UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA CIVIL Y AMBIENTAL

Profesor Patrocinante:
Dr. Verónica Lazcano Castro

DISEÑO DE SOFTWARE EDUCATIVO PARA EL ANALISIS DE FLUJOS DE TUBERÍAS

Proyecto de Título presentado en conformidad a los requisitos para obtener el título de Ingeniero Civil.

BRAULIO RAMÍREZ OLIVA

Concepción, abril de 2019.

Dedicatoria

A la mujer que me acompaña en esta vida frente a cualquier adversidad. Que ha confiado en mis capacidades y me apoya en todas las metas que me propongo. Gracias por tanto Karol.

AGRADECIMIENTOS

En primer lugar, agradecer a mis padres, que con sus esfuerzos y dedicación me ayudaron a culminar mi carrera profesional y brindaron el apoyo suficiente desde pequeño para no decaer cuando todo parecía complicado e imposible.

A mis amigos, hermana, tíos, abuelos y primos, por ser principales promotores de mis sueños, por confiar y creer en mis expectativas, por los consejos, valores y principios que me han inculcado.

Agradecer a los docentes de la universidad del Bío-Bío, por haber compartido sus conocimientos a lo largo de la preparación de nuestra profesión.

Finalmente quiero dedicar esta tesis a mis mejores compañeros de vida, Lilo, León y Kiara. Por ustedes este esfuerzo y sacrificio de años para poder entregarles todo el amor junto a Karol. Sin ustedes sería difícil crecer como persona, y gracias a su amor incondicional.

INDICE

DISEÑO DE SOFTWARE EDUCATIVO PARA EL ANALISIS DE FLUJOS DE TUBERÌAS

Autor: Braulio Patricio Ramírez Oliva

Departamento de Ingeniería Civil y Ambiental, Universidad del Bío-Bío

Correo Electrónico: braramir@alumnos.ubiobio.cl

Profesora Patrocinante: Verónica Angelica Lazcano Castro

Departamento de Ingeniería Civil y Ambiental, Universidad del Bío-Bío

Correo Electrónico: vlazcano@ubiobio.cl

RESUMEN

En el presente proyecto de título se busca realizar un software educativo para el análisis de sistemas de tuberías, específicamente analizar el flujo en tuberías a presión, el cual permitirá analizar variables que influyen en su comportamiento.

El objetivo del software diseñado en el actual proyecto de título es incentivar a los estudiantes a analizar diferentes comportamientos, logrando así aprender a desarrollar mayor la creatividad y manejar de forma más eficiente los conceptos de diseño en sistemas de tuberías.

Existe en este software la posibilidad de analizar diferentes modelos de sistemas de tuberías, tales como tuberías en paralelos, series o gasto distribuido, además de esquematizar las líneas de energía a partir de diversos parámetros, del cual se podrá evaluar singularidades, sifón o perdida de caudal a lo largo de la tubería.

El programa se confecciono a través del software de MATLAB, utilizando la plataforma GUIDE.

Palabras Claves: Flujo de tubería, Matlab, sistemas de tuberías.

5500 palabras Texto + 8*Figuras/Tablas*250 + 6*Figuras/Tablas*500= 10.500 Palabras aprox.

1 INTRODUCCIÓN

El presente documento corresponde a un proyecto desarrollado con el fin de responder a los requisitos exigidos por la Universidad del Bío-Bío en el proceso de titulación para la carrera de Ingeniería Civil.

El proyecto consiste en un software desarrollado con el fin de reducir los tiempos de análisis en el que se exigía resolver ejercicios que traían alguna complicación. Es por esto, que siempre se ha buscado desarrollar programas que apoyen el aprendizaje y que sean una herramienta para su resolución.

Para el estudio de la hidráulica, particularmente en el caso de flujo de tuberías, al momento de enfrentar y resolver un problema teórico practico, se deben manejar una cantidad de fórmulas requeridas para el proceso hidráulico, saber cuál de éstas escoger y de qué forma proponer una estrategia para poder llegar al resultado esperado.

En este proyecto de titulo se espera lograr un programa, el cuál no solo analice el procedimiento, sino también ofrezca una interfaz amigable donde pueda interactuar distintas variables y así analizar y comparar sus resultados.

Cabe mencionar que el uso de un software educativo hace fortalecer la decisión del futuro ingeniero y a la vez compromete al alumno con el aprendizaje.

1.1 Justificación del tema

En el estudio de la hidráulica, particularmente en el flujo de tuberías se debe analizar el comportamiento de sistemas de tuberías a través de múltiples cálculos, lo cual exige un manejo de los principios que rigen el movimiento. Comúnmente el alumno prioriza realizar cálculos y establecer un procedimiento correcto para obtener un resultado, de manera que se pierde el enfoque real del ejercicio que sería analizar el problema y sus respectivos solución.

Por lo anterior, como el alumno se centra en el cálculo, y se olvida del análisis nace la necesidad de crear un software amigable para el alumno, debido a que sería de gran ayuda la utilización de un programa para el apoyo educativo con el fin que el alumno pueda profundizar y entender de forma más clara todos los conceptos teóricos. Este programa resuelve problemas más simples en sistemas de tuberías, donde se aprecia el resultado con solo ingresar los valores correspondientes a la geometría y materialidad de la tubería en cada tramo.

1.2 Objetivos del proyecto de título.

1.2.1 Objetivo general.

- Diseñar un software educativo que permita evaluar, analizar e interpretar problemas de flujos de tuberías

1.2.2 Objetivos específicos.

- Identificar distintas plataformas para el diseño de un software educativo.
- Definir alternativas y propuestas para la creación del diseño conceptual del programa.
- Elaborar y desarrollar la interfaz gráfica.
- Validar interfaz a través de propuestas didácticas.

2 ANTEDECENTES TEÓRICOS

2.1 Sistemas Hidráulicos

La ingeniería hidráulica es la rama de la ingeniería civil, que se ocupa de la proyección y ejecución de obras relacionadas con el agua. Dentro de esta rama cumplen una función importante los diversos sistemas de tuberías que conforman estas obras, por lo anterior, es fundamental conocer todo tipo de aspecto relacionado a su comportamiento.

2.1.1 Tipos de sistemas de tuberías.

El transporte de un fluido requiere muchas veces considerar más de un conducto para su transporte, es por esto que se elaboran distintos sistemas de tuberías la cual pueden variar según lo requerido. A continuación, se presentará y describirá brevemente sistemas de tuberías.

2.1.1.1 Tuberías en serie.

Las tuberías en serie son aquel conjunto de tuberías que forman parte de una misma conducción y que tienen diferente diámetro. Para la obtención de una solución en el caso de tuberías en serie, se debe considerar la energía del sistema, como la suma de pérdidas de fracción que corresponda a cada tramo.

2.1.1.2 Tuberías en paralelo.

Las tuberías en paralelo son aquel conjunto de tuberías que comparten su misma energía en cualquiera de sus tramos. Como el comportamiento de las tuberías logra ser independiente de las otras, el caudal total del sistema es la suma de los caudales individuales a cada una de las tuberías.

2.1.1.3 Gasto en camino o gasto distribuido.

Sistema hidráulico en el cual un caudal, o gasto, se reparte a lo largo de su recorrido.

Aplicando la ecuación de continuidad a la tubería, se tiene que el caudal de salida (P) más la suma del gasto distribuido por la longitud de tubería es igual al caudal de ingreso (Q_0)

$$Q_0 = P + qL$$

Ecuación N°1: Ecuación de continuidad.

Desde la ecuación de Darcy – Weisbach para una tubería de iguales dimensiones y que no entrega gasto distribuido Ds es:

$$h_f = f\frac{L}{D}\frac{V^2}{2g} = \frac{8f}{\pi^2 D^2 g}\, L{Q_D}^2$$

Ecuación N°2: Ecuación perdida de fricción en un estrato con gasto distribuido.

Donde:

Q_D: es un caudal de diseño donde circularía por una tubería que no entrega gasto en camino, de material y dimensiones idénticas a las que entrega gasto y con igual pérdida

Obteniendo

$$Q_D = P + 0.55qL$$

Ecuación N°3: Ecuación de Continuidad.

2.1.1.4 Sifón

Un sifón es una estructura hidráulica que permite transportar un fluido en contorno cerrado, desde un nivel mayor de la superficie libre del escurrimiento y la vierte a una altura menor. Para el sifón invertido existen ciertas limitaciones en el funcionamiento debido a la existencia de bajas presiones cerca del vértice de este.

2.1.2 Pérdidas de Energía

Se considera perdida de carga a la energía del fluido necesaria para vencer la fricción, debido al rozamiento que experimentan las partículas del fluido con la pared. En cualquier sistema de tuberías existen dos tipos de pérdidas de cargas, como perdidas regulares y perdidas por singularidades.

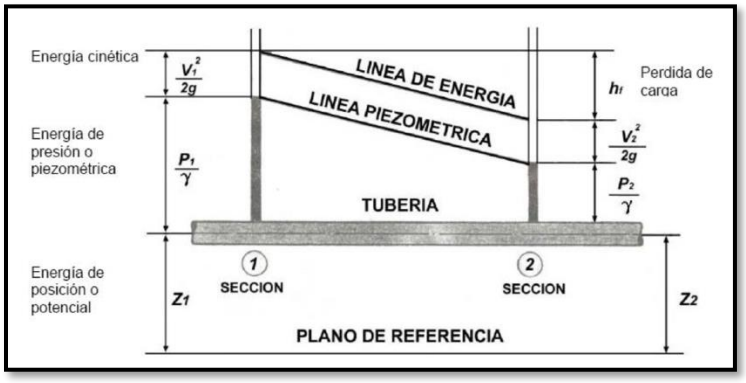


Figura N°1: Pérdida de energía en una conducción a presión.

(Fuente: “La energía del agua”. - www.iagua.es.)

$$z_1 + \frac{{V_1}^2}{2g} + \frac{P_1}{\gamma} = z_2 + \frac{{V_2}^2}{2g} + \frac{P_2}{\gamma} + Perdida\ de\ Energía$$

Ecuación N°3: Balance de energía con entre dos puntos con perdida de energía.

2.1.2.1 Perdidas fraccionales o regulares.

Las pérdidas fraccionales son las pérdidas que se presentan en conductos de sección constante y se producen debido al roce tanto como viscoso o con los contornos del elemento que contiene al fluido (tubería, cañería, canal, etc.).

Según la formulación de Darcy-Weisbach se define las pérdidas fraccionales como:

$$h_f = \frac{fLV^2}{2gD}$$

Ecuación N°4: Ecuación de Darcy-Weisbach.

Donde:

f =Factor de fricción

L =Longitud de la tubería [m]

V: =Velocidad media de flujo [m/s]

D = Diámetro interior de la tubería [m]

2g = Constante [m/s^2]

Para flujos transicionales se puede utilizar la ecuación implícita de Colebrook y White (Figura n°3), la cual requiere un procedimiento iterativo para su resolución. Este método se adapta a un programa computacional y es solucionado gracias al método de Newton Raphson, debido a su alta convergencia y precisión de resultados. Debido a esto determina y mejora la fidelidad del valor, comparado tradicionalmente por la forma manual de la gráfica de Moody.

$$\frac{1}{\sqrt{f}} = -2\log[\frac{\varepsilon}{3{,}7D} + \frac{5{,}1286}{Re^{0.29}}]$$

Ecuación N°5: Factor de fricción según Colebrook y White

Donde:

f = Factor de fricción.

ε = Rugosidad absoluta de la tubería. [m]

D = Diámetro interior de la tubería. [m]

Re = Numero de Reynolds.

Otra forma de calcular la perdida de fricción es a través de la ecuación de Hazen-Williams (Ecuación N°4), particularmente utilizada para determinar la velocidad del agua en tuberías con pared transicional, la cual contiene un coeficiente que está en función de la rugosidad de la tubería, la cual se considera Ch.

$$h_f = \frac{10.64LQ^{1.85}}{{C_H}^{1.85}D^{4.87}}$$

Ecuación N°6: Ecuación de Hazen - Williams

Donde:

C_H = Coeficiente de Hazen Williams, función de la rugosidad de la tubería

2.1.2.2 Pérdidas Singulares

Las pérdidas Singulares son las pérdidas presentes en los accesorios que se encuentran en las tuberías o son provocadas por los cambios en la dirección del flujo de fluido.

$$h_s = \frac{KV^2}{2g}$$

Ecuación N°7: Perdida de energía por singularidad.

Donde:

K =Constante de singularidad. Este valor depende de las características físicas del accesorio.

V =Velocidad media de flujo. [m/s]

2g =Constante [m/s^2]

2.2 Sistema Computacional

La estrecha relación entre los programas y los ingenieros o estudiantes de ingeniería resulta ser de gran utilidad, ya que permite relacionar los conocimientos académicos adquiridos con los problemas comunes presentes en la práctica. El uso de un programa debe basarse respecto al conocimiento total del problema a resolver, tal como: procedimiento, ecuaciones involucradas y herramientas matemáticas, con el objetivo de comprender de manera clara y correcta la interpretación de los resultados que arroja.

De acuerdo con lo anterior surge la necesidad de implementar un programa con el fin de analizar el cálculo de tuberías y líneas de flujos, donde se promueva el interés de revisar diferentes comportamientos, en el que se interactúe con la mayor cantidad de elementos posible, sin transgredir la claridad que merece un programa para la docencia.

2.2.1 Software educativo.

“Los buenos recursos educativos multimedia tienen un alto potencial didáctico ya que su carácter audiovisual e interactivo resalta atractivo y motivador para los estudiantes” (García, 2004).

"Se puede definir Modelo educativo como la forma en que se lleva a cabo el proceso enseñanza-aprendizaje. El concepto de proceso enseñanza aprendizaje encierra tres palabras que se deben tener muy claras al momento de buscar el perfil del alumno que se quiere producir" (Lillo, 1998).

Se considera implementar un sistema, que promueva la capacidad de motivación y que ayude a interactuar tanto como en realización de actividades, representaciones visuales, imágenes y resolución gráfica.

Si se compara la docencia del modelo tradicional sin software educativo y a la vez la docencia con el apoyo de un software educativo, se logra distinguir diferentes evaluaciones. A partir de las referencias "*7 claves para incorporar tecnología digital al proceso educativo*", EducaChile-2009 y *"Sistema WEB de apoyo a la optimización de los recursos y toma de decisiones de la PYME del área de transporte"* Sánchez Arias, René (2018), se pueden esbozar las siguientes comparaciones las cuáles serán sus ventajas y desventajas en el uso de un software educativo.

Tabla N°1: Docencia con apoyo de software educativo con respecto al modelo tradicional.

(Fuente: Elaboración propia.)

MODELO	VENTAJAS	DESVENTAJAS
Tradicional	Adquiere mejor control de los alumnos.	Relaciones comunicacionales complicadas si se presenta un numero alto de participantes en la sala
	Incentiva a practicar para promover el dominio del tema.	No logran dimensionar lo que se requiere diseñar.
	Promueve diferentes formas de aprendizaje para la explicación del tema.	No se encuentran didácticamente atentos.
Aplicación de tecnologías de información.	Muestra a los participantes compromiso y dedicación al tema.	Se pierde manejo del control de los alumnos si se trabaja en aulas.
	Logra identificar valores.	Repiten ejercicios planteados.
	Aprender de forma autónoma.	Potencia el individualismo, al existir una débil comunicación entre alumnos

2.2.2 Aplicaciones de la tecnología en la docencia

Incentivar a ejercitar al alumno y poder tener una resolución del problema es uno de los objetivos principales de un software educativo. La cual trata de producir y mejorar el proceso de enseñanza de la clase tradicional y a su vez complementar el proceso de enseñanza que dicta el profesor. Este software debe tener como condición estar familiarizado por el usuario para que este pueda complementar su uso y no producir lo contrario. ("7Claves para incorporar tecnología digital al proceso educativo", EducaChile-2009)

Al validar un programa se deben considerar las ventajas y facilidades que le brinda el software al usuario y así el alumno pueda revisar la información y temas a fines sin ninguna dificultad, pudiendo interiorizar de forma eficaz en el tema. Un punto sobresaliente es la potencia que debe tener el interfaz para que llegue a incentivar al usuario a ejecutarla, debido a que tendrá la mayor atención con el usuario y se buscara que sus utilizaciones de graficar y entregas de datos no sean ambiguos con respecto a lo que se quiere adquirir.

2.2.3 Lenguajes de herramientas computacionales

Existen diversas herramientas computacionales que se utilizan para el diseño de un software educativo, entre ellos está el caso Visual Basic. En este caso se debe tener cuidado con la ambigüedad y la redundancia, en el sentido de que las figuras entreguen muy poca información o que se replique la información. Aunque sea un programa valido, muchas veces al ser recargada con información, debilita notoriamente la interfaz gráfica.

Por otro lado, Lenguaje Java, igual se utiliza como herramienta computacional, en el cual presenta plataforma independiente y ejecutable sin importar la arquitectura de la computadora. Presenta una interfaz gráfica que permite una gran interacción con el usuario, debido su gran nivel de máquina virtual. Esta plataforma sirve para demostrar una infinidad de ejercicios, se adapta a cualquier tipo de elemento multimedia, pero a la vez presenta un alto nivel de lenguaje de programación y también que su lenguaje evoluciona muy lento.

Las dos herramientas computacionales anteriormente descritos presentan cualidades que generalmente no es percibida por los alumnos, y es que su presentación, menús y pantallas son las estandarizadas por el sistema operativo.

Para mejorar la habilidad de lenguaje y además facilitar el aprendizaje se conocerá una plataforma que es utilizada comúnmente por el alumno, ya que a su vez es de fácil acceso y genera una reacción favorable al utilizarla. Es Matlab una plataforma de desarrollo, investigación y análisis. Más usado comúnmente en universidades o centro de investigación. Se adapta a cualquier tipo de optimización, es rápido ejecutable y además de alta precisión. A pesar de que presenta una desventaja, de que solo puede realizar un solo cálculo

en tiempo de ejecución. Esto traduce a que el alumno no podría realizar comparaciones entre distintas opciones sin tener que abrir de forma paralela el programa. De acuerdo con las propuestas entregadas en Matlab es una de las plataformas que más rápido evalúa y verifica los valores entregados.

La tabla N°2 muestra una comparación entre las 3 herramientas computacionales descritas anteriormente, que detallan en sus ventajas y sus desventajas.

Tabla N°2: Comparación de herramientas computacionales: Java, Visual Basic y Matlab.

(Fuente: Elaboración propia.)

Modelo	Ventajas	Desventajas
Java	Plataforma independiente y multifuncional.	Contiene una filosofía de lenguaje basado a objetos
	Orientados a Objetos.	Exige un nivel computacional alto para su diseño.
	Fácil lenguaje de programación tanto como diseño gráfico.	Su lenguaje evoluciona lentamente
Visual Basic	Familiarizado por el usuario, al ser compilado por Excel.	No soporta información gráfica.
	Presenta diseños y formularios predeterminado por Windows.	Incapaz de crear plataformas multihilos
	Compatible con Office.	Lenguaje basado a objetos y no orientados a esos
Matlab	Perfecto en optimización. Rápido en ejecución y de alta precisión.	Problemas eventuales a velocidad
	Amplio soporte matemático.	Proceso laborioso susceptible a cometer errores de programación
	Plataformas prediseñadas para trabajar en conjunto.	Distribución de ejecutables.

Considerando que estos tres tipos de software cumplen con lo establecido inicialmente, es de principal importancia saber cuál de ellos cumple con todos los estándares que se proyectan una vez finalizado el proceso. Al ser evaluados cada uno de los casos, se deben considerar factores externos que afectan directamente a la hora de escoger el óptimo.

Matlab es una plataforma orientada para llevar a cabo proyectos en donde se encuentran cálculos matemáticos y representaciones de graficas. Matlab dispone de un amplio abanico de programas de apoyo especializado, denominado Toolboxes, que extienden significamente el número de funciones incorporadas en el programa principal. Estos Toolboxes cubren prácticamente casi todas las áreas principales de ingeniería y simulación.

Matlab presenta una plataforma con interfaces graficas con el usuario, conocido como GUI. Estás permiten con un control sencillo para elaborar aplicaciones sin la necesidad de aprender un lenguaje o a escribir comandos a fin de ejecutar la aplicación.

A través de este software Matlab propone automatizar una tarea o un cálculo. Para esto incorpora GUI, que incluye todos los controles gráficos, como menús, barra de herramientas, botones y controles deslizantes.

GUIDE (entorno de desarrollo de GUI) es una herramienta capaz e diseñar interfaz de usuario para diversas aplicaciones personalizadas. Es por esto que se puede construir una interfaz, con el fin de interactuar en la misma ventana.

3 METODOLOGIA

3.1 Metodología para el desarrollo del software

La metodología que se utilizó en el desarrollo del proyecto fue la de desarrollo iterativo incremental. Este proyecto de titulo se planifica en diversos bloques temporales llamado iteraciones, los cuales se conforman en varias etapas de análisis, diseño, programación y pruebas antes de una implementación y una evaluación de parte de los desarrolladores y el cliente en conjunto tal como se visualiza en la siguiente ilustración.

Figura N°2: Método iterativo incremental.

(Fuente: Sánchez Arias, René (2018). *Sistema WEB de apoyo a la optimización de los recursos y toma de decisiones de la PYME del área de transporte*.)

3.2 Diagrama de flujo

Se desarrolla un diagrama de flujo que detalla la lógica del programa. En la figura N°3, se presenta el diagrama de flujo, el cual exhibe los paso a paso del programa.

Este diagrama se origina a partir de los datos solicitados, el cual se denominara como ingreso de datos. Los valores ingresados se calificarán si son consistentes, de tal forma que puedan continuar con el proceso. A este proceso desde el ingreso de datos hasta el proceso de verificación lo denominaremos módulo de incorporación.

Una vez superada esta etapa de verificación, se inicia el proceso de cálculo dependiendo del tipo de variable que se quiere analizar. Es aquí en este proceso donde se encontrara todo proceso matemático necesario para la resolución del problema. Para finalmente entregar la información de forma gráfica o a través de información numérica. Desde que discretizamos la información hasta poder llegar a la recepción de datos denominaremos este tramo como módulo de cálculo.

Para finalizar este proceso, esta información es enviada al caso correspondiente según lo que se requiere analizar. En el caso que se requiera presentación de gráfico, se deberá diseñar curvas las cuales representen al sistema y puedan ser visualizadas por el usuario. Si se requiere información con respecto a la entrega de datos, se definirán las dimensiones que correspondan al valor que debiese ser mostrado.

Cabe destacar que el código que se utiliza en el proyecto, en la mayoría de los casos presenta una similitud en apariencia, siendo algunos casos excepcionales donde podría haber alguna diferencia que dependería de lo que se quiere evaluar. Pero en general, existe una solitud de datos, verificación de valores, cálculo de variable y finalización de proceso. Solo marca diferencia que, al tratarse de sistemas de tuberías, en estos casos no existiría una gráfica asociada.

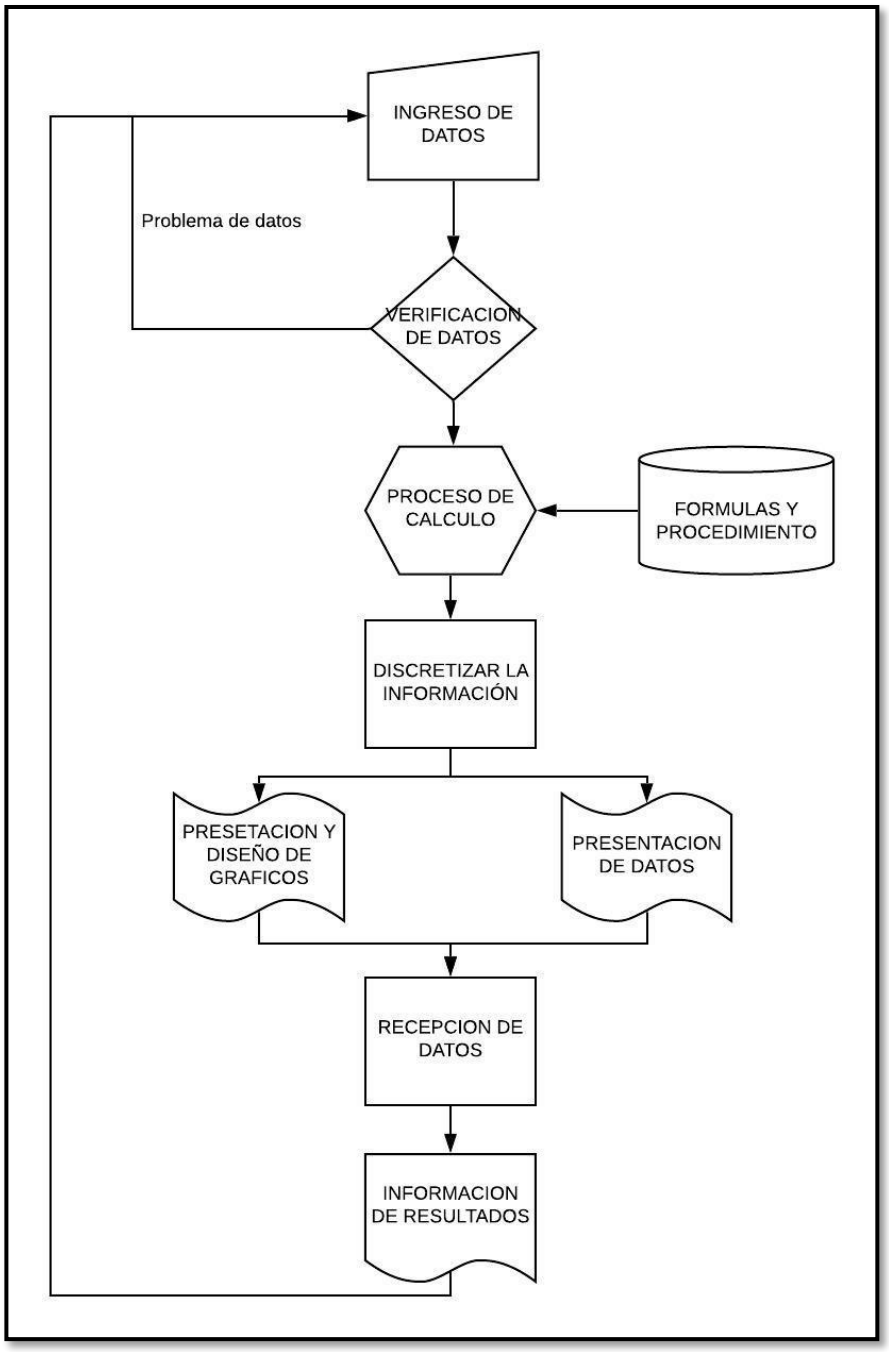


Figura N°3: Diagrama de flujo general del software educativo.

(Fuente: Elaboración propia)

4 DISEÑO DEL SOFTWARE

En el siguiente diagrama de menús, se pretende informar sobre el funcionamiento general del programa. El diseño del software comienza con una presentación, que tiene como objetivo, dar a conocer el programa, indicar acerca del autor y la institución que se encuentra como respaldo.

Una vez en la pantalla de inicio de análisis, donde se visualiza la selección de datos, se muestran 2 posibilidades de acción, la cual sería líneas de flujos o sistemas de tuberías. En ambas opciones se encuentran ejercicios determinados, como se refleja en la figura N°4

Independiente del ejercicio que se quiera analizar, se considera el mismo procedimiento descrito en el diagrama de flujo (Figura N°3). Este procedimiento se verá reflejado en la interfaz, y será explicado en el tema a continuación.

Cabe recordar que se consideró la opción de realizar este programa en la aplicación Matlab, debido a que contiene la plataforma GUIDE. Además, para esquematizar la estructura se presenta la figura N°4

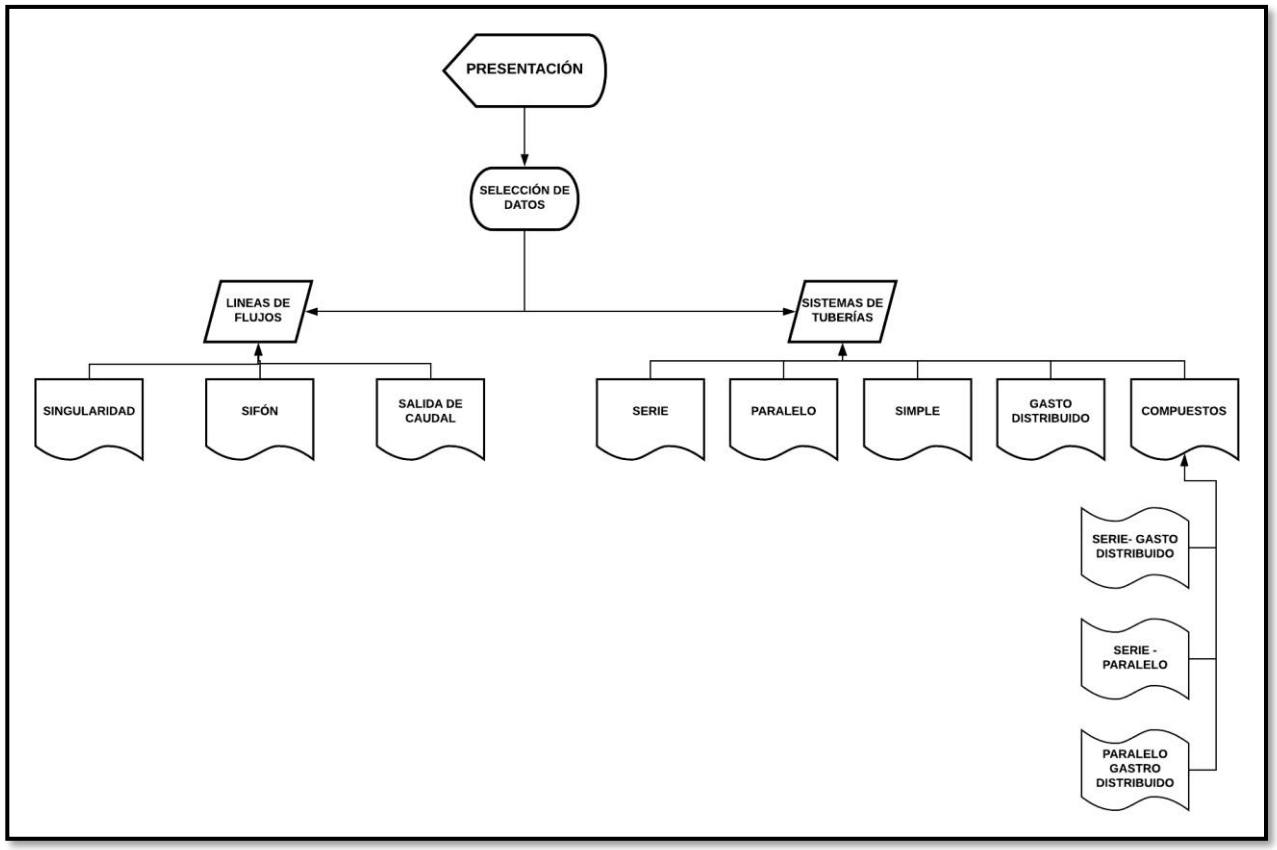


Figura N°4: Diagrama de menús.

(Fuente: Elaboración propia.)

4.1 Resultado del software.

La plataforma incorpora tres módulos, de incorporación, de cálculo y de entrega de datos. En cada uno de estos módulos existen variables que deben ser ingresadas por el usuario y valores que sean modificados por el programa.

4.1.1 Módulo de incorporación.

El módulo principal de este programa será el módulo de incorporación, que tiene como objetivo adquirir los valores que son dejados en pantalla por el usuario. Este proceso comienza desde el ingreso de datos, donde se deben ingresar los valores por el usuario para poder dirigirlo a la verificación de datos.

En la sección de ingreso de datos aparece el listado de que el usuario debe ingresar con las unidades de medidas respectivas.

Figura N°5: Cuadro ingreso de datos.

(Fuente: Elaboración propia.)

Al interior del módulo de ingreso existe la tarea de verificación de validez de los datos. En este caso a través del módulo de incorporación, se establece una rutina que evalúa el error al ingresar un dato, por ejemplo, agregar alguna letra en vez de un número, se mostrara un mensaje de error como la que se muestra en la siguiente forma. (Figura N°6)

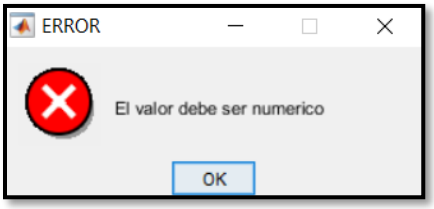


Figura N°6:Cuadro de error.

(Fuente: Elaboración propia.)

Luego de finalizar el proceso anterior de validación, estos valores serán incorporados al módulo de cálculo, el cual tiene como objetivo adquirir valores recopilados en el módulo de incorporación, almacenarlos y procesar a través del proceso iterativo. En aquel proceso se entrega todo el material algebraicamente necesario para que la información sea procesada y sea complementada con los valores ingresados.

4.1.2 Módulo de cálculo

Módulo de cálculo se encuentra en la parte de "radio botón" (figura N°7 o figura N°8), la cual es aquí donde se concentra la mayor parte del algoritmo del programa. Se debe considerar que el usuario no ve los algoritmos.

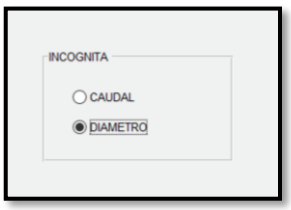


Figura N°7: Cuadro para cálculo tubería simple.

(Fuente: Elaboración propia.)

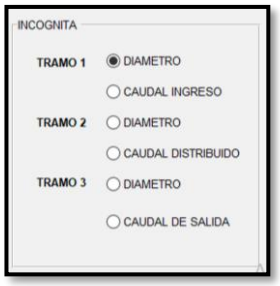


Figura N°8: Cuadro para cálculo tubería en serie.

(Fuente: Elaboración propia.)

El módulo de cálculo varía a partir del ejercicio a analizar. Esto corresponde a que cada ejercicio contiene un procedimiento diferente y presenta una solución especifica. A continuación, se visualiza cada incógnita que presentara cada sistema de tubería.

- Tubería simple: Al interior de este sistema de tubería encontramos el caudal y el diámetro como incógnita.
- Tubería en serie: Al tratarse de una tubería en serie, se podrá calcular el caudal del sistema, además del diámetro de cada tramo.
- Tubería en paralelo: Dado que se un caudal del sistema, se puede determinar el caudal por cada tramo y además el diámetro por tramo como incógnita.
- Gasto distribuido: Como incógnita en este caso encontramos el caudal de salida y el diámetro de la tubería.

Una vez realizado todo el proceso de verificación, se deberá reunir los datos de información obtenida y enviarla al módulo entrega de datos.

4.1.3 Modulo de entrega de resultados.

Cuando los valores sean recibidos por el módulo de entrega de resultados tendrá como objetivo entregar el resultado a partir de la información adquirida en el proceso anterior. Esta información se envía al módulo correspondiente según lo que se requiere analizar. En el caso que se requiera presentación de gráfico, se deberá diseñar curvas las cuales representen al sistema y puedan ser visualizadas por el usuario. Si se requiere información con respecto a la entrega de datos, se definirán las dimensiones que correspondan al valor que debiese ser mostrado.

Cabe recordar que, en caso de ingresar nuevamente valores al interfaz del programa, este automáticamente volverá al proceso inicial para poder adquirir nuevos mensajes de respuestas que se llevaran a visualización del usuario para que sean analizadas.

El mensaje que se recibe en el modulo de entrega es fundamental, ya que al ingresar en esta parte ya se sabrá lo que realmente quiere el alumno, es por eso que a partir de este modulo entrega de datos se confecciona grafica (Figura N°9) o también se presentan grafica de análisis (Figura N°10) para que el alumno lo analice.

Entonces este módulo presenta los datos de información referente al desarrollo del programa

Figura N°9: Resultado tubería simple.

(Fuente: Elaboración propia.)

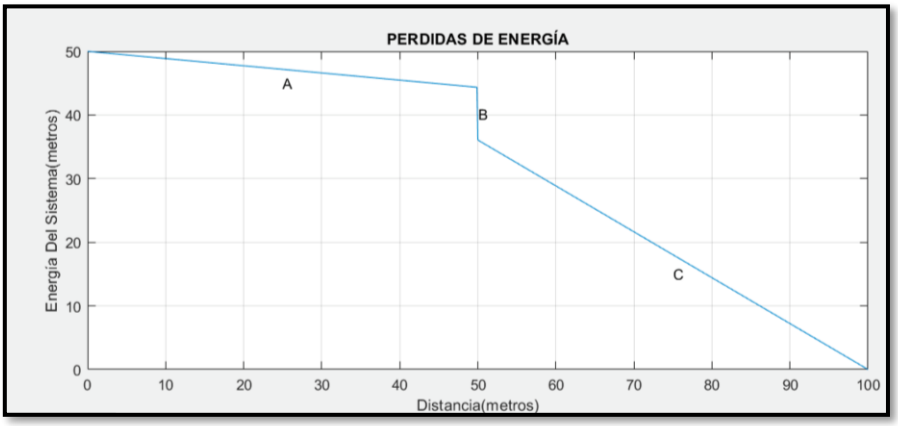


Figura N°10: Perdidas de energía en un sistema con salida de caudal.

(Fuente: Elaboración propia.)

El módulo de entrega de resultado varía a partir del ejercicio a analizar. Esto corresponde a que cada ejercicio contiene un procedimiento diferente y presenta una solución especifica. A continuación, se visualiza cada concepto que se adquiere de cada sistema de tubería.

- Tubería simple: Al tratar del método de de Darcy-Weisbach, se obtiene como resultado el caudal o diámetro, su número de Reynolds y su factor de fricción. Si se opta por el método de Hazen Williams se obtiene como resultado el diámetro o caudal.
- Tubería en serie: Al interior de este sistema de tubería en serie, se obtienen todos los resultados correspondientes al sistema, energía del sistema, caudal general o si se evalúa por tramos se obtiene su diámetro con su respectivo factor de fricción.
- Tubería en paralelo: Para este sistema de tubería, se analiza como resultado por tramo como caudal, diámetro y factor de fricción correspondiente.
- Gasto distribuido: Como resultados obtenemos el caudal de entrada, su número de Reynolds, su diámetro y factor de fricción.

Una vez realizado todo el proceso de verificación, se deberá reunir los datos de información obtenida y enviarla a la información de resultados.

5 VALIDACIÓN DEL PROGRAMA.

5.1 Tubería simple

Una tubería uniforme de 6000 m de longitud y rugosidad de 0.05 mm , transita un caudal de 200 litros/segundo, desde un estanque que se encuentra a 1905 m.s.n.m y pretende llegar a un estanque que se encuentra a 1795 m.s.n.m. Calcular su diámetro.

Resolución.

Para ingresar los datos al sistema, primero se debe seleccionar sistema de tubería simple, la cual se dará a elección dos alternativas a desarrollar. Para este caso se utilizara el método de Darcy-Weisbach. Se recomienda visualizar las unidades que se entrega el problema, con respecto a lo que señala el programa.

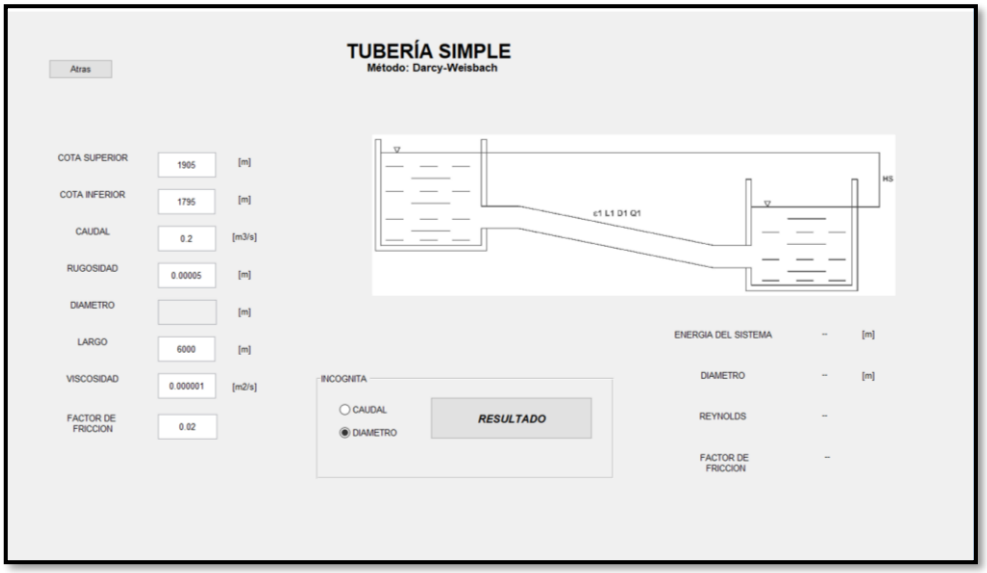


Figura N°11: Ingreso de Valores. Sistema de tubería simple

(Fuente: Elaboración propia.)

En la figura se muestra los valores agregados al sector izquierdo, con su respectivo cambio de unidad correspondiente. Se destaca que los valores de factor de fricción y viscosidad cinemática, ya se encuentran predeterminados.

Una vez que se selecciona la incógnita, en este caso seria el diámetro. Y seleccionando el botón "RESULTADO" arroja en pantalla el valor correspondiente al diámetro de la ecuación.

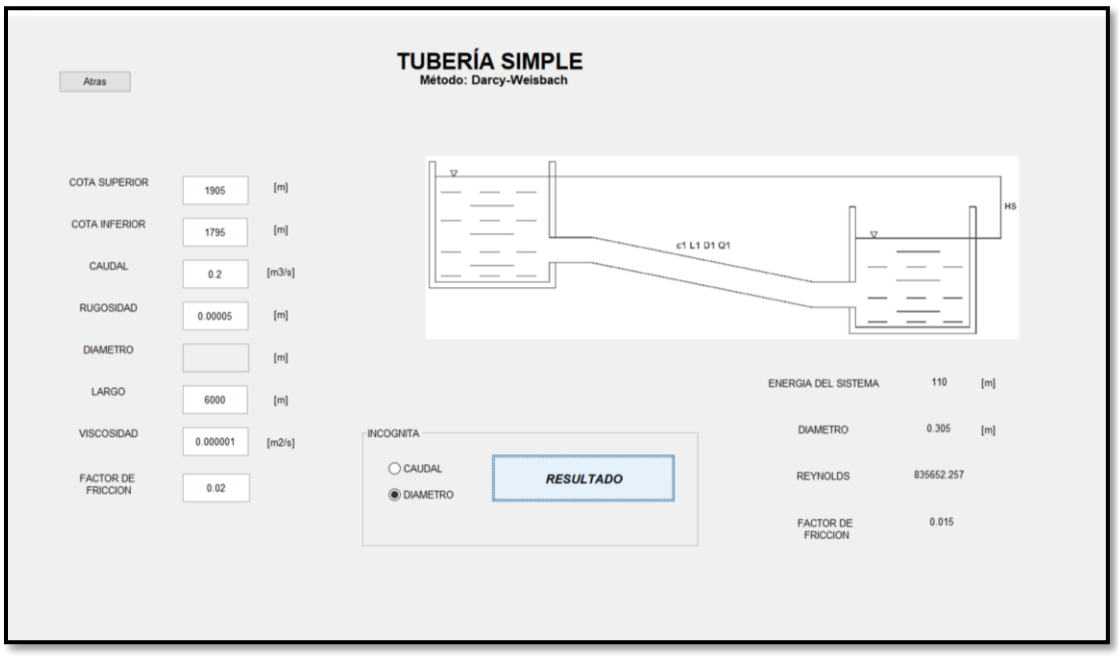


Figura N°12: Resultados. Sistema de tubería simple

(Fuente: Elaboración propia.)

A partir de los resultados arrojados en pantalla, es ahí donde comienza el análisis del estudiante. Desde ese momento es importante analizar y incentivar al alumno a interactuar con el resultado. En este caso de problema su incógnita era el diámetro, por lo tanto, se pueden debatir más de dos opiniones. Una de ellas seria el caso que el diámetro que se arrojó no es el adecuado con lo que se requiere en terreno, la cual se tendría que analizar otro caso de sistema de tubería para transportar un caudal menor, y de esta forma disminuir el diámetro. Otro de los casos a analizar, será que el programa arroja un diámetro del cual, no se encuentra normalizado, es por esto que el alumno tendría que sugerir un diámetro comercial mayor al valor arrojado en pantalla. Y es aquí donde se podría utilizar el cambio de incógnita a partir de los mismos valores determinados, pero solamente que ahora la incógnita seria el caudal, también llamado caudal de diseño.

5.2 Tubería con singularidades.

Un embalse A entrega agua al embalse B a través de 2 tuberías, una es de 300 mm y la otra es de 200 mm de diámetro, Aguas arriba en el cambio de sección (el que se asume gradual), ocurre una singularidad, la primera tubería tiene una distancia de 500 metros de largo al igual que la segunda tubería, y contienen una rugosidad de 0.003m. La energía total disponible es de 25 m. Calcule el caudal de salida y dimensione el sistema suponiendo que contiene al comienzo una singularidad de 2, una final de 2.5 y en el cambio de sección una 0.5

Resolución.

Se debe considerar las mismas unidades que se encuentran en el programa antes de ingresar los valores. En este caso de ejercicio no explican la cota de superior o menor, es por esto que la energía total disponible se considerara como cota superior, para el caso de cota inferior se denominara como 0.

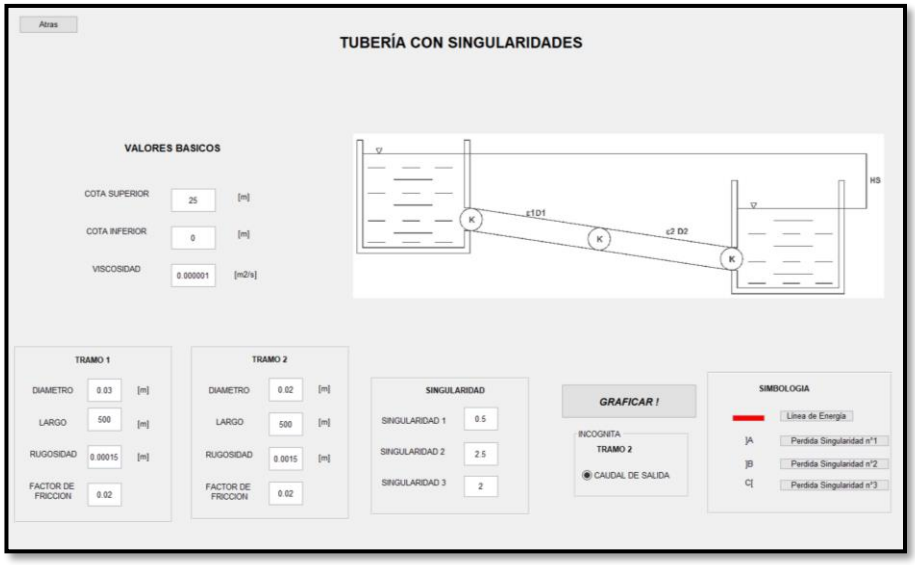


Figura N°13: Ingreso de Valores. Tubería con singularidades

(Fuente: Elaboración propia.)

Una vez apretado el botón “Graficar”, se demuestra en la grafica las perdidas de cada tramo, la cual se encuentra perdidas por singularidad, y además perdidas por fricción.

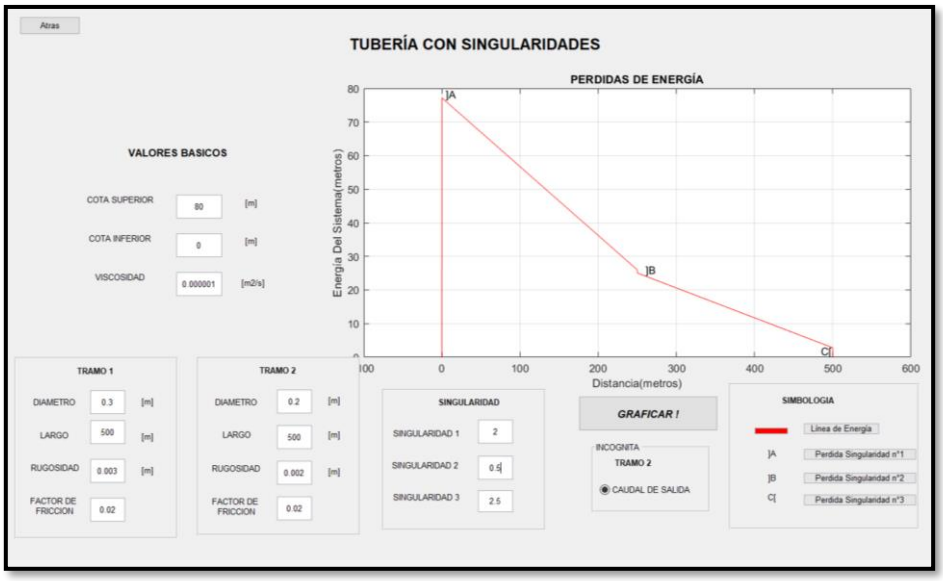


Figura N°14: Resultados. Tubería con singularidades.

(Fuente: Elaboración propia.)

Se podrá analizar estos casos a través de una gráfica, debido a que puede cuantificar cuanta energía va a llegar a cada punto. Al requerirse que llegue una cierta cantidad de energía al cambio de sección, se debería cambiar la geometría del primer tramo para que se requiera más o se ajuste de acuerdo con lo que se exija.

Se calcula la magnitud en el sistema que tiene cada perdida, las diferencias de tener singularidades mayores o menores en comparación a la perdida que existe por fricción.

6 CONCLUSIONES

La realización de este programa educativo permitirá mejorar el procedimiento correcto en el análisis de flujos de tuberías, aportando una mayor facilidad en la determinación de problemas en el área hidráulica.

Comparando las plataformas investigadas, dando a conocer sus ventajas y desventajas, se da por finalizar un programa más accesible y cercano al alumno, que demuestre con claridad que el programa es capacitado para elaborar una plataforma educacional con el objetivo de involucrarse didácticamente en el tema, lo que ayudara a incentivar a los alumnos a dimensionar conceptos teóricos de mejor forma y poder enfocar directamente la intención de diseñar sistemas hidráulicos. La principal tarea de este software educativo es analizar el problema y sus respectivas soluciones.

El programa fue diseñado con el lenguaje computacional Matlab, ya que, mediante los criterios de evaluación en nuestro proyecto de título, es el óptimo para realizar el software propuesto. Creando de esta forma una herramienta computacional que cumpla con los procesos de sistemas de tuberías, el cual resuelve problemas simples sin la necesidad de generar algún tipo de cálculo y a su vez, el alumno pueda ejercitar con el programa sin la tutela directa del profesor.

Esta herramienta computacional que funcionará como software educativo, queda propuesta a todo estudiante que quiera analizar diferentes condiciones. Incluso se encuentran propuestas de ejercicios compuestos de sistemas de tuberías, con el fin de que el alumno analice diferentes condiciones y sirvan de apoyo para desarrollar diferentes alternativas en su vida profesional.

Finalmente, se puede concluir que esta herramienta permitiría ayudar a disminuir el tiempo en la resolución de problemas, priorizando que el estudiante pueda dimensionar y analizar tanto el problema como el resultado de este.

Se invita además a continuar con este proyecto de título, para que se estudie si efectivamente mejora el rendimiento en los estudiantes o ayuda a analizar de forma más eficaz los resultados, ya que tiene como fin apoyar pedagógicamente al alumno.

7 BIBLIOGRAFÍA

- Bravo González, Adrián. *Software educativo para apoyar el proceso de enseñanza- aprendizaje de la asignatura Planeamiento y Operación de los Recursos Hidráulicos (PORH)* . - Disponibilidad: http://dspace.uclv.edu.cu/

- "Creación de apps con interfaces graficas de usuario en Matlab" Lugar de publicación: Mathworks Recuperado: https://la.mathworks.com/discovery/matlab-gui.html

- García Vidal, G. (2004*). Multimedia didáctica como vía para proporcionar el aprendizaje del tema: aspectos generales de las máquinas de corriente directa.*

- Lazcano Castro, Verónica. (2001). *Hidráulica en contornos cerrados*. Concepción: Universidad del Bío-Bío. Departamento de Ingeniería Civil, 2001.
- MATLAB User's Guide, The MathWorks, Inc., Massachusetts, 1995.
- Monje Redondo, Miguel. *"La energía del agua "*- Lugar de publicación: iagua Recuperado: https://www.iagua.es/blogs/miguel-angel-monge-redondo/fbh3-energia-agua

- Parada S, D. (2009). *7 claves para incorporar tecnología digital al proceso educativo*. [ebook] Santiago: Área de Educación Fundación País Digital. Available at: http://ww2.educarchile.cl/UserFiles/P0001/File/CR_Articulos/libro_siete_claves.pdf

- Pérez Correa, Rodrigo. (2001). *Diseño interactivo de alineamientos geométricos horizontales*. Concepción: Universidad del Bío-Bío. Departamento de Ingeniería Civil.

- Sánchez Arias, René (2018). *Sistema WEB de apoyo a la optimización de los recursos y toma de decisiones de la PYME del área de transporte.*

UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA CIVIL Y AMBIENTAL

Profesor Patrocinante:
Dr. Verónica Lazcano Castro

A N E X O S

DISEÑO DE SOFTWARE EDUCATIVO PARA EL ANALISIS DE FLUJOS DE TUBERÍAS

Proyecto de Título presentado en conformidad a los requisitos para obtener el título de Ingeniero Civil.

BRAULIO RAMÍREZ OLIVA

Concepción, Abril de 2019.

ANEXO

“MANUAL DEL USUARIO”

MANUAL DEL USUARIO.

1 INTRODUCCION

El programa denominado “Sistema de tuberías” diseñado a través de la plataforma MATLAB, es ejecutable en cualquier versión. Solamente es necesario que el computador tenga requisitos mínimos para instalar el producto, La velocidad de desarrollo del programa dependerá directamente con los componentes del computador.

En el programa se podrá encontrar diversas tareas, tales como:

- Cálculo de energía del sistema.
- Cálculos de variables y parámetros de diseño.
- Diseño de sistemas de tuberías compuestos para el análisis del problema
- Gráfico de diseño geométrico, utilizando plataformas propuestas
- Verificar la existencia de cavitación.

1.1 Recurso tecnológico para el desarrollo.

Se necesito el siguiente equipo para desarrollar el proyecto de una manera eficiente y completa.

Marca: MacBook pro

Modelo: 13-inch, 2017

Procesador: 2,3 GHz inter core i5

Memoria: 8 GB 2133 MHz LPDDR3

Disco duro: 121 GB.

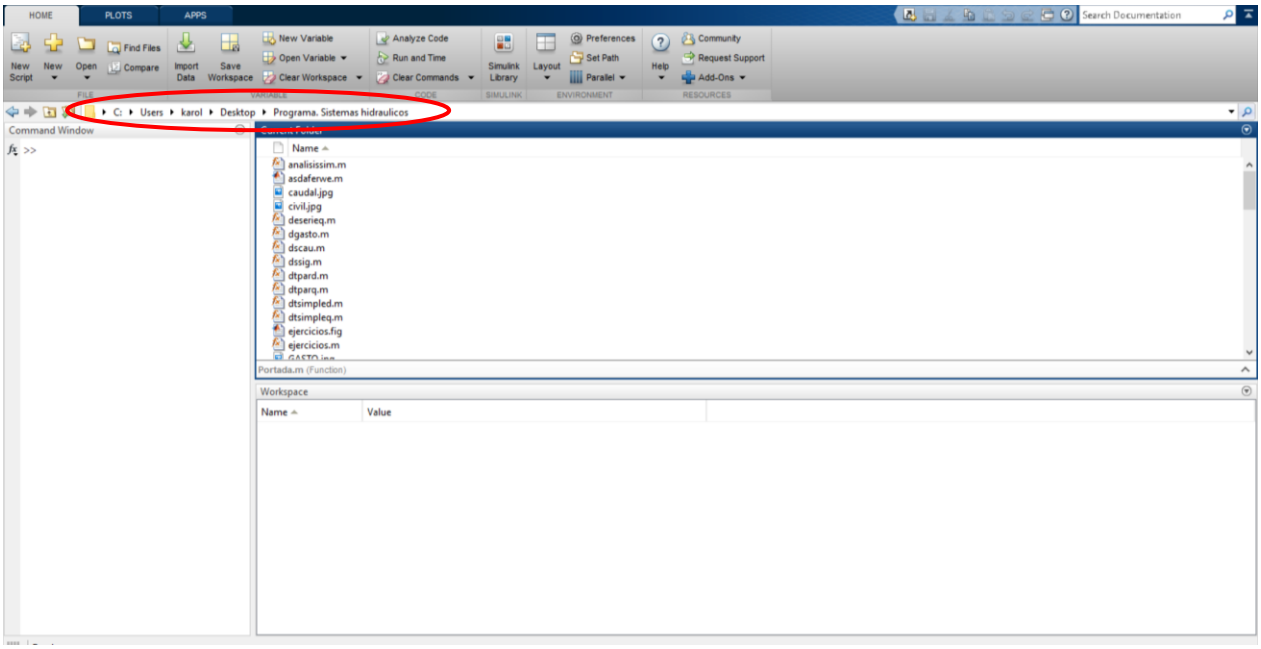
Pantalla: Integrada 13,3 pulgadas (2560x1600)

Sistema Operativo: macOS High Sierra versión 10.13.6

2 INSTRUCCIÓN PARA EL DISEÑO

Para iniciar la plataforma, haz doble clic sobre el icono para poder comenzar con el proceso.

Una vez que la ventana se haya ejecutado, debes localizar la carpeta en donde se encuentra el programa. En esta oportunidad, dejaremos que la carpeta se encuentre en nuestro escritorio con el nombre de “Programa. Sistemas de tuberías”. En este caso, la localización de nuestra carpeta quedara designada con el siguiente código: C:\Users\karol\Desktop\Programa. Sistemas hidráulicos



Una vez seleccionado la ubicación de la carpeta, se cargaran automáticamente todos archivos del programa en la columna llamada “Current Folder”

Es aquí donde se debe buscar el archivo “Portada.m”, hacer doble click en el archivo y una vez abierto. Debes dirigirte en la parte superior del programa donde se encuentran, seleccionar el comando “Run” (Figura N°1). La cual emergerá una pantalla que dará bienvenida al programa (Figura N°2)

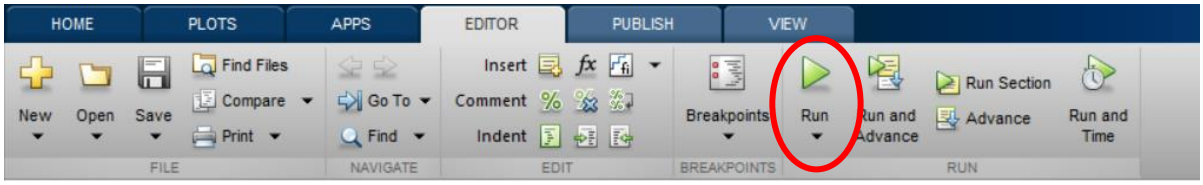


Figura N°1: Comando Matlab. “Run”

Figura N°2: Presentación

Al seleccionar el botón comenzar (Figura N°1). Emergerá una ventana la cual aparecerán dos opciones. Una opción es seleccionar el caso de líneas de flujo, la cual presenta el diseño gráfico del sistema.

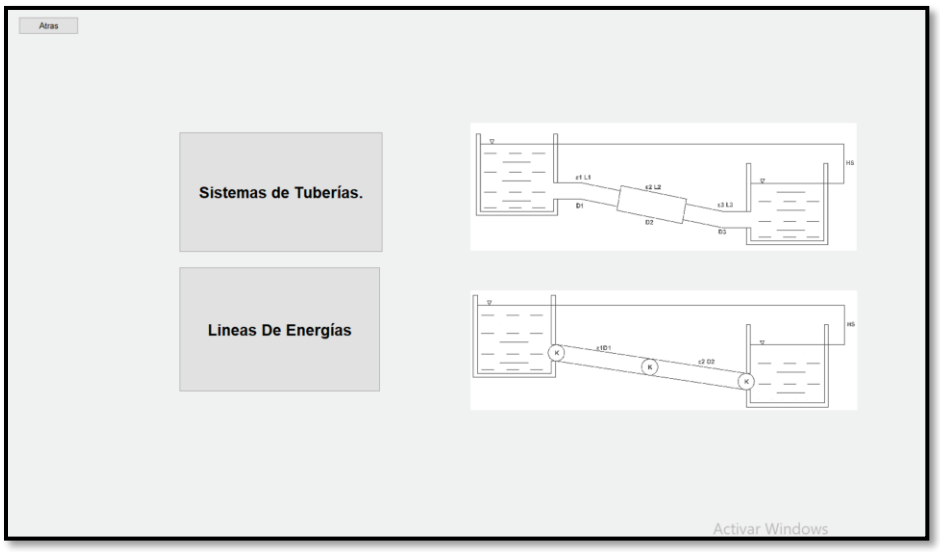


Figura N°3: Opción inicial de la figura.

Opción n°1: Sistemas de Tuberías.

1.-Al seleccionar sistema de tuberías, se encontra un listado (Figura N°4) de problemas que puedes llevar a analizar, es por esto que se debe elegir una para continuar con el proceso.

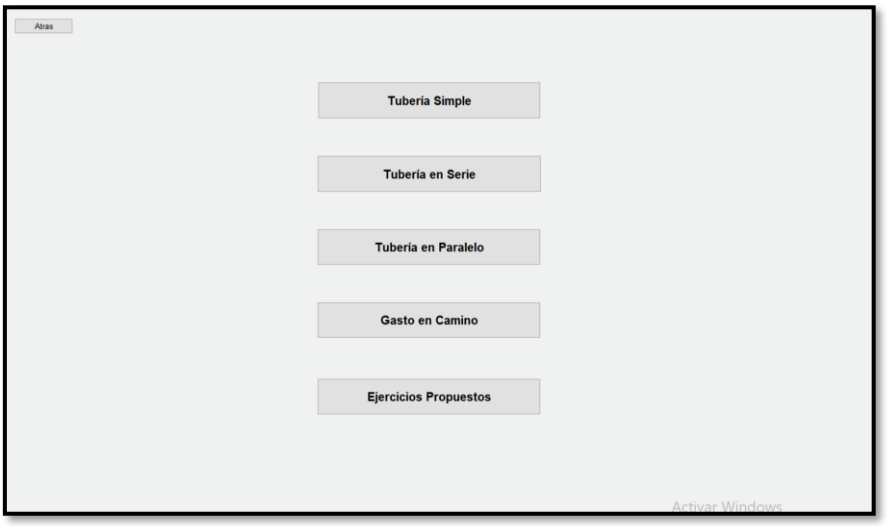


Figura N°4: Sistemas de Tuberías.

2.-Considerando la elección que se opta, debes ingresar los valores que se exigen en pantalla. Cuando comiences a completarla, deberás asumir que el valor ingresado tenga el mismo sistema de unidades que se expresa en pantalla (Figura N°5). En caso contrario debes hacer el cambio de unidad correspondiente para que los valores no se expresen erróneamente frente al resultado del problema

Figura N°5: Ingreso de valores.

3.- Al mismo modo, debes escoger tu incógnita la cual será la opción que podrás analizar (Figura N°6). Existen múltiples alternativas, es por esto que antes de comenzar con el desarrollo del programa debes escoger que alternativa vas a elegir.

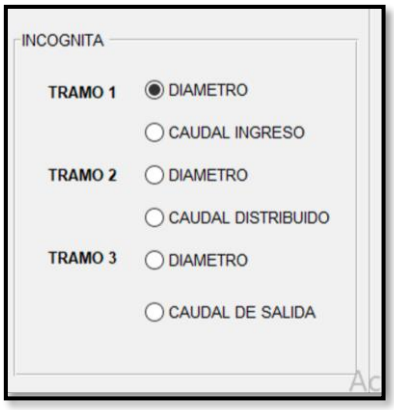


Figura N°6: Alternativa de incógnita

4.- Clickkeando en la etiqueta de desarrollo podrás visualizar que los valores ya fueron calculados y a la vez se encuentran en pantalla.

5.-Una vez que quieras reiterar este proceso, solamente debes cambiar algún valor inicial y volver al paso 2 para que puedas continuar con el proceso. O también puedes variar en cualquiera de las opciones (Figura N°7) solamente dejando el botón atrás.

Figura N°7: Botón de regreso.

Ejercicios

Como complemento al manual del programa, se presenta 2 propuesta para la visualización de ingreso de valores con la finalidad de analizar la resolución del problema.

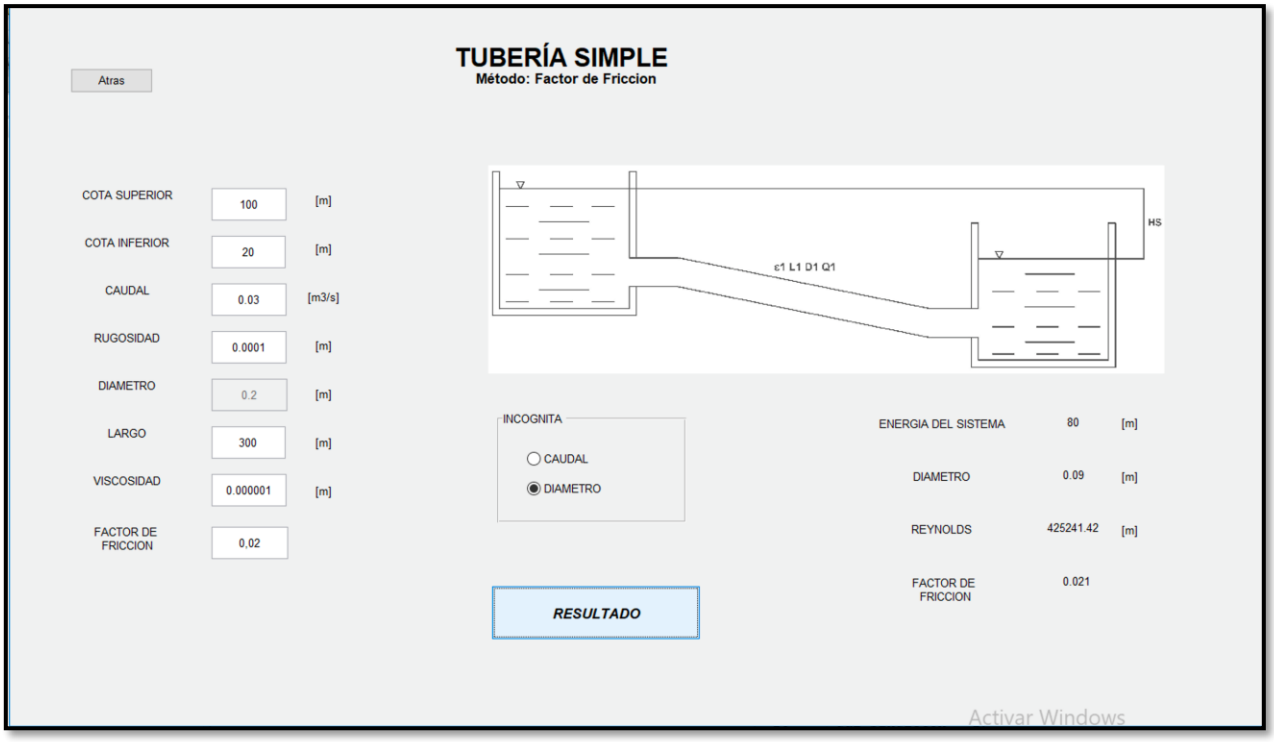
Tubería Simple (Figura N°8)

Ingreso de parámetros para el diseño geométrico.

Los resultados utilizados se expresan en la misma figura

Se podrá analizar el caso de caudal, considerar diámetro comercial para analizar de forma alternativa la propuesta de diseño

Figura N°8: Tubería simple – Método Factor de Fricción.



Al analizar los datos obtenidos, las interrogantes son:

¿Qué conllevó a obtener valores obtenidos? La respuesta esta en el procedimiento, dado que se encuentra en un proceso de iteración, se obtuvo que, al tener un factor de fricción de igual condiciones, se podría obtener el diámetro que muestra la figura.

¿Qué se podría analizar del diámetro obtenido? A partir del diámetro obtenido desde un caudal de diseño, este sería el valor mínimo que requiere el sistema para satisfacer al problema. A partir de este resultado, se podrá analizar el valor, aproximar a un estimado de diámetro comercial y estimar que cambio podría resultar disminuir o aumentar el diámetro en esta sección.

¿Que aporte tendría obtener el Reynolds como resultado? Obtener el Reynolds como resultado no significaría un aporte muy importante en el momento de diseño, pero sería de gran importancia para saber el comportamiento que tiene el fluido al interior de la tubería.

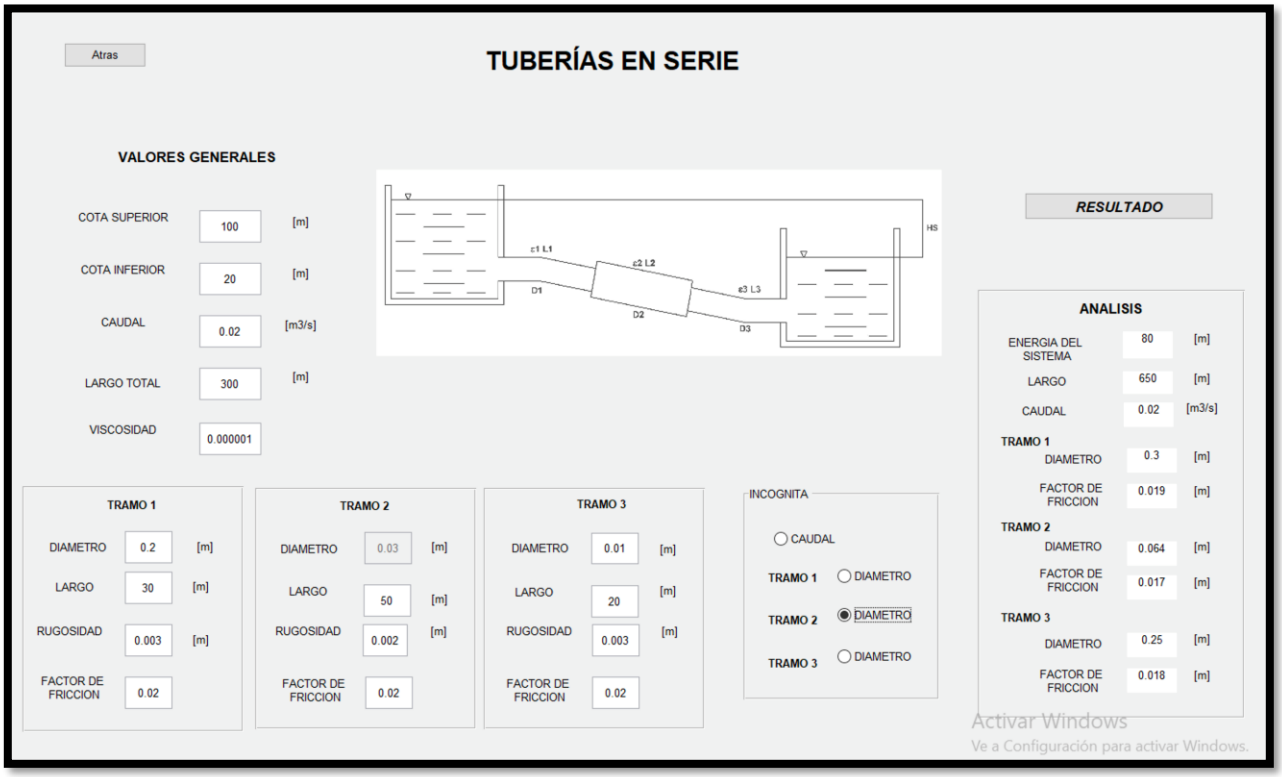
Tubería en serie. (Figura N°9)

Ingreso de parámetros para el diseño geométrico. Se ingresan valores generales, tanto como por tramos.

Los resultados utilizados se expresan en la misma figura

Se podrá analizar el caso de caudal general del sistema, considerar diámetro comercial para cada tramo.

Figura N°9: Tubería en serie.



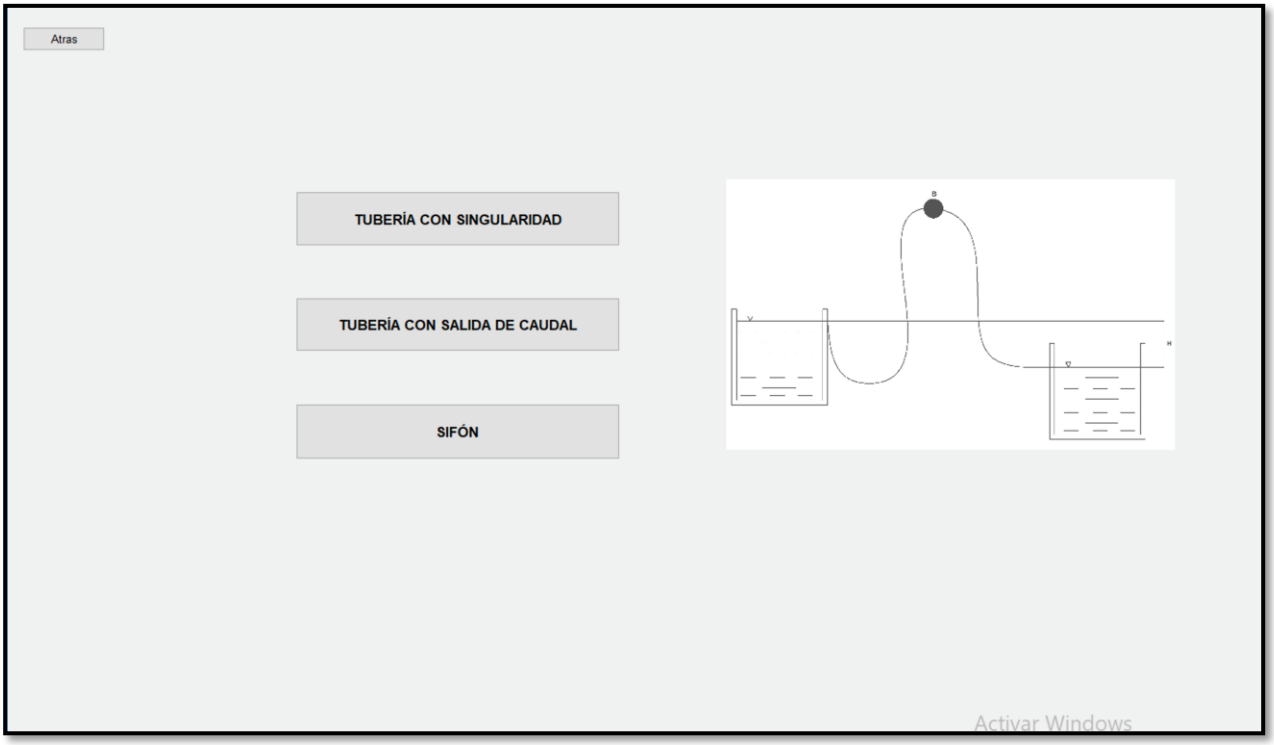
Al analizar los datos obtenidos, las interrogantes son:

¿Qué conllevó a obtener valores obtenidos? Al ver el lado izquierdo de la pantalla podemos visualizar conceptos generales, como energía del sistema, y también ver tramos puntuales del sistema, lo que sucede en cada tramo.

¿Qué se podría analizar del diámetro obtenido? A partir del diámetro obtenido desde un caudal de diseño, este sería el valor mínimo que requiere el sistema para satisfacer al problema. A partir de este resultado, se podrá analizar el valor, aproximar a un estimado de diámetro comercial y estimar que cambio podría resultar disminuir o aumentar el diámetro en esta sección.

Opción n°2

Figura N°10: Líneas de energía



1.-Al ingresar al sistema “Líneas de Flujo” te encontraras con 3 opciones donde podrás a llevar a cabo la imaginación y el diseño.

2.- Considerando la elección que se opta, debes ingresar los valores que se exigen en pantalla. Cuando comiences a completarla, deberás considerar que tenga el mismo sistema de unidades que se expresa en pantalla. En caso contrario debes hacer el cambio de unidad correspondiente para que los valores no se expresen erróneamente frente al resultado del problema (Figura n°3)

3.- Una vez que tengas los valores ingresados, debes saber que lo que vas a graficar, será las líneas de energía del sistema, es por esto, que no tendrás un resultado numérico, si no, que podrás analizar gráficamente los valores que entregaste para el diseño que elegiste. Para poder practicar de mejor forma, es importante que puedas modificar los valores para ver el comportamiento de la gráfica a diferentes valores

4.- Clickkeando en la etiqueta de "Graficar" (Figura N°11) podrás analizar los valores que entrega en cuadricula y a la vez poder incentivar a analizar las gráficas en diferentes aspectos.

Figura N°11: Botón para opción de graficar

5.-Una vez que quieras reiterar este proceso, solamente debes cambiar algún valor inicial y volver al paso 2 para que puedas continuar con el proceso. O también puedes variar en cualquieras de las opciones (Figura N°4) solamente dejando el botón atrás.

Ejercicios

Como complemento al manual del programa, se presenta 1 propuesta para la visualización de ingreso de valores con la finalidad de analizar la resolución del problema.

Tubería con Singularidades (Figura N°12)

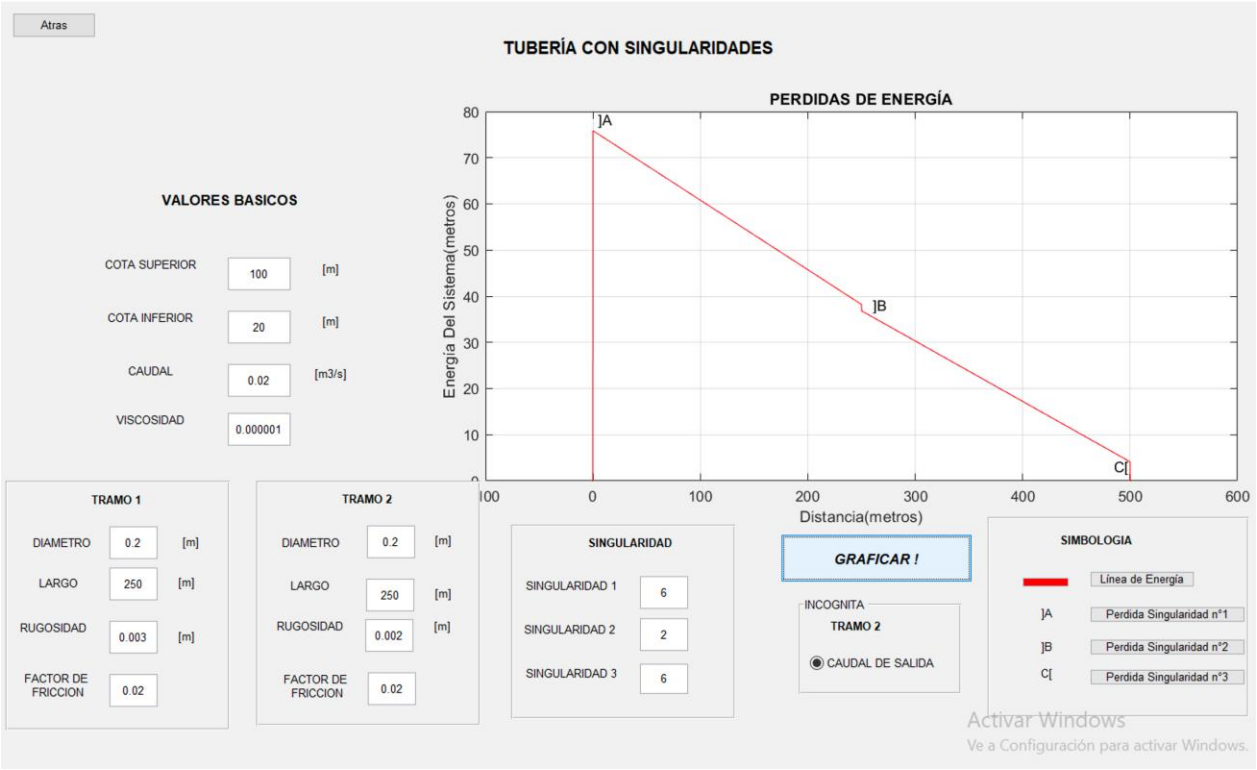


Figura N°12: Tubería con singularidad.

Ingreso de parámetros para el diseño geométrico.

Los resultados utilizados se expresan a través de una grafica

Se analiza las perdidas que tiene los tramos, considerando los saltos de perdida que se origina al tener una singularidad, en este caso, la propuesta entrega 3 singularidades, uno al comienzo, al final y uno a mitad de la tubería.

Además de graficar, ver continuamente los saltos que se produce por las pérdidas, se encuentra en el sector derecho inferior una simbología para complementar al grafico de los datos entregados

Al analizar los datos obtenidos, las interrogantes son:

¿Qué conllevo a obtener valores obtenidos? Se demuestra a través de un gráfico las pérdidas que se originan al tener singularidades en la tubería, en caso de no tener solamente se encontraría una línea continua a lo largo de los tramos. Se visualiza en el lado izquierdo la energía del sistema.

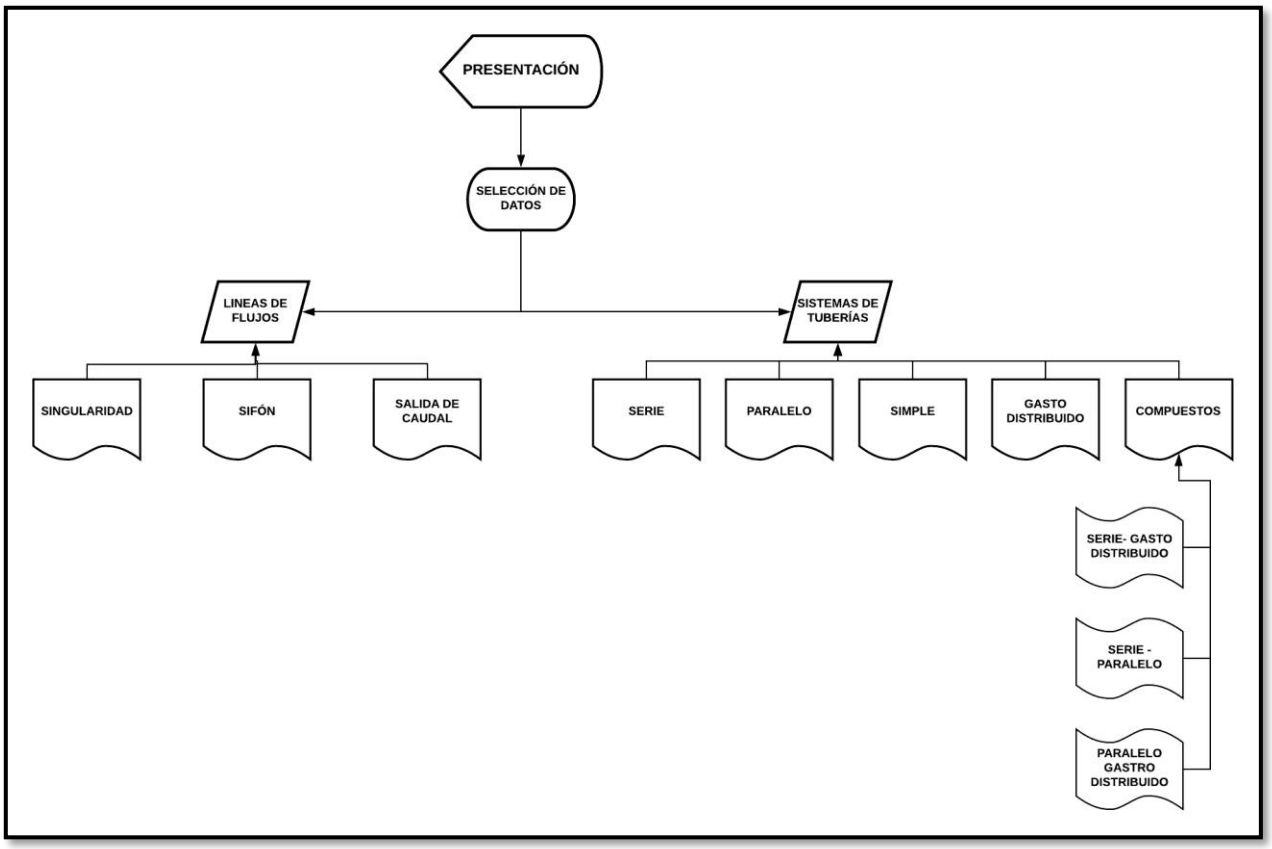
¿Qué aporte tendrá analizar cada singularidad en especial? Al existir múltiples singularidades que podremos analizar, se podrá analizar en detalle el comportamiento que tiene y como afecta en magnitud a la energía que entrega el sistema.

ANEXO

“Código Fuente de Sistemas de Tuberías - Matlab”

CÓDIGO FUENTE de Sistemas de tuberías – Matlab

MAPA DE FORMULARIO Y MODULOS



CÓDIGO FUENTE.

Formulario madre

Portada.

```
function varargout = Portada(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Portada_OpeningFcn, ...
                   'gui_OutputFcn',  @Portada_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

function Portada_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for Portada
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes Portada wait for user response (see UIRESUME)
% uiwait(handles.figure1);
axes(handles.imag1);
background=imread('civil.jpg');
axis off
imshow(background);

axes(handles.imag2);
background=imread('ubb.jpg');
axis off
imshow(background);

axes(handles.axes3);
background=imread('silueta3.jpg');
axis off
imshow(background);
% --- Outputs from this function are returned to the command line.
function varargout = Portada_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

% --- Executes on button press in Indice.
function Indice_Callback(hObject, eventdata, handles)
close(Portada);
Indice
```

Índice

```
function varargout = Indice(varargin)
% INDICE MATLAB code for Indice.fig
%      INDICE, by itself, creates a new INDICE or raises the existing
%      singleton*.
%
%      H = INDICE returns the handle to a new INDICE or the handle to
%      the existing singleton*.
%
%      INDICE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in INDICE.M with the given input arguments.
%
%      INDICE('Property','Value',...) creates a new INDICE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Indice_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Indice_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 Indice

% Last Modified by GUIDE v2.5 07-Feb-2019 17:35:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Indice_OpeningFcn, ...
                   'gui_OutputFcn',  @Indice_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 Indice is made visible.
function Indice_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 Indice (see VARARGIN)

% Choose default command line output for Indice
```

```
handles.output = hObject;
axes(handles.axes3);
background=imread('serie.jpg');
axis off
imshow(background);

axes(handles.axes4);
background=imread('singularidad.jpg');
axis off
imshow(background);
% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = Indice_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 ejercicios.
function ejercicios_Callback(hObject, eventdata, handles)
% hObject    handle to ejercicios (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% esta funcion no se de donde la sacaste

% --- Executes on button press in tuberias.
function tuberias_Callback(hObject, eventdata, handles)
% hObject    handle to tuberias (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%botok=tuberias('style','pushbotton','units','normalized','position',[.84 .83
.12 .05],'string','continuar','callback','clear all','clc','tuberias');
close(Indice);
tuberias

% --- Executes on button press in lineadeenergia.
function lineadeenergia_Callback(hObject, eventdata, handles)
% hObject    handle to lineadeenergia (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%botok=lde('style','pushbotton','units','normalized','position',[.84 .83 .12
.05],'string','continuar','callback','clear all','clc','lde');
close(Indice);
lde
```

```
% --- Executes on key press with focus on tuberias and none of its controls.
function tuberias_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to tuberias (see GCBO)
% eventdata  structure with the following fields (see
MATLAB.UI.CONTROL.UICONTROL)
%   Key: name of the key that was pressed, in lower case
%   Character: character interpretation of the key(s) that was pressed
%   Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)


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

Tuberías

```
function varargout = tuberias(varargin)
% TUBERIAS MATLAB code for tuberias.fig
%      TUBERIAS, by itself, creates a new TUBERIAS or raises the existing
%      singleton*.
%
%      H = TUBERIAS returns the handle to a new TUBERIAS or the handle to
%      the existing singleton*.
%
%      TUBERIAS('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TUBERIAS.M with the given input arguments.
%
%      TUBERIAS('Property','Value',...) creates a new TUBERIAS or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before tuberias_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to tuberias_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 tuberias

% Last Modified by GUIDE v2.5 29-Mar-2019 14:26:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @tuberias_OpeningFcn, ...
                   'gui_OutputFcn',  @tuberias_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 tuberias is made visible.
function tuberias_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 tuberias (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = tuberias_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 tubsimple.
function tubsimple_Callback(hObject, eventdata, handles)
% hObject    handle to tubsimple (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%botok=tubsimple('style','pushbotton','units','normalized','position',[.84 .83
.12 .05],'string','continuar','callback','clear all','clc','tuberias');
close(tuberias);
tubsimplelec

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

```
%botok=tubsimple('style','pushbotton','units','normalized','position',[.84 .83
.12 .05],'string','continuar','callback','clear all','clc','tuberias');
close(tuberias);
tubserie

% --- Executes on button press in tubgas.
function tubgas_Callback(hObject, eventdata, handles)
% hObject    handle to tubgas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%botok=tubgas('style','pushbotton','units','normalized','position',[.84 .83
.12 .05],'string','continuar','callback','clear all','clc','tuberias');
close(tuberias);
tubgas

% --- Executes on button press in tubparalelo.
function tubparalelo_Callback(hObject, eventdata, handles)
% hObject    handle to tubparalelo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%botok=tubparalelo('style','pushbotton','units','normalized','position',[.84
.83 .12 .05],'string','continuar','callback','clear all','clc','tuberias');
close(tuberias);
tubparalelo


% Braulio, te falta un callback de la ultima opcion, ese de ejercicios


% --- Executes on button press in atras.
function atras_Callback(hObject, eventdata, handles)
% hObject    handle to atras (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%botok=Indice('style','pushbotton','units','normalized','position',[.84 .83
.12 .05],'string','continuar','callback','clear all','clc','tuberias');
close(tuberias);
Indice


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

tubsimplelec

```
function varargout = tubsimplelec(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @tubsimplelec_OpeningFcn, ...
                   'gui_OutputFcn',  @tubsimplelec_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 tubsimplelec is made visible.
function tubsimplelec_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 tubsimplelec (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = tubsimplelec_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 hazen.
function hazen_Callback(hObject, eventdata, handles)
% hObject    handle to hazen (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
close(tubsimplelec);
hazen

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

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

factsimplefactor

```
function varargout = tubsimfactor(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @tubsimfactor_OpeningFcn, ...
                   'gui_OutputFcn',  @tubsimfactor_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
% --- Executes just before tubsimfactor is made visible.
function tubsimfactor_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 tubsimfactor (see VARARGIN)

% Choose default command line output for tubsimfactor
handles.output = hObject;
set(handles.v_cau, 'Enable', 'Off');

axes(handles.axes1);
background=imread('simple.jpg');
axis off
imshow(background);
% Update handles structure
```

```
guidata(hObject, handles);
function varargout = tubsimfactor_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;

function v_lar_Callback(hObject, eventdata, handles)
v_lar=str2double(get(hObject,'String'));
if isnan(v_lar)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_lar_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function v_vis_Callback(hObject, eventdata, handles)
% hObject    handle to v_vis (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 v_vis as text
%        str2double(get(hObject,'String')) returns contents of v_vis as a
double
v_vis=str2double(get(hObject,'String'));
global v_vis;
if isnan(v_vis)
    errordlg('El valor debe ser numerico','ERROR');
    %set(hObject.v_vis,'String',0);
end
function v_vis_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_vis (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 v_fdf_Callback(hObject, eventdata, handles)
% hObject    handle to v_fdf (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 v_fdf as text
%        str2double(get(hObject,'String')) returns contents of v_fdf as a
double
v_fdf=str2double(get(hObject,'String'));
global v_fdf;
```

```
if isnan(v_fdf)
    errordlg('El valor debe ser numerico','ERROR');
    %set(hObject.v_fdf,'String',0);
end
function v_fdf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_fdf (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 v_rug_Callback(hObject, eventdata, handles)
% hObject    handle to v_rug (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 v_rug as text
%        str2double(get(hObject,'String')) returns contents of v_rug as a
double
v_rug=str2double(get(hObject,'String'));
if isnan(v_rug)
    errordlg('El valor debe ser numerico','ERROR');
    %set(hObject.v_rug,'String',0);
end
function v_rug_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_rug (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 v_emin_Callback(hObject, eventdata, handles)
% hObject    handle to v_emin (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 v_emin as text
%        str2double(get(hObject,'String')) returns contents of v_emin as a
double
v_emin=str2double(get(hObject,'String'));
if isnan(v_emin)
    errordlg('El valor debe ser numerico','ERROR');
    %set(hObject.v_min,'String',0);
end
function v_emin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emin (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 v_emax_Callback(hObject, eventdata, handles)
% hObject    handle to v_emax (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 v_emax as text
%        str2double(get(hObject,'String')) returns contents of v_emax as a
double
global v_emax;
v_emax=str2double(get(hObject,'String'));
if isnan(v_emax)
    errordlg('El valor debe ser numerico','ERROR');
    %set(hObject.v_emax,'String',0);
end
function v_emax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emax (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 v_cau_Callback(hObject, eventdata, handles)
% hObject    handle to v_cau (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 v_cau as text
%        str2double(get(hObject,'String')) returns contents of v_cau as a
double
v_cau=str2double(get(hObject,'String'));
if isnan(v_cau)
    errordlg('El valor debe ser numerico','ERROR');
    %set(hObject.v_cau,'String',0);
end
if (hObject == handles.caudal)
    set(handles.texvar, 'String', 'CAUDAL');
    set(handles.v_dia, 'Enable', 'On');
    set(handles.v_cau, 'Enable', 'Off');
    set(handles.texunid, 'String', '[m3/s]');

end
function v_cau_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_cau (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 resultado.
function resultado_Callback(hObject, eventdata, handles)

enerMax= get(handles.v_emax,'String');
enerMax=str2double(enerMax);
enerMin= get(handles.v_emin,'String');
enerMin=str2double(enerMin);

hs=enerMax-enerMin;
set(handles.f_hs, 'String', num2str(hs));


if (hObject == handles.caudal)
    caudal= get(handles.cauda,'String');
    caudal=str2double(caudal);
    set(handles.valvar, 'String', num2str(caudal));
end

if (hObject == handles.diametro)
    diametro= get(handles.diame,'String');
    diametro=str2double(diametro);
    set(handles.valvar, 'String', num2str(diametro));

end

function v_dia_Callback(hObject, eventdata, handles)
% hObject    handle to v_dia (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 v_dia as text
%        str2double(get(hObject,'String')) returns contents of v_dia as a
double
global v_dia;
v_dia=str2double(get(hObject,'String'));
if isnan(v_dia)
    errordlg('El valor debe ser numerico','ERROR');
    %set(hObject.v_dia,'String',0);
end
function v_dia_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_dia (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 atras.
function atras_Callback(hObject, eventdata, handles)
close(tubsimfactor);
tubsimplelec


function incognita_SelectionChangedFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in incognita
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%        global v_emax
if (hObject == handles.caudal)
    set(handles.texvar, 'String', 'CAUDAL');
    set(handles.v_dia, 'Enable', 'On');
    set(handles.v_cau, 'Enable', 'Off');
    set(handles.texunid, 'String', '[m3/s]');

end
if (hObject == handles.diametro)
    set(handles.texvar, 'String', 'DIAMETRO');
    set(handles.v_cau, 'Enable', 'On');
    set(handles.v_dia, 'Enable', 'Off');
    set(handles.texunid, 'String', '[m]');
end

v_emax= get(handles.v_emax,'String');
v_emax=str2double(v_emax);

v_emin= get(handles.v_emin,'String');
v_emin=str2double(v_emin);

v_lar= get(handles.v_lar,'String');
v_lar=str2double(v_lar);

v_fdf= get(handles.v_fdf,'String');
v_fdf=str2double(v_fdf);

v_cau= get(handles.v_cau,'String');
v_cau=str2double(v_cau);

v_vis= get(handles.v_vis,'String');
v_vis=str2double(v_vis);

v_rug= get(handles.v_rug,'String');
v_rug=str2double(v_rug);

v_dia= get(handles.v_dia,'String');
v_dia=str2double(v_dia);


if (hObject == handles.caudal)
        %Codigo HAZEN WILLIAMS
        hs=v_emax-v_emin;
        %igualamos nuestra energia del sistema en factor de friccion
        fdd=0;
```

```
        %tenemos f,
        %necesitamos calcular un valor, la cual debe cohincidir con el
        %valor ingresado. En caso de no ser igual, seguimos trabajando con el
        %resultante hasta que estos coincidan.
        n=10;
        for i=1:n
            %CAUDAL
            cauda=((v_dia^(5)*(hs*pi*pi*9.8))/(v_fdf*v_lar*8))^(1/2);
            %Reynolds
            re=((4*cauda)/(pi*v_vis*v_dia));
            %Darcy Williams
            a=( v_rug*(3.7*v_dia)^(-1) )  +(5.1286 * (re^(0.89))^(-1));
            b=(-2)*log10(a);
            %Nuevo Factor de Ficcion.
            fdd=(1*(b^(-1))^(2));
            %Iteracion.
            v_fdf=fdd;
        end
        factor=v_fdf;
        %OBTENEMOS CAUDAL
        %Resultado Numerico
    re=round(re,3);
    factor=round(factor,3);
    cauda=round(cauda,3);

    set(handles.valvar, 'String', num2str(cauda));


end
if (hObject == handles.diametro)
        %DIAMETRO
        hs=v_emax-v_emin;
        %igualamos nuestra energia del sistema en factor de friccion
        fdd=0;
        %tenemos f,
        %necesitamos calcular un valor, la cual debe cohincidir con el
        %valor ingresado. En caso de no ser igual, seguimos trabajando con el
        %resultante hasta que estos coincidan.
        n=10;
        for i=1:n
            %calculamos caudal
            diame=((v_fdf*v_lar*8*(v_cau^(2))/(hs*pi*pi*9.8)))^(1/5);
            %%calculamos re
            re=((4*v_cau)/(pi*v_vis*diame));
            %Darcy Williams
            a=( v_rug*(3.7*diame)^(-1) )  +(5.1286 * (re^(0.89))^(-1));
            b=(-2)*log10(a);
            fdd=((1/b)^(2));
            v_fdf=fdd;
        end
        factor=v_fdf;
        %OBTENEMOS CAUDAL, REINOLDS Y FACTOR DE FRICCION
        %Resultado Numerico
        diame=round(diame,3);
        set(handles.valvar, 'String', num2str(diame));

        %Resultado Unidad
        %set(handles.f_rey, 'String', num2str(re));
end
```

```
re=round(re,3);
factor=round(factor,3);
set(handles.f_rey, 'String', num2str(re));
set(handles.f_fdd, 'String', num2str(factor));
```

hazen

```
function varargout = hazen(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @hazen_OpeningFcn, ...
                   'gui_OutputFcn',  @hazen_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

function hazen_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
handles.output = hObject;
set(handles.v_cau, 'Enable', 'Off');

axes(handles.axes1);
background=imread('simple.jpg');
axis off
imshow(background);
% Update handles structure
guidata(hObject, handles);
function varargout = hazen_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;

function v_lar_Callback(hObject, eventdata, handles)
v_lar=str2double(get(hObject,'String'));
if isnan(v_lar)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_lar_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_lar (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 v_ch_Callback(hObject, eventdata, handles)
v_ch=str2double(get(hObject,'String'));
if isnan(v_ch)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_ch_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_ch (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 v_emin_Callback(hObject, eventdata, handles)
v_emin=str2double(get(hObject,'String'));
if isnan(v_emin)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_emin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emin (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 v_emax_Callback(hObject, eventdata, handles)
v_emax=str2double(get(hObject,'String'));
if isnan(v_emax)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_emax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emax (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 v_cau_Callback(hObject, eventdata, handles)
v_cau=str2double(get(hObject,'String'));
```

```
if isnan(v_cau)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_cau_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_cau (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 resultado.
function resultado_Callback(hObject, eventdata, handles)
enerMax= get(handles.v_emax,'String');
enerMax=str2double(enerMax);
enerMin= get(handles.v_emin,'String');
enerMin=str2double(enerMin);

hs=enerMax-enerMin;
set(handles.f_hs, 'String', num2str(hs));

function v_dia_Callback(hObject, eventdata, handles)
v_dia=str2double(get(hObject,'String'));
if isnan(v_dia)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_dia_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_dia (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 atras.
function atras_Callback(hObject, eventdata, handles)
close(hazen)
tubsimplelec

% --- Executes when selected object is changed in incognita.
function incognita_SelectionChangedFcn(hObject, eventdata, handles)
if (hObject == handles.caudal)
    set(handles.texvar, 'String', 'CAUDAL');
    set(handles.v_dia, 'Enable', 'On');
    set(handles.v_cau, 'Enable', 'Off');
    set(handles.texunid, 'String', '[m3/s]');

end
if (hObject == handles.diametro)
    set(handles.texvar, 'String', 'DIAMETRO');
```

```
    set(handles.v_cau, 'Enable', 'On');
    set(handles.v_dia, 'Enable', 'Off');
    set(handles.texunid, 'String', '[m]');
end

v_emax= get(handles.v_emax,'String');
v_emax=str2double(v_emax);

v_emin= get(handles.v_emin,'String');
v_emin=str2double(v_emin);

v_lar= get(handles.v_lar,'String');
v_lar=str2double(v_lar);

v_ch= get(handles.v_ch,'String');
v_ch=str2double(v_ch);

v_cau= get(handles.v_cau,'String');
v_cau=str2double(v_cau);

v_dia= get(handles.v_dia,'String');
v_dia=str2double(v_dia);

if (hObject == handles.caudal)
    hs=v_emax-v_emin;
    %PERDIDA POR FACTOR DE HASEN WILLIAMS
    v_cau=(hs*v_ch.^(1.852)*v_dia.^(4.87)/(10.67*v_lar)).^(1/1.852);
    v_cau=round(v_cau,3);
    set(handles.valvar, 'String', num2str(v_cau));
end
if (hObject == handles.diametro)
    hs=v_emax-v_emin;
    %PERDIDA POR FACTOR DE HASEN WILLIAMS
    v_dia=(10.67*v_lar*v_cau.^(1.852)/(v_ch.^(1.852)*hs)).^(1/4.87);
    v_dia=round(v_dia,3);
    set(handles.valvar, 'String', num2str(v_dia));
end
```

tubserie

```
function varargout = tubserie(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @tubserie_OpeningFcn, ...
                   'gui_OutputFcn',  @tubserie_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

function tubserie_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 tubserie (see VARARGIN)
axes(handles.axes1);
background=imread('serie.jpg');
axis off
imshow(background);
% Choose default command line output for tubserie
handles.output = hObject;
handles.output = hObject;
set(handles.v_cau, 'Enable', 'Off');
% Update handles structure
guidata(hObject, handles);

function varargout = tubserie_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;

function v_emax_Callback(hObject, eventdata, handles)
v_emax=str2double(get(hObject,'String'));
if isnan(v_emax)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_emax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emax (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 resultado.
function resultado_Callback(hObject, eventdata, handles)
enerMax= get(handles.v_emax,'String');
enerMax=str2double(enerMax);
enerMin= get(handles.v_emin,'String');
enerMin=str2double(enerMin);

hs=enerMax-enerMin;
set(handles.f_hs, 'String', num2str(hs));



function atras_Callback(hObject, eventdata, handles)
close(tubserie)
tuberias
function v_emin_Callback(hObject, eventdata, handles)
v_emin=str2double(get(hObject,'String'));
if isnan(v_emin)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_emin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emin (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 v_cau_Callback(hObject, eventdata, handles)
v_cau=str2double(get(hObject,'String'));
if isnan(v_cau)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_cau_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_cau (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 v_lar_Callback(hObject, eventdata, handles)
v_lar=str2double(get(hObject,'String'));
if isnan(v_lar)
```

```
    errordlg('El valor debe ser numerico','ERROR');
end
function v_lar_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_lar (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 v_vis_Callback(hObject, eventdata, handles)
v_vis=str2double(get(hObject,'String'));
if isnan(v_vis)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_vis_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_vis (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 t3_fdf_Callback(hObject, eventdata, handles)
t3_fdf=str2double(get(hObject,'String'));
if isnan(t3_fdf)
    errordlg('El valor debe ser numerico','ERROR');
end
function t3_fdf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t3_fdf (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 t3_lar_Callback(hObject, eventdata, handles)
t3_lar=str2double(get(hObject,'String'));
if isnan(t3_lar)
    errordlg('El valor debe ser numerico','ERROR');
end
function t3_lar_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t3_lar (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 t3_rug_Callback(hObject, eventdata, handles)
t3_rug=str2double(get(hObject,'String'));
if isnan(t3_rug)
    errordlg('El valor debe ser numerico','ERROR');
end
function t3_rug_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t3_rug (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 t3_dia_Callback(hObject, eventdata, handles)
t3_dia=str2double(get(hObject,'String'));
if isnan(t3_dia)
    errordlg('El valor debe ser numerico','ERROR');
end
function t3_dia_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t3_dia (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 t2_fdf_Callback(hObject, eventdata, handles)
t2_fdf=str2double(get(hObject,'String'));
if isnan(t2_fdf)
    errordlg('El valor debe ser numerico','ERROR');
end
function t2_fdf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t2_fdf (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 t2_lar_Callback(hObject, eventdata, handles)
t2_lar=str2double(get(hObject,'String'));
if isnan(t2_lar)
    errordlg('El valor debe ser numerico','ERROR');
end
function t2_lar_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t2_lar (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 t2_rug_Callback(hObject, eventdata, handles)
t2_rug=str2double(get(hObject,'String'));
if isnan(t2_rug)
    errordlg('El valor debe ser numerico','ERROR');
end
function t2_rug_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t2_rug (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 t2_dia_Callback(hObject, eventdata, handles)
t2_dia=str2double(get(hObject,'String'));
if isnan(t2_dia)
    errordlg('El valor debe ser numerico','ERROR');
end
function t2_dia_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t2_dia (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 t1_rug_Callback(hObject, eventdata, handles)
t1_rug=str2double(get(hObject,'String'));
if isnan(t1_rug)
    errordlg('El valor debe ser numerico','ERROR');
end
function t1_rug_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t1_rug (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 t1_fdf_Callback(hObject, eventdata, handles)
t1_fdf=str2double(get(hObject,'String'));
if isnan(t1_fdf)
    errordlg('El valor debe ser numerico','ERROR');
end
function t1_fdf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t1_fdf (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 t1_lar_Callback(hObject, eventdata, handles)
t1_lar=str2double(get(hObject,'String'));
if isnan(t1_lar)
    errordlg('El valor debe ser numerico','ERROR');
end
function t1_lar_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t1_lar (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 t1_dia_Callback(hObject, eventdata, handles)
t1_dia=str2double(get(hObject,'String'));
if isnan(t1_dia)
    errordlg('El valor debe ser numerico','ERROR');
end
function t1_dia_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t1_dia (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 when selected object is changed in incognita.
function incognita_SelectionChangedFcn(hObject, eventdata, handles)
v_emax= get(handles.v_emax,'String');
v_emax=str2double(v_emax);

v_emin= get(handles.v_emin,'String');
v_emin=str2double(v_emin);

v_lar= get(handles.v_lar,'String');
v_lar=str2double(v_lar);


v_cau= get(handles.v_cau,'String');
v_cau=str2double(v_cau);

v_vis= get(handles.v_vis,'String');
v_vis=str2double(v_vis);

t1_fdf= get(handles.t1_fdf,'String');
t1_fdf=str2double(t1_fdf);

t1_rug= get(handles.t1_rug,'String');
t1_rug=str2double(t1_rug);

t1_dia= get(handles.t1_dia,'String');
t1_dia=str2double(t1_dia);

t1_lar= get(handles.t1_lar,'String');
t1_lar=str2double(t1_lar);

t2_fdf= get(handles.t2_fdf,'String');
t2_fdf=str2double(t2_fdf);

t2_rug= get(handles.t2_rug,'String');
t2_rug=str2double(t2_rug);

t2_dia= get(handles.t2_dia,'String');
t2_dia=str2double(t2_dia);

t2_lar= get(handles.t2_lar,'String');
t2_lar=str2double(t2_lar);

t3_fdf= get(handles.t3_fdf,'String');
t3_fdf=str2double(t3_fdf);

t3_rug= get(handles.t3_rug,'String');
t3_rug=str2double(t3_rug);

t3_dia= get(handles.t3_dia,'String');
t3_dia=str2double(t3_dia);

t3_lar= get(handles.t3_lar,'String');
```

```
t3_lar=str2double(t3_lar);

hs=10;

v_lar=650;

v_vis= 0.000001;

if (hObject == handles.caud)
    set(handles.t3_dia, 'Enable', 'On');
    set(handles.t2_dia, 'Enable', 'On');
    set(handles.t1_dia, 'Enable', 'On');
    set(handles.v_cau, 'Enable', 'Off');

    hs=v_emax-v_emin;
    n=10;
    for i=1:n
    cau=((hs*pi*pi*9.8/8)/(t1_fdf*t1_lar/t1_dia^5 + t2_fdf*t2_lar/t2_dia^5
+t3_fdf*t3_lar/t3_dia^5))^(1/2);
    %Tramo1
    re1=((4*cau)/(pi*v_vis*t1_dia));
    a1=( t1_rug*(t1_dia)^(-1) )  +(21.25 * (re1^(-0.9)));
    b1=1.14-(2)*log10(a1);
    fdd1=((1*(b1^(-1)))^(2));
    fdd1=round(fdd1,3);
    t1_fdf=fdd1;

    %Tramo2
    re2=((4*cau)/(pi*v_vis*t2_dia));
    a2=( t2_rug*(t2_dia)^(-1) )  +(21.25 * (re2^(-0.9)));
    b2=1.14-(2)*log10(a2);
    fdd2=((1*(b2^(-1)))^(2));
    fdd2=round(fdd2,3);
    t2_fdf=fdd2;

    %Tramo3
    re3=((4*cau)/(pi*v_vis*t3_dia));
    a3=( t3_rug*(t3_dia)^(-1) )  +(21.25 * (re3^(-0.9)));
    b3=1.14-(2)*log10(a3);
    fdd3=((1*(b3^(-1)))^(2));
    fdd3=round(fdd3,3);
    t3_fdf=fdd3;
    end

set(handles.ft1_dia, 'String', num2str(t1_dia));
set(handles.ft1_fdf, 'String', num2str(t1_fdf));

set(handles.ft2_dia, 'String', num2str(t2_dia));
set(handles.ft2_fdf, 'String', num2str(t2_fdf));

set(handles.ft3_dia, 'String', num2str(t3_dia));
set(handles.ft3_fdf, 'String', num2str(t3_fdf));
cau=round(cau,3);
set(handles.f_cau, 'String', num2str(cau));
v_lar=round(v_lar,3);
set(handles.f_lar, 'String', num2str(v_lar));
```

```
end

if (hObject == handles.diat1)
    set(handles.t3_dia, 'Enable', 'On');
    set(handles.t2_dia, 'Enable', 'On');
    set(handles.t1_dia, 'Enable', 'Off');
    set(handles.v_cau, 'Enable', 'On');
    hs=v_emax-v_emin;
    n=10;
    cau=v_cau;
    for i=1:n
    t1_dia=((8*v_cau.^2*t1_fdf*t1_lar/(pi*pi*9.8))*(hs-
(8*v_cau.^2*t2_fdf*t2_lar/(pi*pi*9.8*t2_dia.^5)-
(8*v_cau.^2*t3_fdf*t3_lar/(pi*pi*9.8*t3_dia.^5)))).^(-1)).^(1/5);
    %Tramo1
    re1=((4*cau)/(pi*v_vis*t1_dia));
    a1=( t1_rug*(t1_dia)^(-1) )  +(21.25 * (re1^(-0.9)));
    b1=1.14-(2)*log10(a1);
    fdd1=((1*(b1^(-1)))^(2));
    fdd1=round(fdd1,3);
    t1_fdf=fdd1;
        % ramo2
    re2=((4*cau)/(pi*v_vis*t2_dia));
    a2=( t2_rug*(t2_dia)^(-1) )  +(21.25 * (re2^(-0.9)));
    b2=1.14-(2)*log10(a2);
    fdd2=((1*(b2^(-1)))^(2));
    fdd2=round(fdd2,3);
    t2_fdf=fdd2;
    %Tramo3
    re3=((4*cau)/(pi*v_vis*t3_dia));
    a3=( t3_rug*(t3_dia)^(-1) )  +(21.25 * (re3^(-0.9)));
    b3=1.14-(2)*log10(a3);
    fdd3=((1*(b3^(-1)))^(2));
    fdd3=round(fdd3,3);
    t3_fdf=fdd3;
    end
t1_dia=round(t1_dia,3);
set(handles.ft1_dia, 'String', num2str(t1_dia));
t1_fdf=round(t1_fdf,3);
set(handles.ft1_fdf, 'String', num2str(t1_fdf));

set(handles.ft2_dia, 'String', num2str(t2_dia));
set(handles.ft2_fdf, 'String', num2str(t2_fdf));

set(handles.ft3_dia, 'String', num2str(t3_dia));
set(handles.ft3_fdf, 'String', num2str(t3_fdf));

set(handles.f_cau, 'String', num2str(cau));
set(handles.f_lar, 'String', num2str(v_lar));
end

if (hObject == handles.diat2)
    set(handles.t3_dia, 'Enable', 'On');
    set(handles.t2_dia, 'Enable', 'Off');
    set(handles.t1_dia, 'Enable', 'On');
    set(handles.v_cau, 'Enable', 'On');
    hs=v_emax-v_emin;
    n=10;
```

```
    cau=v_cau;
    for i=1:n
    t2_dia=((8*v_cau.^2*t2_fdf*t2_lar/(pi*pi*9.8))*(hs-
(8*v_cau.^2*t1_fdf*t1_lar/(pi*pi*9.8*t1_dia.^5)-
(8*v_cau.^2*t3_fdf*t3_lar/(pi*pi*9.8*t3_dia.^5)))).^(-1)).^(1/5);
    %Tramo1
    re1=((4*cau)/(pi*v_vis*t1_dia));
    a1=( t1_rug*(t1_dia)^(-1) )  +(21.25 * (re1^(-0.9)));
    b1=1.14-(2)*log10(a1);
    fdd1=((1*(b1^(-1)))^(2));
    fdd1=round(fdd1,3);
    t1_fdf=fdd1;
    %Tramo2
    re2=((4*cau)/(pi*v_vis*t2_dia));
    a2=( t2_rug*(t2_dia)^(-1) )  +(21.25 * (re2^(-0.9)));
    b2=1.14-(2)*log10(a2);
    fdd2=((1*(b2^(-1)))^(2));
    fdd2=round(fdd2,3);
    t2_fdf=fdd2;
    %Tramo3
    re3=((4*cau)/(pi*v_vis*t3_dia));
    a3=( t3_rug*(t3_dia)^(-1) )  +(21.25 * (re3^(-0.9)));
    b3=1.14-(2)*log10(a3);
    fdd3=((1*(b3^(-1)))^(2));
    fdd3=round(fdd3,3);
    t3_fdf=fdd3;
    end
set(handles.ft1_dia, 'String', num2str(t1_dia));
set(handles.ft1_fdf, 'String', num2str(t1_fdf));

t2_dia=round(t2_dia,3);
t2_fdf=round(t2_fdf,3);
set(handles.ft2_dia, 'String', num2str(t2_dia));
set(handles.ft2_fdf, 'String', num2str(t2_fdf));

set(handles.ft3_dia, 'String', num2str(t3_dia));
set(handles.ft3_fdf, 'String', num2str(t3_fdf));

set(handles.f_cau, 'String', num2str(cau));
set(handles.f_lar, 'String', num2str(v_lar));
end

if (hObject == handles.diat3)
    set(handles.t3_dia, 'Enable', 'Off');
    set(handles.t2_dia, 'Enable', 'On');
    set(handles.t1_dia, 'Enable', 'On');
    set(handles.v_cau, 'Enable', 'On');
    hs=v_emax-v_emin;
    n=10;
    cau=v_cau;
    for i=1:n
    t3_dia=((8*v_cau.^2*t3_fdf*t3_lar/(pi*pi*9.8))*(hs-
(8*v_cau.^2*t1_fdf*t1_lar/(pi*pi*9.8*t1_dia.^5)-
(8*v_cau.^2*t2_fdf*t2_lar/(pi*pi*9.8*t2_dia.^5)))).^(-1)).^(1/5);
    %Tramo1
    re1=((4*cau)/(pi*v_vis*t1_dia));
    a1=( t1_rug*(t1_dia)^(-1) )  +(21.25 * (re1^(-0.9)));
    b1=1.14-(2)*log10(a1);
    fdd1=((1*(b1^(-1)))^(2));
```

```
    fdd1=round(fdd1,3);
    t1_fdf=fdd1;
    %Tramo2
    re2=((4*cau)/(pi*v_vis*t2_dia));
    a2=( t2_rug*(t2_dia)^(-1) )  +(21.25 * (re2^(-0.9)));
    b2=1.14-(2)*log10(a2);
    fdd2=((1*(b2^(-1)))^(2));
    fdd2=round(fdd2,3);
    t2_fdf=fdd2;
    %Tramo3
    re3=((4*cau)/(pi*v_vis*t3_dia));
    a3=( t3_rug*(t3_dia)^(-1) )  +(21.25 * (re3^(-0.9)));
    b3=1.14-(2)*log10(a3);
    fdd3=((1*(b3^(-1)))^(2));
    fdd3=round(fdd3,3);
    t3_fdf=fdd3;
    end
set(handles.ft1_dia, 'String', num2str(t1_dia));
set(handles.ft1_fdf, 'String', num2str(t1_fdf));

set(handles.ft2_dia, 'String', num2str(t2_dia));
set(handles.ft2_fdf, 'String', num2str(t2_fdf));

t3_dia=round(t3_dia,3);
t3_fdf=round(t3_fdf,3);
set(handles.ft3_dia, 'String', num2str(t3_dia));
set(handles.ft3_fdf, 'String', num2str(t3_fdf));

set(handles.f_cau, 'String', num2str(cau));
set(handles.f_lar, 'String', num2str(v_lar));
end
```

tubparalelo

```
function varargout = tubparalelo(varargin)
% TUBPARALELO MATLAB code for tubparalelo.fig
%      TUBPARALELO, by itself, creates a new TUBPARALELO or raises the
existing
%      singleton*.
%
%      H = TUBPARALELO returns the handle to a new TUBPARALELO or the handle
to
%      the existing singleton*.
%
%      TUBPARALELO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TUBPARALELO.M with the given input
arguments.
%
%      TUBPARALELO('Property','Value',...) creates a new TUBPARALELO or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before tubparalelo_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to tubparalelo_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 tubparalelo

% Last Modified by GUIDE v2.5 21-Mar-2019 15:27:12

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @tubparalelo_OpeningFcn, ...
                   'gui_OutputFcn',  @tubparalelo_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

function tubparalelo_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 tubparalelo (see VARARGIN)
handles.output = hObject;
set(handles.v_cau3, 'Enable', 'Off');

axes(handles.axes1);
background=imread('paralelo.jpg');
axis off
imshow(background);
% Choose default command line output for tubparalelo
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
function varargout = tubparalelo_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;

function v_emax_Callback(hObject, eventdata, handles)

v_emax=str2double(get(hObject,'String'));
if isnan(v_emax)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_emax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emax (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 resultado.
function resultado_Callback(hObject, eventdata, handles)

v_lar= get(handles.v_lar,'String');
v_lar=str2double(v_lar);

v_vis= get(handles.v_vis,'String');
v_vis=str2double(v_vis);

v_emax= get(handles.v_emax,'String');
v_emax=str2double(v_emax);
v_emin= get(handles.v_emin,'String');
v_emin=str2double(v_emin);

hs=v_emax-v_emin;
set(handles.f_hs, 'String', num2str(hs));
```

```
Largo= get(handles.v_lar,'String');
Largo=str2double(Largo);
set(handles.f_lar, 'String', num2str(Largo));

% --- Executes on button press in atras.
function atras_Callback(hObject, eventdata, handles)
close(tubparalelo)
tuberias

function v_emin_Callback(hObject, eventdata, handles)
v_emin=str2double(get(hObject,'String'));
if isnan(v_emin)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_emin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emin (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 v_cau_Callback(hObject, eventdata, handles)
v_cau=str2double(get(hObject,'String'));
if isnan(v_cau)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_cau_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_cau (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 v_lar_Callback(hObject, eventdata, handles)
v_lar=str2double(get(hObject,'String'));
if isnan(v_lar)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_lar_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_lar (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 v_vis_Callback(hObject, eventdata, handles)
v_vis=str2double(get(hObject,'String'));
if isnan(v_vis)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_vis_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_vis (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 v_fdf3_Callback(hObject, eventdata, handles)
v_fdf3=str2double(get(hObject,'String'));
if isnan(v_fdf3)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_fdf3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_fdf3 (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 v_dia3_Callback(hObject, eventdata, handles)
v_dia3=str2double(get(hObject,'String'));
if isnan(v_dia3)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_dia3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_dia3 (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 v_cau3_Callback(hObject, eventdata, handles)
v_cau3=str2double(get(hObject,'String'));
if isnan(v_cau3)
    errordlg('El valor debe ser numerico','ERROR');
end
```

```
function v_cau3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_cau3 (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 v_rug3_Callback(hObject, eventdata, handles)
v_rug3=str2double(get(hObject,'String'));
if isnan(v_rug3)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_rug3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_rug3 (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 v_fdf2_Callback(hObject, eventdata, handles)
v_fdf2=str2double(get(hObject,'String'));
if isnan(v_fdf2)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_fdf2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_fdf2 (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 v_dia2_Callback(hObject, eventdata, handles)
v_dia2=str2double(get(hObject,'String'));
if isnan(v_dia2)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_dia2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_dia2 (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 v_cau2_Callback(hObject, eventdata, handles)
v_cau2=str2double(get(hObject,'String'));
if isnan(v_cau2)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_cau2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_cau2 (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 v_rug2_Callback(hObject, eventdata, handles)
v_rug2=str2double(get(hObject,'String'));
if isnan(v_rug2)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_rug2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_rug2 (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 v_cau1_Callback(hObject, eventdata, handles)
v_cau1=str2double(get(hObject,'String'));
if isnan(v_cau1)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_cau1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_cau1 (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 v_rug1_Callback(hObject, eventdata, handles)
v_rug1=str2double(get(hObject,'String'));
```

```
if isnan(v_rug1)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_rug1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_rug1 (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 v_fdf1_Callback(hObject, eventdata, handles)
v_fdf1=str2double(get(hObject,'String'));
if isnan(v_fdf1)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_fdf1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_fdf1 (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 v_dia1_Callback(hObject, eventdata, handles)
v_dia1=str2double(get(hObject,'String'));
if isnan(v_dia1)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_dia1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_dia1 (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 when selected object is changed in alternativas.
function alternativas_SelectionChangedFcn(hObject, eventdata, handles)
v_lar= get(handles.v_lar,'String');
v_lar=str2double(v_lar);
v_vis= get(handles.v_vis,'String');
v_vis=str2double(v_vis);
v_emax= get(handles.v_emax,'String');
v_emax=str2double(v_emax);
```

```
v_emin= get(handles.v_emin,'String');
v_emin=str2double(v_emin);

v_cau1= get(handles.v_cau1,'String');
v_cau1=str2double(v_cau1);
v_cau2= get(handles.v_cau2,'String');
v_cau2=str2double(v_cau2);
v_cau3= get(handles.v_cau3,'String');
v_cau3=str2double(v_cau3);
v_dia1= get(handles.v_dia1,'String');
v_dia1=str2double(v_dia1);
v_dia2= get(handles.v_dia2,'String');
v_dia2=str2double(v_dia2);
v_dia3= get(handles.v_dia3,'String');
v_dia3=str2double(v_dia3);
v_rug1= get(handles.v_rug1,'String');
v_rug1=str2double(v_rug1);
v_rug2= get(handles.v_rug2,'String');
v_rug2=str2double(v_rug2);
v_rug3= get(handles.v_rug3,'String');
v_rug3=str2double(v_rug3);
v_fdf1= get(handles.v_fdf1,'String');
v_fdf1=str2double(v_fdf1);
v_fdf2= get(handles.v_fdf2,'String');
v_fdf2=str2double(v_fdf2);
v_fdf3= get(handles.v_fdf3,'String');
v_fdf3=str2double(v_fdf3);

if (hObject == handles.cau1)
    set(handles.v_dia1, 'Enable', 'On');
    set(handles.v_cau1, 'Enable', 'Off');
    set(handles.v_dia2, 'Enable', 'On');
    set(handles.v_cau2, 'Enable', 'On');
    set(handles.v_dia3, 'Enable', 'On');
    set(handles.v_cau3, 'Enable', 'On');
    hs=v_emax-v_emin;
    fdd=0;
    n=10;
for i=1:n
    cauda=((v_dia1^(5)*(hs*pi*pi*9.8))/(v_fdf1*v_lar*8))^(1/2);
    re=((4*cauda)/(pi*v_vis*v_dia1));
    a=( v_rug1*(3.7*v_dia1)^(-1) )  +(5.1286 * (re^(0.89))^(-1));
    b=(-2)*log10(a);
    fdd=((1*(b^(-1)))^(2));
    v_fdf=fdd;
end
    factor=v_fdf;
    set(handles.f_cau1, 'String', num2str(cauda));
    set(handles.f_dia1, 'String', num2str(v_dia1));
    set(handles.f_fdf1, 'String', num2str(factor));

    %Restantes
    set(handles.f_cau2, 'String', num2str(v_cau2));
    set(handles.f_dia2, 'String', num2str(v_dia2));
    set(handles.f_fdf2, 'String', num2str(v_fdf2));

    set(handles.f_cau3, 'String', num2str(v_cau3));
    set(handles.f_dia3, 'String', num2str(v_dia3));
    set(handles.f_fdf3, 'String', num2str(v_fdf3));
```

```
end

if (hObject == handles.cau2)
    set(handles.v_dia1, 'Enable', 'On');
    set(handles.v_cau1, 'Enable', 'On');
    set(handles.v_dia2, 'Enable', 'On');
    set(handles.v_cau2, 'Enable', 'Off');
    set(handles.v_dia3, 'Enable', 'On');
    set(handles.v_cau3, 'Enable', 'On');
    hs=v_emax-v_emin;
    fdd=0;
    n=10;
for i=1:n
    cauda=((v_dia2^(5)*(hs*pi*pi*9.8))/(v_fdf2*v_lar*8))^(1/2);
    re=((4*cauda)/(pi*v_vis*v_dia2));
    a=( v_rug2*(3.7*v_dia2)^(-1) )  +(5.1286 * (re^(0.89))^(-1));
    b=(-2)*log10(a);
    fdd=((1*(b^(-1)))^(2));
    v_fdf=fdd;
end
    factor=v_fdf;
    set(handles.f_cau2, 'String', num2str(cauda));
    set(handles.f_dia2, 'String', num2str(v_dia2));
    set(handles.f_fdf2, 'String', num2str(factor));

    %Restantes
    set(handles.f_cau1, 'String', num2str(v_cau1));
    set(handles.f_dia1, 'String', num2str(v_dia1));
    set(handles.f_fdf1, 'String', num2str(v_fdf1));

    set(handles.f_cau3, 'String', num2str(v_cau3));
    set(handles.f_dia3, 'String', num2str(v_dia3));
    set(handles.f_fdf3, 'String', num2str(v_fdf3));
end

if (hObject == handles.cau3)
    set(handles.v_dia1, 'Enable', 'On');
    set(handles.v_cau1, 'Enable', 'On');
    set(handles.v_dia2, 'Enable', 'On');
    set(handles.v_cau2, 'Enable', 'On');
    set(handles.v_dia3, 'Enable', 'On');
    set(handles.v_cau3, 'Enable', 'Off');
   %Codigo HAZEN WILLIAMS
    hs=v_emax-v_emin;
    fdd=0;
    n=10;
for i=1:n
    cauda=((v_dia3^(5)*(hs*pi*pi*9.8))/(v_fdf3*v_lar*8))^(1/2);
    re=((4*cauda)/(pi*v_vis*v_dia3));
    a=( v_rug3*(3.7*v_dia3)^(-1) )  +(5.1286 * (re^(0.89))^(-1));
    b=(-2)*log10(a);
    fdd=((1*(b^(-1)))^(2));
    v_fdf=fdd;
end
    factor=v_fdf;
    set(handles.f_cau3, 'String', num2str(cauda));
    set(handles.f_dia3, 'String', num2str(v_dia3));
    set(handles.f_fdf3, 'String', num2str(factor));
```

```
    %Restantes
    set(handles.f_cau2, 'String', num2str(v_cau2));
    set(handles.f_dia2, 'String', num2str(v_dia2));
    set(handles.f_fdf2, 'String', num2str(v_fdf2));

    set(handles.f_cau1, 'String', num2str(v_cau1));
    set(handles.f_dia1, 'String', num2str(v_dia1));
    set(handles.f_fdf1, 'String', num2str(v_fdf1));
end

if (hObject == handles.dia1)
    set(handles.v_dia1, 'Enable', 'Off');
    set(handles.v_cau1, 'Enable', 'On');
    set(handles.v_dia2, 'Enable', 'On');
    set(handles.v_cau2, 'Enable', 'On');
    set(handles.v_dia3, 'Enable', 'On');
    set(handles.v_cau3, 'Enable', 'On');
    hs=v_emax-v_emin;
    fdd=0;
    n=10;
for i=1:n
    diame=((v_fdf1*v_lar*8*(v_cau1^(2))/(hs*pi*pi*9.8)))^(1/5);
    re=((4*v_cau1)/(pi*v_vis*diame));
    %Darcy Williams
    a=( v_rug1*(3.7*diame)^(-1) )  +(5.1286 * (re^(0.89))^(-1));
    b=(-2)*log10(a);
    fdd=((1/b)^(2));
    v_fdf=fdd;
end
factor=v_fdf;
    set(handles.f_cau1, 'String', num2str(v_cau1));
    set(handles.f_dia1, 'String', num2str(diame));
    set(handles.f_fdf1, 'String', num2str(factor));

    %Restantes
    set(handles.f_cau2, 'String', num2str(v_cau2));
    set(handles.f_dia2, 'String', num2str(v_dia2));
    set(handles.f_fdf2, 'String', num2str(v_fdf2));

    set(handles.f_cau3, 'String', num2str(v_cau3));
    set(handles.f_dia3, 'String', num2str(v_dia3));
    set(handles.f_fdf3, 'String', num2str(v_fdf3));
end

if (hObject == handles.dia2)
    set(handles.v_dia1, 'Enable', 'On');
    set(handles.v_cau1, 'Enable', 'On');
    set(handles.v_dia2, 'Enable', 'Off');
    set(handles.v_cau2, 'Enable', 'On');
    set(handles.v_dia3, 'Enable', 'On');
    set(handles.v_cau3, 'Enable', 'On');
    hs=v_emax-v_emin;
    fdd=0;
    n=10;
for i=1:n
    diame=((v_fdf2*v_lar*8*(v_cau2^(2))/(hs*pi*pi*9.8)))^(1/5);
    re=((4*v_cau2)/(pi*v_vis*diame));
```

```
    %Darcy Williams
    a=( v_rug2*(3.7*diame)^(-1) )  +(5.1286 * (re^(0.89))^(-1));
    b=(-2)*log10(a);
    fdd=((1/b)^(2));
    v_fdf=fdd;
end
factor=v_fdf;
    set(handles.f_cau2, 'String', num2str(v_cau2));
    set(handles.f_dia2, 'String', num2str(diame));
    set(handles.f_fdf2, 'String', num2str(factor));

    %Restantes
    set(handles.f_cau1, 'String', num2str(v_cau1));
    set(handles.f_dia1, 'String', num2str(v_dia1));
    set(handles.f_fdf1, 'String', num2str(v_fdf1));

    set(handles.f_cau3, 'String', num2str(v_cau3));
    set(handles.f_dia3, 'String', num2str(v_dia3));
    set(handles.f_fdf3, 'String', num2str(v_fdf3));
end

if (hObject == handles.dia3)
    set(handles.v_dia1, 'Enable', 'On');
    set(handles.v_cau1, 'Enable', 'On');
    set(handles.v_dia2, 'Enable', 'On');
    set(handles.v_cau2, 'Enable', 'On');
    set(handles.v_dia3, 'Enable', 'Off');
    set(handles.v_cau3, 'Enable', 'On');
    hs=v_emax-v_emin;
    fdd=0;
    n=10;
for i=1:n
    diame=((v_fdf3*v_lar*8*(v_cau3^(2))/(hs*pi*pi*9.8)))^(1/5);
    re=((4*v_cau3)/(pi*v_vis*diame));
    %Darcy Williams
    a=( v_rug3*(3.7*diame)^(-1) )  +(5.1286 * (re^(0.89))^(-1));
    b=(-2)*log10(a);
    fdd=((1/b)^(2));
    v_fdf=fdd;
end
factor=v_fdf;
    set(handles.f_cau3, 'String', num2str(v_cau3));
    set(handles.f_dia3, 'String', num2str(diame));
    set(handles.f_fdf3, 'String', num2str(factor));

    %Restantes
    set(handles.f_cau1, 'String', num2str(v_cau1));
    set(handles.f_dia1, 'String', num2str(v_dia1));
    set(handles.f_fdf1, 'String', num2str(v_fdf1));

    set(handles.f_cau2, 'String', num2str(v_cau2));
    set(handles.f_dia2, 'String', num2str(v_dia2));
    set(handles.f_fdf2, 'String', num2str(v_fdf2));
end
```

lde

```
function varargout = lde(varargin)
% LDE MATLAB code for lde.fig
%      LDE, by itself, creates a new LDE or raises the existing
%      singleton*.
%
%      H = LDE returns the handle to a new LDE or the handle to
%      the existing singleton*.
%
%      LDE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in LDE.M with the given input arguments.
%
%      LDE('Property','Value',...) creates a new LDE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before lde_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to lde_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 lde

% Last Modified by GUIDE v2.5 10-Mar-2019 16:54:49

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @lde_OpeningFcn, ...
                   'gui_OutputFcn',  @lde_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

function lde_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 lde (see VARARGIN)

% Choose default command line output for lde
handles.output = hObject;
axes(handles.axes1);
background=imread('sifon.jpg');
```

```
axis off
imshow(background);

% Update handles structure
guidata(hObject, handles);

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

% --- Outputs from this function are returned to the command line.
function varargout = lde_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 Singularidades.
function Singularidades_Callback(hObject, eventdata, handles)
% hObject    handle to Singularidades (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(lde);
singularidades

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

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

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

Singularidad

```
function varargout = singularidades(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @singularidades_OpeningFcn, ...
                   'gui_OutputFcn',  @singularidades_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 singularidades is made visible.
function singularidades_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
handles.output = hObject;
axes(handles.axes1);
background=imread('singularidad.jpg');
axis off
imshow(background);
% Update handles structure
guidata(hObject, handles);

function varargout = singularidades_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;

function v_emax_Callback(hObject, eventdata, handles)
function v_emax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emax (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 atras.
function atras_Callback(hObject, eventdata, handles)
close(singularidades)
```

```
lde

function v_emin_Callback(hObject, eventdata, handles)
v_emin=str2double(get(hObject,'String'));
if isnan(v_emin)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_emin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emin (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 v_cau_Callback(hObject, eventdata, handles)
v_cau=str2double(get(hObject,'String'));
if isnan(v_cau)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_cau_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_cau (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 v_rug_Callback(hObject, eventdata, handles)
v_rug=str2double(get(hObject,'String'));
if isnan(v_rug)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_rug_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_rug (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 graficar.
function graficar_Callback(hObject, eventdata, handles)
x=-1:0.1:501;
%Singularidad CAUDAL.
%v_emax=100;
```

```
v_emax= get(handles.v_emax,'String');
v_emax=str2double(v_emax);
%v_emin=20;
v_emin= get(handles.v_emin,'String');
v_emin=str2double(v_emin);
%v_rug1=0.0004;
v_rug1= get(handles.v_rug1,'String');
v_rug1=str2double(v_rug1);
%v_rug2=0.0004;
v_rug2= get(handles.v_rug2,'String');
v_rug2=str2double(v_rug2);
%v_lar1=250;
v_lar1= get(handles.v_lar1,'String');
v_lar1=str2double(v_lar1);
%v_lar2=250;
v_lar2= get(handles.v_lar2,'String');
v_lar2=str2double(v_lar2);
%v_dia1=0.2;
v_dia1= get(handles.v_dia1,'String');
v_dia1=str2double(v_dia1);
%v_dia2=0.2;
v_dia2= get(handles.v_dia2,'String');
v_dia2=str2double(v_dia2);
v_vis=0.000001;
%v_fdf1=0.02;
v_fdf1= get(handles.v_fdf1,'String');
v_fdf1=str2double(v_fdf1);
%v_fdf2=0.02;
v_fdf2= get(handles.v_fdf2,'String');
v_fdf2=str2double(v_fdf2);
%%v_k1=6;
v_k1= get(handles.v_k1,'String');
v_k1=str2double(v_k1);
%v_k2=2;
v_k2= get(handles.v_k2,'String');
v_k2=str2double(v_k2);
%v_k3=6;
v_k3= get(handles.v_k3,'String');
v_k3=str2double(v_k3);
fdd=0;

n=10;
hs=v_emax-v_emin;

for i=1:n
    %CAUDAL
    cauda=((hs*pi*pi*9.8/8)*(v_fdf1*v_lar1/(v_dia1^5)+
v_fdf2*v_lar2/(v_dia2^5) +v_k1/v_dia1^4+ v_k2/v_dia1^4 +v_k3/v_dia2^4)^(-
1))^(1/2);
    %Reynolds Tramo 1
    re1=((4*cauda)/(pi*v_vis*v_dia1));
                %Tramo2
    re2=((4*cauda)/(pi*v_vis*v_dia2));
    %Darcy Williams Tramo 1
    a1=( v_rug1*(v_dia1)^(-1) )  +(21.25 * (re1^(-0.9)));
    b1=1.14-(2)*log10(a1);
    fdd1=((1*(b1^(-1)))^(2));
    v_fdf1=fdd1;
    %Darcy Williams Tramo 2
```

```
    a2=( v_rug2*(v_dia2)^(-1) )  +(21.25 * (re2^(-0.9)));
    b2=1.14-(2)*log10(a2);
    fdd2=((1*(b2^(-1)))^(2));
    v_fdf2=fdd2;
end
factor1=v_fdf1;
factor2=v_fdf2;

%ENERGIA MAXIMA HS
%1° Perdida
perdk1=(8*v_k1*cauda^2)/(pi*pi*9.8*v_dia1^4);
a=hs-perdk1;
e1=a;


%2° Perdida
perdf1=(8*factor1*v_lar1*cauda^2)/(pi*pi*9.8*v_dia1^5);
b=a-(perdf1);
e2=a+(x-0)*(b-a)/(250);

%3° Perdida
perdk2=(8*v_k2*cauda^2)/(pi*pi*9.8*v_dia1^4);
c=b-(perdk2);
e3=b;

%4° Perdida
perdf2=(8*factor2*v_lar2*cauda^2)/(pi*pi*9.8*v_dia2^5);
d=c-(perdf2);
e4=c+(x-250)*(d-c)/(250);

%5° Perdida
perdk3=(8*v_k3*cauda^2)/(pi*pi*9.8*v_dia2^4);
e5=perdk3;

y1=(((e1).*(x==0))+(e2).*((0<x)&(x<250))+(e3).*(x==250)+e4.*((250<x)&(x<500))+
e5.*(x==500));
plot(x,y1,'r')
grid on
title ('PERDIDAS DE ENERGÍA')
xlabel ('Distancia(metros)')
ylabel ('Energía Del Sistema(metros)')
text(5,hs-1.75,']A')
text(260,b,']B')
text(485,d-1,'C[')

function v_k2_Callback(hObject, eventdata, handles)
v_k2=str2double(get(hObject,'String'));
if isnan(v_k2)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_k2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_k2 (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 v_k3_Callback(hObject, eventdata, handles)
v_k3=str2double(get(hObject,'String'));
if isnan(v_k3)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_k3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_k3 (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 v_k1_Callback(hObject, eventdata, handles)
v_k1=str2double(get(hObject,'String'));
if isnan(v_k1)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_k1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_k1 (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 v_fdf2_Callback(hObject, eventdata, handles)
v_fdf2=str2double(get(hObject,'String'));
if isnan(v_fdf2)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_fdf2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_fdf2 (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 v_lar2_Callback(hObject, eventdata, handles)
```

```
v_lar2=str2double(get(hObject,'String'));
if isnan(v_lar2)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_lar2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_lar2 (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 v_rug2_Callback(hObject, eventdata, handles)
v_rug2=str2double(get(hObject,'String'));
if isnan(v_rug2)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_rug2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_rug2 (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 v_dia2_Callback(hObject, eventdata, handles)
v_dia2=str2double(get(hObject,'String'));
if isnan(v_dia2)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_dia2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_dia2 (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 v_rug1_Callback(hObject, eventdata, handles)
v_rug1=str2double(get(hObject,'String'));
if isnan(v_rug1)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_rug1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_rug1 (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 v_fdf1_Callback(hObject, eventdata, handles)
v_fdf1=str2double(get(hObject,'String'));
if isnan(v_fdf1)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_fdf1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_fdf1 (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 v_lar1_Callback(hObject, eventdata, handles)
v_lar1=str2double(get(hObject,'String'));
if isnan(v_lar1)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_lar1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_lar1 (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 v_dia1_Callback(hObject, eventdata, handles)
v_dia1=str2double(get(hObject,'String'));
if isnan(v_dia1)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_dia1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_dia1 (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 edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (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 edit18 as text
%        str2double(get(hObject,'String')) returns contents of edit18 as a
double


% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (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 edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (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 edit19 as text
%        str2double(get(hObject,'String')) returns contents of edit19 as a
double


% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (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 edit20_Callback(hObject, eventdata, handles)
% hObject    handle to edit20 (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 edit20 as text
%        str2double(get(hObject,'String')) returns contents of edit20 as a
double


% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit20 (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 edit21_Callback(hObject, eventdata, handles)
% hObject    handle to edit21 (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 edit21 as text
%        str2double(get(hObject,'String')) returns contents of edit21 as a
double


% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit21 (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 edit22_Callback(hObject, eventdata, handles)
% hObject    handle to edit22 (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 edit22 as text
%        str2double(get(hObject,'String')) returns contents of edit22 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit22 (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
```

salida

```
function varargout = Salida(varargin)
% SALIDA MATLAB code for Salida.fig
%      SALIDA, by itself, creates a new SALIDA or raises the existing
%      singleton*.
%
%      H = SALIDA returns the handle to a new SALIDA or the handle to
%      the existing singleton*.
%
%      SALIDA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SALIDA.M with the given input arguments.
%
%      SALIDA('Property','Value',...) creates a new SALIDA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Salida_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Salida_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 Salida

% Last Modified by GUIDE v2.5 29-Mar-2019 20:38:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Salida_OpeningFcn, ...
                   'gui_OutputFcn',  @Salida_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
function Salida_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 Salida (see VARARGIN)

% Choose default command line output for Salida
handles.output = hObject;
axes(handles.axes2);
background=imread('caudal.jpg');
axis off
imshow(background);
% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = Salida_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;

function v_emax_Callback(hObject, eventdata, handles)
v_emax=str2double(get(hObject,'String'));
if isnan(v_emax)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_emax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emax (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 atras_Callback(hObject, eventdata, handles)
close(Salida)
lde

function v_emin_Callback(hObject, eventdata, handles)
v_emin=str2double(get(hObject,'String'));
if isnan(v_emin)
    errordlg('El valor debe ser numerico','ERROR');
end
```

```
function v_emin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emin (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 v_vis_Callback(hObject, eventdata, handles)
v_vis=str2double(get(hObject,'String'));
if isnan(v_vis)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_vis_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_vis (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 grafica_Callback(hObject, eventdata, handles)
%SALIDA DE CAUDAL

%v_vis=0.000001;
v_vis= get(handles.v_vis,'String');
v_vis=str2double(v_vis);
%v_emax=;
v_emax= get(handles.v_emax,'String');
v_emax=str2double(v_emax);
%v_emin=;
v_emin= get(handles.v_emin,'String');
v_emin=str2double(v_emin);
%v_caue=0.2;
v_caue= get(handles.v_caue,'String');
v_caue=str2double(v_caue);
%v_caus=0.07;
v_caus= get(handles.v_caus,'String');
v_caus=str2double(v_caus);
%v_caur=0.30;
v_caur= get(handles.v_caur,'String');
v_caur=str2double(v_caur);
%v_dia1=0.200;
v_dia1= get(handles.v_dia1,'String');
v_dia1=str2double(v_dia1);
%v_dia2=0.32;
v_dia2= get(handles.v_dia2,'String');
v_dia2=str2double(v_dia2);
%v_fdf1=0.02;
v_fdf1= get(handles.v_fdf1,'String');
```

```
v_fdf1=str2double(v_fdf1);
%v_fdf2=0.02;
v_fdf2= get(handles.v_fdf2,'String');
v_fdf2=str2double(v_fdf2);
v_lar1=50;
v_lar2=50;
%v_rug1=0.003;
v_rug1= get(handles.v_rug1,'String');
v_rug1=str2double(v_rug1);
%v_rug2=0.004;
v_rug2= get(handles.v_rug2,'String');
v_rug2=str2double(v_rug2);
%%
%%%CAUDAL FINAL
hs=v_emax-v_emin;
v_caue=0.1;
v_caur=0.40;
v_caus=0.30;
n=10;
for i=1:n
    %CAUDAL DIAMETRO
    syms x

eqn=(hs==(v_fdf1*v_lar1*8*v_caue.^2)/(pi*pi*9.8*v_dia1.^5)+(v_fdf2*v_lar2*8*v_
caus.^2)/(pi*pi*9.8*x.^5)+(8*v_caur.^2)/(pi*pi*9.8*v_dia1.^4));
    v_dia2=vpa(solve(eqn,x,'Real', true));
    v_dia2=double(v_dia2);

    %Reynolds Tramo 1
    re1=((4*v_caue)/(pi*v_vis*v_dia1));
                %Tramo2
    re2=((4*v_caus)/(pi*v_vis*v_dia2));

    %Darcy Williams Tramo 1
    a1=( v_rug1*(v_dia1).^(-1) )  +(21.25 * (re1.^(-0.9)));
    b1=1.14-(2)*log10(a1);
    fdd1=((1*(b1.^-1)).^(2));
    v_fdf1=fdd1;
    %Darcy Williams Tramo 2
    a2=( v_rug2*(v_dia2).^(-1) )  +(21.25 * (re2.^(-0.9)));
    b2=1.14-(2)*log10(a2);
    fdd2=((1*(b2.^(-1))).^(2));
    v_fdf2=fdd2;

end
factor1=v_fdf1;
factor2=v_fdf2;


%%
x=0:0.1:100;
%Grafica
%Perdida Tramo 1
hf1=(8*v_fdf1*v_lar1*v_caue^(2))/(pi*pi*v_dia1^(5)*9.8);
A=hs-hf1;
e1=hs-hf1*x/50;

%Perdida Tramo 2
```

```
hf2=(8*v_caur^(2))/(pi*pi*v_dia1^(4)*9.8);
B=A-hf2;
e2=B;

%Perdida Tramo 3
hf3=(8*v_fdf2*v_lar2*v_caus^(2))/(pi*pi*v_dia2^(5)*9.8);
e3=hf3-hf3*(x-50)/50;

y1=((e1).*((0<=x)&(x<50))+(e2).*((50==x))+(e3).*((50<x)&(x<=100)));

plot(x,y1)
hold off
grid on
title ('PERDIDAS DE ENERGÍA')
xlabel ('Distancia(metros)')
ylabel ('Energía Del Sistema(metros)')
text(0,hs,'A')
text(50,(A+B)*0.5,'B')
text(100,5,'C')
function v_caue_Callback(hObject, eventdata, handles)
v_caue=str2double(get(hObject,'String'));
if isnan(v_caue)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_caue_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_caue (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 v_caur_Callback(hObject, eventdata, handles)
v_caur=str2double(get(hObject,'String'));
if isnan(v_caur)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_caur_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_caur (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 v_caus_Callback(hObject, eventdata, handles)
v_caus=str2double(get(hObject,'String'));
if isnan(v_caus)
    errordlg('El valor debe ser numerico','ERROR');
end
```

```
function v_caus_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_caus (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 v_fdf2_Callback(hObject, eventdata, handles)
v_fdf2=str2double(get(hObject,'String'));
if isnan(v_fdf2)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_fdf2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_fdf2 (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 v_lar2_CreateFcn(hObject, eventdata, handles)

function v_rug2_Callback(hObject, eventdata, handles)
v_rug2=str2double(get(hObject,'String'));
if isnan(v_rug2)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_rug2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_rug2 (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 v_dia2_Callback(hObject, eventdata, handles)
v_dia2=str2double(get(hObject,'String'));
if isnan(v_dia2)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_dia2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_dia2 (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 v_rug1_Callback(hObject, eventdata, handles)
v_rug1=str2double(get(hObject,'String'));
if isnan(v_rug1)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_rug1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_rug1 (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 v_fdf1_Callback(hObject, eventdata, handles)
v_fdf1=str2double(get(hObject,'String'));
if isnan(v_fdf1)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_fdf1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_fdf1 (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 v_lar1_CreateFcn(hObject, eventdata, handles)
function v_dia1_Callback(hObject, eventdata, handles)
v_dia1=str2double(get(hObject,'String'));
if isnan(v_dia1)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_dia1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_dia1 (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 v_lar1_Callback(hObject, eventdata, handles)

function edit17_Callback(hObject, eventdata, handles)
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_lar2 (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 edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (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 edit18 as text
%        str2double(get(hObject,'String')) returns contents of edit18 as a
double


% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (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 edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (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 edit19 as text
%        str2double(get(hObject,'String')) returns contents of edit19 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (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 edit20_Callback(hObject, eventdata, handles)
% hObject    handle to edit20 (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 edit20 as text
%        str2double(get(hObject,'String')) returns contents of edit20 as a
double


% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit20 (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 edit21_Callback(hObject, eventdata, handles)
% hObject    handle to edit21 (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 edit21 as text
%        str2double(get(hObject,'String')) returns contents of edit21 as a
double


% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit21 (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 edit22_Callback(hObject, eventdata, handles)
% hObject    handle to edit22 (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 edit22 as text
%        str2double(get(hObject,'String')) returns contents of edit22 as a
double


% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit22 (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
```

sifon

```
function varargout = sifon(varargin)
% SIFON MATLAB code for sifon.fig
%      SIFON, by itself, creates a new SIFON or raises the existing
%      singleton*.
%
%      H = SIFON returns the handle to a new SIFON or the handle to
%      the existing singleton*.
%
%      SIFON('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SIFON.M with the given input arguments.
%
%      SIFON('Property','Value',...) creates a new SIFON or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before sifon_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to sifon_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 sifon

% Last Modified by GUIDE v2.5 29-Mar-2019 20:43:17

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @sifon_OpeningFcn, ...
                   'gui_OutputFcn',  @sifon_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 sifon is made visible.
function sifon_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 sifon (see VARARGIN)

handles.output = hObject;
axes(handles.axes3);
background=imread('sifon.jpg');
axis off
imshow(background);
% Choose default command line output for sifon
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
function varargout = sifon_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 atras.
function atras_Callback(hObject, eventdata, handles)
close(sifon)
lde
% --- Executes on button press in graficar.
function graficar_Callback(hObject, eventdata, handles)
v_emax= get(handles.v_emax,'String');
v_emax=str2double(v_emax);
```

```
v_emin= get(handles.v_emin,'String');
v_emin=str2double(v_emin);

v_lar= get(handles.v_lar,'String');
v_lar=str2double(v_lar);

v_fdf= get(handles.v_fdf,'String');
v_fdf=str2double(v_fdf);

v_cau= get(handles.v_cau,'String');
v_cau=str2double(v_cau);

v_vis= get(handles.v_vis,'String');
v_vis=str2double(v_vis);

v_rug= get(handles.v_rug,'String');
v_rug=str2double(v_rug);

v_dia= get(handles.v_dia,'String');
v_dia=str2double(v_dia);

v_inter= get(handles.v_inter,'String');
v_inter=str2double(v_inter);

x=linspace(0,5000,100);
ci=v_emin;
ca=100;
ci=108;
cf=60;

%Graficas
a=(ci-60)/3125000;
b=(20-ci)/2500;

ecv=(a*x.^2 + b*x+ ca);
lr=(cf-ci)*(x-2500)/2500 +ci;
%ci=Cota Intermedia
%e.cv= Ecuacion concava hacia abajo
%lr= Linea recta punto superior hasta el inferior
y1=100*((0<=x)&(x<3))+(ecv).*((3<=x)&(x<2500))+(ci).*(x==2500)+(lr).*((2500<=x
)&(x<5000)+(cf).*(x==5000));



%Segunda Ecuación.
%Desconocer Diametro
v_rug=0.0000035;
v_cau=0.03;
v_lar=5000;
v_vis=0.000001;
v_fdf=0.02;
v_emax=ca;
v_emin=cf;

%DIAMETRO
hs=v_emax-v_emin;
%igualamos nuestra energia del sistema en factor de friccion
```

```
fdd=0;
%tenemos f,
%necesitamos calcular un valor, la cual debe cohincidir con el
%valor ingresado. En caso de no ser igual, seguimos trabajando con el
%resultante hasta que estos coincidan.
n=10;
for i=1:n
    %calculamos caudal
    syms x
    eqn=(1+v_fdf*v_lar/x)*(1/x^4)==((9.8*pi*pi*hs)/(8*v_cau^2));
    diame=vpa(solve(eqn,x,'Real', true));
    diame=double(diame);
    %%calculamos re
    re=((4*v_cau)/(pi*v_vis*diame));
    %Darcy Williams
    a=( v_rug*(diame)^(-1) )  +(21.25 * (re^(-0.9)));
    b=1.14-(2)*log10(a);
    fdd=((1*(b^(-1)))^(2));
    fdd=round(fdd,3);
    %a=solve(1*(f)^(-1/2)==(-2)*log10((v_rug*(3.7*v_dia)^(-1) )  +(5.1286 *
(re^((f)^(1/2)))^(-1))),f);
    v_fdf=fdd;
end
    factor=v_fdf;

    %CALCULAR DIAMETRO COMERCIAL
diame=round(diame,1);

    %CALCULAR DIAGRAMA DE LA FIGURA
v_emax=ca;
v_emin=cf;
v_dia=diame;
hs=v_emax-v_emin;


%igualamos nuestra energia del sistema en factor de friccion
fdd=0;
%tenemos f,
%necesitamos calcular un valor, la cual debe cohincidir con el
%valor ingresado. En caso de no ser igual, seguimos trabajando con el
%resultante hasta que estos coincidan.
n=10;
for i=1:n
    %CAUDAL
    cauda=(hs*(diame/(diame+v_fdf*v_lar))*(9.8*pi*pi*diame^4)/8)^(1/2);
    %Reynolds
    re=((4*cauda)/(pi*v_vis*v_dia));
    %Darcy Williams
    a=( v_rug*(v_dia)^(-1) )  +(21.25 * (re^(-0.9)));
    b=1.14-(2)*log10(a);
    %Nuevo Factor de Ficcion.
    fdd=((1*(b^(-1)))^(2));
    %Iteracion.
    v_fdf=fdd;
end
factor=v_fdf;
```

```
%Valor punto medio.
Pinicial=ca+10.33;
Pintermedio=ca-((factor*2500*8*cauda^2)/(pi*pi*9.8*diame^5));
Pfinal=ca-((factor*5000*8*(cauda^(2))/(pi*pi*9.8*diame^(5))));

%Ecuaciones de Grafica
x=linspace(0,5000,100);
%Grafica 2
y2=x*(Pfinal-100)/5000 +100;

Patm=Pfinal+10.33;
%Grafica 3
y3=x*(Pfinal+10.33-110.33)/5000 +110.33;

plot(x,y1)
hold on
plot(x,y2)
axis manual
plot(x,y3)
hold off
grid on
title ('PERDIDAS DE ENERGÍA')
xlabel ('Distancia(metros)')
ylabel ('Energía Del Sistema(metros)')
text(0,v_emax,'A')
text(2500,ci,'B')
text(5000,v_emin,'C')

function v_lar_Callback(hObject, eventdata, handles)
v_lar=str2double(get(hObject,'String'));
if isnan(v_lar)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_lar_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_lar (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 v_vis_Callback(hObject, eventdata, handles)
v_vis=str2double(get(hObject,'String'));
if isnan(v_vis)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_vis_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_vis (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 v_fdf_Callback(hObject, eventdata, handles)
v_vis=str2double(get(hObject,'String'));
if isnan(v_vis)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_fdf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_fdf (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 v_rug_Callback(hObject, eventdata, handles)
v_rug=str2double(get(hObject,'String'));
if isnan(v_rug)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_rug_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_rug (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 v_cau_Callback(hObject, eventdata, handles)
v_cau=str2double(get(hObject,'String'));
if isnan(v_cau)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_cau_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_cau (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 v_emax_Callback(hObject, eventdata, handles)
v_emax=str2double(get(hObject,'String'));
```

```
if isnan(v_emax)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_emax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emax (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 v_emin_Callback(hObject, eventdata, handles)
v_emin=str2double(get(hObject,'String'));
if isnan(v_emin)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_emin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_emin (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 v_inter_Callback(hObject, eventdata, handles)
v_inter=str2double(get(hObject,'String'));
if isnan(v_inter)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_inter_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_inter (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 v_dia_Callback(hObject, eventdata, handles)
v_dia=str2double(get(hObject,'String'));
if isnan(v_dia)
    errordlg('El valor debe ser numerico','ERROR');
end
function v_dia_CreateFcn(hObject, eventdata, handles)
% hObject    handle to v_dia (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 edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (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 edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a
double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (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 edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (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 edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a
double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (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 edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (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 edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12 as a
double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (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 edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (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 edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13 as a
double


% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (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 edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (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 edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14 as a
double


% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (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 edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (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 edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15 as a
double


% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (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 edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (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 edit16 as text
%        str2double(get(hObject,'String')) returns contents of edit16 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit16 (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 edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit17 (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 edit17 as text
%        str2double(get(hObject,'String')) returns contents of edit17 as a
double


% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (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 edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (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 edit18 as text
%        str2double(get(hObject,'String')) returns contents of edit18 as a
double


% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (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
```