determinado, para ello controlamos el avance del embolo, este se moverá a velocidad constante, en nuestro caso la jeringa será de 20 mL ya que es lo suficientemente grande para satisfacer el caudal de aire necesario. Podremos trabajar con diferentes agujas de diferentes diámetros de salida, ya que se podrán acoplar a la base del depósito principal. Tanto la aguja como el orificio de salida deben de estar alineadas para que la interacción entre el vórtice generado y la burbuja se dé a lo largo del eje de simetría axial de ambos elementos.

Como hemos dicho al principio el fluido presurizado debe de ser inyectado un tiempo determinado y controlado a través del orificio con el fin de generar un vórtice, este tiempo será controlado a través de un controlador electrónico (*Arduino UNO*). Implementando un algoritmo regularemos el accionamiento de un relé (*JQC-3FF-S-Z*), éste a su vez controlará la electroválvula (válvula de solenoide normalmente cerrada), el uso del relé es obligado ya que la electroválvula trabaja a 220 V AC y la salida del controlador es de 5V DC. La instalación está preparada para la interacción vórtice con burbuja, encontrando un detector fotoeléctrico (*XUBLANCNL2*), conectado al controlador, encargado de detectar la salida de la burbuja para que, pasado un tiempo, se accione la electroválvula. Para llevar a cabo los experimentos de caracterización de los anillos de vorticidad, donde no es necesario generar burbujas, se dispone de un pulsador que puentea la señal y permite la generación de vórtices sin la presencia de burbujas.

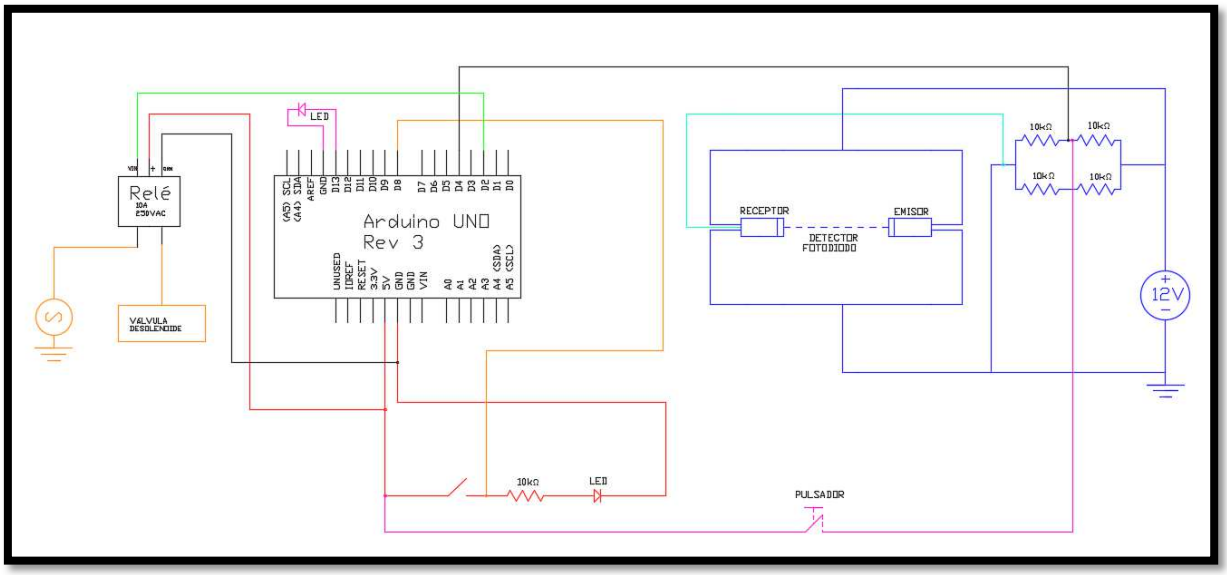

FIGURA 9: ESQUEMA ELECTRÓNICO DE CONTROL DE LA ELECTROVÁLVULA.

El controlador también tiene asociada una entrada regulada por un interruptor que permite la apertura de manera continua de la electroválvula, accionándolo solo para el proceso de llenado.

Como podemos observar en el esquema de control mostrado en la **Figura 9**, el detector fotoeléctrico está alimentado por 12V DC, dando una señal de salida de 12V DC cuando este se encuentra en continuo, en el caso de detectar una discontinuidad entre su emisor y receptor dará una salida de 0V DC, por ello debemos enfrentar la salida del receptor a una fuente de tensión de 12V DC para que no se produzca una diferencia de tensión en el caso de que el detector fotoeléctrico se encuentre en continuo. En caso contrario, se generará una diferencia de tensión de 12V DC que gracias a un divisor de tensión se transformará en una señal continua capaz de ser leída por el controlador para accionar la electroválvula, el pulsador es el encargado de puentear esta señal. Los algoritmos de control se establecerán en su apartado correspondiente dentro de los Anexos.

ANÁLISIS DE VÓRTICES:

Recurriremos a una técnica conocida como PIV, son las siglas de *Particle Image Velocimetry* una técnica cuantitativa que nos permite determinar el campo de velocidades de un fluido, es decir, cómo se desplazan las partículas fluidas a lo largo de un plano en un tiempo conocido.

Esta técnica se fundamenta en la generación de dos haces de luz láser separados un tiempo establecido, que posteriormente a través de un sistema de lentes los transformarán en un par de planos que inciden directamente en el fluido, objeto de estudio. El fluido debe de contener micro partículas reflectantes que reflejan la luz del láser que incide sobre ellas. Sincronizando la captación de imágenes con la generación de planos, podremos obtener imágenes de las partículas en cada instante de tiempo, con el fin de procesarlas y determinar el desplazamiento de cada una y por tanto su velocidad. De ahí la importancia de llenar primero el depósito principal y sembrarlo con micro partículas reflectantes (*PRTCL – GLSS – HLLW MD=8 – 12, modelo 10089 de TSI*) y posteriormente conectar el depósito principal con el secundario para llenarlo, así en ambos depósitos encontremos un fluido con la misma densidad de partículas.

Por tanto, necesitaremos un láser YAG de pulsos (*Litron Lasers modelo NANO L200-15PIV*), este lleva asociado un controlador encargado del encendido o apagado del láser, del encendido de la bomba del sistema de control de temperatura, de abrir o cerrar el obturador (*shutter*) y de regular el nivel de intensidad del láser, también integra una torre (*LPU 550 de Litron Lasers*) que realiza la función de fuente de alimentación, control de temperatura de trabajo y medio de conexión con sistemas externos. Con la intención de redirigir el par de puntos de luz generados por el láser a una posición determinada se dispone de un brazo articulado. En la salida del brazo articulado debemos de colocar una serie de lentes (*1000 mm y -15 mm*) que a partir del punto de luz creen un haz, para que la iluminación sea óptima y satisfaga el rango espacial de nuestro experimento debemos de poner la salida del brazo articulado a una distancia 100 cm del orificio de salida. Cabe decir que dicho haz de luz debe de contener el eje de simetría axial del vórtice.

Necesitaremos también una cámara CCD (*TSI modelo 630159 POWERVIEW PLUS 4MP*) la cual colocaremos enfocando en una dirección perpendicular al plano generado por el láser, esta cámara dispondrá de un objetivo (*NIKON AF NIKKOR 35/F2D*) con el cual enfocaremos a la zona de interés en cuestión, del mismo modo haremos con el obturador, modificándolo en función de nuestro interés. Trabajaremos con una distancia focal de 55cm y un rango espacial de captación de 20 cm a lo largo del eje de simetría axial. La distancia focal y la distancia de la salida

del brazo articulado al orificio se manifiestan de manera esquematizada en la **Figura 10**, así mismo la instalación real es mostrada en la **Figura 11**.

La finalidad de la instalación es sincronizar la captación de imágenes con la generación de haces de luz procedentes del láser, para ello necesitaremos de un sincronizador (*TSI modelo 310035*).

Tanto la cámara como el sincronizador se encontrarán conectados a un ordenador donde se encuentra instalado el software *INSIGHT 3G*, gracias a él controlaremos la sincronización.

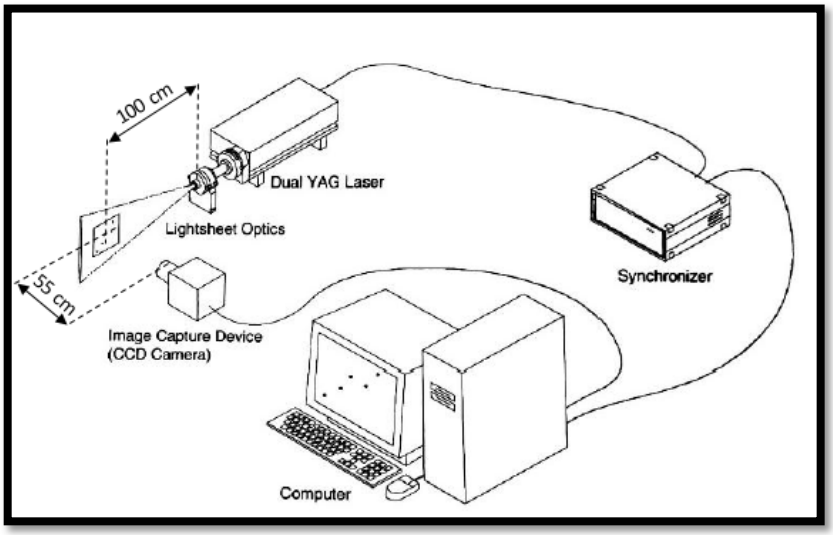

FIGURA 10: ESQUEMA DE LA INSTALACIÓN (TSI).

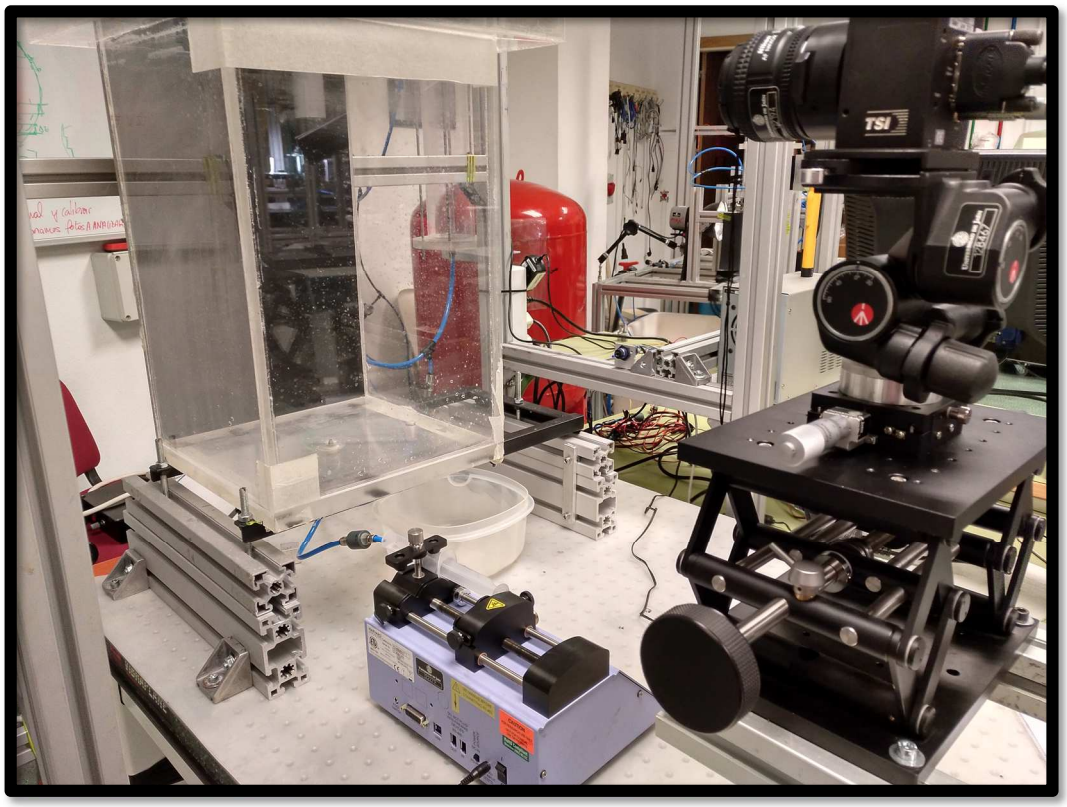

FIGURA 11: FOTOGRAFÍA DE LA INSTALACIÓN PARA LA CARACTERIZACIÓN DE VÓRTICES.

El modo de proceder será el siguiente:

En primer lugar, debemos de imponer las distancias geométricas establecidas anteriormente de 55 cm y 100 cm para la distancia focal de la cámara al orificio de salida y la distancia de la salida del brazo articulado a la tobera respectivamente. Por otro lado, es necesario garantizar que el plano generado por el láser contenga el eje de simetría axial. En una primera calibración es difícil de conseguir, por lo tanto, una vez puesto en marcha el láser debemos de lanzar varios haces de luz para ver por donde pasa el plano, reajustando la posición de la tobera en caso de no cumplir nuestros requisitos.

Al trabajar con un láser de alta potencia debemos de colocar en la entrada al laboratorio las señales de advertencia y colocarnos las gafas de seguridad (*FLEX SEAL, ARGON/NDGA: YAG de SPERIAN*). Una vez tomadas las precauciones pertinentes podremos encender el láser de pulsos, para ello seguiremos la guía de encendido de dicho láser.

Guía de encendido:

- Conectar la alimentación y encender el interruptor de la parte trasera de la torre (interruptor negro). Se iluminarán los *interlocks* (leds de la parte delantera) de *Simmer* y *Water Flow*. En este momento el agua de refrigeración comenzará a calentarse, esperar hasta que en el display se pueda observar 100.
- Girar la llave hacia la derecha para accionar el controlador.
- En el controlador, pulsar *System on* y *pump on*.
- Para accionar el láser, pulsar *Laser on.*
- Una vez establecidas todas las medidas de seguridad, podremos abrir el obturador *Open Shutter* y quitar las protecciones manuales tanto del láser como del brazo.
- Finalmente conectar el sincronizador y la cámara para poder iniciar el software *INSIGHT 3G*

Para apagar el equipo debemos de seguir los pasos establecidos anteriormente de manera inversa.

Los parámetros establecidos dentro del controlador serán un *Repetition Rata* de 10 y la *Energy/Volt* de 9 y 10 para el láser 1 y láser 2 respectivamente.

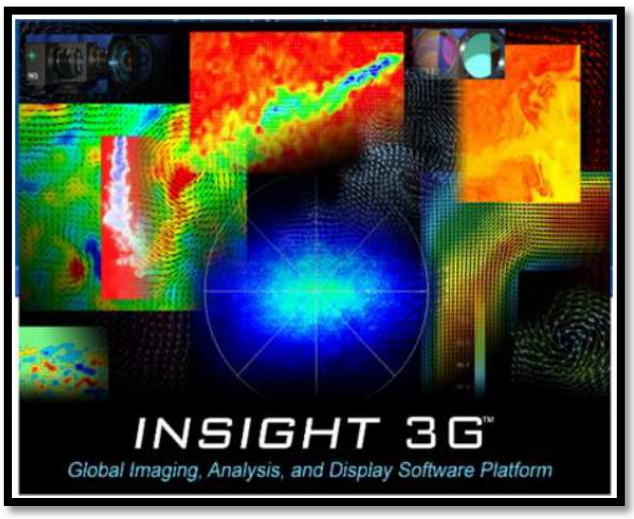

FIGURA 12: IMAGEN INICIAL DEL PROGRAMA INSIGHT 3G (INSIGHT 3G).

Una vez inicializados todos los elementos del sistema, procedemos a ejecutar el programa *INSIGHT 3G.* En primer lugar, debemos de crear un experimento nuevo para ello debemos de clicar en la ventana que aparece en el cuadro superior (*Experiment/Run > New Experiment*), una vez introducido el nombre, aparecerá en el árbol de experimentos (*Exp. Tree*), lo seleccionamos y debemos de crear una nueva rutina (*New Run*), aquí será donde debemos de establecer los parámetros necesarios para nuestro experimento.

Como medida preventiva y de comprobación, dentro de herramientas y a su vez en componentes (*Tools > Component Setup*), podremos ver un resumen (*Summary*) de todos los elementos exteriores vinculados al programa, también se podrá modificar la frecuencia de emisión del láser (*Laser Setup*), estando ésta limitada por la frecuencia de captación de la cámara, trabajaremos con una frecuencia de 14.5 Hz.

En el caso de faltar alguno de los elementos, tendremos que introducirlos en el apartado de hardware (*Tools > Hardware Setup*) como muestra la **Figura 13**.

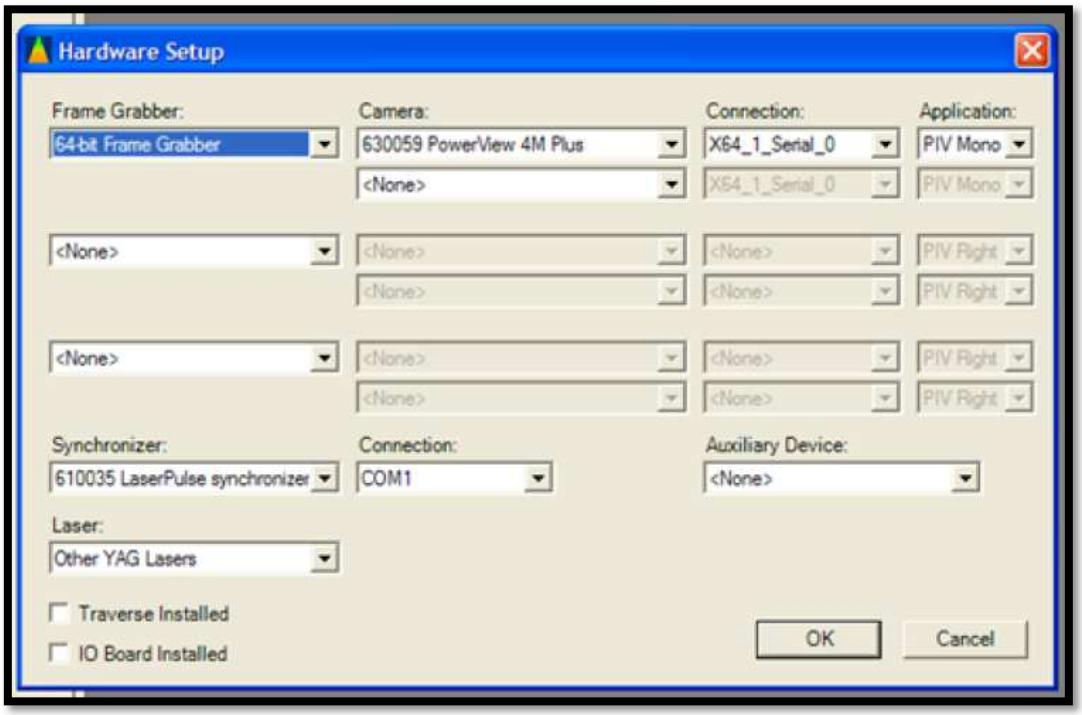

FIGURA **13**: *ELEMENTOS Y PUERTOS DE CONEXIÓN (INSIGHT 3G).*

Para garantizar la iluminación del plano gracias al láser en cada par de fotos tenemos que modificar los parámetros característicos que encontraremos en los ajustes del tiempo de captura (*Capture > Timing Setup*). Imponemos que *PIV Frame Mode* sea *Straddle* y que *Pulse Rep. Rate* sea el máximo, que en nuestro caso será 7.25Hz, la mitad de la frecuencia de emisión. Con el resto de valores debemos de jugar hasta garantizar que tenemos un pulso de láser por tiempo de exposición de la cámara. En nuestro caso, se ha trabajado con un *Laser Pulse Delay* de 400µs, un *Delta T* de 1000µs y un *PIV Exposure* de 500µs.

Dentro del menú de captura, aparecen tres ventanas en cascada, *Application, Exposure* y *Capture*, la primera debe de estar siempre en PIV, y el resto las cambiaremos así sea nuestro

interés. Si lo que deseamos es obtener la imagen de calibración y para ello queremos ver de manera continua la imagen obtenida a través de la cámara, en *Exposure* pondremos *Free* y en *Capture, Continuous,* así podremos enfocar al objeto de referencia (regla métrica) situado en el plano de análisis. En el caso de querer solamente una imagen, en *Capture* pondremos *Single.*

En los casos anteriores sólo se ha tenido en cuenta el funcionamiento de la cámara, desvinculada al láser, si bien el fin de esta instalación es sincronizar un elemento con el otro, para ello en *Exposure* estableceremos *Synchronized* y en *Capture, Sequence.* Para cualquier caso el inicio de la operación se realizará al pulsar el botón de *Capture,* diferenciado por tener un dibujo de una cámara, también podremos establecer el número de imágenes a tomar, seleccionando un botón donde se aprecia un dibujo de una carpeta, en nuestro experimento realizaremos 30 pares de fotos por caso, aunque posteriormente descartaremos aquellas en las que no se encuentre nuestro vórtice seccionado.

Este último caso se debe de desarrollar a oscuras para que la cámara solamente detecte las partículas reflectantes debido a la incidencia del plano del láser. A la vez que se esté obteniendo imágenes se ha de generar un vórtice, para ello accionaremos el pulsador que puentea la señal y permite la apertura de la electroválvula, la repetición de cada caso debe de ir acompañada de un tiempo de reposo, para que se difundan las perturbaciones generadas por el vórtice anterior.

Una vez tomadas las imágenes debemos de guardarlas (*Save RAM Images*) para posteriormente poder procesarlas, para ello recurriremos a dos *GUI* (*Graphical User Interface*) desarrolladas en Matlab: *PIVlab* y *Caract_Vortex.*

PIVlab

PIVlab es una aplicación de código libre desarrollada por William Thielicke basada en el análisis, validación, postprocesado, visualización y simulación de datos de *PIV.*

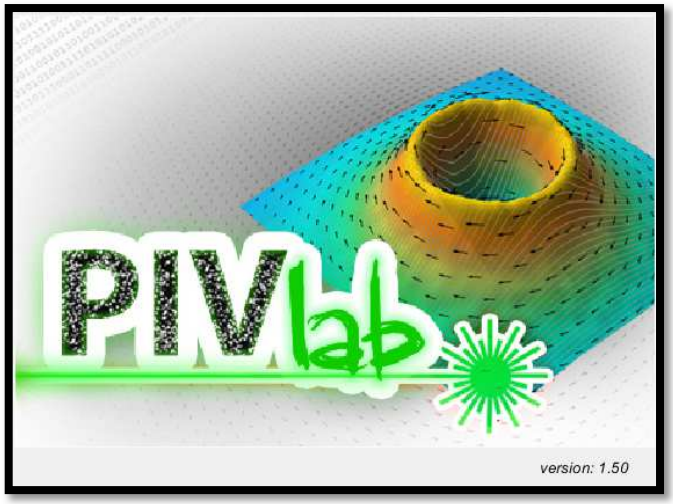

FIGURA 14: LOGO DEL PROGRAMA PIVLAB (PIVLAB).

Esta aplicación tiene una interfaz muy intuitiva, lo primero que tenemos que hacer es cargar la imagen de calibración (*Load images*), seleccionamos dos veces la imagen, previamente habiendo deshabilitado la eliminación del duplicado de imágenes (*Remove duplicates*). Al importar la imagen (*Import*), nos la mostrará en la pantalla, esta primera solo nos servirá para

establecer la relación de píxeles con milímetros y el paso de tiempo, por tanto, nos dirigimos a calibración y seleccionamos una distancia de referencia (*Calibration > Select reference distance*), tendremos que establecer el valor en milímetros de la línea generada a partir de los dos puntos determinados con el ratón, igualmente tendremos que establecer el paso de tiempo (*time step*), para nosotros 1ms.

A continuación, procedemos a cargar las imágenes generadas por *INSIGHT 3G.* No hay que cerrar la sesión, estableceremos una sesión nueva, de esta manera conservaremos las condiciones de calibración establecidas (*File > New sesión > Load images*), para este caso habilitaremos la eliminación de duplicados.

Una vez cargadas las imágenes, tenemos que establecer el área de interés de estudio, esto será imposible sin que previamente pre procesemos la imagen (*Analyses settings > Image pre-procesing > Preview current frame*). En este apartado se encuentra habilitado por defecto *CLAHE* (*Contrast Limited Adaptive Histogram Equalization*) **[11]**. *CLAHE* recurre a histogramas implantandos en regiones reducidas de la imagen analizada (*tiles*), en nuestro caso se utilizarán *tiles* de 20 px de tamaño, siendo este tamaño el adecuado, ya que tenemos un sembrado de partículas e iluminación óptima. Las intensidades más frecuentes del histograma son resaltadas frente al resto de datos de la imagen, consiguiendo así recalcar en nuestro caso el reflejo de todas las partículas. Tras la realización de los histogramas para cada *tile*, se usan interpolaciones bilineales entre *tiles* vecinos para eliminar los contornos de cada *tile*.

Para seleccionar el área de interés simplemente tendremos que generar un rectángulo que encierre dicha área (*Analyses settings > Exclusions > Select ROI*). Finalmente, solo queda analizar (*Analysis > Analyze all frames*), esto generará el campo vectorial de velocidades y sus derivados. Previo al análisis, podremos seleccionar el algoritmo a usar en la correlación de pares de imágenes (*Analyses settings > PIV settings > PIV algorithm*), utilizaremos el que viene por defecto, *FFT windows deformation*, este utiliza *FFT* (*Fast Fourier Transform*), multipasos y deformación de las ventanas **[11]**, teniendo un alto coste computacional pero mejores resultados. Al usar *FFT* debemos de tener en cuenta el área de interrogación (*Interrogation area*) en cada uno de los pasos y del mismo modo, el número de pasos impuestos, ya que al introducir más pasos el área de interrogación se irá reduciendo. Esta área debe de ser mayor que el desplazamiento de la partícula en cada par de *frames*, si no se cumpliera, la matriz que correla invertiría la posición de la partícula **[11]**. De esta manera, sólo con 2 pasos, el primero de 64 px y el segundo de 32 px, tendremos cubierto dicho problema. Para homogeneizar la solución y que todo punto tenga asociado un vector de velocidad tenemos que postprocesar las imágenes, interpolando los datos que puedan faltar (*Post processing > Vector validation > Interpolate missig data > Apply to all frames*). Toda la información la guardaremos en formato *MAT-file* incluyendo también la información derivada (*File > Save > MAT-file > Include derivatives > Save all frames*). Solo nos serán válidos aquellos archivos generados asociados a imágenes que contengan el vórtice, el resto de archivos se eliminarán.

Caract_Vortex

Para postprocesar de manera automatizada todos los archivos procedentes de *PIVlab,* se ha implementado un programa, con el cual se podrá calcular la circulación, diámetro del vórtice, radio del núcleo, posición del núcleo en función del tiempo y por tanto la velocidad de avance del vórtice, todos estos cálculos se manifestarán en el documento *Excell* que genera el programa.

El programa dispone de una interfaz gráfica intuitiva (ver en la **Figura 15**), en donde podremos editar las propiedades según el caso que estemos analizando. Dentro de las propiedades encontraremos:

- Escala: valor que afectará en la escala de visualización del campo de velocidades para casa archivo analizado.
- Densidad del agua en condiciones normales.
- Viscosidad del agua en condiciones normales.
- Tiempo de apertura de la electroválvula.
- Tiempo entre avance: será el valor inverso de la frecuencia de captación de imágenes, en nuestro caso es de 1/7,25Hz.
- Presión: Presión del depósito secundario presurizado.
- Diámetro de la tobera.

También se podrá modificar el nombre del archivo *Excell* generado.

FIGURA 15: INTERFACE DEL PROGRAMA CARACT_VORTEX.

En el caso de tener alguna duda del funcionamiento, el programa integra una venta de ayuda, donde se explica paso a paso lo que se debe de hacer para el buen funcionamiento del mismo. El contenido del menú de ayuda se establece en la **Figura 16.**

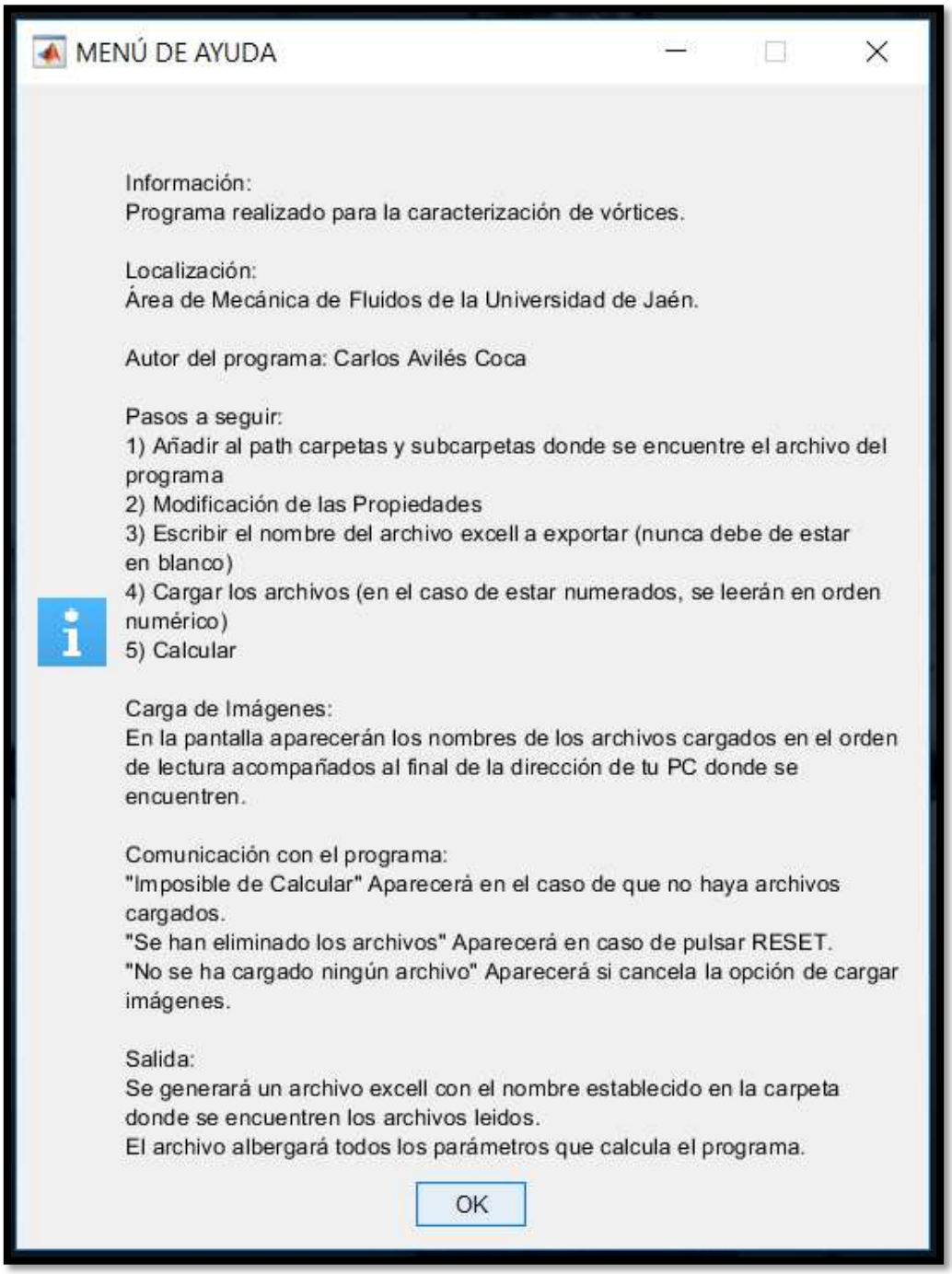

FIGURA 16: PESTAÑA DE AYUDA DEL PROGRAMA CARACT_VORTEX.

Además del archivo Excell, se generarán una serie de gráficas adicionales mostradas en la **Figura 17**. Hablamos del campo vectorial de velocidades, la vorticidad y velocidad axial a lo largo del eje longitudinal que pase por el núcleo, la posición del núcleo, el diámetro del vórtice y la

circulación, esta última calculada a través de una línea cerrada y de la superficie que encierra dicha línea para ratificar el *Teorema de Stokes*.

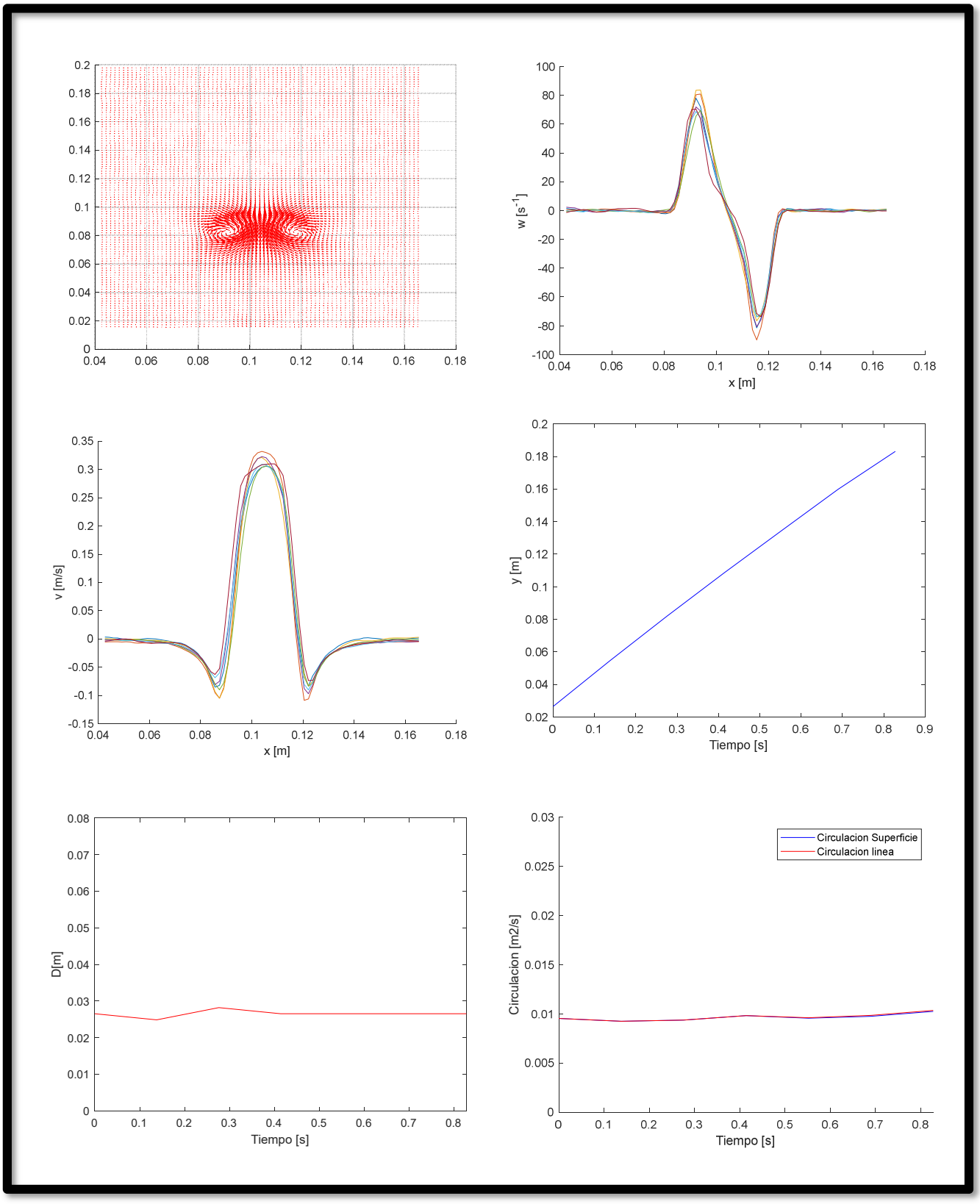

FIGURA 17: EJEMPLOS DE GRÁFICAS GENERADAS EN EL PROGRAMA CARACT_VORTEX.

ANÁLISIS DE BURBUJAS:

Para la captación de imágenes de burbujas, recurriremos a la misma instalación utilizada para la captación de vórtices, pero en este caso desactivaremos el láser, jugando solo con la cámara. Para este estudio cambiaremos de objetivo (*105mm F2.8 DG MACRO* de *SIGMA*) y de distancia focal, en este caso hablamos de 18cm, dejando un rango espacial de 15mm a lo largo del eje de simetría axial. Para obtener la imagen de calibración procederemos de la misma forma ya expuesta en el apartado de análisis de vórtices, pero en este caso enfocando a un plano que seccione a la aguja que genera la burbuja como muestra la **Figura 18**. En cuanto al parámetro a modificar dentro del programa *INSIGHT 3G* será *PIV Exposure* que podremos encontrar dentro de *Timing Setup,* el resto de parámetros los mantendremos iguales, ya que solo afectarán a la emisión de haces de luz por parte del láser y este se encontrará apagado. Para la captación de imágenes donde se pueda distinguir el contorno necesitaremos de un sistema de iluminación externa (*Torch PRO-X 1250W de Multiblitz*), enfocado a un papel blanco pegado a la urna en el lado opuesto al que enfoca la cámara, con la finalidad de homogenizar la iluminación.

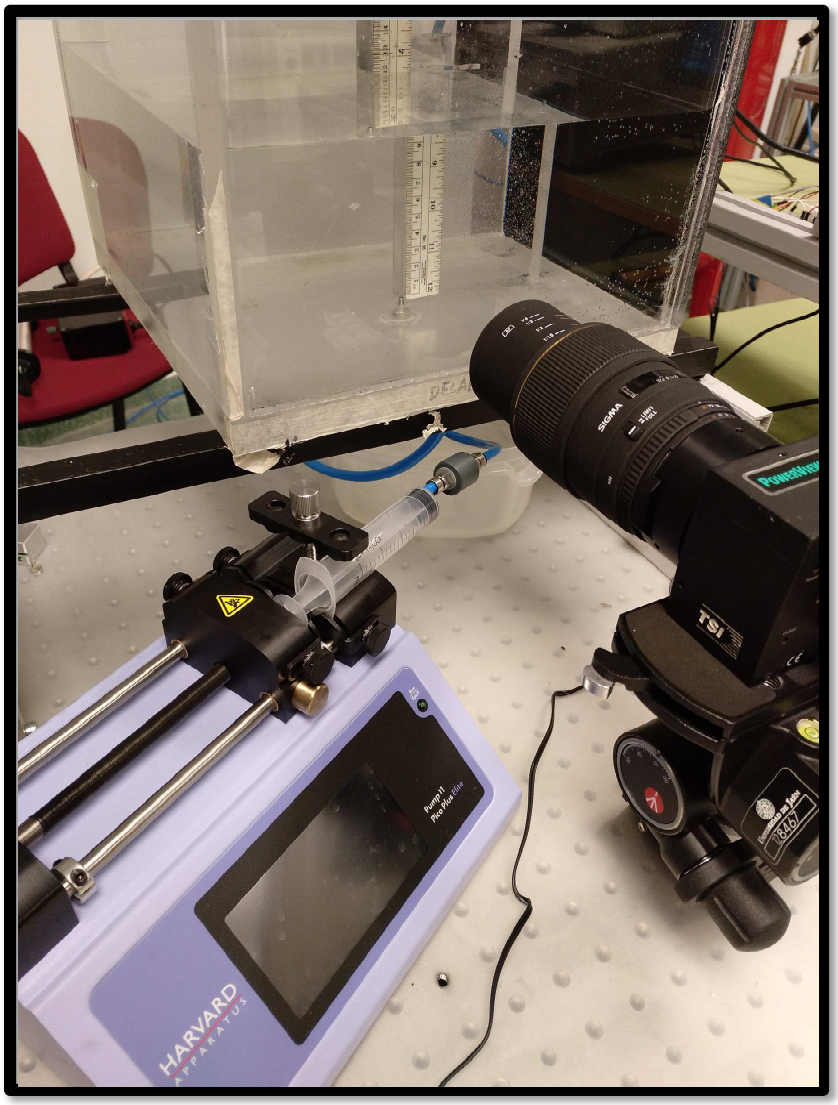

FIGURA 18: FOTOGRAFÍA DE LA INSTALACIÓN PARA LA CARACTERIZACIÓN DE BURBUJAS.

Según el valor que introduzcamos en *PIV Exposure* necesitaremos más potencia o menos por parte de la iluminación externa, se ha comprobado que el rango de operación está entre 100-5000µs, para mi distancia focal trabajaremos con un *PIV Exposure* de 300µs con una alta potencia de iluminación.

Para la captación de imágenes, debemos de proceder como si la cámara estuviera sincronizada con el láser, no obstante, éste lo tenemos apagado gracias al controlador del mismo (*Laser off*). Mientras que se estén capturando imágenes, se deben generar burbujas a través de la bomba de jeringa.

De los pares de fotos, solo nos servirá el *frame A,* ya que el B satura completamente. Aquellas imágenes operativas, las procesaremos a través de un programa que hemos desarrollado en Matlab para el análisis de burbujas.

CARACTBUBBLES

Debida a la necesidad de post procesar imágenes de burbujas, se ha implementado un programa que realiza dicha función de manera automatizada. El programa dispone de una interfaz gráfica a través de la cual nos permitirá cargar imágenes de calibración y calibrar, como también determinar las propiedades de una imagen que contenga una burbuja.

La principal finalidad del programa será determinar los principales parámetros geométricos de la burbuja, teniendo con fin fundamental la detección del contorno de la misma. El proceso empleado lo explicaremos a través de un ejemplo.

Partiremos de una imagen cualquiera, en nuestro caso una imagen de una burbuja (**Figura 19**).

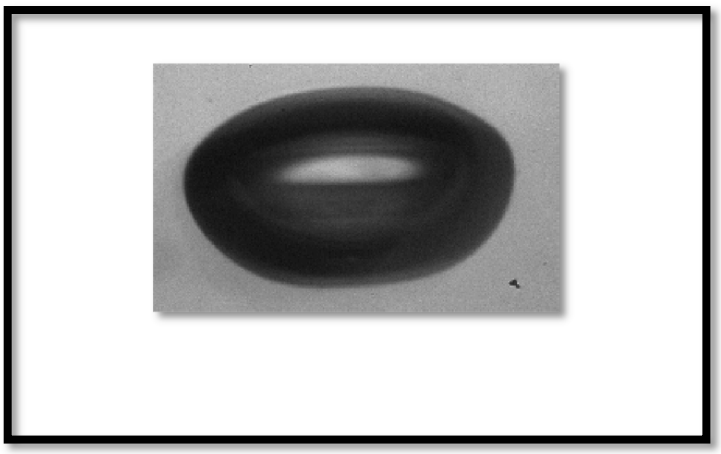

FIGURA 19: FOTOGRAFÍA DE UNA BURBUJA.

El paso siguiente será binarizar la imagen, quedando definida solamente a partir del blanco y el negro. Para esto, establecemos el umbral de la imagen dentro de la escala de grises, parámetro necesario en la binarización.

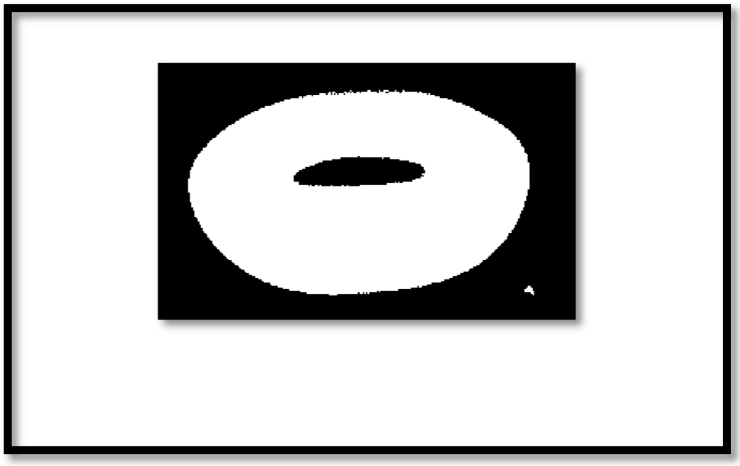

FIGURA 20: FOTOGRAFÍA DE UNA BURBUJA BINARIZADA.

Como podemos observar al binarizar la imagen, la **Figura 20**, suelen aparecer elementos no deseados que impiden el correcto análisis de la imagen, por lo que nos vemos obligados a filtrarla. Los filtros utilizados consisten en la elimiación elementos blancos detectados de pequeña superficie (menos de 30 px) y en el vaciado de toda superficie blanca, es decir se elimina toda superficie negra contenida dentro de una superficie blanca.

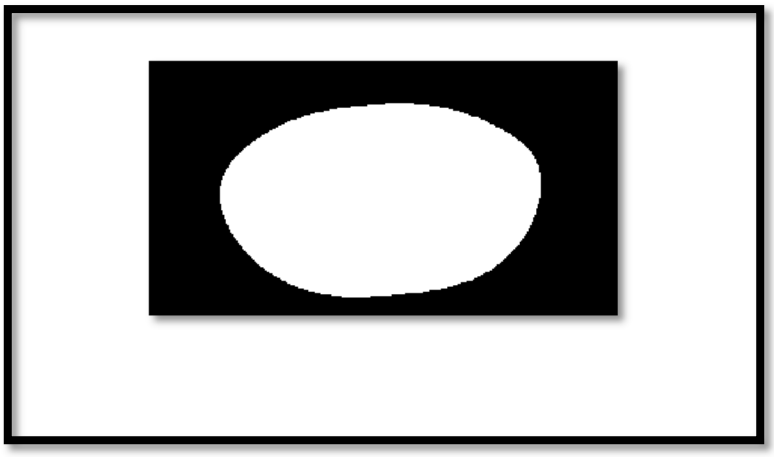

FIGURA 21: FOTOGRAFÍA DE UNA BURBUJA FILTRADA.

Finalmente tendremos una imagen limpia (**Figura 21**), a partir de la cual se podrá calcular fácilmente el contorno del elemento analizado en cuestión, dicho contorno se puede apreciar en rojo junto con la imagen de partida en la **Figura 22.**

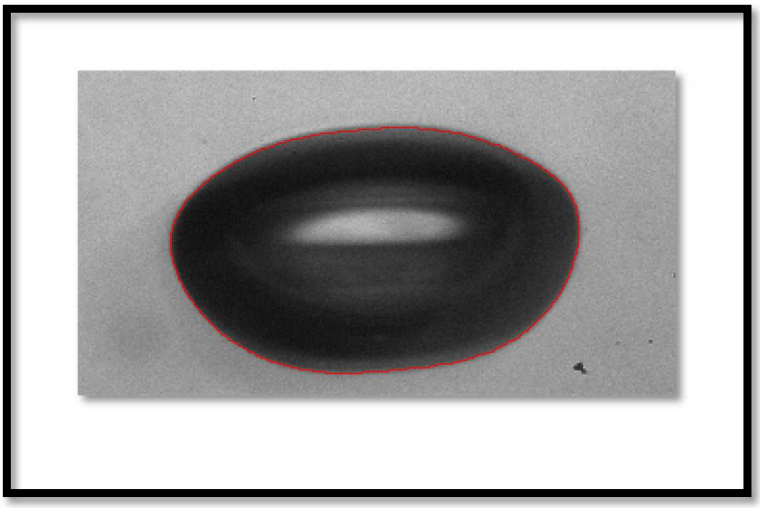

FIGURA 22: FOTOGRAFÍA DE UNA BURBUJA Y SU CONTORNO.

El contorno se guardará en dos vectores que definirán la posición de cada punto que lo conforma, trabajando con dichas coordenadas estableceremos las propiedades geométricas: la superficie y el volumen.

El concepto utilizado para la determinación de estos parámetros, será la de integral numérica. Para determinar el área, usaremos la sumatoria de áreas sencillas (rectángulos) y para el volumen, la sumatoria de volúmenes sencillos de revolución (cilindros).

A una altura cualquiera del eje de ordenadas de la imagen, que seccione al elemento, podremos establecer los dos puntos que a dicha altura limitan la superficie a analizar, si calculamos la distancia que hay entre ellos, tendremos la base de nuestro rectángulo o el diámetro de la base del cilindro para dicha altura. Si realizamos esto para todas las ordenadas del elemento tendremos una función de puntos que varía según el eje de ordenadas, pudiendo operar con ella. Si dicha función la integramos, tendremos un área y si la revolucionamos un volumen.

$$V = \int \pi R^2 dy \approx \sum_{i\,=\,y_{min}}^{i\,=\,y_{max}} \pi R_i^2 \Delta y \qquad [\text{Ec. } 26]$$

$$A = \int R dy \approx \sum_{i=\,y_{min}}^{i=\,y_{max}} R_i \Delta y \qquad [\text{Ec. } 27]$$

Siendo R_i la función de puntos que representa la mitad de las distancia entre los dos puntos asociados a cada ordenada del elemento, Δy es el paso de maya del eje de ordenadas, y_{min} la ordenada mínima e y_{max} la ordenada máxima del elemento.

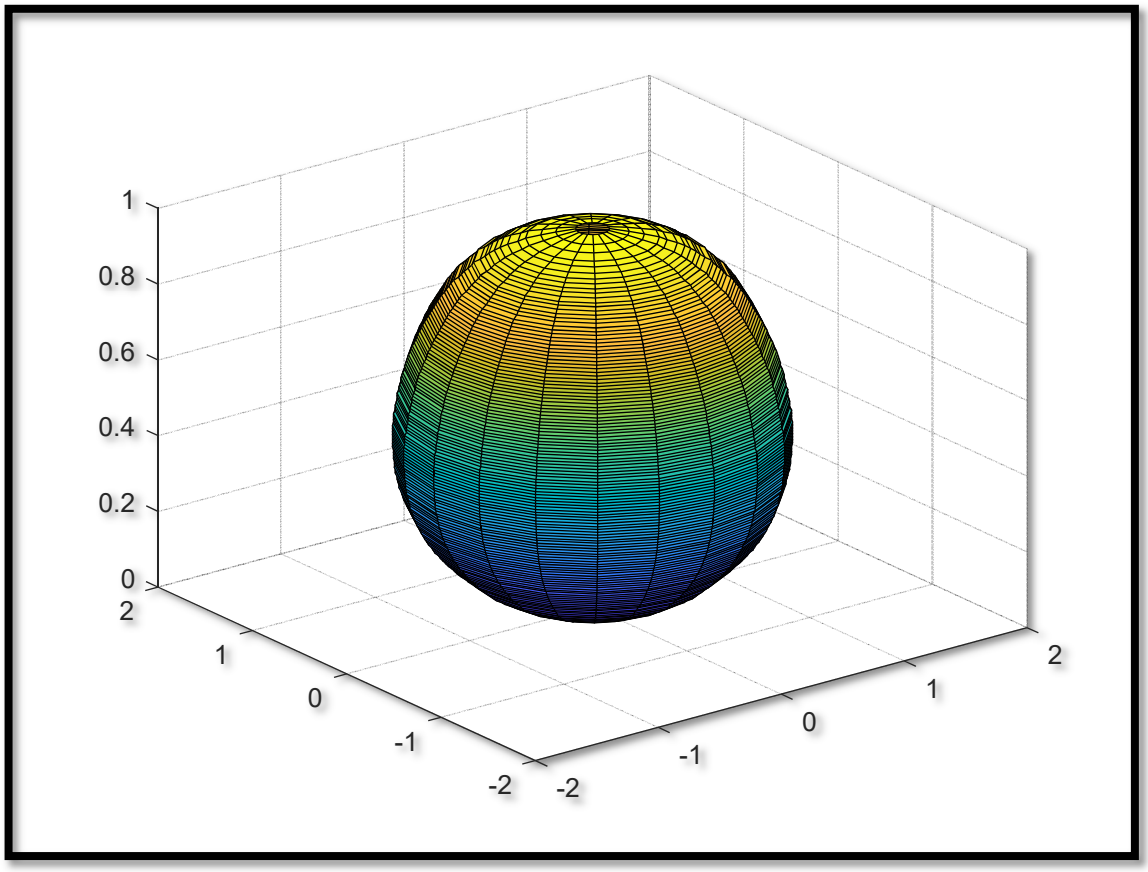

FIGURA 23: VOLUMEN DE LA BURBUJA GENERADO A PARTIR DEL CONTORNO Y LA SUPERPOSICIÓN DE CILINDROS.

En la **Figura 23** se pueden observar la superposición de cilindros que finalmente darán volumen al elemento analizado.

En lo referido a la interface del programa (**Figura 24**), en primer lugar, tendremos que obtener la escala de calibración, la relación entre los milímetros y los píxeles, para ello cargaremos la imagen de calibración y seleccionaremos una distancia vertical con el ratón determinando su equivalencia en milímetros, esta operación se debe de repetir hasta que deje de salir la imagen de calibración.

En segundo lugar, cargaremos la imagen de la burbuja y calcularemos las propiedades, en este instante deberemos de seleccionar el área de interés, señalando dos puntos (de izquierda a derecha) que generen un rectángulo que encierre la burbuja, finalmente arrojará los resultados en la ventana principal del programa.

También será capaz de determinar las propiedades críticas de *Fritz,* junto con números adimensionales que caracterizarán la burbuja (*Ga* y *Bo*), para ello debemos de modificar las propiedades antes de calcular, así como el diámetro de la aguja.

Este programa también se utilizará en la interacción vórtice con burbuja, este caso se desarrollará posteriormente.

FIGURA 24: INTERFACE DEL PROGRAMA CARACTBUBBLES.

En el caso de tener alguna duda con el funcionamiento del programa, se dispone de una ventana de ayuda (ver **Figura 25**) donde se exponen todos los pasos a seguir para el buen funcionamiento del programa.

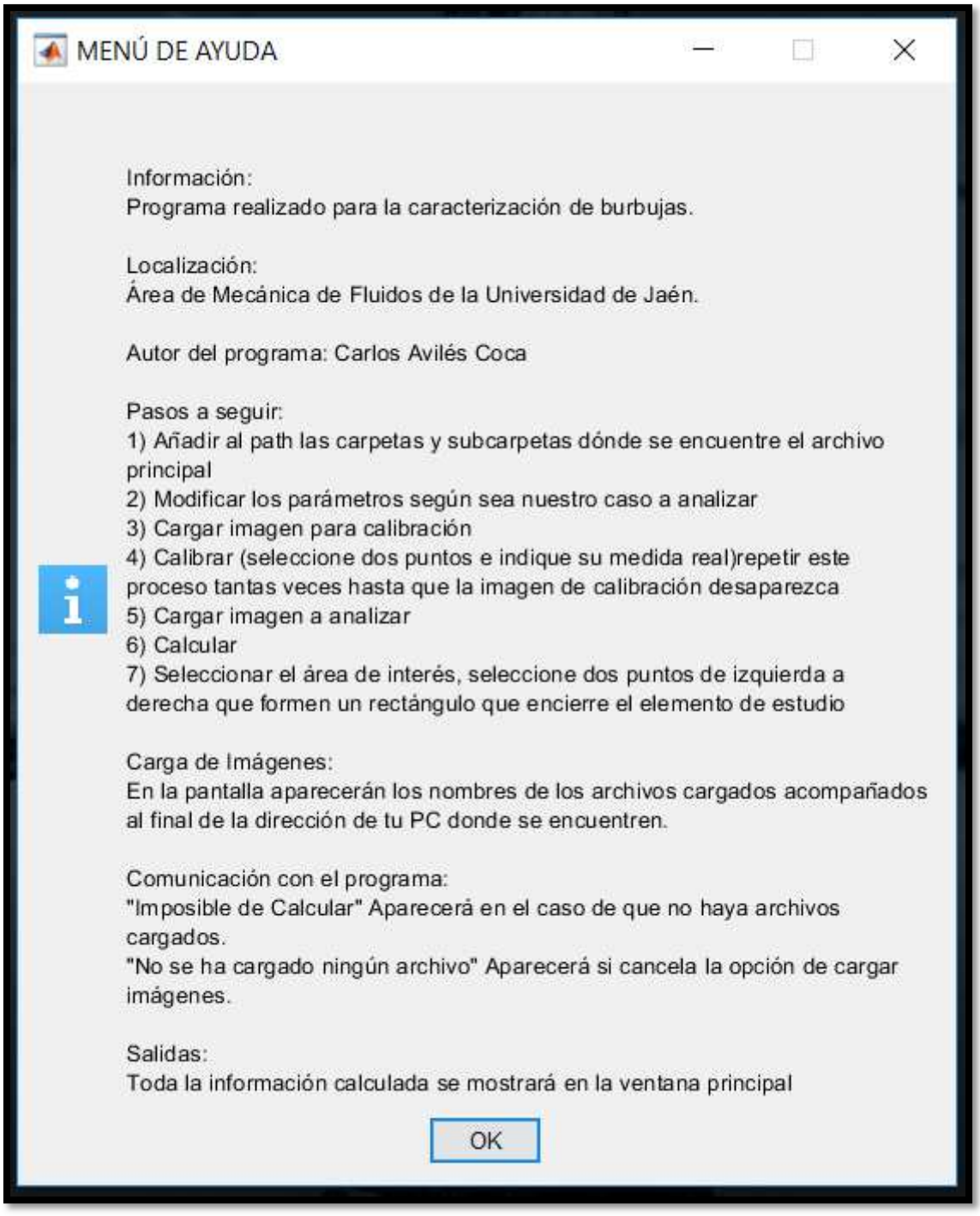

*FIGURA 25: PESTAÑA DE AYUDA DEL PROGRAMA **CARACTBUBBLES**.*

ANÁLISIS DE LA INTERACCIÓN DE BURBUJAS CON ANILLOS DE VORTICIDAD:

En este apartado se utilizará las instalaciones hidráulicas y neumáticas junto con el sistema electrónico descrito anteriormente, apoyados con un sistema de captación de imágenes.

Para la interacción debemos imponer el alineamiento entre la tobera y la aguja, para que tanto el vórtice como la burbuja en su generación compartan el mismo eje de simetría axial. En este tipo de análisis, el detector fotodiodo, cumplirá un papel importantísimo. Será el encargado de detectar la salida de la burbuja con el fin de que la interacción se dé en el rango espacial deseado, para ello se establecerá un *delay* entre la generación de la burbuja y la generación del vórtice.

Analizar la interacción a simple vista nos dará una idea de lo que nos encontraremos, si bien no es posible detectar el comportamiento de la burbuja, por lo que debemos recurrir a grabaciones de vídeos de alta velocidad. Este equipo estará integrado por una cámara de alta velocidad (*Fastcam APX RS* de *Photron*) con un objetivo (*105mm F2.8 DG MACRO* de *SIGMA*) y una iluminación de panel led, estando ambos elementos enfrentados. Para homogeneizar la iluminación nos ayudaremos de un folio en flanco pegado a la superficie en la que incide directamente la iluminación. Todos los elementos utilizados se muestran en la **Figura 26.**

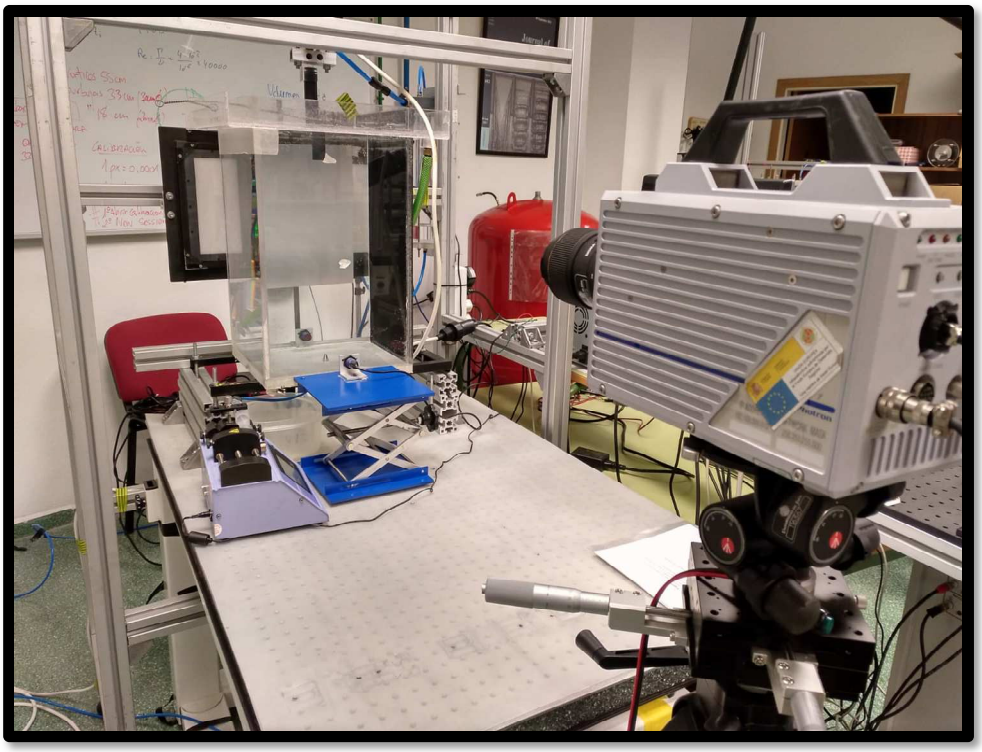

FIGURA 26: FOTOGRAFÍA DE LA INSTALACIÓN PARA EL ANÁLISIS DE LA INTERACCIÓN.

Este tipo de cámaras tienen asociado un programa específico para su control, en nuestro caso será *PFV* (*Photron Software*). Dentro de este programa estableceremos la velocidad de grabación y el *shutter*, el programa nunca funcionará si la cámara no se encuentra encendida (*POWER ON*) antes de lanzar el programa.

Para establecer los parámetros característicos de grabación se debe de proceder de la misma manera. En primer lugar, taparemos el objetivo, encenderemos la cámara y lanzaremos *PFV*.

Una vez el programa se haya abierto, en la derecha encontraremos una serie de pestañas con las que podremos interaccionar con el programa, tendremos que seleccionar *Camera.* Dentro de esta pestaña primero tendremos que seleccionar la velocidad de grabación (*Frame Rate*), en nuestro caso trabajaremos con 1000 fps, no sería conveniente aumentar la velocidad de grabación puesto que la cámara tiene una memoria limitada y esto reduciría el tiempo de grabación, además, tendríamos que aumentar la luminancia del sistema de iluminación.

En segundo lugar, dentro también de *Camera*, estableceremos las propiedades del *Shutter,* en nuestro caso 1/8000 sec. Siempre que modifiquemos alguno de los dos parámetros anteriores debemos de calibrar, haciendo hincapié en que se debe de hacer con el objetivo tapado (*Camera > Shading > Calibrate*).

Establecidos los parámetros característicos, procederemos a enfocar con el objetivo el plano deseado, para ello colocaremos una regla milimetrada, fijando el objetivo cuando dicha regla sea vista con nitidez. En el caso de querer iniciar una grabación, dentro de *Camera* seleccionaremos *Record.* Cabe destacar que para facilitar la grabación nos ayudaremos de un pulsador que iniciará la grabación cuando este sea pulsado, debido a esto tendremos que establecer dicha condición dentro del programa (*Camera > Trigger Mode > Start*).

Finalizada la grabación el programa nos dirigirá automáticamente a la pestaña *Data Save* donde podremos establecer el nombre del archivo (*File name*), dónde guardarlo (*Save path*), en qué formato lo queremos (AVI) e incluso recortar el video según nos interese.

Finalmente, cuantificaremos el número adimensional característico de la interacción, el *Weber*, utilizaremos de nuevo el programa *CARACTBUBBLES*, este calculará el *We* en función del radio equivalente calculado, las propiedades establecidas y la circulación del vórtice con el que se encuentre. Esto nos servirá de referencia para saber el entorno de trabajo, no obstante, también se calculará con las propiedades promedias.

RESULTADOS OBTENDIOS:

CARACTERIZACIÓN:

En este apartado se establecerán los valores característicos tanto de los vórtices como de las burbujas analizadas.

VÓRTICES:

La generación de vórtices se ha realizado siempre con una presión de 6 bar en el depósito secundario, el cual se encuentra presurizado, cambiando en cada caso el diámetro del orificio de salida y el tiempo de apertura de la electroválvula.

A continuación, se representarán por separado la circulación en función del desplazamiento axial, el desplazamiento en función del tiempo y la circulación adimensional en función del desplazamiento adimensional. Finalizando con un resumen que aunará todos los datos obtenidos.

Orificio de salida de Ø20 mm:

Para el orificio de salida de 20 mm de diámetro se han analizado seis casos iguales para un mismo tiempo de apertura de la electroválvula, obteniendo:

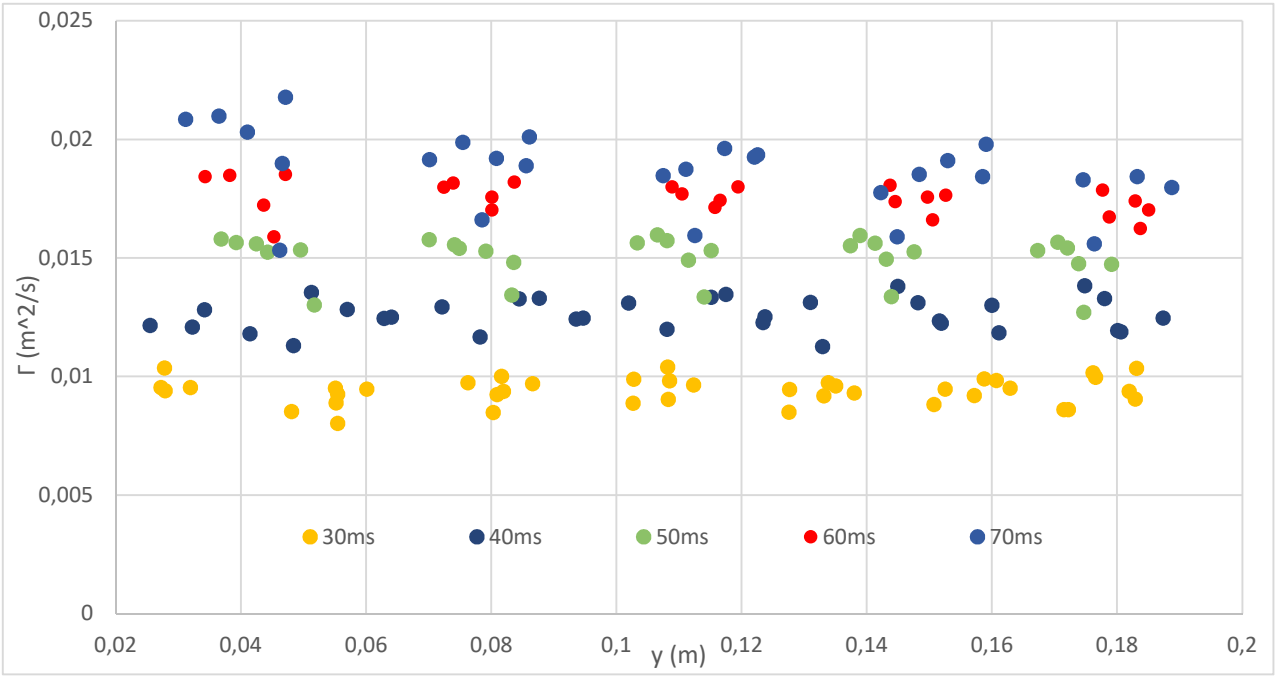

FIGURA 27: CIRCULACIÓN EN FUNCIÓN DEL DESPLAZAMIENTO AXIAL PARA DIFERENTES TIEMPOS DE APERTURA DE LA ELECTROVÁLVULA CON UN ORIFICIO DE SALIDA DE $\emptyset 20$ **mm.**

Podemos observar en la **Figura 27** que la circulación para cada caso se mantiene constante, aumentando con forme aumentamos el tiempo de apertura. No se observa un aumento de la circulación al inicio ni una disminución al final de la región analizada ya que se han despreciado los casos donde le vórtice no se encontrase totalmente desarrollado.

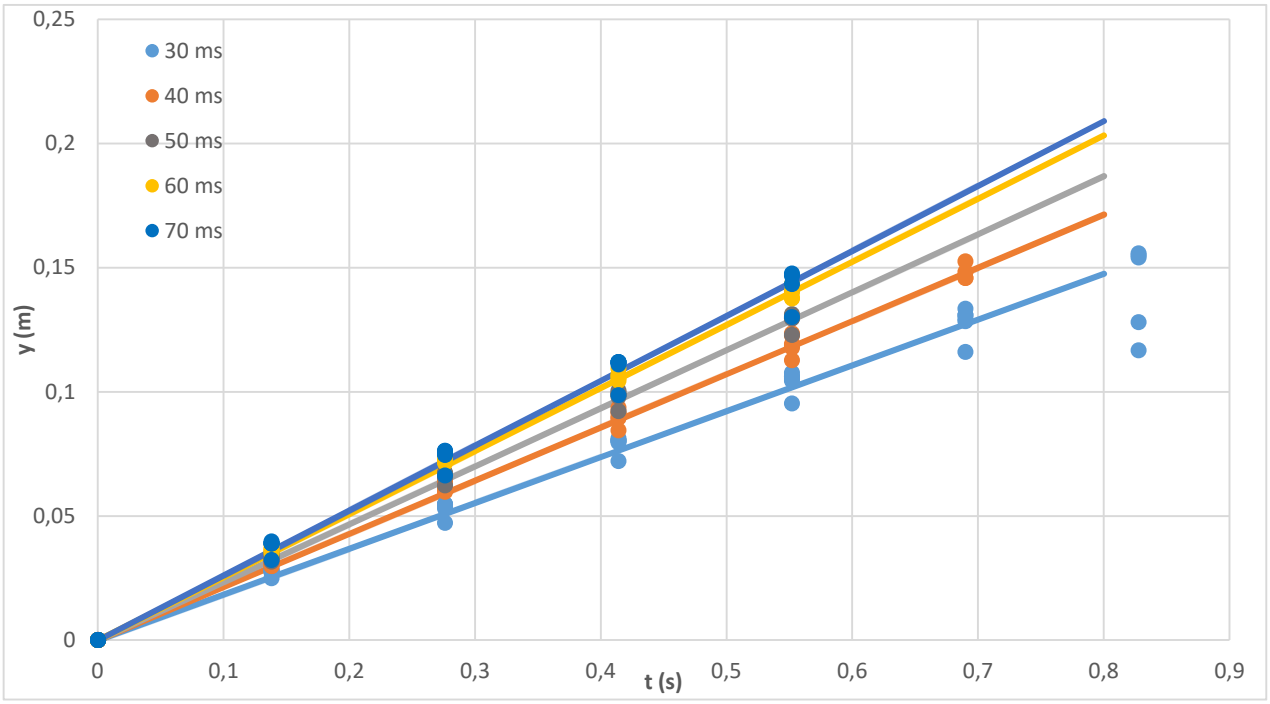

FIGURA 28: DESPLAZAMIENTO AXIAL EN FUNCIÓN DEL TIEMPO PARA PARA DIFERENTES TIEMPOS DE APERTURA DE LA ELECTROVÁLVULA CON UN ORIFICIO DE SALIDA DE $\emptyset 20$ **mm**.

Las rectas representadas en la **Figura 28**, son el reflejo de la velocidad promedia de todas las repeticiones de cada tiempo de apertura, apreciándose como a medida que aumentamos el tiempo de apertura, el vórtice avanza como mayor velocidad, la pendiente va aumentando.

El hecho de que todos los casos partan del origen, es debido a que se ha tomado como punto de referencia al primer caso analizado, manifestando una posición relativa. Esto no supone ningún problema ya que, analizando la posición desde un punto relativo o absoluto, la velocidad se mantiene constante.

Si queremos comparar los diferentes casos debemos de recurrir a números adimensionales, en nuestro caso representaremos la circulación adimensional, definida anteriormente, frente un desplazamiento adimensionalizado con el diámetro del anillo de vorticidad.

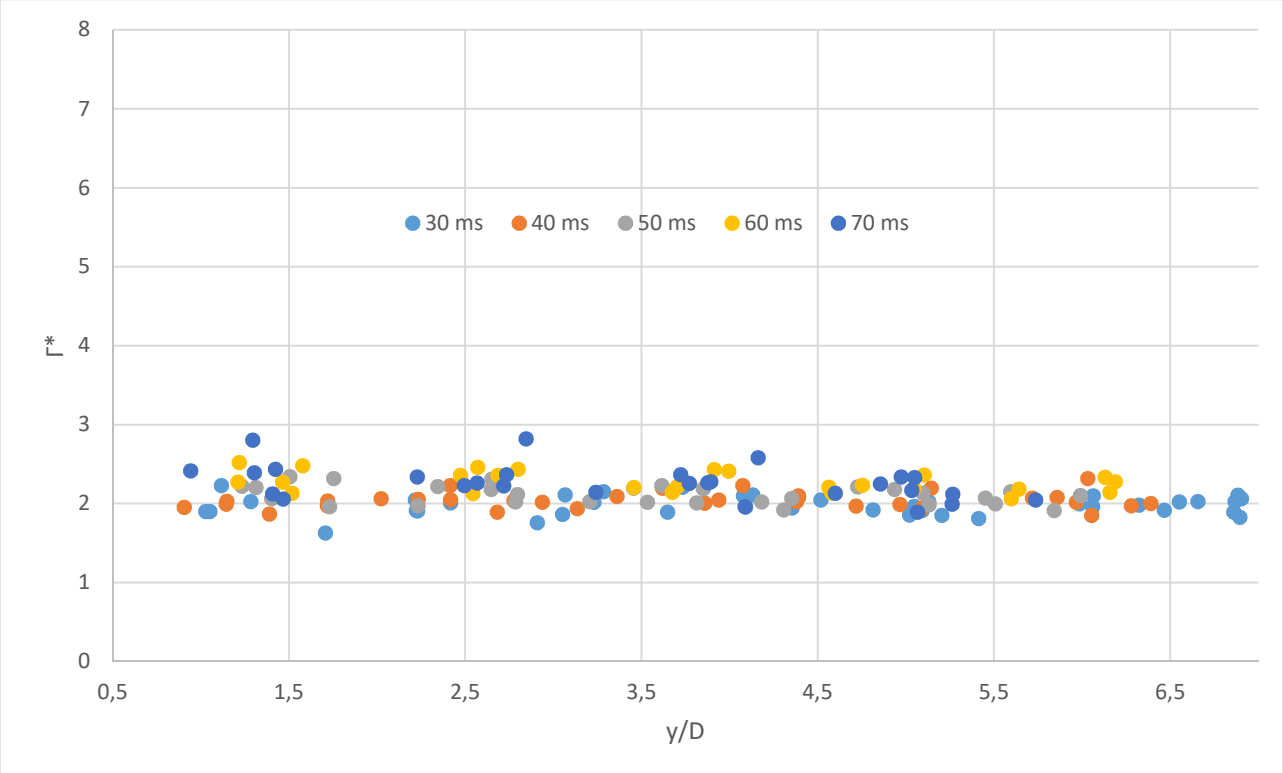

FIGURA 29: CIRCULACIÓN ADIMENSIONAL EN FUNCIÓN DEL DESPLAZAMIENTO AXIAL ADIMENSIONAL PARA DIFERENTES TIEMPOS DE APERTURA DE LA ELECTROVÁLVULA CON UN ORIFICIO DE SALIDA DE Ø20 mm.

La circulación adimensional representada en la **Figura 29**, se mantiene constante entorno al valor 2, manifestando la semejanza entre todos los casos analizados. A diferencia de la **Figura 28**, en las **Figuras 27** y **29**, el desplazamiento axial adimensionalizado con el diámetro del anillo de vorticidad, es absoluto.

Orificio de salida de Ø30 mm:

Para el orificio de salida de 30 mm de diámetro, el número de casos analizados como la representación de los datos obtenidos ha sido de manera análoga al caso anterior de 20 mm, planteando los mismos razonamientos.

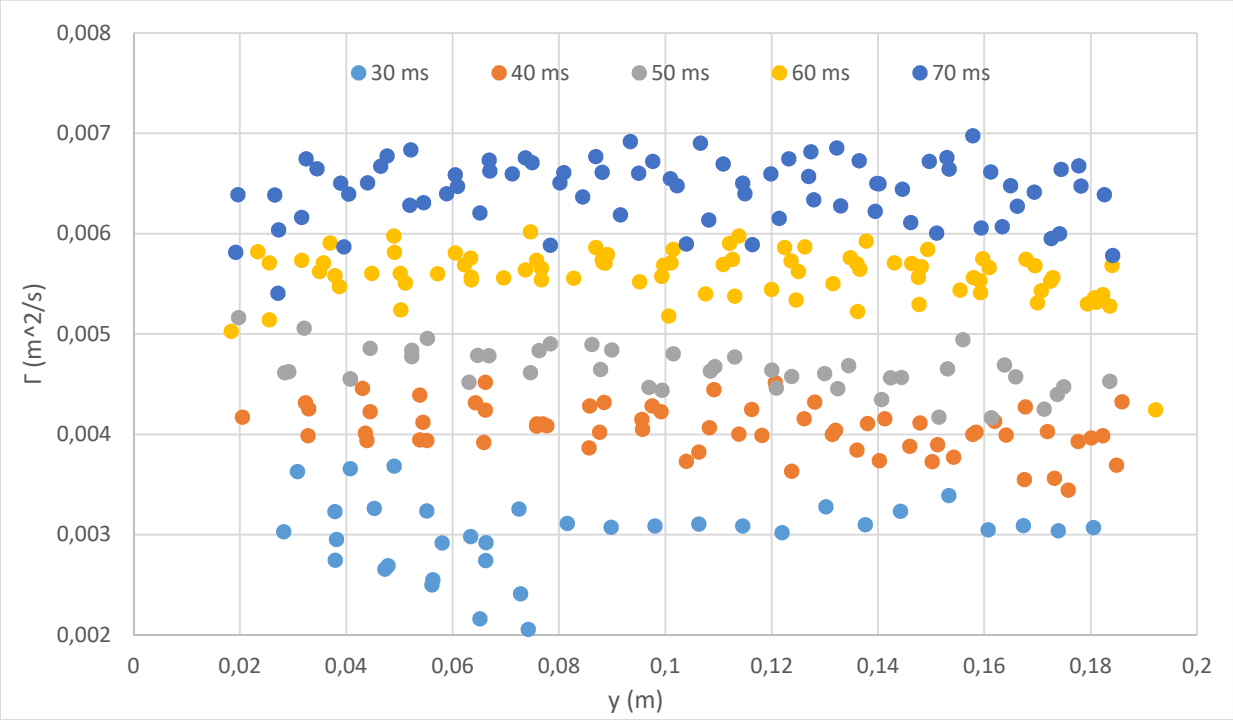

FIGURA 30: CIRCULACIÓN EN FUNCIÓN DEL DESPLAZAMIENTO AXIAL PARA DIFERENTES TIEMPOS DE APERTURA DE LA ELECTROVÁLVULA CON UN ORIFICIO DE SALIDA DE $\emptyset 30\ \mathbf{mm}$.

El comportamiento de la circulación en la **Figura 30**, es igual al caso anterior, aumenta en función del tiempo de apertura de la electroválvula, pasando lo mismo con la velocidad.

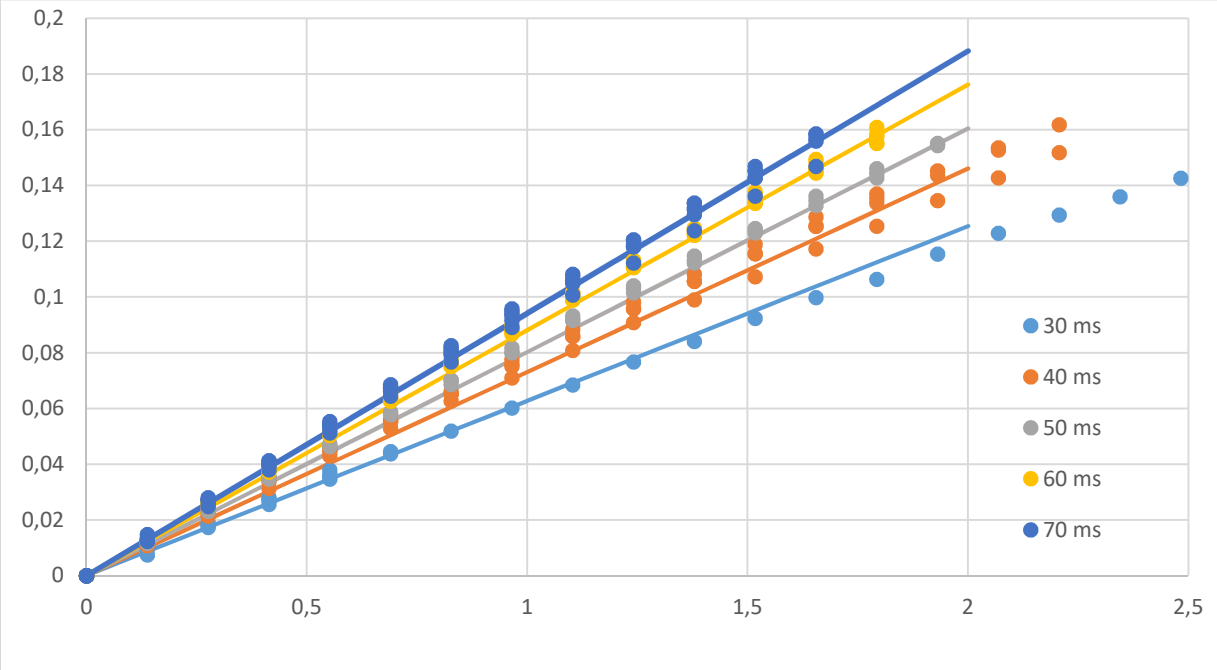

FIGURA 31: DESPLAZAMIENTO AXIAL EN FUNCIÓN DEL TIEMPO PARA PARA DIFERENTES TIEMPOS DE APERTURA DE LA ELECTROVÁLVULA CON UN ORIFICIO DE SALIDA DE $\emptyset 30\ \mathbf{mm}$.

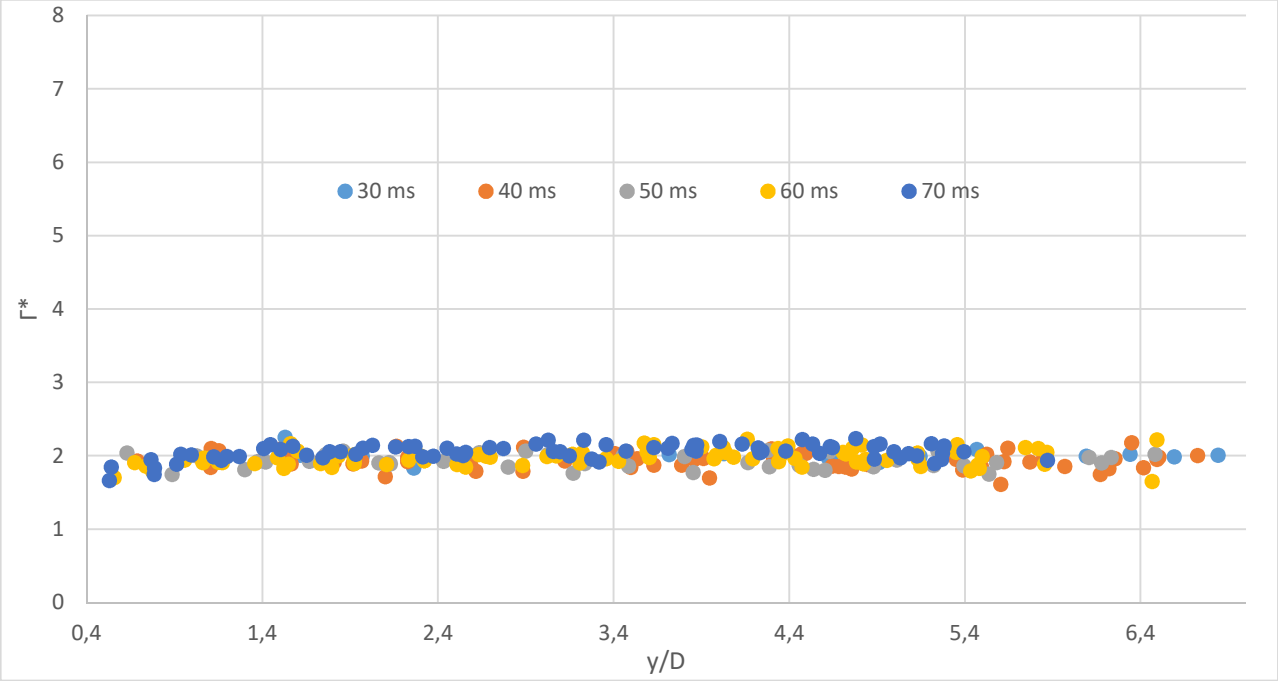

FIGURA 32: CIRCULACIÓN ADIMENSIONAL EN FUNCIÓN DEL DESPLAZAMIENTO AXIAL ADIMENSIONAL PARA DIFERENTES TIEMPOS DE APERTURA DE LA ELECTROVÁLVULA CON UN ORIFICIO DE SALIDA DE Ø30 mm.

Resumen:

A continuación, se representarán los datos de los dos orificios analizados.

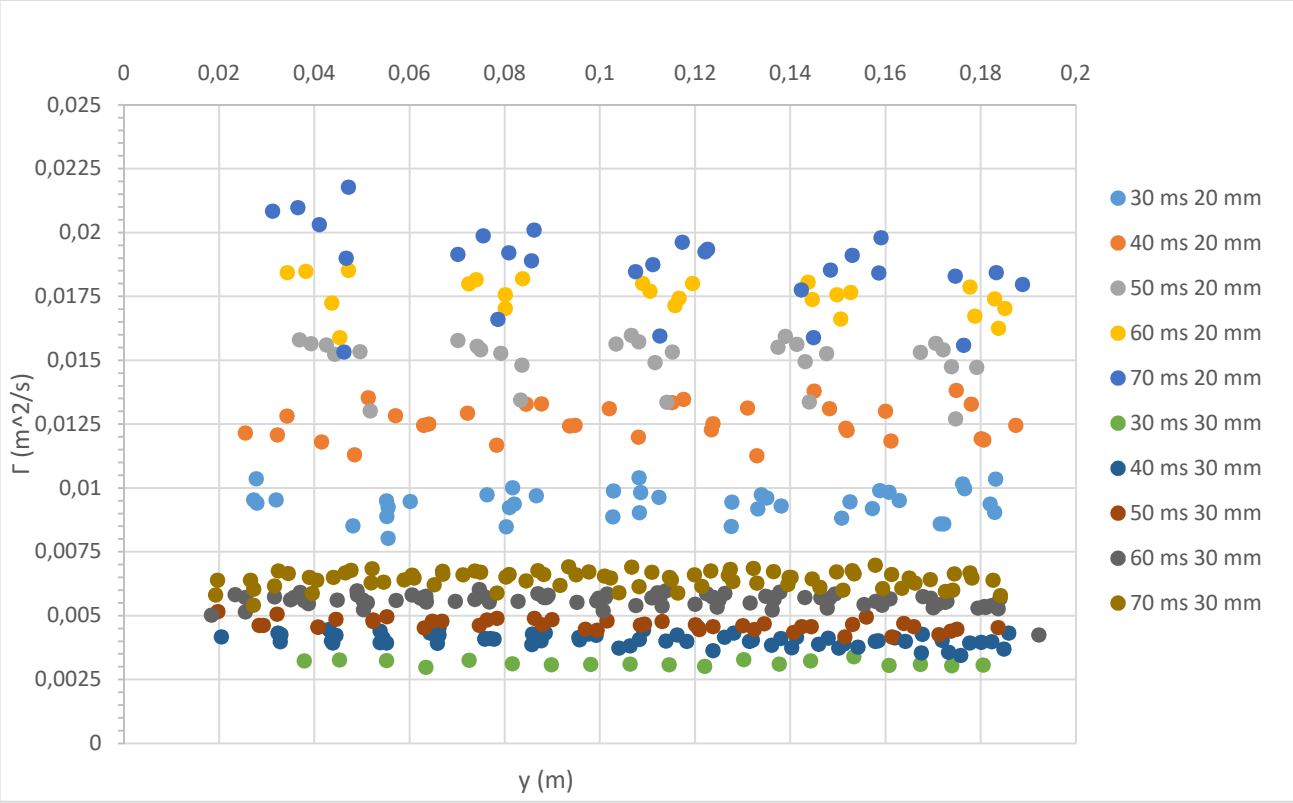

FIGURA 33: CIRCULACIÓN EN FUNCIÓN DEL DESPLAZAMIENTO AXIAL PARA DIFERENTES TIEMPOS DE APERTURA DE LA ELECTROVÁLVULA Y DIFERENTES DIÁMETROS DE ORIFICIOS DE SALIDA.

Es evidente que conforme aumentamos el tiempo de la electroválvula aumenta la circulación y en el caso de aumentar el diámetro de la tobera, la circulación disminuye, ratificado a través de la **Figura 33**. La velocidad es directamente proporcional a la circulación, como el caudal se mantiene constante en todo momento, si aumentamos la sección de paso (el diámetro), la velocidad debe de disminuir.

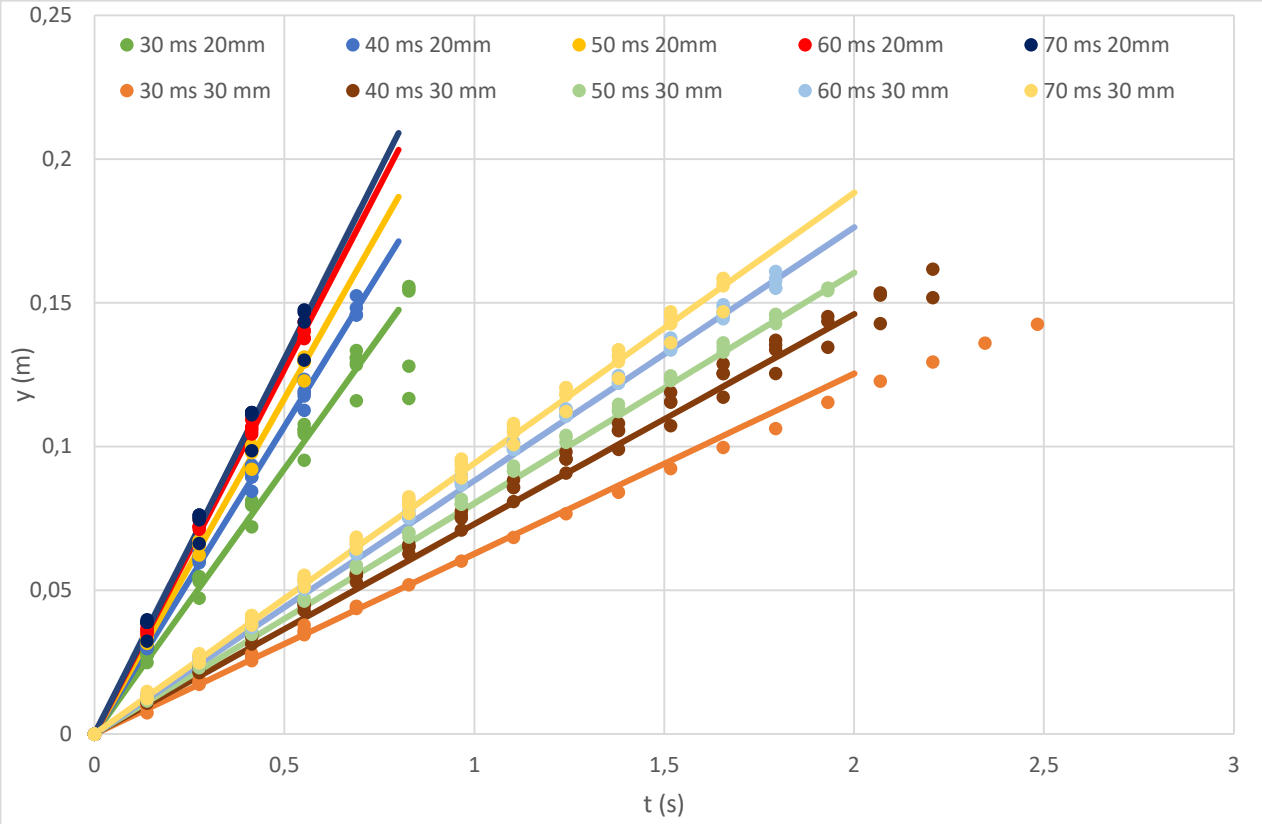

FIGURA 34: DESPLAZAMIENTO AXIAL EN FUNCIÓN DEL TIEMPO PARA DIFERENTES TIEMPOS DE APERTURA DE LA ELECTROVÁLVULA Y DIFERENTES DIÁMETROS DE ORIFICIOS DE SALIDA.

En la **Figura 34**, podemos ratificar lo dicho anteriormente, que la velocidad aumenta con el aumento del tiempo de apertura de la electroválvula y disminuye con el aumento del diámetro de la tobera, en el gráfico se pueden apreciar dos regiones bien diferenciadas, siendo la zona de menor pendiente la asociada al orificio de salida de 30 mm de diámetro.

En cuanto refiere a la circulación adimensional, para todos los orificios de salida y tiempos de apertura, observamos que concurren en un mismo valor, por tanto, hay consistencia en afirmar que existe semejanza entre todos ellos (ver **Figura 35)**.

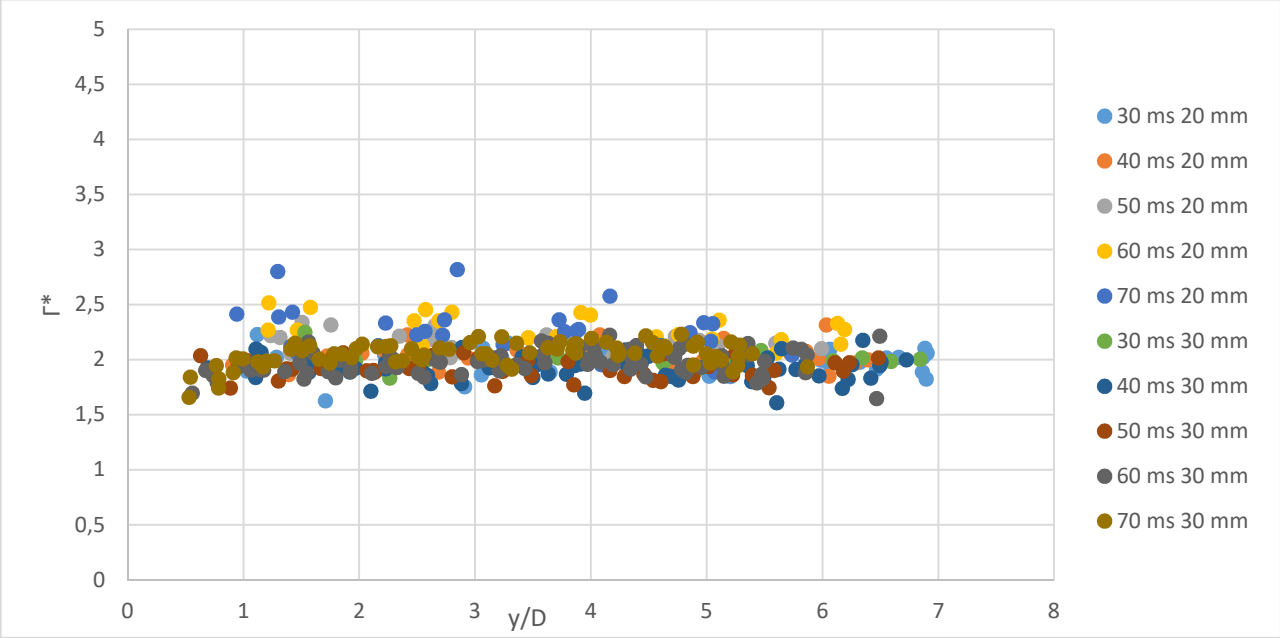

FIGURA 35: CIRCULACIÓN ADIMENSIONAL EN FUNCIÓN DEL DESPLAZAMIENTO AXIAL ADIMENSIONAL PARA DIFERENTES TIEMPOS DE APERTURA DE LA ELECTROVÁLVULA Y DIFERENTES DIÁMETROS DE ORIFICIOS DE SALIDA.

En las siguientes tablas, se mostrarán las propiedades características de todos los casos de anillos de vorticidad analizados a modo de resumen.

Tobera de Ø30mm y 6 bar								
$t_a\ (ms)$	$\Gamma\left(\frac{m^2}{s}\right)$	$V_a\left(\frac{m}{s}\right)$	$V_{at}\left(\frac{m}{s}\right)$	$\frac{V_a}{V_{at}}$	$D\ (m)$	$r\ (m)$	$r_{V_a}\ (m)$	Γ^*
30	0,00300	0,0627	0,0638	0,984	0,0230	0,00316	0,00366	2,08
40	0,00405	0,0731	0,0757	0,965	0,0288	0,00369	0,00350	1,925
50	0,00464	0,0803	0,0849	0,945	0,0301	0,00310	0,00359	1,923
60	0,00559	0,0882	0,0966	0,912	0,0322	0,00321	0,00415	1,968
70	0,00643	0,0942	0,1056	0,892	0,0333	0,00342	0,00486	2,05

TABLA 1: CARACTERÍSTICAS DE VÓRTICES GENERADOS A PARTIR DE UNA TOBERA DE 20 MM DE DIÁMETRO Y 6 BAR PARA DIFERENTES TIEMPOS DE APERTURA DE LA ELECTROVÁLVULA.

Tobera de Ø20mm y 6 bar								
$t_a\ (ms)$	$\Gamma\left(\frac{m^2}{s}\right)$	$V_a\left(\frac{m}{s}\right)$	$V_{at}\left(\frac{m}{s}\right)$	$\frac{V_a}{V_{at}}$	$D\ (m)$	$r\ (m)$	$r_{V_a}\ (m)$	Γ^*
30	0,00939	0,1845	0,208	0,885	0,0259	0,00227	0,00332	1,969
40	0,01260	0,214	0,253	0,848	0,0289	0,00246	0,00417	2,03
50	0,01503	0,234	0,276	0,847	0,0305	0,00291	0,00486	2,11
60	0,01753	0,254	0,297	0,855	0,0303	0,00384	0,00592	2,27
70	0,01868	0,261	0,323	0,810	0,0316	0,00341	0,00623	2,26

TABLA 2: CARACTERÍSTICAS DE VÓRTICES GENERADOS A PARTIR DE UNA TOBERA DE 30 MM DE DIÁMETRO Y 6 BAR PARA DIFERENTES TIEMPOS DE APERTURA DE LA ELECTROVÁLVULA.

BURBUJAS:

En cuanto a la formación de burbujas se han generado de manera cuasi estática, manteniendo la misma bomba de jeringa, pero cambiando para cada caso la aguja.

$2a\ (mm)$	$V_b\ (mm^3)$	$R_b\ (mm)$	$V_F\ (mm^3)$	Ga	Bo
2	39,28	2,11	46,72	302,31	0,597
3	65,55	2,50	70,08	390,63	0,840
3,5	71,11	2,57	81,76	407,06	0,887

TABLA 3: CARACTERÍSTICAS DE BURBUJAS PARA DIFERENTES TAMAÑOS DE AGUJAS.

El volumen de la burbuja se mantiene constante, y se ratifica la formación cuasi estática debido a que el volumen medido es inferior al volumen de Fritz.

Para definir la trayectoria de avance de la burbuja, debemos de fijarnos en los valores del *Ga* y del *Bo*, estos muestran que las burbujas ascienden describiendo una trayectoria caótica o una espiral elíptica **[12]**, ratificando dicho comportamiento a través del análisis de su avance (ver **Figura 36**).

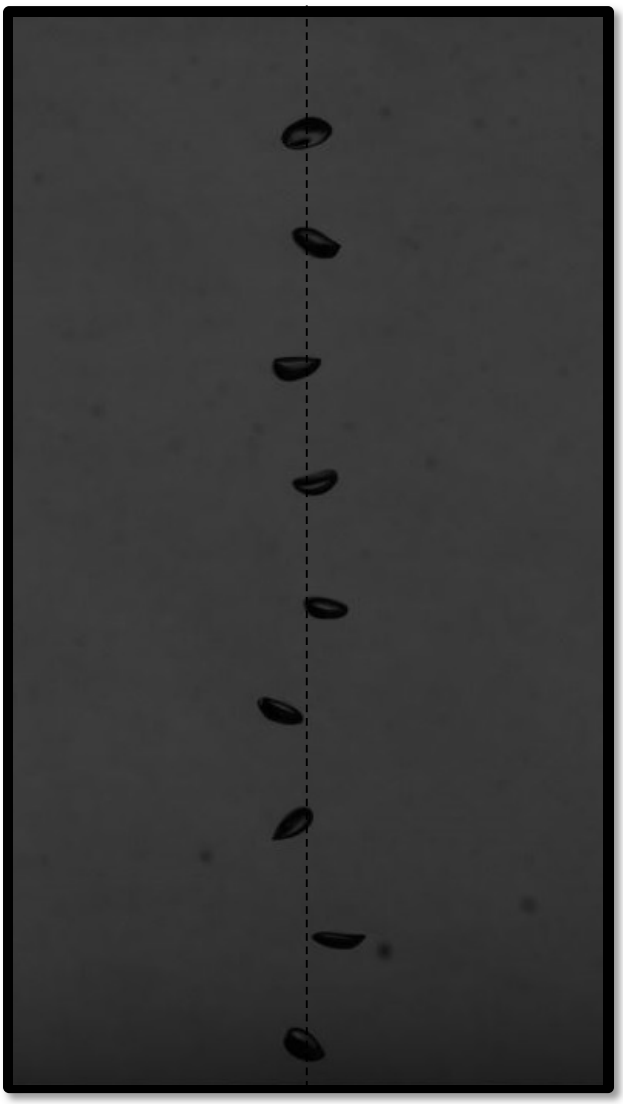

FIGURA 36: AVANCE DE UNA BURBUJA GENERADA POR UNA AGUJA DE **2** *MM Y UN PASO TEMPORAL DE* **0,05** *S* **(GA=302,31, BO=0,597).**

Si nos detenemos a analizar la amplitud del movimiento, la distorsión del movimiento es mucho menor que el diámetro del vórtice, garantizando que la burbuja atraviesa al vórtice por el interior del mismo.

La imagen anterior ha sido realizada a través de la superposición de imágenes de una misma burbuja, entre cada imagen hay cincuenta *frames* y ha sido grabado a una velocidad de 1000 fps, por lo que podemos concluir el paso temporal de avance, siendo este de 0,05 s.

INTERACCIÓN:

El fin del trabajo es predecir el comportamiento de la interacción entre un vórtice y una burbuja, al disponer de distintos casos de vórtices y de burbujas, podremos analizar un gran abanico de casos, todos ellos recogidos en las siguientes tablas.

En dichas tablas podremos observar para cada caso, el número de *We* y la relación entre el diámetro del vórtice y el diámetro equivalente de la burbuja.

	$2a$	2 mm		3 mm		3,5 mm	
Ø20 mm	$t_a\ (ms)$	We	D/D_b	We	D/D_b	We	D/D_b
	30	0,386	6,14	0,458	5,17	0,471	5,03
	40	0,555	6,86	0,659	5,78	0,677	5,63
	50	0,710	7,24	0,843	6,11	0,866	5,94
	60	0,976	7,20	1,158	6,07	1,191	5,91
	70	1,0224	7,50	1,213	6,32	1,247	6,15

TABLA 4: NÚMEROS DE WE Y RELACIONES ENTRE DIÁMETROS DE VÓRTICES Y DIÁMETROS DE BURBUJAS PARA TODAS LAS COMBINACIONES POSIBLES ENTRE LOS VÓRTICES GENERADOS CON EL ORIFICIO DE SALIDA DE Ø20 MM Y TODAS LAS BURBUJAS.

	$2a$	2 mm		3 mm		3,5 mm	
Ø30 mm	$t_a\ (ms)$	We	D/D_b	We	D/D_b	We	D/D_b
	30	0,0498	5,46	0,0591	4,60	0,0608	4,47
	40	0,0579	6,84	0,0687	5,77	0,0706	5,61
	50	0,0697	7,14	0,0827	6,02	0,0850	5,86
	60	0,0881	7,65	0,1045	6,45	0,1074	6,27
	70	0,1090	7,91	0,1293	6,67	0,1329	6,48

TABLA 5: NÚMEROS DE WE Y RELACIONES ENTRE DIÁMETROS DE VÓRTICES Y DIÁMETROS DE BURBUJAS PARA TODAS LAS COMBINACIONES POSIBLES ENTRE LOS VÓRTICES GENERADOS CON EL ORIFICIO DE SALIDA DE Ø30 MM Y TODAS LAS BURBUJAS.

Tras la documentación reiterada de todos los diferentes casos se concluye que par *We* < 0,2 nunca se aprecia una rotura, no obstante, cuando aumentamos dicho valor, empiezan a aparecer algunas roturas de burbujas. Por este motivo nos vemos obligados a hacer un estudio probabilístico, donde analizaremos los casos para que el $We \in [0{,}45, 1{,}2]$.

El estudio probabilístico se basa en el análisis de 50 casos para cada *We* pudiendo sacar así un porcentaje de rotura que describa el comportamiento.

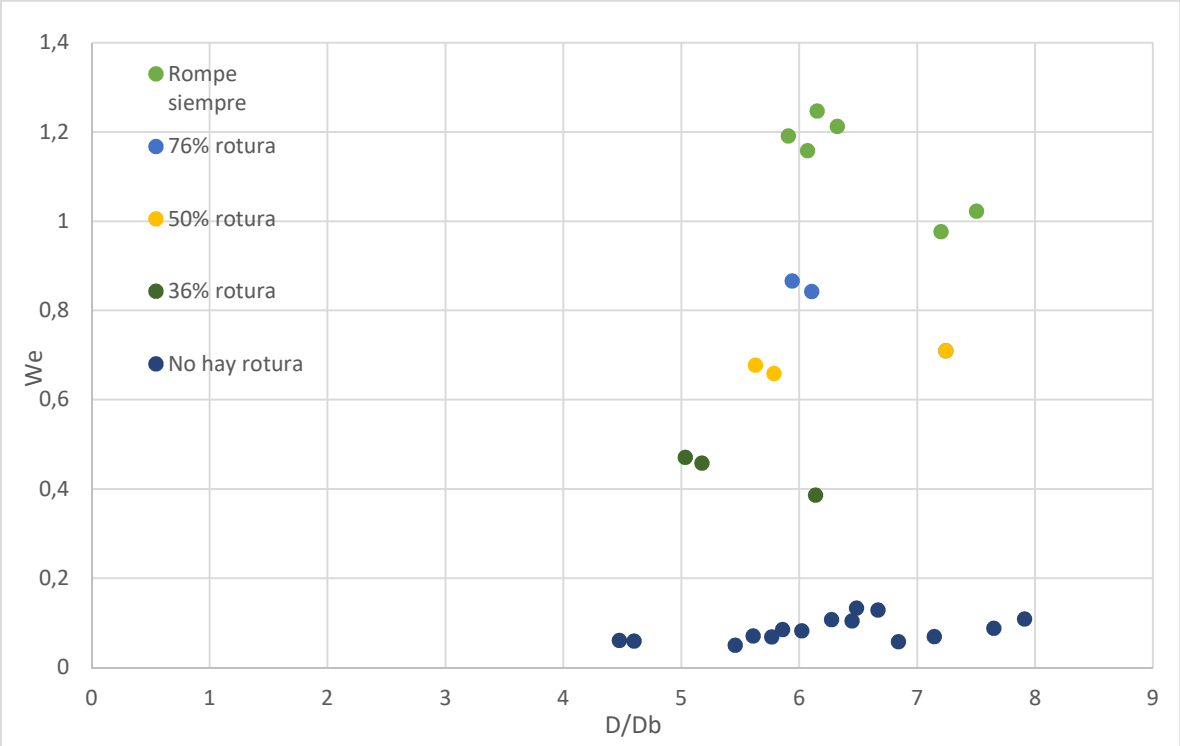

FIGURA 37: PORCENTAJE DE ROTURAS, NÚMEROS DE WE Y RELACIONES ENTRE DIÁMETROS DE VÓRTICES Y DIÁMETROS DE BURBUJAS PARA TODAS LAS COMBINACIONES POSIBLES ENTRE LOS VÓRTICES GENERADOS Y TODAS LAS BURBUJAS

Los resultados mostrados en la **Figura 37**, tiene el comportamiento esperado, conforme va aumentando el número de *We*, va aumentando el porcentaje de roturas, el hecho de que con *We* bajos observemos roturas, se debe a los valores de *Ga* que describen a las burbujas puestas en juego. Hablamos de un $Ga \in [302{,}31, 407{,}06]$, abarca un rango donde la burbuja es inestable por sí misma teniendo una tendencia a partirse por sí sola.

La casuística de la rotura siempre es la misma, aunque el resultado puede variar. La burbuja atraviesa al vórtice por el centro, se ve influenciada por el campo de velocidad inducido por el vórtice, describiendo la burbuja un giro que la introduce dentro del núcleo del vórtice. Una vez dentro del núcleo del vórtice la burbuja tiende a estirarse a lo largo del mismo, rompiéndose en algunas ocasiones en dos burbujas de aparente igual volumen y en otras, en dos burbujas, pero donde una es notoriamente más grande que la otra. En las **Figuras 38** y **39** se puede observar un ejemplo de interacción entre un anillo de vorticidad y una burbuja, donde se produce una rotura.

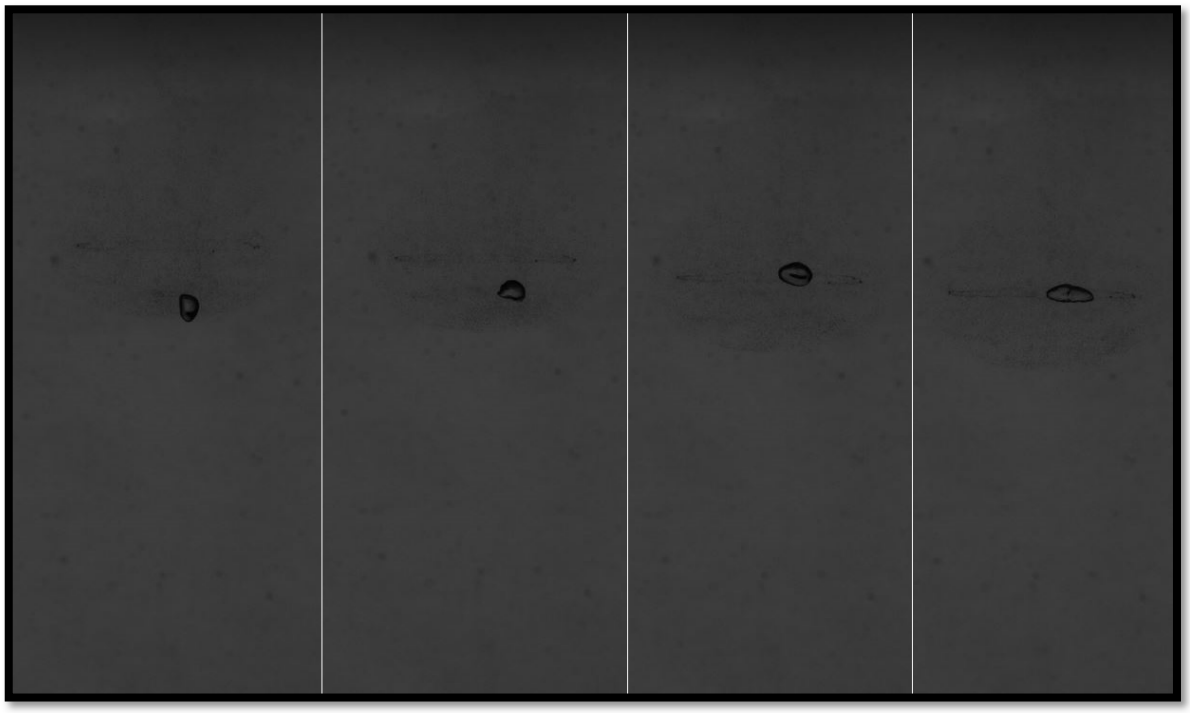

FIGURA 38: PUNTO DE CONTACTO DEL VÓRTICE CON LA BURBUJA **(WE=1,0224, D/DB=7,50).**

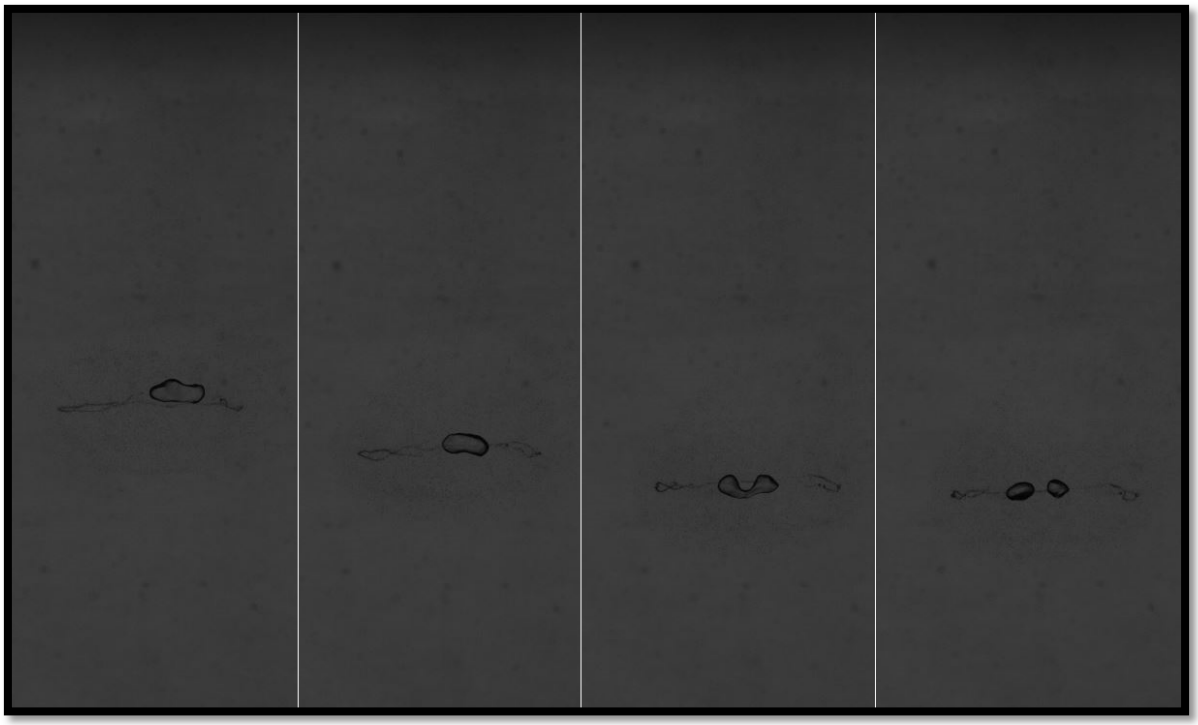

FIGURA 39: PUNTO DE ROTURA EN LA INTERACCIÓN **(WE=1,0224, D/DB=7,50).**

CONCLUSIONES:

En los casos analizados se aprecia un valor de *We* crítico, la unidad, a partir de este valor, todas las interacciones generadas para un rango de $D/D_b \in [4,8]$ generan rotura. Del mismo modo, se evidencia una clara estratificación en el porcentaje de roturas en la medida de que el *We* aumenta, representada en la **Figura 37**.

Se hace evidente que el factor predominante para que en la interacción se produzca una rotura es el *We*, no pudiendo valorar la influencia de la relación de diámetros puesto que todos los casos se encuentran dentro de un mismo rango. Por esta razón, resultaría interesante el estudio de una variada casuística, donde se pongan en juego otros valores de *We* y relaciones de diámetros.

Las relaciones geométricas de diámetros restantes serán aquellas donde el tamaño del vórtice y de la burbuja sea del mismo orden de magnitud y donde el tamaño de la burbuja sea de un orden de magnitud superior al tamaño del vórtice, es decir, $D/D_b \approx 1$ y $D/D_b \approx 0{,}1$ respectivamente.

Se ha plasmado que para *We* muy pequeños, no se generan roturas, haciendo hincapié en la rotura, sería conveniente el estudio de casos donde número adimensional sea siempre mayor a la unida.

A. Revuelta **[13]** recoge de manera numérica los casos analizados como la casuística restante de este trabajo, ilustrándolo de manera gráfica.

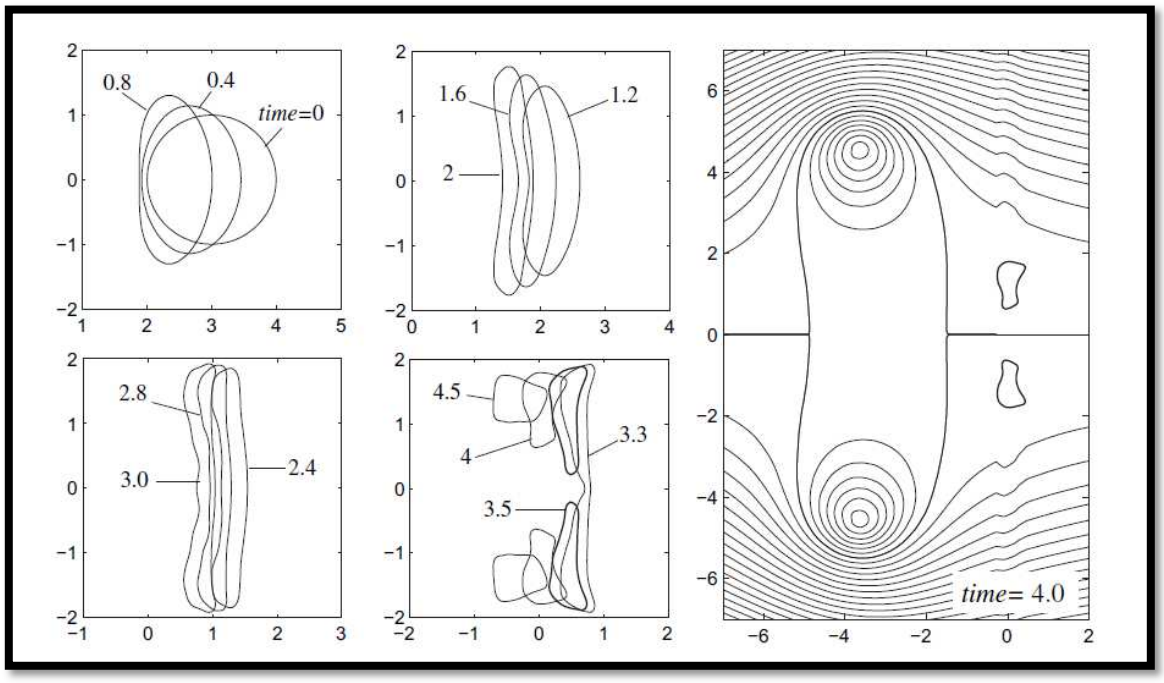

FIGURA 40: DEFORMACIÓN LENTICULAR Y ROTURA ANULAR EN UN VÓRTICE MUCHO MÁS GRANDE QUE LA BURBUJA **(WE=8, D/DB=5, RE=1000) [13].**

En la **Figura 40**, se muestra claramente la morfología de rotura observada en nuestros casos analizados.

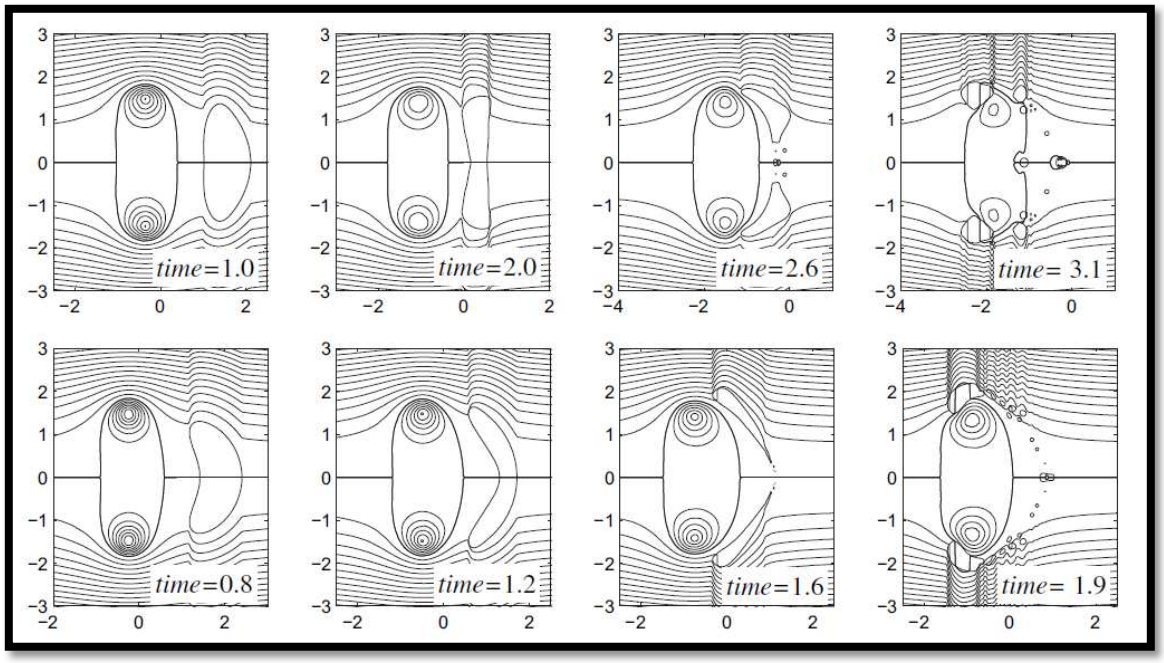

FIGURA 41: ROTURA DE BURBUJA POR UN VÓRTICE DE TAMAÑO SIMILAR A LA BURBUJA **(RE=1000, D/DB=1,5, Y DOS WE DIFERENTES WE=2 (ARRIBA) Y WE=10 (ABAJO)) [13].**

La burbuja experimenta una deformación lenticular, y cuando el vórtice está lo suficientemente cerca, la burbuja lo intenta envolver, mientras que esto ocurre, la burbuja se parte en múltiples fragmentos desde el centro hacia los ejes [13], tal y como se observa en la **Figura 41**.

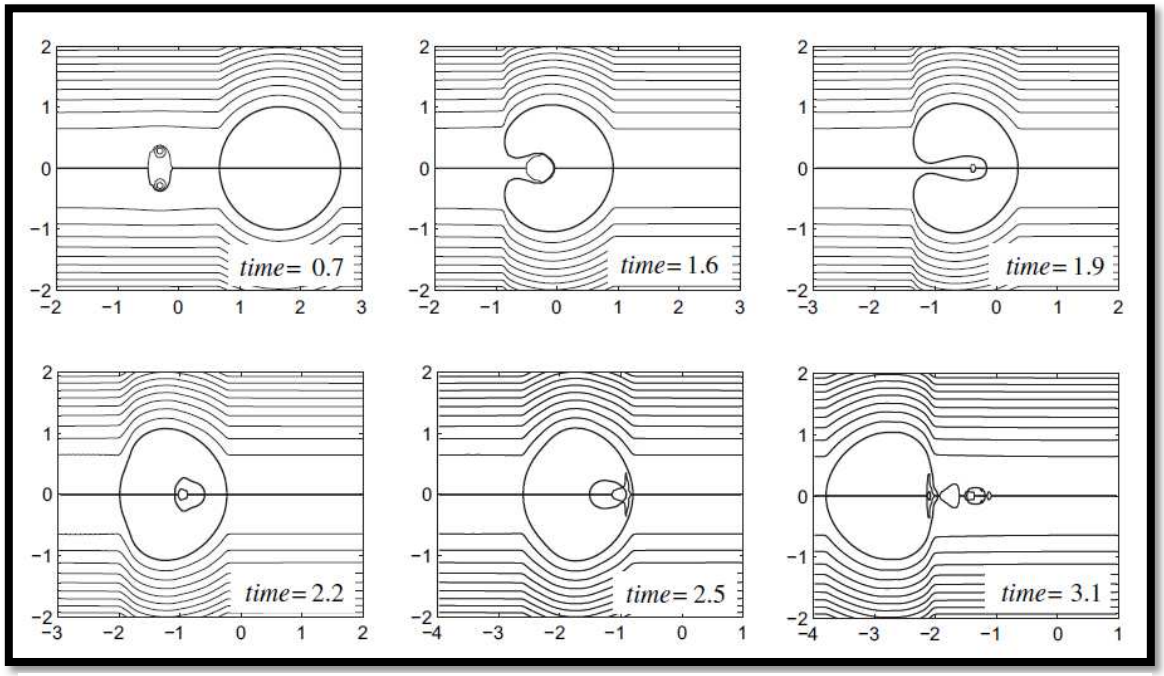

FIGURA 42: INTERACCIÓN DE UNA BURBUJA CON UN VÓRTICE PEQUEÑO **(WE=4, D/DB=0,3, RE=1000) [13].**

La **Figura 42** muestra como El vórtice atraviesa el contorno de la burbuja, pero en este caso no se aprecia una partición binaria, sino que deja una estela de pequeñas burbujas cuando termina de atravesarla [13].

ANEXOS:

Código de *Caract_Vortex*:

```
function varargout = Caract_Vortex(varargin)
% CARACT_VORTEX MATLAB code for Caract_Vortex.fig
%      CARACT_VORTEX, by itself, creates a new CARACT_VORTEX or raises the existing
%      singleton*.
%
%      H = CARACT_VORTEX returns the handle to a new CARACT_VORTEX or the handle to
%      the existing singleton*.
%
%      CARACT_VORTEX('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CARACT_VORTEX.M with the given input arguments.
%
%      CARACT_VORTEX('Property','Value',...) creates a new CARACT_VORTEX or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Caract_Vortex_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Caract_Vortex_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Caract_Vortex

% Last Modified by GUIDE v2.5 16-Jan-2019 13:07:25

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Caract_Vortex_OpeningFcn, ...
                   'gui_OutputFcn',  @Caract_Vortex_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Caract_Vortex is made visible.
```

```
function Caract_Vortex_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Caract_Vortex (see VARARGIN)

% Choose default command line output for Caract_Vortex
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Caract_Vortex wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = Caract_Vortex_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
function uipanel1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% ------------------------------------------------------------------------
function uitable1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to uitable1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton_cargar.
function pushbutton_cargar_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_cargar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global arch
global direc
```

```
[arch,direc]=uigetfile({'*.mat'},'Seleccione los archivos a analizar:','MultiSelect', 'on');
ArchDireccion=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%SOLO PUEDE LEER ARCHIVOS .MAT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%arch   %nombre del archivo cell de array
%para convertir cell array en char, usar char
%para convertir char en cell, cellstr(direc)
% tambien esta la funcion cell2mat
%direccion;  %char localizacion del archivo  C:\....

%CONCLUSION PARA GUARDARLOS TODOS DEBO DE INTRODUCIRLOS EN UN VECTOR DE
%CELL Y PARA TRABAJAR CON ELLOS CONVERTIRLOS A CHAR

if length(direc) == 1
cancelar =0;
set(handles.pushbutton_cargar,'UserData',cancelar)
else
ArchDireccion=[arch cellstr(direc)];
%ultima posicion del vector es la localizacion del archivo
set(handles.pushbutton_cargar,'UserData',ArchDireccion);
end


if length(ArchDireccion)==1
   a='No se ha cargado ningún archivo';
   b='';
   set(handles.text_file,'String',a);
   set(handles.text_file2,'String',b);
else
   n=length(ArchDireccion);
   if n<14
   set(handles.text_file,'String',cellstr(ArchDireccion'));
   else
   set(handles.text_file,'String',cellstr(ArchDireccion(1:14)));
   set(handles.text_file2,'String',cellstr(ArchDireccion(14:n)));
   end
end




% --- Executes on button press in pushbutton_calcular.
function pushbutton_calcular_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_calcular (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

Escala=str2num(get(handles.edit_escala,'String'));
Densidad=str2num(get(handles.edit_densidad,'String'));
Viscosidad_cine=str2num(get(handles.edit_viscosidad,'String'));
```

```
Tiempo_apertura=str2num(get(handles.edit_tapertura,'String'));
DeltaT=str2num(get(handles.edit_tavance,'String'));
Presion=str2num(get(handles.edit_presion,'String'));
D=str2num(get(handles.edit_diametro,'String'));
Export=[char(get(handles.edit_excell,'String')),'.xlsx'];;

ArchDireccion=get(handles.pushbutton_cargar,'UserData');

if length(ArchDireccion)==1||length(ArchDireccion)==0

uiwait(msgbox({'        Imposible de calcular'},'ERROR','error'));

else
n=length(ArchDireccion);

Tfin=(n-1)*DeltaT;
%time=linspace(0,Tfin,n-1);
time=[0];

for j=1:n-2
time=[time, DeltaT*j];
end

direccion= char(ArchDireccion(n));

cd(direccion)


for i=1:n-1

load(char(ArchDireccion(i)));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%auxiliar_quiver(x,y,u,v,Escala)


figure(); hold on
quiver(x,y,smooth2a(u,2),smooth2a(v,2),Escala,'r');grid on;%Representa velocidad y posicion
hold off;



% _________________SEGUIMIENTO DEL PUNTO DE MAXIMA VORTICIDAD_________


%  pos1=ginput(2);                          % Derecha-izquierda   Extraigo la posicion del vortice
```

```
%  posvorty1(i)=(pos1(1,2)+pos1(2,2))/2;
%  posvortx1(i)=(pos1(1,1)+pos1(2,1))/2;
sz = size(x);
x2=1;                %Izq    x
x3=sz(2);
y1=1;                %Abajo  y
y2=sz(1);             %Arriba y

%si no metemos el suavizador nos saldrá el valor original de la vorticidad
%del vórtice
%vorticidad=smooth2a(vort(y1:y2,x2:x3),2);
vorticidad=(vort(y1:y2,x2:x3)); %no meter smooth para sacar el diámetro
%velocidadaxial=smooth2a(v(y1:y2,x2:x3),2);
velocidadaxial=(v(y1:y2,x2:x3));

[MVfila, MVcolum]=find(vorticidad==max(max(vorticidad)));
[mVfila, mVcolum]=find(vorticidad==min(min(vorticidad)));

%diámetro del vórtice:
Diametrodelnucleo(i)=abs(x(MVfila,MVcolum)-x(mVfila,mVcolum));

posvorty1(i)=(y(MVfila,MVcolum)+y(mVfila,mVcolum))/2;
posvortx1(i)=(x(MVfila,MVcolum)+x(mVfila,mVcolum))/2;

% _____DELIMITACION DE CONTORNO Y EJE_____

sz = size(x);                    %Tamaño de matriz posicion
eje=posvortx1(i);
pos_eje(i,:)=round((MVcolum+mVcolum)/2);

x1=pos_eje(i);          %Eje    x
x2=1;                %Izq    x
x3=sz(2);
y1=1;                %Abajo  y
y2=sz(1);             %Arriba y

%propiedades del core1:
velocidadaxial2=(v(y1:y2,x2:x1));
[mVaxfila, mVaxcolum]=find(velocidadaxial2==min(min(velocidadaxial2)));
%radio del core1:
core1(i)=abs(x(MVfila,MVcolum)-x(mVaxfila, mVaxcolum));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%propiedades del core2:
velocidadaxial3=(v(y1:y2,x1:x3));
[mVax2fila, mVax2colum]=find(velocidadaxial3==min(min(velocidadaxial3)));
%radio del core2:
core2(i)=abs(x(mVfila,mVcolum)-(x(mVax2fila, mVax2colum+x1-1)));
%radio del core promedio
corepromedio(i)=(core1(i)+core2(i))*0.5;
% porcentaje del radio del core respecto al radio del vórtice
porcentajecore(i)=corepromedio(i)/(Diametrodelnucleo(i)*0.5);
```

```
dx=mean(diff(x(1,:)));          %Dx
dy=mean(diff(y(:,1)));          %Dy

C11a= dy *  sum(v(y1:y2,x1));       %Velocidad contorno y
C12a= -sum(u(y2,x2:x1)) *  dx;      %Velocidad contorno x
C13a=  -dy * sum(v(y1:y2,x2));      %Velocidad contorno y
C14a= sum (u(y1,x2:x1))*dx;         %Velocidad contorno x

C1a(i)=C11a+C12a+C13a+C14a;

C11b=dy*sum(v(y1:y2,x1));
C12b=sum(u(y2,x1:x3))*dx;
C13b=-dy*sum(v(y1:y2,x3));
C14b=-sum(u(y1,x1:x3))*dx;

C1b(i)=C11b+C12b+C13b+C14b;

C1(i)=(abs(C1a(i))+abs(C1b(i)))/2; %media de la circulación a ambos lados a partir del
perímetro y una línea cerrada

G11(i)=sum(sum(vort(y1:y2,x2:x1)))*abs(dx*dy);
G12(i)=sum(sum(vort(y1:y2,x1:x3)))*abs(dx*dy);
G1(i)=(abs(G11(i))+abs(G12(i)))/2; %la media de de la circulacióin a ambos lados a partir de la
vorticidad y la superficie

% Diametro_anillo(i)=abs((pos1(1,1)-pos1(2,1)));
Diametro_anillo(i)=Diametrodelnucleo(i);



%perfil de vorticidad
figure(51); hold on
plot(x(MVfila,x2:x3),vorticidad(MVfila,x2:x3));
%plot(x(MVfila,x2:x3),velocidadaxial(MVfila,x2:x3));
ylabel('w [s^-^1]');
%legend('w x 10^-^2 [s^-^1]','v [m/s]');
xlabel('x [m]');
hold off;

figure(52); hold on
plot(x(MVfila,x2:x3),velocidadaxial(MVfila,x2:x3));
ylabel('v [m/s]');
xlabel('x [m]');
hold off;


disp('    nº      Eje      Diametro vortice'); %Solo para mostrar por
```

```
[     i    pos_eje(i)  Diametro_anillo(i)]      %pantalla en trabajo
end


vel=polyfit(time,[posvorty1],1);
velocidadinicio=vel(1);
VELOCIDAD_AVANCE=ones(1,length(posvorty1))*vel(1);
%velocidad promedio
%____________________ADIMENSIONALES_________________

%siendo G1 la media de la circulación G1=v*dl

%P1=G1./(VELOCIDAD_AVANCE*D);  %Adimensional_P1 = G1/(v_avance_vórtice * D_tobera)
P1=G1./(velocidadinicio*D);
if P1(1)==Inf
   P1(1)=1e100;
end
P2=(G1./(Presion*Tiempo_apertura))*Densidad;   %Adimensional_P2 =
G1*rho/(Presion*Tiempo_apertura)
Re=G1/Viscosidad_cine;  %Reynolds


%____________________REPRESENTACION GRAFICA_____________________________


figure(32)
hold on
plot(time,G1,'b-')
plot(time,C1,'r-')
legend('Circulacion Superficie','Circulacion linea')
ylabel('Circulacion [m2/s]');
xlabel('Tiempo [s]');
axis([0 max(time) 0 0.03]);
hold off

figure(33)
plot(time,posvorty1,'b-')
ylabel('y [m]');
xlabel('Tiempo [s]');


figure(34)
plot(time,Diametro_anillo,'r-')
ylabel('D[m]');
xlabel('Tiempo [s]');
axis([0 max(time) 0 0.08]);


%tad=time.*VELOCIDAD_AVANCE/(D*0.5); % velocidad de avance debe de ser constante
tad=time.*velocidadinicio/(D*0.5);
zad=posvorty1/(D*0.5);
```

```
%promediado radio del core
porcentajeradiocore1=core1./((Diametro_anillo)*0.5);
porcentajepromedio1=sum(porcentajeradiocore1)/length(porcentajeradiocore1);
porcentajeradiocore2=core2./((Diametro_anillo)*0.5);
porcentajepromedio2=sum(porcentajeradiocore2)/length(porcentajeradiocore1);
promediocore=(core1+core2)/2;
%____________________EXPORTADO A EXCEL DE LOS DATOS_____________________


% Export ='Archivo_excell.xsll';

%ESCRIBIMOS CABEZERAS DE LAS TABLAS
xlswrite(Export,{'posición del vórtice'},1,'A1') %Posicion del vortice
xlswrite(Export,{'tiempo'},1,'B1')
xlswrite(Export,{'Diametro anillo'},1,'C1')
xlswrite(Export,{'velocidad avance'},1,'D1')

xlswrite(Export,{'Circulación de linea'},1,'E1')
xlswrite(Export,{'Circulación de superficie'},1,'F1')
%anteriormente viene definido los elementos que integran los
%adimensionales:
xlswrite(Export,{'AD(P1)'},1,'G1') %Adimensional1 de C/(v*l)
xlswrite(Export,{'AD(P2)'},1,'H1') %Adimensional2 C*rho/(P*t)
xlswrite(Export,{'Re'},1,'I1') %REYNOLDS

xlswrite(Export,{'t*'},1,'J1') %timpo adimensional t*=t*Uc/R
xlswrite(Export,{'z/R'},1,'K1') %distancia adimensional

xlswrite(Export,{'Rcore1'},1,'L1')
xlswrite(Export,{'Rcore2'},1,'M1')
xlswrite(Export,{'promedio del core'},1,'N1')

%INTRODUCIMOS DATOS EN CADA COLUMNA DE LA TABLA CREADA
xlswrite(Export,posvorty1',1,'A2')
xlswrite(Export,time',1,'B2')
xlswrite(Export,Diametro_anillo',1,'C2')
xlswrite(Export,VELOCIDAD_AVANCE',1,'D2')
xlswrite(Export,C1',1,'E2')
xlswrite(Export,G1',1,'F2')
xlswrite(Export,P1',1,'G2')
xlswrite(Export,P2',1,'H2')
xlswrite(Export,Re',1,'I2')
xlswrite(Export,tad',1,'J2') %timpo adimensional t*=t*Uc/R
xlswrite(Export,zad',1,'K2') %distancia adimensional
xlswrite(Export,core1',1,'L2') %Radiocore/Radiovortice
xlswrite(Export,core2',1,'M2')
xlswrite(Export,promediocore',1,'N2')
end

% -----------------------------------------------------------------
```

```
function uitable2_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to uitable2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit_escala_Callback(hObject, eventdata, handles)
% hObject    handle to edit_escala (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Escala ;
Escala = str2num(get(hObject,'String'));
set(handles.edit_escala,'String',Escala);

% Hints: get(hObject,'String') returns contents of edit_escala as text
%        str2double(get(hObject,'String')) returns contents of edit_escala as a double


% --- Executes during object creation, after setting all properties.
function edit_escala_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_escala (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_densidad_Callback(hObject, eventdata, handles)
% hObject    handle to edit_densidad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Densidad;
Densidad= str2num(get(hObject,'String'));
set(handles.edit_densidad,'String',Densidad);


% Hints: get(hObject,'String') returns contents of edit_densidad as text
%        str2double(get(hObject,'String')) returns contents of edit_densidad as a double


% --- Executes during object creation, after setting all properties.
function edit_densidad_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_densidad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end


function edit_tavance_Callback(hObject, eventdata, handles)
% hObject    handle to edit_tavance (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global DeltaT;
DeltaT= str2num(get(hObject,'String'));
   set(handles.edit_tavance,'String',DeltaT);


% Hints: get(hObject,'String') returns contents of edit_tavance as text
%        str2double(get(hObject,'String')) returns contents of edit_tavance as a double


% --- Executes during object creation, after setting all properties.
function edit_tavance_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_tavance (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


function edit_presion_Callback(hObject, eventdata, handles)
% hObject    handle to edit_presion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 global Presion;
 Presion = str2num(get(hObject,'String')) ;
   set(handles.edit_presion,'String',Presion);


% Hints: get(hObject,'String') returns contents of edit_presion as text
%        str2double(get(hObject,'String')) returns contents of edit_presion as a double


% --- Executes during object creation, after setting all properties.
function edit_presion_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_presion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_diametro_Callback(hObject, eventdata, handles)
% hObject    handle to edit_diametro (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global D;
D=str2num(get(hObject,'String'));
    set(handles.edit_diametro,'String',D);
% Hints: get(hObject,'String') returns contents of edit_diametro as text
%        str2double(get(hObject,'String')) returns contents of edit_diametro as a double


% --- Executes during object creation, after setting all properties.
function edit_diametro_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_diametro (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_tapertura_Callback(hObject, eventdata, handles)
% hObject    handle to edit_tapertura (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Tiempo_apertura;
Tiempo_apertura= str2num(get(hObject,'String'));
set(handles.edit_tapertura,'String',Tiempo_apertura);


% Hints: get(hObject,'String') returns contents of edit_tapertura as text
%        str2double(get(hObject,'String')) returns contents of edit_tapertura as a double


% --- Executes during object creation, after setting all properties.
function edit_tapertura_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_tapertura (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_viscosidad_Callback(hObject, eventdata, handles)
% hObject    handle to edit_viscosidad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Viscosidad_cine;
Viscosidad_cine= str2num(get(hObject,'String'));
set(handles.edit_viscosidad,'String',Viscosidad_cine);


% Hints: get(hObject,'String') returns contents of edit_viscosidad as text
%        str2double(get(hObject,'String')) returns contents of edit_viscosidad as a double


% --- Executes during object creation, after setting all properties.
function edit_viscosidad_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_viscosidad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --------------------------------------------------------------------
function Ayuda_Callback(hObject, eventdata, handles)
% hObject    handle to Ayuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
uiwait(msgbox({'Información:';'Programa realizado para la caracterización de
vórtices.';'';'Localización:';'Área de Mecánica de Fluidos de la Universidad de Jaén.';'';'Autor del
programa: Carlos Avilés Coca';'';'Pasos a seguir:';'1) Añadir al path carpetas y subcarpetas
donde se encuentre el archivo del programa';'2) Modificación de las Propiedades';'3) Escribir el
nombre del archivo excell a exportar (nunca debe de estar en blanco)';'4) Cargar los archivos
(en el caso de estar numerados, se leerán en orden numérico)'; '5) Calcular';'';'Carga de
Imágenes:';'En la pantalla aparecerán los nombres de los archivos cargados en el orden de
lectura acompañados al final de la dirección de tu PC donde se encuentren.';'';'Comunicación
con el programa:';'"Imposible de Calcular" Aparecerá en el caso de que no haya archivos
cargados.';'"Se han eliminado los archivos" Aparecerá en caso de pulsar RESET.';'"No se ha
cargado ningún archivo" Aparecerá si cancela la opción de cargar imágenes.';'';'Salida:';'Se
```

```
generará un archivo excell con el nombre establecido en la carpeta donde se encuentren los archivos leidos.';'El archivo albergará todos los parámetros que calcula el programa.';} ,'MENÚ DE AYUDA','help'));


% --- Executes on button press in radiobutton_valores_defecto.
function radiobutton_valores_defecto_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton_valores_defecto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if hObject==handles.radiobutton_valores_defecto
   %PARÁMETROS POR DEFECTO
   Defecto=1;
   Escala=4;
   Densidad=998.2;
   Viscosidad_cine=1.007e-6;
   Tiempo_apertura=30e-3;
   DeltaT=1/7.25;
   Presion=6e5;
   D=2/100;
   Excell='Excell_Document';
   set(handles.edit_escala,'String',Escala);
   set(handles.edit_densidad,'String',Densidad);
   set(handles.edit_viscosidad,'String',Viscosidad_cine);
   set(handles.edit_tapertura,'String',Tiempo_apertura);
   set(handles.edit_tavance,'String',DeltaT);
   set(handles.edit_presion,'String',Presion);
   set(handles.edit_diametro,'String',D);
   set(handles.edit_excell,'String',Excell);
end
   % Hint: get(hObject,'Value') returns toggle state of radiobutton_valores_defecto


% --- Executes on button press in pushbutton_reset.
function pushbutton_reset_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ArchDireccion=0;
set(handles.pushbutton_cargar,'UserData',ArchDireccion);
a='Se han eliminado los archivos';
b='';
set(handles.text_file,'String',a);
set(handles.text_file2,'String',b);

% --- Executes during object creation, after setting all properties.
function text_file_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% --- Executes during object creation, after setting all properties.
function text_file2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text_file2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function text_SAVE_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text_SAVE (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



function edit_excell_Callback(hObject, eventdata, handles)
% hObject    handle to edit_excell (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_excell as text
%        str2double(get(hObject,'String')) returns contents of edit_excell as a double
global Excell;
Excell=get(hObject,'String');
set(handles.edit_excell,'String',Excell);

% --- Executes during object creation, after setting all properties.
function edit_excell_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_excell (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over edit_excell.
function edit_excell_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to edit_excell (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Código de *CARACTBUBBLES*:

```
function varargout = CARACTBUBBLES(varargin)
% CARACTBUBBLES MATLAB code for CARACTBUBBLES.fig
%      CARACTBUBBLES, by itself, creates a new CARACTBUBBLES or raises the existing
%      singleton*.
%
%      H = CARACTBUBBLES returns the handle to a new CARACTBUBBLES or the handle to
%      the existing singleton*.
%
%      CARACTBUBBLES('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CARACTBUBBLES.M with the given input arguments.
%
%      CARACTBUBBLES('Property','Value',...) creates a new CARACTBUBBLES or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before CARACTBUBBLES_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to CARACTBUBBLES_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help CARACTBUBBLES

% Last Modified by GUIDE v2.5 31-Mar-2019 13:54:15

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @CARACTBUBBLES_OpeningFcn, ...
                   'gui_OutputFcn',  @CARACTBUBBLES_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before CARACTBUBBLES is made visible.
function CARACTBUBBLES_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to CARACTBUBBLES (see VARARGIN)

% Choose default command line output for CARACTBUBBLES
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes CARACTBUBBLES wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = CARACTBUBBLES_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton_cimcalibrar.
function pushbutton_cimcalibrar_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_cimcalibrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[archCalibrar,direcCalibrar]=uigetfile({'*.tif';'*.JPG';'*.bmp'},'Seleccione foto para calibrar:');

if length(direcCalibrar) == 1
Nocal='No se ha cargado ninguna imagen para calibrar.';
set(handles.text_docCal,'String',Nocal);
cancelar=0;
set(handles.pushbutton_cimcalibrar,'UserData',cancelar);
else
ArchDireccionCali=[archCalibrar cellstr(direcCalibrar)];
%ultima posicion del vector es la localizacion del archivo
set(handles.text_docCal,'String',ArchDireccionCali);
set(handles.pushbutton_cimcalibrar,'UserData',ArchDireccionCali);
end


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton_calibrar.
function pushbutton_calibrar_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_calibrar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ArchDireccionCali=get(handles.pushbutton_cimcalibrar,'UserData');

if length(ArchDireccionCali)==1||length(ArchDireccionCali)==0

uiwait(msgbox({'        Imposible de calcular'},'ERROR','error'));

else

for i=1:6
n=length(ArchDireccionCali);
direccion= char(ArchDireccionCali(n));
cd(direccion);
calibracion=imread(char(ArchDireccionCali(1)));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tiff=char(ArchDireccionCali(1));
ntiff=length(tiff);
nmtiff=[];
for i=ntiff:-1:(ntiff-2)
nmtiff=[nmtiff tiff(i)];
end
if  nmtiff == ['FIT']
 Blanco=double(max(max(calibracion)));
 calibracion=uint16(round(double(calibracion)*65535/Blanco));
end
if nmtiff == ['fit']
 Blanco=double(max(max(calibracion)));
 imagen=uint16(round(double(calibracion)*65535/Blanco));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
figure(98);
imshow(calibracion);
[x,y]=ginput(2); %posicion de la tobera para su medida
pixel=floor(abs(y(1)-y(2))); %calibración de medidas verticales
%pixel=floor(abs(x(1)-x(2)));%calibración de medidas horizontales
%d=input('Introduce la equivalencia en mm:');

A=inputdlg('Equivalencia en mm:','Dato:',1);
B=length(str2num(char(A)));
i=1;
if B==0 || str2num(char(A(1)))==0
d=0;
i=0;
else
d=str2num(char(A(1)));
end

coef(i)=d/pixel; %coeficiente para transformar longitudes mm/pixel
end
coef=sum(coef)/length(coef);

if i==1
set(handles.pushbutton_calibrar,'UserData',coef);
set(handles.text_escala,'String',coef);
else
set(handles.pushbutton_calibrar,'UserData',0);
set(handles.text_escala,'String','No hay Escala');
end
close figure 98;
end
```

```
% --- Executes on button press in pushbutton_cargB.
function pushbutton_cargB_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_cargB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[archBur,direcBur]=uigetfile({'*.tif';'*.JPG';'*.bmp'},'Seleccione foto para analizar:');

if length(direcBur) == 1
NoBur='No se ha cargado ninguna imagen para analizar.';
set(handles.text_docBur,'String',NoBur);
cancelar=0;
set(handles.pushbutton_cargB,'UserData',cancelar)
else
ArchDireccionBur=[archBur cellstr(direcBur)];
```

```
%ultima posicion del vector es la localizacion del archivo
set(handles.text_docBur,'String',ArchDireccionBur);
set(handles.pushbutton_cargB,'UserData',ArchDireccionBur)

end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%



% --- Executes on button press in pushbutton_prop.
function pushbutton_prop_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_prop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ArchDireccionBur=get(handles.pushbutton_cargB,'UserData');
ArchDireccionCali=get(handles.pushbutton_cimcalibrar,'UserData');
calibrarescala=get(handles.pushbutton_calibrar,'UserData');

if length(ArchDireccionBur)==1||length(ArchDireccionBur)==0 || length(ArchDireccionCali)==1
|| length(ArchDireccionCali)==0 || calibrarescala==0

uiwait(msgbox({'        Imposible de calcular'},'ERROR','error'));

else

coef=str2num(get(handles.text_escala,'String'));
coef2=coef^2;
coef3=coef^3;


n=length(ArchDireccionBur);
direccion= char(ArchDireccionBur(n));
cd(direccion);
imagen=imread(char(ArchDireccionBur(1)));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%PARA LEER IMAGENES EN FORMATO .TIFF
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tiff=char(ArchDireccionBur(1));
```

```
ntiff=length(tiff);
nmtiff=[];
for i=ntiff:-1:(ntiff-2)
nmtiff=[nmtiff tiff(i)];
end
if  nmtiff == ['FIT']
 Blanco=double(max(max(imagen)));
 imagen=uint16(round(double(imagen)*65535/Blanco));
 %imagen=uint8(imagen/256);
end
if nmtiff == ['fit']
 Blanco=double(max(max(imagen)));
 imagen=uint16(round(double(imagen)*65535/Blanco));
 %imagen=uint8(imagen/256);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(99);imshow(imagen);
[xfinal,yfinal]=ginput(2);
xfinal=round(xfinal);yfinal=round(yfinal);
imagen=imagen(yfinal(1):yfinal(2),xfinal(1):xfinal(2));


figure(1);imshow(imagen);

bina=im2bw(imagen,graythresh(imagen));
figure(1000001); imshow(bina);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%5
bina=bina==0; %para burbujas negras en fondo claro
%bina=bina==1; %para burbujas blancas en fondo negro
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
figure(1000002); imshow(bina);

% remove all object containing fewer than 30 pixels
bina = bwareaopen(bina,30);%JUGAR CON ESTE VALOR PARA ELIMIAR
%No meter numeros grandes porque tarda

% fill a gap in the pen's cap
se = strel('disk',30);%JUGAR CON ESTE VALOR PARA ADAPTAR
%No meter numeros grandes porque tarda
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Comentar en el caso de comprobacion de imágenes
bina= imclose(bina,se);
bina=bwareaopen(bina,2000);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% fill any holes, so that regionprops can be used to estimate
% the area enclosed by each of the boundaries
bina= imfill(bina,'holes');
figure(1000004); imshow(bina);
```

```
[B,L,N] = bwboundaries(bina,'holes');
figure(1000003); imshow(bina);
figure(2)
%imagesc(bina)
imshow(imagen);
hold on
for i=1:length(B);
   plot(B{i}(:,2),B{i}(:,1),'r','linewidth',1)
end
hold off


%CALCULA LAS AREAS DE DENTRO
CC = bwconncomp(bina); %calcula area de la burbuja y de la tobera
S = regionprops(CC,'Solidity','Area');
%disp(S(1));%numero de pixeles totales encerrados
area=[S.Area];

mm2=max(area)*coef2; %area de la burbuja en mm^2
Rarea=(mm2/pi)^(0.5);
%disp('El area de la burbuja es de:');disp(mm2);disp('mm^2');

%calulo del volumen
CELL=cell2mat(B);
figure(25);
plot(CELL(:,2),CELL(:,1)); %x=colum2, y=colum1

x=CELL(:,2);
y=CELL(:,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
%calcular el centro a partir de los discos de la deteccion de ambos puntos
%ademas del calculo de los anteriores
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

%CALCULO DEL VOLUMEN POR DISCOS DE DOS PUNTOS ENFRENTADOS
%Discos

DD=[];
for i=min(y):1:max(y)
k=find(i==y);
a=max(x(k));
b=min(x(k));
DD=[DD,a-b];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%VOLUMEN DE LA BUBUJA POR SUPERPOSICIÓN DE CILINDROS
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
figure (100)
[X,Y,Z] = cylinder(DD/2*coef);
surf(X,Y,Z)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
VolDD=0;
ADD=0;
for i=1:1:length(DD);
VolDD=VolDD+pi*DD(i)^2*0.25*1;
ADD=ADD+abs(DD(i))*1; %calcula el area de nuevo mm^2
end
VolDD=VolDD*coef^3;
ADD=ADD*coef^2;
RVolDD=(VolDD*3*0.25/pi)^(1/3);

set(handles.text_VolDD,'String',VolDD);
set(handles.text_ADD,'String',ADD);
set(handles.text_RVolDD,'String',RVolDD);


%CALCULO DE LOS PUNTOS Y DEL VOLUMEN POR EL LADO DE LA DERECHA

iniciox=floor((min(x)+max(x))/2); %coge la parte entera
inicioy=floor((min(y)+max(y))/2);


ptos=[];
for i=1:length(x);
if CELL(i,2) >=iniciox %&& CELL(i,1)>=inicioy
   ptos=[ptos; x(i),y(i)];
end
end

x=ptos(:,1)-iniciox;
y=ptos(:,2)-min(y);

Aderecha=trapz(y,x)*2*coef2; %calcula el area con los trapecios

figure(26);
plot(x,y,'.') % x y

% Volumen = Sumatoria de pi*R^2*Az  Az=1 pixel  R=vector en que corresponda
% a cada Az
R=[];
y(length(x)+1)=y(length(x));

for i=1:1:length(x); %calulo de los volumenes gracias a la sumatoria de trapecios
   if y(i+1)> y(i)
   R=[R;x(i)];
```

```
   end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
Vol=0;AD=0;
for i=1:1:length(R);
Vol=Vol+pi*R(i)^2*1;
AD=AD+abs(R(i))*1; %calcula el area de nuevo mm^2
end

AD=AD*coef2*2;
Vol=Vol*coef3;
%disp('El volumen por la derecha es de:');disp(Vol);disp('mm^3');
Rvder=(Vol*3*0.25/pi)^(1/3);


% %CALCULO DE LOS PUNTOS Y DEL VOLUMEN POR EL LADO DE LA izquierda
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
x=CELL(:,2);
y=CELL(:,1);

ptos2=[];
for i=1:length(x);
if CELL(i,2)<= iniciox %&& CELL(i,1)>=inicioy
   ptos2=[ptos2; x(i),y(i)];
end
end

x=ptos2(:,1)-iniciox;
y=ptos2(:,2)-min(y);

Aizquierda=trapz(y,x)*2*coef2;

figure(36);
plot(x,y,'.') % x y

% Volumen = Sumatoria de pi*R^2*Az  Az=1 pixel  R=vector en que corresponda
% a cada Az
Ri=[];
y(length(x)+1)=y(length(x));
%sentido de las agujas del reloj
for i=1:1:length(x);
   if y(i+1)<y(i)
   Ri=[Ri;x(i)];
   end
end

Voli=0;AI=0;
for i=1:1:length(Ri);
Voli=Voli+pi*Ri(i)^2*1;
```

```
AI=AI+abs(Ri(i))*1;
end

AI=AI*coef2*2;%calcula tambien el area mm^2
Voli=Voli*coef3;
%disp('El volumen por la izquierda es de:');disp(Voli);disp('mm^3');
Rvizq=(Voli*3*0.25/pi)^(1/3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%RADIOS
%disp('Radio según area:');disp(Rarea);disp('mm');
%disp('Radio según volumen derecha:');disp(Rvder);disp('mm');
%disp('Radio según volumen izquierda:');disp(Rvizq);disp('mm');
set(handles.text_RA,'String',Rarea);
set(handles.text_RVD,'String',Rvder);
set(handles.text_RVI,'String',Rvizq);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
set(handles.text_VD,'String',Vol);
set(handles.text_VI,'String',Voli);
set(handles.text_A,'String',mm2);


TSA=str2num(get(handles.edit_TSA,'String'));% TENSION SUPEFICIAL AGUA
DSA=str2num(get(handles.edit_DSA,'String'));%densidad del agua
DAR=str2num(get(handles.edit_DAR,'String'));%densidad del aire
VCA=str2num(get(handles.edit_VCAR,'String'));%viscosidad cinematica del agua
L=str2num(get(handles.text_RVolDD,'String'))*10^-3; %RADIO EQUIVALENTE VOLUMEN SIN
SIMETRÍA

CIRCU=str2num(get(handles.edit_circu,'String'));
DVORTI=str2num(get(handles.edit_DVortice,'String'));
DAGUJA=str2num(get(handles.edit_Daguja,'String'));



GAL=sqrt(9.81*(L)^3/VCA^2);
set(handles.text_GAL,'String',GAL);
BOND=9.81*L^2*(DSA-DAR)/TSA;
set(handles.text_BOND,'String',BOND);

We=DSA*(CIRCU/(pi*DVORTI))^2/(TSA/L);
set(handles.text_We,'String',We);

%VOLUMEN DE FRITZ CON CAUDALES INFERIORES AL CRÍTICO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
% Para comprobar si estamos dentro de un estado cuasiestático, debemos
% comparar el radio con el obtenido a partir de la aproximación de Fritz
%
%TAMAÑO DE LA AGUJA/NEEDLE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
aa=DAGUJA/2; % m radio de la aguja que genera la burbuja Rf>>>>>>>aa PARA QUE SE
CUMPLA
%aa=0.00863e-3;
%PROBAR CON BURBUJAS NO GENERADAS CON SUSTANCIA HIDROFÓBICA
RFRITZ=(3*TSA*aa/(2*9.81*DSA))^(1/3)*10^3;%mm
VolFritz=4/3*pi*RFRITZ^3;%mm^3
VolFritz2=2*pi*aa*10^9*TSA/(9.81*DSA);
QcritFRITZ=pi*(16/(3*9.81^2))^(1/6)*(TSA*aa/DSA)^(5/6)*10^9;%mm^3/s

set(handles.text_RFRITZ,'String',RFRITZ);
set(handles.text_VFRITZ,'String',VolFritz);
set(handles.text_QFRITZ,'String',QcritFRITZ);
%HABIENDO REALIZADO LA COMPROBACION CON ACEITE DE SILICONA SE CUMPLE EL
%PRINCIPIO TEÓRICO.


end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%


function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end


% --- Executes during object creation, after setting all properties.
function text3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% --- Executes on button press in pushbutton_reset.
function pushbutton_reset_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
reset='';

set(handles.text_RA,'String',reset);
set(handles.text_RVD,'String',reset);
set(handles.text_RVI,'String',reset);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
set(handles.text_VD,'String',reset);
set(handles.text_VI,'String',reset);
set(handles.text_A,'String',reset);

set(handles.pushbutton_calibrar,'UserData',0);
set(handles.text_escala,'String',reset);

set(handles.pushbutton_cargB,'UserData',0);
set(handles.pushbutton_cimcalibrar,'UserData',0);
set(handles.text_docBur,'String',reset);
set(handles.text_docCal,'String',reset);

set(handles.text_VolDD,'String',reset);
set(handles.text_ADD,'String',reset);
set(handles.text_RVolDD,'String',reset);

set(handles.text_RFRITZ,'String',reset);
set(handles.text_VFRITZ,'String',reset);
set(handles.text_QFRITZ,'String',reset);

set(handles.text_BOND,'String',reset);
set(handles.text_GAL,'String',reset);
set(handles.text_We,'String',reset);


% --- Executes during object creation, after setting all properties.
function text_docCal_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text_docCal (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% --- Executes during object creation, after setting all properties.
function text_A_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text_A (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% ------------------------------------------------------------------
function Ayuda_Callback(hObject, eventdata, handles)
% hObject    handle to Ayuda (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
uiwait(msgbox({'Información:';'Programa realizado para la caracterización de
burbujas.';'';'Localización:';'Área de Mecánica de Fluidos de la Universidad de Jaén.';'';'Autor
del programa: Carlos Avilés Coca';'';'Pasos a seguir:';'1) Añadir al path las carpetas y
subcarpetas dónde se encuentre el archivo principal';'2) Modificar los parámetros según sea
nuestro caso a analizar';'3) Cargar imagen para calibración';'4) Calibrar (seleccione dos puntos
e indique su medida real)repetir este proceso tantas veces hasta que la imagen de calibración
desaparezca';'5) Cargar imagen a analizar'; '6) Calcular';'7) Seleccionar el área de interés,
seleccione dos puntos de izquierda a derecha que formen un rectángulo que encierre el
elemento de estudio ';'';'Carga de Imágenes:';'En la pantalla aparecerán los nombres de los
archivos cargados acompañados al final de la dirección de tu PC donde se
encuentren.';'';'Comunicación con el programa:';'"Imposible de Calcular" Aparecerá en el caso
de que no haya archivos cargados.';'"No se ha cargado ningún archivo" Aparecerá si cancela la
opción de cargar imágenes.';'';'Salidas:';'Toda la información calculada se mostrará en la
ventana principal';} ,'MENÚ DE AYUDA','help'));


% --- Executes during object creation, after setting all properties.
function text_VolDD_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text_VolDD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function text_ADD_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text_ADD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function text_RVolDD_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text_RVolDD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
function edit_TSA_Callback(hObject, eventdata, handles)
% hObject    handle to edit_TSA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_TSA as text
%        str2double(get(hObject,'String')) returns contents of edit_TSA as a double
TSA = str2num(get(hObject,'String'));
set(handles.edit_TSA,'String',TSA);

% --- Executes during object creation, after setting all properties.
function edit_TSA_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_TSA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit_VCAR_Callback(hObject, eventdata, handles)
% hObject    handle to edit_VCAR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_VCAR as text
%        str2double(get(hObject,'String')) returns contents of edit_VCAR as a double
VCAR= str2num(get(hObject,'String'));
set(handles.edit_VCAR,'String',VCAR);

% --- Executes during object creation, after setting all properties.
function edit_VCAR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_VCAR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end



function edit_DSA_Callback(hObject, eventdata, handles)
% hObject    handle to edit_DSA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_DSA as text
%        str2double(get(hObject,'String')) returns contents of edit_DSA as a double
DSA = str2num(get(hObject,'String'));
set(handles.edit_DSA,'String',DSA);

% --- Executes during object creation, after setting all properties.
function edit_DSA_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_DSA (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_DAR_Callback(hObject, eventdata, handles)
% hObject    handle to edit_DAR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_DAR as text
%        str2double(get(hObject,'String')) returns contents of edit_DAR as a double
DAR = str2num(get(hObject,'String'));
set(handles.edit_DAR,'String',DAR);

% --- Executes during object creation, after setting all properties.
function edit_DAR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_DAR (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function text_GAL_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text_GAL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% --- Executes during object creation, after setting all properties.
function text_BOND_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text_BOND (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



function edit_DVortice_Callback(hObject, eventdata, handles)
% hObject    handle to edit_DVortice (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_DVortice as text
%        str2double(get(hObject,'String')) returns contents of edit_DVortice as a double


% --- Executes during object creation, after setting all properties.
function edit_DVortice_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_DVortice (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_circu_Callback(hObject, eventdata, handles)
% hObject    handle to edit_circu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_circu as text
%        str2double(get(hObject,'String')) returns contents of edit_circu as a double


% --- Executes during object creation, after setting all properties.
function edit_circu_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_circu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function edit_Daguja_Callback(hObject, eventdata, handles)
% hObject    handle to edit_Daguja (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_Daguja as text
%        str2double(get(hObject,'String')) returns contents of edit_Daguja as a double


% --- Executes during object creation, after setting all properties.
function edit_Daguja_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_Daguja (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes during object creation, after setting all properties.
function text_We_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text_We (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

Código del sistema electrónico de control de la EV (Arduino UNO):

```
//VARIABLES:
  // inicialización de las entradas y salidas:
int entrada=4;
int led=13;
int salida=2;
int llenado=8;
 // Inicialización de los parámetros característicos
int DelayTime=5000; //tiempo de espera en ms para que el sistema no admita entradas
int Open=70; //tiempo de apertura de la electroválvula
int Estable=6000; // tiempo necesario para que se estabilice el arduino en ms
int wait=1000; //tiempo que hay que esperar para que el vórtice salga a tiempo en ms



void setup() {
pinMode(entrada,INPUT);
pinMode(led,OUTPUT);
pinMode(salida, OUTPUT);
pinMode(llenado,INPUT);
delay(Estable);
}

void loop() {
if (digitalRead(entrada)==1){
delay(wait);//tiempo de espera para la interacción
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
digitalWrite(led,HIGH);
digitalWrite(salida,HIGH);
delay(Open);// tiempo que está abierta la electroválvula
digitalWrite(salida,LOW);
delay(DelayTime);//tiempo que el sistema no admite señales de entrada;
%%%%%%%%%%%%%%%%%%%
digitalWrite(led,LOW);
```

```
}
if (digitalRead(llenado)==1) {
digitalWrite(salida,HIGH);
digitalWrite(led,HIGH);
}
else if (digitalRead(llenado)!=1){
digitalWrite(salida,LOW);
digitalWrite(led,LOW);
}

}
```

BIBLIOGRAFÍA:

[1] UNIVERSIDAD DE VIGO (2011). *Departamento de Tecnología Electrónica.* <http://quintans.webs.uvigo.es/recursos/Web_electromagnetismo/magnetismo_rotacionalydivergencia.htm>

[2] ARÉVALO GONZÁLEZ, G. A. (2010). *Estudio experimental de la interacción de anillos de vorticidad en una topología tridimensional.* Tesis. Universidad de Chile, Santiago de Chile.

[3] ZEDNIKOVA, M., STANOVSKY, P., TRAVNICKOVA, T., ORVALHO, S., LADISLAV, H. Y VEJRAZKA, J. (2019). "Experiments on Bubble Breakup Induced by Collision with a Vortex Ring", *Chemical Engineering & Technology,* vol. 42, Issue 4, 843-850. Disponible en *Wiley Online Library.*

[4] CRESPO MARTÍNEZ, A. (2006). *Mecánica de Fluidos.* Madrid: Paraninfo.

[5] MARTÍNEZ BAZÁN, C. (2015). "About bubbles and vortex rings", *Journal of Fluid Mechanics,* Cambridge University Press, vol. 780, pp. 1-4.

[6] JHA, NARSING K. Y GORVARDHAN, R. N. (2015). "Interaction of a vortex ring with a single bubble: bubble and vorticity dynamics", *Journal of Fluid Mechanics,* Cambridge University Press, vol. 773, pp. 460-497.

[7] TRYGGESON, H. (2007). "Analytical Vortex Solutions to the Navier-Stokes Equation", *Växjö University Press*. Tesis. Växjö University, Suecia.

[8] OGUZ, H. N. Y PROSPERETTI, A. (1993). "Dinamics of bubble growth and detachment from a needle", *Journal of Fluid Mechanics,* Cambridge University Press, vol. 257, pp. 111-145.

[9] CHESTERS, A. K. (1977). "Modes of bubble growth in the slow-formation regime of nucleate pool boiling", *Int. J. Multiphase Flow*, vol. 4, pp. 279-302.

[10] LINARES UREÑA, A. (2017). *Estudio experimental de interacción de burbujas y anillos de vorticidad.* Trabajo Fin de Grado. Escuela Politécnica Superior de Jaén, España.

[11] THIELICKE, W. (2014). *The flapping flight of birds: Analysis and application.* Chapter 2: *Digital Particle Image Velocimetry*. Tesis. University of Groningen, Países Bajos.

[12] CANO-LOZANO, J. C., MARTÍNEZ-BAZÁN, C., MAGNAUDET, J. Y TCHOUFAG, J. (2016). "Path and wakes of deformable nearly spheroidal rising bubbles close to the transition to path instability", *Physical Review Fluids,* vol. 1, No. 5.

[13] REVUELTA, A. (2009). "On the interacction of a bubble and a vortex ring at high Reynolds numbers", *European Journal of Mechanics - B/Fluids*, vol. 29, issue 2, pp. 119-126.