



**INSTITUTO POLITÉCNICO NACIONAL**  
**ESCUELA SUPERIOR DE INGENIERÍA Y ARQUITECTURA**  
**UNIDAD ZACATENCO**

**SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN**

---

*"Análisis hidráulico de redes de distribución de agua potable, con algoritmo genético y su aplicación a la red del municipio de Tezoyuca en el Estado de México"*

**TESIS**

**PARA OBTENER EL GRADO DE  
MAESTRO EN INGENIERÍA CIVIL**

**PRESENTA:**

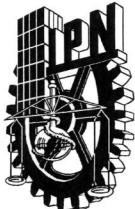
*ING. JOSUE EMMANUEL CRUZ BARRAGAN*

**DIRECTORES:**

*M. C. LUCIO FRAGOSO SANDOVAL  
M. C. LUIS POMPOSO VIGUERAS MUÑOZ*



*Ciudad de México diciembre, 2021*



# INSTITUTO POLITÉCNICO NACIONAL

## SECRETARIA DE INVESTIGACIÓN Y POSGRADO

SIP-13  
REP 2017

### ACTA DE REGISTRO DE TEMA DE TESIS Y DESIGNACIÓN DE DIRECTOR DE TESIS

Ciudad de México 30 de abril del 2021

El Colegio de Profesores de Posgrado de la E.S.I.A.-U. Z. en su Sesión

(Unidad Académica)

ordinaria No. 4 celebrada el día 23 del mes abril de 2021 conoció la solicitud presentada por el (la) alumno (a):

Apellido Paterno:	CRUZ	Apellido Materno:	BARRAGAN	Nombre (s):	JOSUE EMMANUEL
-------------------	------	-------------------	----------	-------------	----------------

Número de registro: B | 1 | 9 | 0 | 3 | 0 | 3

del Programa Académico de Posgrado: MAESTRÍA EN INGENIERÍA CIVIL

Referente al registro de su tema de tesis; acordando lo siguiente:

1.- Se designa al aspirante el tema de tesis titulado:

Análisis hidráulico de redes de distribución de agua potable, con algoritmo genético y su aplicación a la red del municipio de Tezoyuca en el Estado de México.

Objetivo general del trabajo de tesis:

Desarrollar un programa computacional que automatice el cálculo del diámetro de tuberías de una red de distribución de agua potable, utilizando los algoritmos genéticos como método para obtener los diámetros más económicos que cumplan con la velocidad y presión dadas por el usuario en el municipio de Tezoyuca, Estado de México.

2.- Se designa como Directores de Tesis a los profesores:

Director: M. en C. Lucio Fragoso Sandoval 2º Director: M. en C. Luis Pomposo Vigueras Muñoz  
No aplica: □

3.- El Trabajo de investigación base para el desarrollo de la tesis será elaborado por el alumno en:

La Sección de Estudios de Posgrado e Investigación de la E.S.I.A.-U.Z.

que cuenta con los recursos e infraestructura necesarios.

4.- El interesado deberá asistir a los seminarios desarrollados en el área de adscripción del trabajo desde la fecha en que se suscribe la presente, hasta la aprobación de la versión completa de la tesis por parte de la Comisión Revisora correspondiente.

Director(a) de Tesis

M. en C. Lucio Fragoso Sandoval

Aspirante

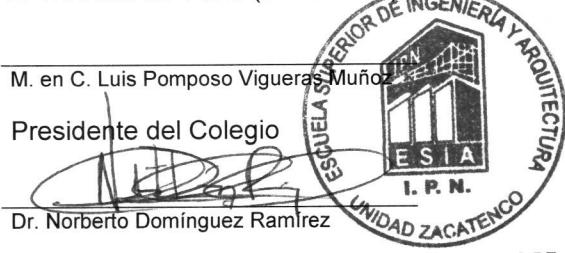
Josue Emmanuel Cruz Barragan

2º Director de Tesis (en su caso)

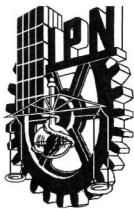
M. en C. Luis Pomposo Vigueras Muñoz

Presidente del Colegio

Dr. Norberto Domínguez Ramírez



SECCIÓN DE ESTUDIOS DE  
POSGRADO E INVESTIGACIÓN



# INSTITUTO POLITÉCNICO NACIONAL

## SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

SIP-14  
REP 2017

### ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 18:00 horas del día 7 del mes de junio

del 2021 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Posgrado de: E.S.I.A. – U.Z. para examinar la tesis titulada:

Análisis hidráulico de redes de distribución de agua potable, con algoritmo genético y su aplicación a la red del municipio de Tezoyuca en el Estado de México. del (la) alumno (a):

Apellido Paterno: <span style="border: 1px solid black; padding: 2px;">CRUZ</span>	Apellido Materno: <span style="border: 1px solid black; padding: 2px;">BARRAGAN</span>	Nombre (s): <span style="border: 1px solid black; padding: 2px;">JOSUE EMMANUEL</span>
--	--	--

Número de registro: B | 1 | 9 | 0 | 3 | 0 | 3

Aspirante del Programa Académico de Posgrado: MAESTRÍA EN INGENIERÍA CIVIL

Una vez que se realizó un análisis de similitud de texto, utilizando el software antiplagio, se encontró que el trabajo de tesis tiene 3 % de similitud. **Se adjunta reporte de software utilizado.**

Después que esta Comisión revisó exhaustivamente el contenido, estructura, intención y ubicación de los textos de la tesis identificados como coincidentes con otros documentos, concluyó que en el presente trabajo **SI  NO  SE CONSTITUYE UN POSIBLE PLAGIO.**

**JUSTIFICACIÓN DE LA CONCLUSIÓN:** (*Por ejemplo, el % de similitud se localiza en metodologías adecuadamente referidas a fuente original*)

El porcentaje de similitud corresponde a referencias, títulos y nombres propios, no a similitudes puntuales en metodologías o procedimientos; la tesis presenta un texto original y adecuado.

**\*\*Es responsabilidad del alumno como autor de la tesis la verificación antiplagio, y del Director o Directores de tesis el análisis del % de similitud para establecer el riesgo o la existencia de un posible plагio.**

Finalmente y posterior a la lectura, revisión individual, así como el análisis e intercambio de opiniones, los miembros de la Comisión manifestaron **APROBAR  SUSPENDER  NO APROBAR**  la tesis por **UNANIMIDAD  o MAYORÍA**  en virtud de los motivos siguientes:

La tesis aquí presentada cumple con los parámetros requeridos y el nivel de conocimientos establecidos en los objetivos y en la metodología planteada.

### COMISIÓN REVISORA DE TESIS

M. en C. Lucio Fragoso Sandoval  
Director de Tesis

M. en C. Jaime Roberto Ruiz y Zurvia Flores

M. en C. Luis Pomposo Vigueras Muñoz  
2º Director de Tesis

Dr. Juan Manuel Navarro Pineda





**INSTITUTO POLITÉCNICO NACIONAL**  
**SECRETARÍA DE INVESTIGACIÓN Y POSGRADO**

*CARTA CESIÓN DE DERECHOS*

En la Ciudad de México el día 01 del mes diciembre del año 2021, el (la) que suscribe **Josue Emmanuel Cruz Barragan** alumno (a) del Programa de Maestría en Ingeniería Civil con número de registro **B-190303**, adscrito a Escuela Superior de Ingeniería y Arquitectura U. Z., manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección del **M. C. Lucio Fragoso Sandoval y M. C. Luis Pomposo Vigueras Muñoz** y cede los derechos del trabajo intitulado **“Análisis hidráulico de redes de distribución de agua potable, con algoritmo genético y su aplicación a la red del municipio de Tezoyuca en el Estado de México”**, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección **ing.cruzbaragan@gmail.com**. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Josue Emmanuel Cruz Barragan



## AGRADECIMIENTOS.

A **CONACYT** por brindarme todos los recursos y herramientas que fueron necesarios para llevar a cabo el proceso de investigación. No hubiese podido arribar a estos resultados de no haber sido por su incondicional ayuda.

Al **INSTITUTO POLITÉCNICO NACIONAL** mi Alma Máter, por haber compartido su esencia de coraje, dedicación y superación poniendo siempre "*la técnica al servicio de la patria*".

A la **ESCUELA SUPERIOR DE INGENIERÍA Y ARQUITECTURA U. Z.**, por haber sido durante estos años mi recinto y darme las bases para formarme en el área de Ingeniería Civil.

A mis directores de tesis, **M. C. Lucio Fragoso Sandoval y M. C. Luis Pomposo Vigueras Muñoz** por haberme guiado, no solo en la elaboración de este trabajo de titulación, sino también por sus consejos y correcciones para hoy poder culminar este trabajo.

A todos mis profesores: **Dr. Francisco Antelmo Díaz Guerra, Dr. Oscar Cruz Castro, Dr. Sergio Cruz León, Dr. Juan Manuel Navarro Pineda, Dr. Mario Ulloa Ramírez, Dr. Héctor Jaime Mora López y M. C. Pino Durán Escamilla**, por transmitirme sus conocimientos y experiencias.



## DEDICATORIAS.

Esta tesis está dedicada a:

Mis padres **Victor Manuel Cruz Gonzalez y Maria de Jesus Barragan Rosas** quienes con su amor, paciencia, esfuerzo, consejos y palabras de aliento hicieron de mí una mejor persona y de una u otra forma me acompañan en todos mis sueños y metas. Gracias por permitirme llegar a cumplir hoy un sueño más.

Mi novia, **Evelin Hernández González** que indudablemente ha sabido estar conmigo en cada paso de mi vida desde que nos conocimos, creíste en mí y me apoyaste, guiaste y sostuviste cuando más te he necesitado. Eres sin duda la mejor persona que podría haberme cruzado en el camino, eres mi pilar y mi amor.

Mis hermanos **Brenda Cruz Barragan y Ryan Nathanael Cruz Barragan** por su cariño, paciencia y apoyo incondicional, durante todo este proceso, por estar conmigo en todo momento gracias. A **Sparky** por siempre cuidarme y desvelarse a mi lado todas las noches.

Todos mis amigos, a los cuales no terminaría de nombrar y mucho menos de agradecer el apoyo, la dedicación, la fortaleza, el poder escuchar, de estar ahí cuando más los necesito, los quiero mucho y aunque ya no nos vemos tanto como cuando íbamos en la escuela, siempre están en mi mente y en mi corazón.

¡GRACIAS TOTALES!



# “ANÁLISIS HIDRÁULICO DE REDES DE DISTRIBUCIÓN DE AGUA POTABLE, CON ALGORITMO GENÉTICO Y SU APLICACIÓN A LA RED DEL MUNICIPIO DE TEZOYUCA EN EL ESTADO DE MÉXICO.”

**Resumen.**

**Abstract.**

**Estado del arte.**

**Problema.**

**Antecedentes y justificaciones.**

**Objetivos.**

**Hipótesis.**

**I. Red de distribución de agua potable.**

**II. Introducción a los algoritmos genéticos.**

**III. Estructura del programa.**

**IV. Estructura de la Interfaz Gráfica de Usuario.**

**V. Ejemplo de Aplicación del Programa.**

**Conclusiones.**

**Recomendaciones.**

**Bibliografía.**

**Anexos.**



## Contenido

RESUMEN .....	6
ABSTRACT .....	7
ESTADO DEL ARTE .....	8
Internacionales .....	8
Nacionales .....	9
PROBLEMA .....	11
ANTECEDENTES Y JUSTIFICACIÓN .....	11
Redes de distribución de agua .....	11
Algoritmos genéticos .....	12
OBJETIVOS .....	13
HIPÓTESIS .....	14
I. RED DE DISTRIBUCIÓN DE AGUA POTABLE .....	15
1.1 Tipología .....	15
1.1.1 Redes abiertas .....	15
1.1.2 Redes cerradas .....	16
1.2 Consideraciones según las normas .....	17
1.3 Ecuaciones hidráulicas .....	18
1.4 Análisis hidráulico .....	21
1.4.1 Método del gradiente .....	21
II. INTRODUCCIÓN A LOS ALGORITMOS GENÉTICOS .....	26
2.1 ¿Qué es un algoritmo genético? .....	26
2.2 Algoritmo genético .....	26
2.3 Codificación .....	27
2.4 Población .....	28
2.4.1 Tamaño de población .....	28
2.4.2 Población inicial .....	28
2.5 Operadores genéticos .....	29
2.5.1 Selección .....	29
2.5.2 Cruce .....	30
2.5.3 Mutación .....	31
2.5.4 Elitismo .....	32



2.6 Función de evaluación (Fitness) .....	32
2.6.1 Restricciones hidráulicas.....	32
2.6.2 Función de penalización.....	33
2.6.3 Función de costos .....	33
III. ESTRUCTURA DEL CÓDIGO .....	36
3.1 Python como lenguaje a usar .....	36
3.1.1 ¿Qué es Python? .....	36
3.1.2 ¿Por qué Python? .....	36
3.1.3 Bibliotecas utilizadas .....	37
3.2 Archivo: GradienteHidraulico.py .....	39
3.2.1 Clase Nodos .....	39
3.2.2 Clase Tuberías .....	40
3.2.3 Clase RedHidraulica .....	41
3.2.4 Clase Gradiente .....	42
3.3 Archivo: AlgoritmoGenetico.py .....	51
3.3.1 Clase Individuo.....	51
3.3.2 Clase AlgoritmoGenetico.....	55
IV. ESTRUCTURA DE LA INTERFAZ GRÁFICA DE USUARIO .....	65
4.1 Qt Designer como herramienta de diseño.....	65
4.1.1 ¿Qué es Qt Designer? .....	65
4.1.2 ¿Por qué usar Qt Designer?.....	65
4.2 Diseño de la Interfaz Gráfica de Usuario .....	66
4.2.1 Inicialización del programa .....	66
4.2.2 Página de inicio.....	67
4.2.3 Menú desplegable .....	68
4.2.4 Introducir datos .....	68
4.2.5 Análisis de Redes Hidráulicas .....	75
4.2.6 Algoritmo Genético.....	76
V. EJEMPLO DE APLICACIÓN DEL PROGRAMA .....	81
5.1 Red Hidráulica Propuesta .....	81
5.1.1 Red del municipio de Tezoyuca en el Estado de México.....	81
5.1.2 Modificaciones a la Red Original .....	82



5.1.3 Ingresar Datos al Programa REDGENHID .....	83
5.1.4 Ejecución del Algoritmo Genético.....	84
5.1.5 Análisis Hidráulico en REDGENHID.....	86
5.1.6 Análisis Hidráulico en Epanet 2.....	90
5.1.7 Comparativa entre los resultados del programa REDGENHID y Epanet 2 .....	93
Conclusiones .....	96
Recomendaciones .....	97
Bibliografía .....	98
Anexos.....	100
Anexo A. Archivo: GradienteHidraulico.py .....	100
Anexo B. Archivo: AlgoritmoGenetico.py .....	107
Anexo C. Archivo: FileTools.py .....	113
Anexo D. Archivo: ui_Functions.py .....	117
Anexo E. Archivo: clases_mpl_pyqt.py .....	120
Anexo F. Archivo: progressBarPYQT5.py.....	121
Anexo G. Archivo: ErrorPYQT5.py .....	123
Anexo H. Archivo: MultiMenu.py.....	123

## Tablas

Tabla 1. Velocidades máximas permisibles (Normas Técnicas, 2008) .....	18
Tabla 2. Longitud de las tuberías, metros .....	82
Tabla 3. Datos de los nodos.....	82
Tabla 4. Datos de los costos .....	83
Tabla 5. Tiempo de ejecución del AG para las diferentes soluciones.....	85
Tabla 6. Diámetros óptimos encontrados en cada ejecución del AG .....	86
Tabla 7. Resultados de las tuberías derivados del análisis hidráulico .....	89
Tabla 8. Resultados de los nodos derivados del análisis hidráulico .....	90
Tabla 9. Tabla de resultados derivados de las tuberías, Epanet 2. ....	92
Tabla 10. Tabla de resultados derivados de los nodos, Epanet 2 .....	93
Tabla 11. Diferencias entre los resultados del programa REDGENHID y Epanet 2, parte 2.....	95
Tabla 12. Diferencias entre los resultados del programa REDGENHID y Epanet 2, parte 1.....	95



## Figuras

Figura 1. Red abierta. (www.tutorialesaldia.com, s.f.) .....	15
Figura 2. Desventaja de la red abierta (www.tutorialesaldia.com, s.f.) .....	16
Figura 3. Red cerrada (www.tutorialesaldia.com, s.f.) .....	16
Figura 4. Desventaja de la red cerrada (www.tutorialesaldia.com, s.f.) .....	17
Figura 5. Algoritmo Genético Canónico (Melián Batista, Moreno Pérez, & Moreno Vega, 2009) .....	27
Figura 6. Individuo Binario.....	28
Figura 7. Cruce de un punto.....	30
Figura 8. Cruce de 3 puntos .....	31
Figura 9. Inicialización del Programa .....	66
Figura 10. Página de inicio.....	67
Figura 11. Menú desplegable .....	68
Figura 12. Introducir datos .....	69
Figura 13. Componentes Generales de la Sección de Datos .....	70
Figura 14. Opción Manual .....	71
Figura 15. Opción archivo de Excel.....	71
Figura 16. Opción archivo CSV .....	72
Figura 17. Opción archivo de Epanet .....	73
Figura 18. Componentes Generales de la Sección de Red hidráulica .....	74
Figura 19. Análisis de Redes Hidráulicas .....	75
Figura 20. Algoritmo Genético.....	79
Figura 21. Circular progressbar .....	79
Figura 22. Red hidráulica del municipio de Tezoyuca .....	81
Figura 23.Introducir Datos.....	83
Figura 24. Parámetros introducidos en el AG .....	84
Figura 25. Mejores soluciones encontradas en cada generación .....	85
Figura 26. Enviar los diámetros a analizar .....	86
Figura 27. Análisis hidráulico .....	87
Figura 28. Resumen de velocidades y presiones .....	88
Figura 29. Resultado del análisis hidráulico en Epanet 2 .....	91



## RESUMEN.

Según la CONAGUA una red de distribución de agua potable: “es el conjunto de tuberías, accesorios y estructuras que conducen el agua desde tanques de servicio hasta las tomas domiciliarias o hidrantes públicos. Su finalidad es proporcionar agua a los usuarios para consumo doméstico, público, comercial, industrial y para condiciones extraordinarias como el extinguir incendios. La red debe proporcionar este servicio todo el tiempo, en cantidad suficiente, con la calidad requerida y a una presión adecuada. Los límites de calidad del agua, para que pueda ser considerada como potable se establecen en la Norma Oficial Mexicana NOM-127-SSA1 vigente” (Normas Técnicas, 2008).

Para que esta definición sea válida, es necesario determinar el tamaño de los componentes de la red. Por tanto, determinar el diámetro ideal de la tubería que constituye cualquier sistema de distribución de agua no es un problema directo, y por su complejidad, se relaciona con un problema combinado denominado NP-Duro, lo que significa que no se puede resolver mediante un método determinista.

Para solucionar este problema es necesario adoptar un método especial y utilizar un tiempo de cálculo razonable para que la configuración del diámetro cumpla con los límites máximos y mínimos de velocidad y presión, y logre valores aceptables en la consecución de objetivos (reducción de costos). El uso de algoritmos genéticos puede resolver este tipo de problema de combinación NP-Duro. Por tanto, optimiza la red de suministro de agua.

Para encontrar el mejor conjunto de soluciones, se desarrolló un programa informático llamado **REDGENHID**, basado en el uso de algoritmos genéticos, propuesto por el Dr. John Henry Holland, adaptado en esta tesis para su aplicación en redes de distribución de agua potable para cumplir con exigencias económicas, y bajo los límites de presión y velocidad dados por el usuario. Dicho programa se desarrolló en el lenguaje de programación Python, usando para crear su interfaz gráfica a Qt Designer. Para demostrar la eficacia del programa se usó como ejemplo la red de Tezoyuca en el Estado de México, originalmente constituida por tres circuitos cerrados, 21 nodos de demanda, 23 tuberías de conducción y un único embalse de altura constante de 20m. Los nodos se encuentran a diferentes cotas.



## ABSTRACT.

According to CONAGUA, a drinking water distribution network: "is the set of pipes, accessories and structures that conduct water from service tanks to household outlets or public hydrants. Its purpose is to provide water to users for domestic, public, commercial, industrial consumption and for extraordinary conditions such as extinguishing fires. The network must provide this service all the time, in sufficient quantity, with the required quality and at an adequate pressure. The water quality limits, so that it can be considered as potable, are established in the current Official Mexican Standard NOM-127-SSA1" (Normas Técnicas, 2008).

For this definition to be effective, it is necessary to determine the dimensions of the network components. Thus, the establishment of the ideal diameters of the ducts that make up any water distribution system is not a direct problem and due to its complexity, it is associated with the combinatorial problem called NP-Hard, which means that it is not possible to use a deterministic method to solve it.

To solve this problem, special methodologies are required that, using a reasonable computational time, allow to obtain a configuration of diameters that complies with the limitations (speed and pressure), as well as to achieve acceptable values in the objectives sought (lower cost). The use of genetic algorithms allows solving this type of NP-Hard combinatorial problems; consequently, it optimizes a water network.

To find the best set of solutions, a computer program called **REDGENHID** was developed, based on the use of genetic algorithms, proposed by Dr. John Henry Holland, adapted in this thesis for application in drinking water distribution networks to comply with economic requirements, and under the pressure and speed limits given by the user. This program was developed in the Python programming language, using Qt Designer to create its graphical interface. To demonstrate the effectiveness of the program, the Tezoyuca network in the State of Mexico was used as an example, originally constituted by three closed circuits, 21 demand nodes, 23 conduction pipes and a single reservoir with a constant height of 20m. The other nodes are at different elevations.



## ESTADO DEL ARTE.

Después de realizar la búsqueda de antecedentes de la investigación se encontraron tres de nivel internacional y dos nacionales, las cuales se muestran a continuación.

### Internacionales.

(Rabuñal Dopico, 2007) en su tesis doctoral: “Uso de Técnicas de Inteligencia Artificial en Ingeniería Civil”, tuvo como objetivo general dar un nuevo enfoque en la resolución de problemas en la Ingeniería Civil haciendo uso de las Redes Neuronales Artificiales, los Algoritmos Genéticos y la Programación Genética. Se desarrolló una herramienta informática que permita a ingenieros no familiarizados con la Inteligencia Artificial hacer uso de estas técnicas de una forma cómoda y práctica sin la necesidad de tener un profundo conocimiento en esta área. Este sistema se probó en dos áreas de la ingeniería civil:

- ***Ingeniería en construcción.***

Debido a que no existen formas de medir empíricamente la longitud de anclaje necesaria para lograr una buena adherencia entre las vigas de concreto armado y una barra de acero, se utilizó Programación Genética para encontrar la longitud de anclaje necesaria. Así mismo se utilizaron Redes Neuronales Artificiales para, de una manera menos interpretable, dar un resultado más predictivo.

- ***Hidrología.***

Se demostró cómo las Redes Neuronales Artificiales y las Programación Genética pueden trabajar conjuntamente para resolver el problema de la predicción del caudal de una cuenca urbana.

A partir de los resultados obtenidos en la investigación, se tuvo como conclusiones que debido a los ensayos extensos realizados se entrenó una Red Neuronal Artificial con una alta capacidad de predicción (0.96 de coeficiente de correlación), pero cuya desventaja es que un ingeniero civil al no estar familiarizado con estas técnicas no puede entender o explicar los valores que produce lo cual es un obstáculo para su utilización. Por otro lado, al hacer uso de la Programación Genética la expresión se puede manipular para obtener una fórmula para la longitud de anclaje susceptible de compararse con normas nacionales e internacionales la cual se ajusta a las características de las normas utilizadas actualmente. Dentro del área de la hidrología, los caudales obtenidos se aproximan de forma satisfactoria a los caudales reales, demostrando ser mejores que las técnicas clásicas de hidrología y automatizando todo el proceso.

(Carhuapoma Mendoza & Chahuayo Durán, 2019) en su tesis: “Diseño del Sistema de Abastecimiento de Agua Potable en la Rinconada de Pamplona Alta, aplicando EPANET y Algoritmos Genéticos para la Localización de Válvulas Reductoras de



Presión”, tuvo como objetivo general presentar una metodología que permitiera localizar automáticamente válvulas reductoras de presión para realizar el diseño de un sistema de abastecimiento que cumpla con el reglamento vigente. Utilizó un algoritmo genético multiobjetivo, el cual en su codificación incluye el uso del Epanet Programmer’s Toolkit, el cual fue usado como motor de cálculo hidráulico. A partir de los resultados obtenidos en la investigación, tuvo como conclusiones que en algunas tuberías los resultados de velocidad de flujo se encontraban por debajo de 0.6 m/s contrastando esto con su reglamento que indica de forma no restrictiva, en lo posible la velocidad en las líneas de agua no debe ser menor de 0.6 m/s. También se comprobó que el programa EPANET tiene la ventaja de poder adaptarse a las necesidades de este tipo de cálculo gracias a su librería EPANET Programmer’s Toolkit, la cual permite asignar de manera automática el consumo en los nodos de la red.

(Flores García, 2019) en su tesis: “Aplicación del Algoritmo Genético para el Cálculo del Diámetro de las Tuberías de una Red de Distribución de Agua Potable en el Distrito de Tarapoto 2018”, tuvo como objetivo general mejorar el cálculo del diámetro de tuberías de una red de distribución de agua potable utilizando un algoritmo genético. Para realizar la tesis se tomó en cuenta que las velocidades deben estar entre 3 y 10 m/s y las presiones entre 10 y 50 m.c.a., también se hizo hincapié en el costo de las tuberías y el tiempo que tarda en calcular los diámetros el algoritmo. A partir de los resultados obtenidos se demostró que un profesional de la carrera de ingeniería civil utilizando un programa como Excel tarda aproximadamente 45 horas en calcular los diámetros de la red de distribución teniendo hasta un 53% de las velocidades inadecuadas y un 32% de las presiones inadecuadas lo cual causaría rupturas en las tuberías y transportaría de manera inadecuada el agua. Mientras que el sistema informático basado en el algoritmo genético tiene el 100% de las velocidades y de las presiones adecuadas en la red debido a que se obtuvo el diámetro adecuado en un tiempo de sólo 11 minutos lo cual representa el 99.50% de reducción de tiempo. Se tuvo como conclusión que el método del gradiente es el mejor para el cálculo hidráulico y que combinándolo con un algoritmo genético se reduce el costo de adquisición de las tuberías en un 82.03%.

## Nacionales.

(Jarquín Laguna, 2014) en su tesis: “Aplicación de Algoritmos Genéticos en Ingeniería Civil”, muestra a los algoritmos genéticos como una herramienta útil que puede aplicarse a la ingeniería civil tomando como ejemplos:

- **Proyectos carreteros:**

Lo describe como una herramienta de ayuda en la optimización de la programación y la selección de proyectos carreteros, dando resultados más económicos debido a una mejor logística.



- **Sistema de bombeo:**

Con el uso de algoritmos genéticos se puede saber que bombas deberán estar encendidas y cuales apagadas para un determinado gasto inicial en el sistema de tuberías en serie.

- **Estimación de fugas:**

Con el algoritmo genético se pretende localizar y estimar las fugas de agua en una red de tuberías de agua potable.

- **Diseño sísmico:**

Se realizó un diseño sísmico de un edificio de estructuras metálicas por medio de algoritmos genéticos. La tarea del algoritmo genético es seleccionar elementos que cumplan con los requisitos de diseño del catálogo de componentes de acero para brindar comodidad y seguridad a la estructura.

- **Distribución de amortiguadores viscosos no lineales:**

El algoritmo genético se usa como herramienta para obtener la mejor distribución de amortiguadores viscosos no lineales de tal manera que se cumplan los criterios de diseño deseados.

A partir de los resultados obtenidos en la investigación, se tuvo como conclusiones que a pesar de que los resultados finales presentaron algunos detalles, se obtuvieron aproximaciones muy buenas a la solución óptima. Por ejemplo, solo una fuga no fue localizada de manera correcta pero las demás tuvieron un margen de error muy pequeño en su localización. Para la Distribución de amortiguadores viscosos no lineales solo hubo un error en la dirección X en dos entrepisos, pero en los ocho entrepisos restantes se obtuvieron resultados muy semejantes. Se demostró que los algoritmos genéticos, a pesar de tener diferentes criterios de optimización para cada problema, pueden solucionar problemas de ingeniería civil en todas sus áreas. Y, además de todo, pueden entregar alternativas de solución que pudieran no ser la solución ideal, pero pueden ser de ayuda en casos donde no se disponga del acceso a ciertos materiales como suele ocurrir en lugares alejados del centro del país.

(Guzmán Hurtado, 2009) en su tesis: "Algoritmos Genéticos y Epanet 2.0 para la Localización Optima de Válvulas Reductoras de Presión en Redes de Distribución de Agua Potable", tuvo como objetivo general desarrollar un método automatizado mediante un algoritmo genético y Epanet 2.0, para ayudar en el control de presiones en redes de distribución de agua potable, mediante la localización óptima de válvulas reductoras de presión. A partir de los resultados obtenidos en la investigación, tuvo como conclusiones que como se consideraron todos los tubos de la red como posibles candidatos el espacio de búsqueda que se tiene que enfrentar es enorme y aun así el algoritmo genético demostró ser una técnica



robusta que puede solventar estos inconvenientes, pero no descarta la posibilidad de que si se toman en cuenta solo unos tubos como posible solución, el algoritmo genético necesitará un menor número de simulaciones de la red para encontrar la solución.

## PROBLEMA

Considerando que los conocimientos impartidos en la profesión de la ingeniería civil son escasos o nulos en el área de hidráulica en México sobre los lenguajes de programación y los algoritmos genéticos, ¿Cómo hacerles notar a los ingenieros civiles las ventajas de su funcionamiento sin que estos tengan un profundo conocimiento en el tema?, ¿Cómo mostrar que el uso de esta herramienta mejoraría de manera considerable la productividad de cualquier ingeniero civil?

## ANTECEDENTES Y JUSTIFICACIÓN

A pesar de las increíbles aplicaciones que se le pueden dar a los lenguajes de programación en las diferentes áreas de la ingeniería, en el caso de ingeniería civil su uso es aún muy básico o casi nulo. Esta tesis surge de esta inquietud.

Por lo tanto, se buscó un caso de estudio para la realización de un programa que además de cumplir con las inquietudes descritas anteriormente fuese un tema de importancia e interés. Por esta razón se eligió realizar un programa orientado a la reducción de costos en redes de distribución de agua potable utilizando algoritmos genéticos.

### Redes de distribución de agua

Según la CONAGUA una red de distribución de agua potable: “es el conjunto de tuberías, accesorios y estructuras que conducen el agua desde tanques de servicio hasta las tomas domiciliarias o hidrantes públicos. Su finalidad es proporcionar agua a los usuarios para consumo doméstico, público, comercial, industrial y para condiciones extraordinarias como el extinguir incendios. La red debe proporcionar este servicio todo el tiempo, en cantidad suficiente, con la calidad requerida y a una presión adecuada. Los límites de calidad del agua, para que pueda ser considerada como potable se establecen en la Norma Oficial Mexicana NOM-127-SSA1 vigente” (Normas Técnicas, 2008).

Para que esta definición se cumpla, es necesario determinar el tamaño de los componentes de la red. Por tanto, determinar el diámetro ideal de las tuberías que componen cualquier sistema de distribución de agua no es una cuestión directa y determinista, de la misma forma en que no lo es los elementos que la componen.

Por lo tanto, el problema del diseño óptimo de la red de agua no se puede resolver de manera completamente satisfactoria, y esto es en parte debido a la gran diversidad de factores que afecta al problema. Por esta razón los modelos de diseño



no se han introducido aún con la misma fuerza que los modelos de análisis, debido probablemente a limitaciones como:

- La cantidad de modelos de demanda que pueden manejar.
- El número de combinaciones de diámetros que hay que considerar.
- El tamaño de la red (las cuales pueden ser muy extensas).
- La incorporación o no de una aproximación al diseño fiable de la red.
- La dificultad para entender la metodología matemática en la cual se basan.

## Algoritmos genéticos

“Los algoritmos genéticos surgen por primera vez de la tesis de J. D. Bagley en 1967. Dicha tesis influyó decisivamente en J. H. Holland, quien se puede considerar como el pionero de los algoritmos genéticos.” (Ponce Cruz, 2010). Este tipo de algoritmos está basado en la teoría de la evolución que propuso Charles Darwin (Darwin, 1859), y pueden definirse como: “algoritmos de búsqueda basados en la mecánica de selección natural y de la genética natural. Combinan la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado, aunque aleatorizado, para constituir así un algoritmo de búsqueda que tenga algo de las genialidades de las búsquedas humanas” (Goldberg, 1989).

Actualmente esta técnica evolutiva tiene varias aplicaciones en diferentes campos de las ingenierías, el interés de la presente tesis radica en acercar a los ingenieros civiles a esta técnica moderna que por lo general carece de material de referencia básico en español, se busca también el motivar futuras investigaciones en el área de hidráulica y sobre todo demostrar el beneficio que se obtiene al poder optimizar problemas cuyos espacios de solución son tan grandes que resulten imposibles de resolver de una manera convencional por métodos tradicionales o problemas de naturaleza compleja, como tratar con datos que muestran gran incertidumbre y poder transformarlos en modelos matemáticos, que sean capaces de predecir la vida útil de una estructura que esté en entornos agresivos que afecten su integridad estructural, problemas que como ingenieros civiles tratamos cada día.

Con este programa computacional se pretenden resolver problemas de redes hidráulicas haciendo uso de técnicas evolutivas que permitan soluciones más rápidas y eficientes que un experto. El tipo de soluciones buscadas deben tener errores mínimos comparados con los del experto, cumplir con los requisitos normativos correspondientes y tener tiempo para procesamientos eficientes.

Finalmente, el programa computacional utilizando algoritmos con técnicas evolutivas se pondrá a disposición de la comunidad académica para ser utilizado de manera ilustrativa en el área de hidráulica.



## OBJETIVOS

Los objetivos planteados para completar este trabajo son los siguientes:

Aprender, entender y poder aplicar un lenguaje de programación (**Python**) que me era desconocido. Así como saber complementar este lenguaje con el uso de herramientas de interfaces gráficas como lo es **Qt Designer**. Los conocimientos del autor antes de la creación de este programa eran básicos, tales como condicionales, bucles, declaración de variables y funciones, todos estos aplicados a **Matlab** el cual es un programa que requiere licencia.

1. Utilizar los algoritmos genéticos como una herramienta útil para resolver problemas de optimización, donde el espacio de búsqueda es demasiado grande.
2. Poder contribuir a la sociedad desarrollando un programa computacional que aplique las técnicas de análisis de redes hidráulicas y los algoritmos genéticos en el ámbito de la ingeniería civil, particularmente en el área de la hidráulica.
3. Que los usuarios finales, es decir, los estudiantes, puedan utilizarlo sin la necesidad de invertir grandes cantidades de tiempo en el aprendizaje del programa y sin miedo a que el programa devuelva respuestas extrañas que pueda confundirlos.
4. Acercar y motivar a los estudiantes de ingeniería civil al aprendizaje de técnicas más modernas como son los algoritmos genéticos aplicados a problemas de hidráulica de tuberías.
5. Automatizar el cálculo del diámetro de tuberías de una red de distribución de agua potable, utilizando los algoritmos genéticos como método para obtener los diámetros más económicos que cumplan con la velocidad y presión dadas por el usuario.
6. Servir como un complemento de **Epanet 2** para los fines citados en el punto anteriormente mencionado, pues también se trabaja con archivos de extensión **\*.inp**.



## HIPÓTESIS.

A través de una aplicación utilizando Algoritmos Genéticos programada en Python y Qt Designer, se desarrolla un programa para determinar los diámetros más adecuados de tuberías en redes para que al realizar un análisis hidráulico se cumpla con las siguientes características:

1. El costo la red sea el mínimo posible.
2. Se cumpla con las normas de velocidad y presión dadas por el usuario.
3. Los resultados dados por el programa deben ser corroborados usando **Epanet 2**, con un margen de error menor al 1% de los datos buscados.
4. Usar el programa computacional no requiere un conocimiento profundo de programación o algoritmos genéticos.

# CAPITULO I

## RED DE DISTRIBUCIÓN DE AGUA POTABLE

### I. RED DE DISTRIBUCIÓN DE AGUA POTABLE

Una red de distribución es una colección de tuberías, accesorios y estructuras que transportan agua desde los tanques de agua de servicio hasta las tomas de agua domésticas o las bocas de incendio públicas. Su finalidad es proporcionar a los usuarios agua para uso doméstico, público, comercial, industrial y en condiciones especiales (como extinción de incendios). (CONAGUA, 2007).

#### 1.1 Tipología

Las redes hidráulicas se dividen en dos grandes grupos, redes abiertas y redes cerradas. A continuación, se hace una breve descripción de cada uno de estos grupos explicando tanto sus ventajas como sus desventajas.

##### 1.1.1 Redes abiertas

Las redes abiertas o ramificadas se caracterizan por no tener ningún “circuito cerrado” en el sistema (Saldarriaga, 1998).

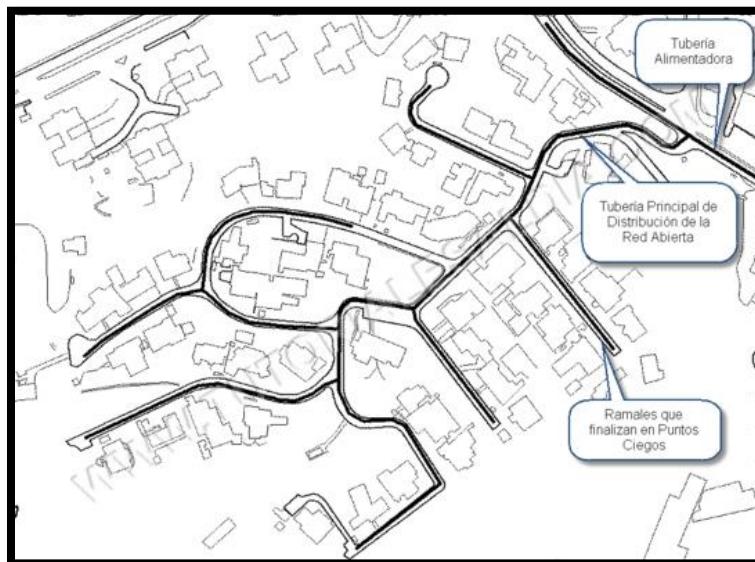


Figura 1. Red abierta. ([www.tutorialesaldia.com](http://www.tutorialesaldia.com), s.f.)

La desventaja de este tipo de red es que, ante la falla o rotura de alguna de sus tuberías, para poder repararlo, se tendrá que dejar sin servicio a los usuarios que son atendidos por las tuberías que están debajo de la rotura, mientras se realicen los trabajos de reparación.

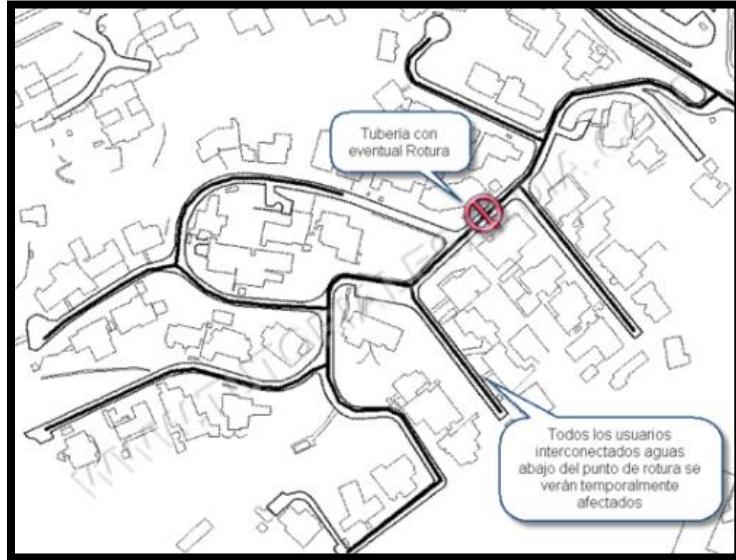


Figura 2. Desventaja de la red abierta ([www.tutorialesaldia.com](http://www.tutorialesaldia.com), s.f.)

La ventaja de este tipo de red es en cuanto al cálculo ya que su resolución es directa y no se toma en cuenta ciertas condiciones hidráulicas como el cálculo de las pérdidas de cada tubería, para los caudales en tránsito ([www.tutorialesaldia.com](http://www.tutorialesaldia.com), s.f.).

### 1.1.2 Redes cerradas

Conocidas también como sistemas con circuitos cerrados o ciclos. Su característica principal es tener algún tipo de circuito cerrado en el sistema (Saldarriaga, 1998).

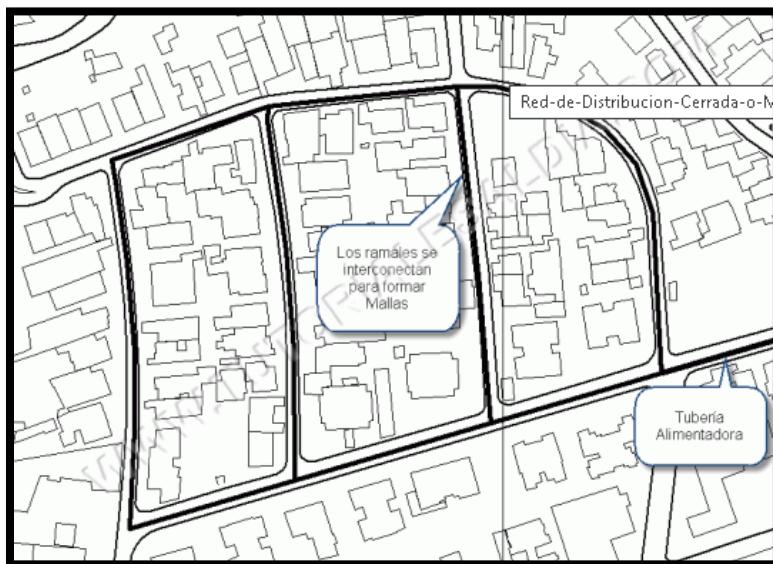


Figura 3. Red cerrada ([www.tutorialesaldia.com](http://www.tutorialesaldia.com), s.f.)

La desventaja de este tipo de red es a nivel de cálculo ya que se tiene que tomar en cuenta las condiciones hidráulicas tales como caudales en tránsito, presión en los nodos y pérdidas de carga, dada esta complejidad se tiene que recurrir a métodos iterativos como el método de Cross para su resolución y esto se complica aún más de acuerdo con la cantidad de tuberías que se tiene en la red y hacerlo de manera manual tiene ciertos inconvenientes.

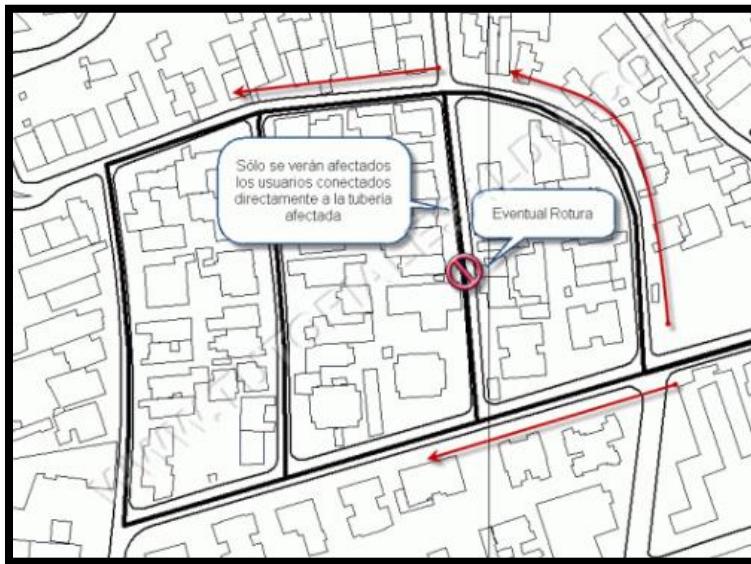


Figura 4. Desventaja de la red cerrada  
([www.tutorialesaldia.com](http://www.tutorialesaldia.com), s.f.)

La ventaja de este tipo de red es del punto de vista de la eficiencia y garantía del servicio, ya que se establecen rutas alternas para el flujo del agua.

Por lo tanto, en la mayoría de los desarrollos urbanos, la opción es establecer una red de distribución de agua potable cerrada, aunque esto puede significar la instalación de un mayor número de tuberías y el correspondiente aumento de los costos. En estos casos, los estándares operativos o de confiabilidad tienen prioridad sobre los estándares económicos.

## 1.2 Consideraciones según las normas

Para que la red sea adecuada, se deben considerar las siguientes restricciones:

Para evitar la sedimentación de partículas, la velocidad mínima del flujo de agua es de 0,5 m/s (Normas Técnicas, 2008).

La velocidad máxima permitida para evitar la corrosión de la tubería es la siguiente (el agua se considera limpia o ligeramente turbia):



Material de la tubería	Velocidad máxima (m/s)
Concreto simple hasta 0.45 m de diámetro	3.0
Concreto reforzado de 0.60 m de diámetro o mayores	3.5
Asbesto-cemento	5.0
Acero galvanizado	5.0
Acero sin revestimiento	5.0
Acero con revestimientos	5.0
Polietileno de alta densidad	5.0
Plástico PVC	5.0

Tabla 1. Velocidades máximas permisibles (Normas Técnicas, 2008)

La presión estática en cualquier punto de la red no debe ser superior a 50 m. Bajo la condición de demanda máxima por hora, la presión dinámica no será inferior a 10 m.

### 1.3 Ecuaciones hidráulicas

#### Conservación de la masa.

$$\sum Q_{in} = \sum Q_{out} + Q_c \quad (1)$$

Donde:

$Q_{in}$ ,  $Q_{out}$  y  $Q_c$ , son los gastos de entrada, de salida y de consumo respectivamente.

#### Conservación de la energía

$$\sum_{j=1}^T \Delta H_j = 0 \quad \forall T \in M \quad (2)$$

Donde  $M$  es el número de mallas y  $T$  el número de tuberías que pertenece a esa malla.

#### Ecuación de Darcy-Weisbach

$$h_f = f \frac{lv^2}{2dg} \quad (3)$$

Donde:



$h_f$ , es la pérdida por fricción.

$f$ , es el factor de fricción.

$l$ , es la longitud de la tubería.

$v$ , es la velocidad del fluido en la tubería.

$d$ , diámetro de la tubería.

$g$ , aceleración de la gravedad.

## Área de la sección de la tubería

$$A = \frac{\pi d^2}{4} \quad (4)$$

Donde:

$d$ , es el diámetro de la tubería.

## Coeficiente de fricción de Reynolds

$$R_e = \frac{Vd}{\nu} \quad (5)$$

Donde:

$V$ , es la velocidad del fluido en la tubería.

$d$ , es el diámetro de la tubería.

$\nu$ , es la viscosidad cinemática del fluido.

## Rugosidad relativa

$$\epsilon = \frac{k_s}{d} \quad (6)$$

Donde:

$k_s$ , es la rugosidad absoluta de la tubería.

$d$ , es el diámetro de la tubería.

## Factor de fricción de Darcy

Si se cumple que  $R_e \leq 2200$  el factor de fricción se calcula usando la fórmula de Hagen - Poiseuille:

$$f = \frac{64}{R_e} \quad (7)$$

Donde:

$R_e$ , es el número de Reynolds.

Si  $R_e > 4000$  el factor de fricción se calcula usando la fórmula de Swamee – Jain:

$$f = \frac{0.25}{\left[ \log_{10} \left( \frac{k_s}{3.7d} + \frac{5.74}{R_e^{0.9}} \right) \right]^2} \quad (8)$$



Donde:

$f$ , es el factor de fricción.

$k_s$ , es la rugosidad absoluta de la tubería.

$d$ , es el diámetro de la tubería.

$R_e$ , es el número de Reynolds.

En caso de que  $2200 < R_e < 4000$  utilizamos la ecuación de Colebrook-White:

$$\frac{1}{\sqrt{f}} = -2 \log_{10} \left( \frac{k_s}{3.7d} + \frac{2.51}{R_e \sqrt{f}} \right) \quad (9)$$

Con el fin de averiguar el valor del factor de fricción  $f$  de Darcy en la ecuación no explícita de Colebrook-White hacemos el siguiente cambio de variable en la ecuación (9):

$$x = \frac{1}{\sqrt{f}}$$

Obtenemos  $g(x) - x = 0$ .

Aplicando el método de Newton-Raphson, obtenemos la siguiente fórmula para aproximar el valor de la raíz en la iteración  $i + 1$  con base en la aproximación de la iteración  $i$ :

$$x_{i+1} = x_i - \frac{g(x_i) - x_i}{g'(x_i) - 1}$$

O de forma más explícita:

$$x_{i+1} = x_i - \frac{-2 \log_{10} \left( \frac{k_s}{3.7d} + \frac{2.51}{R_e} x_i \right) - x_i}{-\frac{2}{\ln 10} \left( \frac{\frac{2.51}{R_e}}{\frac{k_s}{3.7d} + \frac{2.51}{R_e} x_i} \right) - 1} \quad (10)$$

Se procede de la siguiente manera:

- 1) Usamos dos valores semilla  $x_i = 0.0001$  y una tolerancia  $Tol = 0.0001$ .
- 2) Se calcula  $x_{i+1}$  con la fórmula (10).
- 3) Comparamos  $|x_{i+1} - x_i| \leq Tol$ .
- 4) Si el paso 3 no se cumple se hace  $x_i = x_{i+1}$  y se repiten los pasos 2 y 3.



- 5) Si el paso 3 se cumple se hace  $f = 1/x_{i+1}^2$ .

## Velocidad en una tubería

$$V = \frac{Q}{A} \quad (11)$$

Donde:

$Q$ , es el caudal que fluye en la tubería.

$A$ , es el área de la sección de la tubería.

## Sumatoria de pérdidas menores

$$\sum h_m = \left( \sum k_m \right) \frac{V^2}{2g} \quad (12)$$

Donde:

$k_m$ , es el coeficiente de pérdidas menores.

$V$ , es la velocidad del fluido en la tubería.

$g$ , aceleración de la gravedad.

## 1.4 Análisis hidráulico

El análisis hidráulico en régimen estacionario en una red de tuberías por donde circula el fluido incompresible tiene como objetivo estudiar el nivel de presión y el caudal circulante generado por el sistema en unas determinadas condiciones de contorno, que se mantienen estables en el tiempo.

Para establecer las ecuaciones del problema a resolver, se utilizan las ecuaciones de energía y continuidad, y estas ecuaciones deben ser satisfechas en todo el sistema, así como a la ecuación de Darcy-Weisbach, que rige la pérdida de carga en una tubería.

El objetivo que se pretende es el planteamiento de dichas ecuaciones en una forma sistemática, utilizando para ello una formulación matricial; de esta manera no solo se logra una descripción muy clara, sino que las fórmulas son muy adecuadas para aplicar procesos o algoritmos numéricos que pueden ser resueltos por una computadora. (Navarro, 1998).

### 1.4.1 Método del gradiente

Fue desarrollado por los profesores Ezio Todini y Enda P. O'Connell (Todini & Pilati, 1987) en la Universidad de Newcastle upon Tyne y por R. Salgado, como parte de su tesis doctoral en 1982-1983. Consiste en que las ecuaciones de energía individuales para cada tubo se combinan con las ecuaciones de masa individuales en cada unión, con el fin de obtener una solución simultánea tanto de los caudales en las tuberías, como en las alturas piezométricas en los nudos. Este método se basa en que, al tener siempre disponibilidad del fluido, se garantiza que se



cumplirán las ecuaciones de conservación de la masa en los nodos de la red y la de conservación de la energía en cada tubería. (Saldarriaga, 1998).

Este método se basa en las siguientes tres condiciones:

- 1) En cada nodo se debe cumplir la ecuación de continuidad:

$$\sum_{j=1}^{NT_i} Q_{ij} - Q_{Di} + Q_{ei} = 0 \quad (13)$$

Donde:

$Q_{Di}$ , son los caudales consumidos en cada uno de los nodos.

$Q_{ei}$ , son los caudales que alimentan la red de distribución.

$Q_{ij}$ , es el caudal que pasa por la tubería  $ij$  hacia el nodo  $i$  desde el nodo  $j$ .

- 2) Debe haber una relación no lineal entre las pérdidas por fricción y el caudal para cada uno de los tubos que conforman la red:

$$Q = -2 \frac{\sqrt{2gdh_f}}{\sqrt{l}} A \log_{10} \left( \frac{k_s}{3.7d} + \frac{2.51\nu\sqrt{l}}{\sqrt{2gd^3}\sqrt{h_f}} \right) \quad (14)$$

Donde:

$h_f$ , es la pérdida por fricción.

$g$ , es la aceleración de la gravedad.

$k_s$ , es la rugosidad absoluta de la tubería.

$d$ , es el diámetro de la tubería.

$l$ , es la longitud de la tubería.

$A$ , es el área de la sección transversal de la tubería.

$\nu$ , es la viscosidad cinemática del fluido.

- 3) En cada tubo la energía total disponible se gasta en pérdidas por fricción y en pérdidas menores:

$$H_t = h_f + \sum h_m \quad (15)$$

Donde:

$h_f$ , es la pérdida por fricción.

$\sum h_m$ , es la sumatoria de pérdidas menores.

En la ecuación (14) se ha utilizado la ecuación de Darcy-Weisbach junto con la ecuación de Colebrook-White, ya que, durante el proceso de diseño de redes, el programador no tiene control sobre el número de Reynolds lo cual invalida el uso de la ecuación de Hazen-Williams.



Si se consideran las pérdidas menores ocasionadas por cualquier tipo de accesorios y la posible presencia de bombas en algunas tuberías de la red, la fórmula (15) toma la siguiente forma general y es válida para todas las tuberías:

$$H_t = \alpha Q^n + \beta Q + \gamma \quad (16)$$

Donde:

$n$ , es el exponente que depende de la ecuación de fricción utilizada (en este caso la de Darcy-Weisbach,  $n = 2$ ).

Y  $\alpha, \beta, \gamma$ , son los parámetros característicos del tubo, las válvulas y las bombas respectivamente.

Si en una tubería solo ocurren pérdidas que son función de la altura de velocidad únicamente, la ecuación (15) se puede utilizar para establecer el valor de  $\alpha$ :

$$H_t = \alpha Q^n$$
$$h_f + \sum h_m = \alpha Q^n$$

Si se utiliza la ecuación de Darcy-Weisbach para describir las pérdidas por fricción, esta última ecuación se transforma en:

$$\left( f \frac{l}{d} + \sum k_m \right) \frac{Q^2}{2gA^2} = \alpha Q^2$$

Por consiguiente:

$$\alpha = \frac{\left( f \frac{l}{d} + \sum k_m \right)}{2gA^2} \quad (17)$$

De la ecuación (17) es importante destacar dos cosas. Que  $\alpha$  es un parámetro característico del tubo que incluye los factores de pérdidas por fricción y pérdidas menores. Por otro lado, en el caso de accesorios en la tubería, como algunos tipos de válvulas, el parámetro  $\alpha$  establece su relación con  $Q^n$  adicional a las relaciones para las pérdidas antes mencionadas. Para bombas colocadas en las tuberías se requieren los tres parámetros  $\alpha, \beta$  y  $\gamma$  ya que la relación entre la altura piezométrica suministrada por la bomba y el canal es polinomial.

En segundo lugar, el hecho de utilizar más de un término dentro de la ecuación que relaciona la energía total gastada en una tubería con el caudal que pasa por esta (ecuación 16) permite introducir cualquier tipo de accesorio o bomba.

Para el método de "gradiente hidráulico", se realizan las siguientes definiciones adicionales para describir la topología de la red en forma de matriz



$NT$  = número de tuberías en la red.

$NN$  = número de nodos con altura piezométrica desconocida.

$[A_{12}]$  = matriz de conectividad asociada con cada uno de los nodos de la red. Su dimensión es  $NT \times NN$  con sólo dos elementos diferentes de cero en la  $i$ -ésima fila:

- -1 en la columna correspondiente al nudo inicial del tramo  $i$ .
- 1 en la columna correspondiente al nudo final del tramo  $i$ .

$NS$  = número de nodos de altura piezométrica fija o conocida.

$[A_{10}]$  = matriz topológica tramo a nodo para los  $NS$  nodos de altura piezométrica fija. Su dimensión es  $NT \times NS$  con un valor igual a -1 en las filas correspondientes a los tramos conectados a nodos de altura piezométrica fija.

$[A_{11}]$  = matriz diagonal de  $NT \times NT$  definida como sigue:

$$[A_{11}] = \begin{bmatrix} \alpha_1 Q_1^{(n1-1)} + \beta_1 + \frac{\gamma_1}{Q_1} & 0 & 0 & \dots & 0 \\ 0 & \alpha_2 Q_2^{(n2-1)} + \beta_2 + \frac{\gamma_2}{Q_2} & 0 & \dots & 0 \\ 0 & 0 & \alpha_3 Q_3^{(n3-1)} + \beta_3 + \frac{\gamma_3}{Q_3} & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \alpha_{NT} Q_{NT}^{(nNT-1)} + \beta_{NT} + \frac{\gamma_{NT}}{Q_{NT}} \end{bmatrix}$$

$[Q]$  = vector de caudales con dimensión  $NT \times 1$ .

$[H]$  = vector de alturas piezométricas desconocidas con dimensión  $NN \times 1$ .

$[H_0]$  = vector de alturas piezométricas fijas con dimensión  $NS \times 1$ .

$[A_{21}]$  = matriz transpuesta de  $[A_{12}]$ .

$[q]$  = vector de consumo (demanda) o de entrada (oferta) en cada nodo de la red, con dimensión  $NN \times 1$ .

$[N]$  = matriz diagonal  $(n_1, n_2, \dots, n_{NT})$  con dimensión  $NT \times NT$ .

$[A_{11}]'$  = matriz con dimensión  $NT \times NT$  definida como:

$$[A_{11}]' = \begin{bmatrix} \alpha_1 Q_1^{(n1-1)} & 0 & 0 & \dots & 0 \\ 0 & \alpha_2 Q_2^{(n2-1)} & 0 & \dots & 0 \\ 0 & 0 & \alpha_3 Q_3^{(n3-1)} & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \alpha_{NT} Q_{NT}^{(nNT-1)} \end{bmatrix}$$

$[I]$  = matriz identidad con dimensión  $NT \times NT$ .

El método del Gradiente Hidráulico consiste en solucionar iterativamente los siguientes sistemas de ecuaciones:



$$\begin{aligned}[H_{i+1}] = & -\{[A21]([N][A11])^{-1}[A12]\}^{-1} \\ & * \{[A21]([N][A11])^{-1}([A11][Q_i] + [A10][H_0]) \\ & - ([A21][Q_i] - [q])\}\end{aligned}\quad (18)$$

$$\begin{aligned}[Q_{i+1}] = & \{[I] - ([N][A11])^{-1} - [A11]\}[Q_i] \\ & - \{([N][A11])^{-1} * ([A12][H_{i+1}] + [A10][H_0])\}\end{aligned}\quad (19)$$

El paso más complicado en cada iteración es la solución del sistema representado por la ecuación (18), ya que se trata de un sistema de ecuaciones disperso, simétrico, lineal y definido positivamente con  $NN$  incógnitas: las alturas piezométricas desconocidas  $[H_{i+1}]$  en los nodos de la red. Una vez se conoce este vector es posible calcular  $[Q_{i+1}]$  en forma explícita, mediante la ecuación (19) (Saldarriaga, 1998).



## CAPITULO II

# ALGORITMOS GENÉTICOS

## II. INTRODUCCIÓN A LOS ALGORITMOS GENÉTICOS

### 2.1 ¿Qué es un algoritmo genético?

“Los algoritmos genéticos (AG) son algoritmos de optimización paralelos sin gradiente que usan un criterio de desempeño para la evaluación y una población de posibles soluciones a la búsqueda de un óptimo global” (Chambers, 2001). Este tipo de algoritmos parte de una población inicial de posibles soluciones generadas al azar. Estos individuos buscan imitar el comportamiento biológico de las especies, es decir “evolucionarán tomando como base los esquemas propuestos por Darwin sobre la selección natural, y se adaptarán en mayor medida tras el paso de cada generación a la solución requerida” (Gestal, Rivero, Rabuñal, Dorado, & Pazos, 2010).

Para llevar a cabo la tarea de encontrar el óptimo global a través de un AG se deben establecer las siguientes correspondencias:

- Se debe codificar de manera adecuada cada posible solución de la misma manera que un cromosoma representa únicamente a los individuos de una especie.
- Siempre debe haber una población inicial.
- Tiene que existir una función de evaluación que mida la calidad del cromosoma.
- Debe haber un criterio de selección que obligue a que solo los cromosomas más aptos puedan sobrevivir y tener descendencia de manera semejante que los individuos de una población compiten en la naturaleza.
- Se definirán operadores genéticos de cruce y mutación que operen sobre uno o varios cromosomas. Estos operadores juegan un papel similar en el proceso evolutivo biológico natural.

### 2.2 Algoritmo genético

Como se verá en las siguientes secciones para poder realizar el algoritmo se requerirá de una población inicial generada aleatoriamente, que estará constituida de individuos. Estos individuos necesitan poder codificarse al problema de una forma que resulte adecuada al mismo. Además, durante la ejecución del algoritmo se requiere de una función que asigne a cada individuo codificado un número real para poder evaluarlo. Una vez evaluados, si se desea optar por una estrategia elitista los mejores individuos se copian en una población temporal para evitar perderlos. Los mejores individuos a los cuales llamaremos padres tendrán derecho a reproducirse (o copiarse) generando dos hijos sobre los cuales actuará el operador de mutación.



El proceso finaliza cuando se alcanzan algunos de los siguientes puntos:

- Los mejores individuos de la población tienen soluciones suficientemente buenas para el problema que se quiere resolver.
- Se ha alcanzado el número máximo de generaciones.
- La población converge.

En la figura 5, se presenta un pseudocódigo, donde puede observarse la estructura básica de un Algoritmo Genético Canónico:

La versión *simple* o *canónica* del algoritmo genético trabaja siguiendo los siguientes pasos:

1. Generar una población inicial de soluciones.
2. Seleccionar, de la población actual, las soluciones mejor adaptadas.
3. Cruzar algunas soluciones para obtener su descendencia.
4. Mutar algunas soluciones para obtener las soluciones mutadas.
5. Elegir las soluciones que sobreviven y formarán la nueva generación.
6. Si no se alcanza el criterio de parada volver al paso 2.

Al finalizar los pasos anteriores, la mejor solución de la población es la que se propone como solución del problema.

**Figura 5. Algoritmo Genético Canónico (Melián Batista, Moreno Pérez, & Moreno Vega, 2009)**

## 2.3 Codificación

Según (Ponce Cruz, 2010) “Los algoritmos genéticos parten de la premisa de emplear la evolución natural como un procedimiento de optimización que se caracteriza por tener operaciones básicas que son:

- Selección.
- Cruzamiento.
- Mutación.

Para poder evaluar las operaciones antes mencionadas es necesario que la información a optimizar se encuentre en cadena de bits {0,1}, y que esta representación sea una cadena finita siendo cada cadena un individuo que conforma una población”. A esta cadena de bits se le llama cromosoma y a los ceros (o unos) se les llama genes los cuales corresponden a los parámetros del problema a resolver. Si bien el uso de {0,1} para representar los genes no es una obligación, la teoría en la que se fundamentan los Algoritmos Genéticos, utiliza dicha nomenclatura desde los primeros trabajos de John Holland.

Para la codificación, se asigna un cierto número de bits a cada parámetro y se discretizan las variables representadas por cada gen. Esta codificación no debe ser hecha de forma trivial ya que es uno de los factores más determinantes a la hora de

que el AG funcione de manera correcta. Este es un ejemplo de un individuo binario que codifica 3 parámetros:

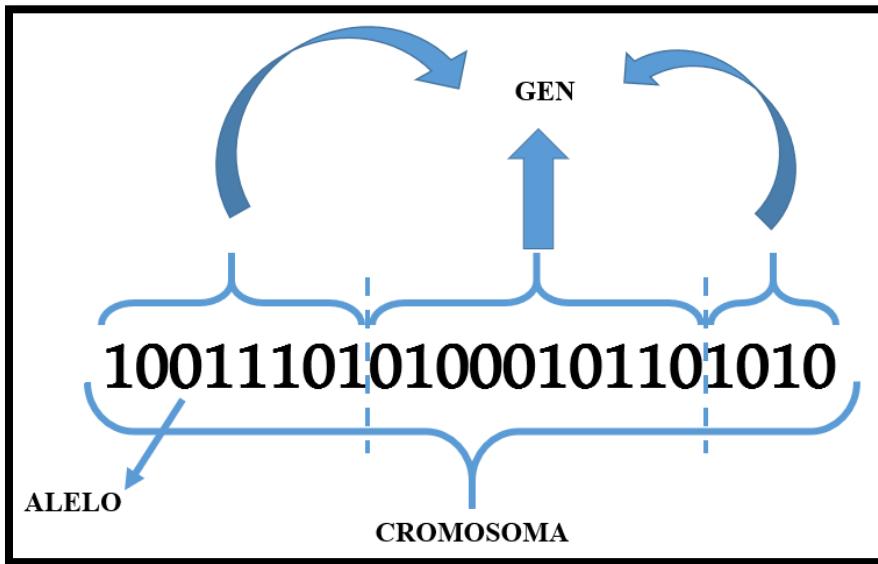


Figura 6. Individuo Binario

Otras maneras de representar variables serían:

- En cadenas formadas por un valor entero real (como se hará en la presente tesis) o en punto flotante.
- En cadenas formadas por letras.

## 2.4 Población

### 2.4.1 Tamaño de población

El tamaño de la población es una variable que hay que considerar con mucho cuidado pues una población demasiado pequeña puede correr el riesgo de no cubrir toda el área de soluciones y quedar demasiado limitada o incluso estancada, mientras que una población demasiado grande puede ser demasiado costosa computacionalmente.

### 2.4.2 Población inicial

La población inicial debe generarse de manera aleatoria y debe estar dotada de individuos diversos para evitar caer en óptimos locales o tener una convergencia demasiado prematura.

Una técnica usada frecuentemente consiste en partir de un individuo particular y a partir de este crear los demás individuos copiándolo y aplicando pequeñas



mutaciones sobre el mismo individuo. Esta técnica es útil cuando se tiene un individuo que se acerca a la solución, pero se quiere seguir mejorándolo.

## 2.5 Operadores genéticos

Durante el proceso de ejecución del AG para pasar de una generación a otra se necesitan ciertos operadores para mantener la diversidad genética de la población. Cada uno de estos operadores genéticos tiene como objetivo crear nuevos individuos. Estos individuos pueden mejorar o afectar a la población de la nueva generación, por esta razón debe tenerse cuidado en que porcentaje se emplearán.

A continuación, se describen los operadores genéticos con más detalle.

### 2.5.1 Selección

Después de que los individuos han sido evaluados, este operador tiene como principal tarea seleccionar a los mejores individuos de la población. Por lo tanto, la selección de un individuo está relacionada con su evaluación. Los más aptos tienen mayor probabilidad de reproducirse (de manera semejante a como ocurre en la naturaleza, donde solo los más fuertes sobreviven) pero sin dejar de lado a los menos aptos, ya que de lo contrario la población se volvería homogénea.

#### 2.5.1.1 Selección por Ruleta

Uno de los métodos más usados es el de **Selección por Ruleta** también llamado, **Muestreo Estocástico con Reemplazo**. Este método coloca a todos los individuos en una ruleta con porcentajes en relación con su fitness y a su fitness relativo, es decir a su fitness dividido por la suma del fitness de todos los individuos de la población, y girar la ruleta tantas veces como individuos se tengan.

El procedimiento es el siguiente:

1. Calcular el valor del fitness  $fitness(I_i)$  para cada individuo  $I_i$  de la población.
2. Calcular la sumatoria de todos los fitness,  $F_t = \sum_{i=1}^n fitness(I_i)$ .
3. Calcular la probabilidad  $P_i$ , para cada individuo:

$$P_i = \frac{fitness(I_i)}{F_t}, \text{ donde } i = 1, 2, \dots, I$$

4. Calcular la suma de las probabilidades:

$$q_i = \sum_{I=1}^i P_I$$

Para seleccionar un elemento se procede de la siguiente manera el número de veces como individuos se tengan:

1. Se genera un número al azar  $r$ , en un rango  $[0,1]$ .
2. Seleccionar el  $i$ -ésimo individuo  $I_i$ , tal que  $q_{i-1} < r \leq q_i$ .

#### 2.5.1.2 Selección por torneo determinístico

La idea principal de este método consiste en comparar de manera directa los fitness de los individuos.

En esta versión determinística, se escogen al azar un número  $p$  de individuos (generalmente  $p = 2$ ) y se comparan sus fitness. De los individuos seleccionados el que tenga un mejor fitness pasa a la siguiente generación.

### 2.5.1.3 Selección por torneo probabilístico

Esta versión solo tiene diferencia con la anterior en relación con cómo se selecciona al ganador. En lugar de escoger al ganador por comparación directa de sus fitness, se genera un número aleatorio en el rango  $[0,1]$ , si es mayor que un parámetro  $p$  (fijado para todo el proceso evolutivo) se escoge el individuo óptimo y en caso contrario al menos óptimo.

### 2.5.2 Cruce

El cruce o crossover, es el operador genético más importante debido a que a través de este se pueden generar nuevas soluciones ampliando el espacio de búsqueda hasta llegar a un óptimo.

El objetivo de este operador genético es que a partir de dos individuos ya existentes (**Padres**), se generen nuevos individuos (**Hijos**), que combinen las características de los anteriores heredando soluciones mejores que cada uno de los padres por separado. En caso de que este cruce no llegara a heredar en alguno de los hijos mejores soluciones, no significa que se haya dado un paso atrás en la búsqueda de la solución deseada ya que en posteriores cruces podrían recuperarse estos padres incluso aunque se tengan leves mutaciones.

#### 2.5.2.1 Cruce de un punto

Es la más simple de todas las técnicas de cruce y de la cual se derivan las demás. Una vez seleccionados dos individuos se cortan sus cromosomas por un punto al azar para formar dos subcadenas en cada una, una a la izquierda del punto de corte y otra a la derecha. Se hace un cambio en las colas de los individuos para generar nuevos descendientes, ver figura 7.

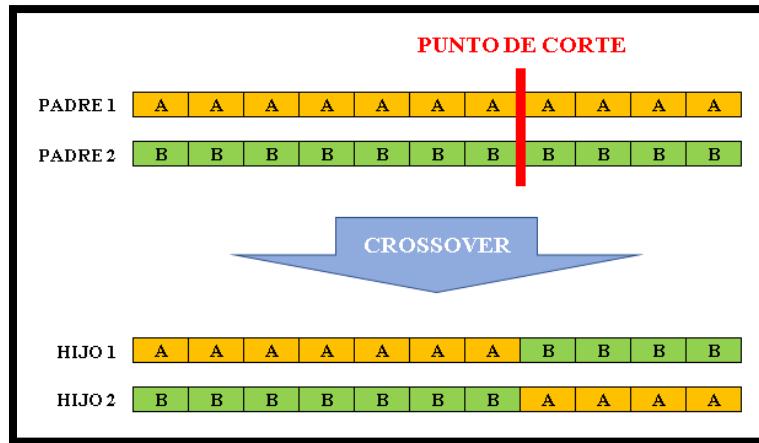


Figura 7. Cruce de un punto

### 2.5.2.2 Cruce multipunto

Si generalizamos el cruce de un punto, podemos llegar a algoritmos de cruce multipunto. Los puntos de corte se generarían al azar y se mezclaría información de los padres intercambiando el valor de sus respectivas subcadenas. A continuación, en la figura 8, se muestra un cruce multipunto para 3 puntos.

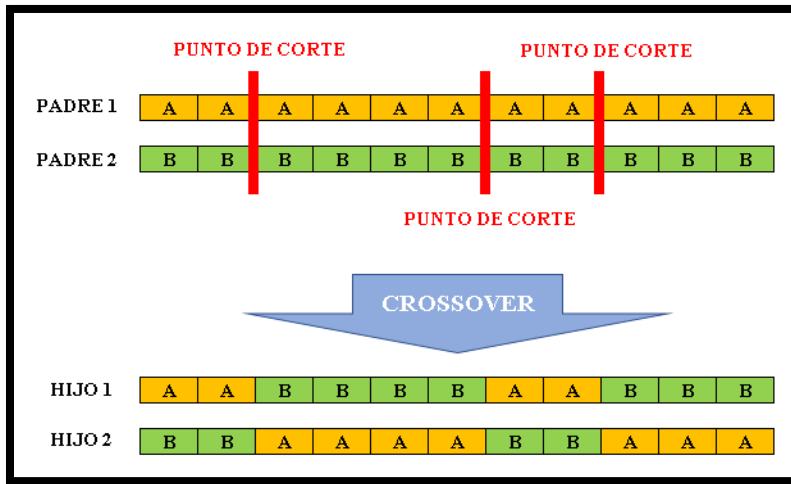


Figura 8. Cruce de 3 puntos

(De Jong, 1975) Estudió el problema de si era recomendable añadir un número de puntos mayor a 2 al operador de cruce y concluyó que esto reduce el rendimiento del algoritmo genético. Si aumentamos el número de puntos hará que los segmentos originados sean corrompibles, en otras palabras, perderían las características ganadas por sus padres, convirtiendo al algoritmo genético en una búsqueda al azar simple.

### 2.5.3 Mutación

De la misma forma a como ocurre en la naturaleza, la descendencia de los padres no siempre es perfecta y siempre viene acompañada de alguna especie de error, por lo general sin ser grave. La mutación es un operador cuya función principal es la de producir pequeños cambios en la población, haciendo que varíe su valor de forma aleatoria.

La mutación es controlada por el factor  $P_m$  al que llamamos tasa de mutación. La mutación trabaja de manera conjunta con el operador de cruce, si el cruce tiene éxito entonces uno, o ambos de los descendientes se muta con cierta probabilidad  $P_m \leq 5\%$ .

Por cada bit (o alelo) de los individuos de nuestra población actual generamos un número  $\rho$  al azar en el rango  $[0,1]$ . Si  $\rho < P_m$ , entonces se muta ese bit y se invierte su valor, en otras palabras, si ese bit era 0 ahora vale 1 y viceversa.



Si la codificación del gen no es binaria, el operador de mutación deberá modificarse para adaptarlo a la codificación en cuestión.

El uso de este operador radica en que al aplicarlo dejamos fuera la posibilidad de que ningún espacio de búsqueda tenga una probabilidad nula de ser examinado.

#### 2.5.4 Elitismo

En este caso la idea es muy simple, se trata de escoger a los mejores padres evaluados, es decir a los que ofrezcan valores más altos y copiarlos a la siguiente generación. De esta forma se puede tener la seguridad de que el proceso evolutivo no dará un paso atrás, pero debe tenerse especial cuidado pues esto pudiera ocasionar una convergencia rápida y caer en un mínimo local.

Una forma segura de aplicar este operador consiste en que al pasar una generación no se haya mejorado la población mediante los operadores de cruce o mutación, entonces se copien a los mejores o al mejor individuo a la generación actual para aumentar su calidad.

### 2.6 Función de evaluación (Fitness)

Para el correcto funcionamiento de un algoritmo genético algo que resulta de vital importancia es el poder evaluar cada uno de sus individuos.

Dado un individuo, la función de evaluación consiste en asignarle un valor numérico de adaptación, el cual se supone sea proporcional a la utilidad o habilidad del individuo representado, esto puede considerarse como la probabilidad de que ese individuo pueda sobrevivir en el ecosistema del problema hasta que tenga la capacidad de reproducirse y se reproduzca.

Otra característica que debe tener esta función es la de castigar las malas soluciones y premiar las buenas soluciones para que esta última sea una solución que se propague más rápido.

Esta función de evaluación es única para cada tipo de problema, pues estará en función de la codificación de los individuos del problema, la cual también es única. Para nuestro caso, se pretende optimizar la función que se muestra a continuación:

$$fitness = Cc(d, L) * (1 + P(R)) \quad (20)$$

Donde:

$Cc(d, L)$ , es la función de costo.

$P(R)$ , es la función de penalización.

#### 2.6.1 Restricciones hidráulicas

##### 2.6.1.1 Velocidades máximas y mínimas

$$R_1 = 10 * \min(V_{\text{min\_red}} - V_{\text{min\_norma}}, 0) \quad (21)$$



$$R_2 = 10 * \left\| \min(V_{\max\_norma} - V_{\max\_red}, 0) \right\| \quad (22)$$

Donde:

$V_{\min\_red}$ , es la velocidad más pequeña en una tubería de la red.

$V_{\max\_red}$ , es la velocidad más grande en una tubería de la red.

$V_{\min\_norma}$ , es la velocidad mínima permitida dada por el usuario en cada tubería.

$V_{\max\_norma}$ , es la velocidad máxima permitida dada por el usuario en cada tubería.

### 2.6.1.2 Presiones máximas y mínimas

$$R_3 = \left\| \min(H_{\min\_red} - H_{\min\_norma}, 0) \right\| \quad (23)$$

$$R_4 = \left\| \min(H_{\max\_norma} - H_{\max\_red}, 0) \right\| \quad (24)$$

Donde:

$H_{\min\_red}$ , es la altura piezométrica más pequeña en un nodo de la red.

$H_{\max\_red}$ , es la altura piezométrica más grande en un nodo de la red.

$H_{\min\_norma}$ , es la altura piezométrica mínima permitida dada por el usuario en cada nodo.

$H_{\max\_norma}$ , es la altura piezométrica máxima permitida dada por el usuario en cada nodo.

### 2.6.2 Función de penalización

La función de penalización no es más que la sumatoria de las restricciones:

$$P(R) = \sum_{i=1}^r R_i \quad (25)$$

### 2.6.3 Función de costos

En el diseño de redes de distribución de agua potable las variables que se tienen que combinar, para encontrar la solución más económica son los diámetros comerciales de cada una de las tuberías de la red que se está analizando. Estos diámetros, al ser discretos, nos obligan a crear una función de costos basada en regresiones. En el programada usamos tres tipos de regresión: lineal, potencial o exponencial. El usuario debe usar la que mejor coeficiente de correlación cuadrático  $R^2$  tenga (valor más cercano a 1).

#### 2.6.3.1 Regresión Lineal.

$$K_1 = \frac{\sum x_i * \sum y_i - N * \sum x_i y_i}{(\sum x_i)^2 - N * \sum x_i^2} \quad (26)$$



$$K_2 = \bar{Y} - K_1 * \bar{X} \quad (27)$$

Donde:

$x_i$ , diámetro i-ésimo de la lista a considerar en la red.

$y_i$ , costo de construcción por metro lineal del diámetro i-ésimo.

$\bar{X}$ , promedio de la lista de diámetros.

$\bar{Y}$ , promedio de la lista de costo de construcción por metro lineal del diámetro.

Para este tipo de regresión la función de costos queda de la siguiente forma:

$$Cc(lineal) = \sum_{i=1}^{NT} L_i * [K_1 * d_i + K_2] \quad (28)$$

Donde:

$NT$ , número de tuberías en la red.

$L_i$ , longitud de la i-ésima tubería de la red.

$d_i$ , diámetro de la i-ésima tubería de la red.

$K_1, K_2$ , coeficientes de regresión lineal.

### 2.6.3.2 Regresión Potencial.

$$K_2 = \frac{\sum x'_i * \sum y'_i - N * \sum x'_i y'_i}{(\sum x'_i)^2 - N * \sum x'^2_i} \quad (29)$$

$$K_1 = 10^{(\bar{Y}' - K_2 * \bar{X}')} \quad (30)$$

Donde:

$x'_i$ , logaritmo base 10 del diámetro i-ésimo de la lista a considerar en la red.

$y'_i$ , logaritmo base 10 del costo de construcción por metro lineal del diámetro i-ésimo.

$\bar{X}'$ , promedio de la lista de los logaritmos base 10 de los diámetros.

$\bar{Y}'$ , promedio de la lista de los logaritmos base 10 de los costos de construcción por metro lineal del diámetro.

Para este tipo de regresión la función de costos queda de la siguiente forma:

$$Cc(potencial) = \sum_{i=1}^{NT} L_i * [K_1 * d_i^{(K_2)}] \quad (31)$$



### 2.6.3.3 Regresión Exponencial.

$$K_2 = \frac{\sum x_i * \sum y'_i - N * \sum x_i y'_i}{(\sum x_i)^2 - N * \sum x_i^2} \quad (32)$$

$$K_1 = 10^{(\bar{Y}' - K_2 * \bar{X})} \quad (33)$$

Donde:

$x_i$ , diámetro i-ésimo de la lista a considerar en la red.

$y'_i$ , logaritmo base 10 del costo de construcción por metro lineal del diámetro i-ésimo.

$\bar{X}$ , promedio de la lista de diámetros.

$\bar{Y}'$ , promedio de la lista de los logaritmos base 10 de los costos de construcción por metro lineal del diámetro.

Para este tipo de regresión la función de costos queda de la siguiente forma:

$$Cc(exponencial) = \sum_{i=1}^{NT} L_i * [K_1 * 10^{(K_2 * d_i)}] \quad (34)$$

### 2.6.3.4 Coeficiente de determinación.

$$R^2 = 1 - \frac{\sigma_r^2}{\sigma_p^2} \quad (35)$$

Donde:

$\sigma_r^2$ , varianza residual.

$\sigma_p^2$ , varianza de la variable dependiente.



## CAPITULO III

# ESTRUCTURA DEL PROGRAMA

### III. ESTRUCTURA DEL CÓDIGO

En este capítulo se hará una explicación detallada de cómo está conformado el código utilizado en el programa que se desarrolló.

En los anexos se presenta el código completo para una visión general del mismo.

#### 3.1 Python como lenguaje a usar

##### 3.1.1 ¿Qué es Python?

Python nació a principios de la década de 1990 y fue desarrollado originalmente por la afición del ingeniero holandés Guido Van Rossum.

Python es un lenguaje de programación de alto nivel que se utiliza en muchas aplicaciones y sistemas operativos. Tiene una curva de aprendizaje moderada y su filosofía hace hincapié en proporcionar una sintaxis de código claro y legible.

Es un lenguaje de programación de uso general, multiparadigma, debido a que soporta orientación a objetos, programación imperativa y en menor medida programación funcional. Es interpretado de tipado dinámico y multiplataforma. Una de las razones de su éxito es que tiene una licencia de código abierto y se puede utilizar en cualquier entorno. Esto lo convierte en uno de los primeros lenguajes de muchos programadores, y por lo tanto se enseña en escuelas y universidades de todo el mundo.

A nivel científico, cuenta con una enorme biblioteca de recursos, con un enfoque particular en matemáticas, con el fin de atraer a programadores interesados en campos profesionales.

##### 3.1.2 ¿Por qué Python?

A continuación, se explican las ventajas de usar Python:

- **Simplificado y rápido:** este lenguaje simplifica enormemente la programación y es un excelente lenguaje para la creación de scripts.
- **Elegante y flexible:** el lenguaje ofrece muchas comodidades a los programadores porque es fácil de leer y explicar.
- **Programación eficiente:** fácil de aprender, con una curva de aprendizaje moderada. Es muy fácil comenzar a programar y puede mejorar la eficiencia del trabajo.
- **Limpieza y orden:** es legible y sus módulos están bien organizados.
- **Portátil:** este es un lenguaje muy portátil. Hoy, podemos usarlo en casi cualquier sistema.



- **Numerosos complementos:** Al igual que los lenguajes como JavaScript, tiene bibliotecas y frameworks de Python que se adaptan a las necesidades específicas de codificación.
- **Comunidad:** Tiene una gran cantidad de usuarios. Su comunidad participa activamente en el desarrollo del lenguaje.

### 3.1.3 Bibliotecas utilizadas

#### 3.1.3.1 Biblioteca: math

Python proporciona una biblioteca matemática como parte de su "biblioteca estándar" y proporciona funciones matemáticas para el campo de números reales. Algunas de las características proporcionadas incluyen:

- Funciones numéricas
  - **math.gcd(a, b):** retorna el máximo común divisor de los números a y b.
  - **math.ceil(x):** retorna el entero más cercano mayor o igual que x.
- Funciones de potencia y logarítmicas
  - **math.exp(x):** retorna  $e^x$ .
  - **math.log10(x):** dado un número x retorna su logaritmo en base 10.
  - **math.pow(x, y):** retorna  $x^y$ .
  - **math.sqrt(x):** retorna la raíz cuadrada de un número x.
- Funciones trigonométricas y transformación de ángulos
  - **math.cos(x):** retorna el coseno de x.
  - **math.sin(x):** retorna el seno de x.
  - **math.degrees(x):** transforma un ángulo en grados a radianes.
  - **math.radians(x):** transforma un ángulo en radianes a grados.
- Constantes ( $\pi$ , e, etc.)

#### 3.1.3.2 Biblioteca: random

Python también proporciona la biblioteca aleatoria. Proporciona generadores de números pseudoaleatorios para varias distribuciones.

Algunas funciones son:

- **random.randint(a, b):** dado un rango [a, b], devuelve un número entero aleatorio derivado de una distribución uniforme.
- **random.random():** dado el rango [0, 1), devuelve un número real aleatorio derivado de una distribución uniforme.
- **random.choices():** devuelve el conjunto de n elementos extraídos de la población especificada y los reemplaza.

#### 3.1.3.3 Biblioteca: numpy

Aunque Python tiene muchos tipos de datos estructurados, en realidad no son adecuados para cálculos numéricos, es aquí donde entra el uso de **Numpy**.



**Numpy** es una biblioteca de Python que ayuda a lidar con arrays (vectores y matrices). Las matrices son datos estructurados y se usan ampliamente en análisis de datos, computación científica y aprendizaje automático. Permite declarar arreglos con diferentes dimensiones, estos arreglos pueden contener grandes cantidades de datos del mismo tipo y relacionados entre sí. Además, proporciona muchas formas de manipular matrices. Y acceder y procesar la información de manera muy eficiente.

#### **3.1.3.4 Biblioteca: sys**

Este módulo proporciona acceso a las funciones y objetos mantenidos por el intérprete.

#### **3.1.3.5 Biblioteca: matplotlib**

Matplotlib es una librería de Python especializada en la creación de gráficos en dos dimensiones. Permite crear y personalizar los tipos de gráficos más comunes, entre ellos:

- Diagramas de barras
- Histogramas
- Diagramas de sectores
- Diagramas de caja y bigotes
- Diagramas de violín
- Diagramas de dispersión o puntos
- Diagramas de líneas
- Diagramas de áreas
- Diagramas de contorno
- Mapas de color

y combinaciones de todos ellos.

#### **3.1.3.6 Biblioteca: timeit**

Esta biblioteca se utiliza cuando queremos saber el tiempo que dura una determinada parte del código que estamos programando.

#### **3.1.3.7 Biblioteca: csv**

El llamado CSV (Valores Separados por Comas) es el formato más común de importación y exportación de hojas de cálculo y bases de datos. Esta biblioteca permite describir el formato CSV que otras aplicaciones pueden entender o definir a su propio formato CSV para propósitos específicos.

#### **3.1.3.8 Biblioteca: xlrd**

Esta una biblioteca que nos facilita la tarea de extraer información de hojas de cálculo (en formato .xls) de Microsoft Excel.



### 3.1.3.9 Biblioteca: *epanettools*

Esta es una biblioteca que nos permite llamar a todas las funciones del "epanet programmers toolkit" dentro de los scripts de Python.

### 3.1.3.10 Biblioteca: *PyQt5*

Es un enlace entre el lenguaje de programación Python y la biblioteca de gráficos QT. **PyQT5** nos permite usar Python para crear interfaces gráficas de forma rápida y sencilla. La legibilidad del código Python hace que sea una tarea extremadamente fácil crear interfaces gráficas, y también tiene una interfaz de diseño para crear interfaces gráficas. La flexibilidad de esta biblioteca es que podemos diseñar completamente nuestra interfaz y luego comenzar a programar.

A continuación, se irá describiendo lo más claro posible cada archivo, clase, método y función en el código utilizado para la realización de esta tesis.

## 3.2 Archivo: *GradienteHidraulico.py*

En esta sección se explican detalladamente las clases:

- Nodos(object).
- Tuberias(object).
- RedHidraulica(object).
- Gradiente(object).

Estas clases sirven para poder solucionar redes hidráulicas matricialmente.

### 3.2.1 Clase Nodos

#### 3.2.1.1 Inicialización de la Clase Nodos.

Esta clase crea un objeto llamado **Nodos** que almacena toda la información que se necesita de un nodo hidráulico, tiene como parámetros de entrada:

- **NúmeroNodo:** es el número (etiqueta) que se le asigna a ese nodo en particular para poder ubicarlo dentro de la red.
- **Gasto\_o\_Altura:** si el nodo es de tipo “N” hace referencia al gasto de demanda (en L.P.S.), pero si es del tipo “E” hace referencia a la altura del embalse que suministra el agua que entra a la red (en metros).
- **Tipo\_elemento:** se considera como nodo de gasto de demanda a los nodos de tipo “N” y como embalse a los nodos de tipo “E”.
- **Nivel\_Topo**: es la cota (en metros) a la cual se encuentra colocado el nodo.

Además de las variables de entrada en cada objeto **Nodos** que se genera se pueden obtener las siguientes variables (dependiendo de la clasificación del nodo):

- **h:** la altura, en metros, del nivel del tanque de almacenamiento si se trata de un nodo tipo “E”.



- **q:** el gasto de demanda, en m<sup>3</sup>/s, si se trata de un nodo tipo "N".

## Clase Nodos

```
class Nodos(object):
    def __init__(self, NumeroNodo, Gasto_o_Altura, Tipo_elemento,
                 Nivel_Topografico):

        self.nn = NumeroNodo
        self.tipo = Tipo_elemento
        self.msnm = Nivel_Topografico
        self.q = 0
        self.h = 0

        if self.tipo == 'E':
            self.h = Gasto_o_Altura
        elif self.tipo == 'N':
            self.q = Gasto_o_Altura / 1000
```

### 3.2.2 Clase Tuberías

#### 3.2.2.1 Inicialización de la Clase Tuberías

Esta clase crea un objeto llamado **Tuberías**, que almacena toda la información que se necesita de una tubería hidráulica, tiene como parámetros de entrada:

- **NumeroTub:** es el número (etiqueta) que se le asigna a esa tubería en particular para poder ubicarla dentro de la red.
- **n\_ini:** número (etiqueta) de nodo inicial, que hace referencia a la dirección en que fluye el flujo dentro de la tubería.
- **n\_fin:** número (etiqueta) de nodo final, que hace referencia a la dirección en que fluye el flujo dentro de la tubería.
- **Longitud:** longitud, en metros, de la tubería a la que se hace referencia.
- **diametro:** es el diámetro, en milímetros, de la tubería a la que se hace referencia.
- **perd\_menores:** coeficiente de pérdidas menores de la tubería a la que se hace referencia.
- **Rugosidad\_Absoluta:** rugosidad absoluta, en milímetros, de la tubería a la que se hace referencia.

Dentro de la clase **Tuberías** hay dos funciones las cuales son:

- [Direccion\(self\)](#)
- [Area\(self\)](#)



## Clase Tuberias

```
class Tuberias(object):
    def __init__(self, Numerotub, n_ini, n_fin, Longiud, diametro,
                 perd_menores, Rugosidad_Absoluta):

        self.nt = Numerotub
        self.ni = n_ini
        self.nf = n_fin
        self.L = Longiud
        self.d = diametro / 1000
        self.km = perd_menores
        self.ks = Rugosidad_Absoluta / 1000

    def Direccion(self):
        return [self.ni, self.nf]

    def Area(self):
        return 0.25 * math.pi * self.d ** 2
```

### 3.2.2.2 Función: Direccion(self).

Esta función se encarga de retornar un vector cuyos valores son el nodo inicial y el nodo final. Esto nos permite (a manera de tanteo) dar una primera suposición de la dirección que toma el flujo dentro de la tubería que se está analizando. En caso de que la dirección que el usuario introduzca sea incorrecta, el programa, a través de la función [InvertirTuberia\(self, Caudal\)](#) la corregirá de manera automática.

### 3.2.2.3 Función: Area(self).

Como la función lo indica, sirve para calcular el área de la sección transversal (en metros cuadrados) de la tubería que se está analizando.

## 3.2.3 Clase RedHidraulica

### 3.2.3.1 Inicialización de la Clase RedHidraulica

Esta clase crea un objeto llamado [RedHidraulica](#), que almacena toda la información que se necesita de una red hidráulica, tiene como parámetros de entrada:

- **data\_nod**: es una lista (extraída de la GUI) que contiene los gastos nodales, altura del embalse, tipo de elemento y la cota de todos los nodos de la red para crear una lista de objetos de la clase [Nodos](#).
- **data\_tub**: es una lista (extraída de la GUI) que contiene el nodo inicial y final, longitud, diámetro, pérdidas menores y rugosidad absoluta de la red para crear una lista de objetos de la clase [Tuberias](#).
- **Pdmax**: presión máxima de diseño, (m.c.a.).
- **Pdmin**: presión mínima de diseño, (m.c.a.).
- **Vdmax**: velocidad máxima de diseño, (m/s).
- **Vdmin**: velocidad mínima de diseño, (m/s).



- **DiametroComercial:** lista de diámetros (extraída de la GUI) a usar para hallar la combinación de red optima que solucione el problema.

## Clase RedHidraulica

```
class RedHidraulica(object):
    def __init__(self, data_nod, data_tub, Pdmax, Pdmin, Vdmax, Vdmin,
                 DiametroComercial):

        # Listas
        self.dataN = data_nod
        self.dataT = data_tub

        # Restricciones hidráulicas
        self.Pd_max = Pdmax # Presion Maxima de Diseño, (m.c.a.).
        self.Pd_min = Pdmin # Presion Minima de Diseño, (m.c.a.).
        self.Vd_max = Vdmax # Velocidad Maxima de Diseño, (m/s).
        self.Vd_min = Vdmin # Velocidad Minima de Diseño, (m/s).

        # Diámetros y valores de regresiones
        self.Dcom = DiametroComercial # Diametro comercial

        self.Nod = []
        row_n = len(self.dataN)
        for i in range(row_n):
            self.Nod.append(Nodos(self.dataN[i][0], self.dataN[i][1],
                                  self.dataN[i][2], self.dataN[i][3]))

        self.Tub = []
        row_t = len(self.dataT)
        for i in range(row_t):
            self.Tub.append(Tuberias(self.dataT[i][0], self.dataT[i][1],
                                    self.dataT[i][2], self.dataT[i][3],
                                    self.dataT[i][4], self.dataT[i][5],
                                    self.dataT[i][6]))

        self.num_nod = len(self.Nod) # Numero total de Nodos
        self.num_tub = len(self.Tub) # Numero total de Tuberias
```

### 3.2.4 Clase Gradiente

#### 3.2.4.1 Inicialización de la Clase Gradiente

Esta clase crea un objeto llamado **Gradiente** que hace uso del **Método del Gradiente** explicado en el **Capítulo I** de esta tesis. Tiene como parámetros de entrada:

- **Tuberías:** lista de objetos **Tuberia**. Esta lista es un vector cuyos elementos son las características de cada tubería en la red hidráulica a analizar.



- **Nodos:** lista de objetos **Nodos**. Esta lista es un vector cuyos elementos son las características de cada nodo en la red hidráulica a analizar.
- **Viscociudad\_cinemática:** es la viscosidad cinemática del fluido que fluye por la red, en este caso por defecto se le da el valor de  $1.141(10^{-6}) m^2/s$  que corresponde al agua.

## Clase Gradiente

```
class Gradiente(object):
    def __init__(self, Tuberias, Nodos, Viscociedad_dinamica = None):

        self.Lt = Tuberias
        self.Ln = Nodos
        if Viscociedad_dinamica == None:
            self.v = 1.141*10**-6
        else:
            self.v = Viscociedad_dinamica

        self.num_nod = len(self.Ln)
        self.NT = len(self.Lt)
        self.NS = 0
        self.NN = 0
        self.Cota = []
        self.D_Cota = []
        self.Qd = []
        self.Ho = []
        self.n_dw = 2
        self.ini_fin = []
        self.I = np.eye(self.NT) I
        self.N = np.eye(self.NT) * self.n_dw
```

Debido a que el **Método del Gradiente** se basa no solo en las características geométricas sino también en la topografía del terreno y de la red en general, se hicieron uso de las siguientes variables de inicialización auxiliares que se calculan de forma automática:

- **Num\_nod:** número total de nodos, sin hacer distinción entre los nodos de demanda y los nodos de embalse.
- **NT:** número total de tuberías.
- **NS:** número total de embalses.
- **NN:** número total de nodos de demanda.
- **Cota:** vector de la cota de cada nodo de demanda, en metros.
- **D\_Cota:** diferencia de alturas entre la cota del embalse y las cotas de los nodos de demanda, en metros.
- **Qd:** vector de los gastos de demanda de cada nodo,  $m^3/s$ .
- **Ho:** vector de alturas piezométricas fijas (embalses), en metros.



- ***n\_dw***: debido a que en el programa se utiliza la ecuación de Darcy – Weisbach como la ecuación de fricción, esta variable tiene el valor constante de 2.
- ***ini\_fin***: vector de vectores que almacena todas las direcciones de flujo de cada tubería (nodo inicial – nodo final) , para en caso de ser necesario, sean corregidas en su dirección correcta.
- ***I***: vector Identidad.
- ***N***: matriz diagonal Darcy – Weisbach.

A continuación, se calcularon los vectores mencionados anteriormente:

### Clase Gradiente (Continuación)

```
# DEMANDAS, ALTURAS Y COTAS
Cota = []
Demanda = []
AlturaEmbalce = []
for i in range(self.NT):
    if i <= (self.num_nod-1):
        if self.Ln[i].tipo == 'E':
            AlturaEmbalce.append([self.Ln[i].h])
            self.NS += 1
        elif self.Ln[i].tipo == 'N':
            Demanda.append([self.Ln[i].q])
            self.NN += 1
        Cota.append([self.Ln[i].msnm])
    self.ini_fin.append([self.Lt[i].Direccion()[0],
                         self.Lt[i].Direccion()[1]])

self.Cota = np.array(Cota)
self.D_Cota = [self.Cota[0] - val for val in self.Cota]
self.Qd = np.array(Demanda)
self.Ho = np.array(AlturaEmbalce)

# Matrices A12, A21 y A10
self._MatricesPrincipales()
```

Dentro de la clase ***Gradiente*** están las siguientes funciones:

- [\*\*\*MatricesPrincipales \(self\)\*\*\*](#)
- [\*\*\*Vel\(self, A, Q\)\*\*\*](#)
- [\*\*\*Re\(self, V, d\)\*\*\*](#)
- [\*\*\*HagenPoiseuille\(self, Re\)\*\*\*](#)
- [\*\*\*SwameeJain\(self, Re, d, ks\)\*\*\*](#)
- [\*\*\*ColebrookWhite\(self, Re, d, ks\)\*\*\*](#)
- [\*\*\*Fn Friccion\(self, Re, d, ks\)\*\*\*](#)
- [\*\*\*InvertirTuberia\(self, Caudal\)\*\*\*](#)



- [Calcular\(self\)](#)
- [Resolver\(self\)](#)

### 3.2.4.2 Función: MatricesPrincipales (self)

Esta función tiene como valores de retorno a las matrices, A12, A21 y A10.

Donde:

- A12: Es la matriz de conectividad (-1 inicio y +1 final). Cada renglón de la matriz representa una tubería, el nodo inicial de la tubería tiene como valor -1, y el nodo final de la tubería tiene como valor +1.
- A21: Es la transpuesta de la matriz A12.
- A10: Es la matriz topológica tramo a nudo para los  $NS$  nodos de altura piezométrica fija (embalses).

#### Clase Gradiente (Continuación)

```
def __MatricesPrincipales(self):  
  
    # Matriz de conectividad (-1 inicio y 1 final)  
    self.A12 = []  
    A12 = np.zeros((self.NT, self.NN + self.NS))  
    for i in range(self.NT):  
        ni = self.ini_fin[i][0] - 1  
        nf = self.ini_fin[i][1] - 1  
        A12[i][ni] = -1  
        A12[i][nf] = 1  
    self.A12 = A12[:, self.NS:(self.NN + self.NS)]  
  
    # Matriz Transpuesta de A12 --> A21  
    self.A21 = []  
    self.A21 = np.transpose(self.A12)  
  
    # Matriz Topologica tramo a nudo  
    self.A10 = []  
    self.A10 = A12[:, 0:self.NS]  
  
    return self.A12, self.A21, self.A10
```

### 3.2.4.3 Función: Vel(self, A, Q)

Esta función recibe como parámetros de entrada el área de la sección transversal y el gasto de la tubería a analizar. Retorna la velocidad del fluido en  $m/s$ .

#### Clase Gradiente (Continuación)

```
def __Vel(self, A, Q):  
    return Q / A
```



### 3.2.4.4 Función: [Re\(self, V, d\)](#)

Esta función recibe como parámetros de entrada la velocidad del fluido y el diámetro de la sección transversal de la tubería. Retorna el número de Reynolds.

#### Clase Gradiente (Continuación)

```
def __Re(self, V, d):  
    return abs(V * d / self.v)
```

### 3.2.4.5 Función: [HagenPoiseuille\(self, Re\)](#)

Esta función recibe como parámetro de entrada el número de Reynolds. Retorna el factor de fricción. Solo es válida para valores de Reynolds menores a 2200.

#### Clase Gradiente (Continuación)

```
def __HagenPoiseuille(self, Re):  
    f = 64 / Re  
    return f
```

### 3.2.4.6 Función: [SwameeJain\(self, Re, d, ks\)](#)

Esta función se utilizará como valor semilla para inicializar de una manera más aproximada la ecuación de Colebrook-White para valores de Reynolds mayores o iguales a 2200.

#### Clase Gradiente (Continuación)

```
def __SwameeJain(self, Re, d, ks):  
    v1 = ks / (3.71 * d)  
    v2 = 5.74 / (Re ** 0.9)  
    v3 = math.log10(v1 + v2)  
    v4 = v3 ** 2  
    f = 0.25 / v4  
    return f
```

### 3.2.4.7 Función: [ColebrookWhite\(self, Re, d, ks\)](#)

Esta función recibe como parámetro de entrada el número de Reynolds, el diámetro de la sección transversal de la tubería y la rugosidad absoluta de la tubería. Retorna el factor de fricción. Será usada para los valores de Reynolds en el intervalo mayor a 2200.

Para resolver la ecuación de Colebrook–White, se utilizó el método de Newton-Raphson tal como se describe en el **Capítulo I** de esta tesis.



La tolerancia permitida de error usada fue de  $10^{-5}$ , comenzando con un valor semilla calculado con la función de Swamee-Jain para una mejor aproximación.

### Clase Gradiente (Continuación)

```
def __ColebrookWhite(self, Re, d, ks):

    def Fricc(Re, ks, d, x):
        a = ks / (3.7 * d)
        b = 2.51 * x / Re
        return - 2 * math.log10(a + b) - x

    def Fricc_p(Re, ks, d, x):
        a = - 2 / math.log(10)
        b = ks / (3.7 * d)
        c = 2.51 * x / Re
        d = 2.51 / Re
        return (a * d / (b + c)) - 1

    tol = 0.00001 # Tolerancia permitida del error
    error = 100 # Error de inicialización

    while error >= tol:
        if error == 100:
            x1 = self.__SwameeJain(Re, d, ks)
            Fxp = Fricc_p(Re, ks, d, x1)
            Fx = Fricc(Re, ks, d, x1)
            x2 = x1 - (Fx / Fxp)
            error = abs((x2 - x1) / x2) * 100
            x1 = x2
        f = 1 / (x1 ** 2)
    return f
```

#### 3.2.4.8 Función: *Fn\_Friccion(self, Re, d, ks)*

Esta función recibe como parámetro de entrada el número de Reynolds, el diámetro de la sección transversal de la tubería y la rugosidad absoluta de la tubería. Retorna el factor de fricción.

Se encarga de, en función del número de Reynolds, asignar la ecuación correspondiente para calcular el factor de fricción.

### Clase Gradiente (Continuación)

```
def Fn_Friccion(self, Re, d, ks):

    if Re < 2200:
        return self.__HagenPoiseuille(Re)
    elif Re >= 2200:
        return self.__ColebrookWhite(Re, d, ks)
```



### 3.2.4.9 Función: InvertirTuberia(self, Caudal)

Esta función recibe como parámetro de entrada el caudal de las tuberías. Si algún caudal de la tubería  $j$  tiene un valor negativo, invierte la dirección del vector **ini\_fin** en la posición  $j$ .

Retorna el vector de dirección del flujo de la tubería con las direcciones corregidas.

#### Clase Gradiente (Continuación)

```
def __InvertirTuberia(self, Caudal):  
  
    aux = self.ini_fin  
    self.ini_fin = []  
    for i in range(self.NT):  
        if Caudal[i] < 0:  
            self.ini_fin.append([aux[i][1], aux[i][0]])  
        else:  
            self.ini_fin.append([aux[i][0], aux[i][1]])  
    return self.ini_fin
```

### 3.2.4.10 Función: Calcular(self)

Esta función es la encargada de realizar los cinco pasos que se describen a continuación:

1. Se asignan unos caudales iniciales con valor de  $0.1 \text{ m}^3/\text{s}$  en cada tubería de la red creando el vector  $[Q_i]$ . Estos caudales, al estar asignados arbitrariamente, no necesitan estar balanceados, lo cual ayuda a agilizar los cálculos de la red pues se irán corrigiendo de manera iterativa.
2. Se resuelve el sistema dado por la ecuación matricial (18) usando **Numpy**, la cual es una biblioteca para el lenguaje de programación Python que admite la creación de grandes matrices y vectores multidimensionales, así como una gran cantidad de funciones matemáticas avanzadas para manipularlos.
3. Una vez calculado el valor del vector  $[H_{i+1}]$  es posible calcular  $[Q_{i+1}]$  en forma explícita, mediante la ecuación (19).
4. Con este nuevo vector  $[Q_{i+1}]$ , se hace  $[Q_i] = [Q_{i+1}]$  y se vuelve a construir el sistema de la ecuación (18) para encontrar un nuevo vector  $[H_{i+1}]$ .
5. El proceso se detiene hasta que se cumplan cualquiera de las siguientes condiciones:
  - ✓  $\|[Q_{i+1}]\| - \|Q_i\| \leq 10^{-6}$
  - ✓ El número de iteraciones es mayor a dos veces el número de tuberías en la red.



Retorna los vectores de caudal ( $m^3/s$ ), altura (m), presión (m) y un seguimiento de cómo fue disminuyendo el error iteración tras iteración, hasta converger, para ser usado en una gráfica en la interfaz de usuario.

## Clase Gradiente (Continuación)

```
def __Calcular(self):
    itera = 1
    er = 100
    ErrorLista = []

    # Iteraciones
    while er > 0.00001 and itera < 2 * self.NN:
        A11 = np.zeros((self.NT, self.NT))
        A11_p = np.zeros((self.NT, self.NT))
        if itera == 1:
            Qi = np.ones((self.NT, 1)) * 0.1

        for j in range(self.NT):
            Area = self.Lt[j].Area()
            diametro = self.Lt[j].d
            ks = self.Lt[j].ks
            km = self.Lt[j].km
            V = self.__Vel(Area, Qi[j])
            Re = self.__Re(V, diametro)
            f = self.Fn_Friccion(Re, diametro, ks)
            Longitud = self.Lt[j].L
            alpha = ((f * Longitud / diametro) + km) / (19.62 *
                (Area ** 2))
            beta = 0
            gamma = 0
            a = alpha * (Qi[j] ** (self.n_dw - 1))
            b = beta
            c = gamma / Qi[j]
            A11[j][j] = a + b + c
            A11_p[j][j] = a

        # Calculo de la Altura Piezometrica H(i+1)
        h1 = np.linalg.inv(np.matmul(self.N, A11_p))
        h2 = np.linalg.inv(np.matmul(np.matmul(self.A21, h1),
            self.A12))
        h3 = np.matmul(A11, Qi) + np.matmul(self.A10, self.Ho)
        h4 = np.matmul(np.matmul(self.A21, h1), h3)
        h5 = h4 - np.matmul(self.A21, Qi) + self.Qd
        H = -np.matmul(h2, h5)

        # Calculo del Caudal Q(i+1)
        Qanterior = np.linalg.norm(Qi)
        q1 = self.I - np.matmul(h1, A11)
        q2 = np.matmul(self.A12, H) + np.matmul(self.A10, self.Ho)
        Qi = np.matmul(q1, Qi) - np.matmul(h1, q2)
        Qactual = np.linalg.norm(Qi)

        # Error Relativo Porcentual
        er = abs((Qactual - Qanterior) / Qactual) * 100
        ErrorLista.append(er)
```



```
itera += 1

Caudal = Qi * 1000
Presion = np.zeros((self.NN, 1))
Altura = np.zeros((self.NN, 1))

for i in range(self.NN):
    Presion[i] = H[i] + self.D_Cota[i + self.NS]
    Altura[i] = self.Cota[i + self.NS] + Presion[i]

return Caudal, Altura, Presion, ErrorLista
```

### 3.2.4.11 Función: *Resolver(self)*

Esta función tiene como tarea principal el buscar si el vector **Caudal** derivado de la función [Calcular\(self\)](#), tiene algún valor negativo en la posición *j*, y en caso de tenerlo, invertir el sentido del flujo del vector **ini\_fin** en la posición *j* para siempre asegurar el tener caudales positivos.

El proceso se detiene hasta que el número de iteraciones es mayor al número de tuberías en la red.

Retorna los vectores de caudal ( $m^3/s$ ), velocidad ( $m/s$ ), altura ( $m$ ), presión ( $mts.$ ), factor de fricción, perdidas menores ( $mts.$ ),y un seguimiento de cómo fue disminuyendo el error iteración tras iteración, hasta converger, para ser usado en una gráfica en la interfaz de usuario.

### Clase Gradiente (Continuación)

```
def Resolver(self):
    iter = 0
    Velocidad = np.zeros((self.NT, 1))
    Perd = np.zeros((self.NT, 1))
    Fricc = np.zeros((self.NT, 1))

    while iter < self.NT:
        self.__MatricesPrincipales()
        [self.Caudal, self.Altura, self.Presion, self.er] =
            self.__Calcular()
        if self.Caudal.min() < 0:
            self.__InvertirTuberia(self.Caudal)
            iter += 1
        elif self.Caudal.min() >= 0:
            for i in range(self.NT):
                Velocidad[i] = (self.Caudal[i] / 1000) /
                    self.Lt[i].Area()
                diam2 = self.Lt[i].d
                ks2 = self.Lt[i].ks
                Reynolds2 = self.__Re(Velocidad[i], diam2)
                Fricc[i] = self.Fn_Friccion(Reynolds2, ks2, diam2)
                Perd[i] = (Fricc[i] * self.Lt[i].L / diam2) *
                    ((Velocidad[i] ** 2) / 19.62)
```



```
        break
    return self.Caudal, Velocidad, self.Altura, self.Presion, Fricc,
    Perd, self.er, self.ini_fin
```

### 3.3 Archivo: AlgoritmoGenetico.py

En esta sección se explican detalladamente las clases:

- Individuo(Object).
- AlgoritmoGenetico(Object).

Estas clases sirven para poder encontrar los diámetros más económicos y que cumplan con las normas de velocidad y presión de la red hidráulica analizada.

#### 3.3.1 Clase Individuo

##### 3.3.1.1 Inicialización de la Clase Individuo

Esta clase crea un objeto llamado **Individuo** que almacena toda la información que se necesita de una red hidráulica, tiene como parámetros de entrada:

- **RedHidraulica**: esta clase tiene en sus propiedades las siguientes listas de objetos: **Nodos** y **Tuberia** las cuales servirán para definir al individuo a evolucionar.
- **K1**: constante de regresión (Lineal, Exponencial o Potencial):  $y = f(a, b, x)$ .
- **K2**: constante de regresión (Lineal, Exponencial o Potencial):  $y = f(a, b, x)$ .
- **RegTipo**: este parámetro le dice al algoritmo que tipo de regresión usará.
- **DiametroTuberias**: diámetros de la red hidráulica a analizar. Esta variable viene predefinida como vacía.

#### Clase Individuo

```
class Individuo:
    def __init__(self, RedHidraulica, K1, K2, RegTipo,
                 DiametroTuberias = []):
        self.I_cn = RedHidraulica.Nod
        self.I_ct = RedHidraulica.Tub
        self.Dcom = RedHidraulica.Dcom
        self.diametros = DiametroTuberias
        self.K1 = K1
        self.K2 = K2
        self.RT = RegTipo
```



Debido a que cada red hidráulica es un objeto **Individuo** que debe ser evaluado de manera específica para poder compararlo con otras redes hidráulicas para su futura comparación y discriminación, se hicieron uso de las siguientes variables de inicialización auxiliares que se calculan de forma automática:

- **Pdmax**: presión máxima de diseño. La presión máxima para evitar un posible golpe de ariete viene predefinida como 50 m.c.a.
- **Pdmin**: presión mínima de diseño. Para garantizar que el agua llegue con la presión mínima adecuada a los nodos viene predefinida como 15m.c.a.
- **Vdmax**: velocidad máxima de diseño. La velocidad máxima permisible para evitar la erosión de la tubería viene predefinida como 5 m/s.
- **Vdmin**: velocidad mínima de diseño. Para evitar que se sedimenten partículas que arrastre el agua, el flujo viene predefinido con velocidad mínima de 0.6 m/s.
- **num\_tub**: es el número de tuberías que tiene la red hidráulica. En el contexto de Algoritmo Genético esta variable sería la longitud del cromosoma.
- **Penalizacion**: es el Costo Hidráulico de la red. Como se define en la sección 2.6.2 es la sumatoria de las restricciones (velocidades y presiones tanto máximas como mínimas).
- **Fit**: es el valor de la función de evaluación que consiste en asignarle un valor numérico de adaptación, el cual se supone sea proporcional a la utilidad o habilidad del individuo representado, esto puede considerarse como la probabilidad de que ese individuo pueda sobrevivir en el ecosistema del problema hasta que tenga la capacidad de reproducirse y se reproduzca.

### Clase Individuo (Continuación)

```
# Restricciones Hidraulicas
self.Pd_max = RedHidraulica.Pd_max
self.Pd_min = RedHidraulica.Pd_min
self.Vd_max = RedHidraulica.Vd_max
self.Vd_min = RedHidraulica.Vd_min

# Inicializamos el cromosoma, el diametro y el fitness en vacio
# pues recibirán datos despues
self.num_tub = len(self.I_ct)
self.Penalizacion = 0
self.Fit = 0
```

Al invocar la clase **Individuo** la variable **DiametroTuberias** puede tener dos posturas, o está vacía o está llena:

- Vacía. Si no se conocen los diámetros a evaluar en la red, esta variable se conoce como vacía. En este caso se toman diámetros aleatorios de la lista



**DiametroComercial** haciendo uso de la biblioteca **random**. Estos diámetros se pasan a metros para su posterior uso.

- Llena. Si se conocen los diámetros a evaluar en la red, esta variable se conoce como llena. En este caso los diámetros solo se pasan a metros para su posterior uso.

### Clase Individuo (Continuación)

```
# Creacion de la lista DiametroTuberias
if len(self.diametros) == 0:
    # Se crean diametros al azar
    for i in range(self.num_tub):
        val = round(random.randint(0, len(self.Dcom) - 1))
        self.diametros.append(self.Dcom[val])
        self.I_ct[i].d = self.diametros[i] / 1000
else:
    for i in range(self.num_tub):
        self.I_ct[i].d = self.diametros[i] / 1000
```

Teniendo todos los requisitos necesarios se invoca a la clase **Gradiente** y su función **Resolver(self)** para poder obtener las velocidades y presiones máximas y mínimas específicas de esa red hidráulica analizada.

### Clase Individuo (Continuación)

```
# Ejecutamos el Metodo del Gradiente para evaluar al individuo
# Hidráulicamente

# [Q, V, H, P, f, hf, er]
self.cg = gh.Gradiente(self.I_ct, self.I_cn).Resolver()

self.Vel = self.cg[1]
self.v_min = float(min(self.Vel))
self.v_max = float(max(self.Vel))

self.Press = self.cg[3]
self.p_min = float(min(self.Press))
self.p_max = float(max(self.Press))
```

Dentro de la clase **Individuo** están las siguientes funciones:

- [\*\*Cc\(self\)\*\*](#)
- [\*\*penalizacion\(self\)\*\*](#)
- [\*\*Fitness\(self\)\*\*](#)



### 3.3.1.2 Función: Cc(self)

Esta función usa las constantes de regresión (lineal, exponencial o potencial), las tuberías y los diámetros asociados a esas tuberías para calcular el costo de construcción de la red hidráulica analizada, sección 2.6.3. Retorna el costo de construcción.

#### Clase Individuo (Continuación)

```
def Cc(self):
    self.Costo = 0

    # Costo = L * ( K1 * D + K2 )
    if self.RT == "Lineal":
        for i in range(self.num_tub):
            self.Costo += self.I_ct[i].L * (self.K1 *
                self.diametros[i] + self.K2)

    # Costo = L * (K1 * 10 ** (K2 * D))
    elif self.RT == "Exponencial":
        for i in range(self.num_tub):
            self.Costo += self.I_ct[i].L * (self.K1 * 10 ** (self.K2
                * self.diametros[i]))

    # Costo = L * (K1 * D ** K2)
    elif self.RT == "Potencial":
        for i in range(self.num_tub):
            self.Costo += self.I_ct[i].L * (self.K1 *
                self.diametros[i] ** self.K2)

    return self.Costo
```

### 3.3.1.3 Función: penalizacion(self)

Esta función evalúa cada restricción hidráulica tal como se describe en la sección 2.6.1. Retorna el valor de penalización como la sumatoria de las restricciones hidráulicas (sección 2.6.2).

#### Clase Individuo (Continuación)

```
def penalizacion(self):

    # Penalización Hidráulica
    if (self.v_min < self.Vd_max) and (self.v_min >= self.Vd_min):
        P_vmin = 0
    else:
        P_vmin = 10*abs(self.v_min - self.Vd_min)

    if (self.v_max <= self.Vd_max) and (self.v_max > self.Vd_min):
        P_vmax = 0
    else:
        P_vmax = 10*abs(self.v_max - self.Vd_max)
```



```
if (self.p_max <= self.Pd_max) and (self.p_max > self.Pd_min) :  
    P_pmax = 0  
else:  
    P_pmax = abs(self.p_max - self.Pd_max)  
  
if (self.p_min < self.Pd_max) and (self.p_min >= self.Pd_min) :  
    P_pmin = 0  
else:  
    P_pmin = abs(self.p_min - self.Pd_min)  
  
# Penalizacion Hidraulica total para el individuo  
self.Penalizacion = P_pmax + P_pmin + P_vmax + P_vmin  
  
return self.Penalizacion
```

### 3.3.1.4 Función: *Fitness(self)*

Tal como se muestra en la ecuación (20), la función *Fitness(self)* está en función de la función de costos de construcción y costos hidráulicos. Esta función retorna la evaluación de la red, desde el punto de vista de que tan óptima es comparada con las demás redes.

### Clase Individuo (Continuación)

```
def Fitness(self) :  
    self.Fit = self.Cc() * (1 + self.penalizacion())  
    return self.Fit
```

## 3.3.2 Clase AlgoritmoGenetico

### 3.3.2.1 Inicialización de la Clase AlgoritmoGenetico

Esta clase crea un objeto llamado *AlgoritmoGenetico* que hace uso del **Método de Optimización** explicado en el **Capítulo II** sección 2.2 de esta tesis. Tiene como parámetros de entrada:

- **RedHidraulica:** esta clase tiene en sus propiedades las siguientes listas de objetos: *Nodos* y *Tuberia* las cuales servirán para definir al individuo a evolucionar.
- **K1:** constante de regresión (Lineal, Exponencial o Potencial):  $y = f(a, b, x)$ .
- **K2:** constante de regresión (Lineal, Exponencial o Potencial):  $y = f(a, b, x)$ .
- **RegTipo:** este parámetro le dice al algoritmo que tipo de regresión usará.
- **tamano\_poblacion:** es la cantidad de redes a analizar (Individuos) que tratan de cubrir toda el área de soluciones. No puede ser demasiado pequeña pues no alcanzaría a encontrar la solución óptima, pero tampoco demasiado grande porque sería demasiado costosa computacionalmente.



- **tasa\_mutacion:** es la probabilidad de que un gen (diámetro) del cromosoma (red) pueda cambiar a otro valor. Por lo general se toman valores menores al 5%.
- **num\_generaciones:** es el número de veces que se va a correr el algoritmo genético.

### Clase AlgoritmoGenetico

```
class AlgoritmoGenetico():
    def __init__(self, RedHidraulica ,k1, k2, RegreTipo,
                 tamano_poblacion, tasa_mutacion, num_generaciones):

        self.rh = RedHidraulica
        self.K1 = k1
        self.K2 = k2
        self.RT = RegreTipo
        self.tam_pob = tamano_poblacion
        self.tasa_mutacion = tasa_mutacion
        self.num_gene = num_generaciones
```

Para poder manipular la población (espacio de soluciones) y sus evaluaciones (fitness), se hicieron uso de las siguientes variables de inicialización auxiliares que se calculan de forma automática:

- **Dcom:** lista de diámetros (extraída de la GUI) a usar para hallar la combinación de red optima que solucione el problema.
- **poblacion:** es una lista de los objetos **Individuo**.
- **poblacion\_evaluada:** es una lista que almacena el valor del **Fitness** de cada uno de los individuos. Se inicializa como lista vacía.
- **suma\_porcentajes:** es una lista que almacena el valor del **Fitness** de cada uno de los individuos, pero de manera porcentual como se define en la sección 2.5.1.1. Se inicializa como lista vacía.
- **mejores\_padres:** es una lista que almacena los objetos **Individuo** que pasaron el filtro de ruleta aleatoria como se define en la sección 2.5.1.1. Se inicializa como lista vacía.
- **mejor\_individuo:** es una lista que almacena el mejor objeto **Individuo** de cada generación. Se inicializa como lista vacía.

### Clase AlgoritmoGenetico (Continuación)

```
self.Dcom = RedHidraulica.Dcom
self.poblacion = []
self.poblacion_evaluada = []
self.suma_porcentajes = []
```



```
self.mejores_padres = []
self.mejor_individuo = []
```

Dentro de la clase [AlgoritmoGenetico](#) están las siguientes funciones:

- [inicializar\\_poblacion\(self\)](#)
- [poblacion\\_eval\(self\)](#)
- [ordenar\\_poblacion\(self\)](#)
- [mejor\\_individuo\\_especie\(self, gene\)](#)
- [porcentaje\\_acumulado\(self\)](#)
- [ruleta\\_aleatoria\(self\)](#)
- [elitismo\(self\)](#)
- [nueva\\_poblacion\(self, new\\_padres\)](#)
- [crossover\(self\)](#)
- [mutacion\(self, hijos\)](#)
- [TiempoEjecucion\(self, seg\)](#)
- [resolver\(self\)](#)

### 3.3.2.2 Función: [inicializar\\_poblacion\(self\)](#)

Esta función es la encargada de inicializar la población y darle valores a la lista de la variable **poblacion**. Retorna una lista de longitud igual al número del tamaño de la población.

#### Clase AlgoritmoGenetico (Continuación)

```
def inicializar_poblacion(self):
    self.poblacion = []
    for i in range(self.tam_pob):
        ind = Individuo(self.rh, self.K1, self.K2, self.RT, [])
        self.poblacion.append(ind)
    return self.poblacion
```

### 3.3.2.3 Función: [poblacion\\_eval\(self\)](#)

Esta función almacena el fitness de cada individuo y lo almacena en la lista de la variable **poblacion\_evaluada**.

#### Clase AlgoritmoGenetico (Continuación)

```
def poblacion_eval(self):
    self.poblacion_evaluada = []
    for i in range(self.tam_pob):
        self.poblacion_evaluada.append(self.poblacion[i].Fitness())
```



```
return self.poblacion_evaluada
```

### 3.3.2.4 Función: ordenar\_poblacion(self)

Haciendo uso de la función **sorted** nativa de *Python* ordenamos la lista **poblacion** en función del fitness de cada individuo en orden descendiente.

#### Clase AlgoritmoGenetico (Continuación)

```
def ordenar_poblacion(self):
    self.poblacion = sorted(self.poblacion,
                           key=lambda poblacion: poblacion.Fit,
                           reverse=True)
```

### 3.3.2.5 Función: mejor\_individuo\_especie(self, gene)

Esta función recibe como parámetro de entrada el número de la generación actual.

Se encarga en cada generación de obtener el fitness, diámetro, costo hidráulico y número de generación del mejor individuo de la generación actual.

Si la generación es la inicial, estos valores obtenidos se guardan en la variable **mejor\_individuo**. Para las demás generaciones, se evalúa si el fitness del mejor individuo actual es menor que el mejor individuo de generaciones pasadas, si esto se cumple se almacena en la variable **mejor\_individuo**, en caso contrario no se almacena.

#### Clase AlgoritmoGenetico (Continuación)

```
def mejor_individuo_especie(self, gene):
    mejor_de_la_especie = min(self.poblacion, key=lambda poblacion:
                               poblacion.Fit)

    diam = mejor_de_la_especie.diametros
    val = mejor_de_la_especie.Fit
    hid = mejor_de_la_especie.Penalizacion
    cos = mejor_de_la_especie.Costo

    if gene == 0:
        self.mejor_individuo.append([diam, hid, val, cos, gene])
    if (gene > 0) and (val < self.mejor_individuo[-1][2]):
        self.contador = 0
        self.mejor_individuo.append([diam, hid, val, cos, gene])
    else:
        self.contador += 1

    return self.mejor_individuo
```



### 3.3.2.6 Función: porcentaje\_acumulado(self)

Esta función primero calcula la inversa del fitness de cada individuo de la población, **Fitness\_inv**, para obtener el mínimo de la función objetivo (**Fitness(self)**). Después suma todos estos fitness invertidos, **Sfx**. Por último, crea una lista de suma de porcentajes, **fx\_Sfx**, bajo dos condiciones:

1. Si  $i = 0$ , se divide el primer elemento entre la sumatoria total
2. Para los demás casos, solo se obtiene el porcentaje acumulado, de tal forma que el último elemento de la lista tiene como valor 1.

#### Clase AlgoritmoGenetico (Continuación)

```
def porcentaje_acumulado(self):  
    Fitness_inv = []  
    for i in range(self.tam_pob):  
        Fitness_inv.append(1 / self.poblacion_evaluada[i])  
    Sfx = sum(Fitness_inv)  
    fx_Sfx = []  
    for i in range(self.tam_pob):  
        if i == 0:  
            fx_Sfx.append(Fitness_inv[i] / Sfx)  
        else:  
            fx_Sfx.append(fx_Sfx[i - 1] + Fitness_inv[i] / Sfx)  
    self.suma_porcentajes = fx_Sfx  
    return self.suma_porcentajes
```

### 3.3.2.7 Función: ruleta aleatoria(self)

De la misma manera que una ruleta física, el proceso es el siguiente:

1. Se elige un numero finito de tiros para lanzar a la ruleta, en este caso, el número de tiros es igual al tamaño de la población. Esto se hace para tener la misma cantidad de individuos en la población.
2. El valor del tiro, **tiro**, es un número aleatorio  $r$ , entre [0,1).
3. Se selecciona el  $i$ -ésimo individuo (**población[i].diametros**), tal que  $q_{i-1} < r \leq q_i$ .
4. Donde  $q_i$  es la suma de las probabilidades:

$$q_i = \sum_{I=1}^i P_I$$

Este procedimiento garantiza, en la mayoría de las veces, siempre elegir la mejor combinación de diámetros. La ventaja de que no se elijan siempre los mejores diámetros ayuda a tener una población más diversa y así no limitar nuestra búsqueda a un mínimo local.

#### Clase AlgoritmoGenetico (Continuación)



```
def ruleta_aleatoria(self):
    self.mejores_padres = []
    num_tiros = self.tam_pob
    for i in range(0, num_tiros - 1):
        tiro = random.random()
        if tiro <= self.suma_porcentajes[0]:
            self.mejores_padres.append(self.poblacion[0].diametros)
        else:
            for j in range(1, num_tiros):
                if self.suma_porcentajes[j - 1] < tiro and tiro <=
                    self.suma_porcentajes[j + 1]:
                    self.mejores_padres.append(self.poblacion[j]
                                                .diametros)
                    break
    self.elitismo()
    return self.mejores_padres
```

### 3.3.2.8 Función: elitismo(self)

Después de obtener los mejores padres usando la función [ruleta\\_aleatoria\(self\)](#) se corre el riesgo de que no se haya encontrado, en cierta generación, a una mejor solución al problema. Para evitar esto, se recurre a la función [elitismo\(self\)](#), cuyo único propósito es la de buscar en generaciones pasadas a la mejor solución y traerla a la generación presente para usar sus genes y mejorar las generaciones actuales.

#### Clase AlgoritmoGenetico (Continuación)

```
def elitismo(self):
    self.mejores_padres.append(self.mejor_individuo[-1][0])
```

### 3.3.2.9 Función: nueva\_poblacion(self)

Esta función recibe como parámetro de entrada la lista de los mejores diámetros de la generación evaluada.

Se encarga en cada generación de actualizar la lista de población haciendo uso de la clase [Individuo](#).

#### Clase AlgoritmoGenetico (Continuación)

```
def nueva_poblacion(self, new_padres):
    self.poblacion = []
    for i in range(self.tam_pob):
        ind = Individuo(self.rh, self.K1, self.K2, self.RT,
                        new_padres[i])
        self.poblacion.append(ind)
```



```
return self.poblacion
```

### 3.3.2.10 Función: crossover(self)

Esta función se encarga de reproducir o cruzar los mejores individuos de la población heredando soluciones mejores que cada uno de los padres por separado.

Utilizando la variable **población[i].diametros**, genera dos listas, una llamada **madres** y otra llamada **padres**, y como se explica en la sección **2.5.2.2 Cruce multipunto**, se generan puntos de corte aleatorios a cada cromosoma del individuo para mezclar información de los padres intercambiando el valor de sus respectivos genes.

Retorna una lista llamada **Hijos**, la cual contiene los nuevos descendientes “mejorados” de la generación actual.

### Clase AlgoritmoGenetico (Continuación)

```
def crossover(self):
    padres = []
    madres = []
    '''Separamos en padres y madres'''
    for i in range(0, self.tam_pob, 2):
        padres.append(self.poblacion[i].diametros)
        madres.append(self.poblacion[i + 1].diametros)
    ''' Cruzamos los padres y madres'''
    Hijos = []
    for i in range(len(padres)):
        padre = padres[i]
        madre = madres[i]
        corte = round(random.random() * (len(padre) - 1))
        Hijo1 = padre[0:corte] + madre[corte::]
        Hijo2 = madre[0:corte] + padre[corte::]
        Hijos.append(Hijo1)
        Hijos.append(Hijo2)
    return Hijos
```

### 3.3.2.11 Función: mutacion(self)

Esta función recibe como parámetro la lista **Hijos**

Es la encargada de modificar uno o varios alelos del cromosoma en función de la tasa de mutación (**tasa\_mutacion**) como se explica en la sección **2.5.3 Mutación**.

Funciona de la siguiente forma:

1. Por cada alelo se genera un número aleatorio  $\rho$ .



- Si  $\rho < \text{tasa\_mutacion}$ , entonces se muta ese alelo cambiando su valor por otro valor aleatorio en la lista *DiametroComercial*.

### Clase AlgoritmoGenetico (Continuación)

```
def mutacion(self, hijos):
    mut = []
    for i in range(len(hijos)):
        h = hijos[i]
        for j in range(len(h)):
            if random.random() < self.tasa_mutacion:
                h[j] = self.Dcom[round(random.randint(0,
                len(self.Dcom) - 1))]
        mut.append(h)
    return mut
```

#### 3.3.2.12 Función: *TiempoEjecucion(self, seg)*

Esta función se encarga de transformar los segundos que tardó en ejecutar el algoritmo genético y los transforma a horas, minutos y segundos para poder después visualizarlos en la GUI.

### Clase AlgoritmoGenetico (Continuación)

```
def TiempoEjecucion(self, seg):
    hr = seg / 3600
    hora = int(hr) # Hora
    minu = (hr - hora) * 60
    minuto = int(minu) # Minutos
    segundo = int(round((minu - minuto) * 60, 0))
    return hora, minuto, segundo
```

#### 3.3.2.13 Función: *resolver(self)*

Esta es la función principal del algoritmo genético, pues es la encargada de combinar todas las funciones anteriormente creadas para realizar los cálculos de manera ordenada.

Funciona de la siguiente forma:

- Inicializamos una población de  $n$  Individuos (con  $n$  par), y creamos un contador con valor inicial en cero.
- Evaluamos el fitness de la población recientemente creada.
- Ordenamos la población de peor a mejor individuo.
- Seleccionamos al mejor individuo de la especie,  $I_{mejor}$ .



5. Si la generación no es la inicial, se compara el  $I_{mejor}$  actual con el pasado, y si no mejora, se incrementa el contador en una unidad. En caso contrario se vuelve a evaluar en cero.
6. Ponemos a competir a todos los individuos usando el método de ruleta aleatoria.
7. Creamos una nueva población en función de los que pasen el filtro de ruleta aleatoria.
8. Evaluamos la población creada.
9. Seleccionamos al mejor individuo de la especie,  $I_{mejor}$ .
10. Si la generación no es la inicial, se compara el  $I_{mejor}$  actual con el pasado, y si no mejora, se incrementa el contador en una unidad. En caso contrario se vuelve a evaluar en cero.
11. Cruzamos a los individuos.
12. Mutamos a los individuos al azar.
13. Se crea una nueva población.
14. Si el valor del contador es igual o mayor a 100, o si el número de generaciones requeridas se ha alcanzado, se detiene el algoritmo, retornando la variable: **mejor\_individuo**.
15. En caso de que no se cumpla con las restricciones del paso 14, se repite el algoritmo desde el paso 2.

### Clase AlgoritmoGenetico (Continuación)

```
def resolver(self):  
    tic = timeit.default_timer()  
  
    # Inicializamos la poblacion:  
    self.inicializar_poblacion()  
    self.contador = 0  
    self.main = ProgBar()  
    self.main.show()  
  
    for i in range(self.num_gene):  
  
        # self.generacion.append(i)  
        QCoreApplication.processEvents()  
        self.main.repaint()  
        self.main.progressBarValue(i, self.num_gene)  
  
        # Evaluamos la poblacion para posteriormente ordenarla:  
        self.poblacion_eval()  
  
        # Ordenamos la poblacion:  
        self.ordenar_poblacion()  
        self.poblacion_eval()  
  
        # Seleccionamos al mejor individuo de la poblacion  
        self.mejor_individuo_especie(i)  
  
        # Calculamos un porcentaje acumulado para cada individuo que
```



```
servira para el filtro de la ruleta:  
self.porcentaje_acumulado()  
  
# Ponemos a competir a los individuos a traves de la  
seleccion por ruleta:  
Rul = self.ruleta_aleatoria()  
  
# Creamos una nueva poblacion tomando en cuenta el filtro de  
ruleta:  
self.nueva_poblacion(Rul)  
self.poblacion_eval()  
  
# Seleccionamos al mejor individuo de la poblacion  
self.mejor_individuo_especie(i)  
  
# Reproducimos la poblacion (Crossover):  
crom_cross = self.crossover()  
  
# Mutamos la poblacion:  
muta = self.mutacion(crom_cross)  
  
# Creamos una nueva poblacion tomando en cuenta los genes  
mutados:  
self.nueva_poblacion(muta)  
  
if self.contador >= 100:  
    print("Generacion con solucion mas optima: ", i)  
    break  
  
# Tiempo de ejecución del AG  
toc = timeit.default_timer()  
Dtt = (toc - tic)  
[hora, minuto, segundo] = self.TiempoEjecucion(Dtt)  
print(str(hora) + " hrs : " + str(minuto) + " min : " +  
      str(segundo) + " seg")  
  
return self.mejor_individuo, hora, minuto, segundo
```



## CAPITULO IV

# ESTRUCTURA DE LA INTERFAZ GRÁFICA

## IV. ESTRUCTURA DE LA INTERFAZ GRÁFICA DE USUARIO

En este capítulo se hará una explicación detallada de cómo está diseñada la interfaz gráfica en **Qt Designer**.

En los anexos se presenta el código completo para una visión general del mismo.

### 4.1 Qt Designer como herramienta de diseño

#### 4.1.1 ¿Qué es Qt Designer?

Qt Designer es un programa que nos permite desarrollar una interfaz gráfica de usuario (GUI). Nos permite escribir y personalizar ventanas o cuadros de diálogo.

Qt se encuentra disponible para sistemas tipo unix (Linux, BSDs, Unix), para Apple Mac OS X y Windows.

Qt cuenta actualmente con un sistema de triple licencia: GPL v2 / v3 para desarrollar programas libres y de código abierto, licencia QPL para desarrollar aplicaciones comerciales y licencia LGPL gratuita.

#### 4.1.2 ¿Por qué usar Qt Designer?

A continuación, se explican las ventajas de usar Qt Designer:

- **Utiliza el editor visual:** las herramientas de desarrollo visual permiten a los diseñadores y desarrolladores crear una interfaz más hermosa y fluida. También ayuda mucho a la hora de ajustar el tamaño y las características de ciertos elementos de la interfaz de usuario.
- **Utiliza un IDE potente y completo:** *Qt Creator* combina herramientas de edición, depuración, gestión de proyectos, localización y compilación. Tiene todo lo necesario para crear programas para PC y teléfonos inteligentes. Y todos los componentes están diseñados para funcionar juntos sin cambiar el entorno gráfico o la aplicación.
- **Es gratis:** todas las herramientas de Qt se pueden descargar y usar de forma gratuita, y puedes usarlas para crear proyectos comerciales.
- **Multiplataforma:** sin Qt, sería muy difícil desarrollar aplicaciones que se puedan utilizar tanto en Mac como en Windows. Es por eso que algunas aplicaciones populares (como Google Earth o Skype) lo usan.
- **Excelente comunidad en línea:** si se tiene alguna pregunta, siempre habrá una comunidad amigable, bien informada y dispuesta a ayudar.
- **Uso de Python para escribir código reutilizable:** los desarrolladores de Python pueden usar la interfaz de Qt para crear aplicaciones multiplataforma.

## 4.2 Diseño de la Interfaz Gráfica de Usuario

A continuación, se detallan los componentes de la interfaz gráfica de usuario (GUI).

### 4.2.1 Inicialización del programa

Al inicializar el programa aparecerá la ventana que se muestra en la figura 9, su función es la de cargar el **Progressbar** desde cero hasta 100%. El diseño mediante Qt Designer es el siguiente:

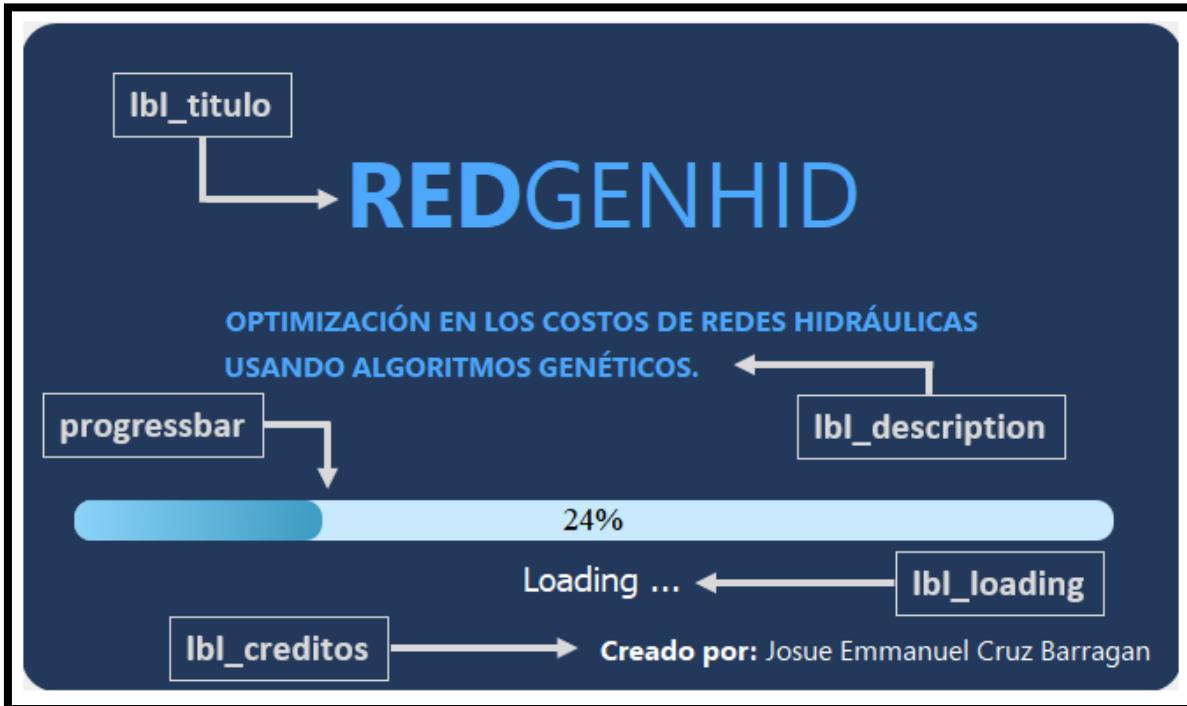


Figura 9. Inicialización del Programa

Los tipos de cada uno de sus componentes son:

- + ***lbl\_titulo, lbl\_descripcion, lbl\_loading, lbl\_creditos:*** QLabel.
- + ***progressbar:*** QProgressBar.

La tarea de cada componente es:

#### A. QLabel

- a. ***lbl\_titulo:*** este *label* muestra el nombre del programa.
- b. ***lbl\_descripcion:*** aquí se describe brevemente el programa.
- c. ***lbl\_loading:*** este *label* solo sirve de etiqueta para avisar que se están cargando los componentes para inicializar el programa.
- d. ***lbl\_creditos:*** muestra el nombre de quien realizó el programa.

#### B. QProgressBar

- a. ***Progressbar:*** muestra como avanza la barra de progreso y su respectivo porcentaje de avance.

#### 4.2.2 Página de inicio

Esta es la página con la que se recibe al usuario, su única función es la de mostrar el nombre del programa y dar una pequeña reseña de su función. El diseño mediante Qt Designer es el siguiente:



Figura 10. Página de inicio.

Los tipos de cada uno de sus componentes son:

- + *lbl\_home, lbl\_database, lbl\_crh, lbl\_adn, lbl\_titulo\_prin, lbl\_descripcion\_prin*: QLabel.
- + *btn\_min, btn\_X*: QPushButton.

La tarea de cada componente es:

##### A. QLabel

- a. *lbl\_home*: este *label* indica la página de inicio.
- b. *lbl\_database*: este *label* indica la página donde se cargan los datos.
- c. *lbl\_crh*: este *label* indica la página donde se realizan las operaciones para solucionar la red de agua potable.
- d. *lbl\_adn*: este *label* indica la página donde se realizan las operaciones del algoritmo genético.
- e. *lbl\_titulo\_prin*: este *label* muestra el nombre del programa.
- f. *lbl\_descripcion\_prin*: este *label* da una breve descripción del programa.

##### B. QPushButton

- a. *btn\_min*: este botón minimiza el programa.
- b. *btn\_X*: este botón cierra el programa.

#### 4.2.3 Menú desplegable

Independientemente de la página en la que nos encontremos, siempre estará un menú colocado en el lado izquierdo de nuestro programa que se desplegará cada que el ratón pase encima de él. Este menú nos permitirá navegar entre las diferentes páginas disponibles. El diseño mediante Qt Designer es el siguiente:



Figura 11. Menú desplegable

Los tipos de cada uno de sus componentes son:

- ✚ **btn\_page\_0, btn\_page\_1, btn\_page\_3, btn\_page\_5:** QPushButton.

La tarea de cada componente es:

A. QPushButton.

- a. **btn\_page\_0:** este botón nos dirige a la página de: **Inicio**.
- b. **btn\_page\_1:** este botón nos dirige a la página de: **Introducir Datos**.
- c. **btn\_page\_3:** este botón nos dirige a la página de: **Análisis de Redes Hidráulicas**.
- d. **btn\_page\_5:** este botón nos dirige a la página de: **Algoritmo Genético**.

#### 4.2.4 Introducir datos

Esta página (figura 12) se compone de dos QGroupBox que se describen a continuación:

Los tipos de cada uno de sus componentes son:

⊕ **groupBox, groupBox\_14: QGroupBox.**

La tarea de cada componente es:

**A. QGroupBox**

- groupBox:** aquí se introducen los datos necesarios para analizar las redes de agua potable. En este componente se proporciona la información relevante a la red, en la figura 14 a la 17 se explica a detalle cada componente dentro de este grupo.
- groupBox\_14:** en este componente se visualiza la información proporcionada relevante a la red, en la figura 18 se explica a detalle cada componente dentro de este grupo.

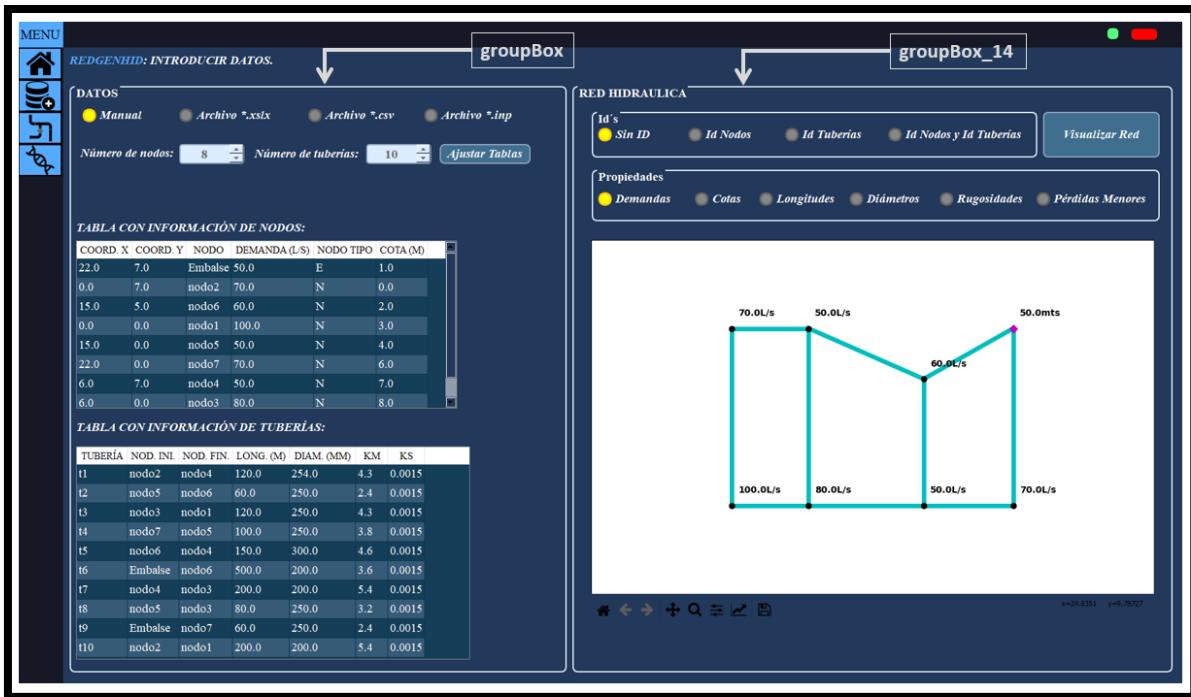


Figura 12. Introducir datos

#### 4.2.4.1 QGroupBox: DATOS

Para el **groupBox** los tipos de cada uno de sus componentes, mostrados en la figura 13, son:

- ⊕ **rb\_manual, rb\_xlsx, rb\_csv, rb\_inp:** QRadioButton.
- ⊕ **tbl\_nod, tbl\_tub:** QTableWidget.

La tarea de cada componente es:

**A. QRadioButton**

- rb\_manual:** este muestra la interfaz como la seleccionada con el rectángulo rojo, figura 14.

- b. ***rb\_xslx***: este cambia la interfaz (seleccionada con el rectángulo rojo) a como se muestra en la figura 15.
- c. ***rb\_csv***: este cambia la interfaz (seleccionada con el rectángulo rojo) a como se muestra en la figura 16.
- d. ***rb\_inp***: este cambia la interfaz (seleccionada con el rectángulo rojo) a como se muestra en la figura 17.

## B. QTableWidget

- a. ***tbl\_nod***: en esta tabla se carga toda la información necesaria (coordenadas, cotas, demandas, etc.) de los nodos en la red a analizar.
- b. ***tbl\_tub***: en esta tabla se carga toda la información necesaria (longitud, diámetro, rugosidad, etc.) de las tuberías en la red a analizar.

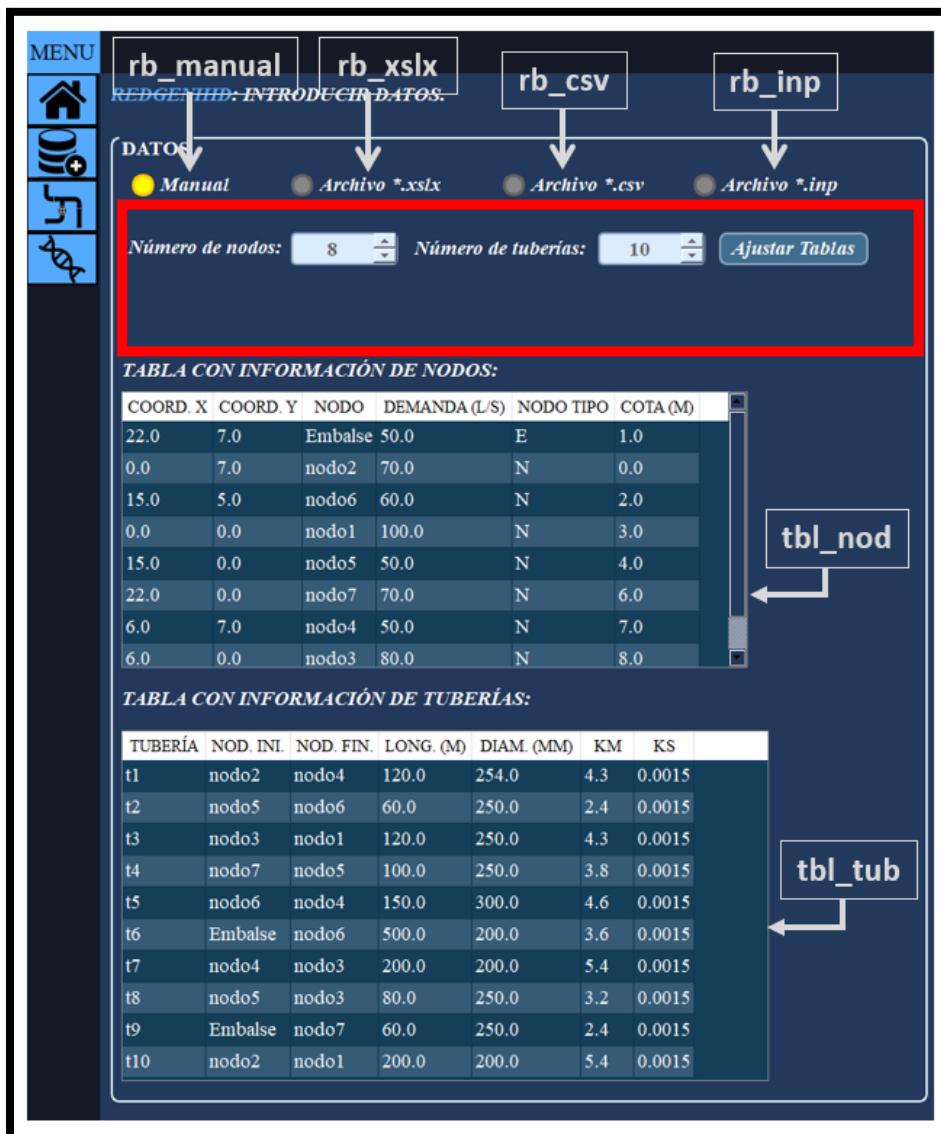


Figura 13. Componentes Generales de la Sección de Datos

## Tipo manual

Si se selecciona el ***rb\_manual***, la interfaz se muestra como:

The screenshot shows a user interface titled 'DATOS'. At the top, there are four radio buttons: 'Manual' (selected), 'Archivo \*.xlsx', 'Archivo \*.csv', and 'Archivo \*.inp'. Below these are two QSpinBoxes: 'Número de nodos:' set to 8 and 'Número de tuberías:' set to 10. To the right of these spin boxes is a blue button labeled 'Ajustar Tablas'. Below the spin boxes are two QLineEdit components: 'txt\_manual\_nod' and 'txt\_manual\_tub'. To the right of these is a blue QPushButton labeled 'btn\_load\_info'. Arrows point from the labels 'txt\_manual\_nod' and 'txt\_manual\_tub' to their respective QLineEdit fields, and another arrow points from 'btn\_load\_info' to its button.

Figura 14. Opción Manual

Los tipos de cada uno de sus componentes son:

- + ***txt\_manual\_nod*, *txt\_manual\_tub***: QSpinBox.
- + ***btn\_load\_info***: QPushButton.

La tarea de cada componente es:

- A. ***QSpinBox*.**
  - txt\_manual\_nod***: aquí se debe indicar cuantos nodos tiene la red.
  - txt\_manual\_tub***: aquí se debe indicar cuantas tuberías tiene la red.
- B. ***QPushButton*.**
  - btn\_load\_info***: este botón limpia las tablas de nodos (***tbl\_nod***) y tuberías (***tbl\_nod***), y después, le pone el número de renglones en cada tabla.

## Tipo archivo de Excel

Si se selecciona el ***rb\_xlsx***, la interfaz se muestra como:

The screenshot shows a user interface titled 'DATOS'. At the top, there are four radio buttons: 'Manual' (unchecked), 'Archivo \*.xlsx' (selected), 'Archivo \*.csv', and 'Archivo \*.inp'. Below these is a label 'Nombre de la Hoja con Información de:'. Below the radio buttons are two QLineEdit components: 'Nodos:' containing 'nodos\_xlsx' and 'Tuberías:' containing 'tuberias\_xlsx'. To the right of these is a blue button labeled 'Cargar Excel'. Below the QLineEdit components are two more: 'txt\_excel\_nod' and 'txt\_excel\_tub'. To the right of these is a blue QPushButton labeled 'btn\_load\_info'. Arrows point from the labels 'txt\_excel\_nod' and 'txt\_excel\_tub' to their respective QLineEdit fields, and another arrow points from 'btn\_load\_info' to its button.

Figura 15. Opción archivo de Excel

Los tipos de cada uno de sus componentes son:

- + ***txt\_excel\_nod*, *txt\_excel\_tub***: QLineEdit.
- + ***btn\_load\_info***: QPushButton.



La tarea de cada componente es:

**A. QLineEdit**

- txt\_excel\_nod:** aquí se debe indicar el nombre de la pestaña en el archivo de *Excel* donde se encuentran los datos de los nodos que tiene la red.
- txt\_excel\_tub:** aquí se debe indicar el nombre de la pestaña en el archivo de *Excel* donde se encuentran los datos de las tuberías que tiene la red.

**B. QPushButton**

- btn\_load\_info:** este botón limpia las tablas de nodos (**tbl\_nod**) y tuberías (**tbl\_tub**), después abre el buscador de archivos para seleccionar el archivo de *Excel*, donde se tengan los datos de la red y vacía los datos correspondientes del archivo de Excel en las tablas correspondientes.

## Tipo archivo CSV

Si se selecciona el **rb\_csv**, la interfaz se muestra como:

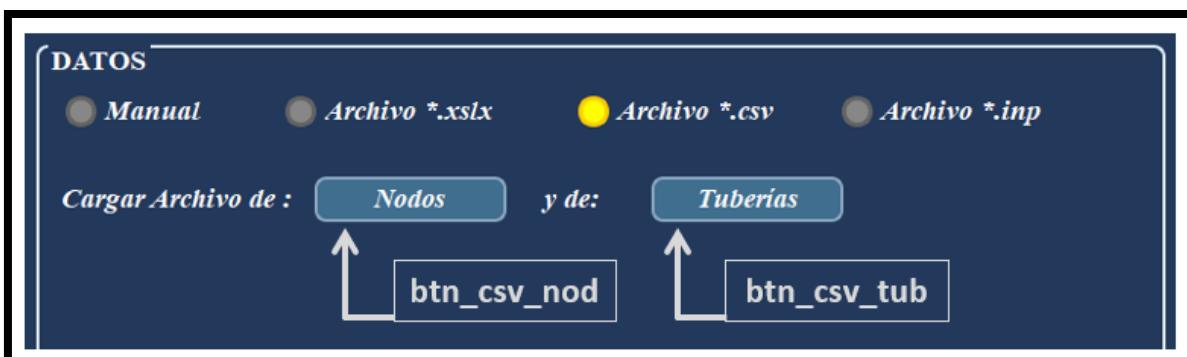


Figura 16. Opción archivo CSV

Los tipos de cada uno de sus componentes son:

- ⊕ **btn\_csv\_nod, btn\_csv\_tub:** QPushButton.

La tarea de cada componente es:

**A. QPushButton**

- btn\_csv\_nod:** este botón limpia la tabla de nodos (**tbl\_nod**), después abre el buscador de archivos para seleccionar el archivo **\*csv** donde se tengan los datos de los nodos de la red y los vacía en la tabla de nodos.
- btn\_csv\_tub:** este botón limpia la tabla de tuberías (**tbl\_tub**), después abre el buscador de archivos para seleccionar el archivo **\*csv** donde se tengan los datos de las tuberías de la red y los vacía en la tabla de tuberías.



## Tipo archivo de Epanet

Si se selecciona el **rb\_inp**, la interfaz se muestra como:

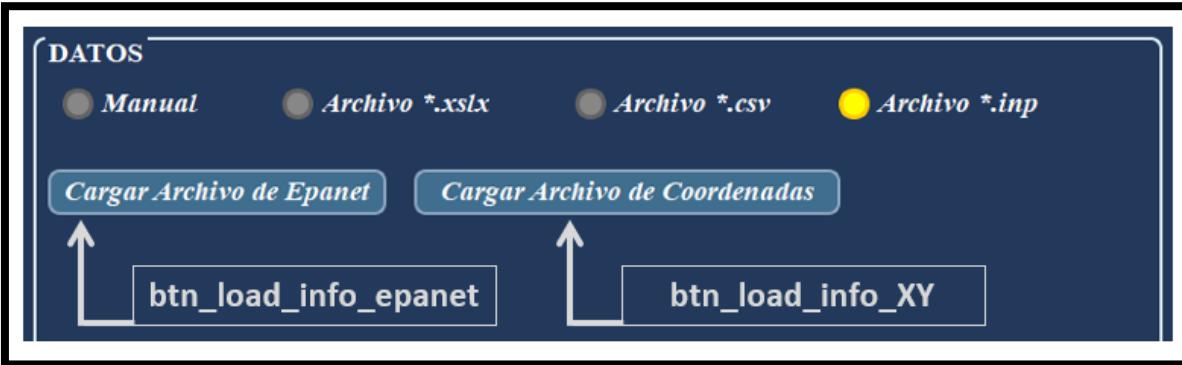


Figura 17. Opción archivo de Epanet

Los tipos de cada uno de sus componentes son:

- + **btn\_load\_info\_epanet**: QPushButton.
- + **btn\_load\_info\_XY**: QPushButton.

La tarea de cada componente es:

### A. QPushButton

- a. **btn\_load\_info\_epanet**: este botón limpia las tablas de nodos (**tbl\_nod**) y tuberías (**tbl\_tub**), después abre el buscador de archivos para seleccionar el archivo de *Epanet* donde se tengan los datos de la red y vacía los datos correspondientes del archivo de *Epanet* en las tablas correspondientes.
- b. **btn\_load\_info\_XY**: este botón abre el buscador de archivos para seleccionar el archivo **\*csv** donde se tengan los datos de las coordenadas de los nodos y el embalse, y vacía los datos correspondientes en las columnas de **COORD. X** y **COORD. Y** en la tabla de nodos (**tbl\_nod**).

#### 4.2.4.2 QGroupBox: RED HIDRAULICA

Para el **groupBox\_14** los tipos de cada uno de sus componentes, mostrados en la figura 18, son:

- + **rb\_sinID, rb\_idNodos, rb\_idTuberias, rb\_idNodTub, rb\_Demandas, rb\_Cotas, rb\_Long, rb\_Diam, rb\_ks, rb\_km**: QRadioButton.
- + **btn\_visualizar**: QPushButton.
- + **Widget\_Visualizar\_RH**: QMatplotlibWidget11\_sin.

La tarea de cada componente es:

### A. QRadioButton

- a. **rb\_sinID**: oculta la etiqueta de los nodos y las tuberías en la gráfica.
- b. **rb\_idNodos**: muestra la etiqueta de los nodos en la gráfica.

- c. ***rb\_idTuberias***: muestra la etiqueta de las tuberías en la gráfica.
- d. ***rb\_idNodTub***: muestra la etiqueta de los nodos y las tuberías en la gráfica.
- e. ***rb\_Demandas***: muestra las demandas de los nodos en la gráfica.
- f. ***rb\_Cotas***: muestra las cotas de los nodos en la gráfica.
- g. ***rb\_Long***: muestra las longitudes de las tuberías en la gráfica.
- h. ***rb\_Diam***: muestra los diámetros de las tuberías en la gráfica.
- i. ***rb\_ks***: muestra las rugosidades de las tuberías en la gráfica.
- j. ***rb\_km***: muestra las pérdidas por fricción de las tuberías en la gráfica.

## B. QPushButton

- a. ***btn\_visualizar***: permite visualizar la red hidráulica en la gráfica.

## C. QMatplotlibWidget11\_sin

- a. ***Widget\_Visualizar\_RH***: este componente permite crear gráficos, en este caso, permite dibujar la red hidráulica en función de los datos suministrados por el usuario.

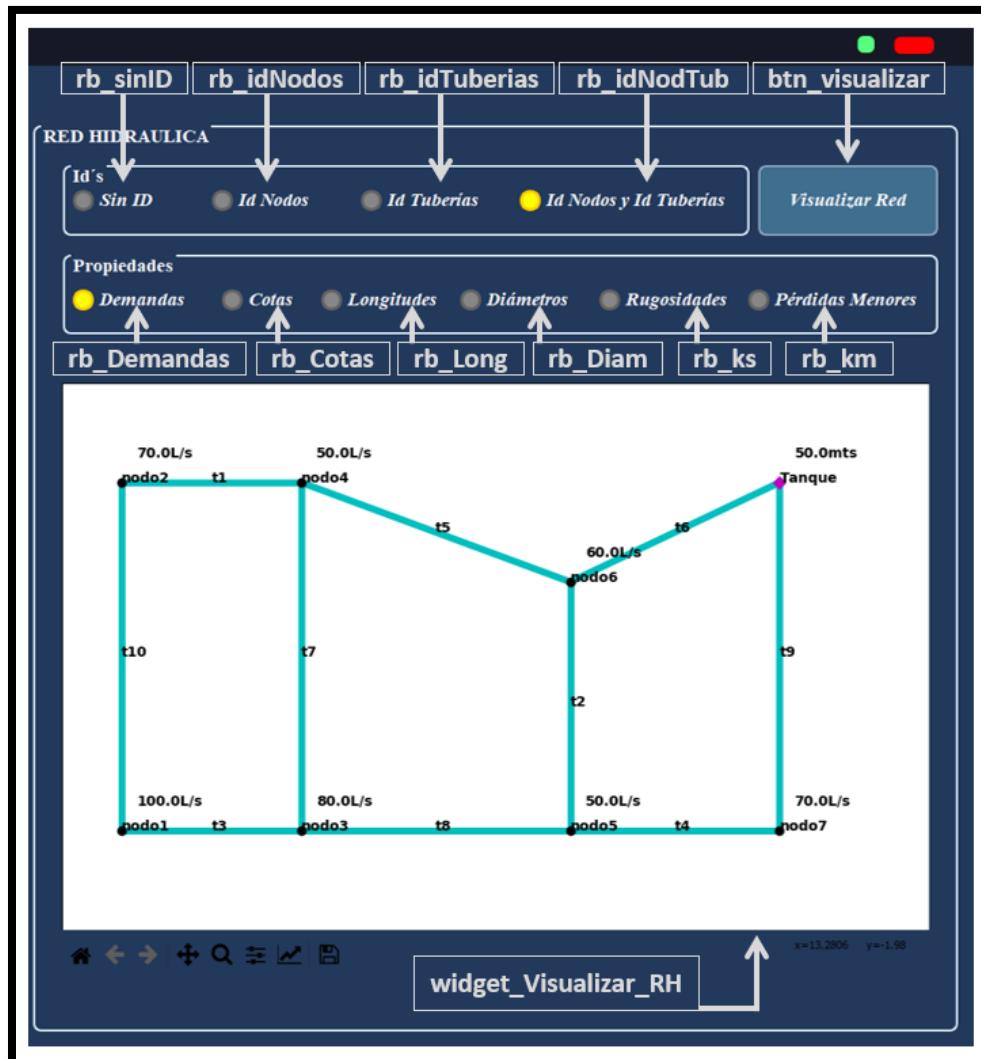


Figura 18. Componentes Generales de la Sección de Red hidráulica

#### 4.2.5 Análisis de Redes Hidráulicas

En esta página se muestran los resultados del análisis de la red hidráulica utilizando el método del gradiente. El diseño mediante Qt Designer es el siguiente:

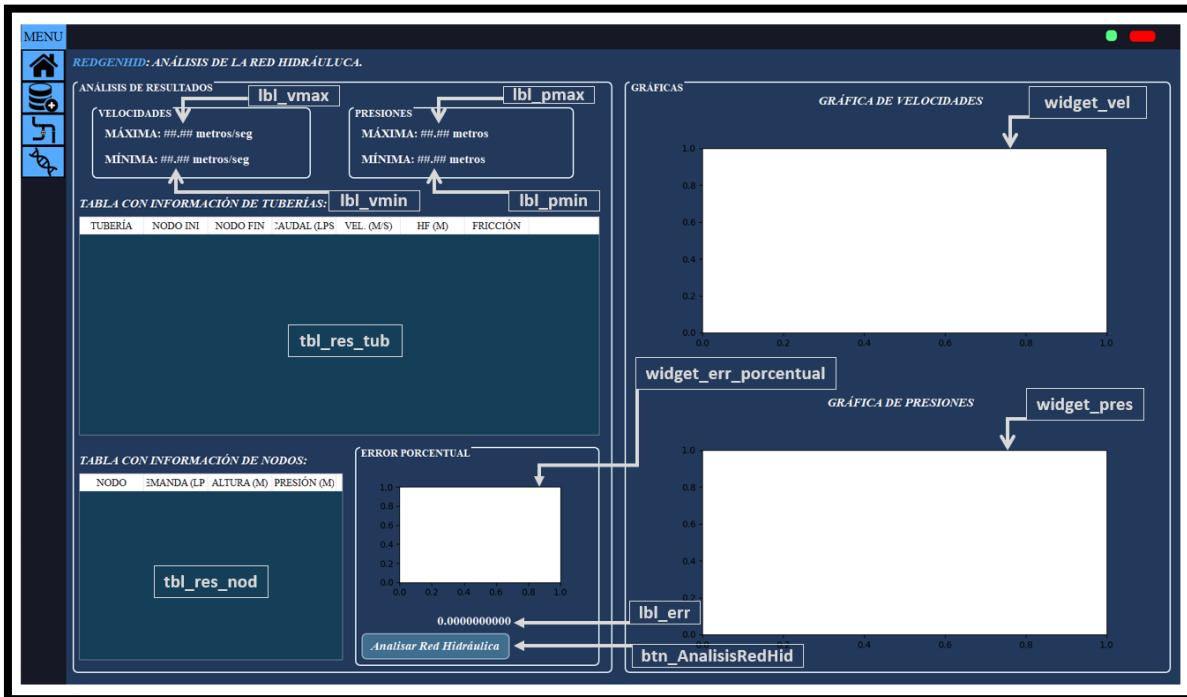


Figura 19. Análisis de Redes Hidráulicas

Los tipos de cada uno de sus componentes son:

- + **lbl\_vmax, lbl\_vmin, lbl\_pmax, lbl\_pmin, lbl\_err**: QLabel.
- + **btn\_AnalisisRedHid**: QPushButton.
- + **tbl\_res\_nod, tbl\_res\_tub**: QTableWidget.
- + **widget\_err\_porcentual, widget\_vel, widget\_pres**: QMatplotlibWidget.

La tarea de cada componente es:

##### A. **QLabel**

- lbl\_vmax, lbl\_vmin**: retorna la velocidad máxima y mínima de la red que se obtiene al ejecutar el algoritmo del Gradiente Hidráulico.
- lbl\_pmax, lbl\_pmin**: retorna la presión máxima y mínima de la red que se obtiene al ejecutar el algoritmo del Gradiente Hidráulico.
- lbl\_err**: retorna el error relativo porcentual al ejecutar el algoritmo del Gradiente Hidráulico. Como tolerancia se considera que la ejecución fue correcta si el error es menor a  $10^{-6}$ .

##### B. **QPushButton**

- btn\_AnalisisRedHid**: ejecuta el algoritmo de Gradiente Hidráulico. Grafica las presiones, velocidades, error porcentual y da los



resultados del análisis en las tablas de tuberías (**tbl\_res\_tub**) y nodos (**tbl\_res\_nod**).

#### C. QTableWidget

- a. **tbl\_res\_nod**: en esta tabla se muestran los resultados del algoritmo del Gradiente Hidráulico, referente a los nodos. Como outputs se obtienen sus respectivas demandas, alturas y presiones de cada nodo.
- b. **tbl\_res\_tub**: en esta tabla se muestran los resultados del algoritmo del Gradiente Hidráulico, referente a las tuberías. Como outputs se obtienen sus respectivos caudales, velocidades, perdidas por fricción y factor de fricción de cada tubería.

#### D. QMatplotlibWidget11\_sin

- a. **widget\_err\_porcentual**: este componente permite crear gráficos, en este caso, permite dibujar la curva de error que muestra como fue convergiendo el programa hasta llegar a la mejor solución.
- b. **widget\_pres**: este gráfico de barras muestra las presiones de todos los nodos para una rápida visualización. Permite identificar de manera visual cuales son los nodos que no cumplen con las presiones máximas y mínimas de diseño.
- c. **widget\_vel**: este gráfico de barras muestra las velocidades de todas las tuberías para una rápida visualización. Permite identificar de manera visual cuales son las tuberías que no cumplen con las velocidades máximas y mínimas de diseño.

#### 4.2.6 Algoritmo Genético

En esta página (figura 20) se debe dar información adicional al programa antes de poder ejecutar el algoritmo genético, el cual nos dará los diámetros óptimos para la red hidráulica.

Los tipos de cada uno de sus componentes son:

- + **btn\_AG, btn\_crear\_tabla, btn\_guardar\_diam\_temp, btn\_exportar\_diam, btn\_cargar\_diam**: QPushButton.
- + **txt\_tam\_pob, txt\_num\_gen, txt\_tasa\_mut, txt\_VDmax, txt\_VDmin, txt\_PDmax, txt\_PDmin, txt\_num\_diametros**: QSpinBox.
- + **lbl\_mejor\_gen, lbl\_CH, lbl\_CC, lbl\_res\_vmax, lbl\_res\_vmin, lbl\_res\_pmax, lbl\_res\_pmin, lbl\_R2, lbl\_time**: QLabel.
- + **tbl\_Dcom, tbl\_mejor\_diam, tbl\_diam\_temp**: QTableWidget.
- + **rb\_lineal, rb\_potencial, rb\_exponencial**: QRadioButton.
- + **cb\_diam\_cargar**: QComboBox.
- + **txt\_nom\_export**: QLineEdit.
- + **widget\_gen\_vs\_costo**: QMatplotlibWidget11\_sin.

La tarea de cada componente es:



#### A. QPushButton

- a. **btn\_AG:** ejecuta el algoritmo genético y muestra el progressbar circular que se muestra en la figura 21.
- b. **btn\_crear\_tabla:** crea una tabla de dos columnas con  $n$  renglones, dados por el componente: **txt\_num\_diametros**. El usuario puede llenar la tabla con el diámetro comercial a usar para el análisis y su respectivo costo por unidad por metro lineal.
- c. **btn\_guardar\_diam\_temp:** envía los resultados del algoritmo genético (generación, costo de construcción, costo hidráulico y diámetros) a la tabla **tbl\_diam\_temp**.
- d. **btn\_exportar\_diam:** exporta los datos de la tabla **tbl\_diam\_temp** en un archivo **\*.csv**. Este archivo se guarda en una carpeta llamada **“RedGenHid”** ubicada en el disco **“C:/”**. Si la carpeta no existe, se crea una, siempre con dirección **“C:/ RedGenHid”**.
- e. **btn\_cargar\_diam:** carga la combinación de diámetros óptimos (que el usuario seleccione) a la tabla **tbl\_tub**, para volverlos a evaluar.

#### B. QSpinBox

- a. **txt\_tam\_pob:** el usuario indica el tamaño de la población. El tamaño de la población debe ser un número **k** par, en caso contrario, el programa lo hace par usando **k+1**.
- b. **txt\_num\_gen:** el usuario indica el número de generaciones que tendrá el algoritmo. Debe tenerse en cuenta que si el número de generaciones es muy grande, el programa puede tardar mucho en converger.
- c. **txt\_tasa\_mut:** el usuario indica la tasa de mutación. Al igual que en la naturaleza, la probabilidad de que un individuo mute es muy baja, por esta razón, se recomienda usar tasas de mutación menores al 5%.
- d. **txt\_VDmax, txt\_VDmin:** el usuario indica la velocidad máxima y mínima de diseño. Estas velocidades se usarán para acotar las velocidades en las tuberías que deben correr en la red y ayudaran a encontrar la mejor combinación de diámetros que satisfagan las características de la red de agua potable.
- e. **txt\_PDmax, txt\_PDmin:** el usuario indica la presión máxima y mínima de diseño. Estas presiones se usarán para acotar las presiones nodales que deben correr en la red y ayudaran a encontrar la mejor combinación de diámetros que satisfagan las características de la red de agua potable.
- f. **txt\_num\_diametros:** el usuario indica el número de diámetros comerciales a usar en la red hidráulica a evaluar.

#### C. QLabel

- a. **lbl\_mejor\_gen:** retorna como resultado la generación en la que se encontró a la mejor combinación de diámetros que soluciona el problema.



- b. **Ibl\_CH:** retorna el costo hidráulico de la red. La red óptima es la que tiene un costo hidráulico de cero.
- c. **Ibl\_CC:** retorna el costo de construcción de la red. Si el costo hidráulico de la red es cero, el costo de construcción de la red es el óptimo.
- d. **Ibl\_res\_vmax, Ibl\_res\_vmin:** retorna la velocidad máxima y mínima de la red que se obtiene al usar la mejor combinación de diámetros que soluciona el problema. Si el costo hidráulico de la red es cero, estas velocidades son las óptimas.
- e. **Ibl\_res\_pmax, Ibl\_res\_pmin:** retorna la presión máxima y mínima de la red que se obtiene al usar la mejor combinación de diámetros que soluciona el problema. Si el costo hidráulico de la red es cero, estas presiones son las óptimas.
- f. **Ibl\_R2:** retorna el R2. El R-cuadrado es una medida estadística de qué tan cerca están los datos de la línea de regresión ajustada. También se conoce como coeficiente de determinación, o coeficiente de determinación múltiple si se trata de regresión múltiple.
- g. **Ibl\_time:** retorna el tiempo que tardó el algoritmo genético en ejecutarse. Utiliza el formato: **## hrs : ## min : ## seg.**

#### D. QTableWidget

- a. **tbl\_Dcom:** es la tabla que debe llenar el usuario para que el programa sepa qué diámetros comerciales usar en el problema combinatorio.
- b. **tbl\_mejor\_diam:** en esta tabla se muestra el mejor diámetro comercial a usar asignado a su respectiva tubería. Si el costo hidráulico de la red es cero, estos diámetros son los óptimos.
- c. **tbl\_diam\_temp:** guarda temporalmente de forma ordenada la generación, el costo de construcción, costo hidráulico y los mejores diámetros que se obtienen en la mejor generación cada que se presiona el botón **btn\_guardar\_diam\_temp.**

#### E. QRadioButton

- a. **rb\_lineal:** se usa una regresión lineal para hallar una ecuación Diámetro-Costo y no usar datos discretos.
- b. **rb\_potencial:** se usa una regresión potencial para hallar una ecuación Diámetro-Costo y no usar datos discretos.
- c. **rb\_exponencial:** se usa una regresión exponencial para hallar una ecuación Diámetro-Costo y no usar datos discretos.

#### F. QCombobox

- a. **cb\_diam\_cargar:** permite al usuario seleccionar la combinación de diámetros a cargar en la tabla **tbl\_tub**, para volverlos a evaluar.

#### G. QLineEdit

- a. **txt\_nom\_export:** permite al usuario ponerle el nombre al archivo **\*csv** donde se guardarán los diámetros que estaban en la tabla **tbl\_diam\_temp.**

## H. QMatplotlibWidget11\_sin

- a. **widget\_gen\_vs\_costo:** permite dibujar la curva de Generación-Costo que muestra como fue convergiendo el programa hasta llegar a la solución óptima para el problema de redes.

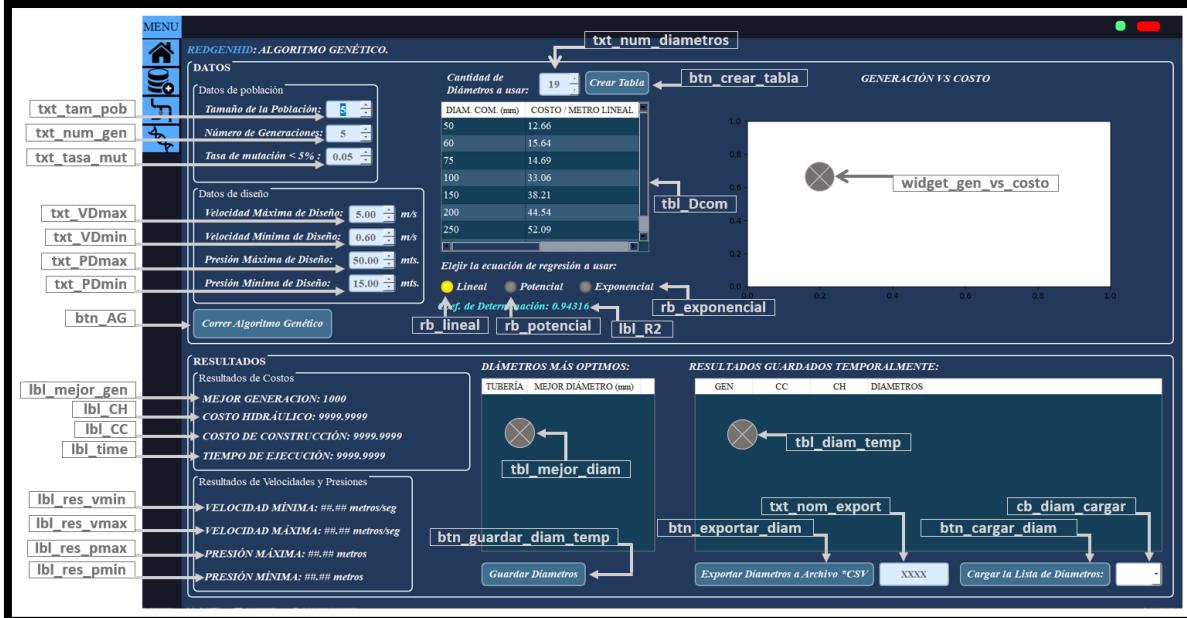


Figura 20. Algoritmo Genético

### 4.2.6.1 Circular progressbar

Debido a que el algoritmo genético tiende a demorar en su ejecución, se informa al usuario el progreso del algoritmo usando un **progressbar**, de esta forma se sabe cuánto porcentaje de avance lleva. El diseño mediante Qt Designer es el siguiente:

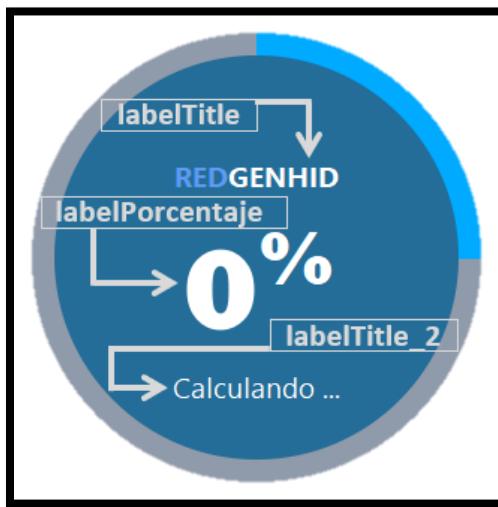


Figura 21. Circular progressbar

Los tipos de cada uno de sus componentes son:



■ *labelTitle, labelPorcentaje, labelTitle\_2: QLabel.*

La tarea de cada componente es:

A. **QLabel**

- a. *labelTitle*: muestra el nombre del programa.
- b. *btn\_crear\_tabla*: muestra el porcentaje de avance del algoritmo genético.
- c. *labelTitle\_2*: informa que se está ejecutando el algoritmo genético..

# CAPITULO V

## EJEMPLO DE APLICACIÓN DEL PROGRAMA REDGENHID.

### V. EJEMPLO DE APLICACIÓN DEL PROGRAMA

#### 5.1 Red Hidráulica Propuesta

##### 5.1.1 Red del municipio de Tezoyuca en el Estado de México

Para verificar el uso de este programa, se utilizará la red del municipio de Tezoyuca (figura 22). Esta red se encuentra en el Estado de México. Originalmente está constituida por 3 circuitos cerrados, 21 nodos de demanda, 23 tuberías de conducción y un único embalse (nodo 1) de altura constante de 20m, (ver tabla 2, 3 y 4). La presión mínima requerida en los nodos es de 10 m.c.a. La lista de diámetros disponibles es la siguiente: 2, 2 ½, 3, 4, 6, 8, 10 y 12 pulgadas. La red de Tezoyuca está planteada para solucionarse por la ecuación de Hazen-Williams, con un coeficiente de rugosidad C=130

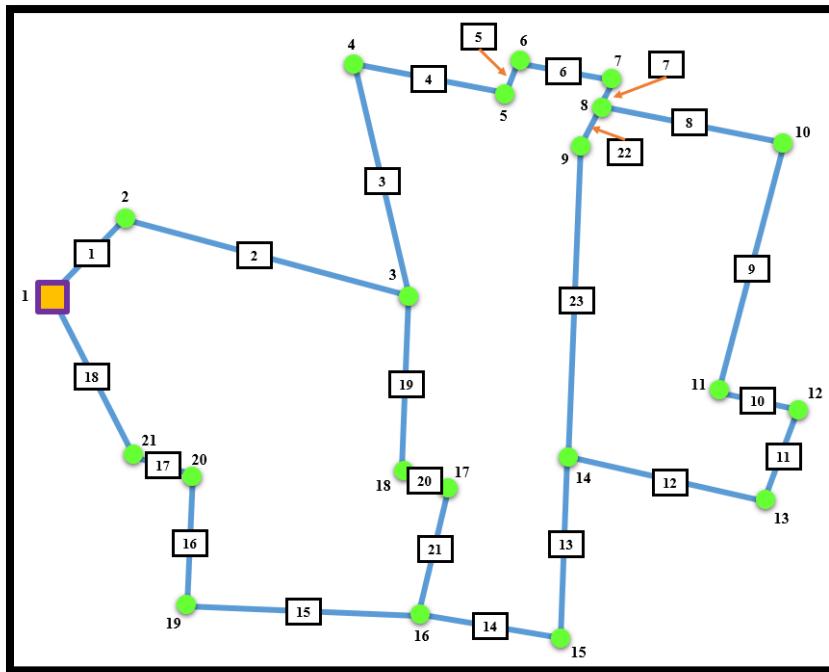


Figura 22. Red hidráulica del municipio de Tezoyuca



### 5.1.2 Modificaciones a la Red Original

Si quisieramos resolver esta red por fuerza bruta, es decir, tratando de comprobar cada combinación de diámetros posible por tubería en la computadora hasta hallar la óptima, tendríamos que lidiar con  $8^{23} = 5.902958104 \times 10^{20}$  combinaciones. Suponiendo que la computadora pueda calcular cada combinación en  $1\mu s$ , realizar esta tarea le tomaría aproximadamente 18,718 milenios. Teniendo en cuenta ese número extraordinariamente alto, resulta claro que el problema debe ser resuelto de manera diferente.

Las modificaciones que se harán a la red original de Tezoyuca son las siguientes:

- El uso del método del gradiente junto con las ecuaciones de Darcy-Weisbach y Colebrook-White, en lugar de la ecuación de Hazen Williams y su respectivo coeficiente de rugosidad.
- El valor de rugosidad absoluta considerado es de  $k_s = 1.5 \times 10^{-6} m$ .
- Se desprecian las pérdidas menores en las tuberías,  $k_m = 0$ .
- Se transportará agua a  $15^\circ C$  con una viscosidad del fluido de  $\nu = 1.14 \times 10^{-6} m^2/s$ .
- Se considerará para fines didácticos una presión mínima y máxima requerida en los nodos de 10 m.c.a. y 50 m.c.a. respectivamente.
- Se considerará para fines didácticos velocidades mínimas y máximas requeridas en las tuberías de 0.5 m/s y 5 m/s respectivamente.

ID	Longitud	ID	Longitud	ID	Longitud
1	202.7	9	352.68	17	84.43
2	392.4	10	110.47	18	321.21
3	317.83	11	138.9	19	242.66
4	211.17	12	270.38	20	69.1
5	45.88	13	248.5	21	179.04
6	132.9	14	194.63	22	53.51
7	39.98	15	312.78	23	424.62
8	255.93	16	184.75		

Tabla 2. Longitud de las tuberías, metros

Nodo	Demanda Base (m <sup>3</sup> /hr)	Cota (m)	Nodo	Demanda Base (m <sup>3</sup> /hr)	Cota (m)	Nodo	Demanda Base (m <sup>3</sup> /hr)	Cota (m)
1	----	2329.885	8	13.80	2302.000	15	17.50	2288.502
2	31.50	2333.461	9	18.88	2303.421	16	27.11	2295.400
3	37.63	2309.516	10	24.03	2277.000	17	9.80	2302.458
4	20.89	2312.584	11	18.29	2283.000	18	24.99	2303.927
5	10.15	2308.474	12	9.85	2278.055	19	19.65	2301.432
6	7.06	2309.763	13	16.16	2280.634	20	10.63	2314.993
7	6.83	2301.312	14	37.26	2302.010	21	16.02	2318.983

Tabla 3. Datos de los nodos

Diámetros (in)	Diámetros (mm)	Costo (\$/m)
2	50	12.66
2 1/2	60	15.64
3	75	14.69
4	100	33.06
6	150	38.21
8	200	44.54
10	250	52.09
12	300	60.08

Tabla 4. Datos de los costos

### 5.1.3 Ingresar Datos al Programa REDGENHID.

Los datos se ingresan en el programa **REDGENHID** (Figura 23) que fue diseñado para calcular y diseñar redes de agua de forma óptima usando AG.

Permite el modelado de embalses, nodos y tuberías. Fue ejecutado en una computadora con las siguientes características:

- Sistema Operativo: Windows 10
- Procesador: Inter® Core™ i7-9750H CPU @ 2.60GHz
- Memoria RAM: 8GB
- Sistema: 64 bits.

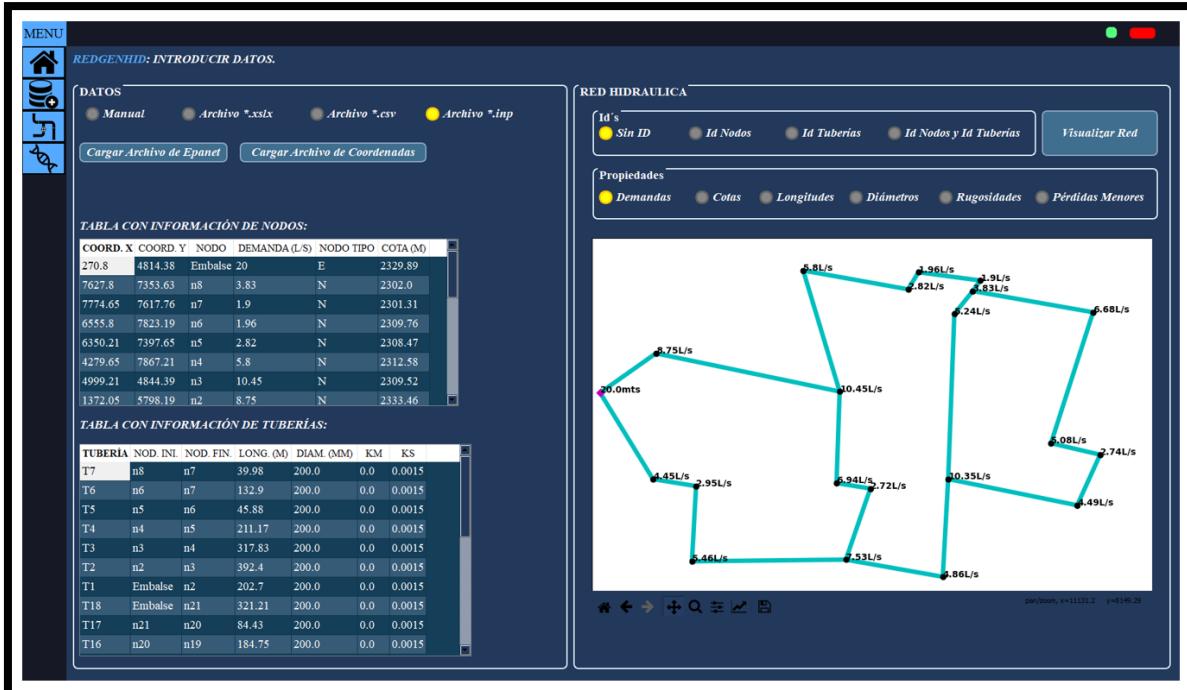


Figura 23. Introducir Datos

Para la red en estudio, se empleó la optimización según 4 objetivos, por lo cual se procedió a modelar la red, ingresar los datos respectivos e iniciar el proceso de diseño.

#### 5.1.4 Ejecución del Algoritmo Genético.

Los parámetros para la optimización del AG introducidos en **REDGENHID** (Figura 24) fueron los siguientes:

- Tamaño de la población: 20.
- Número de generaciones: 500.
- Tasa de mutación: 0.05.
- Velocidad mínima de diseño: 0.5 m/s.
- Velocidad máxima de diseño: 5 m/s.
- Presión mínima de diseño: 10 m.c.a.
- Presión máxima de diseño: 50 m.c.a.
- Ecuación de regresión: Potencial

The screenshot shows the REDGENHID software interface with the following parameters set:

- DATOS** (Data):
  - Datos de población:
    - Tamaño de la Población: 20
    - Número de Generaciones: 500
    - Tasa de mutación < 5% : 0.05
  - Datos de diseño:
    - Velocidad Máxima de Diseño: 5.00 m/s
    - Velocidad Mínima de Diseño: 0.50 m/s
    - Presión Máxima de Diseño: 50.00 mts.
    - Presión Mínima de Diseño: 10.00 mts.
- Cantidad de Diámetros a usar: 8 (Create Table button)
- Table: DIAM. COM. (mm) COSTO / METRO LINEAL
 

DIAM. COM. (mm)	COSTO / METRO LINEAL
60	15.64
75	14.69
100	33.06
150	38.21
200	44.54
250	52.09
300	60.08
- Elegir la ecuación de regresión a usar:
  - Lineal
  - Potencial
  - Exponencial
- Coef. de Determinación: 0.98771
- Correr Algoritmo Genético

Figura 24. Parámetros introducidos en el AG

Se ejecutó el algoritmo ocho veces para obtener diferentes soluciones optimas. Para cada solución encontrada (figura 25) se guardó su correspondiente costo de construcción, costo hidráulico, diámetros y el número de generaciones que transcurrieron hasta llegar al óptimo más cercano:



RESULTADOS GUARDADOS TEMPORALMENTE:				
	GEN	CC	CH	DIAMETRO
1	241	151466.42	0	[60.0, 60.0, 50.0, 50.0, 100.0, 100.0, 200.0, 200.0, 200.0, 200.0, 50.0, 50.0, 50.0,
2	125	168646.2	1.23	[200.0, 250.0, 300.0, 150.0, 200.0, 250.0, 200.0, 200.0, 300.0, 250.0, 100.0, 100.0,
3	163	146937.73	0	[100.0, 100.0, 100.0, 250.0, 150.0, 200.0, 250.0, 150.0, 150.0, 100.0, 100.0, 75.0,
4	107	181667.24	1.9	[250.0, 200.0, 150.0, 250.0, 250.0, 250.0, 200.0, 300.0, 100.0, 100.0, 150.0,
5	32	206728.2	2.46	[150.0, 100.0, 100.0, 300.0, 250.0, 75.0, 300.0, 250.0, 150.0, 250.0, 60.0, 200.0, 1
6	159	177057.78	1.89	[75.0, 60.0, 150.0, 100.0, 200.0, 300.0, 150.0, 200.0, 300.0, 300.0, 75.0, 100.0, 10
7	69	142356.39	0	[100.0, 250.0, 150.0, 200.0, 200.0, 250.0, 250.0, 75.0, 75.0, 75.0, 75.0, 60.0, 75.0]
8	46	157190.98	1.88	[150.0, 75.0, 100.0, 200.0, 100.0, 200.0, 200.0, 150.0, 150.0, 200.0, 200.0, 50.0, 7

Figura 25. Mejores soluciones encontradas en cada generación

Como se observa en la figura 25, la mejor solución de este conjunto de soluciones fue la número 7 pues su costo hidráulico es de **\$0** y su costo de construcción fue de **\$142, 356.39**, el menor de todos. A continuación, se muestra una tabla con el tiempo que tardó cada ejecución del AG:

Ejecución	Tiempo de Ejecución	Generación	Costo Const.	Costo Hid.
1	00 hr : 56 min : 05seg	241	\$151,466.42	\$0.00
2	00 hr : 29 min : 56seg	125	\$168,646.20	\$1.23
3	00 hr : 24 min : 13seg	163	\$146,937.73	\$0.00
4	00 hr : 30 min : 32seg	107	\$181,667.24	\$1.90
5	00 hr : 15 min : 17seg	32	\$206,728.20	\$2.46
6	00 hr : 35 min : 55seg	159	\$177,057.78	\$1.89
7	00 hr : 38 min : 43seg	69	\$142,356.39	\$0.00
8	00 hr : 10 min : 53seg	46	\$157,190.98	\$1.88

Tabla 5. Tiempo de ejecución del AG para las diferentes soluciones.

Debido a que en la figura 25 no son visibles todos los diámetros para cada solución encontrada en el programa, en la tabla 6 se muestran los diámetros obtenidos en cada generación asociados con su respectiva tubería y número de ejecución.

Como se mencionó anteriormente, la mejor solución es la número 7, cuyo tiempo de ejecución fue de **38 min con 43 segundos**, y se encontró una solución óptima en la generación **69** con un costo de construcción de **\$142,356.39**. La interpretación de que el costo hidráulico fuera **cero** se debe a que el algoritmo genético encontró una combinación de diámetros (tabla 6), lo suficientemente satisfactoria (**sin**



penalizaciones) como para que se respetaran los objetivos propuestos de velocidad y presión establecidos por el usuario.

Tuberia	Generación							
	241	125	163	107	32	159	69	46
	Ejecución							
	1	2	3	4	5	6	7	8
T1	200	200	250	250	300	150	250	200
T2	100	250	200	250	75	300	250	200
T3	100	200	150	250	250	200	200	100
T4	50	150	250	250	300	100	200	200
T5	50	300	100	150	100	150	150	100
T6	60	250	100	200	100	60	250	75
T7	60	200	100	250	150	75	100	150
T8	150	75	75	50	150	75	60	75
T9	100	75	75	60	200	50	50	60
T10	50	100	60	60	100	100	75	75
T11	50	100	75	150	200	100	60	50
T12	50	100	100	100	60	75	75	200
T13	250	100	200	250	300	300	100	150
T14	150	150	150	100	150	100	100	150
T15	200	100	200	100	200	250	50	200
T16	200	250	100	100	250	300	75	200
T17	200	300	150	300	150	300	75	150
T18	200	200	150	200	250	200	75	150
T19	60	75	100	50	200	75	200	100
T20	60	60	60	200	200	100	100	200
T21	60	60	60	250	250	100	200	100
T22	250	200	75	250	100	75	100	150
T23	250	200	60	200	200	200	100	150

Tabla 6. Diámetros óptimos encontrados en cada ejecución del AG

### 5.1.5 Análisis Hidráulico en REDGENHID

Para realizar el análisis hidráulico se cargan en el programa **REDGENHID** los diámetros óptimos de la ejecución número 7, seleccionando el número 7 (figura 26) en el programa y presionando el botón: “**Cargar la Lista de Diámetros**”:

RESULTADOS GUARDADOS TEMPORALMENTE:

GEN	CC	CH	DIAMETR
1 241	151466.42	0	[60.0, 60.0, 50.0, 50.0, 100.0, 100.0, 200.0, 200.0, 200.0, 200.0, 50.0, 50.0, 50.0,
2 125	168646.2	1.23	[200.0, 250.0, 300.0, 150.0, 200.0, 250.0, 200.0, 200.0, 300.0, 250.0, 100.0, 100.0,
3 163	146937.73	0	[100.0, 100.0, 100.0, 250.0, 150.0, 200.0, 250.0, 150.0, 150.0, 100.0, 100.0, 75.0,
4 107	181667.24	1.9	[250.0, 200.0, 150.0, 250.0, 250.0, 250.0, 250.0, 200.0, 300.0, 100.0, 100.0, 150.0,
5 32	206728.2	2.46	[150.0, 100.0, 100.0, 300.0, 250.0, 75.0, 300.0, 250.0, 150.0, 250.0, 60.0, 200.0, 1
6 159	177057.78	1.89	[75.0, 60.0, 150.0, 100.0, 200.0, 300.0, 150.0, 200.0, 300.0, 300.0, 75.0, 100.0, 100.0,
7 69	142356.39	0	[100.0, 250.0, 150.0, 200.0, 200.0, 250.0, 250.0, 75.0, 75.0, 75.0, 60.0, 75.0,
8 46	157190.98	1.88	[150.0, 75.0, 100.0, 200.0, 100.0, 200.0, 200.0, 150.0, 150.0, 200.0, 200.0, 50.0, 7

Exportar Diametros a Archivo \*CSV DATOS Cargar la Lista de Díametros: 7

Figura 26. Enviar los diámetros a analizar

Una vez teniendo la combinación de diámetros óptima cargada en la página de “**ANÁLISIS DE LA RED HIDRÁULICA**”, se le da clic al botón “**Analizar Red Hidráulica**” y se muestran los siguientes resultados:



Figura 27. Análisis hidráulico

Al correr el análisis hidráulico (figura 27), vemos que se necesitaron menos de 10 iteraciones para obtener un error de 0.000001787% (figura 28) que es menor a la tolerancia establecida y por lo tanto se considera como ejecución exitosa.

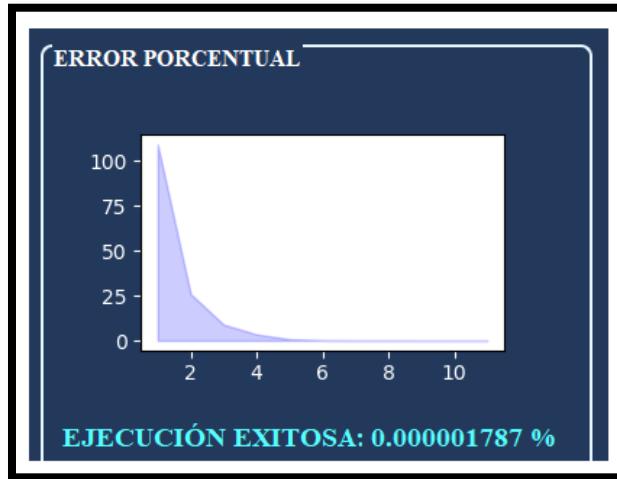


Figura 28. Error porcentual exitoso

Se muestra un resumen de velocidades y presiones máximas y mínimas (figura 29) que aseguran los objetivos de velocidad y presión establecidos por el usuario:

- Velocidad mínima:  $0.62 \text{ m/s} > 0.5 \text{ m/s}$ .

- Velocidad máxima:  $3.62 \text{ m/s} < 5 \text{ m/s}$ .
- Presión mínima:  $10.70 \text{ m.c.a.} > 10 \text{ m.c.a.}$
- Presión máxima:  $42.21 \text{ m.c.a.} < 50 \text{ m.c.a.}$



Figura 28. Resumen de velocidades y presiones

También se muestran los diagramas (figura 30) que nos permiten visualizar gráficamente, que todas las velocidades y presiones se mantienen dentro del rango previamente establecido en la **sección 5.1.2** de la presente tesis.

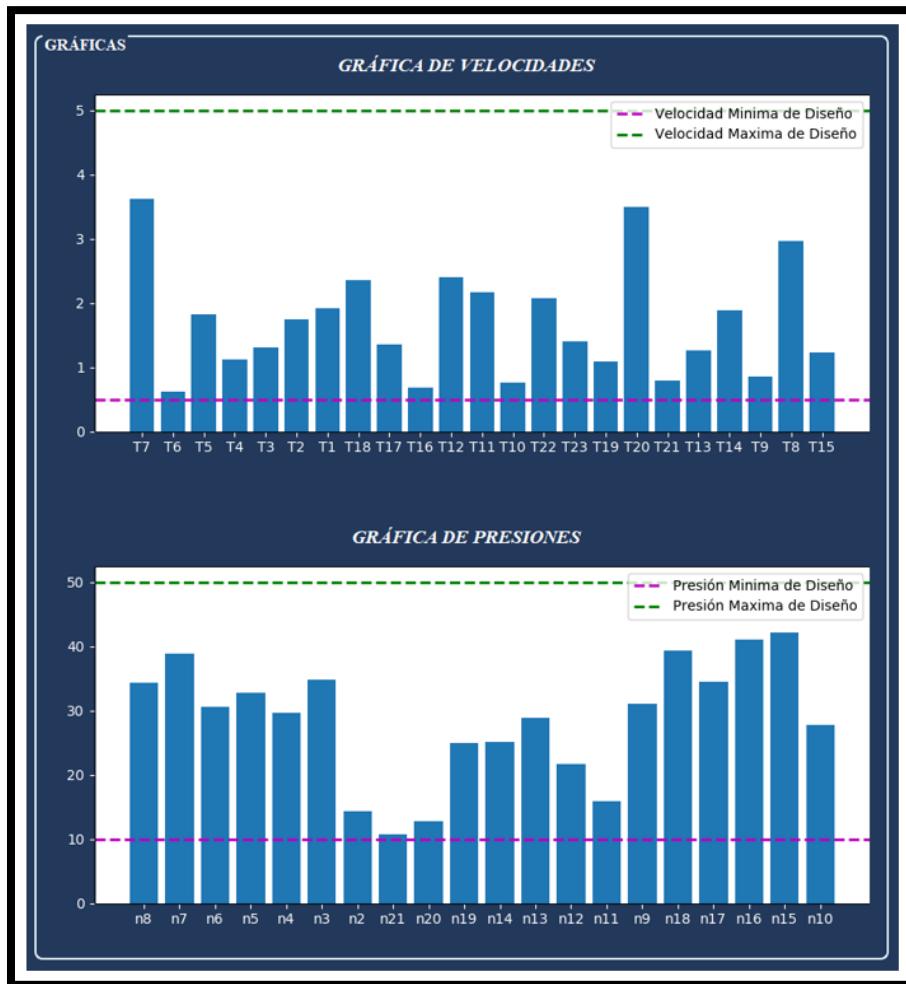


Figura 30. Gráfica de velocidades y presiones



Y por último se tienen dos tablas (Tablas 7 y 8) con información detallada de cada propiedad derivada del análisis referente a los nodos y tuberías de la red los cuales se compararán con Epanet 2.

TABLA CON INFORMACIÓN DE TUBERÍAS:						
TUBERÍA	NODOINI	NODOFIN	CAUDAL (LPS)	VEL. (M/S)	HF (M)	FRICCIÓN
T7	n7	n8	28.47	3.62	3.7	0.0138
T6	n6	n7	30.37	0.62	0.12	0.01154
T5	n5	n6	32.33	1.83	0.66	0.01273
T4	n4	n5	35.15	1.12	0.81	0.01204
T3	n3	n4	40.95	1.3	1.66	0.01204
T2	n2	n3	85.81	1.75	2.82	0.01154
T1	Embalse	n2	94.56	1.93	1.77	0.01154
T18	Embalse	n21	10.44	2.36	17.85	0.01465
T17	n21	n20	5.99	1.36	1.54	0.01465
T16	n20	n19	3.04	0.69	0.87	0.01465
T12	n14	n13	10.62	2.4	15.57	0.01465
T11	n13	n12	6.13	2.17	8.53	0.01536
T10	n12	n11	3.39	0.77	0.65	0.01465
T22	n8	n9	16.27	2.07	1.62	0.0138
T23	n9	n14	11.03	1.4	5.89	0.0138
T19	n3	n18	34.42	1.1	0.89	0.01204
T20	n18	n17	27.48	3.5	5.95	0.0138
T21	n17	n16	24.76	0.79	0.34	0.01204
T13	n15	n14	9.94	1.27	2.8	0.0138
T14	n16	n15	14.8	1.88	4.86	0.0138
T9	n10	n11	1.69	0.86	4.24	0.01599
T8	n8	n10	8.37	2.96	29.25	0.01536
T15	n16	n19	2.42	1.23	7.75	0.01599

Tabla 7. Resultados de las tuberías derivados del análisis hidráulico



TABLA CON INFORMACIÓN DE NODOS:			
NODO	DEMANDA (LPS)	ALTURA (M)	PRESIÓN (M)
n8	3.83	2336.39	34.39
n7	1.9	2340.27	38.96
n6	1.96	2340.44	30.68
n5	2.82	2341.23	32.76
n4	5.8	2342.3	29.72
n3	10.45	2344.4	34.88
n2	8.75	2347.8	14.34
n21	4.45	2329.68	10.7
n20	2.95	2327.73	12.74
n19	5.46	2326.46	25.03
n14	10.35	2327.12	25.11
n13	4.49	2309.56	28.93
n12	2.74	2299.76	21.71
n11	5.08	2298.84	15.84
n9	5.24	2334.51	31.09
n18	6.94	2343.23	39.3
n17	2.72	2336.95	34.49
n16	7.53	2336.47	41.07
n15	4.86	2330.71	42.21
n10	6.68	2304.77	27.77

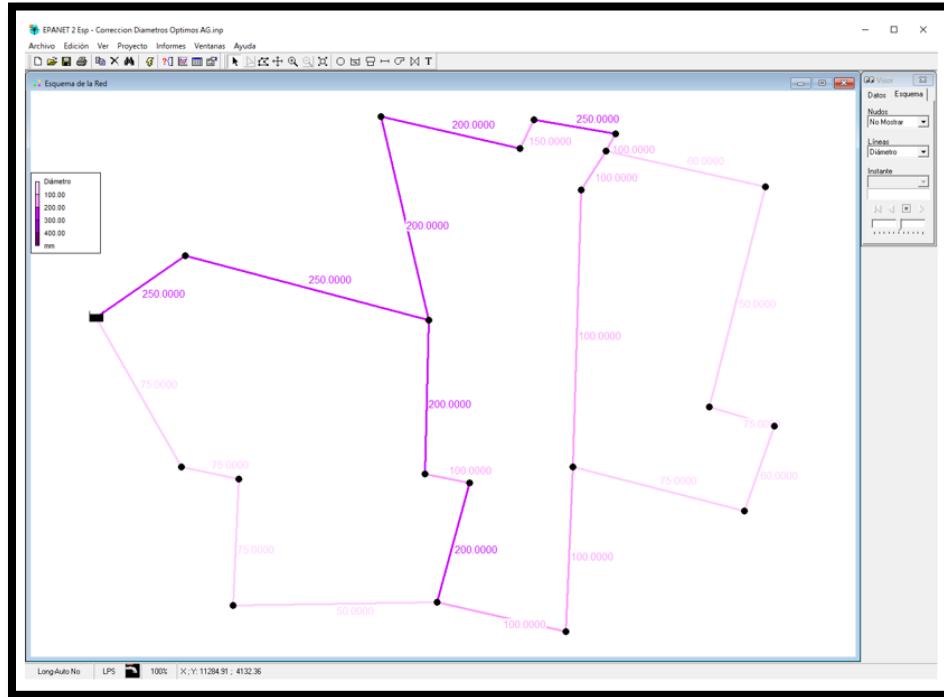
Tabla 8. Resultados de los nodos derivados del análisis hidráulico

### 5.1.6 Análisis Hidráulico en Epanet 2.

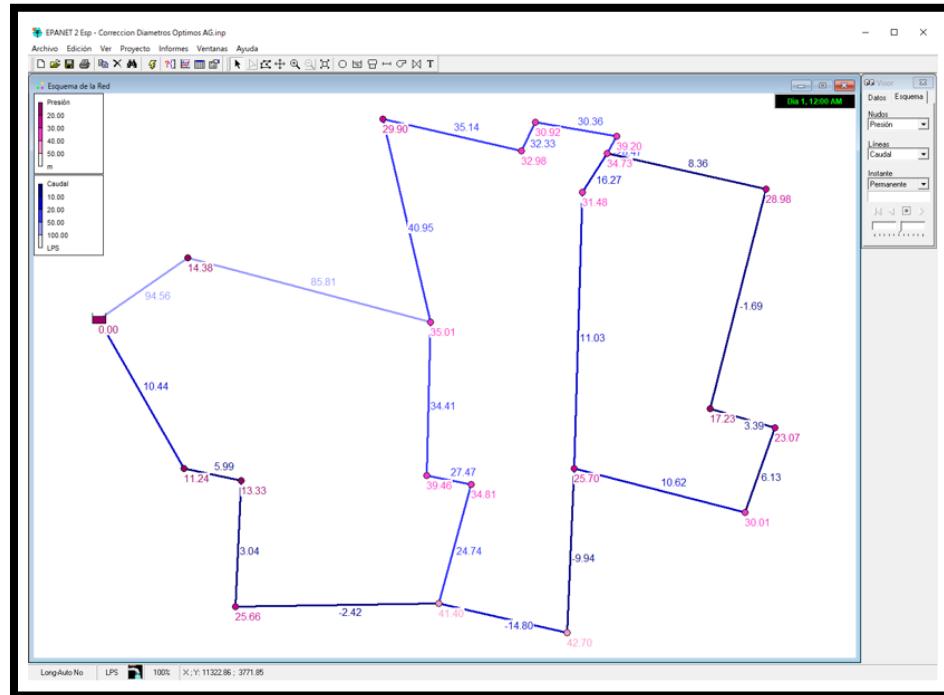
Haciendo uso del programa Epanet 2, se crea una red con las características mostradas en las tablas 2, 3 y 4 de la **sección 5.1.2** y con los diámetros óptimos obtenidos al ejecutar el Algoritmo Genético (tabla 6), tal como se muestra en la figura 31.

Una vez que la red tiene las propiedades correctas en cada tubería y nodo, el siguiente paso es correr un análisis hidráulico (figura 32), donde se muestran las presiones y caudales resultantes del análisis hidráulico.

Los resultados obtenidos con el programa Epanet, se muestran de manera más visible en formato de tabla (tablas 9 y 10).



**Figura 31. Red de Tezoyuca con los diámetros óptimos en Epanet 2**



**Figura 292. Resultado del análisis hidráulico en Epanet 2**



ID Línea	Caudal LPS	Velocidad m/s	Pérdida Unit. m/km	Factor Fricción
Tubería T7	-28.47	3.62	94.55	0.014
Tubería T6	30.36	0.62	1.28	0.016
Tubería T5	32.33	1.83	16.81	0.015
Tubería T4	35.14	1.12	4.89	0.015
Tubería T3	40.95	1.30	6.45	0.015
Tubería T2	85.81	1.75	8.44	0.014
Tubería T1	94.56	1.93	10.08	0.013
Tubería T18	10.44	2.36	61.21	0.016
Tubería T17	5.99	1.36	22.47	0.018
Tubería T16	3.04	0.69	6.67	0.021
Tubería T12	10.62	2.40	63.13	0.016
Tubería T11	6.13	2.17	68.51	0.017
Tubería T10	3.39	0.77	8.14	0.020
Tubería T22	16.27	2.07	34.18	0.016
Tubería T23	11.03	1.40	16.92	0.017
Tubería T19	34.41	1.10	4.71	0.015
Tubería T20	27.47	3.50	88.59	0.014
Tubería T21	24.74	0.79	2.59	0.016
Tubería T13	-9.94	1.27	14.03	0.017
Tubería T14	-14.80	1.88	28.78	0.016
Tubería T9	-1.69	0.86	16.31	0.022
Tubería T8	8.36	2.96	120.13	0.016
Tubería T15	-2.42	1.23	31.04	0.020

Tabla 9. Tabla de resultados derivados de las tuberías, Epanet 2.

ID Nudo	Demanda LPS	Altura m	Presión m
Nudo n8	3.83	2336.73	34.73
Nudo n7	1.90	2340.51	39.20
Nudo n6	1.96	2340.68	30.92
Nudo n5	2.82	2341.45	32.98
Nudo n4	5.80	2342.48	29.90
Nudo n3	10.45	2344.53	35.01
Nudo n2	8.75	2347.84	14.38
Nudo n21	4.45	2330.22	11.24
Nudo n20	2.95	2328.33	13.33
Nudo n19	5.46	2327.09	25.66
Nudo n14	10.35	2327.71	25.70
Nudo n13	4.49	2310.64	30.01
Nudo n12	2.74	2301.13	23.07
Nudo n11	5.08	2300.23	17.23
Nudo n9	5.24	2334.90	31.48
Nudo n18	6.94	2343.39	39.46
Nudo n17	2.72	2337.27	34.81
Nudo n16	7.53	2336.80	41.40
Nudo n15	4.86	2331.20	42.70
Nudo n10	6.68	2305.98	28.98
Embalse Embalse	-105.00	2349.89	0.00

Tabla 10. Tabla de resultados derivados de los nodos, Epanet 2

### 5.1.7 Comparativa entre los resultados del programa REDGENHID y Epanet 2

Para poder observar que tan bien o mal calculados están los resultados dados por el programa **REDGENHID**, con respecto a los que arroja **Epanet 2**, se muestran las gráficas (figura 33) en la que podemos comparar de manera visual cada diferencia

entre propiedades (caudales, presiones, factor de fricción, pérdidas unitarias y velocidades):



Figura 33. Gráfica de comparación de resultados

En las gráficas observamos que los resultados que devuelve el programa **REDGENHID** para los caudales, velocidades y presiones en los nodos son bastante cercanos a los que arroja el programa Epanet 2.

En la comparativa del factor de fricción, los resultados presentan ciertas diferencias claramente visibles en la gráfica correspondiente de la figura 33, y esto se debe a que para el intervalo  $2000 < Re < 4000$  Epanet 2 utiliza una interpolación cúbica al diagrama de Moody (A. Rossman, 1994) mientras que para el programa **REDGENHID** se utilizó la ecuación de Colebrook-White. Sin embargo, estas diferencias no afectan los resultados, y esto se demuestra en las comparativas de



velocidad y pérdidas por fricción que podemos apreciar en la figura 33 y en la tabla 11 y 12, donde se tabulan las diferencias de error para cada propiedad:

Tubería	Caudal (LPS)			Velocidad (m/s)			Factor de Fricción		
	Epanet 2	REDGENHID	Δ	Epanet 2	REDGENHID	Δ	Epanet 2	REDGENHID	Δ
Tubería T7	28.47	28.47	0	3.62	3.62	0	0.014	0.014	0.000
Tubería T6	30.36	30.37	0.01	0.62	0.62	0	0.016	0.012	0.004
Tubería T5	32.33	32.33	0	1.83	1.83	0	0.015	0.013	0.002
Tubería T4	35.14	35.15	0.01	1.12	1.12	0	0.015	0.012	0.003
Tubería T3	40.95	40.95	0	1.3	1.3	0	0.015	0.012	0.003
Tubería T2	85.81	85.81	0	1.75	1.75	0	0.014	0.012	0.002
Tubería T1	94.56	94.56	0	1.93	1.93	0	0.013	0.012	0.001
Tubería T18	10.44	10.44	0	2.36	2.36	0	0.016	0.015	0.001
Tubería T17	5.99	5.99	0	1.36	1.36	0	0.018	0.015	0.003
Tubería T16	3.04	3.04	0	0.69	0.69	0	0.021	0.015	0.006
Tubería T12	10.62	10.62	0	2.4	2.4	0	0.016	0.015	0.001
Tubería T11	6.13	6.13	0	2.17	2.17	0	0.017	0.015	0.002
Tubería T10	3.39	3.39	0	0.77	0.77	0	0.02	0.015	0.005
Tubería T22	16.27	16.27	0	2.07	2.07	0	0.016	0.014	0.002
Tubería T23	11.03	11.03	0	1.4	1.4	0	0.017	0.014	0.003
Tubería T19	34.41	34.42	0.01	1.1	1.1	0	0.015	0.012	0.003
Tubería T20	27.47	27.48	0.01	3.5	3.5	0	0.014	0.014	0.000
Tubería T21	24.74	24.76	0.02	0.79	0.79	0	0.016	0.012	0.004
Tubería T13	9.94	9.94	0	1.27	1.27	0	0.017	0.014	0.003
Tubería T14	14.8	14.8	0	1.88	1.88	0	0.016	0.014	0.002
Tubería T9	1.69	1.69	0	0.86	0.86	0	0.022	0.016	0.006
Tubería T8	8.36	8.37	0.01	2.96	2.96	0	0.016	0.015	0.001
Tubería T15	2.42	2.42	0	1.23	1.23	0	0.02	0.016	0.004

**Tabla 121.** Diferencias entre los resultados del programa REDGENHID y Epanet 2, parte 1

Tubería	Perdidas por Fricc. (m/km)			Nodo	Presión (m)		
	Epanet 2	REDGENHID	Δ		Epanet 2	REDGENHID	Δ
Tubería T7	94.55	92.55	2.00	Nudo n8	34.73	34.39	0.34
Tubería T6	1.28	0.90	0.38	Nudo n7	39.2	38.96	0.24
Tubería T5	16.81	14.39	2.42	Nudo n6	30.92	30.68	0.24
Tubería T4	4.89	3.84	1.05	Nudo n5	32.98	32.76	0.22
Tubería T3	6.45	5.22	1.23	Nudo n4	29.9	29.72	0.18
Tubería T2	8.44	7.19	1.25	Nudo n3	35.01	34.88	0.13
Tubería T1	10.08	8.73	1.35	Nudo n2	14.38	14.34	0.04
Tubería T18	61.21	55.57	5.64	Nudo n21	11.24	10.7	0.54
Tubería T17	22.47	18.24	4.23	Nudo n20	13.33	12.74	0.59
Tubería T16	6.67	4.71	1.96	Nudo n19	25.66	25.03	0.63
Tubería T12	63.13	57.59	5.54	Nudo n14	25.7	25.11	0.59
Tubería T11	68.51	61.41	7.10	Nudo n13	30.01	28.93	1.08
Tubería T10	8.14	5.88	2.26	Nudo n12	23.07	21.71	1.36
Tubería T22	34.18	30.27	3.91	Nudo n11	17.23	15.84	1.39
Tubería T23	16.92	13.87	3.05	Nudo n9	31.48	31.09	0.39
Tubería T19	4.71	3.67	1.04	Nudo n18	39.46	39.3	0.16
Tubería T20	88.59	86.11	2.48	Nudo n17	34.81	34.49	0.32
Tubería T21	2.59	1.90	0.69	Nudo n16	41.4	41.07	0.33
Tubería T13	14.03	11.27	2.76	Nudo n15	42.7	42.21	0.49
Tubería T14	28.78	24.97	3.81	Nudo n10	28.98	27.77	1.21
Tubería T9	16.31	12.02	4.29	Embalse	0	0	0.00
Tubería T8	120.13	114.29	5.84				
Tubería T15	31.04	24.78	6.26				

**Tabla 112.** Diferencias entre los resultados del programa REDGENHID y Epanet 2, parte 2



## Conclusiones

- 1) De acuerdo con los resultados antes presentados, se comprobó la hipótesis de la tesis de partida, que a través del uso del programa **REDGENHID** se optimiza el costo de la red de agua potable al mínimo posible. En la **sección 5.1.6** se aplicó el programa **Epanet 2** para realizar la simulación hidráulica del sistema y del cual se obtuvieron los siguientes resultados:
  - Se cumple con las normas de velocidad y presión dadas por el usuario.
  - Al correr el programa el error de iteración fue de 0. 000004571% que es mucho menor al 1% propuesto.
  - El uso del programa computacional no requiere un conocimiento profundo de programación o algoritmos genéticos.
- 2) Los objetivos planteados al inicio del presente trabajo se cumplieron al utilizar los algoritmos genéticos como una herramienta útil para resolver problemas de optimización, donde el espacio de búsqueda es demasiado grande.
- 3) Los usuarios finales, pueden utilizarlo sin la necesidad de invertir grandes cantidades de tiempo en el aprendizaje del programa y sin miedo a que el programa devuelva respuestas extrañas que pueda confundirlos.
- 4) Se automatizó el cálculo del diámetro de tuberías de una red de distribución de agua potable, utilizando los algoritmos genéticos como método para obtener los diámetros más económicos que cumplan con la velocidad y presión dadas por el usuario.
- 5) Sirve como un complemento de **Epanet 2**, pues también se trabaja con archivos de extensión **\*.inp**.
- 6) Se obtuvo un diseño adecuado y confiable en su funcionamiento hidráulico.



## Recomendaciones

- 1) Se deberá correr el programa varias veces, pues se devolverán diferentes resultados óptimos. Estos resultados deben guardarse para poder elegir de entre ellos el que mejor le convenga al usuario.
- 2) Se recomienda usar como tasa de mutación valores entre el 0.01% y 0.05% para mejores resultados.
- 3) Como tamaño de población se recomienda usar el 10% del número de generaciones.
- 4) En redes muy grandes se recomienda usar un número de generaciones menor a 1000 debido a que el algoritmo puede tardar demasiado.
- 5) En caso de que el costo hidráulico de la red no sea igual a cero tras varias iteraciones, se deberán modificar los valores de velocidades y presiones máximas y mínimas, pues tal vez no existe solución a las condiciones dadas inicialmente.
- 6) Al elegir la ecuación de regresión a usar (lineal, potencial, exponencial) para utilizar como función de costo, se recomienda siempre usar la que mejor coeficiente de correlación cuadrático  $R^2$  tenga (valor más cercano a 1).



## Bibliografía

- A. Rossman, L. (1994). *EPANET users manual: Project summary*. U.S. Environmental Protection Agency, Center for Environmental Research Information (1 Enero 1994).
- Carhuapoma Mendoza, J. C., & Chahuayo Durán, A. R. (2019). *DISEÑO DEL SISTEMA DE ABASTECIMIENTO DE AGUA POTABLE EN LA RINCONADA DE PAMPLONA ALTA, APLICANDO EPANET Y ALGORITMOS GENÉTICOS PARA LA LOCALIZACIÓN DE VÁLVULAS REDUCTORAS DE PRESIÓN*. Lima: Tesis, Universidad Peruana de Ciencias Aplicadas (UPC).
- Chambers, L. D. (2001). *The practical handbook of genetic algorithms, applications*. New York Washington, D.C.: Chapman & Hall/CRC.
- CONAGUA. (2007). *Manual de agua potable, alcantarillado y saneamiento*. Ciudad de México.
- Darwin, C. R. (1859). *On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life*. Londres: Jhon Murray.
- De Jong, K. A. (1975). Tesis Doctoral. *An Analysis of the behaviour of a class of genetic adaptive systems*. University of Michigan.
- Flores García, I. A. (2019). *Aplicación del algoritmo genético para el cálculo del diámetro de las tuberías de una red de distribución de agua potable en el distrito de Tarapoto 2018*. Tarapoto, Perú: Tesis, Universidad Nacional de San Martin-Tarapoto.
- Gestal, M., Rivero, D., Rabuñal, J. R., Dorado, J., & Pazos, A. (2010). *Introducción a los Algoritmos Genéticos y la Programación Genética*. Coruña: Universidad de Coruña.
- Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Michigan: Addison-Wesley.
- Guzmán Hurtado, V. H. (2009). *Algoritmos Genéticos y Epanet 2.0 para la Localización Óptima de Válvulas Reductoras de Presión en Redes de Distribución de Agua Potable*. México, DF.: Tesis, Universidad Nacional Autónoma de México (UNAM).
- Jarquín Laguna, R. (2014). *APLICACIÓN DE ALGORITMOS GENÉTICOS EN INGENIERÍA CIVIL*. México, D.F.: Tesis, Universidad Nacional Autónoma de México (UNAM).
- Melián Batista, B., Moreno Pérez, J. A., & Moreno Vega, J. M. (2009). Algoritmos Genéticos. Una visión práctica. *NÚMEROS*, 29-47.



- Navarro, A. F. (1998). Análisis hidráulico de tuberías por métodos matriciales. *Revista internacional de métodos numéricos para el cálculo y diseño en ingeniería*, 436.
- Normas Técnicas, C. (2008). *NORMAS TÉCNICAS COMPLEMENTARIAS PARA EL DISEÑO Y EJECUCIÓN DE OBRAS E INSTALACIONES HIDRÁULICAS*. Obtenido de NORMAS TÉCNICAS COMPLEMENTARIAS PARA EL DISEÑO Y EJECUCIÓN DE OBRAS E INSTALACIONES HIDRÁULICAS: <http://cgsservicios.df.gob.mx/prontuario/vigente/747.htm>
- Ponce Cruz, P. (2010). *Inteligencia Artificial con aplicaciones a la ingeniería*. México: Alfaomega Grupo Editor.
- Rabuñal Dopico, J. R. (2007). *USO DE TÉCNICAS DE INTELIGENCIA ARTIFICIAL EN INGENIERÍA CIVIL*. Coruña: Tesis, Universidad de Coruña.
- Saldarriaga, J. (1998). *Hidráulica de tuberías abastecimiento de agua, redes y riegos*. Ciudad de México: Alfaomega.
- Todini, E., & Pilati, S. (1987). A gradient method for the analysis of pipe networks. *International Conference on Computer Applications for Water Supply and Distribution*. UK: Leicester Polytechnic.
- www.tutorialesaldia.com. (s.f.). *Tutoriales al día*. Obtenido de Tutoriales Ingeniería Civil: <http://ingenieriacivil.tutorialesaldia.com/red-de-distribucion-de-agua-potable-abierta-o-cerrada/>



## Anexos.

### Anexo A. Archivo: GradienteHidraulico.py

#### **GradienteHidraulico.py**

```
import math
import numpy as np

class Nodos(object):
    def __init__(self, NumeroNodo, Gasto_o_Altura, Tipo_elemento,
Nivel_Topografico):
        """
        *** VARIABLES PRINCIPALES ***
        :param NumeroNodo: Número del nodo a usar.
        :param Gasto_o_Altura: Gasto si es un Nodo, Altura si es un
Tanque (L/s).
        :param Tipo_elemento: Tanque--> 'E', Nodo--> 'N'.
        :param Nivel_Topografico: Cota de Terreno del nodo (Metros).

        """
        self.nn = NumeroNodo
        self.tipo = Tipo_elemento
        self.msnm = Nivel_Topografico
        self.q = 0 # Gasto de demanda nodal
        self.h = 0 # Altura del embalse

        if self.tipo == 'E':
            self.h = Gasto_o_Altura
        elif self.tipo == 'N':
            self.q = Gasto_o_Altura / 1000

    def __str__(self):
        if self.tipo == 'E':
            print('Tipo de Elemento: Embalse o Tanque')
        else:
            print('Tipo de Elemento: Nodo')
        print('Número de nodo: ', str(self.nn))
        print('Cota de Nivel: ', str(self.msnm), ' metros')
        if self.tipo == 'E':
            print('La altura es: ', str(self.h), ' metros')
        else:
            print('La demanda es: ', str(self.q * 1000), ' L/s')
        return ""

class Tuberias(object):
    def __init__(self, Numerotub, n_ini, n_fin, Longiud, diametro,
perd_menores, Rugosidad_Absoluta):
        """
        *** VARIABLES PRINCIPALES ***
        :param Numerotub: Número de la tubería.
        :param n_ini: Nodo inicial de la tubería.
```



```
:param n_fin: Nodo final de la tubería.
:param Longitud: Longitud de la tubería.
:param diametro: Diametro de la tubería (Milímetros).
:param perd_menores: Perdida Menor de la tubería (Adimensional).
:param Rugosidad_Absoluta: Rugosidad Absoluta de la tubería
(Milímetros).
"""
self.ni = n_ini
self.nf = n_fin
self.nt = Numerotub
self.L = Longitud
self.d = diametro / 1000
self.km = perd_menores
self.ks = Rugosidad_Absoluta / 1000

def Direccion(self):
"""
:return: Vector de Direccion de la tuberia.
"""
return [self.ni, self.nf]

def Area(self):
"""
:return: Area de la seccion transversal de la tuberia (Metros).
"""
return 0.25 * math.pi * self.d ** 2

def __str__(self):
print('Número de tubería: ', str(self.nt))
print('[Nodo inicial, Nodo final]: ', str([self.ni, self.nf]))
print('Longitud de la tubería: ', str(self.L), ' metros')
print('Diametro de la tubería: ', str(self.d * 1000), ' milímetros')
print('Pérdidas Menores: ', str(round(self.km, 4)))
print('Rugosidad Absoluta: ', str(round(self.ks * 1000, 5)), ' milímetros')
print('Area de la tubería: ', str(round(self.Area(), 4)), ' m2')
return ""

class RedHidraulica(object):
    def __init__(self, data_nod, data_tub, Pdmax, Pdmin, Vdmax, Vdmin,
DiametroComercial):
        # Listas con
        self.dataN = data_nod
        self.dataT = data_tub

        # Restricciones Hidráulicas
        self.Pd_max = Pdmax # Presión Máxima de Diseño, (m.c.a.).
        self.Pd_min = Pdmin # Presión Mínima de Diseño, (m.c.a.).
        self.Vd_max = Vdmax # Velocidad Máxima de Diseño, (m/s).
        self.Vd_min = Vdmin # Velocidad Mínima de Diseño, (m/s).

        # Diametros y valores de regresiones
        self.Dcom = DiametroComercial # Diametro comercial
```



```
self.Nod = []
row_n = len(self.dataN)
for i in range(row_n):
    self.Nod.append(Nodos(self.dataN[i][0], self.dataN[i][1],
self.dataN[i][2], self.dataN[i][3]))

self.Tub = []
row_t = len(self.dataT)
for i in range(row_t):
    self.Tub.append(
        Tuberias(self.dataT[i][0], self.dataT[i][1],
self.dataT[i][2], self.dataT[i][3], self.dataT[i][4],
                self.dataT[i][5], self.dataT[i][6]))

self.num_nod = len(self.Nod) # Numero total de Nodos (sin
distinguir entre embalces)
self.num_tub = len(self.Tub) # Numero total de Tuberias

def __str__(self):
    print('Numero de tuberias: ', str(self.num_tub))
    print('Numero de nodos: ', str(self.num_nod))
    print('Vdmax: ', self.Vd_max, ' Vdmin: ', self.Vd_min, ' Pdmax: ',
', self.Pd_max, ' Pdmin: ', self.Pd_min)
    print('Diametros comerciales a usar: ', self.Dcom)
    return ""

class Gradiente(object):
    def __init__(self, Tuberias, Nodos, Viscocidad_dinamica = None):
        """
        *** VARIABLES PRINCIPALES ***
        :param Tuberias:
        :param Nodos:
        :param Viscocidad_dinamica:
        """
        self.Lt = Tuberias
        self.Ln = Nodos
        if Viscocidad_dinamica == None:
            self.v = 1.141*10**-6
        else:
            self.v = Viscocidad_dinamica

        """
        *** VARIABLES SECUNDARIAS ***
        :param num_nod: Numero total de Nodos (sin distinguir entre
embalces).
        :param NT: Numero total de Tuberias.
        :param NS: Numero de Embalces.
        :param NN: Numero de Nodos.
        :param Cota: Cotas de cada Nodo.
        :param D_Cota: Diferencia de alturas entre la cota del Embalce y
las cotas de los Nodos.
        :param Qd: Vector de Demandas en los nodos, (m3/s).
        :param Ho: Vector de Altura de los embalses, (Metros).
        :param n_dw: Numero de Darcy-Weisbach.
```



```
:param ini_fin: Vector de Dirección del flujo la tubería.  
"""  
    self.num_nod = len(self.Ln) # Número total de Nodos (sin  
    distinguir entre embalces)  
    self.NT = len(self.Lt) # Número total de Tuberías  
    self.NS = 0 # Número de Embalces  
    self.NN = 0 # Número de Nodos  
    self.Cota = [] # Cotas de cada Nodo  
    self.D_Cota = [] # Diferencia de alturas entre la cota del  
    Embalce y las cotas de los Nodos  
    self.Qd = []  
    self.Ho = []  
    self.n_dw = 2 # Número de Darcy-Weisbach  
    self.ini_fin = []  
    self.I = np.eye(self.NT) # Matriz Identidad I  
    self.N = np.eye(self.NT) * self.n_dw # Matriz diagonal Darcy-  
    Weisbach  
  
    # DEMANDAS, ALTURAS Y COTAS  
    Cota = []  
    Demanda = []  
    AlturaEmbalce = []  
    for i in range(self.NT):  
        if i <= (self.num_nod-1):  
            if self.Ln[i].tipo == 'E':  
                AlturaEmbalce.append([self.Ln[i].h])  
                self.NS += 1  
            elif self.Ln[i].tipo == 'N':  
                Demanda.append([self.Ln[i].q])  
                self.NN += 1  
                Cota.append([self.Ln[i].msnm])  
                self.ini_fin.append([self.Lt[i].Direccion()[0],  
                self.Lt[i].Direccion()[1]])  
  
                self.Cota = np.array(Cota)  
                self.D_Cota = [self.Cota[0] - val for val in self.Cota]  
                self.Qd = np.array(Demanda)  
                self.Ho = np.array(AlturaEmbalce)  
  
    # Matrices A12, A21 y A10  
    self._MatricesPrincipales()  
  
"""  
    METODO PARA OBTENER LAS MATRICES: A12, A21 y A10  
"""  
def __MatricesPrincipales(self):  
    """  
    :return:  
        A12: Matriz de conectividad (-1 inicio y 1 final).  
        A21: Matriz Transpuesta de A12 --> A21.  
        A10: Matriz Topologica tramo a nudo.  
    """  
    # Matriz de conectividad (-1 inicio y 1 final)  
    self.A12 = []
```



```
A12 = np.zeros((self.NT, self.NN + self.NS))
for i in range(self.NT):
    ni = self.ini_fin[i][0] - 1
    nf = self.ini_fin[i][1] - 1
    A12[i][ni] = -1
    A12[i][nf] = 1
self.A12 = A12[:, self.NS:(self.NN + self.NS)]


# Matriz Transpuesta de A12 --> A21
self.A21 = []
self.A21 = np.transpose(self.A12)

# Matriz Topologica tramo a nudo
self.A10 = []
self.A10 = A12[:, 0:self.NS]

return self.A12, self.A21, self.A10


"""
    ECUACION DE VELOCIDAD
"""
def __Vel(self, A, Q):
    """
    :param A: Area de la seccion transversal de la tuberia, (m2).
    :param Q: Caudal que corre por la tuberia, (L/s).
    :return:
        V: Velocidad que corre por la tuberia, (m/s).
    """
    return Q / A


"""
    NUMERO DE REYNOLDS
"""
def __Re(self, V, d):
    """
    :param V: Velocidad que corre por la tuberia, (m/s).
    :param d: Diametro de la tuberia, (Metros).
    :return:
        Re: Numero de Reynolds, (Adimencional).
    """
    return abs(V * d / self.v)


"""
    ECUACION DE SWAMEE-JAIN
"""
def __SwameeJain(self, Re, d, ks):
    """
    :param Re: Numero de Reynolds, (Adimencional).
    :param d: Diametro de la tuberia, (Metros).
    :param ks: Rugosidad Absoluta de la tuberia (Metros).
    :return:
        f: Factor de Friccion, (Adimencinoal).
    """
    v1 = ks / (3.71 * d)
    v2 = 5.74 / (Re ** 0.9)
    v3 = math.log10(v1 + v2)
    v4 = v3 ** 2
```



```
f = 0.25 / v4
return f

def __HagenPoiseuille(self, Re):
    f = 64 / Re
    return f

def __ColebrookWhite(self, Re, d, ks):
    def Fricc(Re, ks, d, x):
        a = ks / (3.7 * d)
        b = 2.51 * x / Re
        return - 2 * math.log10(a + b) - x

    def Fricc_p(Re, ks, d, x):
        a = - 2 / math.log(10)
        b = ks / (3.7 * d)
        c = 2.51 * x / Re
        d = 2.51 / Re
        return (a * d / (b + c)) - 1

    tol = 0.00001 # Tolerancia permitida del error.
    error = 100 # Error de inicialización.
    while error >= tol:
        if error == 100:
            x1 = self.__SwameeJain(Re, d, ks)
            Fxp = Fricc_p(Re, ks, d, x1)
            Fx = Fricc(Re, ks, d, x1)
            x2 = x1 - (Fx / Fxp)
            error = abs((x2 - x1) / x2) * 100
            x1 = x2
        f = 1 / (x1 ** 2)
    return f

def Fn_Friccion(self, Re, d, ks):
    if Re < 2200:
        return self.__HagenPoiseuille(Re)
    elif Re >= 2200:
        return self.__ColebrookWhite(Re, d, ks)

"""
    METODO PARA INVERTIR LA DIRECCION DE UNA TUBERIA
"""
def __InvertirTuberia(self, Caudal):
    """
    :param Caudal: Vector del Caudal en una tubería, (L/s).
    :return:
        ini_fin: Vector de Dirección del flujo la tubería.
    """
    aux = self.ini_fin
    self.ini_fin = []
    for i in range(self.NT):
        if Caudal[i] < 0:
            self.ini_fin.append([aux[i][1], aux[i][0]])
        else:
            self.ini_fin.append([aux[i][0], aux[i][1]])
    return self.ini_fin
```



```
"""
    METODO RESOLVER:
"""

def Resolver(self):
    """
    :return:
        Caudal: Vector del Caudal en una tuberia, (L/s).
        Velocidad: Vector de Velocidad que corre por la tuberia,
        (m/s).
        Altura: Vector de Altura Piezometrica en cada nodo,
        (Metros).
        Presion: Vector de Presion en cada nodo, (Metros).
        er: Error porcentual de la iteracion, (%).
    """

    iter = 0
    Velocidad = np.zeros((self.NT, 1))
    Perd = np.zeros((self.NT, 1))
    Fricc = np.zeros((self.NT, 1))

    while iter < self.NT:
        self._MatricesPrincipales()
        [self.Caudal, self.Altura, self.Presion, self.er] =
        self._Calcular()
        if self.Caudal.min() < 0:
            self._InvertirTuberia(self.Caudal)
            iter += 1
        elif self.Caudal.min() >= 0:
            for i in range(self.NT):
                Velocidad[i] = (self.Caudal[i] / 1000) /
        self.Lt[i].Area()
                diam2 = self.Lt[i].d
                ks2 = self.Lt[i].ks
                Reynolds2 = self._Re(Velocidad[i], diam2)
                Fricc[i] = self.Fn_Friction(Reynolds2, ks2, diam2)
                Perd[i] = (Fricc[i] * self.Lt[i].L / diam2) *
        ((Velocidad[i] ** 2) / 19.62)

            break
        return self.Caudal, Velocidad, self.Altura, self.Presion, Fricc,
        Perd, self.er, self.ini_fin

    def __Calcular(self):
        itera = 1
        er = 100
        ErrorLista = []

        # Iteraciones
        while er > 0.00001 and itera < 2 * self.NN:
            A11 = np.zeros((self.NT, self.NT))
            A11_p = np.zeros((self.NT, self.NT))
            if itera == 1:
                Qi = np.ones((self.NT, 1)) * 0.1

            for j in range(self.NT):
                Area = self.Lt[j].Area()
                diametro = self.Lt[j].d
```



```
ks = self.Lt[j].ks
km = self.Lt[j].km
V = self.__Vel(Area, Qi[j])
Re = self.__Re(V, diametro)
f = self.Fn_Friccion(Re, diametro, ks)
Longitud = self.Lt[j].L
alpha = ((f * Longitud / diametro) + km) / (19.62 *
(Area ** 2))
beta = 0
gamma = 0
a = alpha * (Qi[j] ** (self.n_dw - 1))
b = beta
c = gamma / Qi[j]
A11[j][j] = a + b + c
A11_p[j][j] = a

# Calculo de la Altura Piezometrica H(i+1)
h1 = np.linalg.inv(np.matmul(self.N, A11_p))
h2 = np.linalg.inv(np.matmul(np.matmul(self.A21, h1),
self.A12))
h3 = np.matmul(A11, Qi) + np.matmul(self.A10, self.Ho)
h4 = np.matmul(np.matmul(self.A21, h1), h3)
h5 = h4 - np.matmul(self.A21, Qi) + self.Qd
H = -np.matmul(h2, h5)

# Calculo del Caudal Q(i+1)
Qanterior = np.linalg.norm(Qi)
q1 = self.I - np.matmul(h1, A11)
q2 = np.matmul(self.A12, H) + np.matmul(self.A10, self.Ho)
Qi = np.matmul(q1, Qi) - np.matmul(h1, q2)
Qactual = np.linalg.norm(Qi)

# Error Relativo Porcentual
er = abs((Qactual - Qanterior) / Qactual) * 100
ErrorLista.append(er)
itera += 1

Caudal = Qi * 1000
Presion = np.zeros((self.NN, 1))
Altura = np.zeros((self.NN, 1))

for i in range(self.NN):
    Presion[i] = H[i] + self.D_Cota[i + self.NS]
    Altura[i] = self.Cota[i + self.NS] + Presion[i]

return Caudal, Altura, Presion, ErrorLista
```

## Anexo B. Archivo: AlgoritmoGenetico.py

### AlgoritmoGenetico.py

```
import GradienteHidraulico as gh
import random as random
import timeit
```



```
from progressBarPYQT5 import ProgBar
from PyQt5.QtCore import QApplication

class Individuo:
    def __init__(self, RedHidraulica, K1, K2, RegTipo, DiametroTuberias
= []):
        self.I_cn = RedHidraulica.Nod # Clase Nodos (Datos de los
nodos)
        self.I_ct = RedHidraulica.Tub # Clase Tuberia (Datos de las
tuberias)
        self.Dcom = RedHidraulica.Dcom # Diametro comercial
        self.diametros = DiametroTuberias # Diametros de la tubería
        self.K1 = K1
        self.K2 = K2
        self.RT = RegTipo

        # Restricciones Hidráulicas
        self.Pd_max = RedHidraulica.Pd_max # Presión Máxima de Diseño,
(m.c.a.).
        self.Pd_min = RedHidraulica.Pd_min # Presión Mínima de Diseño,
(m.c.a.).
        self.Vd_max = RedHidraulica.Vd_max # Velocidad Máxima de
Diseño, (m/s).
        self.Vd_min = RedHidraulica.Vd_min # Velocidad Mínima de
Diseño, (m/s).

        # Inicializamos el cromosoma, el diámetro y el fitness en vacío
pues recibirán datos después
        self.num_tub = len(self.I_ct) # Longitud del Cromosoma
        self.Penalización = 0
        self.Fit = 0

        # Creación de la lista DiametroTuberias
        if len(self.diametros) == 0:
            # Se crean diámetros al azar
            for i in range(self.num_tub):
                val = round(random.randint(0, len(self.Dcom) - 1))
                self.diametros.append(self.Dcom[val])
                self.I_ct[i].d = self.diametros[i] / 1000
        else:
            for i in range(self.num_tub):
                self.I_ct[i].d = self.diametros[i] / 1000

        # Ejecutamos el Método del Gradiente para evaluar al individuo
hidráulicamente
        self.cg = gh.Gradiente(self.I_ct, self.I_cn).Resolver() # [Q, V,
H, P, f, hf, er]

        self.Vel = self.cg[1]
        self.v_min = float(min(self.Vel))
        self.v_max = float(max(self.Vel))

        self.Press = self.cg[3]
        self.p_min = float(min(self.Press))
        self.p_max = float(max(self.Press))
```



```
def __str__(self):
    print("Diametros: ", self.diametros)
    print("Velocidades:")
    print("Minima : ", round(self.v_min,3), "m/s, Maxima: ",
round(self.v_max,3), "m/s")
    print("Presiones:")
    print("Minima : ", round(self.p_min,3), "m.c.a., Maxima: ",
round(self.p_max,3), "m.c.a.")
    print("Error porcentual: ",self.cg[6], "%")
    print("Costo de Construcción: ", round(self.Cc(), 3))
    print("Costo Hidráulico: ", round(self.penalizacion(), 3))
    print("Fitness: ", round(self.Fitness(), 3))
    return ""

''' Función del Costo Constructivo '''

def Cc(self):
    self.Costo = 0
    # Costo = L * ( K1 * D + K2 )
    if self.RT == "Lineal":
        for i in range(self.num_tub):
            self.Costo += self.I_ct[i].L * (self.K1 *
self.diametros[i] + self.K2)

    # Costo = L * (K1 * 10 ** (K2 * D))
    elif self.RT == "Exponencial":
        for i in range(self.num_tub):
            self.Costo += self.I_ct[i].L * (self.K1 * 10 ** (self.K2
* self.diametros[i]))

    # Costo = L * (K1 * D ** K2)
    elif self.RT == "Potencial":
        for i in range(self.num_tub):
            self.Costo += self.I_ct[i].L * (self.K1 *
self.diametros[i] ** self.K2)

    return self.Costo # Costo de construcción de este individuo en
pesos

def penalizacion(self):

    # Penalización Hidráulica
    if (self.v_min < self.Vd_max) and (self.v_min >= self.Vd_min):
        P_vmin = 0
    else:
        P_vmin = 10*abs(self.v_min - self.Vd_min)

    if (self.v_max <= self.Vd_max) and (self.v_max > self.Vd_min):
        P_vmax = 0
    else:
        P_vmax = 10*abs(self.v_max - self.Vd_max)

    if (self.p_max <= self.Pd_max) and (self.p_max > self.Pd_min):
        P_pmax = 0
    else:
        P_pmax = abs(self.p_max - self.Pd_max)
```



```
if (self.p_min < self.Pd_max) and (self.p_min >= self.Pd_min):
    P_pmin = 0
else:
    P_pmin = abs(self.p_min - self.Pd_min)

# Penalizacion Hidraulica total para el individuo
self.Penalizacion = P_pmax + P_pmin + P_vmax + P_vmin
return self.Penalizacion

def Fitness(self):
    self.Fit = self.Cc() * (1 + self.penalizacion())
    return self.Fit

class AlgoritmoGenetico():
    def __init__(self, RedHidraulica ,k1, k2, RegreTipo,
tamano_poblacion, tasa_mutacion, num_generaciones):

        self.rh = RedHidraulica
        self.K1 = k1
        self.K2 = k2
        self.RT = RegreTipo
        self.tam_pob = tamano_poblacion
        self.tasa_mutacion = tasa_mutacion
        self.num_gene = num_generaciones

        self.Dcom = RedHidraulica.Dcom
        self.poblacion = []
        self.poblacion_evaluada = []
        self.suma_porcentajes = []
        self.mejores_padres = []
        self.mejor_individuo = []

    def __str__(self):
        print("Tamaño de la poblacion", self.tam_pob)
        print("Tasa de Mutacion: ", self.tasa_mutacion)
        print("Número de generaciones: ", self.num_gene)
        print("Diametros Comerciales a usar: ", self.Dcom)
        print("Tipo de Regresion: ", self.RT)
        return ""

# RedHidraulica, K1, K2, RegTipo, DiametroTuberias = []

def inicializar_poblacion(self):  # -->
    self.poblacion = []
    for i in range(self.tam_pob):
        ind = Individuo(self.rh, self.K1, self.K2, self.RT, [])
        self.poblacion.append(ind)
    return self.poblacion

def poblacion_eval(self):  # -->
    self.poblacion_evaluada = []
    for i in range(self.tam_pob):
        self.poblacion_evaluada.append(self.poblacion[i].Fitness())
    return self.poblacion_evaluada

def ordenar_poblacion(self):  # -->
```



```
self.poblacion = sorted(self.poblacion,
                        key=lambda poblacion: poblacion.Fit,
                        reverse=True)

def porcentaje_acumulado(self): # -->
    Fitness_inv = []
    for i in range(self.tam_pob):
        Fitness_inv.append(1 / self.poblacion_evaluada[i])
    Sfx = sum(Fitness_inv)
    fx_Sfx = []
    for i in range(self.tam_pob):
        if i == 0:
            fx_Sfx.append(Fitness_inv[i] / Sfx)
        else:
            fx_Sfx.append(fx_Sfx[i - 1] + Fitness_inv[i] / Sfx)
    self.suma_porcentajes = fx_Sfx
    return self.suma_porcentajes

def ruleta_aleatoria(self): # -->
    self.mejores_padres = []
    num_tiros = self.tam_pob
    for i in range(0, num_tiros - 1):
        tiro = random.random()
        if tiro <= self.suma_porcentajes[0]:
            self.mejores_padres.append(self.poblacion[0].diametros)
        else:
            for j in range(1, num_tiros):
                if self.suma_porcentajes[j - 1] < tiro and tiro <=
    self.suma_porcentajes[j + 1]:
        self.mejores_padres.append(self.poblacion[j].diametros)
        break
    self.elitismo()
    return self.mejores_padres

def nueva_poblacion(self, new_padres):
    self.poblacion = []
    for i in range(self.tam_pob):
        ind = Individuo(self.rh, self.K1, self.K2, self.RT,
new_padres[i])
        self.poblacion.append(ind)
    return self.poblacion

def crossover(self):
    padres = []
    madres = []
    '''Separamos en padres y madres'''
    for i in range(0, self.tam_pob, 2):
        padres.append(self.poblacion[i].diametros)
        madres.append(self.poblacion[i + 1].diametros)
    '''Cruzamos los padres y madres'''
    Hijos = []
    for i in range(len(padres)):
        padre = padres[i]
        madre = madres[i]
        corte = round(random.random() * (len(padre) - 1))
        Hijos.append(padre[0:corte] + madre[corte::])
```



```
Hijo2 = madre[0:corte] + padre[corte::]
Hijos.append(Hijo1)
Hijos.append(Hijo2)
return Hijos

def mutacion(self, hijos): # --> OK
    mut = []
    for i in range(len(hijos)):
        h = hijos[i]
        for j in range(len(h)):
            if random.random() < self.tasa_mutacion:
                h[j] = self.Dcom[round(random.randint(0,
len(self.Dcom) - 1))]
        mut.append(h)
    return mut

def mejor_individuo_especie(self, gene): # --> OK
    mejor_de_la_especie = min(self.poblacion, key=lambda poblacion:
poblacion.Fit)

    diam = mejor_de_la_especie.diametros
    val = mejor_de_la_especie.Fit
    hid = mejor_de_la_especie.Penalizacion
    cos = mejor_de_la_especie.Costo

    if gene == 0:
        self.mejor_individuo.append([diam, hid, val, cos, gene])
    if (gene > 0) and (val < self.mejor_individuo[-1][2]):
        self.contador = 0
        self.mejor_individuo.append([diam, hid, val, cos, gene])
    else:
        self.contador += 1

    return self.mejor_individuo

def elitismo(self):
    self.mejores_padres.append(self.mejor_individuo[-1][0])

def TiempoEjecucion(self, seg):
    hr = seg / 3600
    hora = int(hr) # Hora
    minu = (hr - hora) * 60
    minuto = int(minu) # Minutos
    segundo = int(round((minu - minuto) * 60, 0))
    return hora, minuto, segundo

def resolver(self):
    tic = timeit.default_timer()
    # Inicializamos la población:
    self.inicializar_poblacion()
    self.contador = 0
    self.main = ProgBar()
    self.main.show()

    for i in range(self.num_gene):
        # self.generacion.append(i)
```



```
QCoreApplication.processEvents()
self.main.repaint()
self.main.progressBarValue(i, self.num_gene)

# Evaluamos la población para posteriormente ordenarla:
self.poblacion_eval()

# Ordenamos la población:
self.ordenar_poblacion()
self.poblacion_eval()

# Seleccionamos al mejor individuo de la población
self.mejor_individuo_especie(i)

# Calculamos un porcentaje acumulado para cada individuo que
serviría para el filtro de la ruleta:
self.porcentaje_acumulado()

# Ponemos a competir a los individuos a través de la
selección por ruleta:
Rul = self.ruleta_aleatoria()

# Creamos una nueva población tomando en cuenta el filtro de
ruleta:
self.nueva_poblacion(Rul)
self.poblacion_eval()

# Seleccionamos al mejor individuo de la población
self.mejor_individuo_especie(i)

# Reproducimos la población (Crossover):
crom_cross = self.crossover()

# Mutamos la población:
muta = self.mutacion(crom_cross)

# Creamos una nueva población tomando en cuenta los genes
mutados:
self.nueva_poblacion(muta)

if self.contador >= 100:
    print("Generación con solución más óptima: ", i)
    break

# Tiempo de ejecución del AG
toc = timeit.default_timer()
Dtt = (toc - tic)
[hora, minuto, segundo] = self.TiempoEjecucion(Dtt)

return self.mejor_individuo, hora, minuto, segundo
```

## Anexo C. Archivo: FileTools.py

### [FileTools.py](#)



```
import csv
import xlrd
from epanettools.epanettools import EPANetSimulation, Node, Link

class FileManipulation(object):
    def __init__(self, File):
        ObjRed = EPANetSimulation(File) # Objeto Red
        self.ObjNodos = ObjRed.network.nodes # Objeto Nodos
        self.ObjTuberias = ObjRed.network.links # Objeto Tuberias
        self.num_nod = len(self.ObjNodos) # Numero de nodos
        self.num_tub = len(self.ObjTuberias) # Numero de tuberias

    def PropiedadesNodos(self):
        """
        :return:
        """
        # PROPIEDADAS DE LAS NODOS
        en = Node.value_type['EN_ELEVATION'] # Elevaciones
        bd = Node.value_type['EN_BASEDEMAND'] # Demandas Base
        tkl = Node.value_type['EN_TANKLEVEL'] # Altura del tanque

        List_Numeros_Nodos = [] # Index entero del elemento.
        List_Idx_Nodos = [] # Index string del elemento.
        List_Tipo_Nodo = [] # Tipo de Nodo ('E' ó 'N').
        List_Elev_Nod = [] # Cota del Nodo, (Metros).
        List_DemBase_Nod = [] # Demanda en el Nodo, (L/s).
        List_Tank_Lev = [] # Elevacion del Tanque, (Metros).

        for i in range(self.num_nod):
            if self.ObjNodos[i + 1].node_type == 0:
                tipo = 'N' # JUNCTION
            elif self.ObjNodos[i + 1].node_type == 2 or self.ObjNodos[i + 1].node_type == 1:
                tipo = 'E' # TANK

            List_Numeros_Nodos.append(self.ObjNodos[i + 1].index)
            List_Idx_Nodos.append(self.ObjNodos[i + 1].id)
            List_Tipo_Nodo.append(tipo)
            List_Elev_Nod.append(round(self.ObjNodos[i + 1].results[en][0],2))
            List_DemBase_Nod.append(round(self.ObjNodos[i + 1].results[bd][0],2))
            List_Tank_Lev.append(round(self.ObjNodos[i + 1].results[tkl][0],2))

        # INFORMACION DE LOS NODOS DE MANERA ORDENADA:
        cont = List_Tipo_Nodo.count('E') # Contador numero de Tanques
        nn = cont + 1 # Numero de nodos demanda
        ee = 0 # Numero de Tanques
        nn_new = 1 # Numero de nodos demanda corregido

        Coord_X = []
        Coord_Y = []
        List_Nodos_str = [] # NumeroNodo
        Gasto_o_Altura = [] # Gasto_o_Altura
```



```
    Tipo_elemento = [] # Tipo_elemento
    Nivel_Topografico = [] # Nivel_Topografico

    for i in range(self.num_nod):
        # NumeroNodo.append(nn_new) # Despues ordenada
        nn_new += 1
        if List_Tipo_Nodo[i] == 'E':
            # n_anteriores.insert(ee, List_Numero_Nodos[i])
            List_Nodos_str.insert(ee, List_Indx_Nodos[i])
            Gasto_o_Altura.insert(ee, List_Tank_Lev[i])
            Tipo_elemento.insert(ee, List_Tipo_Nodo[i])
            Nivel_Topografico.insert(ee, List_Elev_Nod[i])
            Coord_X.insert(ee, 0)
            Coord_Y.insert(ee, 0)
            ee += 1
        elif List_Tipo_Nodo[i] == 'N':
            # n_anteriores.insert(nn, List_Numero_Nodos[i])
            List_Nodos_str.insert(nn, List_Indx_Nodos[i])
            Gasto_o_Altura.insert(nn, List_DemBase_Nod[i])
            Tipo_elemento.insert(nn, List_Tipo_Nodo[i])
            Nivel_Topografico.insert(nn, List_Elev_Nod[i])
            Coord_X.insert(nn, 0)
            Coord_Y.insert(nn, 0)
            nn += 1

    return Coord_X, Coord_Y, List_Nodos_str, Gasto_o_Altura,
    Tipo_elemento, Nivel_Topografico

def PropiedadesTuberias(self):
    """
    :return:
    """
    l = Link.value_type['EN_LENGTH']
    d = Link.value_type['EN_DIAMETER']
    ml = Link.value_type['EN_MINORLOSS']
    r = Link.value_type['EN_ROUGHNESS']
    List_Tuberias = []
    ini_fin = []
    Long_Tub = []
    Diam_Tub = []
    PM_Tub = []
    Rug_Tub = []
    for i in range(self.num_tub):
        ni = self.ObjTuberias[i + 1].start.id
        nf = self.ObjTuberias[i + 1].end.id
        ini_fin.append([ni, nf])
        Long_Tub.append(round(self.ObjTuberias[i +
1].results[l][0], 2))
        Diam_Tub.append(round(self.ObjTuberias[i +
1].results[d][0], 2))
        PM_Tub.append(self.ObjTuberias[i + 1].results[ml][0])
        Rug_Tub.append(self.ObjTuberias[i + 1].results[r][0])
        List_Tuberias.append(self.ObjTuberias[i + 1].id)
    return List_Tuberias, ini_fin, Long_Tub, Diam_Tub, PM_Tub,
    Rug_Tub
```



```
def CSV_Read(filepath,Nod_or_Tub = None):
    with open(filepath) as File:
        reader = csv.reader(File, delimiter=',', quotechar=',',
                             quoting=csv.QUOTE_MINIMAL)
        NodosTuberias = []
        for row in reader:
            NodosTuberias.append(row)

        NodosTuberias.remove(NodosTuberias[0])

        if Nod_or_Tub == 'Nodos':
            NodosTuberias = sorted(NodosTuberias, key=lambda tipo:
tipo[4])

    return NodosTuberias

def XSLX_Read(fileName, Sht_Nod, Sht_Tub):
    # Cargamos el Libro de Excel a usar
    wb = xlrd.open_workbook(fileName)
    # Buscamos las hojas
    sheet_Nod = wb.sheet_by_name(Sht_Nod)
    rows_n = sheet_Nod.nrows # Numero de renglones (hoja nodos)
    sheet_Tub = wb.sheet_by_name(Sht_Tub)
    rows_t = sheet_Tub.nrows # Numero de renglones (hoja tuberías)
    # Variables auxiliares
    data_n = []; values = []; cont = 0
    # Data de los Nodos
    for j in range(rows_n):
        for i in range(6):
            if cont == 6:
                data_n.append(values)
                values = []
                cont = 0
            n = sheet_Nod.cell_value(j, i)
            values.append(n)
            cont += 1
    data_n.append(values)
    data_n.remove(data_n[0])
    data_n = sorted(data_n, key=lambda tipo: tipo[4])

    # Variables auxiliares
    data_t = []; values = []; cont = 0
    # Data de las Tuberías
    for j in range(rows_t):
        for i in range(7):
            if cont == 7:
                data_t.append(values)
                values = []
                cont = 0
            t = sheet_Tub.cell_value(j, i)
            values.append(t)
            cont += 1
    data_t.append(values)
    data_t.remove(data_t[0])

return data_n, data_t
```



## Anexo D. Archivo: ui\_Functions.py

### ui\_Functions.py

```
from PyQt5 import QtCore, QtWidgets

class UIFunctions(QtWidgets.QMainWindow):
    def toggleMenu(self, maxWidth, enable):

        if enable:
            # Get width
            width = self.ui.frame_left_menu.width()
            maxExtend = maxWidth
            standard = 70

            # Set max width
            if width == 70:
                widthExtended = maxExtend
            else:
                widthExtended = standard

            # Animation
            self.animation =
QtCore.QPropertyAnimation(self.ui.frame_left_menu, b'minimumWidth')
            self.animation.setDuration(400)
            self.animation.setStartValue(width)
            self.animation.setEndValue(widthExtended)

        self.animation.setEasingCurve(QtCore.QEasingCurve.InOutQuart)
        self.animation.start()

    def Fn_Inicializar(self):
        """ Menu Desplegable """
        # Inicializar el programa siempre en la pantalla de inicio
        self.ui.Pages_Widget.setCurrentWidget(self.ui.pagina_0)
        # Quitar Titulo Predefinido de Pantalla
        self.setWindowFlag(QtCore.Qt.FramelessWindowHint)
        self.setAttribute(QtCore.Qt.WA_TranslucentBackground)
        # Ocultar los botones del menu desplegable
        self.ui.btn_page_0 setHidden(True)
        self.ui.btn_page_1 setHidden(True)
        self.ui.btn_page_3 setHidden(True)
        self.ui.btn_page_5 setHidden(True)
        """ PAGINA: Introducir Datos """
        # Ocultar los groupBox y cambiar nombre del boton de carga de
archivos
        self.ui.gb_csv setHidden(True)
        self.ui.gb_excel setHidden(True)
        self.ui.gb_manual setHidden(False)
        self.ui.btn_load_info_epanet setHidden(True)
        self.ui.btn_load_info.setText("Ajustar Tablas")
        # Crear Tabla de Nodos
```



```
self.ui.txt_manual_nod.setValue(8)
#           nod, demanda, tipo, cota   'Tanque'
dataNod = [[22.0, 07.0, 'Embalse', 50.0, 'E', 1.0],
           [00.0, 07.0, 'nodo2', 70.0, 'N', 0.0],
           [15.0, 05.0, 'nodo6', 60.0, 'N', 2.0],
           [00.0, 00.0, 'nodo1', 100.0, 'N', 3.0],
           [15.0, 00.0, 'nodo5', 50.0, 'N', 4.0],
           [22.0, 00.0, 'nodo7', 70.0, 'N', 6.0],
           [06.0, 07.0, 'nodo4', 50.0, 'N', 7.0],
           [06.0, 00.0, 'nodo3', 80.0, 'N', 8.0]]
row = len(dataNod)
col = len(dataNod[0])
for i in range(row):
    self.ui.tbl_nod.insertRow(i)
    for j in range(col):
        celda = QtWidgets.QTableWidgetItem(str(dataNod[i][j]))
        self.ui.tbl_nod.setItem(i, j, celda)
self.ui.tbl_nod.resizeColumnsToContents()
# Crear Tabla de Tuberias
self.ui.txt_manual_tub.setValue(10)
dataTub = [['t1', 'nodo2', 'nodo4', 120.0, 254.0, 4.3, 0.0015],
            ['t2', 'nodo5', 'nodo6', 60.0, 250.0, 2.4, 0.0015],
            ['t3', 'nodo3', 'nodo1', 120.0, 250.0, 4.3, 0.0015],
            ['t4', 'nodo7', 'nodo5', 100.0, 250.0, 3.8, 0.0015],
            ['t5', 'nodo6', 'nodo4', 150.0, 300.0, 4.6, 0.0015],
            ['t6', 'Embalse', 'nodo6', 500.0, 200.0, 3.6,
0.0015],
            ['t7', 'nodo4', 'nodo3', 200.0, 200.0, 5.4, 0.0015],
            ['t8', 'nodo5', 'nodo3', 80.0, 250.0, 3.2, 0.0015],
            ['t9', 'Embalse', 'nodo7', 60.0, 250.0, 2.4,
0.0015],
            ['t10', 'nodo2', 'nodo1', 200.0, 200.0, 5.4, 0.0015]]
row = len(dataTub)
col = len(dataTub[0])
for i in range(row):
    self.ui.tbl_tub.insertRow(i)
    for j in range(col):
        celda = QtWidgets.QTableWidgetItem(str(dataTub[i][j]))
        self.ui.tbl_tub.setItem(i, j, celda)
self.ui.tbl_tub.resizeColumnsToContents()
""" PAGINA: Algoritmos Genéticos """
# Condiciones iniciales para la poblacion
self.ui.txt_tam_pob.setValue(5)
self.ui.txt_num_gen.setValue(5)
self.ui.txt_tasa_mut.setValue(0.05)
# Restricciones de Diseño
self.ui.txt_VDmax.setValue(5)
self.ui.txt_VDmin.setValue(0.6)
self.ui.txt_PDmax.setValue(50)
self.ui.txt_PDmin.setValue(15)
# Crear Tabla de Diametros a Usar
self.ui.txt_num_diametros.setValue(19)

dataDcom = [[50, 12.66], [60, 15.64], [75, 14.69], [100, 33.06],
[150, 38.21], [200, 44.54],
[250, 52.09], [300, 60.08]]
```



```
"""dataDcom = [[50, 12.66], [75, 14.69], [100, 33.06], [150,
38.21], [200, 44.54],
[250, 52.09], [300, 60.08], [350, 70.05], [400,
77.99], [450, 86.63],
[500, 97.39], [600, 112.29], [750, 130.60], [800,
153.96], [1000, 173.27],
[1200, 203.32], [1400, 232.70], [1500, 247.17],
[1800, 289.81]]"""
row = len(dataDcom)
col = len(dataDcom[0])
for i in range(row):
    self.ui.tbl_Dcom.insertRow(i)
    for j in range(col):
        celda = QtWidgets.QTableWidgetItem(str(dataDcom[i][j]))
        self.ui.tbl_Dcom.setItem(i, j, celda)
self.ui.tbl_Dcom.resizeColumnsToContents()
# Crear Tabla de Mejores Diametros
self.ui.tbl_mejor_diam.resizeColumnsToContents()

def Fn_Buttons(self):
    """ Menu Desplegable """
    # Cerrar aplicacion
    self.ui.btn_close.clicked.connect(self.CerrarApp)
    # Minimizar la aplicacion
    self.ui.btn_max.clicked.connect(self.Minim)
    # Movernos entre Paginas
    self.ui.btn_page_0.clicked.connect(lambda:
self.ui.Pages_Widget.setCurrentWidget(self.ui.pagina_0))
    self.ui.btn_page_1.clicked.connect(lambda:
self.ui.Pages_Widget.setCurrentWidget(self.ui.pagina_1))
    self.ui.btn_page_3.clicked.connect(lambda:
self.ui.Pages_Widget.setCurrentWidget(self.ui.pagina_3))
    self.ui.btn_page_5.clicked.connect(lambda:
self.ui.Pages_Widget.setCurrentWidget(self.ui.pagina_5))

    """ PAGINA: Introducir Datos """
    # Botones de Carga de Datos para Manual:
    self.ui.btn_load_info.clicked.connect(self.Atr_MANUAL)
    # Botones de Carga de Datos para Excel:
    self.ui.btn_load_info.clicked.connect(self.Atr_XLSX)
    # Botones de Carga de Datos para CSV
    self.ui.btn_csv_nod.clicked.connect(self.Atr_CSVnod)
    self.ui.btn_csv_tub.clicked.connect(self.Atr_CSVtub)
    # Botones de Carga de Datos para Epanet
    self.ui.btn_load_info_epanet.clicked.connect(self.Atr_INP)
    self.ui.btn_load_info_XY.clicked.connect(self.Atr_INP_XY)
    # Botones que permite visualizar la red hidraulica:

    self.ui.btn_visualizar.clicked.connect(self.Atr_DibujarRedHidraulicaSeguros)

    """ PAGINA: Analisis de la Red Hidraulica """

    self.ui.btn_AnalisisRedHid.clicked.connect(self.Atr_AnalisisRedHidraulicaSeguros)
```



```
""" PAGINA: Algoritmo Genetico"""
self.ui.btn_AG.clicked.connect(self.Atr_AlgoritmoGenetico)

self.ui.btn_guardar_diam_temp.clicked.connect(self.Fn_ResultadosTemporales)
    self.ui.btn_exportar_diam.clicked.connect(self.Fn_ExportarDiam)

self.ui.btn_cargar_diam.clicked.connect(self.Fn_CargarListaDiametros)

def Fn_RadioButtons(self):
    """ PAGINA: Introducir Datos """
    self.ui.rb_manual.clicked.connect(lambda:
self.Atr_RadioButton_Interactivo("man"))
        self.ui.rb_inp.clicked.connect(lambda:
self.Atr_RadioButton_Interactivo("inp"))
        self.ui.rb_csv.clicked.connect(lambda:
self.Atr_RadioButton_Interactivo("csv"))
        self.ui.rb_xlsx.clicked.connect(lambda:
self.Atr_RadioButton_Interactivo("xlsx"))

    """ PAGINA: Algoritmo Genetico"""
    self.ui.rb_lineal.clicked.connect(lambda:
self.Atr_RadioButton_Regresiones())
        self.ui.rb_potencial.clicked.connect(lambda:
self.Atr_RadioButton_Regresiones())
        self.ui.rb_exponencial.clicked.connect(lambda:
self.Atr_RadioButton_Regresiones())

def Wgt_SpinBox(self):
    # Añadir renglones a la tabla de diámetros y precios

self.ui.txt_num_diametros.valueChanged.connect(self.Prop_AnadirRenglonesInCombo)
```

## Anexo E. Archivo: clases\_mpl\_pyqt.py

### clases\_mpl\_pyqt.py

```
from PyQt5 import QtWidgets
from matplotlib.figure import Figure
from matplotlib.backends.backend_qt4agg import FigureCanvasQTAgg as
FigureCanvas
from matplotlib.backends.backend_qt4agg import NavigationToolbar2QT as
NavigationToolbar

class MplCanvas11(FigureCanvas):
    def __init__(self):
        self.fig = Figure(facecolor="#23395B")
        self.ax = self.fig.add_subplot(111)
        FigureCanvas.__init__(self, self.fig)
```



```
class MplCanvas22(FigureCanvas):
    def __init__(self):
        self.fig = Figure(facecolor="0.94")
        self.ax1 = self.fig.add_subplot(221)
        self.ax2 = self.fig.add_subplot(222)
        self.ax3 = self.fig.add_subplot(223)
        self.ax4 = self.fig.add_subplot(224)
        FigureCanvas.__init__(self, self.fig)

class MatplotlibWidget11(QtWidgets.QWidget):
    def __init__(self, parent = None):
        QtWidgets.QWidget.__init__(self, parent)
        self.canvas = MplCanvas11()
        self.vbl = QtWidgets.QVBoxLayout()
        self.vbl.addWidget(self.canvas)
        self.toolbar = NavigationToolbar(self.canvas, self)
        self.vbl.addWidget(self.toolbar)
        self.setLayout(self.vbl)

class MatplotlibWidget11_sin(QtWidgets.QWidget):
    def __init__(self, parent = None):
        QtWidgets.QWidget.__init__(self, parent)
        self.canvas = MplCanvas11()
        self.vbl = QtWidgets.QVBoxLayout()
        self.vbl.addWidget(self.canvas)
        self.setLayout(self.vbl)

class MatplotlibWidget22(QtWidgets.QWidget):
    def __init__(self, parent = None):
        QtWidgets.QWidget.__init__(self, parent)
        self.canvas = MplCanvas22()
        self.vbl = QtWidgets.QVBoxLayout()
        self.vbl.addWidget(self.canvas)
        self.toolbar = NavigationToolbar(self.canvas, self)
        self.vbl.addWidget(self.toolbar)
        self.setLayout(self.vbl)

class MatplotlibWidget22_sin(QtWidgets.QWidget):
    def __init__(self, parent = None):
        QtWidgets.QWidget.__init__(self, parent)
        self.canvas = MplCanvas22()
        self.vbl = QtWidgets.QVBoxLayout()
        self.vbl.addWidget(self.canvas)
        self.setLayout(self.vbl)
```

---

## Anexo F. Archivo: progressBarPYQT5.py

### progressBarPYQT5.py

```
import sys
from splash_screen import *

class ProgBar(QtWidgets.QMainWindow):
```



```
def __init__(self, parent = None):
    QtWidgets.QWidget.__init__(self, parent)
    self.ui = Ui_SplashScreen2()
    self.ui.setupUi(self)

    # REMOVE TITLE FROM BAR
    self.setWindowFlag(QtCore.Qt.FramelessWindowHint)
    # BACKGROUND TO TRANSPARENT
    self.setAttribute(QtCore.Qt.WA_TranslucentBackground)

    self.progressBarValue(0,1)
    self.show()

    # GIVE VALUE TO PROGRESSBAR
def progressBarValue(self, value, total):
    # PROGRESS STYLESHEET BASE:
    stylesheet = """
    QFrame{
        border-radius: 150px;
        background-color: qconicalgradient(cx:0.5, cy:0.5, angle:90,
stop:{STOP_1}
            rgba(255, 85, 255, 0), stop:{STOP_2} rgba(0, 170, 255, 255));
    }
    """

    # GET PROGRESS BAR VALUE, CONVERT TO FLOAT AND INVERT VALUES
    # STOP WORKS TO 1.000 TO 0.000
    progress = (total - value)/float(total)

    # GET NEW VALUES
    stop_1 = str(progress-0.001)
    stop_2 = str(progress)

    # SET VALUES TO NEW STYLESHEET
    newstylesheet =
stylesheet.replace("{STOP_1}",stop_1).replace("{STOP_2}",stop_2)

    #APPLY STYLESHEET WITH NEW VALUES
    self.ui.circularProgress.setStyleSheet(newstylesheet)
    self.ui.labelPorcentaje.setText('<html><head/><body><p>' +
                                    str(int(value*100/total)) +
                                    '<span style=" vertical-
align:super;">%</span></p></body></html>')

if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    myapp = ProgBar()
    myapp.show()
    sys.exit(app.exec_())
```



## Anexo G. Archivo: ErrorPYQT5.py

### ErrorPYQT5.py

```
import sys
from ui_Error import *

class ErrorWindow(QtWidgets.QMainWindow):
    def __init__(self, parent = None):
        QtWidgets.QWidget.__init__(self, parent)
        self.ui = Ui_UI_Error()
        self.ui.setupUi(self)

        # REMOVE TITLE FROM BAR
        self.setWindowFlag(QtCore.Qt.FramelessWindowHint)
        # BACKGROUND TO TRANSPARENT
        self.setAttribute(QtCore.Qt.WA_TranslucentBackground)

        # BUTTON
        self.ui.btnClose.clicked.connect(self.close)

        self.show()

        self.TextoError("Error de Presión")

    def TextoError(self, texto):
        self.ui.label.setText(texto)

if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    myapp = ErrorWindow()
    myapp.show()
    sys.exit(app.exec_())
```

## Anexo H. Archivo: MultiMenu.py

### MultiMenu.py

```
import sys
from os import mkdir as newDir
import math
from ui_ProgramAG import *
from ui_Functions import *
from ErrorPYQT5 import ErrorWindow
from FileTools import *
from clases_mpl_pyqt import *
import GradienteHidráulico as gh
import AlgoritmoGenético as ag

class UIMainWindow(QtWidgets.QMainWindow):
    def __init__(self, parent = None):
        QtWidgets.QWidget.__init__(self, parent)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
```



```
"""
    """
    MANEJADOR DE EVENTOS
    """

# Dar capacidad de mover la pantalla
def moveWindow(event):
    #
    if event.buttons() == QtCore.Qt.LeftButton:
        self.move(self.pos() + event.globalPos() - self.dragPos)
        self.dragPos = event.globalPos()
        event.accept()
    self.ui.frame_top.mouseMoveEvent = moveWindow

# Abrir y cerrar menu principal
self.ui.frame_left_menu.enterEvent = lambda
e:self.Fn_MenuInteractivo(e)
self.ui.frame_left_menu.leaveEvent = lambda
e:self.Fn_MenuInteractivo(e)

""" Acciones a realizar cuando se incia el programa """
UIFunctions.Fn_Inicializar(self)

""" Acciones que realizan los Botones """
UIFunctions.Fn_Botons(self)

""" Acciones que realizan los RadioButtons """
UIFunctions.Fn_RadioButtons(self)
self.ui.rb_lineal.click()

''' Añadir renglones a la tabla de diametros y precios '''
UIFunctions.Wgt_SpinBox(self)

"""

#####
# ATRIBUTOS DE LA GUI #
#####

"""

def Fn_MenuInteractivo(self,e):
    """Funcion para hacer interactivo el menu"""
    if e.type() == 10:
        UIFunctions.toggleMenu(self, 400, True)
        self.ui.btn_page_0 setHidden(False)
        self.ui.btn_page_1 setHidden(False)
        self.ui.btn_page_3 setHidden(False)
        self.ui.btn_page_5 setHidden(False)
    elif e.type() == 11:
        UIFunctions.toggleMenu(self, 400, True)
        self.ui.btn_page_0 setHidden(True)
        self.ui.btn_page_1 setHidden(True)
        self.ui.btn_page_3 setHidden(True)
        self.ui.btn_page_5 setHidden(True)

def mousePressEvent(self, event):
    """Funcion para arrastrar la pantalla"""
    self.dragPos = event.globalPos()
```



```
def CerrarApp(self):
    #Cerrar la aplicacion
    self.close()

def Minim(self):
    """Funcion para minimizar la pantalla"""
    self.showMinimized()

def msgErrorBox(self,mensage):
    print("si")
    self.msgError = ErrorWindow()
    self.msgError.show()
    self.msgError.TextoError(mensage)

def Fn_EmptyTable(self, table):
    val = 0
    row = table.rowCount()
    col = table.columnCount()
    for i in range(row):
        for j in range(col):
            if table.item(i, j).text() == "":
                val+=1
    return val

"""

#####
# PAGINA: ANALISIS DE REDES HIDRAULICAS #
#####
"""

def Atr_RadioButton_Interactivo(self,val):
    """Funcion para hacer interactivo el grupo datos
    oculta o muestra los diferentes grupos de datos
    del groupBox"""
    if val == "man":
        self.ui.gb_csv setHidden(True)
        self.ui.gb_excel setHidden(True)
        self.ui.gb_manual setHidden(False)
        self.ui.btn_load_info setHidden(False)
        self.ui.btn_load_info_epanet setHidden(True)
        self.ui.btn_load_info_XY setHidden(True)
        self.ui.btn_load_info.setText("Ajustar Tablas")
    elif val == "csv":
        self.ui.gb_csv setHidden(False)
        self.ui.gb_manual setHidden(True)
        self.ui.gb_excel setHidden(True)
        self.ui.btn_load_info setHidden(True)
        self.ui.btn_load_info_epanet setHidden(True)
        self.ui.btn_load_info_XY setHidden(True)
    elif val == "xlsx":
        self.ui.gb_csv setHidden(True)
        self.ui.gb_manual setHidden(True)
        self.ui.gb_excel setHidden(False)
        self.ui.btn_load_info setHidden(False)
```



```
        self.ui.btn_load_info_epanet.setHidden(True)
        self.ui.btn_load_info_XY.setHidden(True)
        self.ui.btn_load_info.setText("Cargar Excel")
    elif val == "inp":
        self.ui.gb_csv setHidden(True)
        self.ui.gb_manual setHidden(True)
        self.ui.gb_excel setHidden(True)
        self.ui.btn_load_info setHidden(True)
        self.ui.btn_load_info_XY setHidden(False)
        self.ui.btn_load_info_epanet setHidden(False)

    def Fn_BuscadorDeArchivos(self, formato):
        """Abrir el buscador de Archivos"""
        options = QtWidgets.QFileDialog.Options()
        #options |= QtWidgets.QFileDialog.DontUseNativeDialog
        if formato == "csv":
            fileName, _ = QtWidgets.QFileDialog.getOpenFileName(self,
"QFileDialog.getOpenFileName()", "")
"Archivos csv (" + formato + ")", options=options)
        if fileName:
            return fileName

        if formato == "xlsx":
            fileName, _ = QtWidgets.QFileDialog.getOpenFileName(self,
"QFileDialog.getOpenFileName()", "")
"Archivos xlsx (" + formato + ")", options=options)
        if fileName:
            return fileName

        if formato == "inp":
            fileName, _ = QtWidgets.QFileDialog.getOpenFileName(self,
"QFileDialog.getOpenFileName()", "")
"Archivos inp (" + formato + ")", options=options)
        if fileName:
            return fileName

    def Atr_MANUAL(self):
        """Ajuste de tabla a modo manual para su llenado"""
        if self.ui.rb_manual.isChecked() == True:

            # Fijamos el número de nodos a capturar
            row_n = self.ui.txt_manual_nod.value()
            if row_n == 0:
                self.msgErrorBox("NO SE INDICÓ EL NÚMERO DE NODOS")
            else:
                row_n = self.ui.txt_manual_nod.value()
                self.ui.tbl_nod.setRowCount(row_n)
                self.ui.tbl_nod.resizeColumnsToContents()
                self.ui.tbl_nod.clearContents()

            # Fijamos el número de tuberías a capturar
            row_t = self.ui.txt_manual_tub.value()
            if row_t == 0:
```



```
        self.msgErrorBox("NO SE INDICÓ EL NÚMERO DE TUBERÍAS")
    else:
        self.ui.tbl_tub.setRowCount(row_t)
        self.ui.tbl_tub.resizeColumnsToContents()
        self.ui.tbl_tub.clearContents()

def Atr_XLSX(self):
    """Llenado de tablas de nodos y tuberías usando un archivo de excel"""
    if self.ui.rb_xlsx.isChecked() == True:
        try:
            # Abre el buscador de archivos *.xlsx
            ruta_nombre = self.Fn_BuscadorDeArchivos("xlsx")
            # Obtiene el nombre de las hojas donde están los nodos y tuberías
            Sht_Nod = self.ui.txt_excel_nod.text()
            Sht_Tub = self.ui.txt_excel_tub.text()
            # Vacía toda la información de nodos y tuberías a su respectiva tabla
            data_n, data_t = FileManipulation.XSLX_Read(ruta_nombre, Sht_Nod, Sht_Tub)
            self.Atr_LlenadoTablaNod_CSVXSLX(data_n)
            self.Atr_LlenadoTablaTub_CSVXSLX(data_t)
        except:
            self.msgErrorBox("ERROR EN LOS NOMBRES DE LAS HOJAS")

def Atr_CSVnod(self):
    """Llenado de tabla de nodo usando un archivo de csv"""
    if self.ui.rb_csv.isChecked() == True:
        try:
            ruta_nombre = self.Fn_BuscadorDeArchivos("csv")
            datos = FileManipulation.CSV_Read(ruta_nombre, 'Nodos')
            self.Atr_LlenadoTablaNod_CSVXSLX(datos)
        except:
            self.msgErrorBox("ERROR AL CARGAR ARCHIVO *.csv")

def Atr_CSVtub(self):
    """Llenado de tabla de tuberías usando un archivo de csv"""
    if self.ui.rb_csv.isChecked() == True:
        try:
            ruta_nombre = self.Fn_BuscadorDeArchivos("csv")
            datos = FileManipulation.CSV_Read(ruta_nombre)
            self.Atr_LlenadoTablaTub_CSVXSLX(datos)
        except:
            self.msgErrorBox("ERROR AL CARGAR ARCHIVO *.csv")

def Atr_INP(self):
    if self.ui.rb_inp.isChecked() == True:
        try:
            ruta_nombre = self.Fn_BuscadorDeArchivos("inp")
            if ruta_nombre == None:
                self.msgErrorBox("NO SE CARGO EL ARCHIVO *.inp")
            else:
                EpTools = FileManipulation(ruta_nombre)
                EpTools.PropiedadesNodos()
                Prop_Nod = EpTools.PropiedadesNodos()
```



```
Prop_Tub = EpTools.PropiedadesTuberias()
self.Atr_LlenadoTablaNodTub_INP(Prop_Nod, Prop_Tub)
except:
    self.msgErrorBox("ERROR AL CARGAR ARCHIVO *.inp")

def Atr_LlenadoTablaNodTub_INP(self, datos_n, datos_t, datos_XY =
[]):
    """Vaciamos los datos de los nodos a su tabla"""
    row = len(datos_n[0])
    self.ui.tbl_nod.clearContents()
    self.ui.tbl_nod.setRowCount(0)
    for i in range(0, row):
        self.ui.tbl_nod.insertRow(i)
        coordX = QtWidgets.QTableWidgetItem(str(datos_n[0][i]))
        coordY = QtWidgets.QTableWidgetItem(str(datos_n[1][i]))
        celda_nod = QtWidgets.QTableWidgetItem(str(datos_n[2][i]))
        celda_q = QtWidgets.QTableWidgetItem(str(datos_n[3][i]))
        celda_type = QtWidgets.QTableWidgetItem(str(datos_n[4][i]))
        celda_z = QtWidgets.QTableWidgetItem(str(datos_n[5][i]))
        self.ui.tbl_nod.setItem(i, 0, coordX) # Coordenda X
        self.ui.tbl_nod.setItem(i, 1, coordY) # Coordenada Y
        self.ui.tbl_nod.setItem(i, 2, celda_nod) # Id del nodo
        self.ui.tbl_nod.setItem(i, 3, celda_q) # Gasto o Altura
        self.ui.tbl_nod.setItem(i, 4, celda_type) # Tipo del Nodo
        self.ui.tbl_nod.setItem(i, 5, celda_z) # Nivel
Topografico
    self.ui.tbl_nod.resizeColumnsToContents()

    """Vaciamos los datos de las tuberias a su tabla"""
    row = len(datos_t[0])
    self.ui.tbl_tub.clearContents()
    self.ui.tbl_tub.setRowCount(0)
    ini_fin = datos_t[1]
    for i in range(0, row):
        self.ui.tbl_tub.insertRow(i)
        celda_tub = QtWidgets.QTableWidgetItem(str(datos_t[0][i]))
        celda_ni = QtWidgets.QTableWidgetItem(str(ini_fin[i][0]))
        celda_nf = QtWidgets.QTableWidgetItem(str(ini_fin[i][1]))
        celda_l = QtWidgets.QTableWidgetItem(str(datos_t[2][i]))
        celda_d = QtWidgets.QTableWidgetItem(str(datos_t[3][i]))
        celda_km =
QtWidgets.QTableWidgetItem(str(round(datos_t[4][i], 4)))
        celda_ks =
QtWidgets.QTableWidgetItem(str(round(datos_t[5][i], 6)))
        self.ui.tbl_tub.setItem(i, 0, celda_tub) # ID de la tuberia
        self.ui.tbl_tub.setItem(i, 1, celda_ni) # Nodo inicial
        self.ui.tbl_tub.setItem(i, 2, celda_nf) # Nodo final
        self.ui.tbl_tub.setItem(i, 3, celda_l) # Longitud
        self.ui.tbl_tub.setItem(i, 4, celda_d) # Diametro
        self.ui.tbl_tub.setItem(i, 5, celda_km) # Perdidas Menores
        self.ui.tbl_tub.setItem(i, 6, celda_ks) # Rugosidad
    self.ui.tbl_nod.resizeColumnsToContents()

def Atr_INP_XY(self):
    """Llenado de coordenadas usando un archivo de csv"""
    if self.ui.rb_inp.isChecked() == True:
```



```
try:
    ruta_nombre = self.Fn_BuscadorDeArchivos("csv")
    datos = FileManipulation.CSV_Read(ruta_nombre)
    self.Atr_LlenadoCoordenadas_XY(datos)
except:
    self.msgErrorBox("ERROR AL CARGAR COORDENADAS *.csv")

def Atr_LlenadoCoordenadas_XY(self, datos_XY):
    """Vaciamos los datos de las tuberías a su tabla"""
    row = len(datos_XY)
    for i in range(0, row):
        x = round(float(datos_XY[i][1]), 2)
        y = round(float(datos_XY[i][2]), 2)
        coordX = QtWidgets.QTableWidgetItem(str(x))
        coordY = QtWidgets.QTableWidgetItem(str(y))
        self.ui.tbl_nod.setItem(i, 0, coordX) # Coordenada X
        self.ui.tbl_nod.setItem(i, 1, coordY) # Coordenada Y
    self.ui.tbl_nod.resizeColumnsToContents()

def Atr_LlenadoTablaNod_CSVXSLX(self, datos): # --> OK
    """Vaciamos los datos de los nodos a su tabla"""
    row = len(datos)
    col = len(datos[0])
    self.ui.tbl_nod.clearContents()
    self.ui.tbl_nod.setRowCount(0)
    for i in range(row): # range(0, row)
        self.ui.tbl_nod.insertRow(i) # i-1
        for j in range(col):
            celda = QtWidgets.QTableWidgetItem(str(datos[i][j]))
            self.ui.tbl_nod.setItem(i, j, celda) # i-1
    self.ui.tbl_nod.resizeColumnsToContents()

# CargarTuberias_CSVXSLX
def Atr_LlenadoTablaTub_CSVXSLX(self, datos): # --> OK
    """Vaciamos los datos de las tuberías a su tabla"""
    row = len(datos)
    col = len(datos[0])
    self.ui.tbl_tub.clearContents()
    self.ui.tbl_tub.setRowCount(0)
    for i in range(row):
        self.ui.tbl_tub.insertRow(i)
        for j in range(col):
            celda = QtWidgets.QTableWidgetItem(str(datos[i][j]))
            self.ui.tbl_tub.setItem(i, j, celda)
    self.ui.tbl_tub.resizeColumnsToContents()

"""

#####
# PAGINA: VISUALIZACIÓN DE LA RED HIDRAULICA #
#####

def Atr_DibujarRedHidraulicaSeguros(self):
    if self.Fn_EmptyTable(self.ui.tbl_nod) > 0:
        self.msgErrorBox("CELDA VACIA EN LA TABLA DE NODOS")
```



```
elif self.Fn_EmptyTable(self.ui.tbl_tub) > 0:  
    self.msgErrorBox("CELDA VACIA EN LA TABLA DE TUBERIAS")  
elif self.Fn_EmptyTable(self.ui.tbl_nod) == 0 and  
self.Fn_EmptyTable(self.ui.tbl_tub) == 0:  
    try:  
        self.Atr_DibujarRedHidraulica()  
    except:  
        self.msgErrorBox("ERROR EN LOS VALORES DE LAS TABLAS")  
  
def Atr_DibujarRedHidraulica(self):  
    # Extraer datos de la tabla de nodos  
    [Vec_Nodx, Vec_Nody, Vec_Nodo, Vec_tipo, Vec_QH, Vec_cota,  
T_str2num, T_num2str] = self.Fn_Nodos()  
  
    # Extraer datos de la tabla de tuberias  
    [Vec_tuberia, ini_fin, Vec_Long, Vec_diam, Vec_km, Vec_ks] =  
self.Fn_Tuberias()  
  
    # Acomodar data  
    ndx_coord = []  
    ndy_coord = []  
    nex_coord = []  
    ney_coord = []  
    num_ren_nod = len(Vec_Nodo)  
    for i in range(num_ren_nod):  
        if Vec_tipo[i] == 'E':  
            nex_coord.append(Vec_Nodx[i])  
            ney_coord.append(Vec_Nody[i])  
        elif Vec_tipo[i] == 'N':  
            ndx_coord.append(Vec_Nodx[i])  
            ndy_coord.append(Vec_Nody[i])  
  
    # Dibujar Tuberias (Lineas)  
    self.ploteo = self.ui.widget_Visualizar_RH.canvas  
    self.ploteo.ax.clear()  
    cont = 0  
    for vec in ini_fin:  
        n1x = Vec_Nodx[T_str2num[vec[0]] - 1]  
        n1y = Vec_Nody[T_str2num[vec[0]] - 1]  
        n2x = Vec_Nodx[T_str2num[vec[1]] - 1]  
        n2y = Vec_Nody[T_str2num[vec[1]] - 1]  
        self.ploteo.ax.plot([n1x, n2x], [n1y, n2y], 'c-',  
linewidth=5)  
  
        tx = 0.5 * (n1x + n2x)  
        ty = 0.5 * (n1y + n2y)  
  
        # Visualizacion de los Id's para las tuberias (Solamente)  
        if self.ui.rb_IdTuberias.isChecked() == True or  
self.ui.rb_IdNodTub.isChecked() == True:  
            self.ploteo.ax.text(tx, ty, Vec_tuberia[cont],  
weight="bold")  
  
        # Propiedades de las Tuberias en funcion del CheckBox  
        if self.ui.rb_Long.isChecked() == True:  
            lbl = str(Vec_Long[cont]) + "mts"  
            self.ploteo.ax.text(tx, ty - 0.5, lbl, weight="bold")
```



```
        elif self.ui.rb_Diam.isChecked() == True:
            lbl = str(Vec_diam[cont]) + "mm"
            self.ploteo.ax.text(tx, ty - 0.5, lbl, weight="bold")
        elif self.ui.rb_ks.isChecked() == True:
            lbl = str(Vec_ks[cont]) + "mm"
            self.ploteo.ax.text(tx, ty - 0.5, lbl, weight="bold")
        elif self.ui.rb_km.isChecked() == True:
            lbl = str(Vec_km[cont])
            self.ploteo.ax.text(tx, ty - 0.5, lbl, weight="bold")
        cont+=1

    # Dibujar Nodos (Circulos)
    self.ploteo.ax.plot(ndx_coord, ndy_coord, 'ok')

    # Dibujar Embalse (Diamantes)
    self.ploteo.ax.plot(nex_coord, ney_coord, 'Dm')

    # Visualizaciones de Id's para los nodos (Solamente)
    for i in range(num_ren_nod):
        if self.ui.rb_IdNodos.isChecked() == True or
self.ui.rb_IdNodTub.isChecked() == True:
            self.ploteo.ax.text(Vec_Nodx[i], Vec_Nody[i],
T_num2str[Vec_Nodo[i]], weight="bold")

        # Propiedades de los Nodos en función del CheckBox
        if self.ui.rb_Demandas.isChecked() == True:
            if Vec_tipo[i] == 'E':
                lbl = str(Vec_QH[i]) + "mts"
            else:
                lbl = str(Vec_QH[i]) + "L/s"
            self.ploteo.ax.text(Vec_Nodx[i]+0.5, Vec_Nody[i]+0.5,
lbl, weight="bold")
        elif self.ui.rb_Cotas.isChecked() == True:
            lbl = str(Vec_cota[i]) + "mts"
            self.ploteo.ax.text(Vec_Nodx[i] + 0.5, Vec_Nody[i] +
0.5, lbl, weight="bold")

    # Ajustar Ejes
    xmax = max(Vec_Nodx)
    ymax = max(Vec_Nody)
    xmin = min(Vec_Nodx)
    ymin = min(Vec_Nody)
    self.ploteo.ax.axis([-xmax*0.5, xmax*1.5, -ymax*0.5, ymax*1.5])

    # Color de fondo del mpl
    self.ploteo.ax.set_facecolor('#FFFFFF')

    # Ajuste de los parámetros del plot
    self.ploteo.fig.subplots_adjust(0, 0, 1, 1)  # left,bottom,right,top

    # Mostrar el dibujo en el mpl
    self.ploteo.draw()

"""
#####
# PAGINA: ANALISIS DE LA RED HIDRAULICA #

```



```
#####
"""
def Fn_Nodos(self):
    Vec_Nodx = []
    Vec_Nody = []
    Vec_Nodo = []
    Vec_Nodo_str = []
    Vec_tipo = []
    Vec_QH = []
    Vec_cota = []
    # Extraer datos de la tabla de nodos
    num_ren_nod = self.ui.tbl_nod.rowCount()
    for i in range(num_ren_nod):
        Coord_x = float(self.ui.tbl_nod.item(i, 0).text())
        Coord_y = float(self.ui.tbl_nod.item(i, 1).text())
        nodo = self.ui.tbl_nod.item(i, 2).text()
        qh = float(self.ui.tbl_nod.item(i, 3).text())
        Tipo_nodo = self.ui.tbl_nod.item(i, 4).text()
        cota = float(self.ui.tbl_nod.item(i, 5).text())
        Vec_Nodx.append(Coord_x)
        Vec_Nody.append(Coord_y)
        Vec_Nodo.append(i + 1)
        Vec_Nodo_str.append(nodo)
        Vec_QH.append(qh)
        Vec_tipo.append(Tipo_nodo)
        Vec_cota.append(cota)
    T_str2num = dict(zip(Vec_Nodo_str, Vec_Nodo))
    T_num2str = dict(zip(Vec_Nodo, Vec_Nodo_str))
    return Vec_Nodx, Vec_Nody, Vec_Nodo, Vec_tipo, Vec_QH, Vec_cota,
T_str2num, T_num2str

def Fn_Tuberias(self):
    # Extraer datos de la tabla de tuberías
    Vec_tuberia = []
    ini_fin = []
    Vec_Long = []
    Vec_diam = []
    Vec_km = []
    Vec_ks = []
    num_ren_tub = self.ui.tbl_tub.rowCount()
    for i in range(num_ren_tub):
        tuberia = self.ui.tbl_tub.item(i, 0).text()
        ni = self.ui.tbl_tub.item(i, 1).text()
        nf = self.ui.tbl_tub.item(i, 2).text()
        L = float(self.ui.tbl_tub.item(i, 3).text())
        diam = float(self.ui.tbl_tub.item(i, 4).text())
        km_val = float(self.ui.tbl_tub.item(i, 5).text())
        ks_val = float(self.ui.tbl_tub.item(i, 6).text())
        Vec_tuberia.append(tuberia)
        ini_fin.append([ni, nf])
        Vec_Long.append(L)
        Vec_diam.append(diam)
        Vec_km.append(km_val)
        Vec_ks.append(ks_val)
    return Vec_tuberia, ini_fin, Vec_Long, Vec_diam, Vec_km, Vec_ks
```



```
def Atr_AnalisisRedHidraulicaSeguros(self):
    if self.Fn_EmptyTable(self.ui.tbl_nod) > 0:
        self.msgErrorBox("CELDA VACIA EN LA TABLA DE NODOS")
    elif self.Fn_EmptyTable(self.ui.tbl_tub) > 0:
        self.msgErrorBox("CELDA VACIA EN LA TABLA DE TUBERÍAS")
    elif self.Fn_EmptyTable(self.ui.tbl_nod) == 0 and
self.Fn_EmptyTable(self.ui.tbl_tub) == 0:
        try:
            self.Atr_AnalisisRedHidraulica()
        except:
            self.msgErrorBox("ERROR EN LOS VALORES DE LAS TABLAS")

    def Atr_AnalisisRedHidraulica(self):
        [Vec_Nodx, Vec_Nody, Vec_Nodo, Vec_tipo, Vec_QH, Vec_cota,
T_str2num, T_num2str] = self.Fn_Nodos()
        [Vec_tuberia, ini_fin, Vec_Long, Vec_diam, Vec_km, Vec_ks] =
self.Fn_Tuberias()

        data_nod = []
        for i in range(len(Vec_Nodo)):
            data_nod.append([Vec_Nodo[i], Vec_QH[i], Vec_tipo[i],
Vec_cota[i]])

        data_tub = []
        for i in range(len(Vec_tuberia)):
            data_tub.append([Vec_tuberia[i], T_str2num[ini_fin[i][0]],
T_str2num[ini_fin[i][1]],
                Vec_Long[i], Vec_diam[i], round(Vec_km[i],
2), round(Vec_ks[i], 4)])

        self.PmaxD = self.ui.txt_PDmax.value()
        self.PminD = self.ui.txt_PDmin.value()
        self.VmaxD = self.ui.txt_VDmax.value()
        self.VminD = self.ui.txt_VDmin.value()

        # Extraer Data de la Tabla de Precio-Diametro
        x = []
        self.DiamCom = []
        num_ren_diam = self.ui.tbl_Dcom.rowCount()
        for i in range(num_ren_diam):
            diam = float(self.ui.tbl_Dcom.item(i, 0).text())
            x.append(diam)
        self.DiamCom = x

        rh = gh.RedHidraulica(data_nod, data_tub, self.PmaxD,
self.PminD, self.VmaxD, self.VminD, self.DiamCom)

        Tuberias = rh.Tub
        Nodos = rh.Nod
        Grad = gh.Gradiente(Tuberias, Nodos)
        [Caudal, Velocidad, Altura, Presion, Fricc, Perd, er, ini_fin] =
Grad.Resolver()

        self.ui.lbl_vmax.setText('<html><head/><body><p><span style="
color:#55ffff; font-weight:600;">' +
'<span style="color:#55ffff; font-weight:600;">' +
'MÁXIMA: {:.2f}
metros/seg'.format(max(Velocidad)[0]) +
```



```
'</span></p></body></html>')
self.ui.lbl_vmin.setText('<html><head/><body><p><span style="color:#55ffff; font-weight:600;">' +
'MÍNIMA: {:.2f}'
metros/seg'.format(min(Velocidad)[0]) +
'</span></p></body></html>')
self.ui.lbl_pmax.setText('<html><head/><body><p><span style="color:#55ffff; font-weight:600;">' +
'MÁXIMA: {:.2f}'
metros'.format(max(Presion)[0]) +
'</span></p></body></html>')
self.ui.lbl_pmin.setText('<html><head/><body><p><span style="color:#55ffff; font-weight:600;">' +
'MÍNIMA: {:.2f}'
metros'.format(min(Presion)[0]) +
'</span></p></body></html>')
err_porcentual = er[-1]
if err_porcentual <= 0.00001:
    self.ui.lbl_err.setText('<html><head/><body><p><span style="color:#55ffff; font-weight:600;">' +
'EJECUCIÓN EXITOSA: {:.9f}'
%.format(err_porcentual) +
'</span></p></body></html>')
else:
    self.ui.lbl_err.setText('<html><head/><body><p><span style="color:#ff0000; font-weight:600;">' +
'MALA EJECUCIÓN: {:.6f}'
%.format(err_porcentual) +
'</span></p></body></html>')

"""\Vaciamos los resultados de las tuberías a su tabla"""
row = len(Caudal)
self.ui.tbl_res_tub.clearContents()
self.ui.tbl_res_tub.setRowCount(0)
for i in range(row):
    self.ui.tbl_res_tub.insertRow(i)

    t = QtWidgets.QTableWidgetItem(str(Vec_tuberia[i]))
    ni =
QtWidgets.QTableWidgetItem(str(T_num2str[ini_fin[i][0]]))
    nf =
QtWidgets.QTableWidgetItem(str(T_num2str[ini_fin[i][1]]))
    q = QtWidgets.QTableWidgetItem(str(round(Caudal[i][0],2)))
    v =
QtWidgets.QTableWidgetItem(str(round(Velocidad[i][0],2)))
    hf = QtWidgets.QTableWidgetItem(str(round(Perd[i][0],2)))
    f = QtWidgets.QTableWidgetItem(str(round(Fricc[i][0],5)))

    self.ui.tbl_res_tub.setItem(i, 0, t)
    self.ui.tbl_res_tub.setItem(i, 1, ni)
    self.ui.tbl_res_tub.setItem(i, 2, nf)
    self.ui.tbl_res_tub.setItem(i, 3, q)
    self.ui.tbl_res_tub.setItem(i, 4, v)
    self.ui.tbl_res_tub.setItem(i, 5, hf)
    self.ui.tbl_res_tub.setItem(i, 6, f)

self.ui.tbl_res_tub.resizeColumnsToContents()
```



```
"""Vaciamos los resultados de los nodos a su tabla"""
row = len(Presion)
self.ui.tbl_res_nod.clearContents()
self.ui.tbl_res_nod.setRowCount(0)
for i in range(row):
    self.ui.tbl_res_nod.insertRow(i)

n =
QtWidgets.QTableWidgetItem(str(T_num2str[Vec_Nodo[i+1]]))
dem = QtWidgets.QTableWidgetItem(str(round(Vec_QH[i+1], 2)))
alt = QtWidgets.QTableWidgetItem(str(round(Altura[i][0],
2)))
press = QtWidgets.QTableWidgetItem(str(round(Presion[i][0],
2)))

self.ui.tbl_res_nod.setItem(i, 0, n)
self.ui.tbl_res_nod.setItem(i, 1, dem)
self.ui.tbl_res_nod.setItem(i, 2, alt)
self.ui.tbl_res_nod.setItem(i, 3, press)
self.ui.tbl_res_nod.resizeColumnsToContents()

try:
    self.Fn_GrafResultadosGradiente(Vec_Nodo, Presion,
self.PminD, self.PmaxD,
                                    self.ui.widget_pres.canvas,
T_num2str)

except:
    self.msgErrorBox("ERROR EN LA GRÁFICA DE PRESIÓN")

try:
    self.Fn_GrafResultadosGradiente(Vec_tuberia, Velocidad,
self.VminD, self.VmaxD,
                                    self.ui.widget_vel.canvas)
except:
    self.msgErrorBox("ERROR EN LA GRÁFICA DE VELOCIDAD")

try:
    self.Fn_GrafErrPorcentual(er)
except:
    self.msgErrorBox("ERROR EN LA GRÁFICA DE ERROR")

#####
##### GRÁFICA DE RESULTADOS DE VELOCIDAD Y PRESIÓN #####
#####

def Fn_GrafResultadosGradiente(self, NodTub, Vector, Vecmin, Vecmax,
widget, Trans = None):

    self.ploteo = widget
    self.ploteo.ax.clear()

    # Ajuste de los parámetros del plot
```



```
        self.plotero.fig.subplots_adjust(0.05, 0.1, 1, 1) #  
left,bottom,right,top  
  
        col = []  
        titulo = []  
        Val_Grafica = []  
        for i in range(len(Vector)):  
            Val_Grafica.append(Vector[i][0])  
            if Vector[i][0] <= Vecmin:  
                col.append('#CCECFF')  
            elif Vector[i][0] >= Vecmax:  
                col.append('#CCECFF')  
            else:  
                col.append('#1f77b4')  
  
            if Trans != None:  
                titulo.append(Trans[NodTub[i+1]])  
            elif Trans == None:  
                titulo.append(NodTub[i])  
  
            if Trans != None:  
                self.plotero.ax.axhline(y=Vecmin, linewidth=2, color='m',  
linestyle="--",  
                                label='Presión Minima de Diseño')  
                self.plotero.ax.axhline(y=Vecmax, linewidth=2, color='green',  
linestyle="--",  
                                label='Presión Maxima de Diseño')  
            elif Trans == None:  
                self.plotero.ax.axhline(y=Vecmin, linewidth=2, color='m',  
linestyle="--",  
                                label='Velocidad Minima de Diseño')  
                self.plotero.ax.axhline(y=Vecmax, linewidth=2, color='green',  
linestyle="--",  
                                label='Velocidad Maxima de Diseño')  
            self.plotero.ax.legend()  
            self.plotero.ax.bar(titulo, Val_Grafica, color=col)  
  
            # Color de los lados del Plot  
            self.plotero.ax.tick_params(colors='white', which='both') #  
'both' refers to minor and major axes  
  
            self.plotero.draw()  
  
        """  
        #####  
        # GRAFICA DE ERROR PORCENTUAL #  
        #####  
        """  
  
    def Fn_GrafErrPorcentual(self, V_err):  
        num = [i + 1 for i in range(len(V_err))]  
        self.plotero = self.ui.widget_err_porcentual.canvas  
        self.plotero.ax.clear()  
        # Shade the area between y1 and line y=0  
        self.plotero.ax.fill_between(num, V_err, 0,  
                                    color='blue', # The outline color
```



```
alpha=0.2) # Transparency of the
fill
    # Color de los lados del Plot
    self.ploteo.ax.tick_params(colors='white', which='both') # 
'both' refers to minor and major axes
    # Show the plot
    self.ploteo.draw()

"""
#####
# PAGINA: ALGORITMO GENÉTICO #
#####
"""

def Prop_AnadirRenglonSinCombo(self):
    val = self.ui.txt_num_diametros.value()
    self.ui.tbl_Dcom.setRowCount(val)
    self.ui.tbl_Dcom.update()

def Atr_AlgoritmoGeneticoSeguridad(self):
    # Datos de Poblacion
    tasa_mutacion = self.ui.txt_tasa_mut.value()
    num_generaciones = self.ui.txt_num_gen.value()
    tamano_poblacion = self.ui.txt_tam_pob.value()
    # Datos de Diseño
    self.PmaxD = self.ui.txt_PDmax.value()
    self.PminD = self.ui.txt_PDmin.value()
    self.VmaxD = self.ui.txt_VDmax.value()
    self.VminD = self.ui.txt_VDmin.value()
    val = False
    if tamano_poblacion == 0:
        self.msgErrorBox("TAMAÑO DE POBLACIÓN NO PUEDE SER CERO !! ")
        val = True
    elif num_generaciones == 0:
        self.msgErrorBox("NÚMERO DE GENERACIONES NO PUEDE SER CERO
!!")
        val = True
    elif tasa_mutacion == 0:
        self.msgErrorBox("TASA DE MUTACIÓN NO PUEDE SER CERO !! ")
        val = True
    elif self.PmaxD == 0:
        self.msgErrorBox("PRESIÓN MÁXIMA NO PUEDE SER CERO !! ")
        val = True
    elif self.PmaxD <= self.PminD:
        self.msgErrorBox("PRESIÓN MÁXIMA DEBE SER LA MAYOR !! ")
        val = True
    elif self.VmaxD == 0:
        self.msgErrorBox("VELOCIDAD MÁXIMA NO PUEDE SER CERO !! ")
        val = True
    elif self.VmaxD <= self.VminD:
        self.msgErrorBox("VELOCIDAD MÁXIMA DEBE SER LA MAYOR !! ")
        val = True
    elif self.PminD == 0:
        self.msgErrorBox("PRESIÓN MÍNIMA NO PUEDE SER CERO !! ")
        val = True
    elif self.PminD >= self.PmaxD:
```



```
        self.msgErrorBox("PRESIÓN MÍNIMA DEBE SER LA MENOR !!")
        val = True
    elif self.VminD == 0:
        self.msgErrorBox("VELOCIDAD MÍNIMA NO PUEDE SER CERO !!")
        val = True
    elif self.VminD >= self.VmaxD:
        self.msgErrorBox("VELOCIDAD MÍNIMA DEBE SER LA MENOR !!")
        val = True
    return val

def Atr_RadioButton_Regresiones(self):

    # Extraer Data de la Tabla de Precio-Diametro
    x = []
    y = []
    self.DiamCom = []
    num_ren_diam = self.ui.tbl_Dcom.rowCount()
    for i in range(num_ren_diam):
        diam = float(self.ui.tbl_Dcom.item(i, 0).text())
        precio = float(self.ui.tbl_Dcom.item(i, 1).text())
        x.append(diam)
        y.append(precio)
    self.DiamCom = x
    if self.ui.rb_lineal.isChecked() == True:
        #  $y = m*x + b$ 
        N = len(x)
        SXY = sum([x[i] * y[i] for i in range(N)])
        SX2 = sum([x[i]**2 for i in range(N)])
        SX = sum(x)
        SY = sum(y)
        Xp = sum(x) / N
        Yp = sum(y) / N
        m1 = SX * SY - N * SXY
        m2 = SX**2 - N * SX2
        m = round(m1 / m2, 5)
        b = round(Yp - m * Xp, 5)
        # Coeficiente de determinación R2
        Yr = [m * x[i] + b for i in range(N)]
        Y_Yr_2 = sum([(y[i] - Yr[i])**2 for i in range(N)])
        Y_Yp_2 = sum([(y[i] - Yp)**2 for i in range(N)])
        R2 = 1 - (Y_Yr_2 / Y_Yp_2)
        self.K1 = m
        self.K2 = b
        self.RegTipo = "Lineal"

        self.ui.lbl_R2.setText('<html><head/><body><p><span style="color:#55ffff; font-weight:600;">' +
                               'Coef. de Determinación:' +
                               {:.5f}'.format(R2) +
                               '</span></p></body></html>')

    elif self.ui.rb_potencial.isChecked() == True:
        #  $y = a*x^{**b}$ 
        N = len(x)
        X = [math.log10(i) for i in x]
        Y = [math.log10(i) for i in y]
```



```
SXY = sum([X[i] * Y[i] for i in range(N)])
SX2 = sum([X[i] ** 2 for i in range(N)])
SX = sum(X)
SY = sum(Y)
Xp = sum(X) / N
Yp = sum(Y) / N
b1 = N * SXY - SX * SY
b2 = N * SX2 - SX ** 2
b = round(b1 / b2, 5)
A = Yp - b * Xp
a = round(10 ** A, 5)
# Coeficiente de determinación R2
Yr = [a * x[i] ** b for i in range(N)]
y_Yr_2 = sum([(y[i] - Yr[i]) ** 2 for i in range(N)])
y_Yp_2 = sum([(y[i] - Yp) ** 2 for i in range(N)])
R2 = 1 - (y_Yr_2 / y_Yp_2)
self.K1 = a
self.K2 = b
self.RegTipo = "Potencial"
self.ui.lbl_R2.setText('<html><head/><body><p><span style="color:#55ffff; font-weight:600;">' +
'Coef. de Determinación:' +
'{:.5f}'.format(R2) +
'</span></p></body></html>')

elif self.ui.rb_exponencial.isChecked() == True:
    # y = b*10**mx
    N = len(x)
    Y = [math.log10(i) for i in y]
    SXY = sum([x[i] * Y[i] for i in range(N)])
    SX2 = sum([x[i] ** 2 for i in range(N)])
    SX = sum(x)
    SY = sum(Y)
    Xp = sum(x) / N
    Yp = sum(Y) / N
    m1 = SX * SY - N * SXY
    m2 = SX ** 2 - N * SX2
    m = round(m1 / m2, 5)
    B = Yp - m * Xp
    b = round(10 ** B, 5)
    # Coeficiente de determinación R2
    Yr = [b * math.pow(10, m * x[i]) for i in range(N)]
    y_Yr_2 = sum([(y[i] - Yr[i]) ** 2 for i in range(N)])
    y_Yp_2 = sum([(y[i] - Yp) ** 2 for i in range(N)])
    R2 = 1 - (y_Yr_2 / y_Yp_2)

    self.K1 = b
    self.K2 = m
    self.RegTipo = "Exponencial"
    self.ui.lbl_R2.setText('<html><head/><body><p><span style="color:#55ffff; font-weight:600;">' +
'Coef. de Determinación:' +
'{:.5f}'.format(R2) +
'</span></p></body></html>')

def Atr_AlgoritmoGenetico(self):
    # Datos de Poblacion
```



```
tasa_mutacion = self.ui.txt_tasa_mut.value()
num_generaciones = self.ui.txt_num_gen.value()
tamano_poblacion = self.ui.txt_tam_pob.value()
if tamano_poblacion%2 != 0:
    tamano_poblacion += 1
# Datos de Diseño
self.PmaxD = self.ui.txt_PDmax.value()
self.PminD = self.ui.txt_PDmin.value()
self.VmaxD = self.ui.txt_VDmax.value()
self.VminD = self.ui.txt_VDmin.value()
# Extraer Data de la Tabla de Precio-Diametro
x = []
self.DiamCom = []
num_ren_diam = self.ui.tbl_Dcom.rowCount()
for i in range(num_ren_diam):
    diam = float(self.ui.tbl_Dcom.item(i, 0).text())
    x.append(diam)
self.DiamCom = x
# Data de Tablas
[Vec_Nodx, Vec_Nody, Vec_Nodo, Vec_tipo, Vec_QH, Vec_cota,
T_str2num, T_num2str] = self.Fn_Nodos()
[Vec_tuberia, ini_fin, Vec_Long, Vec_diam, Vec_km, Vec_ks] =
self.Fn_Tuberias()
# Acomodo de la Informacion de las tablas
data_nod = []
for i in range(len(Vec_Nodo)):
    data_nod.append([Vec_Nodo[i], Vec_QH[i], Vec_tipo[i],
Vec_cota[i]])

data_tub = []
for i in range(len(Vec_tuberia)):
    data_tub.append([Vec_tuberia[i], T_str2num[ini_fin[i][0]],
T_str2num[ini_fin[i][1]],
Vec_Long[i], Vec_diam[i], round(Vec_km[i],
2), round(Vec_ks[i], 4)])
# Creacion de la Clase "Red Hidraulica"
rh = gh.RedHidraulica(data_nod, data_tub, self.PmaxD,
self.PminD, self.VmaxD, self.VminD, self.DiamCom)
# Creacion de la Clase "Algoritmo Genetico"
AG = ag.AlgoritmoGenetico(rh, self.K1, self.K2, self.RegTipo,
tamano_poblacion, tasa_mutacion, num_generaciones)
# Mejores diametros encontrados
[MI, hora, minuto, segundo] = AG.resolver()
self.ListaGene = []
self.ListaCostoC = []
for i in range(1, len(MI)):
    self.ListaCostoC.append(round(MI[i][3],2))
    self.ListaGene.append(MI[i][4])
# Velocidades (maximas y minima) y Presiones (maximas y minimas)
Sol_red = ag.Individuo(rh, self.K1, self.K2, self.RegTipo, MI[-1][0])
# Mostrar resultados al FrontEnd
best_diam = MI[-1][0]
row = len(Vec_tuberia)
self.ui.tbl_mejor_diam.clearContents()
self.ui.tbl_mejor_diam.setRowCount(0)
for i in range(row):
```



```
        self.ui.tbl_mejor_diam.insertRow(i)
        t2 = QtWidgets.QTableWidgetItem(str(Vec_tuberia[i]))
        d2 = QtWidgets.QTableWidgetItem(str(round(best_diam[i], 2)))
        self.ui.tbl_mejor_diam.setItem(i, 0, t2)
        self.ui.tbl_mejor_diam.setItem(i, 1, d2)
        self.ui.tbl_mejor_diam.resizeColumnsToContents()

        self.ui.lbl_mejor_gen.setText('<html><head/><body><p><span style=" color:#55ffff; font-weight:600;">' +
                                     'MEJOR GENERACIÓN:<br>' +
                                     '{:.0f}</span></p></body></html>')
        self.ui.lbl_CC.setText('<html><head/><body><p><span style=" color:#55ffff; font-weight:600;">' +
                               'COSTO DE CONSTRUCCIÓN: $' +
                               '{:0,.2f}</span></p></body></html>')
        self.ui.lbl_CH.setText('<html><head/><body><p><span style=" color:#55ffff; font-weight:600;">' +
                               'COSTO HIDRÁULICO: {:.4f}</span></p></body></html>')
        self.ui.lbl_time.setText('<html><head/><body><p><span style=" color:#55ffff; font-weight:600;">' +
                               'TIEMPO DE EJECUCIÓN: {} hrs : {} min : {} seg</span></p></body></html>')

        if MI[-1][1] > 0:
            self.ui.lbl_CH.setText('<html><head/><body><p><span style=" color:#ff0000; font-weight:600;">' +
                                   'COSTO HIDRÁULICO:<br>' +
                                   '{:.4f}</span></p></body></html>')
        else:
            self.ui.lbl_CH.setText('<html><head/><body><p><span style=" color:#55ffff; font-weight:600;">' +
                                   'COSTO HIDRÁULICO:<br>' +
                                   '{:.4f}</span></p></body></html>')

        if Sol_red.v_max >= self.VmaxD:
            self.ui.lbl_res_vmax.setText('<html><head/><body><p><span style=" color:#ff0000; font-weight:600;">' +
                                         'VELOCIDAD MÁXIMA: {:.2f}' +
                                         'metros/seg'.format(Sol_red.v_max) +
                                         '</span></p></body></html>')
        else:
            self.ui.lbl_res_vmax.setText('<html><head/><body><p><span style=" color:#55ffff; font-weight:600;">' +
                                         'VELOCIDAD MÁXIMA: {:.2f}' +
                                         'metros/seg'.format(Sol_red.v_max) +
                                         '</span></p></body></html>')

        if Sol_red.v_min <= self.VminD:
            self.ui.lbl_res_vmin.setText('<html><head/><body><p><span style=" color:#ff0000; font-weight:600;">' +
```



```
'VELOCIDAD MÍNIMA: {:.2f}
metros/seg'.format(Sol_red.v_min) +
'</span></p></body></html>')

else:
    self.ui.lbl_res_vmin.setText('<html><head/><body><p><span
style=" color:#55ffff; font-weight:600;">' +
'VELOCIDAD MÍNIMA: {:.2f}
metros/seg'.format(Sol_red.v_min) +
'</span></p></body></html>')

if Sol_red.p_max >= self.PmaxD:
    self.ui.lbl_res_pmax.setText('<html><head/><body><p><span
style=" color:#ff0000; font-weight:600;">' +
'PRESIÓN MÁXIMA: {:.2f}
metros'.format(Sol_red.p_max) +
'</span></p></body></html>')

else:
    self.ui.lbl_res_pmax.setText('<html><head/><body><p><span
style=" color:#55ffff; font-weight:600;">' +
'PRESIÓN MÁXIMA: {:.2f}
metros'.format(Sol_red.p_max) +
'</span></p></body></html>')

if Sol_red.p_min <= self.PminD:
    self.ui.lbl_res_pmin.setText('<html><head/><body><p><span
style=" color:#ff0000; font-weight:600;">' +
'PRESIÓN MÍNIMA: {:.2f}
metros'.format(Sol_red.p_min) +
'</span></p></body></html>')

else:
    self.ui.lbl_res_pmin.setText('<html><head/><body><p><span
style=" color:#55ffff; font-weight:600;">' +
'PRESIÓN MÍNIMA: {:.2f}
metros'.format(Sol_red.p_min) +
'</span></p></body></html>')

# Graficamos las generaciones:
self.Fn_GeneracionCosto()
# Resultados temporales:
self.Gen_tempo = MI[-1][4]
self.Cc_tempo = MI[-1][3]
self.Ch_tempo = MI[-1][1]
self.Diam_tempo = MI[-1][0]

"""

#####
# GRAFICA DE GENERACION VS COSTO #
#####

def Fn_GeneracionCosto(self):
    self.plot = self.ui.widget_gen_vs_costo.canvas
    self.plot.ax.clear()
    # Ajuste de los parametros del plot
    self.plot.fig.subplots_adjust(0.1, 0.1, 1, 1) #
left,bottom,right,top
```



```
# Shade the area between y1 and line y=0
self.ploteo.ax.fill_between(self.ListaGene, self.ListaCostoC, 0,
                            color='m',    # The outline color
                            alpha=0.2)   # Transparency of the
fill
# Color de los lados del Plot
self.ploteo.ax.tick_params(colors='white', which='both')  #
'both' refers to minor and major axes
# Show the plot
self.ploteo.draw()

"""
#####
# RESULTADOS TEMPORALES #
#####
"""

def Fn_ResultadosTemporales(self):
    # Agregar los resultados en la tabla
    num_ren = self.ui.tbl_diam_temp.rowCount()
    if num_ren == 0:
        self.ui.tbl_diam_temp.setRowCount(1)
        tb_g = QtWidgets.QTableWidgetItem(str(self.Gen_tempo))
        tb_cc =
QtWidgets.QTableWidgetItem(str(round(self.Cc_tempo,2)))
        tb_ch =
QtWidgets.QTableWidgetItem(str(round(self.Ch_tempo,2)))
        tb_d = QtWidgets.QTableWidgetItem(str(self.Diam_tempo))
        self.ui.tbl_diam_temp.setItem(0, 0, tb_g)
        self.ui.tbl_diam_temp.setItem(0, 1, tb_cc)
        self.ui.tbl_diam_temp.setItem(0, 2, tb_ch)
        self.ui.tbl_diam_temp.setItem(0, 3, tb_d)
        self.ui.tbl_diam_temp.resizeColumnsToContents()
    else:
        self.ui.tbl_diam_temp.setRowCount(num_ren+1)
        tb_g = QtWidgets.QTableWidgetItem(str(self.Gen_tempo))
        tb_cc =
QtWidgets.QTableWidgetItem(str(round(self.Cc_tempo,2)))
        tb_ch =
QtWidgets.QTableWidgetItem(str(round(self.Ch_tempo,2)))
        tb_d = QtWidgets.QTableWidgetItem(str(self.Diam_tempo))
        self.ui.tbl_diam_temp.setItem(num_ren, 0, tb_g)
        self.ui.tbl_diam_temp.setItem(num_ren, 1, tb_cc)
        self.ui.tbl_diam_temp.setItem(num_ren, 2, tb_ch)
        self.ui.tbl_diam_temp.setItem(num_ren, 3, tb_d)
        self.ui.tbl_diam_temp.resizeColumnsToContents()

    # Agregar los numeros en el combobox
    for i in range(num_ren+1):
        index = self.ui.cb_diam_cargar.findText(str(i+1))
        self.ui.cb_diam_cargar.removeItem(index)
        self.ui.cb_diam_cargar.addItem(str(i+1))

def Fn_ExportarDiam(self):
    # Titulos de costos y generaciones
    L_1 = ["generacion", "Costo Const", "Costo Hidra"]
    # Titulos de los diametros
    ren0 = self.ui.tbl_mejor_diam.rowCount()
```



```
L_2 = []
for i in range(ren0):
    # Nomenclatura de los diametros
    NOM = self.ui.tbl_mejor_diam.item(i, 0).text()
    L_2.append(NOM)
# Union de los Titulos
L_1.extend(L_2)

ren = self.ui.tbl_diam_temp.rowCount()
Lista_varios = []
Lista_varios.append(L_1)
for i in range(ren):
    # Valores de costos
    GEN = int(self.ui.tbl_diam_temp.item(i, 0).text())
    CC = float(self.ui.tbl_diam_temp.item(i, 1).text())
    CH = float(self.ui.tbl_diam_temp.item(i, 2).text())
    # Primera lista
    List_gchd = [int(GEN), float(CC), float(CH)]
    # Valores de diametros
    DIAM = self.ui.tbl_diam_temp.item(i, 3).text()
    # Segunda lista
    List_str_D = list(DIAM[1:-1].split(","))
    col = len(List_str_D)
    List_D = [float(List_str_D[k]) for k in range(col)]
    # Combinacion de listas 1 y 2
    List_gchd.extend(List_D)
    # Lista de Listas
    Lista_varios.append(List_gchd)
# Transformacion de listas a Zip
Listas_zip = zip(*Lista_varios)
# Guardar en CSV
nombre = self.ui.txt_nom_export.text()
try:
    # Creamos un nuevo directorio
    newDir("C://RedGenHid")
    # Guardamos el archivo
    path = 'C://RedGenHid//'
    pathFile = path + nombre + '.csv'
    with open(pathFile, 'w') as f:
        write = csv.writer(f)
        for val in Listas_zip:
            write.writerow(val)
except OSError:
    # Guardamos el archivo
    path = 'C://RedGenHid//'
    pathFile = path + nombre + '.csv'
    with open(pathFile, 'w') as f:
        write = csv.writer(f)
        for val in Listas_zip:
            write.writerow(val)

def Fn_CargarListaDiametros(self):
    # Obtener el diametro deseado
    val = int(self.ui.cb_diam_cargar.currentText())-1
    # Valores de diametros en cadena de caracteres
    DIAM = self.ui.tbl_diam_temp.item(val, 3).text()
    # Valores de diametros en formato de lista tipo float
```



```
List_str_D = list(DIAM[1:-1].split(","))
col = len(List_str_D)
List_D = [float(List_str_D[k]) for k in range(col)]
# Vaciamos los datos a la tabla y columna correspondiente
row = len(List_D)
for i in range(0, row):
    d_temp = QtWidgets.QTableWidgetItem(str(List_D[i]))
    self.ui.tbl_tub.setItem(i, 4, d_temp) # Diametro

if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    myapp = UIMainWindow()
    myapp.show()
    sys.exit(app.exec_())
```