UNIVERSIDAD DE GUANAJUATO
CAMPUS GUANAJUATO
DIVISIÓN DE INGENIERÍAS

ANÁLISIS HIDRODINÁMICO DEL FLUJO UNIFORME A SUPERFICIE LIBRE

TESIS

PARA OBTENER EL TÍTULO DE:

INGENIERO CIVIL

PRESENTA:

MARTÍN ESTRADA VACA

DIRECTOR: DR. ELADIO DELGADILLO RUIZ

CO-DIRECTOR: DRA. LUZ ADRIANA ARIAS HERNÁNDEZ

octubre de 2021

Dedicatorias

El presente trabajo va dedicado primeramente a mis Padres Ma. Dolores y Antonio, quienes me dieron el apoyo y la confianza en cada paso dado a lo largo de mi vida.

A mi hermano Antonio, que ha sido un confidente y apoyo

A mis abuelitas Sofía e Isabel, quienes me criaron y cuidaron durante muchos años

Agradecimientos

El presente trabajo es la culminación de una etapa llena de horas de trabajo, pero también de buenos momentos, risas y excelentes personas a mi alrededor.

- A Dios, que nunca me ha dejado solo.
- A mi familia, mis padres, mi hermano y mis abuelitas que siempre fueron el pilar fundamental para buscar superarme día con día.
- A mi alma mater, la Universidad de Guanajuato, que me brindó conocimientos para poderme desarrollar como un profesionista responsable y que busca siempre el bien común de la sociedad.
- A mi director y codirector de tesis, el Dr. Eladio y la Dr. Luz Adriana, quienes siempre me apoyaron el desarrollo de este trabajo e incluso me apoyaban en temas ajenos a la tesis.
- A mis sinodales, Dr. David Tirado, Dra. Guadalupe Vázquez, Dr. Saúl Villalobos por sus comentarios con respecto a este trabajo y el cómo puedo mejorarlo.
- A mi primo Emmanuel que siempre fue un guía y lo considero como un hermano.
- A Miriam y Mariana por estar siempre apoyándome tanto en el ámbito académico como en el personal.
- A Sara, por traer siempre la buena vibra, por mostrarme que hasta en lo malo siempre debe existir algo bueno y que hay que verle el lado positivo a cada una de las situaciones.
- A Gloria, por darme mis jalones de orejas cuando me lo merecía, pero también por extenderme su mano cuando lo necesitaba.
- A amigos como Rogelio, Gabriela, Aarón y Rolando, quienes han sido excelentes amigos y compañeros desde el principio de mi carrera.
- A David, Salvador y André, quienes me hacían reír y disfrutar en los descansos entre clase y clase.
- A algunos de mis compañeros de casa, entre ellos Timoteo, Edgar, Omar, Diego, Edwin y Rafael, quienes me dieron ese calor de hogar
- Y agradecido con todos aquellos que de alguna manera me ayudaron en alguna parte de mi trayecto académico, muchas gracias.

Resumen

El flujo uniforme es un concepto básico de los estudios hidráulicos que se realizan desde la perspectiva de un ingeniero civil, su entendimiento es la base para el análisis de otros tipos de flujos que se presentan en la naturaleza y que no tienen esas características de idealidad como ocurre con este elemento. Es por eso, que en este trabajo se abordaron los conceptos de flujo uniforme, la obtención de las características geométricas de secciones comunes de canales y del estado crítico del flujo mediante el desarrollo de un programa computacional "HYDROGECA" de código libre que permite el análisis de estos componentes del flujo a superficie libre, obteniendo como resultados la variación de energía, las características de velocidad y la determinación de la cantidad de movimiento. Los resultados obtenidos representan una ayuda para los estudiantes del programa educativo de ingeniería civil que cursan los contenidos temáticos de la UDA de hidrodinámica, ya que les permite realizar un número mayor de análisis de casos de estudio referentes al diseño y la revisión de infraestructura hidráulica que presenta algún tipo de flujo a superficie libre. Con este producto se pretende coadyuvar en el desarrollo integral de los estudiantes y personal académico de la carrera de ingeniería civil, mediante la utilización de programas computacionales que beneficien en el proceso de enseñanza-aprendizaje.

Palabras clave: flujo uniforme, estado crítico, geometría, programa computacional.

Tabla de contenido

Índice de figuras

Índice de tablas

1 Hidrodinámica y Geometría en Canales (HYDROGECA)

Dentro de la problemática que se ha identificado del plan curricular del programa educativo de ingeniería civil, el análisis de las características del flujo a superficie libre es necesario para el desarrollo cognitivo integral del estudiante, la determinación de las características geométricas, de energía y de movimiento, forman parte de los componentes necesarios para abordar el contenido temático de la UDA de hidrodinámica y sobre todo del análisis de la uniformidad y del estado crítico del flujo. En la actualidad, debido al uso e implementación constante de herramientas computacionales, los procesos ingenieriles se benefician con el desarrollo de programas y softwares enfocados a solucionar los problemas más comunes a los que se enfrenta el estudiante de ingeniería civil (Ladino et al, 2021), entre ellos la adquisición de programas especializados para la resolución de problemas hidráulicos.

La División de Ingenierías del Campus Guanajuato, actualmente no cuenta con un programa computacional desarrollado para facilitar los diseños de canales o tuberías que trabajan en flujo uniforme y a superficie libre, el área de hidráulica de la carrera de ingeniería civil, hace uso de programas o rutinas de cómputo desarrollados en otras universidades como la UNAM, IPN u otros, pero no cuenta con una herramienta propia que pueda ser distribuida de manera gratuita y constante a los alumnos y que forme parte de materiales didácticos propios.

Por otro lado, el análisis del flujo uniforme determina el entendimiento de las ecuaciones fundamentales de la hidráulica, en las cuales se requiere de la obtención de las características geométricas, de energía y de movimiento en diferentes secciones de tipo natural o construidas por la mano del hombre. El cálculo manual de estas características toma un tiempo considerable, lo cual implica que el alumno de la carrera de ingeniería civil pierda tiempo valioso que pueda ser aplicable en otras unidades del curso de hidrodinámica, por lo cual, si se incluyera la aplicación de un programa computacional dentro del contenido temático del curso, se podrían agilizar los tiempos y los contenidos.

Justificación

Los estudiantes del programa educativo de ingeniería civil y el resto de la comunidad académica de la División de Ingenierías del Campus Guanajuato de la Universidad de Guanajuato requieren de herramientas computacionales que les permitan realizar sus actividades académicas con mayor facilidad, además de que sean confiables y sobre todo de fácil acceso. Con la propuesta del desarrollo de esta tesis se pretende suministrar de un programa de código libre a los interesados en el desarrollo de las características geométricas, el flujo uniforme y del régimen crítico del flujo a superficie libre, lo cual garantice el desarrollo del aprendizaje y que propicie el futuro desarrollo de programas asociados a la ingeniería civil y coadyuven al desarrollo de la educación, el proceso enseñanza-aprendizaje y en la práctica profesional.

Objetivo general

Desarrollar un programa computacional y un manual de usuario que determine las características del flujo uniforme a superficie libre, determine las características geométricas y establezca el estado crítico del flujo, además, que sea gratuito y que se encuentre a disposición los alumnos y personal académico del programa educativo de ingeniería civil.

Objetivos específicos

- Incluir un análisis de energía específica, determinación de características geométricas, máxima eficiencia hidráulica, método de convección sección-pendiente, factor de sección, ecuación de continuidad y de régimen crítico del flujo en canales abiertos.
- Desarrollar con un programa computacional gratuito que forme parte del material de apoyo de la UDA de hidrodinámica.

1.1 Entendimiento general del programa

En este capítulo partimos dando a conocer de manera muy general en qué entorno ha sido desarrollado el programa, así como sus componentes.

HYDROGECA (***HYDRO***dinámica y ***GE***ometría de ***CA***nales) fue desarrollado en Visual Studio, el cual es un entorno de desarrollo integrado (IDE). Trabajar en este desarrollador es sencillo ya que permite ir creando una interfaz a la par de la escritura del código, que para este caso se ha usado el lenguaje C#.

Los módulos que integran esta herramienta computacional son 3 los cuales son llamados:

- Propiedades geométricas
- Tirante crítico
- Flujo normal

Algo a destacar es que los módulos "tirante crítico" y "flujo normal" contienen dentro de ellos ciertas divisiones que las llamaremos submódulos y que cada uno realiza algún cálculo referente al módulo que se elige. En el módulo tirante crítico tenemos los submódulos:

- Cálculo de tirante crítico
- Transición de flujo

Las divisiones que presenta el módulo de flujo normal son:

- Factor de sección
- Sección de máxima eficiencia
- Tirante conocido
- Sección – pendiente

Para tener un mejor entendimiento de los componentes que integran a HYDROGECA, se hace uso de un organigrama donde se plasma el desarrollo del programa.

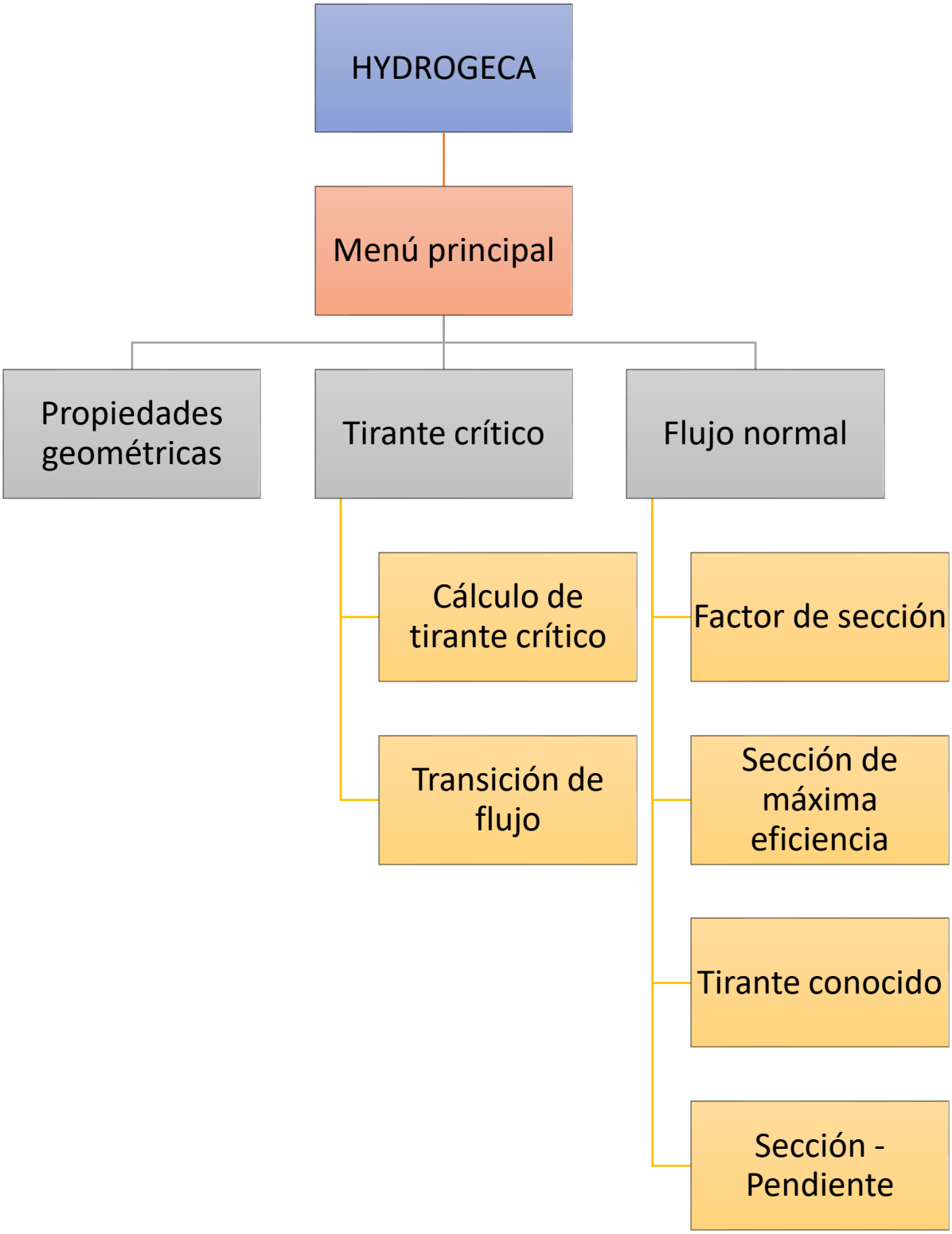


Figura 1.1 Organigrama general del programa HYDROGECA

Fuente: Elaboración propia

La figura 1.1 es simple y con poca especificación tratándose de todo un programa, pero hay que recordar (Chapra & Canale, 2006) que para este capítulo sólo abarcamos de manera general y resumida HYDROGECA. A lo largo de todo el trabajo se genera un capítulo por módulo y en cada uno se han de explicar sus divisiones de una manera más específica.

1.2 Uso de métodos numéricos

El tener acceso a las computadoras generó un cambio en la cuestión del uso y desarrollo en el ámbito de los métodos numéricos (Chapra & Canale, 2006).

Cuando se tiene una variable en cierta función (Costabile & Macchione, 2012) y se quiere conocer su valor, lo primero que imaginamos es el despejar y obtener un valor de manera exacta. Sin embargo, hay cientos de casos donde las ecuaciones no dan la opción de generar una solución analítica. Para estas situaciones se han desarrollado los métodos numéricos. Recordemos que existen los métodos abiertos y los cerrados.

En HYDROGECA se ha empleado uno de cada tipo. El método de bisección, el cual pertenece a los cerrados, nos es bastante útil en secciones circulares ya que, si se ve en cuestión del tirante, el valor solución estará entre 0 y el diámetro. En el caso de los abiertos se optó por el más común, método de Newton – Raphson (Ladino et al, 2020), usado especialmente en los módulos de flujo normal y tirante crítico.

En las figuras 1.2 y 1.3 se hace una representación general de los métodos de bisección y Newton – Raphson respectivamente. Aunque en ambas figuras se representa un método, estos pueden tener leves diferencias al momento de programarse, pero se sigue el mismo principio que se ilustra.

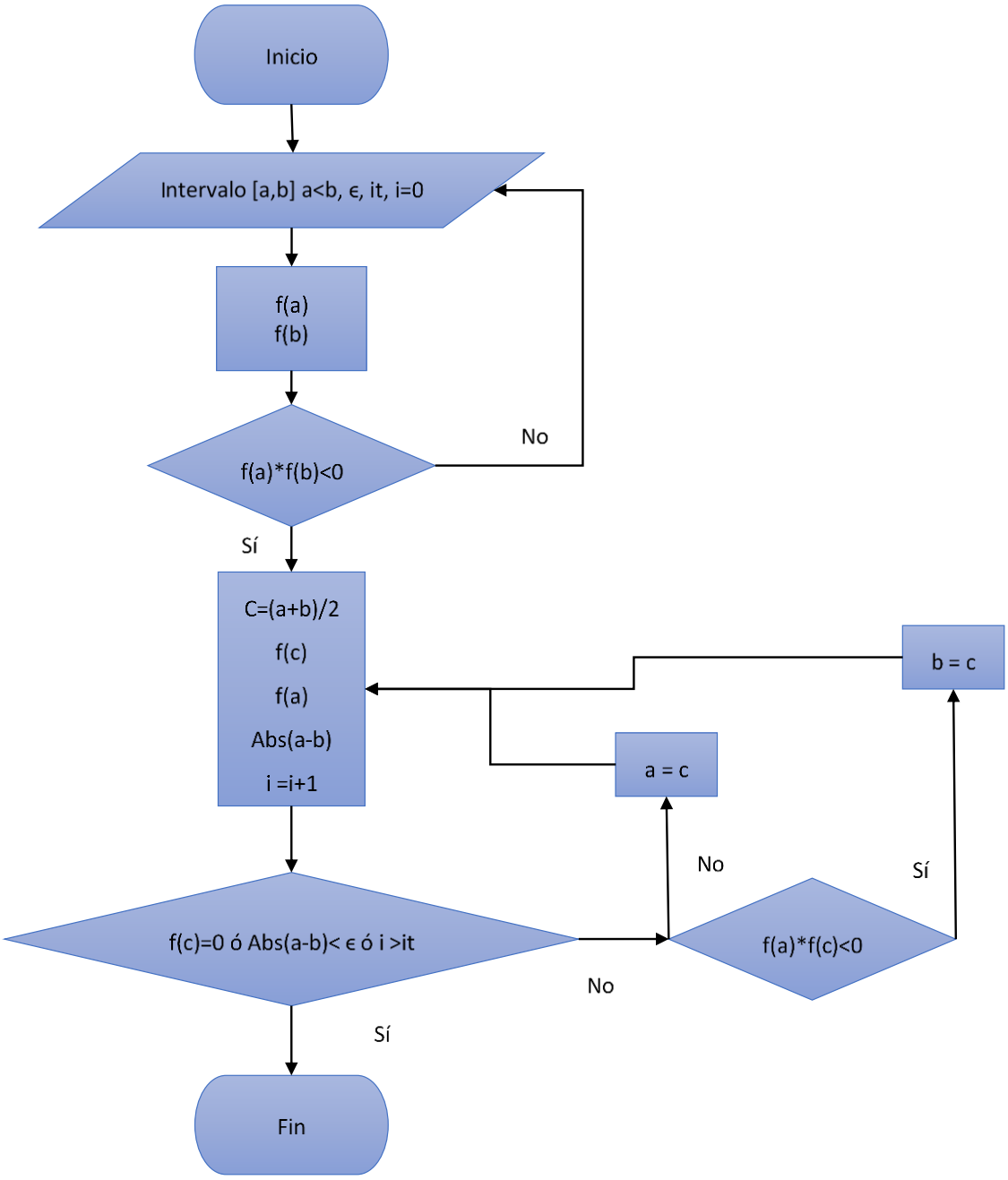


Figura 1.2 Representación del método de bisección a través de diagrama de flujo

Fuente: Elaboración propia

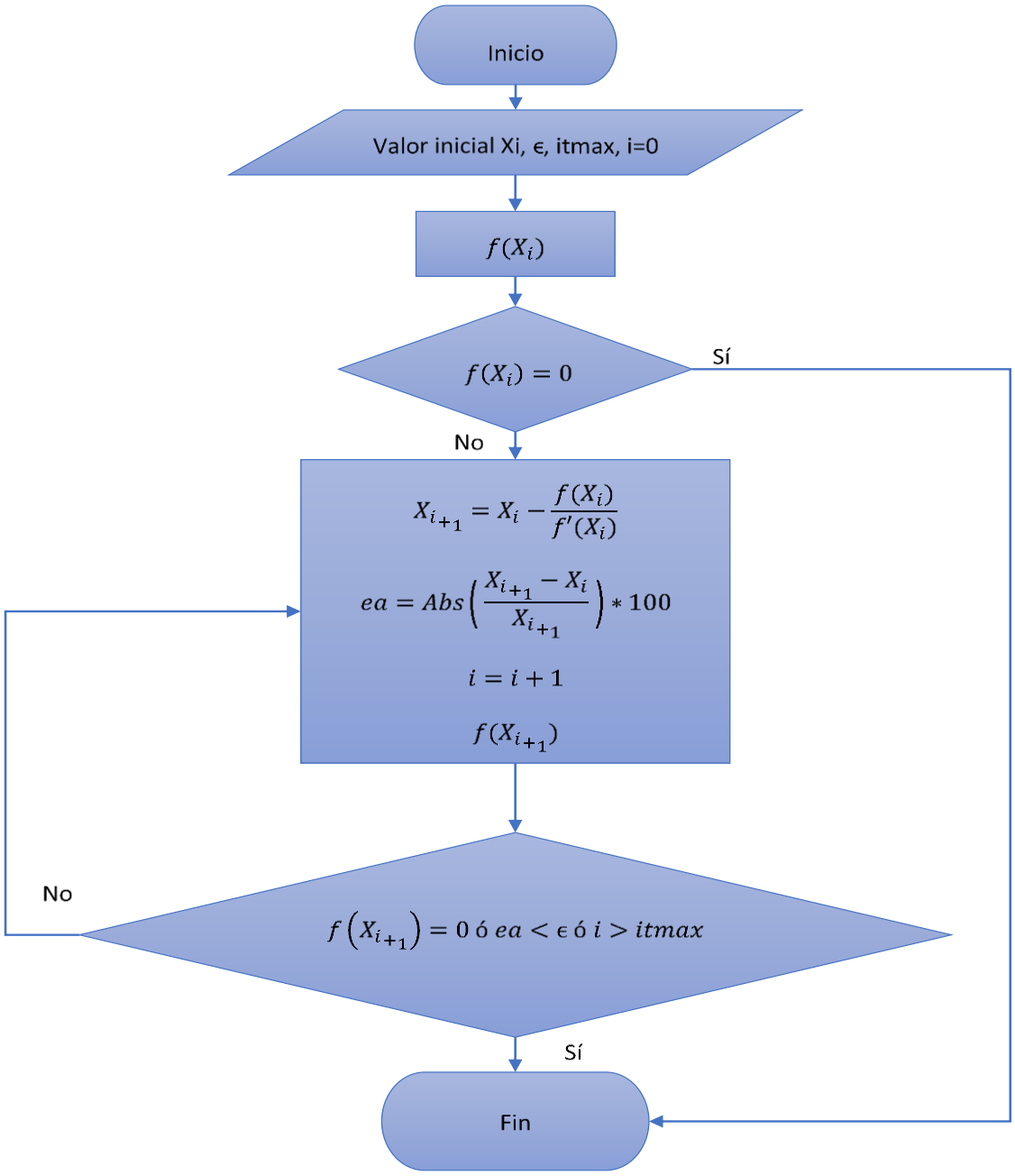


Figura 1.3 Representación del método Newton - Raphson a través de diagrama de flujo

Fuente: Elaboración propia

2 Capítulo Propiedades geométricas

2.1 Términos relacionados a la geometría en canales

Según Ven Te Chow (1994) las definiciones más comunes de los elementos geométricos de una sección son:

Elementos geométricos: Serán entendidos como propiedades de una sección de canal, estos mismos están definidos por la geometría de la sección y por la profundidad de flujo existente.

Profundidad del flujo o tirante hidráulico (y): Es aquella distancia vertical comprendida desde el punto más bajo comprendido en la sección de un canal hasta la denominada superficie libre. Es común manejar esta variable con la letra "y".

Área hidráulica o mojada (A): Hemos de llamar así al área de la sección transversal del flujo que es perpendicular a la dirección en que se presente el flujo.

Perímetro mojado (P): Longitud de la línea de intersección de la superficie de canal mojada y de un plano transversal perpendicular a la dirección del flujo.

Radio hidráulico (R_h): Es el resultado de dividir el área mojada entre su perímetro mojado.

Ancho superficial (T): Es el ancho que se presenta en la sección del canal a la altura de la superficie libre.

Profundidad hidráulica (D): Es el resultado de dividir el área mojada entre el ancho superficial.

Factor de sección (Z): Lo comprenderemos como la división del área mojada elevada a la 1.5 entre la raíz de ancho superficial de la sección del canal.

Talud (z): Rige la inclinación presente en las paredes de la sección (trapecial y triangular) y corresponde a la distancia horizontal que se recorre por cada unidad ascendida en vertical. Para el presente trabajo manejamos z1 y z2 ya que hemos de considerar que los taludes pueden ser simétricos, pero también distintos.

Base (b): Ancho presente en el fondo de los canales rectangular y trapecial.

2.2 Secciones transversales de canales

El uso del término sección de canal en el presente trabajo hace referencia a una sección transversal de canal que es perpendicular al sentido del flujo. También cabe resaltar que sólo se involucra con

secciones de canales artificiales, los cuales podemos entender que son aquellos que fueron diseñados y creados por el hombre.

Se puede llegar a pensar que el siguiente estudio es demasiado simplista debido a que se manejan solamente cinco secciones que son: rectángulo, trapecio, triángulo, círculo y parábola. Además, que estas figuras son demasiado conservadoras. Sin embargo, dentro de la ingeniería siempre se ha de buscar dar una solución lo más simplificada posible pero que, a la vez, sea funcional y la geometría tomada en cuenta sea sencilla pero capaz de cumplir con el objetivo de transportar algún gasto.

En la tabla 2.1 se muestran las secciones de interés y su respectiva fórmula para calcular sus elementos geométricos. Al observar la tabla es notorio que las fórmulas están principalmente en función del tirante hidráulico, base, taludes y diámetro. Para poder simplificar la cantidad de fórmulas y no complicarse con algunas que son demasiado largas como radio hidráulico, profundidad hidráulica y factor de sección, es cuestión de regresar a las definiciones de estos elementos geométricos.

El radio hidráulico de cualquier sección lo podremos calcular de la siguiente manera:

$$R_h = \frac{A}{P} \qquad (2\text{--}1)$$

Donde:

- Rh = Radio hidráulico
- A = Área hidráulica o mojada
- P = Perímetro mojado

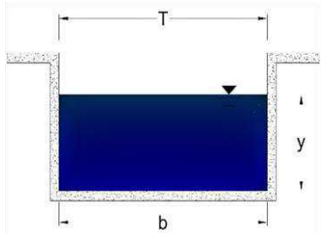
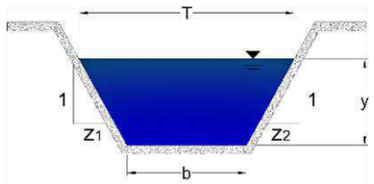
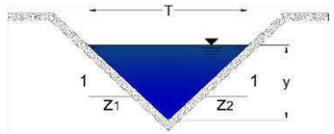
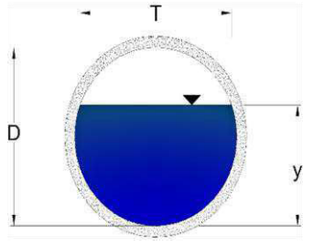
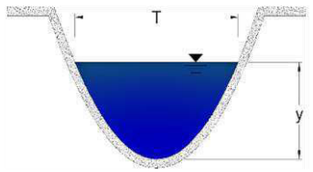
Rectángulo	Trapecio	Triángulo	Círculo	Parábola	
					
$b * y$	$[b + 0.5 * y * (z_1 + z_2)] * y$	$[0.5 * (z_1 + z_2)] * y^2$	$\frac{1}{8} * [\theta - Seno(\theta)] * D^2$	$\frac{2}{3} * T * y$	Área A
$b + 2 * y$	$b + y * \left(\sqrt{1 + z_1^2} + \sqrt{1 + z_2^2}\right)$	$y * \left(\sqrt{1 + z_1^2} + \sqrt{1 + z_2^2}\right)$	$\frac{1}{2} * \theta * D$	$T + \frac{8}{3} * \frac{y^2}{T}$	Perímetro P
$\frac{b * y}{b + 2 * y}$	$\frac{[b + 0.5 * y * (z_1 + z_2)] * y}{b + y * \left(\sqrt{1 + z_1^2} + \sqrt{1 + z_2^2}\right)}$	$\frac{[0.5 * (z_1 + z_2)] * y^2}{y * \left(\sqrt{1 + z_1^2} + \sqrt{1 + z_2^2}\right)}$	$\frac{\frac{1}{8} * [\theta - Seno(\theta)] * D^2}{\frac{1}{2} * \theta * D}$	$\frac{\frac{2}{3} * T * y}{T + \frac{8}{3} * \frac{y^2}{T}}$	Radio hidráulico Rh
b	$b + (z_1 + z_2) * y$	$(z_1 + z_2) * y$	$\frac{Seno\left(\frac{1}{2} * \theta\right) * D}{2\sqrt{y * (D - y)}}$	$\frac{3}{2} * \frac{A}{y}$	Ancho Sup. T
y	$\frac{[b + 0.5 * y * (z_1 + z_2)] * y}{b + (z_1 + z_2) * y}$	$\frac{[0.5 * (z_1 + z_2)] * y^2}{(z_1 + z_2) * y}$	$\frac{\frac{1}{8} * [\theta - Seno(\theta)] * D^2}{Seno\left(\frac{1}{2} * \theta\right) * D}$	$\frac{\frac{2}{3} * T * y}{\frac{3}{2} * \frac{A}{y}}$	Prof. Hidráulica D
$\frac{(b * y)^{1.5}}{\sqrt{b}}$	$\frac{([b + 0.5 * y * (z_1 + z_2)] * y)^{1.5}}{\sqrt{b + (z_1 + z_2) * y}}$	$\frac{([0.5 * (z_1 + z_2)] * y^2)^{1.5}}{\sqrt{(z_1 + z_2) * y}}$	$\frac{\left(\frac{1}{8} * [\theta - Seno(\theta)] * D^2\right)^{1.5}}{\sqrt{Seno\left(\frac{1}{2} * \theta\right) * D}}$	$\frac{\left(\frac{2}{3} * T * y\right)^{1.5}}{\sqrt{\frac{3}{2} * \frac{A}{y}}}$	Factor sección Z

Tabla 2.1 Fórmulas de elementos geométricos

Fuente: Elaboración propia

Para la profundidad hidráulica tenemos opción de usar la fórmula:

$$D = \frac{A}{T} \qquad (2\text{--}2)$$

Donde:

- D = Profundidad hidráulica
- A = Área hidráulica o mojada
- T = Ancho superficial

Recordemos que también es aplicable para todas las secciones al igual que el factor de sección, el cual se describe como:

$$Z = \frac{A^{1.5}}{\sqrt{T}} \qquad (2\text{--}3)$$

Donde

- Z = Factor de sección
- A = Área hidráulica o mojada
- T = Ancho superficial

Es necesario entrar más a detalle en las fórmulas de la sección circular y parabólica, esto porque tienen ciertas particularidades. Todas las fórmulas para una sección circular siempre están en función de la letra griega theta (θ), este símbolo representa un ángulo de abertura.

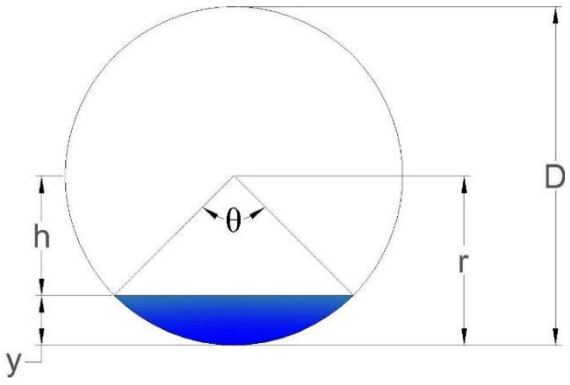


Figura 2.1 Sección circular del ángulo de apertura

Fuente: Elaboración propia

Con la imagen anterior, podemos deducir el valor de theta (θ). Lo primero será dividir el ángulo entre dos, teniendo dos triángulos rectángulos.

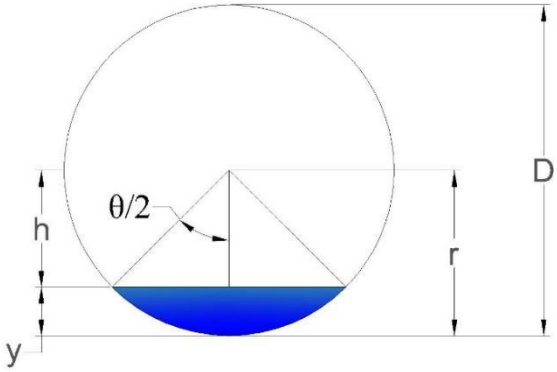


Figura 2.2 Sección circular del ángulo dividido

Fuente: Elaboración propia

El valor de h es la diferencia entre radio y tirante (ecuación 2-4) y a su vez lo nombraremos cateto adyacente.

$$h = r - y \qquad (2\text{--}4)$$

La hipotenusa para este caso será igual al radio y teniendo conocidas estas dos medidas, las asociamos con la función trigonométrica Coseno (Cos), como se muestra en la siguiente ecuación:

$$Cos\left(\frac{\theta}{2}\right) = \frac{\text{cateto adyacente}}{\text{hipotenusa}} = \frac{r-y}{r} = 1 - \frac{y}{r} \qquad (2\text{--}5)$$

despejando nuestro ángulo obtenemos la siguiente ecuación:

$$\theta = 2\text{arcCos}\left(1 - \frac{\text{y}}{\text{r}}\right) = 2\text{arcCos}\left(1 - \frac{\text{y}}{0.5 * \text{D}}\right) \qquad (2\text{--}6)$$

finalmente vemos que nuestro ángulo está en función de tirante hidráulico y diámetro, pudiendo tener valores desde 0 hasta 2π.

Continuando con la parábola, la precaución que se debe tener es, calculando el perímetro mojado, ya que la fórmula mostrada en la tabla de elementos geométricos no siempre es aplicable. Por ello, primero debemos calcular el valor "x", el cual se calcula con la fórmula:

$$x = \frac{4 * y}{T} \qquad (2\text{--}7)$$

Donde:

- x = valor a comparar
- y = tirante hidráulico
- T = ancho superficial

Las opciones conociendo el resultado de la ecuación 2-7 serán las siguientes:

$$P = T + \frac{8}{3} * \frac{y^2}{T} \qquad si\ 0 < x \leq 1 \qquad (2\text{--}8)$$

$$P = \left[\frac{T}{2}\right] * \left[\sqrt{1 + x^2} + \frac{1}{x} * \ln\left(x + \sqrt{1 + x^2}\right)\right] \qquad si\ x > 1 \qquad (2\text{--}9)$$

La primera fórmula para obtener el perímetro mojado (P) de la parábola es una aproximación satisfactoria desarrollada con base en la segunda.

Otro factor tomado en cuenta en el programa, pero que no es visto en la tabla es la constante parabólica definida como:

$$k = \frac{4 * y}{T^2} \qquad (2\text{--}10)$$

Este valor nos dicta la forma que tendrá la sección parabólica.

2.3 Combinaciones

Anteriormente se han descrito las secciones que trabaja el módulo de propiedades geométricas, sus elementos geométricos y las fórmulas con los que se calculan. Ahora en este espacio se muestran las posibles combinaciones para cada sección.

Para entender las combinaciones haremos uso de la sección más simple que es la rectangular. La combinación más común sería conocer base (b) y tirante hidráulico (y), a partir de ingresar estos dos datos se calculan los diferentes elementos geométricos. Otra combinación será usar el mismo valor de base, pero en esta ocasión el otro dato ingresado será el área hidráulica (A). Con lo anterior, el programa es capaz de obtener los valores geométricos incluyendo el tirante hidráulico. En seguida se anexan las tablas de combinatorias.

2.3.1 Sección rectangular

Rectángulo							
Datos ingresados		Datos calculados					
Base b	Tirante hidráulico y	A	P	R_h	T	D	Z
	Área A	y	P	R_h	T	D	Z
	Perímetro mojado P	A	y	R_h	T	D	Z
	Radio hidráulico Rh	A	P	y	T	D	Z
	Profundidad hidrálica D	A	P	R_h	T	y	Z
Tirante y	Área A	b	P	R_h	T	D	Z
	Perímetro mojado P	A	b	R_h	T	D	Z
	Radio hidráulico Rh	A	P	b	T	D	Z
	Ancho superficial T	A	P	R_h	b	D	Z

Tabla 2.2 Combinaciones de la sección rectangular

Fuente: Elaboración propia

En esta primera tabla se observa que siempre es necesario ingresar 2 datos, de los cuales el que vemos en vertical se mantiene constante y el horizontal es el que puede cambiar.

2.3.2 Sección trapecial

Trapecio								
Datos ingresados			Datos calculados					
Tirante y	Base b	Talud z1 y z2	A	P	R_h	T	D	Z
		Área A	z1 yz2	P	R_h	T	D	Z
		Perímetro P	z1 yz2	A	R_h	T	D	Z
		Radio hidráulico Rh	z1 yz2	A	P	T	D	Z
		Ancho superficial T	z1 yz2	A	P	R_h	D	Z
Taludes	Base b	Área A	y	P	R_h	T	D	Z
		Perímetro P	y	A	R_h	T	D	Z
		Radio hidráulico Rh	y	A	P	T	D	Z
		Ancho superficial T	y	A	P	R_h	D	Z
Taludes	Tirante y	Área A	b	P	R_h	T	D	Z
		Perímetro P	b	A	R_h	T	D	Z
		Radio hidráulico Rh	b	A	P	T	D	Z
		Ancho superficial T	b	A	P	R_h	D	Z

Tabla 2.3 Combinaciones de la sección trapecial

Fuente: Elaboración propia

La sección trapecial nos pide ingresar 3 datos. Los elementos geométricos en vertical son los que se quedan constantes y en el tercer elemento pueden ser variables.

Algo importante es que, cuando se conocen los taludes, estos pueden tener la misma relación o pueden ser diferentes el uno del otro. En caso de que sean desconocidos, siempre se considerarán de la misma relación.

2.3.3 Sección triangular

Triángulo							
Datos ingresados		Datos calculados					
Taludes	Tirante hidráulico y	A	P	R_h	T	D	Z
	Área A	y	P	R_h	T	D	Z
	Perímetro mojado P	y	A	R_h	T	D	Z
	Radio hidráulico Rh	y	A	P	T	D	Z
	Ancho superficial T	y	A	P	R_h	D	Z
Tirante y	Área A	z1 y z2	P	R_h	T	D	Z
	Perímetro mojado P	z1 y z2	A	R_h	T	D	Z
	Radio hidráulico Rh	z1 y z2	A	P	T	D	Z
	Ancho superficial T	z1 y z2	A	P	R_h	D	Z

Tabla 2.4 Combinaciones de la sección triangular

Fuente: Elaboración propia

Una sección triangular al igual que una rectangular requiere de 2 valores iniciales para poder llevar a cabo el cálculo de los elementos geométricos restantes.

Esta sección es similar a la trapecial en cuanto a los taludes. La relación de taludes puede ser distinta una de la otra cuando la conocemos, pero, si es un elemento desconocido y que se calculará, ambos taludes mantienen la misma relación.

2.3.4 Sección circular

Círculo							
Datos ingresados		Datos calculados					
Tirante y	Diámetro D	A	P	R_h	T	D	Z
	Área A	D	A	R_h	T	D	Z
	Perímetro mojado P	D	P	R_h	T	D	Z
	Radio hidráulico Rh	D	A	P	T	D	Z
	Ancho superficial T	D	A	P	R_h	D	Z
Diámetro	Área A	y	P	R_h	T	D	Z
	Perímetro mojado P	y	A	R_h	T	D	Z
	Radio hidráulico Rh	y	A	P	T	D	Z
	Ancho superficial T	y	A	P	R_h	D	Z

Tabla 2.5 Combinaciones de la sección circular

Fuente: Elaboración propia

La sección circular nos pide ingresar 2 datos conocidos y para este caso, podremos dejar constante tanto el tirante hidráulico o el diámetro y combinarlo con otro elemento geométrico.

2.3.5 Sección parabólica

Parábola							
Datos ingresados		Datos calculados					
Tirante y	Ancho superficial T	A	P	R_h	K	D	Z
	Constante K	A	P	R_h	T	D	Z
	Área A	K	P	R_h	T	D	Z
Cte K	Ancho superficial T	A	P	R_h	K	D	Z
	Área A	A	P	R_h	T	D	Z
Anch Sup T	Área A	K	P	R_h	T	D	Z

Tabla 2.6 Combinaciones de la sección parabólica

Fuente: Elaboración propia

La última sección de interés pedirá al programa conocer 2 datos para ingresarlos y que se puedan realizar los cálculos correspondientes para obtener los valores de los elementos geométricos restantes.

2.4 Ejemplo ilustrativo

A lo largo del capítulo se ha dado explicación con respecto al módulo de propiedades geométricas, las fórmulas empleadas y las combinaciones que podemos generar. Sin embargo, todo esto lo podemos ver solamente como teoría y puede que el lector aún tenga un par de dudas de HYDROGECA en este módulo, es por ello que se ha de resolver un ejercicio de manera manual y luego se hace uso del programa.

Problema 1. - Un canal de sección rectangular con base b = 3.00 m tiene un perímetro mojado de P = 9.50 m. Calcular los elementos geométricos desconocidos.

De nuestra Tabla 2.1 conocemos las fórmulas y la primera a emplear es la fórmula de perímetro mojado, $P = b + 2 * y$, de ella sólo desconocemos el tirante hidráulico. Al despejar y sustituir valores nos queda:

$$y = \frac{P - b}{2} = \frac{9.50\ m - 3.00\ m}{2} = 3.25\ m$$

Debido a que ya conocemos el valor de nuestro tirante hidráulico y que toda la geometría del canal rectangular está en función del anterior calculado y la base, es posible obtener los valores restantes:

Área mojada o hidráulica

$$A = b * y = 3.00\ m * 3.25\ m = 9.75\ m^2$$

Radio hidráulico

$$R_h = \frac{A}{P} = \frac{9.75\ m^2}{9.50\ m} \approx 1.0263\ m$$

Ancho superficial

$$T = b = 3.00\ m$$

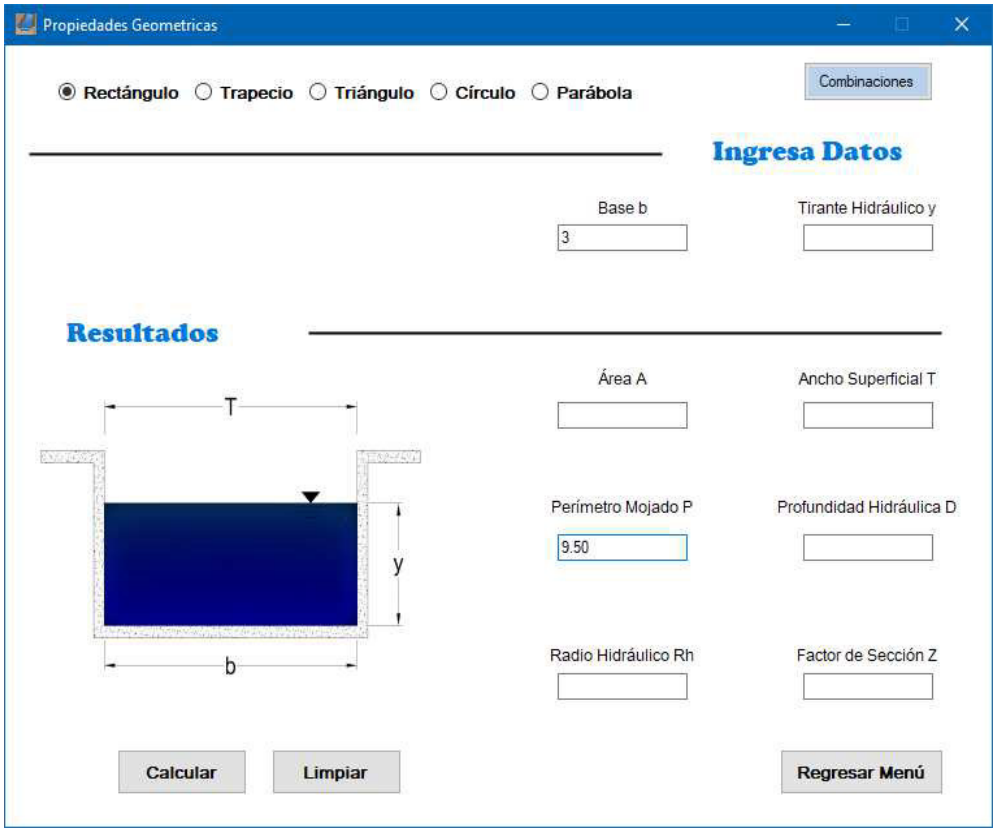
Profundidad hidráulica

$$D = \frac{A}{T} = \frac{9.75\ m^2}{3.00\ m} = 3.25\ m$$

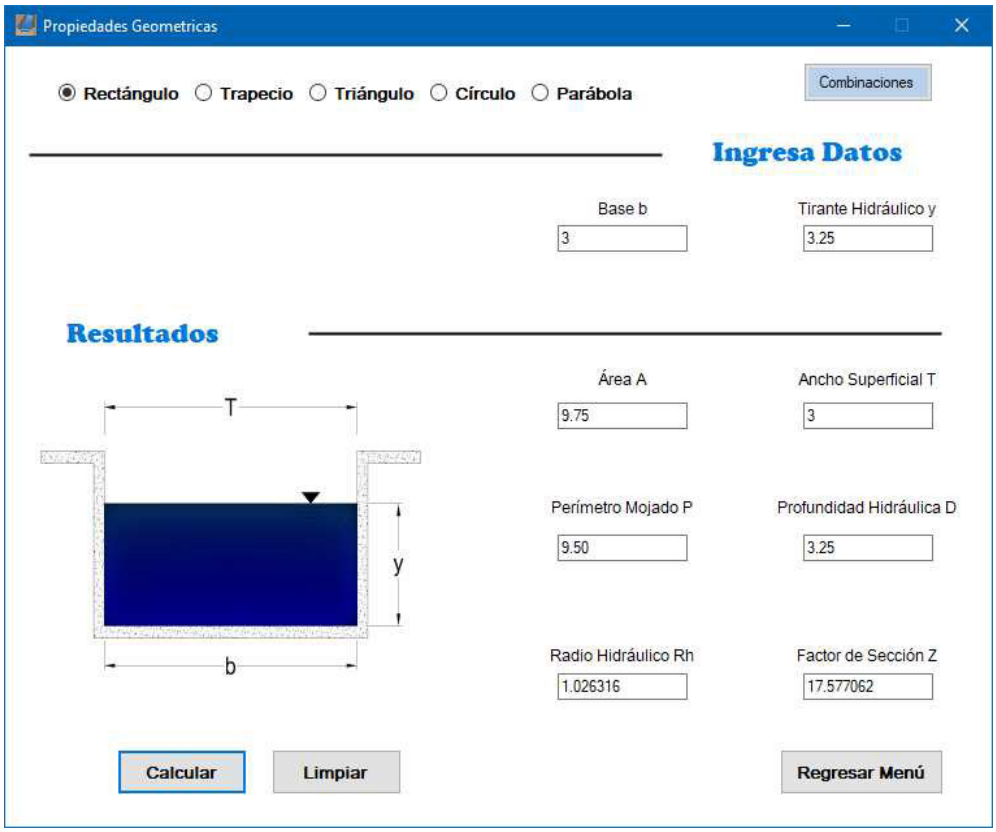
Factor de sección

$$Z = \frac{A^{1.5}}{\sqrt{T}} = \frac{9.75\ m^2}{\sqrt{3.00\ m}} \approx 17.5771\ [adimensional]$$

Lo siguiente es realizar el problema con uso de HYDROGECA.



a)



b)

Figura 2.3 Solución de problema 1 con HYDROGECA

Fuente: Programa HYDROGECA

En la figura 2.3 a) se muestra el ingreso de los datos conocidos (base y perímetro mojado), mientras que el siguiente inciso (figura 2.3 b)) contiene los resultados de la geometría calculada.

Para concluir este ejercicio, hacemos una comparativa de los valores obtenidos de la forma manual contra los obtenidos con HYDROGECA.

Elemento geométrico	Manualmente	HYDROGECA
Base b	3.00 m	3
Tirante hidráulico y	3.25 m	3.25
Área A	9.75 m^2	9.75
Perímetro mojado P	9.50 m	9.5
Radio hidráulico R_h	1.0263 m	1.026316
Ancho superficial T	3.00 m	3
Profundidad hidráulica D	3.25 m	3.25
Factor de sección Z	17.5771	17.577062

Tabla 2.7 Comparativa de resultados del problema 1 manual vs HYDROGECA

Fuente: Elaboración propia

Con esta comparación se comprueba que el programa cumple de manera satisfactoria los cálculos en el ámbito de las propiedades geométricas y las pequeñas diferencias se hacen notar en la cuestión de decimales y las unidades de medida. Mientras que de manera manual se propone una aproximación a 4 decimales, HYDROGECA se ha programado para que nos dé un redondeo hasta los 6 decimales.

Con respecto a las unidades de medida, el usuario debe ser coherente con ellas y no mezclar valores que sean de distinto sistema o unidad, por ejemplo, ingresar la base en metros y el perímetro en pulgadas o centímetros es incorrecto. Porque, a pesar de que es posible realizar el cálculo, este nos da resultados erróneos.

2.5 Solución de problemas

HYDROGECA en su módulo de propiedades geométricas se programó de manera tal que, al entrar en él, se elija en primera instancia el tipo de sección y en seguida se ingresen dos datos conocidos o inclusive tres para algunas secciones (trapecial). Al presionar el botón calcular se hará una comprobación de si la combinación de datos ingresados existe. Si no existe tal combinación el

programa no realiza ningún cálculo y se tendrán que ingresar valores de otros elementos geométricos (se recomienda primero revisar las tablas de combinaciones presentes en la sección “2.3 combinaciones”). En caso de que haya una coincidencia de esa combinación, se comienzan a realizar los cálculos de manera interna y solamente imprime en pantalla los resultados.

También se presenta un botón llamado limpiar, el cual se emplea para eliminar los datos obtenidos de un cálculo anterior y posterior a ello generar uno nuevo.

La mayoría de las combinaciones se resuelven de manera analítica consiguiendo resultados exactos, a excepción de algunos casos en la sección circular. Para esos casos se ha de emplear el método de bisección, mostrado en sección “1.2 Uso de métodos numéricos”.

Todo lo mencionado en este subcapítulo se puede resumir y mostrar de manera visual en la figura 2.4, la cual es una representación de cómo opera el módulo de propiedades geométricas.

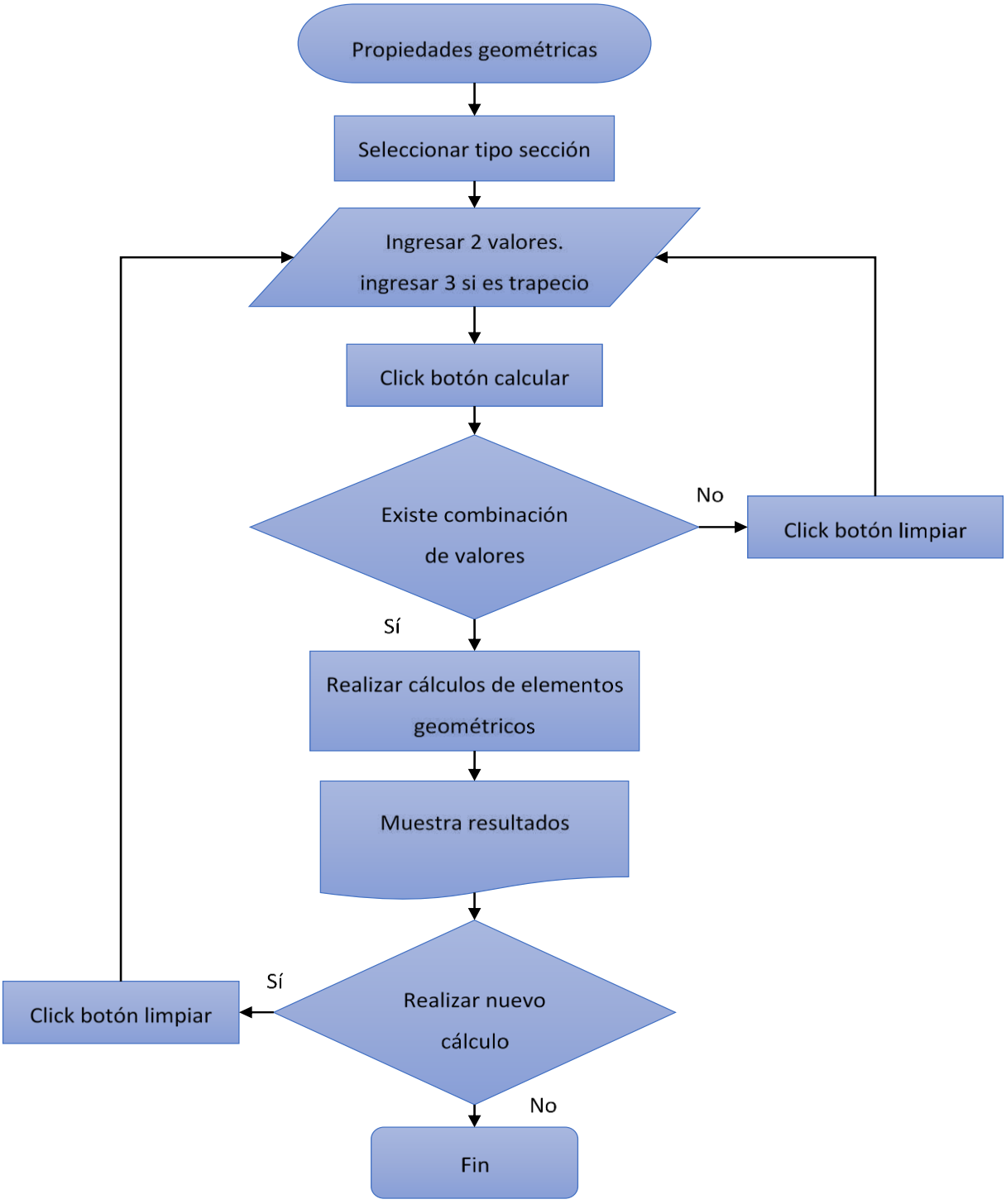


Figura 2.4 Representación del módulo propiedades geométricas a través de diagrama de flujo

Fuente: Elaboración propia

3 Capítulo tirante crítico

El módulo tirante crítico es dividido en dos submódulos: cálculo de tirante crítico y transición de flujo, los cuales se explican en este capítulo. Sin embargo, primero es necesario ver algunos conceptos y deducciones que son la base para la programación del módulo en cuestión.

3.1 Energía en canales

Este subcapítulo empieza haciendo una pequeña definición de la energía presente en canales abiertos, que según Chow (1994); Blalock (1980), la energía total del agua se ha de entender como una altura total, que es correspondiente a sumar 3 elementos: primeramente, una elevación que se encuentra sobre un nivel de referencia establecido y los otros dos son alturas, las cuales son generadas por presión debida al peso propio del flujo y la debida a la carga de velocidad.

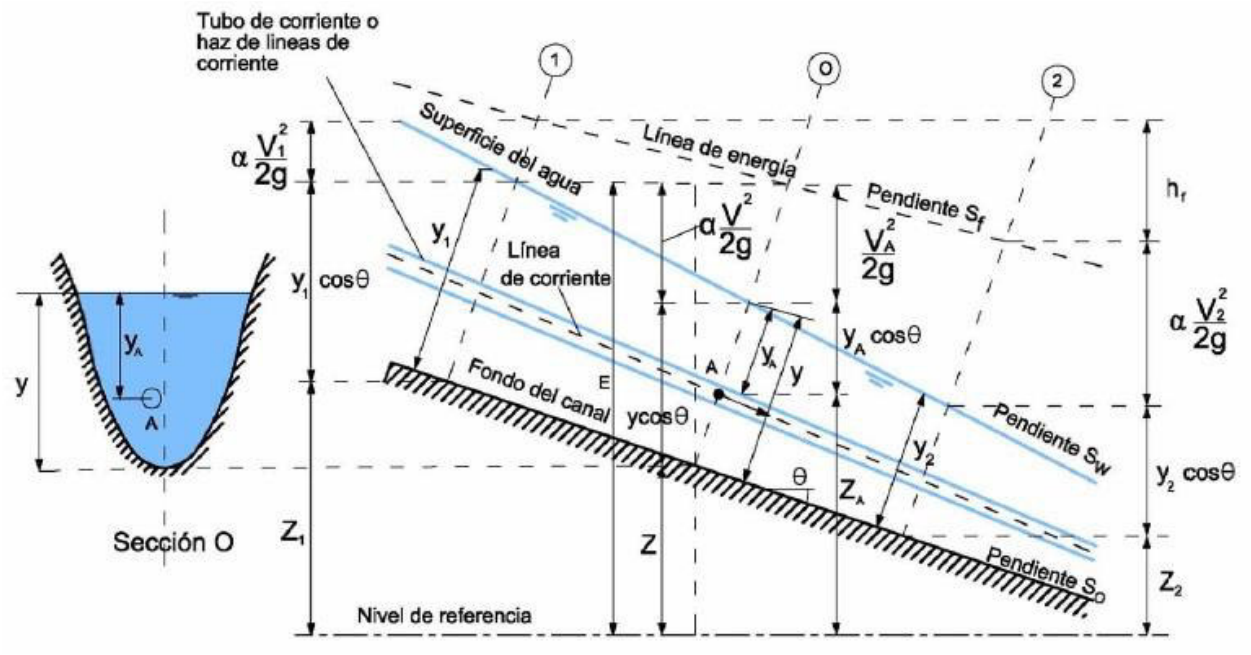


Figura 3.1 Energía de un flujo gradualmente variado en canales abiertos

Fuente: Tomada de (Chow, 1994)

La figura 3.1 nos ayuda a entender mejor lo descrito anteriormente. La sección *O* presente en un canal con pendiente de fondo theta (θ) posee un punto *A* que a su vez se encuentra en una línea de corriente. La intención es conocer la energía *E* en la sección. Guiados de la imagen, la altura o energía se da con la ecuación 3-1:

$$E = Z_A + y_A * Cos(\theta) + \frac{V_A^2}{2 * g} \qquad (3\text{--}1)$$

Donde

- Z_A = Elevación de *A* con respecto al nivel de referencia
- y_A = Distancia entre la posición del punto *A* y la superficie del agua
- θ = Ángulo de pendiente presente en el fondo del canal
- $V_A^2/2g$ = Altura de velocidad del flujo en la línea de corriente que pasa a través de *A*

Hay que hacer mención que, en una sección de canal están contenidas diferentes líneas de corriente y por efecto de una distribución no uniforme de velocidades, cada altura de velocidad presenta una magnitud distinta. Así que, como es muy común en la ingeniería, siempre idealizamos un modelo práctico y para el flujo gradualmente variado se simplifica considerado alturas de velocidad constantes para cualquier punto en la sección, pero también se añade un coeficiente para corrección (Chow, 1994).

$$E = Z + y * Cos(\theta) + \alpha * \frac{V^2}{2 * g} \qquad (3\text{--}2)$$

3.2 Energía específica

Contemplando lo ya dicho, Sotelo (2002) nos complementa diciendo que "la energía específica en una sección de un canal es la que corresponde al flujo por unidad de peso líquido a través de ella, y se mide con respecto al fondo de ésta" (p.165), o es la carga total del flujo respecto al fondo del canal (Lee *et al*., 2019).

Dando un valor de Z = 0 para la ecuación 3-2, esta se trascribe como:

$$E = yCos(\theta) + \alpha * \frac{V^2}{2 * g} \qquad (3\text{--}3)$$

En la práctica profesional es muy común el uso de un valor α = 1 y las pendientes en los canales suelen ser tan cercanas a cero que su valor se puede despreciar, lo que resulta en una simplificación de la ecuación anterior (Ecuación 3-4).

$$E = y + \frac{V^2}{2 * g} \qquad (3\text{--}4)$$

Ver que las ecuaciones anteriores involucran velocidad nos lleva a recordar que ésta se pude expresar como:

$$V = \frac{Q}{A} \qquad (3\text{--}5)$$

Donde:

- Q = Gasto que conduce la sección
- A = Área hidráulica de la sección

Sustituyendo 3-5 en 3-3 y 3-4 se obtienen las siguientes ecuaciones:

$$E = y * Cos(\theta) + \alpha * \frac{Q^2}{2 * g * A^2} \qquad (3\text{--}6)$$

$$E = y + \frac{Q^2}{2 * g * A^2} \qquad (3\text{--}7)$$

Estas dos últimas ecuaciones, serán útiles en especial, cuando se entre en la parte de tirante crítico.

3.2.1 Curva energía específica

Existen funciones sencillas como el caso de y = x, de la cual sabemos que “x” es un variable independiente y “y” es dependiente de “x”. Para la ecuación de energía específica (3-6), E es una variable que depende del tirante hidráulico y (Castro-Orgaz & Chanson, 2016).

Esta distinción de variables se hace debido a que, las funciones o ecuaciones se pueden graficar término independiente contra dependiente y si en un plano cartesiano colocamos en el eje de las ordenadas un valor de tirante hidráulico y en las abscisas el respectivo valor de la energía específica, habremos comenzado a crear lo que conocemos como curva de energía específica (Aiyesimoju, 2010).

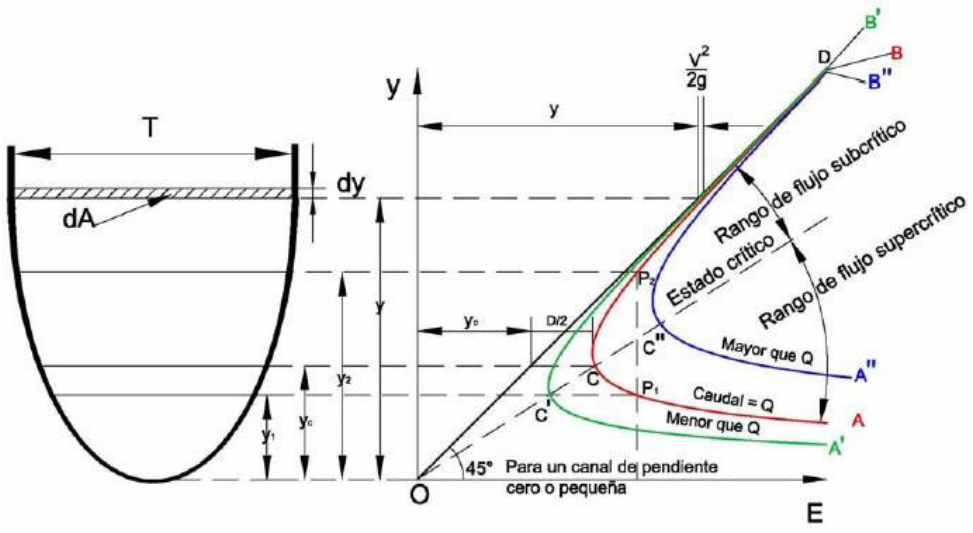


Figura 3.2 Curva de la energía específica

Fuente: Tomada de (Chow, 1994)

El plasmar datos en una gráfica permitirá visualizar de mejor manera el comportamiento de la función y con ello ver ciertas propiedades físicas de importancia como el tirante crítico y los tirantes alternos.

3.3 Cálculo de tirante crítico

Se ha de mencionar que el tirante crítico es la posición de la curva energía específica – tirante que corresponde al punto de quiebre de nuestra curva generada debido a cierto gasto y bajo las mismas condiciones geométricas (Guerrero, 2003). Muestra de esto es la curva ACB de la figura 3.2, donde el punto C es el cambio de dirección de la curva.

El tirante crítico tendrá un solo valor de energía específica, la cual corresponde al valor mínimo necesario para que exista movimiento de flujo en el canal (Lee *et al*., 2002). Para obtener las coordenadas exactas de este punto se requiere más que la gráfica, por ello tenemos que deducir una ecuación capaz de darnos el tirante hidráulico en el punto de quiebre.

Partimos derivando con respecto al tirante hidráulico la ecuación 3-6 perteneciente a la energía específica (ecuaciones 3-8 a 3-10).

$$\frac{dE}{dy} = Cos(\theta) + \alpha \frac{(0)(2gA^2) - (Q^2)(4gA)}{(2gA^2)^2} \frac{dA}{dy} \qquad (3\text{--}8)$$

$$\frac{dE}{dy} = Cos(\theta) + \alpha \frac{-(Q^2)(4gA)}{4g^2A^4} \frac{dA}{dy} \quad (3\text{--}9)$$

$$\frac{dE}{dy} = Cos(\theta) - \alpha \frac{Q^2}{gA^3} \frac{dA}{dy} \quad (3\text{--}10)$$

De la figura 3.2 podemos saber que:

$$dA = T\, dy \quad (3\text{--}11)$$

Despejando el ancho superficial:

$$T = \frac{dA}{dy} \quad (3\text{--}12)$$

Sustituyendo 3-12 en 3-10 se obtiene:

$$\frac{dE}{dy} = Cos(\theta) - \alpha \frac{Q^2}{gA^3} T \quad (3\text{--}13)$$

Una consideración que se debe tomar en cuenta al momento de calcular el tirante crítico de las secciones de estudio es que la diferencial de la energía especifica es igual a cero, por esta razón tenemos:

$$0 = Cos(\theta) - \alpha \frac{Q^2}{gA^3} T \quad (3\text{--}14)$$

$$Cos(\theta) = \alpha \frac{Q^2}{gA^3} T \quad (3\text{--}15)$$

Es necesario colocar de un lado de la igualdad toda variable que esté en función del tirante:

$$\frac{A^3}{T} = \alpha \frac{Q^2}{gCos(\theta)} \quad (3\text{--}16)$$

Para el caso donde tengamos un coeficiente Alpha (α) igual a la unidad y una pendiente muy cercana a cero, la ecuación 3-16 se simplifica a la ecuación 3-17.

$$\frac{A^3}{T} = \frac{Q^2}{g} \qquad (3\text{--}17)$$

Todas las secciones deben de cumplir las ecuaciones 3-16 o 3-17 según sea el caso, pero cada sección se ha de diferenciar por la fórmula de área hidráulica y ancho superficial. Por ejemplo, obtengamos la ecuación del tirante crítico para un canal rectangular. El área del rectángulo está dada por:

$$A = b\, y_c \qquad (3\text{--}18)$$

Hay una importante diferencia entre la fórmula 3-18 y la fórmula que se encuentra la tabla 2.1, ésta es que, para la primera siempre nuestro tirante hidráulico será el crítico y la segunda considerará cualquier tirante.

En una sección rectangular el ancho superficial se considera igual a la base, por ende, la ecuación 3-16 la podemos reescribir como:

$$\frac{(b\, y_c)^3}{b} = \alpha \frac{Q^2}{g\, Cos(\theta)} \qquad (3\text{--}19)$$

Y podemos saber el valor del tirante de una sección rectangular con la ecuación 3-20.

$$y_c = \sqrt[3]{\alpha \frac{Q^2}{b^2 g\, Cos(\theta)}} \qquad (3\text{--}20)$$

Es importante decir que no todas las secciones tendrán una resolución analítica al momento del cálculo de su tirante crítico. La más obvia que debe ser calculada mediante un método numérico es la circular debido a la gran complejidad para despejar el tirante.

$$\frac{\left(\frac{1}{8}\left(2arcCos\left(1-\frac{y_c}{0.5D}\right) - Seno\left(2arcCos\left(1-\frac{y_c}{0.5D}\right)\right)\right)\right)^3}{Seno\left(0.5\left(2arcCos\left(1-\frac{y_c}{0.5D}\right)\right)\right)} = \frac{Q^2}{g\, Cos(\theta)} \qquad (3\text{--}21)$$

El método de bisección (figura 1.2) es el ideal para poder converger a un valor de tirante crítico para el círculo.

A continuación, se dan las ecuaciones para las secciones que al igual que la rectangular tienen una solución analítica

Sección triangular:

$$y_c = \sqrt[5]{\alpha \frac{Q^2}{(0.5)^3 g\ Cos(\theta)(z1 + z2)^2}} \qquad (3\text{--}22)$$

Sección parabólica:

$$y_c = \sqrt[3]{\alpha \frac{27\ Q^2}{8\ g\ Cos(\theta)\ T^2}} \qquad (3\text{--}23)$$

La sección trapecial comparte con la circular la complejidad para despejar el tirante crítico, es por ello que recurrimos a el método de Newton – Raphson, buscando convergencia a la solución adecuada.

3.3.1 Ejemplo ilustrativo submódulo cálculo de tirante crítico

Habiendo deducido la ecuación para hallar el valor exacto del tirante crítico, se presentan dos ejercicios acordes al submódulo, uno con solución analítica y otro empleando método numérico.

Problema 2. – Un canal de sección triangular transporta un gasto Q = 3.5 m³/s y tiene un talud de 0.95, ¿Cuánto vale su tirante crítico y la energía específica si se ha de considerar un coeficiente Alpha α = 1 y una pendiente tan pequeña que se podrá considerar como cero?

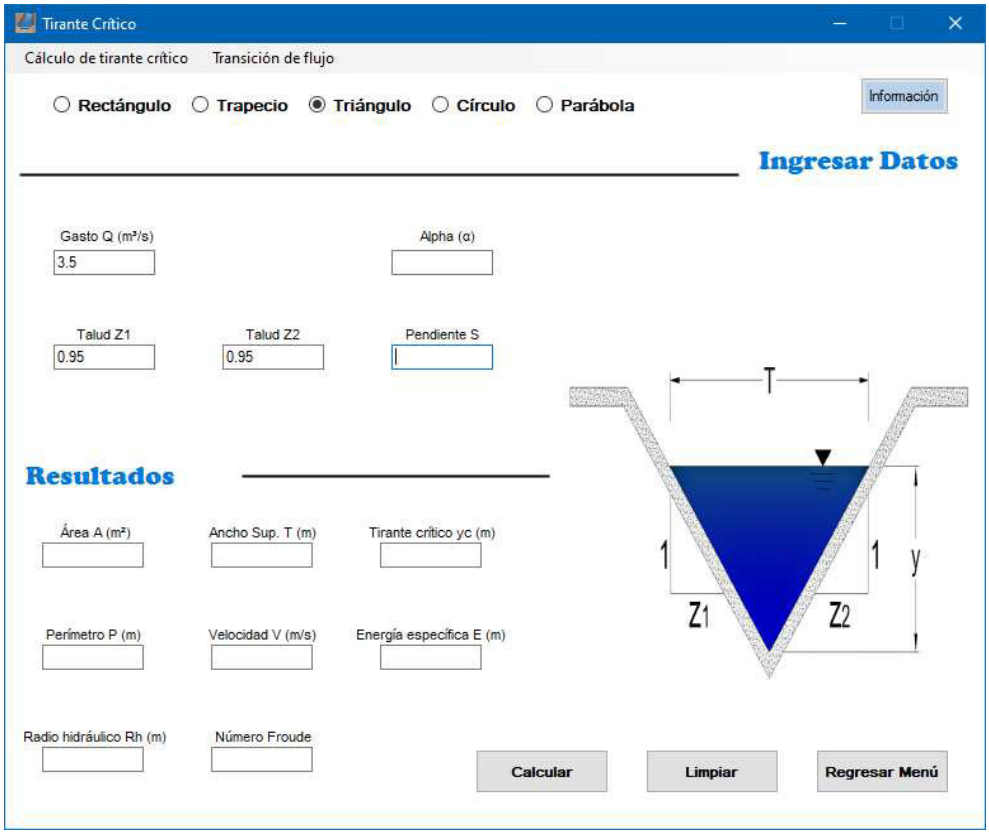
Para la resolución manual sustituimos los valores en la ecuación 3-22.

$$y_c = \sqrt[5]{\frac{\left(3.5 \frac{m^3}{s}\right)^2}{(0.5)^3 \left(9.81 \frac{m}{s^2}\right)(0.95 + 0.95)^2}} = 1.2258\ m$$

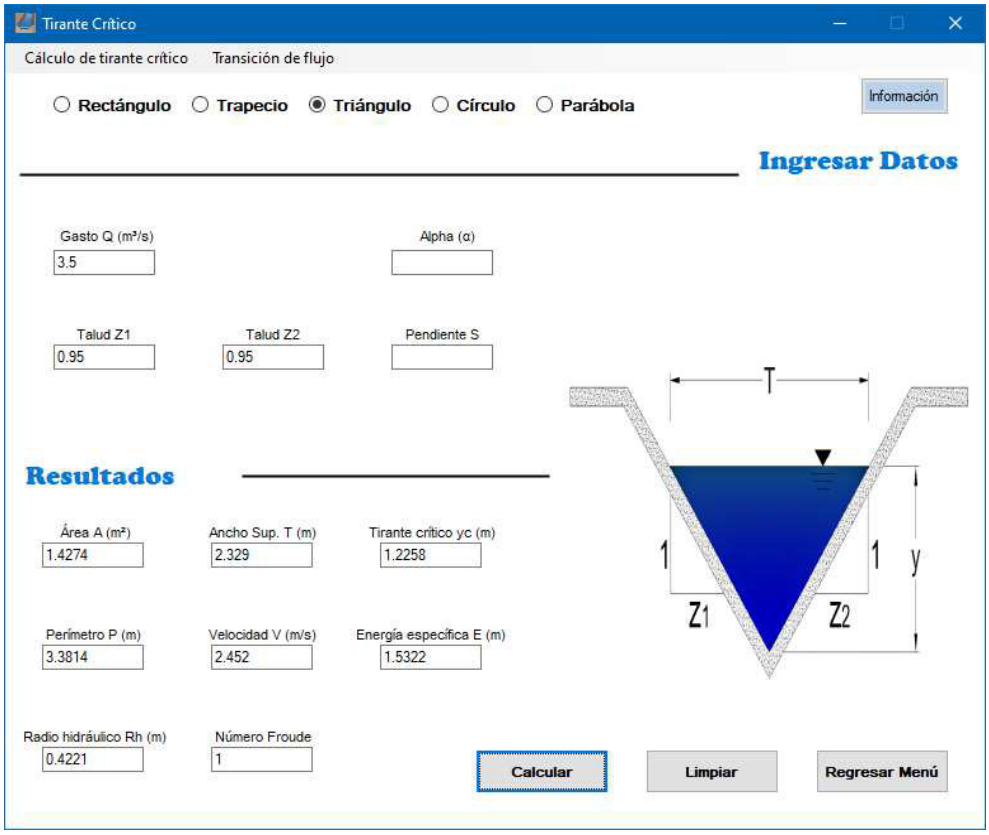
El cálculo de energía específica mínima requerida se da con la ecuación 3-7.

$$E = 1.2258\ m + \frac{\left(3.5 \frac{m^3}{s}\right)^2}{2\left(9.81 \frac{m}{s^2}\right)(0.5(0.95 + 0.95)(1.2258\ m)^2)^2} = 1.5322\ m$$

Con HYDROGECA se registran los siguientes resultados:



a)



b)

Figura 3.3 Solución del problema 2 con HYDROGECA

Fuente: Programa HYDROGECA

Algo importante a comentar es que, si el coeficiente Alpha es igual a 1, no será necesario ingresarlo ya que el programa siempre lo ha de considerar de esa manera en la ausencia del valor. En el caso de la pendiente, si se considera cero no es necesario escribirla, e incluso si se tiene pendientes que generan ángulos menores a 6° y se ingresan, el submódulo las considerará como cero ya que son demasiado pequeñas.

En cuanto a los resultados, de manera manual se hizo una aproximación a 4 decimales. Lo mismo ocurre con HYDROGECA dándonos valores iguales, pero la diferencia es que el programa realiza el cálculo de más elementos que los pedidos en el problema 2.

Problema 3. – En una comunidad existe un canal de sección trapecial, el cual presenta un gasto de 6.98 m^3/s, un ancho de plantilla de 2.01 m, un talud z1= 0.85 y el otro talud z2= 1. Además, presenta un ángulo de 10° en el fondo. Determinar el valor del tirante crítico y su respectiva energía específica si sabemos que Alpha es 1.12.

Debido a que no hay una ecuación para la sección trapecial, recurrimos a la ecuación 3-16. Ésta, en función de los elementos geométricos nos queda:

$$\frac{\left((b+0.5(z1+z2)y_c)y_c\right)^3}{b+(z1+z2)y_c}=\alpha\frac{Q^2}{g\,Cos(\theta)}$$

Sustituyendo los valores dados en el ejercicio e igualando a cero tenemos:

$$\frac{\left((2.01\,m+0.5(0.85+1)y_c)y_c\right)^3}{2.01\,m+(0.85+1)y_c}-1.12\frac{\left(6.98\frac{m^3}{s}\right)^2}{\left(9.81\frac{m}{s^2}\right)Cos(10°)}=0$$

Para encontrar el tirante crítico hacemos uso de la calculadora programable Ti – nspire cx CAS, obteniendo un valor de:

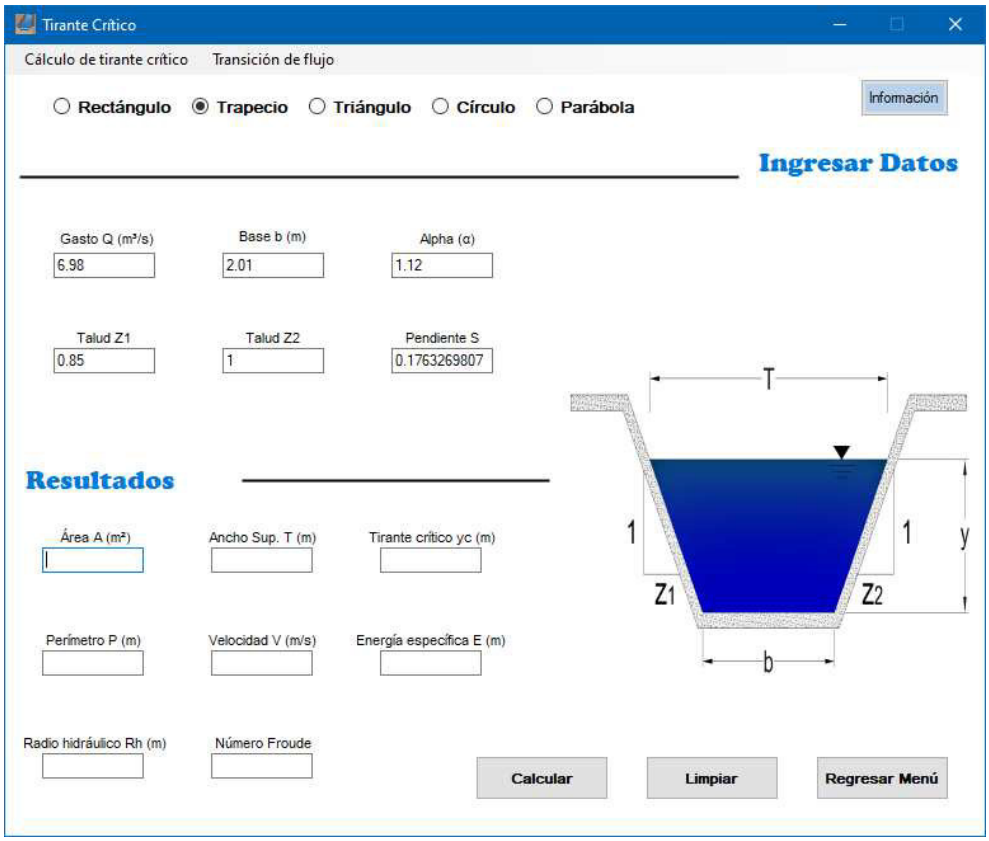
$$y_c=0.9581\,m$$

La energía específica correspondiente al tirante es:

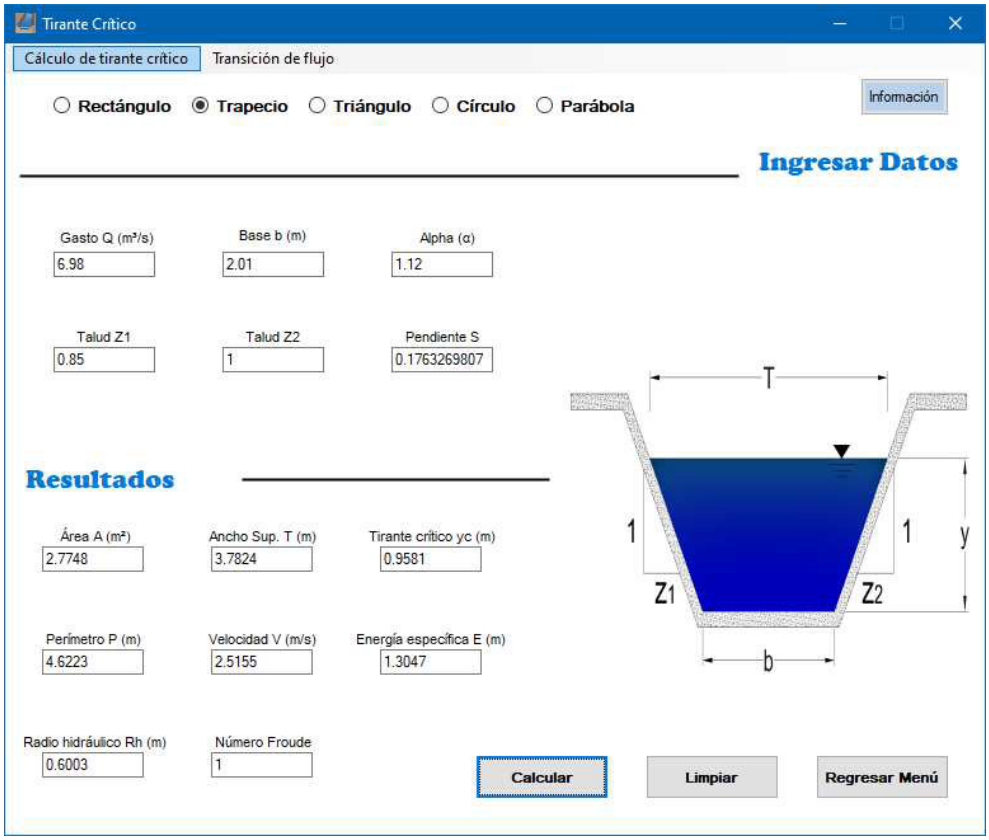
$$E=(0.9581m)Cos(10°)+1.12\frac{\left(6.98\frac{m^3}{s}\right)^2}{2\left(9.81\frac{m}{s^2}\right)\left((2.01\,m+0.5(0.85+1)0.9581m)0.9581m\right)^2}$$

$$E=1.3047\,m$$

El uso de HYDROGECA para solucionar el problema se muestra en la figura 3.4.



a)



b)

Figura 3.4 Solución del problema 3 con HYDROGECA

Fuente: Programa HYDROGECA

A diferencia del problema anterior, este involucra coeficiente Alpha diferente de 1 y el ángulo es mayor a 6°, es por esto que se escriben ambos. La pendiente se obtuvo con la relación trigonométrica de tangente (tan):

$$\tan(\theta) = \frac{\Delta y}{\Delta x} = S$$

3.3.2 Solución de problemas

El submódulo de cálculo de tirante crítico se ha programado para que genere un proceso de solución de la siguiente manera:

Lo primero es seleccionar el tipo de sección de interés, dependiendo de esto, hay ciertos datos obligatorios a ingresar como gasto y base para un rectángulo. Gasto, base y taludes para el trapecio. Gasto y taludes para un triángulo. Gasto y diámetro para la sección circular y por último, gasto y ancho superficial en el caso de una parábola. Los datos que no son indispensables en cualquier sección serán el coeficiente Alpha y la pendiente. En el supuesto de que falte alguno de los valores necesarios y se presiona el botón calcular, se abrirá una ventana con la leyenda “el cálculo es imposible ya que falta el valor de…” y nos dirá los posibles valores faltantes según la sección en que se esté.

Si los datos correspondientes a cada sección se han introducido de manera adecuada, el programa comienza a realizar el cálculo de tirante crítico. Recordemos que se puede hacer de manera analítica para ciertas secciones, pero otras han de recurrir a los métodos numéricos ilustrados en el subcapítulo “1.2 Uso de métodos numéricos”.

Lo fundamental de este submódulo es el cálculo del tirante crítico y su respectiva energía específica, pero nosotros no sólo lo hemos dejado ahí, sino que calcula algunos elementos geométricos y otros datos de interés como la velocidad y el número de Froude. Este número lo hemos de describir en el siguiente subcapítulo.

Al finalizar los cálculos, HYDROGECA procede a imprimir en cada caja de texto su respectivo resultado y con eso termina su trabajo en lo que se refiere al cálculo de tirante crítico. Si se desea analizar con otros datos, será cuestión de presionar el botón limpiar y repetir todo el proceso.

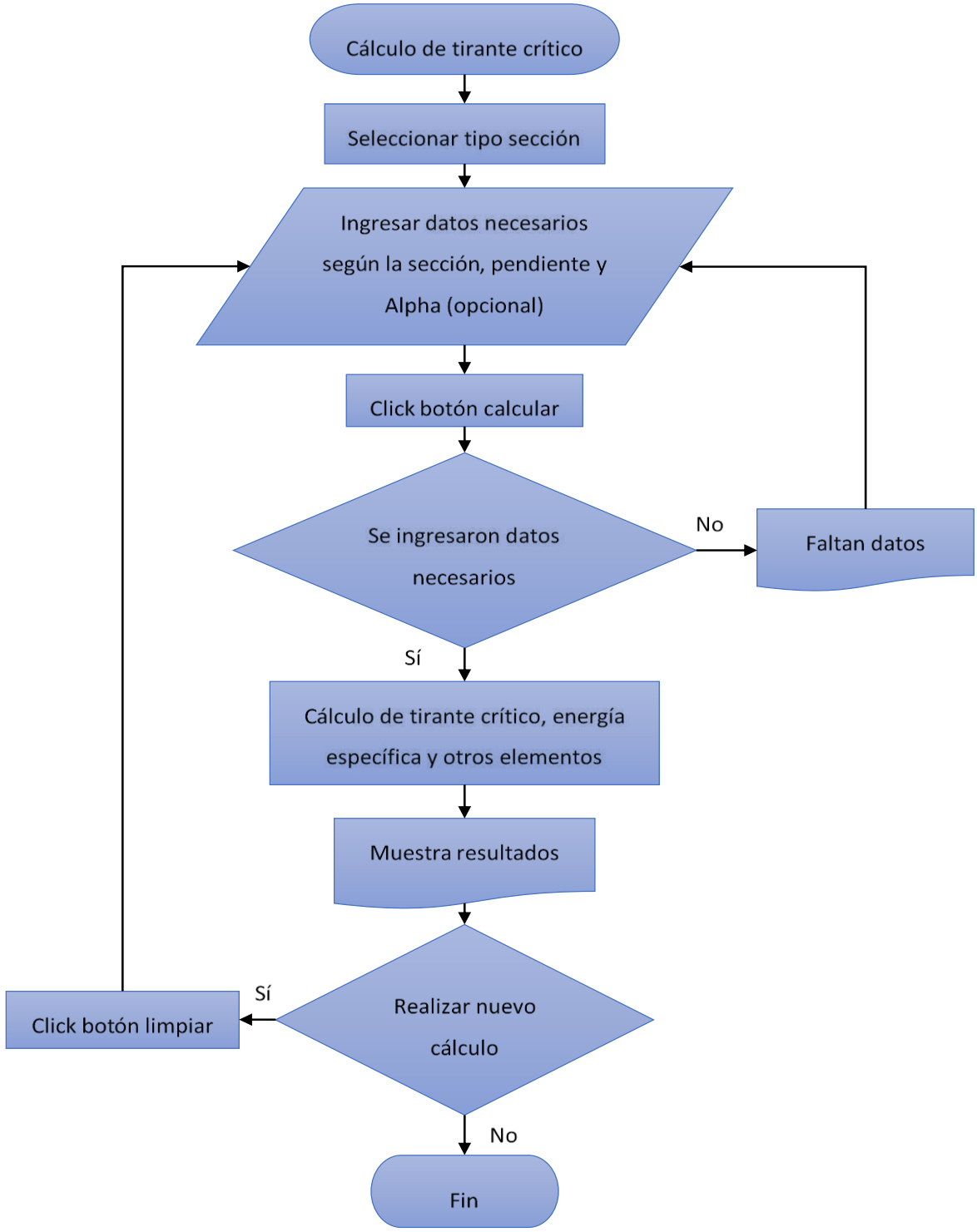


Figura 3.5 Representación del submódulo cálculo del tirante crítico a través de diagrama de flujo

Fuente: Elaboración propia

3.4 Transición de flujo

En el subcapítulo anterior se mencionó que existe una energía mínima, la cual está en función del tirante crítico. Cuando se logra presentar este tirante, podemos decir que estamos en un estado crítico. Pero ¿en qué estado nos encontramos cuando el valor de cierto tirante es mayor o menor que el crítico? Bueno, las dos opciones que tenemos son estado supercrítico para un tirante menor y estado subcrítico para uno mayor.

Hay que tener por seguro que para ir de un estado a otro siempre tendremos que pasar por el estado crítico.

Si tenemos el dato del tirante crítico es sencillo saber el estado según el tirante, pero qué pasa cuando ese valor es desconocido. Para ese caso recurrimos al número de Froude, que se definirá como una relación de las fuerzas inerciales entre las fuerzas gravitacionales (Chow, 1994).

$$F = \frac{V}{\sqrt{gL}} \qquad (3\text{--}24)$$

Donde:

- V = velocidad media de flujo
- g = aceleración de la gravedad
- L = Longitud característica

Los canales nos dictan que esa longitud siempre será la profundidad hidráulica D, mencionada en el segundo capítulo. Por tanto, para el número de Froude también involucraremos el ángulo formado en el fondo del canal y el coeficiente Alpha, manejando la fórmula siguiente:

$$F = \frac{V}{\sqrt{gD\frac{Cos(\theta)}{\alpha}}} \qquad (3\text{--}25)$$

Las distinciones de estado con Froude son las siguientes (Szymkiewicz, 2010):

- Si F = 1, Estado crítico
- Si F < 1, Estado subcrítico
- Si F > 1, Estado supercrítico

Sabiendo esta clasificación tenemos que preguntarnos, de qué tirantes se desea conocer su estado. Lo común en este tema es calcular Froude para los denominados tirantes alternos, los cuales son tres, dos menores al tirante crítico y uno mayor. Estos tienen el detalle que su respectiva energía específica es de la misma magnitud. Sin embargo, uno de los menores siempre es negativo y se ha de despreciar ya que, en un modelo matemático ese valor negativo es una respuesta, pero en cuestión a lo práctico no representa nada o no tiene un sentido físico. Por esto siempre tendremos en cuenta dos tirantes alternos denominados comúnmente como “y1” y “y2”.

3.4.1 Casos de interés en transición de flujo

Existen dos casos en específico con respecto a este tema, los cuales son cuando se mantiene el ancho del canal y cuando se posee un ancho variable.

Cuando se mantiene el ancho hay dos posibilidades, primero se hace mención al caso de una plantilla ascendente que a su vez provoca una contracción vertical y el cambio de pendiente siempre ha de ser positivo. El segundo caso es que se presente un descenso en la plantilla y esto genera una ampliación vertical, además, habrá una pendiente negativa.

Para mostrar ambos casos gráficamente, supongamos una sección aguas arriba, la cual tiene la curva de energía específica la cual se muestra en la figura 3.6.

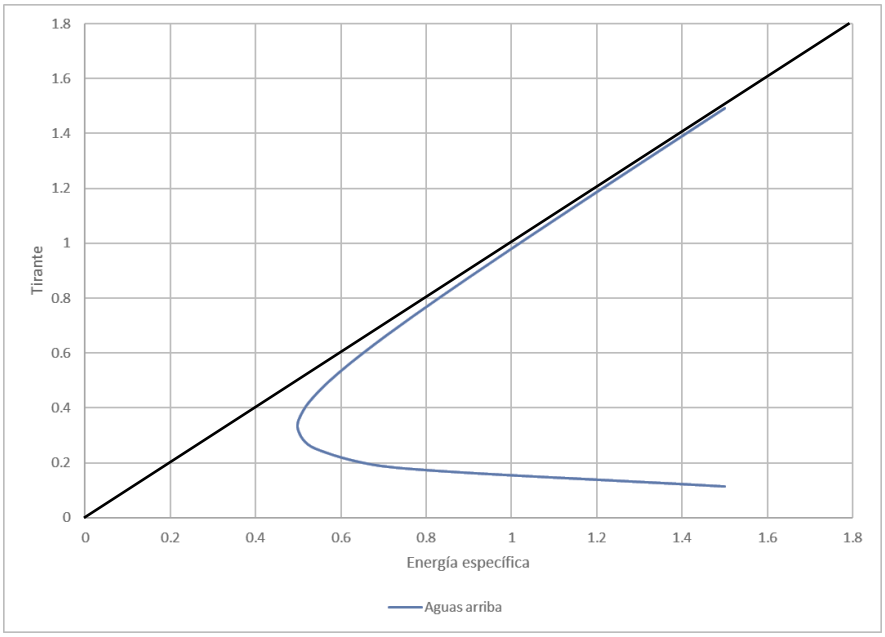
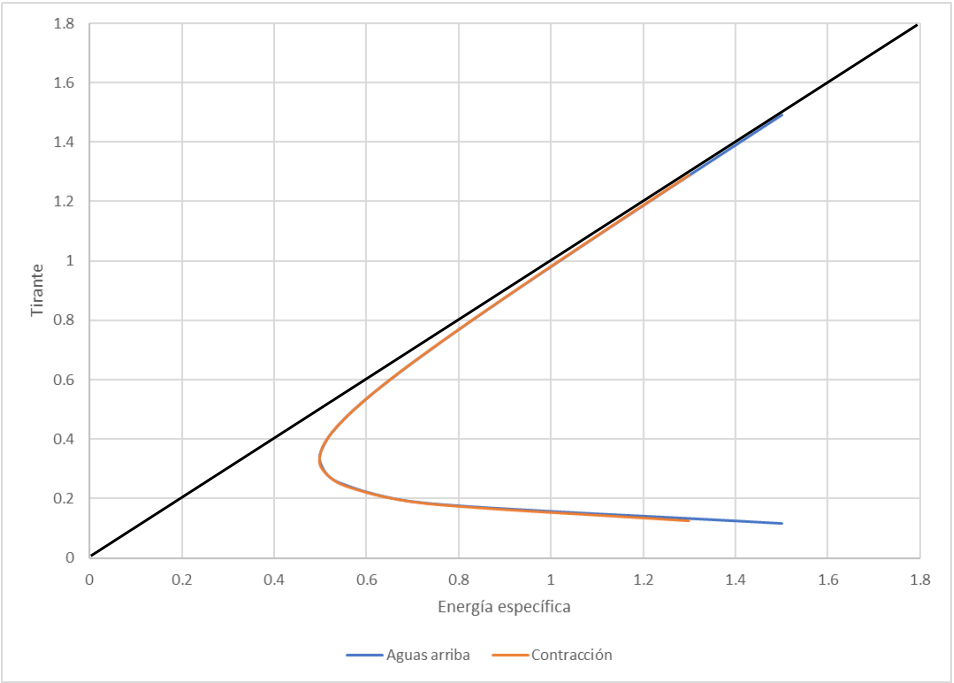


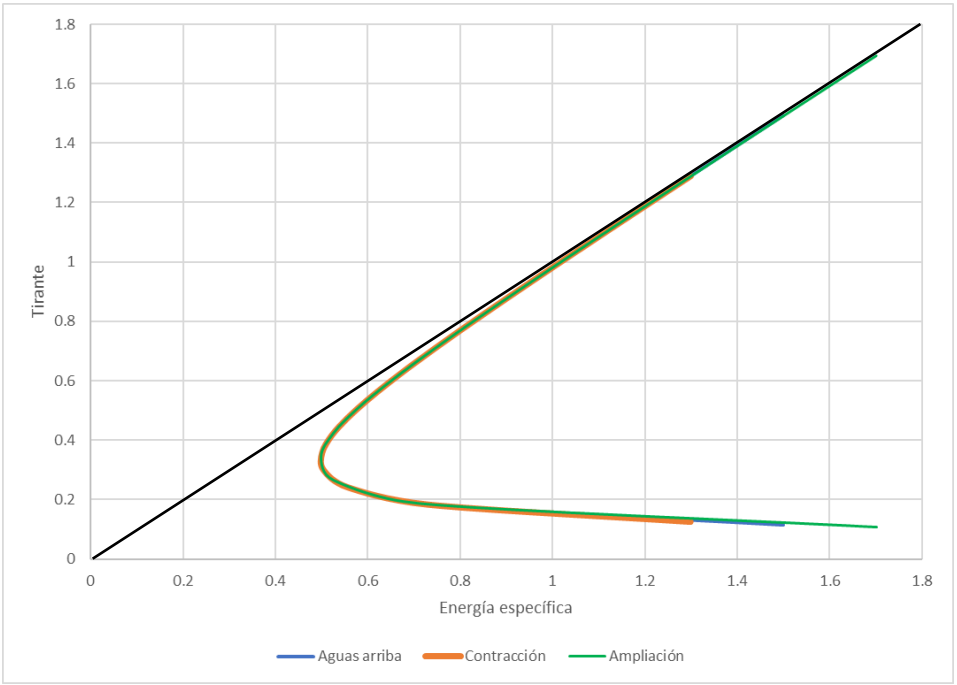
Figura 3.6 Curva para cierta sección aguas arriba

Fuente: Elaboración propia

Si esa misma sección presenta una contracción vertical y luego una ampliación vertical, su curva varía de la manera mostrada en la figura 3.7 a y 3.7 b respectivamente.



a)



b)

Figura 3.7 Curvas de energía específica contracción y ampliación sección constante

Fuente: Elaboración propia

Algo que analizar es que, el valor de energía específica mínima y tirante crítico no cambia en ninguno de los dos casos. Sin embargo, la curva para la contracción se vuelve menos prolongada que la sección aguas arriba y la ampliación nos da una curva más prolongada.

En el caso donde el ancho del canal es variable y se presente una contracción o una ampliación en la base del canal se deberá considerar obtener los valores correspondientes de área y sustituirse en la ecuación de energía específica. Veamos ejemplo del comportamiento de la curva de energía específica si varía el ancho.

Nuevamente consideramos una sección aguas arriba con su respectiva curva ilustrada en la figura 3.8.

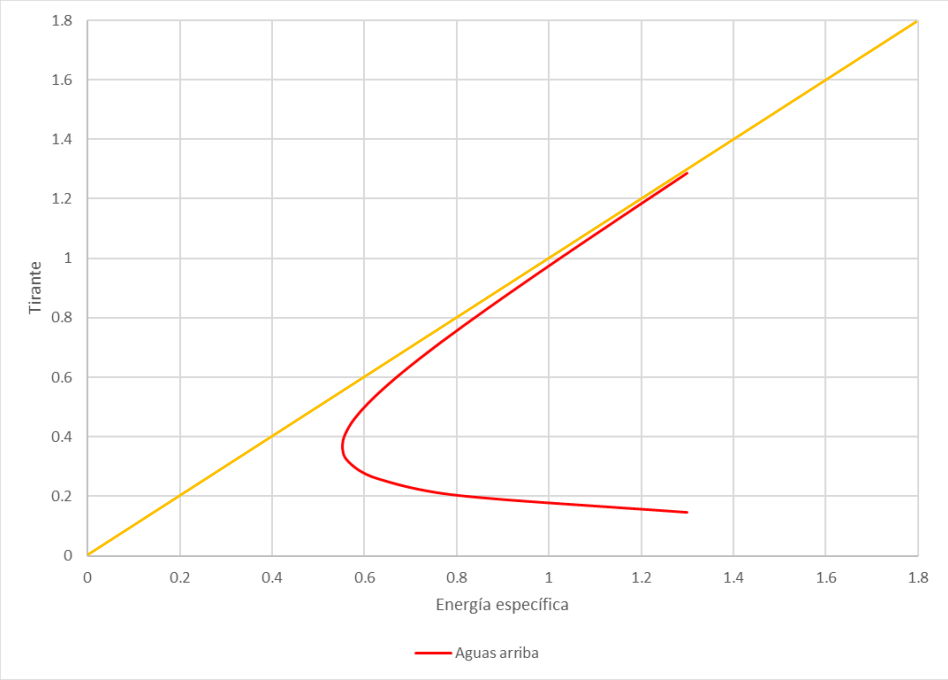
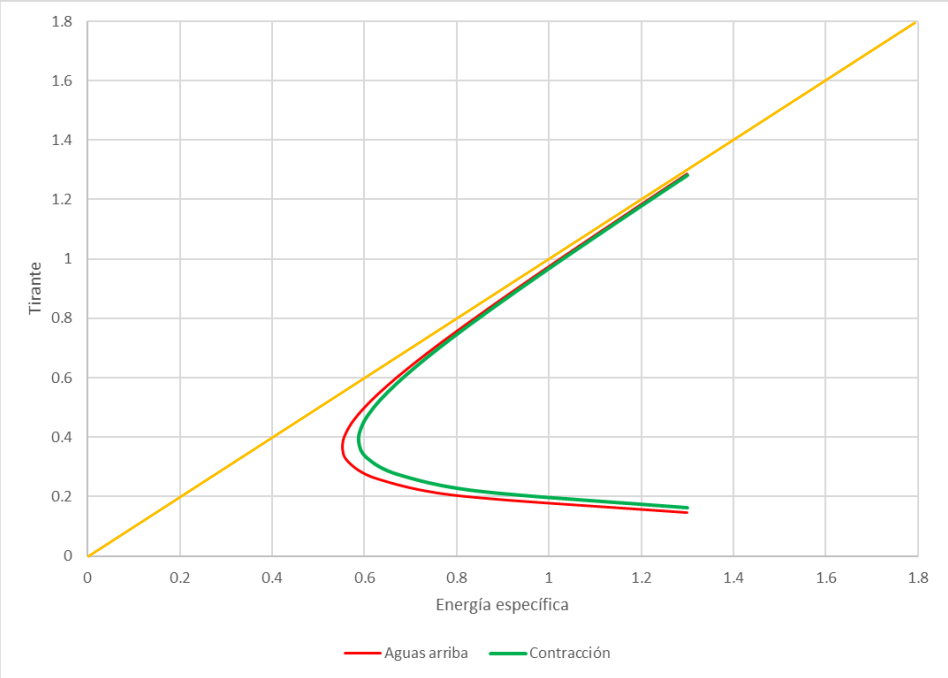


Figura 3.8 Curva de una sección aguas arriba

Si esta sección a lo largo del canal sufre una contracción en el ancho de su base, tendremos el cambio de la curva como se observa en la figura 3.9 a. Caso contrario, si se presenta una ampliación tendremos la curva azul como se muestra en la figura 3.9 b.



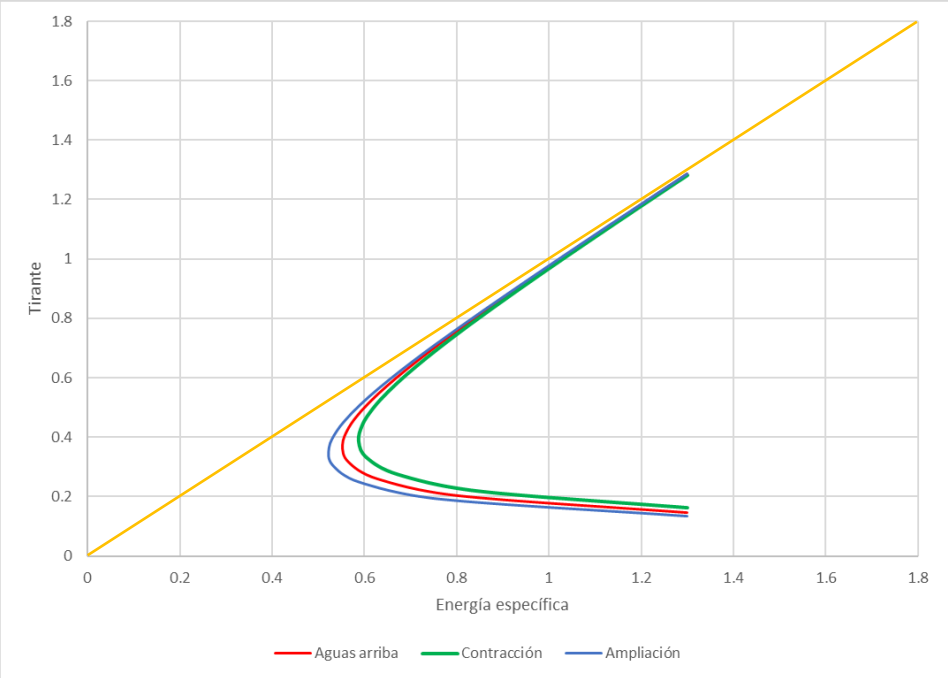


Figura 3.9 Curvas de energía específica contracción y ampliación sección variable

Fuente: Elaboración propia

En este caso cada curva mantiene la misma energía de los tirantes alternos pero la energía mínima ha de cambiar y con ello el tirante crítico.

Los casos de interés son mencionados pero el submódulo de HYDROGECA realiza un cálculo a la vez, esto quiere decir que no podemos graficar varias secciones a la vez.

3.4.2 Ejemplo ilustrativo submódulo transición de flujo

Problema 4.- Se desea conocer el tirante crítico, energía específica mínima, los tirantes alternos y su respectivo estado para una energía de 2 m en un canal rectangular con las siguientes características: base b= 3.25 metros, gatos Q = 5.5 m³/s, pendiente considerada prácticamente cero y un coeficiente Alpha de 1. Encontrar de manera manual y con el uso de HYDROGECA los valores deseados.

El primer paso es conocer el tirante crítico que, si recordamos, existe una fórmula analítica para la sección rectangular:

$$y_c = \sqrt[3]{\alpha \frac{Q^2}{b^2 g\ Cos(\theta)}} = y_c = \sqrt[3]{\frac{(5.5 \frac{m^3}{s})^2}{(3.25\ m)^2 (9.81 \frac{m}{s^2})}} = 0.6634\ m$$

Procedemos a conocer la energía específica mínima:

$$E_{Min} = \ y_c + \frac{Q^2}{2gA^2} = 0.6634\ m + \frac{(5.5 \frac{m^3}{s})^2}{2\left(9.81 \frac{m}{s^2}\right)(0.6634\ m\ x\ 3.25\ m)^2} = 0.9951\ m$$

Es necesario comprobar que la energía para la que se piden los tirantes alternos es mayor que la energía mínima, como se muestra a continuación:

$$E = 2\ m > E_{Min} = 0.9951\ m \ \ \therefore existen\ tirantes\ alternos$$

Para los tirantes alternos tenemos que igualar la fórmula de energía específica a 2 m.

$$2\ m = y + \frac{(5.5 \frac{m^3}{s})^2}{2\left(9.81 \frac{m}{s^2}\right)(y\ x\ 3.25\ m)^2}$$

Igualamos a cero la ecuación anterior:

$$0 = y - 2\ m + \frac{(5.5 \frac{m^3}{s})^2}{2\left(9.81 \frac{m}{s^2}\right)(y\ x\ 3.25\ m)^2}$$

Recurrimos a la calculadora programable Ti – nspire cx CAS para encontrar las soluciones, en donde:

- y1 = 1.9621 m
- y2 = 0.2924 m
- y3 = -0.2546 m

El tirante y3 se desprecia por ser negativo y se contemplan solamente los otros dos. También es necesario calcular la velocidad para poder obtener el número de Froude para los tirantes.

Para y1:

$$V_1 = \frac{Q}{A} = \frac{5.5\frac{m^3}{s}}{(3.25\,m)(1.9621\,m)} = 0.8625\frac{m}{s}$$

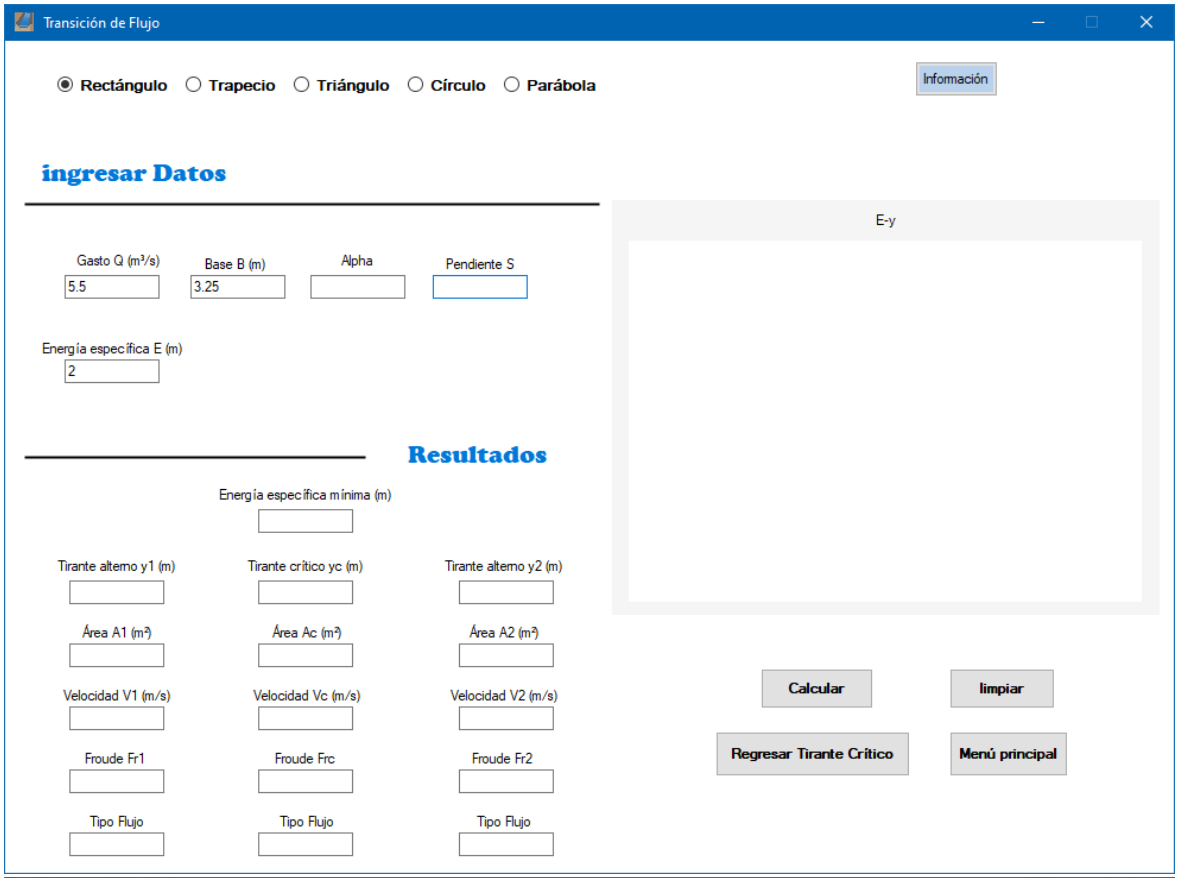
$$F_1 = \frac{V_1}{\sqrt{gD}} = \frac{0.8625\frac{m}{s}}{\sqrt{(9.81\frac{m}{s})(1.9621\,m)}} = 0.1966 \quad \therefore\ y1\ está\ en\ estado\ subcrítico$$

Para y2:

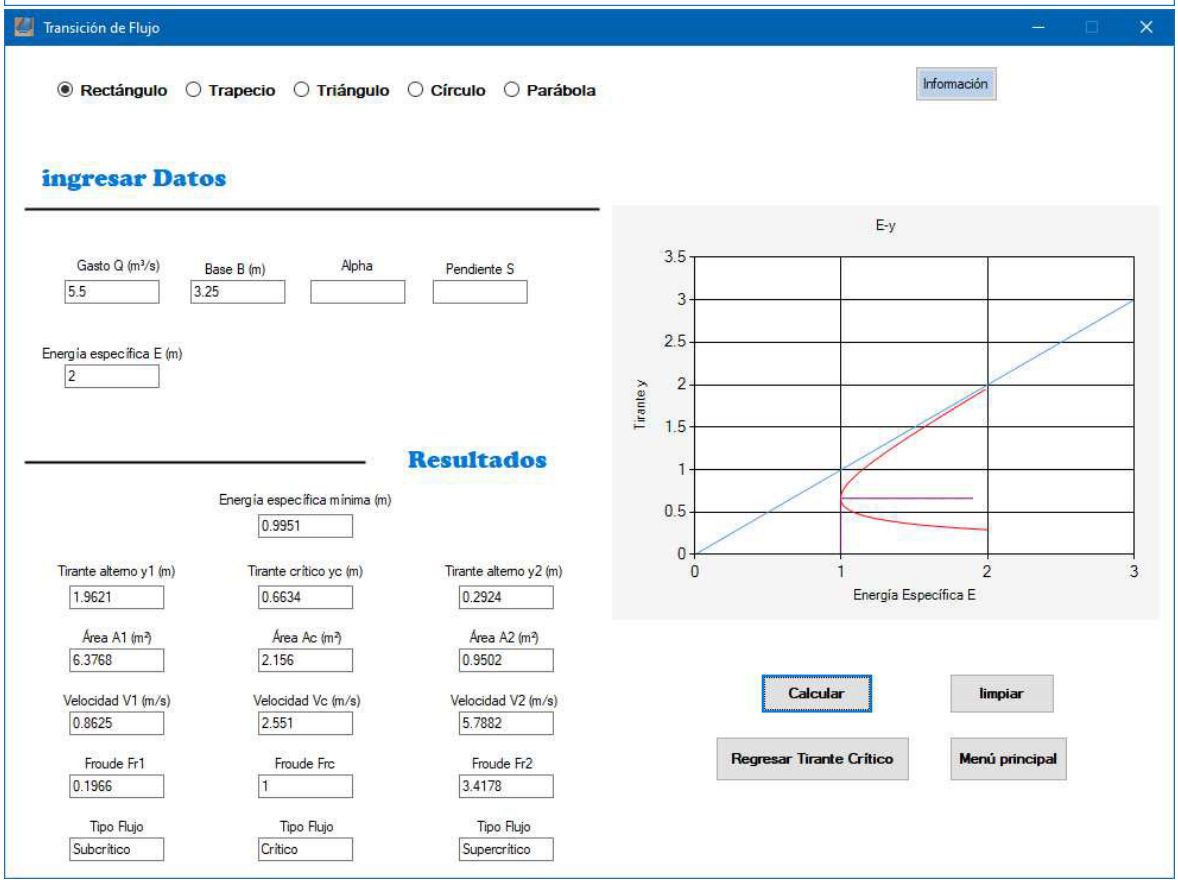
$$V_2 = \frac{Q}{A} = \frac{5.5\frac{m^3}{s}}{(3.25\,m)(0.2924\,m)} = 5.7876\frac{m}{s}$$

$$F_2 = \frac{V_2}{\sqrt{gD}} = \frac{5.7876\frac{m}{s}}{\sqrt{(9.81\frac{m}{s})(0.2924\,m)}} = 3.4173 \quad \therefore\ y2\ está\ en\ estado\ supercrítico$$

Habiendo finalizado el cálculo de Froude para el tirante alterno 2, se procede a realizar el ejercicio con el uso de HYDROGECA.



a)



b)

Figura 3.10 Solución del problema 4 con HYDROGECA

Fuente: Programa HYDROGECA

Debido a que son varios los resultados a comparar, se genera una tabla donde se comparan los valores obtenidos de forma manual contra lo generado por el programa.

Elemento	Manualmente			Programa		
Tirante (m)	1.9621	0.6634	0.2924	1.9621	0.6634	0.2924
Energía (m)	2	0.9951	2	2	0.9951	2
Velocidad (m/s)	0.8625		5.7876	0.8625	2.551	5.7882
Froude	0.1966		3.4173	0.1966	1	3.4178
Estado	Subcrítico	Crítico	Supercrítico	Subcrítico	Crítico	Supercrítico

Tabla 3.1 Comparativa de resultados del problema 4 de forma manual vs HYDROGECA

Fuente: Elaboración propia

Al ver la comparativa de la tabla 3.1 se observa que la mayoría de los resultados son iguales a excepción de la velocidad y el número de Froude para los tirantes alternos y2. Esto puede ser debido a que el programa considera más de 4 decimales al momento de realizar las operaciones de manera interna pero cuando se muestran los resultados se hace un redondeo. A pesar de estas pequeñas diferencias, no es algo significativo y el submódulo cumple satisfactoriamente el objetivo.

3.4.3 Solución de problemas

El submódulo de transición de flujo es muy similar al de cálculo de tirante crítico referente a los datos necesarios para poder realizar los cálculos. Recordemos que para la sección rectangular es obligatorio un gasto y una base; para el trapecio se requiere un gasto, una base y los taludes; el triángulo ocupa un gasto y taludes; el círculo necesita los valores de un gasto y un diámetro y, por último, la parábola al igual que las secciones anteriores ha de conocerse un gasto, pero también un ancho superficial.

La diferencia de este submódulo respecto al anterior es que, en transición de flujo, debemos dar un valor de energía específica para poder conocer los tirantes alternos independientemente del tipo de la sección.

Podemos decir que la forma para dar solución a los problemas relacionados a la transición de flujo es la siguiente:

Primeramente, se selecciona una de las 5 secciones disponibles para proceder a ingresar los datos necesarios (comentados unos párrafos arriba) según la sección elegida y una energía específica, la cual debe ser siempre mayor a la energía mínima perteneciente al tirante crítico. El coeficiente Alpha y la pendiente son opcionales, en caso de no poner valores, se considerará 1 y 0 respectivamente. Seguido de la introducción de datos se presiona el botón calcular, e inmediatamente el programa comienza a realizar la comprobación del ingreso de datos necesarios obligatorios. En caso de que falte algún dato se mostrará una ventana con un mensaje indicándolo. Otra comprobación es que la energía específica para los tirantes alternos sea mayor que la mínima requerida, si no es así, sólo se realiza el cálculo e imprimirá los resultados en las respectivas cajas de texto del tirante crítico y energía específica mínima.

Dado el caso que ningún dato necesario falte y la energía ingresada sea válida, el programa comenzará a realizar las operaciones en cuanto se dé un click al botón calcular.

El primer valor a determinar en cualquier sección es el tirante crítico y la energía mínima. Esto es porque se usa un método abierto para encontrar los tirantes alternos y como estos están cerca del valor crítico, al proponer este valor como inicial en Newton - Raphson se tendrá una convergencia rápida. De ahí en adelante realiza cálculos mediante fórmula y con base en Froude determina qué tipo de flujo es. Otros datos que se buscan son los puntos que integran la curva de energía específica, la cual va desde el tirante alterno 1 y termina en el tirante alterno 2.

Cuando se hayan finalizado las operaciones, se empiezan a imprimir en las cajas de texto cada uno de los resultados y se genera una gráfica de la curva energía específica para el caso en estudio. Esta gráfica contiene tres líneas diferentes a los puntos de tirante contra energía específica, una está a 45° y las otras dos son perpendiculares entre sí. De esas dos, una es paralela al eje de los tirantes y va desde cero hasta el tirante crítico. La que es paralela al eje horizontal irá desde la energía mínima y llegará al valor de la energía específica de los tirantes alternos.

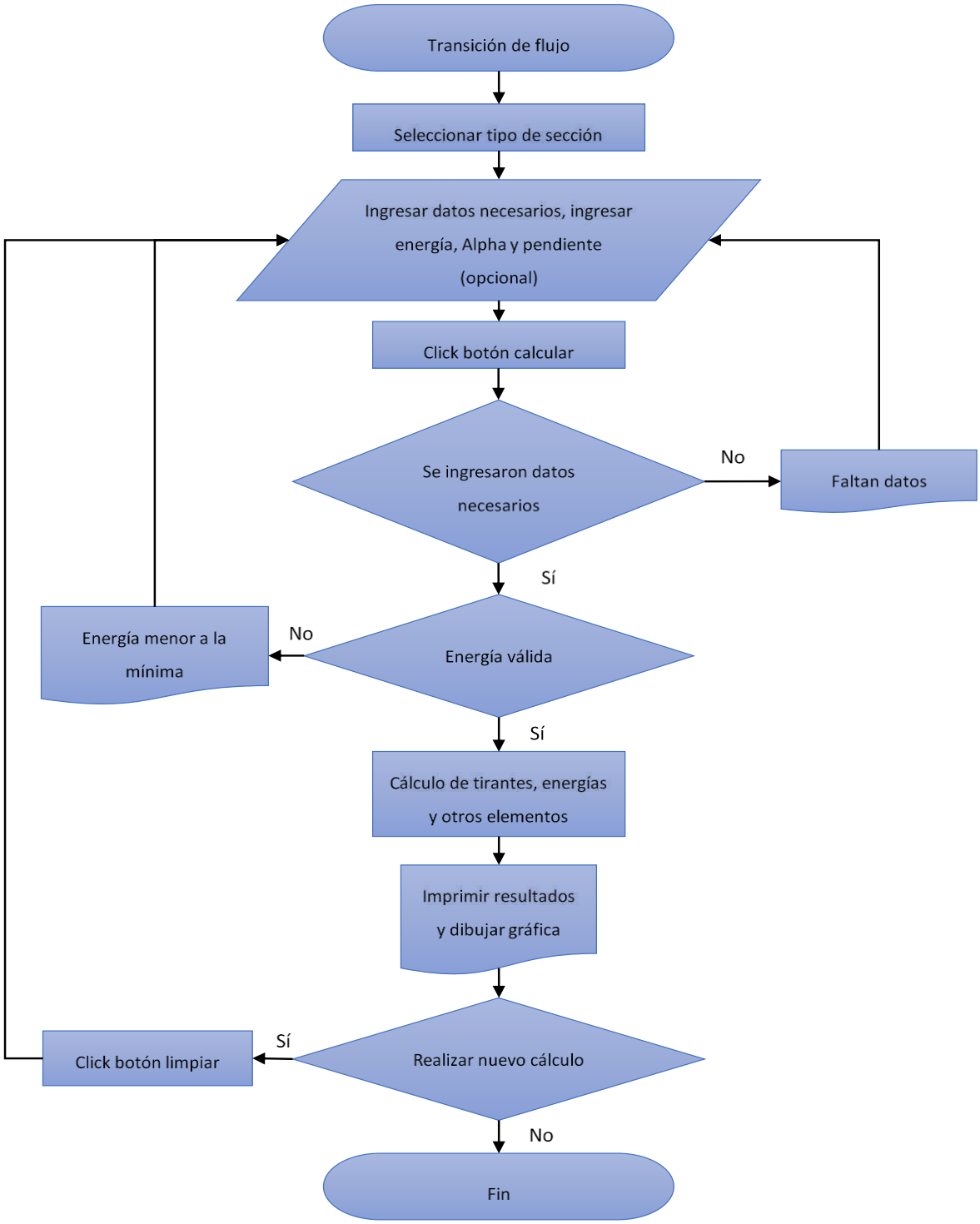


Figura 3.11 Representación del submódulo transición de flujo a través de diagrama de flujo

Fuente: Elaboración propia

4 Capítulo flujo normal

Este es tal vez considerado el capítulo más largo, esto porque el módulo de flujo normal cuenta con cuatro submódulos: Factor de sección, Sección de máxima eficiencia, Tirante conocido y Sección – Pendiente.

Debido a que son varios los temas a tocar, se hará un resumen lo más sencillo posible de la teoría y se tratará de hacer más énfasis en lo que respecta al programa HYDROGECA.

4.1 Flujo uniforme y sus características

Existen ciertos aspectos a cumplir para que podamos decir que se nos presenta un flujo uniforme. Imaginemos un canal que se ha dividido en cierta cantidad de secciones, cada sección debe tener exactamente igual los elementos mencionados a continuación:

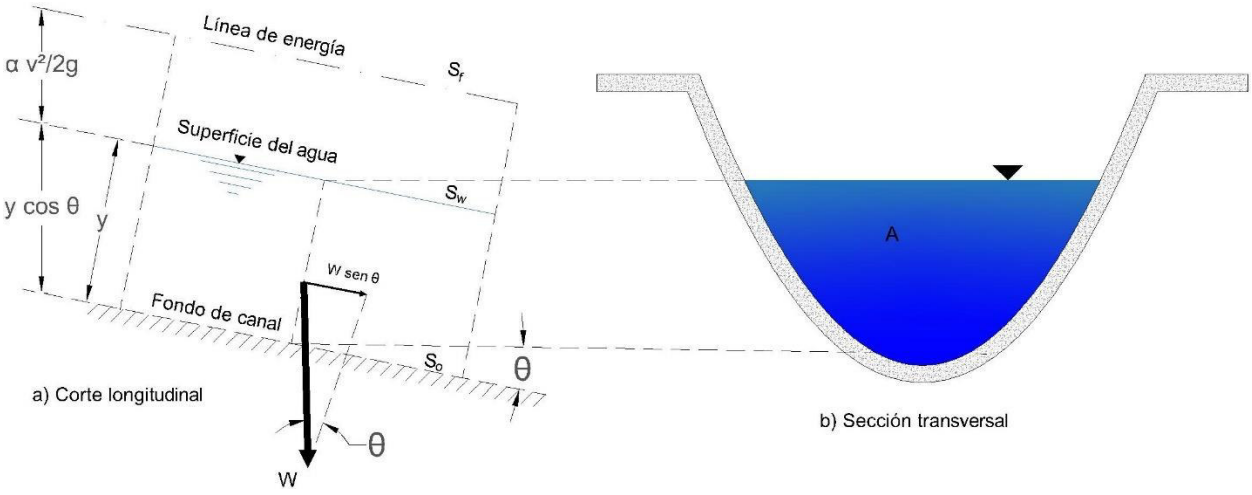
- Profundidad hidráulica
- Área mojada o hidráulica
- Caudal
- Velocidad

En este tema el tirante hidráulico o profundidad hidráulica se le hará mención como tirante normal o profundidad normal.

Las pendientes han de ser un factor fundamental a considerar en un flujo uniforme, siendo tres las de importancia, las cuales deben cumplir la condición de paralelismo, las cuales se describen a continuación:

- S_f = Pendiente de línea de energía
- S_w = Pendiente a nivel de superficie de agua
- S_o = Pendiente en fondo de canal

La figura 4.1 a) representa un esquema donde se aprecian las distintas pendientes mencionadas.



(Cowan, 1956)

Figura 4.1 Esquema de flujo uniforme

Fuente: Elaboración propia

Aunque no son tantas la condiciones que se deben dar para tener un flujo uniforme, en la realidad en un canal natural es muy poco probable que se presente este tipo de flujo o hasta en uno artificial. A pesar de ello, en cualquier diseño de canales es básico considerar esta condición (Sotelo, 2002).

4.2 Velocidad en el flujo uniforme

Conocer la velocidad y el caudal en un cauce natural es elemental en el diseño de las obras hidráulicas. Para la obtención de estos datos podemos mencionar dos formas, la primera consiste en generar esta información a partir de datos topográficos de la sección transversal y de estudios hidrológicos e hidráulicos con los que se determine el gasto en determinada sección de un cauce y por consiguiente la velocidad.

La segunda metodología es medir directamente la velocidad y el gasto en el punto de interés. En este caso empleamos estaciones hidrométricas o bien, se hacen las mediciones con uso de instrumentos como el molinete o medidores ultrasónicos. Comúnmente se recurre al último ya que la información hidrométrica a veces es imprecisa y en otras ocasiones es inexistente.

En canales abiertos la mayoría de los modelos matemáticos para la velocidad media del flujo se representa por la siguiente ecuación:

$$V_{med} = CR_h^x S_o^y \qquad (4\text{--}1)$$

Donde:

- V_{med} = Velocidad media
- C = Coeficiente que asocia la rugosidad del material
- R_h = Radio hidráulico
- S_o = Pendiente en el fondo del canal
- x, y = Exponentes

de la ecuación 4-1 se derivaron varías ecuaciones para la descripción de la velocidad media, de las cuales se mencionan algunas a continuación (Osman, 2006):

- Velocidad media de Chezy
- Velocidad media de Bazin
- Velocidad media Ganguillet y Kutter
- Velocidad media de Powell
- Velocidad media de Manning

Aunque podríamos hacer un análisis de cada ecuación, en el actual trabajo lo dejaremos de lado y solamente nos enfocamos en la velocidad media de Manning. Esta ecuación destaca sobre todas debido a que es la usada en este módulo.

El trabajo del ingeniero Robert Manning en el tema de velocidad media dio como resultado la ecuación 4-2.

$$V = \frac{1.49}{n} R_h^{\frac{2}{3}} S^{\frac{1}{2}} \qquad (4\text{--}2)$$

El programa HYDROGECA no considera esta expresión ya que es usada en el sistema de medición inglés. El sistema métrico expresa la ecuación de Manning de la siguiente forma:

$$V = \frac{1}{n} R_h^{\frac{2}{3}} S^{\frac{1}{2}} \qquad (4\text{--}3)$$

Donde:

- V = Velocidad media
- n = Coeficiente de rugosidad
- R_h = Radio hidráulico
- S = Pendiente del fondo de canal

Aunque podría creerse que no es gran cambio, si se usa HYDROGECA para resolver cierto ejercicio se recomienda que se use solamente el sistema métrico en las unidades.

4.3 Coeficiente de rugosidad

Es común que ciertos efectos o propiedades físicas se traten de explicar con un modelo matemático, en este sentido considerar la posibilidad de que este modelo sea aplicable a la mayor cantidad de casos. En el tema de la rugosidad en canales, es muy difícil tener un modelo exacto el cual proporcione un valor que se pueda considerar por completo como verdadero (Attari et al, 2021).

La complejidad se genera por diferentes factores que han de tener efecto en la rugosidad. Ven Te Chow (1994) menciona que los de mayor influencia son los siguientes:

- Rugosidad superficial
- Vegetación
- Irregularidad del canal
- Alineamiento del canal
- Sedimentación y socavación
- Obstrucción
- Tamaño y forma del canal
- Nivel y caudal

Es necesario recalcar que los puntos anteriores se enfocan más para un canal natural que para un artificial, sin embargo, los canales artificiales no se ven excluidos de todos los factores.

Aunque ya se dijo que no hay ecuación de la cual se obtenga un valor de rugosidad que se pueda interpretar como el exacto, Cowan (1956) genera una fórmula, la cual al tener en cuenta ciertas características presentes en el canal, ayuda a darnos una idea de un coeficiente, el cual se muestra en la ecuación 4-4.

$$n = (n_0 + n_1 + n_2 + n_3 + n_4)m_5 \qquad (4\text{--}4)$$

Para dar valor a cada variable de la ecuación 4-4 es necesario considerar la tabla 4.1.

Condiciones de canal		Valores	
Material involucrado	Tierra	n_0	0.020
	Corte en roca		0.025
	Grava fina		0.024
	Grava gruesa		0.028
Grado de irregularidad	Suave	n_1	0.000
	Menor		0.005
	Moderado		0.010
	Severo		0.020
Variación de la sección transversal	Gradual	n_2	0.000
	Ocasionalmente alternante		0.005
	Frecuentemente alternante		0.010-0.015
Efecto relativo de las obstrucciones	Insignificante	n_3	0.000
	Menor		0.010-0.015
	Apreciable		0.020-0.030
	Severo		0.040-0.060
Vegetación	Baja	n_4	0.005-0.010
	Media		0.010-0.025
	Alta		0.025-0.050
	Muy alta		0.050-0.100
Grado de los efectos por meandros	Menor	m_5	1.000
	Apreciable		1.150
	Severo		1.300

Tabla 4.1 Valores para cálculo de coeficiente de rugosidad

Fuente: (Cowan, 1956)

Para n_2, n_3 y n_4 no hay una forma en específico de seleccionar un valor en los rangos, por lo que se ha de proponer uno que esté dentro de los límites.

En la práctica es común que la rugosidad se elija con base en la experiencia, sin embargo, al momento del diseño es aún más común tomar cierto valor de tablas que contienen la rugosidad para materiales usados en obras hidráulicas como concretos con sus distintos acabados, tuberías o

material terroso. Estas tablas son generadas por dependencias que a la vez realizaron pruebas a los materiales para dictaminar o dar un valor confiable.

4.4 Ecuación a resolver para tirante normal

En el tercer capítulo se involucró la ecuación de continuidad, pero no se dio una descripción de ella. De manera sencilla diremos que esta se basa en que el gasto Q de un fluido permanece constate a lo largo de todo el canal de conducción. Se ha de considerar que esta ecuación es un caso particular del principio de conservación de la masa.

Para dos secciones de un mismo canal tendremos que el gasto es la multiplicación de la velocidad media con la que se desplaza un fluido por el área de la sección transversal (Ecuación 4-5).

$$Q = VA \qquad (4\text{--}5)$$

Donde:

- Q = Gasto
- V = Velocidad media del flujo
- A = Área mojada o hidráulica

Como se ha considerado el mismo gasto para dos secciones, se cumplirá que:

$$Q = V_1 A_1 = V_2 A_2$$

La ecuación 4-5 es de gran importancia debido a que permitirá determinar el tirante normal. Si sustituimos 4-3 en 4-5 obtenemos lo siguiente:

$$Q = \frac{1}{n} R_h^{\frac{2}{3}} S^{\frac{1}{2}} A \qquad (4\text{--}6)$$

Al igual que en la ecuación para tirante crítico, en el tirante normal tenemos que dejar de un lado de la igualdad todos los términos que involucren el tirante. En este caso sólo es el radio hidráulico y el área hidráulica y como ambos están del mismo lado, pasamos el coeficiente de rugosidad y la pendiente con el gasto, quedando:

$$AR_h^{\frac{2}{3}} = \frac{Q\ n}{S^{\frac{1}{2}}} \qquad (4\text{--}7)$$

Para cualquier sección se tendrá que resolver la ecuación 4-7 y para el caso del tirante normal, por lo que no se puede generar una solución analítica como en el tirante crítico.

4.5 Factor de sección

Cuando vamos a diseñar cierta estructura u obra, lo ideal es tener libertad para proponer la solución más factible, sin embargo, mayormente se presentan situaciones que nos limitan y tenemos que adaptarnos. En este tema, hacemos referencia al diseño de un canal bajo algún factor que nos condicione, por ejemplo, imaginemos que en un bulevar existe un canal natural entre los dos carriles y debido a que se quiere tener más control del flujo del agua, se construirá un canal artificial. Es un hecho que el ancho que existe entre los carriles será un factor a considerar ya que, no sería factible ampliar la distancia. Para este caso nos hemos de regir por ese ancho.

Otro caso sería en un canal trapecial que ya existe, el cual requiere que se realice una ampliación, pero se quiere dejar el mismo talud, en ese caso el factor que los condicionará es el talud.

4.5.1 Ejemplo ilustrativo submódulo factor de sección

Problema 5.- En una comunidad hay un canal natural que se quiere revestir de concreto liso pulido (n=0.013). El canal tiene aproximadamente la forma de un trapecio con un ancho de plantilla de 5 metros y una relación de talud de 0.8. ¿Cuál será el tirante normal si se requiere transportar 35 m³/s y se propone una pendiente de 0.005? Además, es necesario conocer la velocidad que se ha de presentar.

Para la solución sustituimos los valores en la ecuación 4-7 como se muestra a continuación:

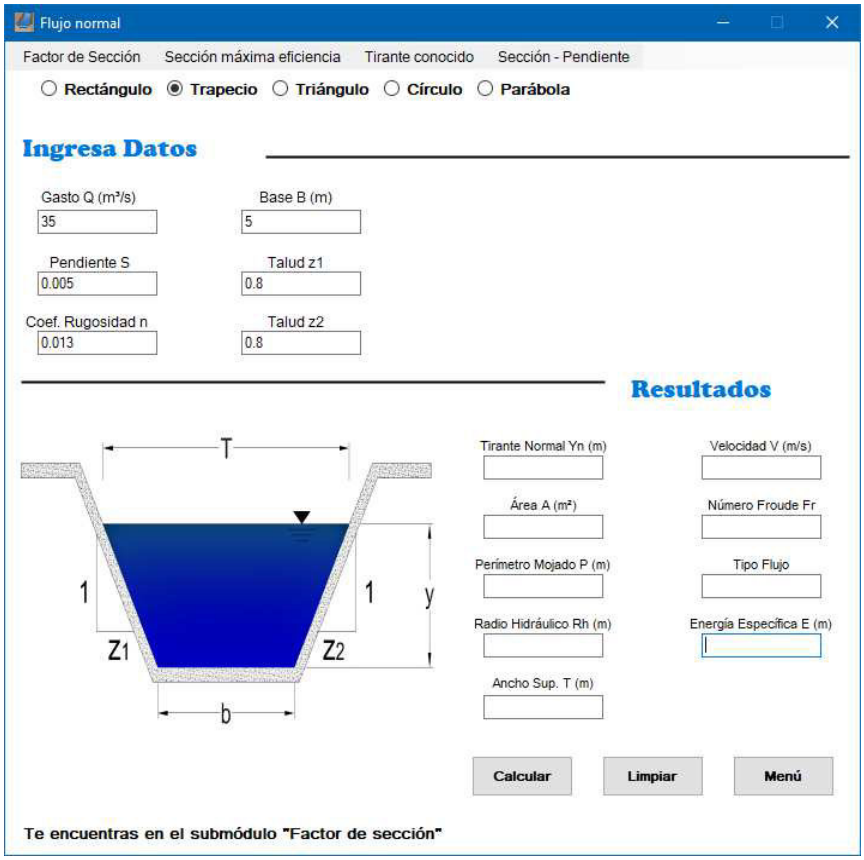
$$[(5m + 0.5(0.8 + 0.8)\ y_n)y_n]\left[\frac{(5m + 0.5(0.8 + 0.8)\ y_n)y_n}{5m + y_n\left(\sqrt{1 + 0.8^2} + \sqrt{1 + 0.8^2}\right)}\right]^{\frac{2}{3}} = \frac{\left(35\frac{m^3}{s}\right)(0.013)}{\sqrt{0.005}}$$

Con el apoyo de la calculadora Ti – nspire cx CAS se resuelve, obteniendo un valor de tirante normal de 1.182358 m.

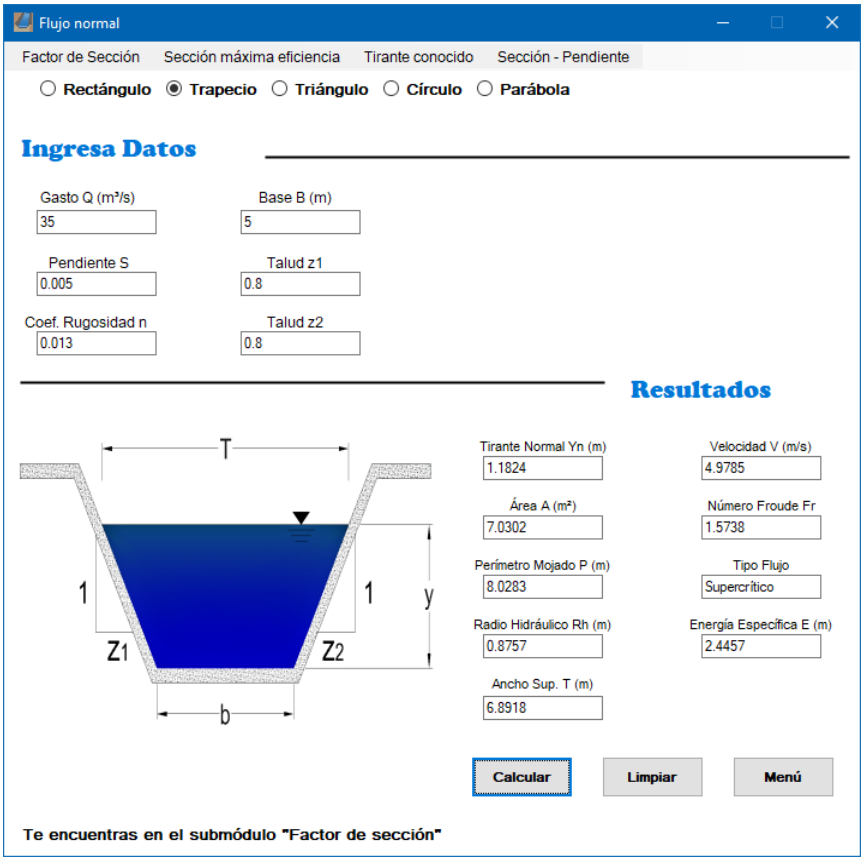
La velocidad se ha de encontrar con la ecuación 4-3, como se muestra en seguida:

$$V = \left(\frac{1}{0.013}\right)\left(\frac{(5m + 0.5(0.8 + 0.8)1.182358\ \text{m}\ )1.182358\ \text{m}}{5m + 1.182358\ \text{m}\left(\sqrt{1 + 0.8^2} + \sqrt{1 + 0.8^2}\right)}\right)^{\frac{2}{3}} \sqrt{0.005} = 4.978545\,\frac{m}{s}$$

Con HYDROGECA se obtienen los siguientes resultados:



a)



b)

Figura 4.2 Solución del problema 5 con HYDROGECA

Fuente: Programa HYDROGECA

Si comparamos los resultados de manera manual contra los arrojados por el programa es notorio la similitud, en donde el manejo de los decimales es una diferencia. Sin embargo, el uso de cuatro decimales es satisfactorio.

Además del tirante normal y la velocidad, HYDROGECA calcula e imprime los elementos geométricos que se ven involucrados en la solución a la ecuación 4-7.

4.5.2 Solución de problemas

Para el submódulo Factor de sección es necesario llenar todas las cajas de texto que se encuentran en el apartado “ingresar datos”. Si se da el caso de que falte algún dato y se presiona el botón calcular saltará un mensaje con la leyenda “Faltan datos para realizar el cálculo”.

Una vez ingresado los datos requeridos, el programa comienza los cálculos internos calculando primeramente el tirante normal con el método de Newton – Raphson. Si por alguna razón no se encuentra convergencia a un valor aceptable, aparecerá una ventana indicándolo. Esta ventana aparecerá comúnmente en la sección circular cuando el gasto sea muy grande y el diámetro no sea capaz de transportarlo.

Una vez que se encuentra un valor de tirante normal válido, calculará los elementos geométricos, la velocidad, energía específica, tipo de flujo y número de Froude. Lo último será imprimir en cada caja de texto su valor correspondiente. Para realizar un nuevo cálculo es necesario presionar el botón limpiar y posterior a ello realizar la introducción correcta de los valores para el nuevo análisis.

El proceso de cómo opera este submódulo se ve reflejado de manera visual en la figura 4.3 “Representación de submódulo transición de flujo a través de diagrama de flujo”

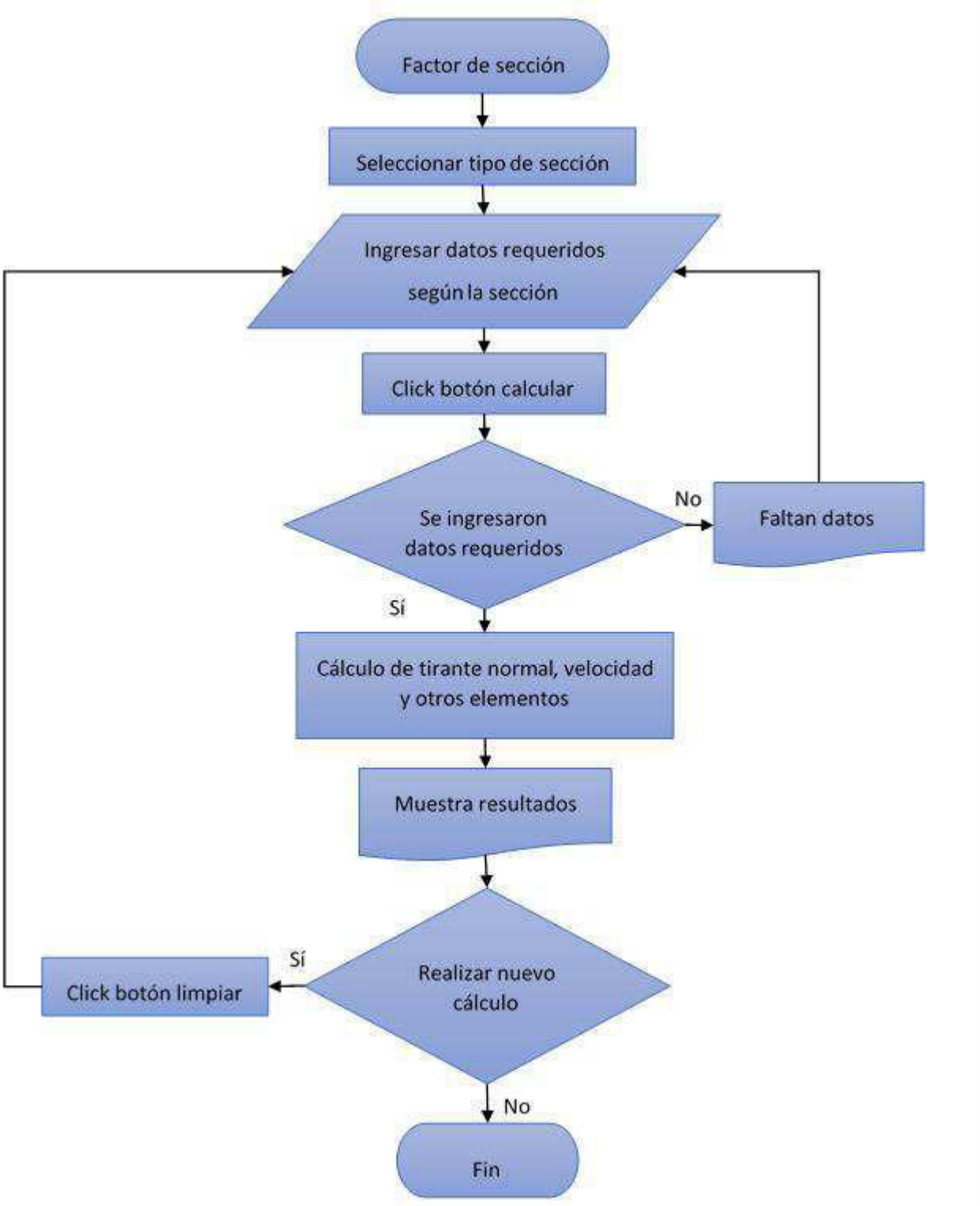


Figura 4.3 Representación del submódulo factor de sección a través de diagrama de flujo

Fuente: Elaboración propia

4.6 Sección de máxima eficiencia

En el subcapítulo de factor de sección se estableció que cuando hay alguna medida prestablecida u obligatoria, el diseño cumple, pero no de la manera más eficiente. Cuando nos referimos a eficiencia en los canales, se habla de que la sección del canal es capaz de transportar cierto gasto a través del menor perímetro mojado posible. Por ejemplo, proponiendo un canal rectangular tendremos 7.85 metros de perímetro mojado para la sección, sin embargo, si se elige una sección trapecial sólo se generan 7.55 metros perimetrales. Como conclusión el trapecio es de mayor eficiencia que el rectángulo, según la teoría de la máxima eficiencia hidráulica.

A pesar de que en el ejemplo dicho previamente solo hay 30 centímetros de diferencia, es obligatorio recordar que un canal puede extenderse kilómetros y esos pocos centímetros a la larga tienen mayor implicación hidráulica, por ende, influyen directamente en el costo de la obra. Es por ello que sea de gran importancia un análisis de este tipo, considerando que los diseños sean funcionales, eficientes y económicos.

Hasta el momento cada módulo y submódulo nos ha dado la opción de analizar las cinco secciones de interés (rectángulo, trapecio, triángulo, círculo y parábola). La excepción llega en la sección de máxima eficiencia debido a que, este tema sólo puede considerar las secciones rectangular y trapecial. En ambos casos la base del canal tiene que estar en función del tirante normal, el cuál es la variable a conocer mediante métodos iterativos.

Para el rectángulo despejando la base del área y al sustituir en el perímetro tenemos:

$$b = \frac{A}{y} \qquad (4\text{--}8)$$

$$P = \frac{A}{y} + 2y \qquad (4\text{--}9)$$

Al derivar el perímetro con respecto al tirante normal nos queda:

$$\frac{dP}{dy} = -\frac{A}{y^2} + 2 \qquad (4\text{--}10)$$

Suponiendo este diferencial igual a cero podemos despejar el área:

$$A = 2y^2 \qquad (4\text{--}11)$$

Al sustituir 4-11 en 4-8 encontramos la relación mostrada en 4-12.

$$b = \frac{2y^2}{y} = 2y \qquad (4\text{--}12)$$

Sustituyendo 4-12 en la ecuación del perímetro se obtiene:

$$P = 2y + 2y = 4y \qquad (4\text{--}13)$$

En la sección trapecial, al igual que en la rectangular, despejamos la base del área, sustituyendo en el perímetro como se muestra a continuación:

$$b = \frac{A}{y} - 0.5(z_1 + z_2)y \qquad (4\text{--}14)$$

$$P = \frac{A}{y} - 0.5(z_1 + z_2)y + y\left(\sqrt{1 + z_1^2} + \sqrt{1 + z_2^2}\right) \qquad (4\text{--}15)$$

Procedemos realizando la derivada del perímetro con respecto al tirante, obteniendo:

$$\frac{dP}{dy} = -\frac{A}{y^2} - 0.5(z_1 + z_2) + \left(\sqrt{1 + z_1^2} + \sqrt{1 + z_2^2}\right) \qquad (4\text{--}16)$$

Igualando 4-16 a cero y despejando el área resulta:

$$A = \left(-0.5(z_1 + z_2) + \left(\sqrt{1 + z_1^2} + \sqrt{1 + z_2^2}\right)\right)y^2 \qquad (4\text{--}17)$$

Al sustituir 4-17 en 4-12 encontramos la base en función del tirante:

$$b = \frac{\left(-0.5(z_1 + z_2) + \left(\sqrt{1 + z_1^2} + \sqrt{1 + z_2^2}\right)\right)y^2}{y} - 0.5(z_1 + z_2)y \qquad (4\text{--}18)$$

$$b = -(z_1 + z_2)y + y\left(\sqrt{1 + z_1^2} + \sqrt{1 + z_2^2}\right)$$

Por último, el perímetro nos resulta de la siguiente forma:

$$P = -(z_1 + z_2)y + 2y\left(\sqrt{1 + z_1^2} + \sqrt{1 + z_2^2}\right) \qquad (4\text{--}19)$$

Ambas secciones han de cumplir la ecuación 4-7, sustituyendo las áreas y perímetros que se han deducido en función del tirante.

4.6.1 Ejemplo ilustrativo submódulo sección de máximo eficiencia

Problema 6.- Un canal artificial se ha de construir en un fraccionamiento para tener mayor control en la circulación del agua. Se tiene registro de un gasto de 36 m³/s en temporada de lluvias y en el levantamiento topográfico se encontró que la pendiente es aproximadamente igual a 0.0035. Se requieren dos propuestas, una sección rectangular y una trapecial con taludes de 0.95. Determine cuál sección es más eficiente, si el material de recubrimiento del canal es de concreto pulido (n=0.013).

Para el rectángulo obtenemos la siguiente ecuación:

$$(2y^2)\left(\frac{2y^2}{4y}\right)^{\frac{2}{3}} = \frac{\left(36\frac{m^3}{s}\right)(0.013)}{\sqrt{0.0035}}$$

Al solucionar con la calculadora Ti – nspire cx CAS obtenemos un tirante de 1.991593 metros, este valor lo hemos de sustituir en la fórmula del perímetro:

$$P = 4y = 4(1.991593\ \text{m}) = \ 7.966372\ m$$

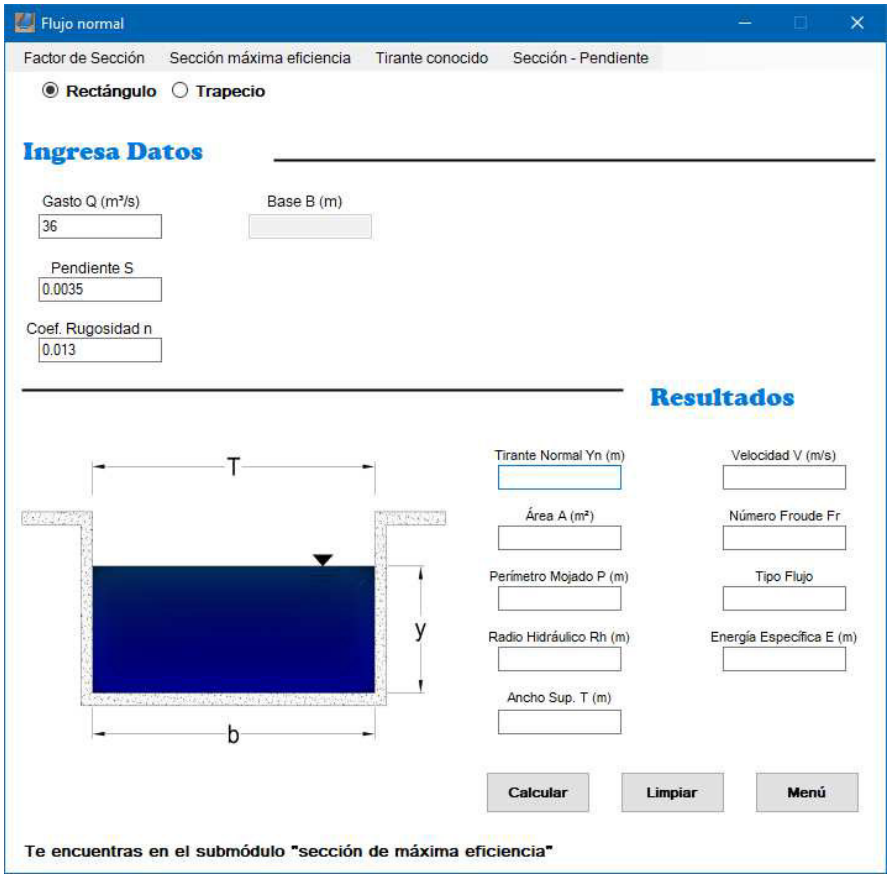
Por otro lado, el trapecio da solución al tirante a través de esta ecuación:

$$\left(-0.5(0.95+0.95) + \left(\sqrt{1+0.95^2} + \sqrt{1+0.95^2}\right)\right)y^2\left(\frac{\left(-0.5(0.95+0.95) + \left(\sqrt{1+0.95^2} + \sqrt{1+0.95^2}\right)\right)y^2}{-(0.95+0.95)y + 2y\left(\sqrt{1+0.95^2} + \sqrt{1+0.95^2}\right)}\right)^{\frac{2}{3}} = \frac{\left(36\frac{m^3}{s}\right)(0.013)}{\sqrt{0.0035}}$$

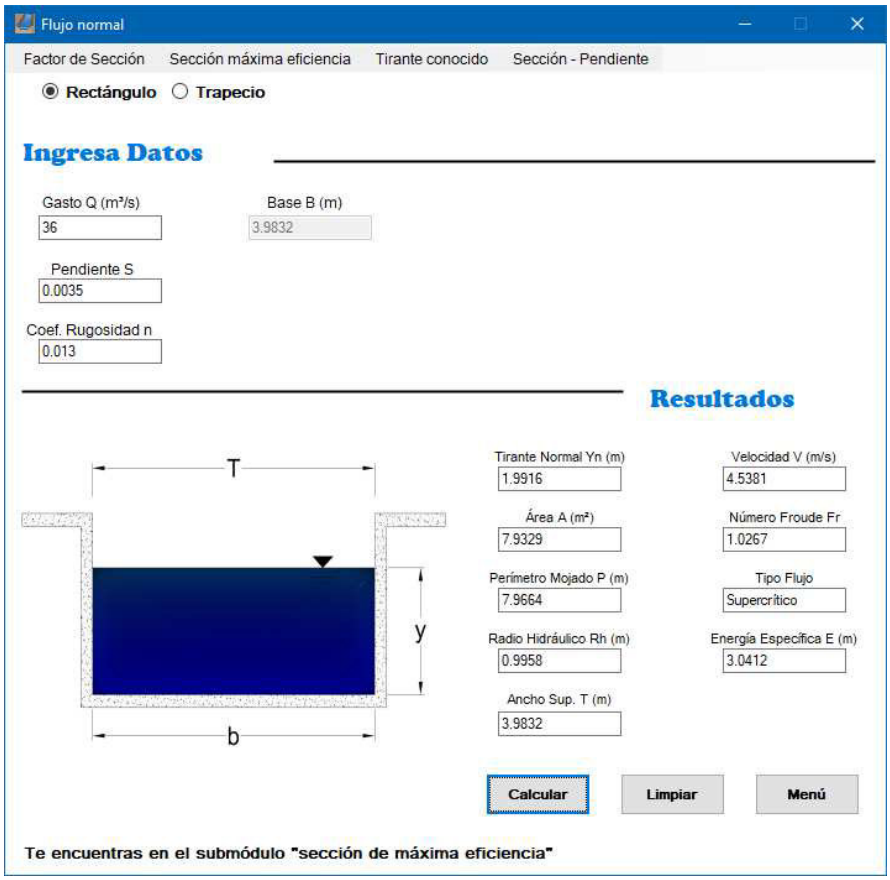
Nuevamente ayudados con la calculadora programable llegamos al valor de 2.068147 metros y un perímetro:

$$P = \left(-(0.95+0.95)(2.068147\ m) + 2(2.068147\ m)\left(\sqrt{1+0.95^2} + \sqrt{1+0.95^2}\right)\right) = 7.480996\ m$$

En comparativa, de manera manual llegamos a la conclusión de que la propuesta del trapecio es más efectiva debido a que tiene un perímetro menor. Con el uso de HYDROGECA esperamos valores muy similares.



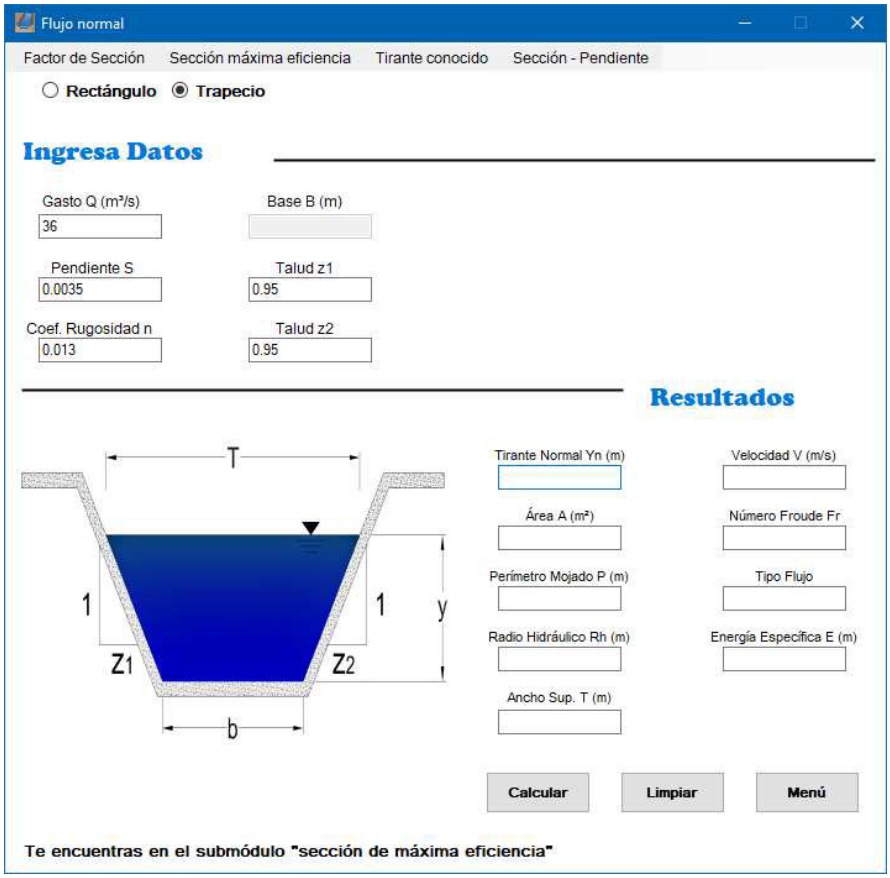
a)



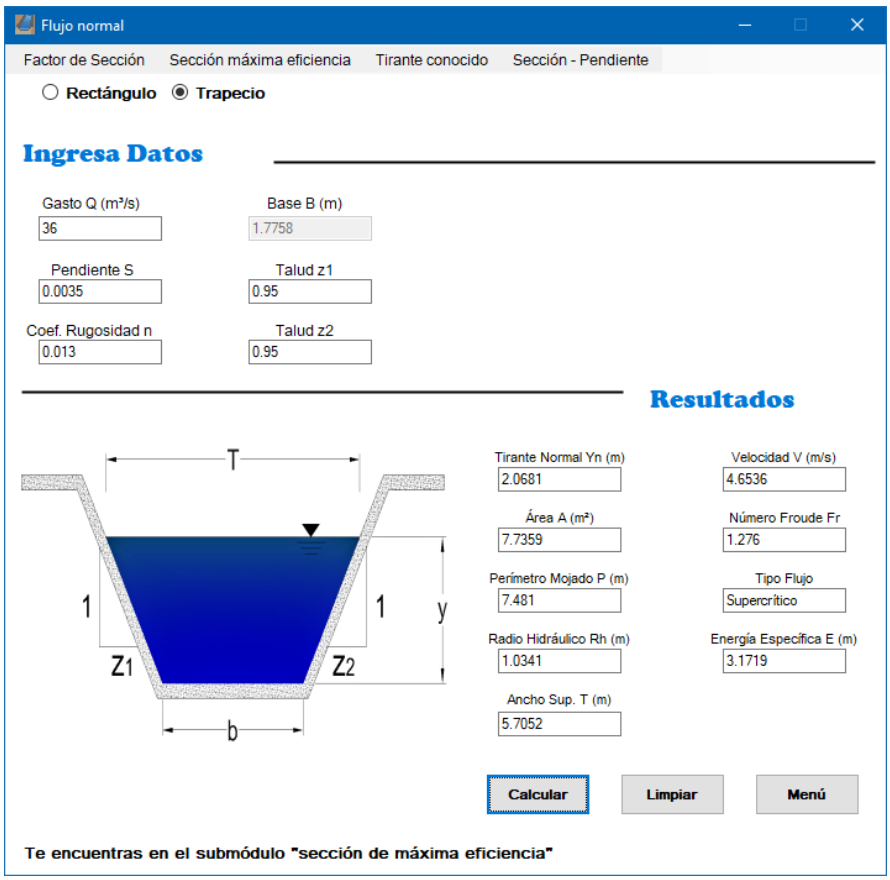
b)

Figura 4.4 Solución del problema 6 (Rectángulo) con HYDROGECA

Fuente: Programa HYDROGECA



a)



b)

Figura 4.5 Solución del problema 6 (Trapecio) con HYDROGECA

Fuente: Programa HYDROGECA

La vista es igual al submódulo “factor de sección”, la principal diferencia es que no ingresamos un valor de base. En cuanto a los resultados, tanto los manuales como los arrojados por HYDROGECA son iguales a excepción de la cantidad de decimales. Además, es necesario destacar que el programa nos muestra el valor de la base, la cual si recordamos está en función del tirante normal.

4.6.2 Solución de problemas

Primero hemos de elegir el tipo de sección de interés, recordemos que para este submódulo sólo hay dos opciones. Para el rectángulo es obligatorio ingresar gasto, pendiente y coeficiente de rugosidad. Si optamos por la sección trapecial es necesario los datos que requiere un rectángulo y añadir un valor de talud, estos pueden ser iguales o de distinta relación.

Se presiona el botón calcular y en dado caso que un dato no fue incluido, saltará una ventana con la frase “Faltan datos para realizar el cálculo”. Aunque esto no genera mayor problema, sólo se da aceptar a la ventana y procedes a ingresar el valor correspondiente.

Una vez que presiones calcular y no salga la ventana, el programa toma los datos y primeramente busca obtener un tirante normal a través del método iterativo Newton – Raphson. Si después de una cantidad de iteraciones el programa no converge a una solución viable, se informa con una ventana que dice “Es altamente probable que el valor del tirante hidráulico no sea el correcto”. En el caso de un resultado aceptable, se procede a calcular los demás valores como lo es área, perímetro, base, velocidad, entre otros.

El último punto es imprimir en pantalla cada valor en su respectiva caja de texto. Se puede generar otro cálculo si se presiona el botón limpiar y se repite de forma correcta el ingreso de valores.

La figura 4.6 plasma a través de un diagrama de flujo la descripción de cómo trabaja este submódulo.

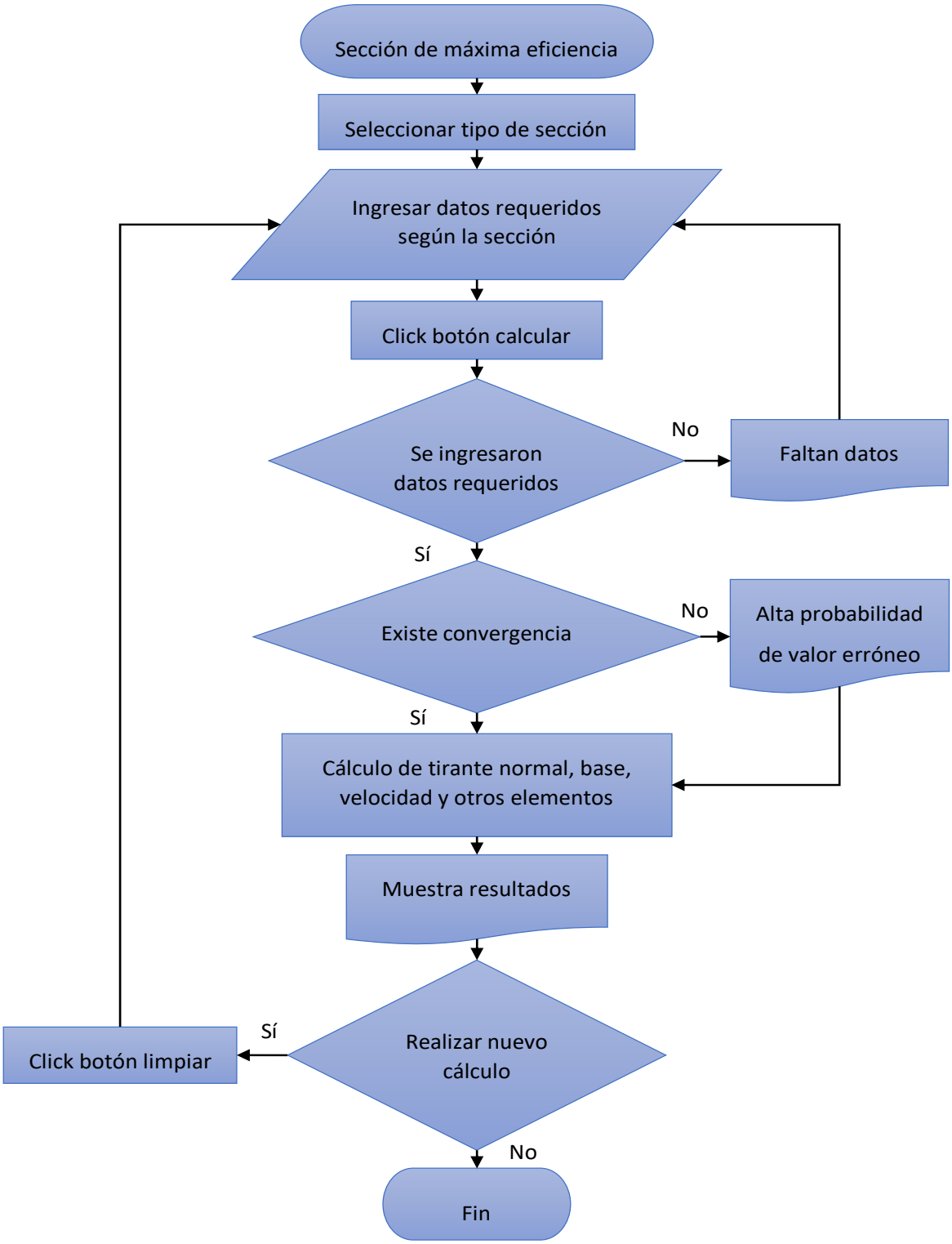


Figura 4.6 Representación del submódulo sección de máxima eficiencia a través de diagrama de flujo

Fuente: Elaboración propia

4.7 Tirante conocido

En este submódulo se ha de hacer la consideración de que se conoce un tirante o se propone el tirante y lo que se busca es otro elemento, ya sea gasto, pendiente, rugosidad o velocidad. Para ello, el programa ha de buscar llegar al resultado mediante una solución analítica contemplando 3 ecuaciones en específico (ecuaciones 4-3, 4-5, 4-7).

El uso de HYDROGECA con este submódulo nos ha de ser útil cuando queramos conocer que el coeficiente de rugosidad es efectivo para cierta situación de diseño, si se desea juagar con las pendientes y ver cuál nos ayudará a satisfacer la necesidad de transportar un fluido o también ver qué gasto ha de ser capaz de pasar por el canal bajo ciertas condiciones.

Puede parecernos demasiado sencillo en comparación con los submódulos anteriores que involucran métodos numéricos, sin embargo, nos ha de ser útil en cálculos sencillos evitando que tengamos que hacer despejes de fórmulas.

4.7.1 Combinaciones existentes

A pesar de que existen 5 secciones distintas, comparten las mismas combinaciones de elementos a encontrar. Por ejemplo, supongamos la combinación para un rectángulo donde se conoce el gasto, la pendiente, la base y el tirante. Esta misma combinación existe para cada sección, como la trapecial cuya diferencia será que se requiera también ingresar taludes. Pero todas siguen el mismo principio. Son 5 combinaciones posibles para cada sección, las cuales se ilustran en la imagen 4.7.

Datos conocidos (combinación 1)		
Gasto	Pendiente	Tirante

Datos conocidos (combinación 2)		
Gasto	Rugosidad	Tirante

Datos conocidos (combinación 3)		
Pendiente	Rugosidad	Tirante

Datos conocidos (combinación 4)		
Pendiente	Velocidad	Tirante

Datos conocidos (combinación 5)		
Rugosidad	Velocidad	Tirante

Figura 4.7 Conjunto de combinaciones posibles en submódulo tirante conocido

Fuente: Elaboración propia

4.7.2 Ejemplo ilustrativo submódulo tirante conocido

Problema 7.- En un canal de sección triangular se registró una velocidad de 2.2655 m/s y se ha medido un tirante de 0.79 metros. ¿Cuál es el gasto que se transporta la sección y su pendiente si se sabe que su coeficiente de rugosidad es de 0.015 y los taludes tiene una relación de 1.15?

Primeramente, podemos conocer los valores del área hidráulica y perímetro mojado, estos a su vez nos ayudarán a llegar al valor del radio hidráulico, lo cual se muestra en las siguientes ecuaciones:

$$A = 0.5(z1 + z2)y^2 = 0.5(1.15 + 1.15)(0.79m)^2 = 0.7177\ m^2$$

$$P = y\left(\sqrt{1 + z_1^2} + \sqrt{1 + z_2^2}\right) = 0.79\ m\left(\sqrt{1 + 1.15^2} + \sqrt{1 + 1.15^2}\right) = 2.4079\ m$$

$$R_h = \frac{A}{P} = \frac{0.7177\ m^2}{2.4079\ m} = 0.2981\ m$$

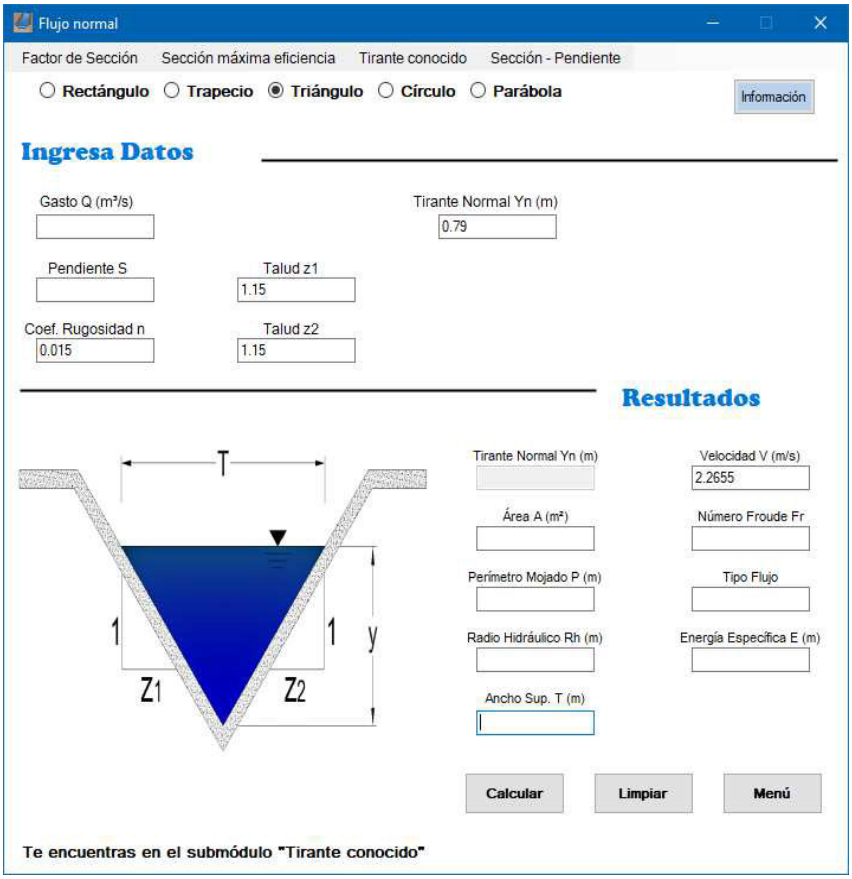
Conocida la velocidad y el radio se sustituyen en la velocidad media de Manning y así se despeja la pendiente:

$$V = \frac{1}{n}R_h^{\frac{2}{3}}S^{\frac{1}{2}};\ S = \left(\frac{Vn}{R_h^{\frac{2}{3}}}\right)^2 = \left(\frac{2.2655\frac{m}{s}\ (0.015)}{(0.2981\ m)^{\frac{2}{3}}}\right)^2 = 0.0058$$

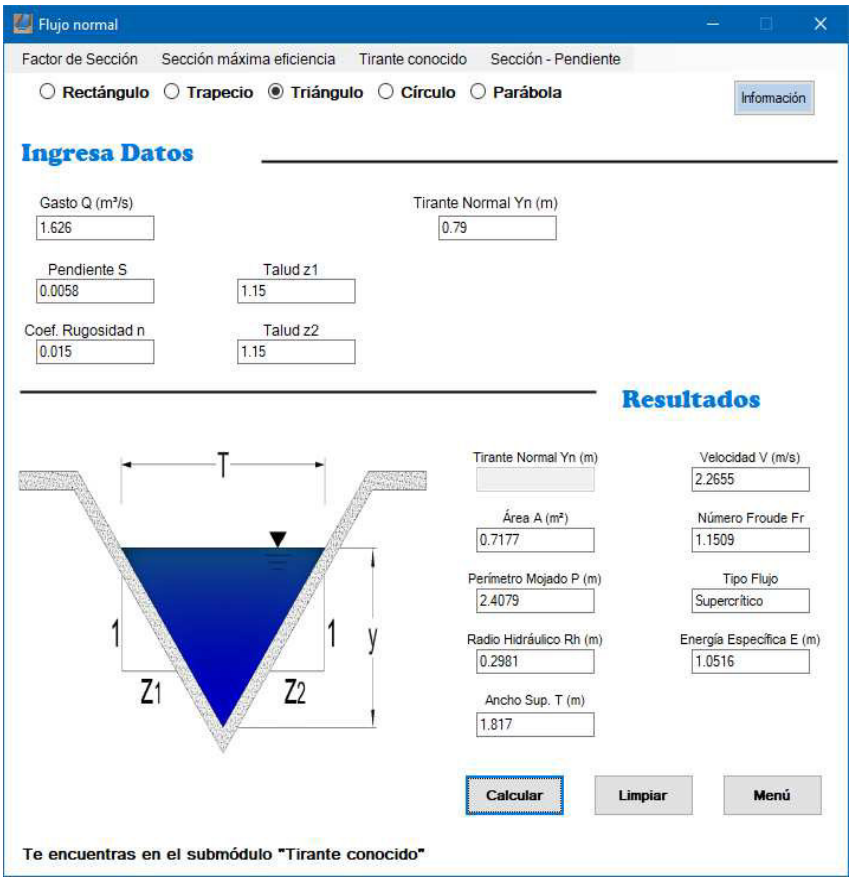
Por último, necesitamos encontrar el gasto, obtenido como se muestra a continuación:

$$AR_h^{\frac{2}{3}} = \frac{Q\ n}{s^{\frac{1}{2}}};\ Q = \frac{AR_h^{\frac{2}{3}}S^{\frac{1}{2}}}{n} = \frac{(0.7177\ m^2)(0.2981\ m)^{\frac{2}{3}}(0.0058)^{\frac{1}{2}}}{0.015} = 1.6260\frac{m^3}{s}$$

Como se mencionó, para este submódulo los cálculos son analíticos y no se ha tenido que recurrir a un método numérico o en su defecto, a la calculadora programable como se realizó en los submódulos anteriores o inclusive a otros módulos como el tirante crítico.



a)



b)

Figura 4.8 Solución del problema 7 con HYDROGECA

Fuente: Programa HYDROGECA

En la pantalla del programa vemos que hay un apartado de ingreso de datos y otro de resultados. Se podría pensar que no se debe ingresar algún valor en el área de resultados, sin embargo, este módulo tiene esa capacidad. La figura 4.8 a) demuestra que es posible, correspondiendo a la combinación número 5 mostrada en la figura 4.7.

En cuanto a los resultados, es satisfactorio lo presentado en la caja de texto y coincide con los resultados generados manualmente.

4.7.3 Solución de problemas

Al seleccionar tirante conocido lo primero a realizar es la selección del tipo de sección y posterior a esto, ingresar los datos conocidos (es recomendable guiarse con la imagen donde se presentan las posibles combinaciones). Cuando se presione el botón calcular, internamente se buscará si la combinación de datos ingresados existe. Si fuera el caso de que no hay coincidencia, el programa simplemente no hace nada. Caso contrario, HIDROGECA ha llegado a una coincidencia de caso y almacenará los valores para continuar con el cálculo.

Hay que recordar que, todas las soluciones serán analíticas e internamente resolverá ecuaciones para conocer ya sea un gasto, una pendiente, una velocidad o un coeficiente de rugosidad. Al finalizar cada uno de los cálculos, se comienzan a imprimir en cada caja de texto el resultado correspondiente.

Si se requiere realizar otro análisis, es necesario que se presione el botón limpiar para que elimine tanto resultados como datos ingresados. La forma en que opera este submódulo es mostrada en el diagrama que se muestra a continuación.

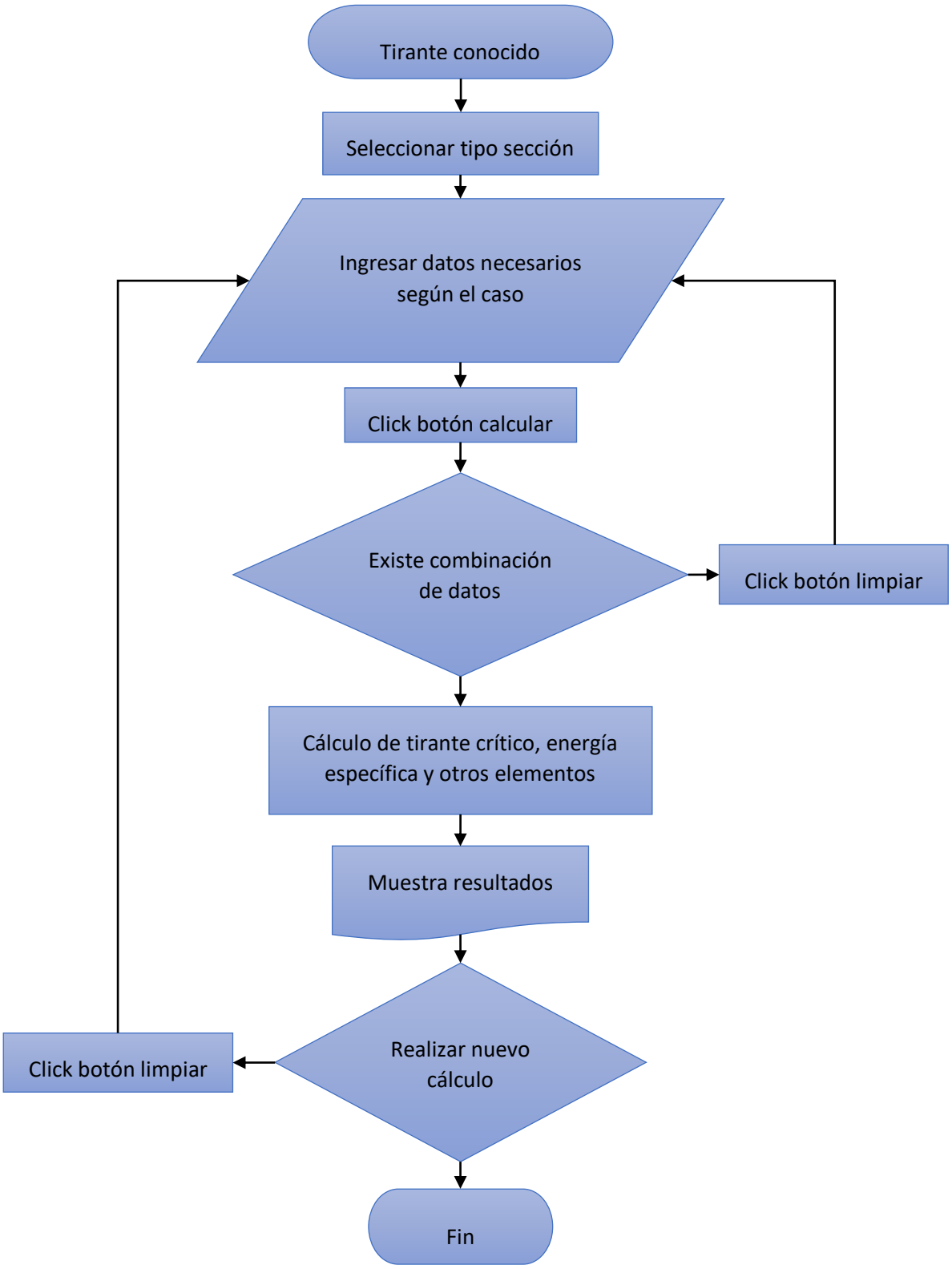


Figura 4.9 Representación del submódulo tirante conocido a través de diagrama de flujo

Fuente: Elaboración propia

4.8 Sección – Pendiente

En este submódulo se deben considerar dos ecuaciones ya vistas en el capítulo actual. Tanto la ecuación de continuidad como la velocidad de Manning son de importancia.

Al sustituir la velocidad en continuidad tenemos la ecuación 4-6:

$$Q = \frac{1}{n} R_h^{\frac{2}{3}} S^{\frac{1}{2}} A$$

Si analizamos la ecuación, podemos juntar en un nuevo término los valores que dependen de la geometría y el material del canal. Esta variable designada como factor de conducción para el cálculo del flujo uniforme en corrientes naturales, denominado por la letra K, como se muestra en la siguiente ecuación:

$$K = \frac{A R_h^{\frac{2}{3}}}{n} \qquad (4\text{--}20)$$

Sustituyendo K en 4-6 obtenemos:

$$Q = K S^{\frac{1}{2}} \qquad (4\text{--}21)$$

Pero, ¿Qué importancia tiene este tema en la hidráulica? Bueno, este método es aplicable en el conocimiento de cauces de ríos. Aunque en los primeros capítulos de este trabajo se hizo mención de que nos enfocaríamos solamente al estudio de canales artificiales, para este caso se hace una excepción.

La aplicación de este método según Sotelo (2002) implica que se cumplan ciertas condiciones y se respeten algunas restricciones. A continuación, se hace mención de esto.

Se selecciona una parte más o menos recta de longitud no menor a 6 veces el ancho del cauce y sección casi uniforme para dividir en 10 secciones aproximadamente iguales. En la selección es importante tomar en cuenta los siguientes puntos:

- Cerciorarse de que haya marcas dejadas por el agua sobre los márgenes e identificar la línea de aguas máximas, definida por las huellas que dejó el pasó de la avenida. Dichas huellas deben ser abundantes y precisas; de lo contrario es cuestionable el uso del método.

- Tramos lo más recto y uniforme posible sin tener contra pendiente. Hay que evitar que existan cambios importantes en la forma de la sección del cauce y de la pendiente, debido a la incertidumbre en la consideración de la pérdida de energía entre secciones.
- Se supondrá que el total del área en cada sección del tramo es efectivo en la conducción del agua, por lo tanto, deben evitarse tramos en que las condiciones provoquen una distribución desbalanceada del flujo.
- Han de excluirse caídas de agua y rápidas, al igual lo sitios donde se presenten desbordamientos.
- En tramos demasiado largos la condición de uniformidad no se han de mantener. Sin embargo, la longitud debe ser suficiente para desarrollar un desnivel apreciable en la superficie del agua entre las dos secciones y una determinación satisfactoria de la pendiente general del cauce. La precisión del método mejora en la medida que la longitud del tramo aumenta.

Un buen levantamiento topográfico y una minuciosa estimación del coeficiente de Manning en cada tramo nos llevará a un análisis más preciso.

4.8.1 Ejemplo ilustrativo submódulo sección – pendiente

Problema 8. - En la siguiente tabla se muestran los datos recabados de un cauce natural. Considere un desnivel de 27 centímetros en una longitud de 235 metros.

Dato	Sección aguas arriba	Sección aguas abajo
Área hidráulica	48 m²	53 m²
Perímetro mojado	22 m	24 m
Coef. Coriolis	1.16	1.13
Coef. Rugosidad	0.019	0.019

Tabla 4.2 Datos correspondientes al problema 8

Parar los cálculos cuando la diferencia de Q_0 y Q_1 sea menor a 6 m³/s

Lo primero a que hacemos es calcular K_A (sección aguas arriba) y K_B (sección aguas abajo)

$$K_A = \frac{A_A\, R_{hA}^{\frac{2}{3}}}{n} = \frac{(48\ m^2)\left(\frac{48\ m^2}{22\ m}\right)^{\frac{2}{3}}}{0.019} = 4248.78$$

$$K_B = \frac{A_B\, R_{hB}^{\frac{2}{3}}}{n} = \frac{(53\ m^2)\left(\frac{53\ m^2}{24\ m}\right)^{\frac{2}{3}}}{0.019} = 4730.41$$

El valor de $\overline{K}$ es el promedio geométrico de los valores que obtuvimos

$$\overline{K} = \sqrt{K_A K_B} = \sqrt{(4248.78)(4730.41)} = 4483.66$$

La pendiente se conoce de la división del desnivel entre la longitud

$$S_0 = \frac{\Delta y}{\Delta x} = \frac{0.27\ m}{235\ m} = 0.001149$$

Con estos datos conocemos el gasto Q_{0-1}

$$Q_{0-1} = \overline{K}\sqrt{S_0} = 4483.66\sqrt{0.001149} = 151.978\frac{m^3}{s}$$

El gasto anterior nos permite conocer la velocidad en las dos secciones

$$V_A = \frac{Q_{0-1}}{A_A} = \frac{151.978\frac{m^3}{s}}{48\ m^2} = 3.1662\frac{m}{s}$$

$$\alpha\frac{V_A^2}{2g} = 1.16\frac{\left(3.1662\frac{m}{s}\right)^2}{2\left(9.81\frac{m}{s^2}\right)} = 0.5927$$

$$V_B = \frac{Q_{0-1}}{A_B} = \frac{151.978\frac{m^3}{s}}{53\ m^2} = 2.8675\frac{m}{s}$$

$$\alpha\frac{V_B^2}{2g} = 1.16\frac{\left(2.8675\frac{m}{s}\right)^2}{2\left(9.81\frac{m}{s^2}\right)} = 0.4736$$

El siguiente paso es calcularS_{1-1}, pero antes es necesario ingresar a la tabla que se muestra debajo y elegir un valor de K_t según el caso en el que caemos.

k_t	Caso
0.5	$(A_1 < A_2; V_1 > V_2)$ *Expansión brusca*
0.7	$(A_1 < A_2; V_1 > V_2)$ *Expansión gradual*
1.0	$(A_1 = A_2; V_1 = V_2)$ *Flujo uniforme*
1.2	$(A_1 > A_2; V_1 < V_2)$ *Contracción gradual*
1.5	$(A_1 > A_2; V_1 < V_2)$ *Contracción brusca*

Tabla 4.3 Valores para K_t según el caso en el que cae

Para este problema sabemos que A1 (A_A) es menor que A2 (A_B), por ende, nos delimitamos a que será un caso de expansión y para ser más exactos, expansión gradual. De este modo, podemos conocer la pendiente S_{1-1}

$$S_{1-1} = \frac{\Delta y + K_t\left(\alpha\frac{V_A^2}{2g} - \alpha\frac{V_B^2}{2g}\right)}{\Delta x} = \frac{0.27\ m + 0.7(0.5927 - 0.4736)}{235} = 0.001504$$

Para finalizar esta primera iteración del método calculamos Q_{1-1} y la diferencia ente Q_{0-1} y Q_{1-1}

$$Q_{1-1} = \overline{K}\sqrt{S_1} = 4483.66\sqrt{0.001504} = 173.87\frac{m^3}{s}$$

$$|Q_{0-1} - Q_{1-1}| = |151.978\frac{m^3}{s} - 173.87\frac{m^3}{s}| = 21.89\frac{m^3}{s}$$

Como la diferencia de gastos es mayor que 6 m³/s, el cálculo se vuelve a repetir. Para simplificar el documento, se agregarán las tablas de resultados de cada iteración que se genere hasta que obtengamos una diferencia menor a la que se menciona.

Solo hay que mencionar que, cada que inicie una nueva iteración, el valor de gasto inicial aumentará uno su subíndice. Esto quiere decir que para la iteración número dos, el gasto se identifica como Q_{0-2}. Este gasto se calcula promediando los gastos generados en la iteración anterior.

$$Q_{0-2} = \frac{Q_{0-1} + Q_{1-1}}{2} = \frac{151.978\frac{m^3}{s} + 173.87\frac{m^3}{s}}{2} = 162.9242\frac{m^3}{s}$$

Segunda iteración	
$\alpha \frac{V_A^2}{2g}$	0.6812
$\alpha \frac{V_B^2}{2g}$	0.5443
S_{1-2}	0.001557
Q_{1-2}	176.9055 m³/s
$\lvert Q_{0-2} - Q_{1-2} \rvert$	13.9813 m³/s

Tercera iteración	
Q_{0-3}	169.9149 m³/s
$\alpha \frac{V_A^2}{2g}$	0.7409
$\alpha \frac{V_B^2}{2g}$	0.5919
S_{1-3}	0.001592
Q_{1-3}	178.9251 m³/s
$\lvert Q_{0-3} - Q_{1-3} \rvert$	9.0102 m³/s

Cuarta iteración	
Q_{0-4}	174.42 m³/s
$\alpha \frac{V_A^2}{2g}$	0.7807
$\alpha \frac{V_B^2}{2g}$	0.6238
S_{1-4}	0.001616
Q_{1-1}	180.259 m³/s
$\lvert Q_{0-4} - Q_{1-4} \rvert$	5.8390 m³/s

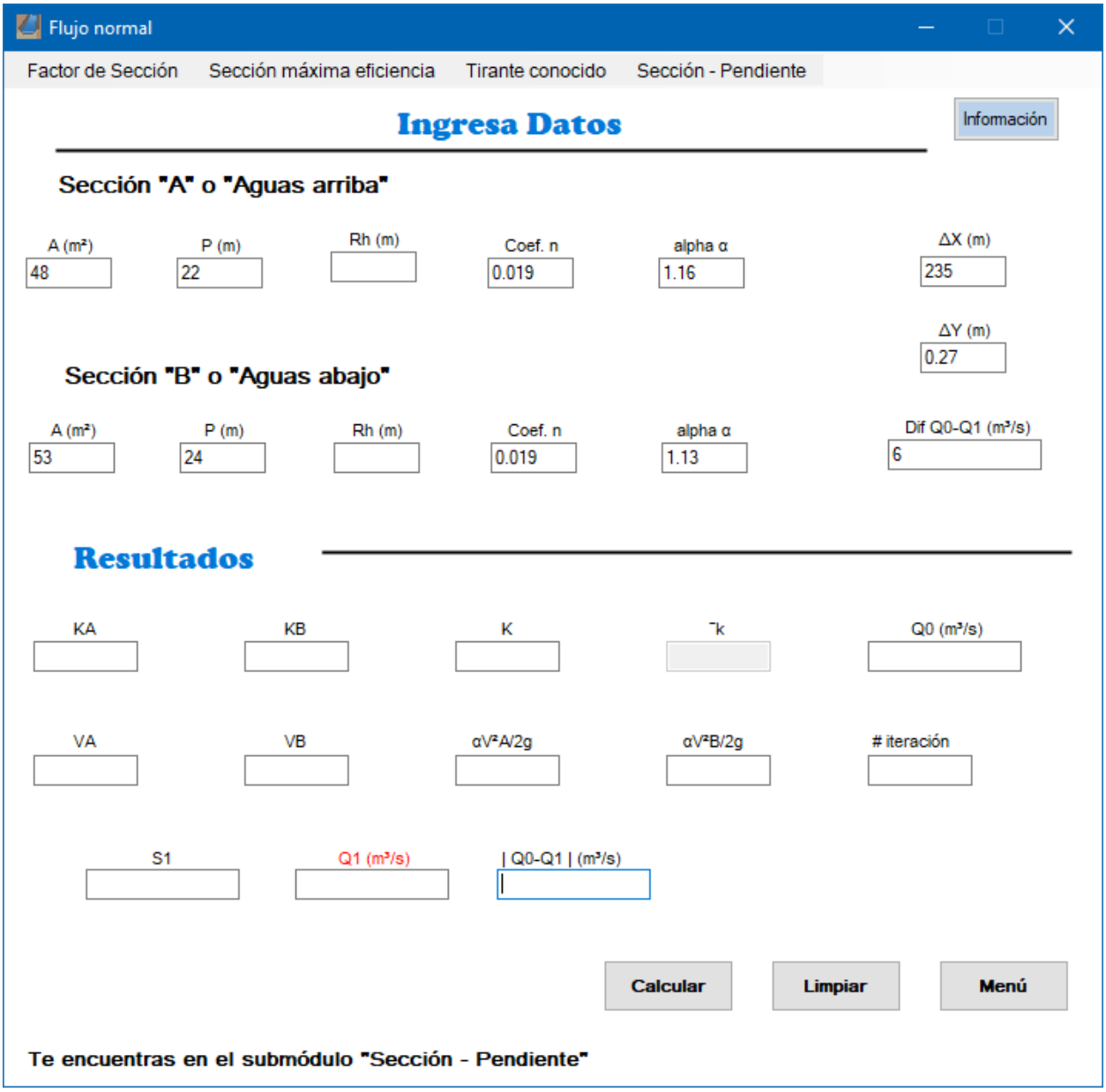


Figura 4.10 Ingreso de datos del problema 8 en HYDROGECA

Fuente: Programa HYDROGECA

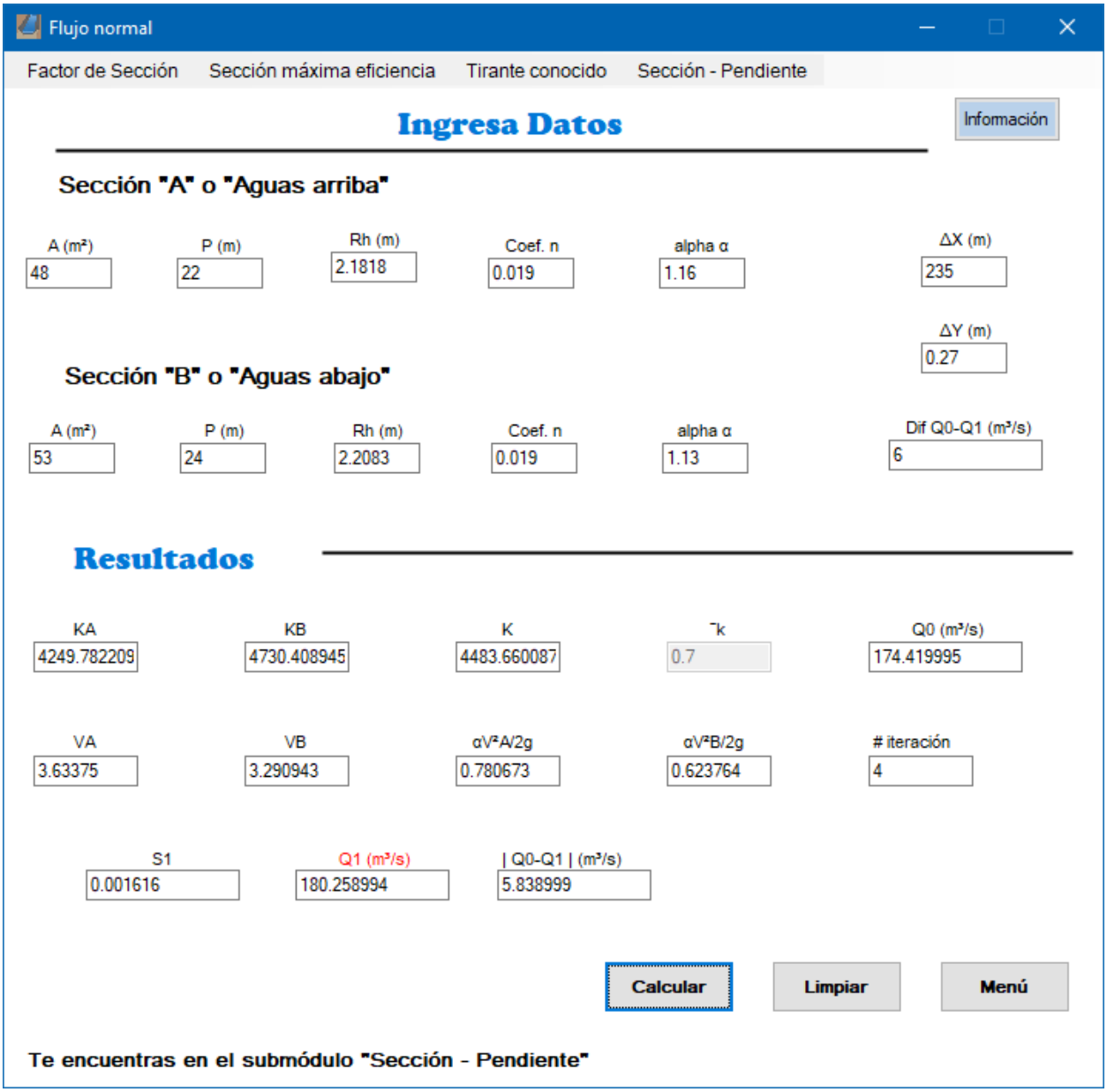


Figura 4.11 Resultados del problema 8 con HYDROGECA

Fuente: Programa HYDROGECA

Si comparamos los resultados que nos arrojó el programa con la tabla de la cuarta iteración se puede ver que los resultados son muy similares y, HYDROGECA nos muestra que en 4 iteraciones se alcanzó una diferencia menor a los 6 m³/s.

Quizá el único detalle que presenta este submódulo es que sólo imprime los valores obtenidos en la última iteración. Para el ejemplo actual no habría problema generar una hoja con los resultados de cada iteración, pero imaginemos que quieran una diferencia de 0.1 m³/s, esto nos llevaría a tener

14 iteraciones y eso en una hoja de resultados ya comienza a ser demasiados números. Es por esto que se ha decidido solamente mostrar la última iteración.

4.8.2 Solución de problemas

Para que se llegue a una solución, es necesario ingresar los siguientes datos para cada una de las secciones:

- Área hidráulica
- Perímetro mojado
- Coeficiente de rugosidad
- Coeficiente de Coriolis

En caso de que se conozca directamente el radio hidráulico, no se debe introducir el perímetro, ya que el programa lo calculará automáticamente. La longitud y el desnivel entre secciones es algo obligatorio para que se pueda realizar el cálculo al igual que se debe dar una diferencia entre gasto inicial y gasto final.

Internamente el programa comenzará a realizar los cálculos como se mostró el en problema 8, pero al momento de definir el primer valor de S_1, el programa abrirá una ventana en la que dejará elegir un K_t según en los posibles que casos que pueda caer tu análisis. Al seleccionar el K_t el programa continúa con las operaciones a través de un ciclo For hasta que se cumpla que la diferencia de gastos sea menor a la que se tecleó al introducir los datos conocidos.

Para finalizar el trabajo de este submódulo tenemos la impresión de los resultados, donde solamente se mostrará lo obtenido de la última iteración. Los datos más importantes serán, los gastos, la diferencia entre estos mismos y la cantidad de iteraciones.

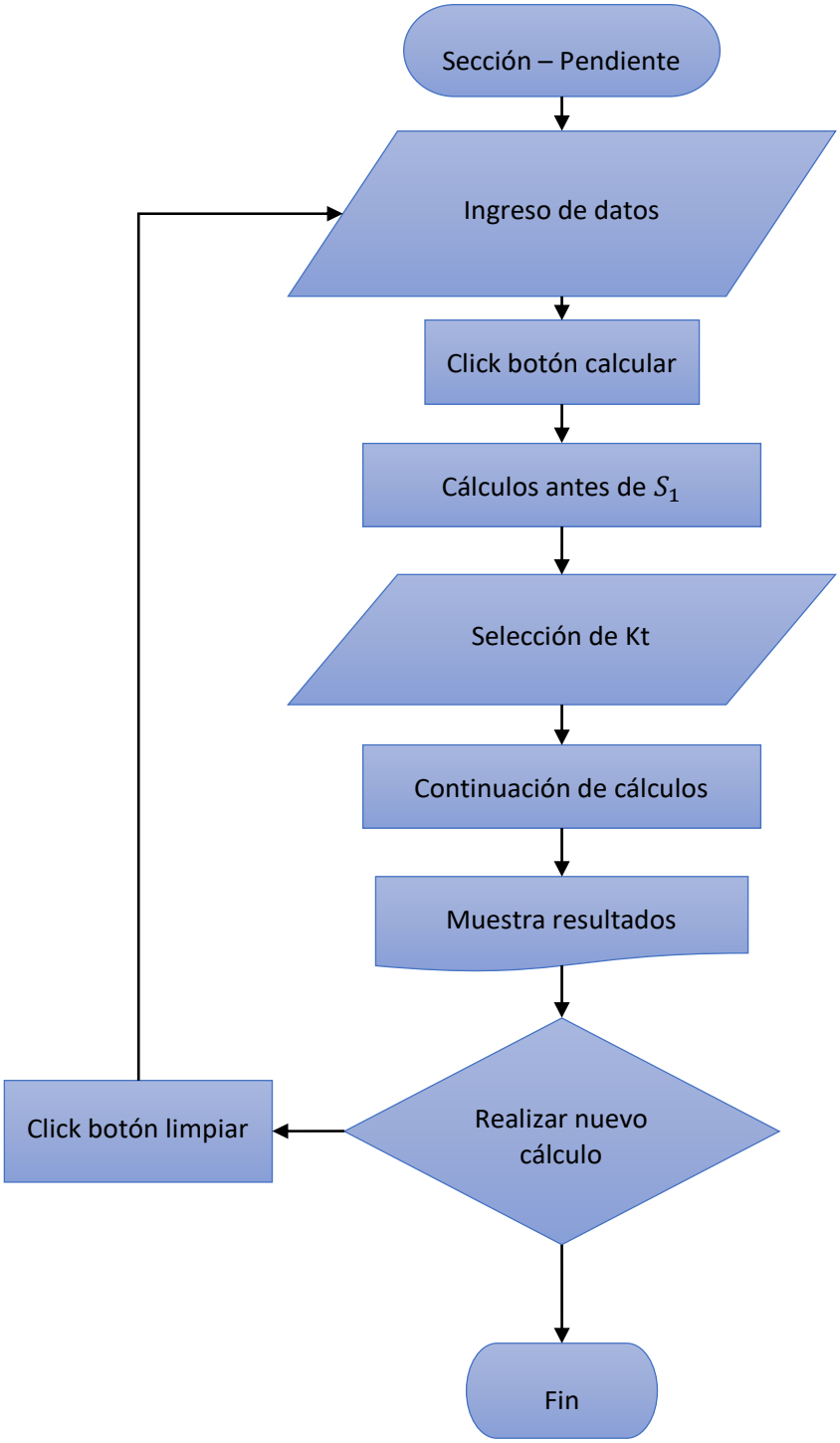


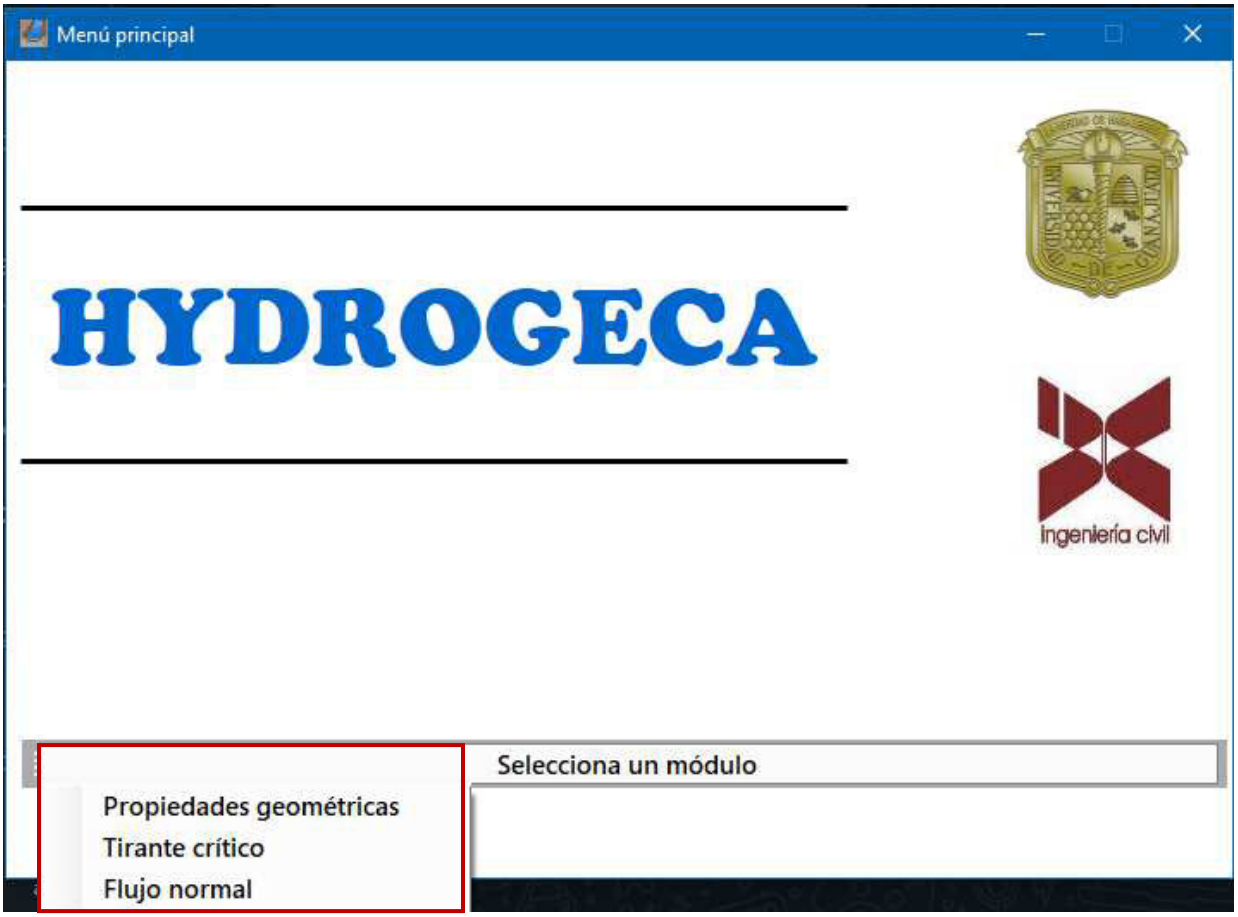
Figura 4.12 Representación del submódulo sección - pendiente a través de diagrama de flujo

Fuente: Elaboración propia

5 Capítulo Manual de usuario

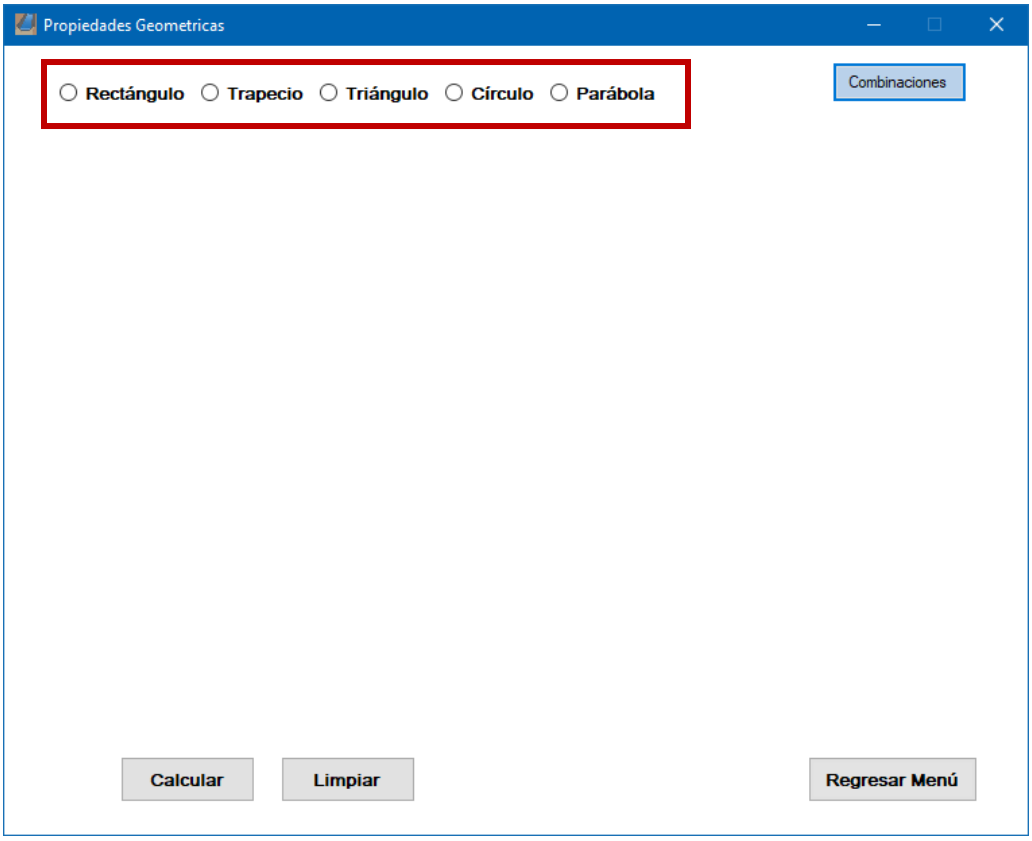
En este capítulo se desarrolló un manual de usuario que se integrará en el programa, para estos casos se llevaron a cabo algunos ejemplos de aplicación y algunas sugerencias y recomendaciones en el uso del programa, sobre todo para que sea de mayor facilidad en el entendimiento de la interfaz y sea un documento de referencia para los alumnos del programa educativo de ingeniería civil, el manual se incluye en la parte de los anexos. (aún no se adjunta al documento de la tesis).

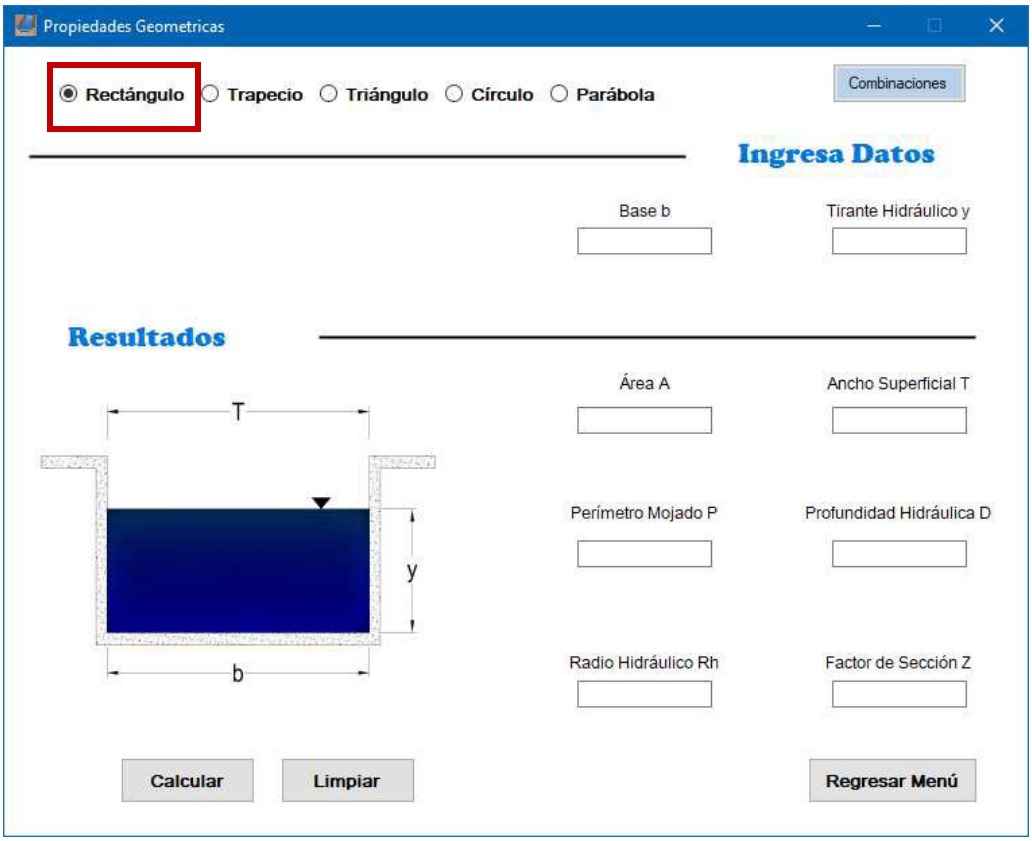
Al iniciar el programa HYDROGECA, el menú principal muestra tanto los logos de la carrera de ingeniería civil como el escudo de la Universidad de Guanajuato y el mismo nombre del programa. La barra gris con la leyenda “Selecciona un módulo” se presiona y despliega 3 opciones. El usuario ha de elegir una según sea su necesidad.

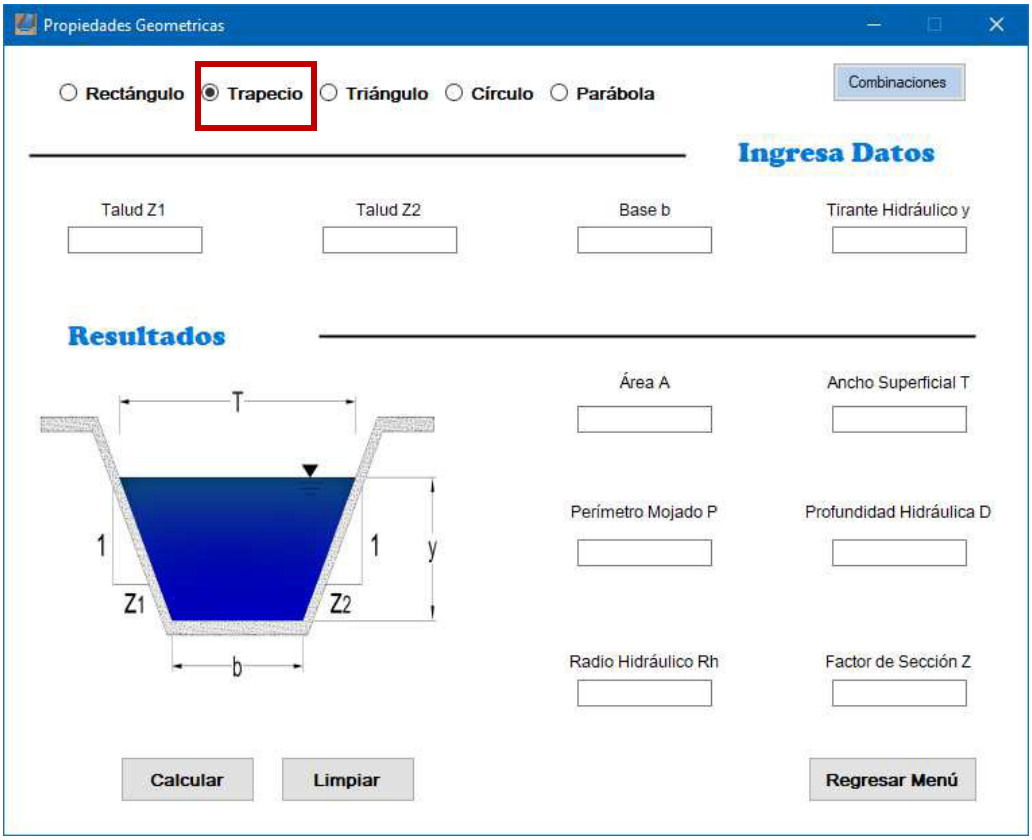


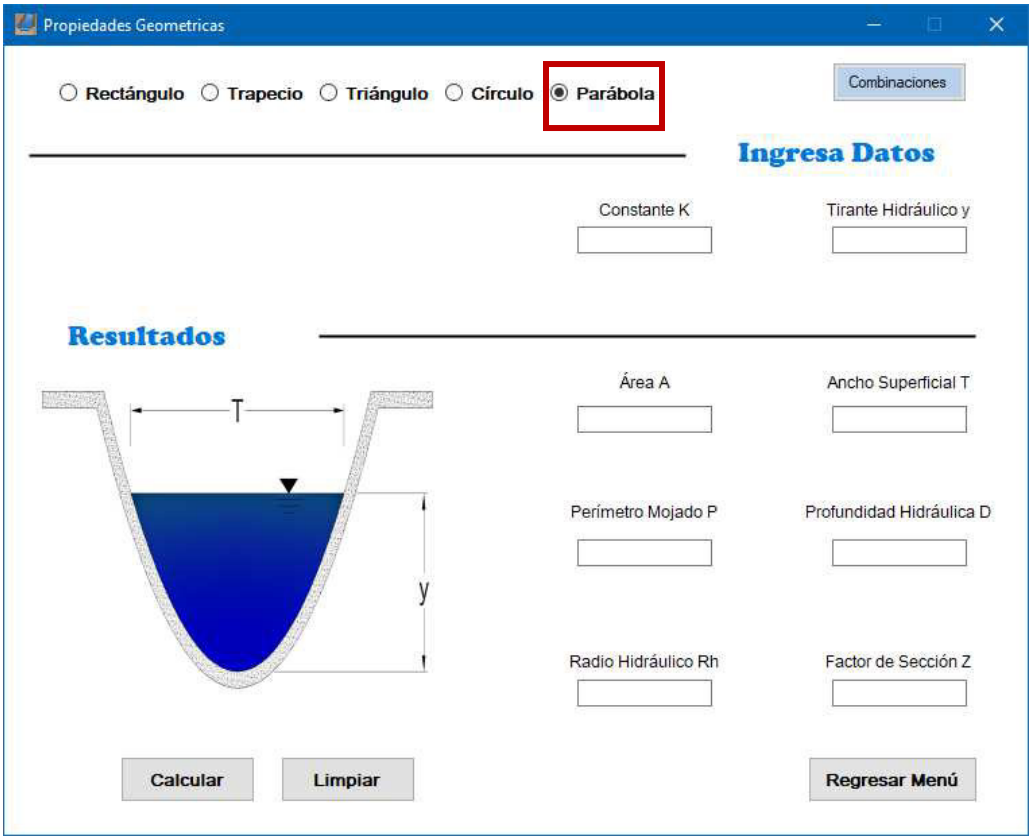
De haber seleccionado el módulo “Propiedades geométricas”, se presenta una pantalla en blanco con 5 opciones de sección para seleccionar. También se muestra los botones “Calcular”, “Limpiar”, “Regresar Menú” y “Combinaciones”. Los primeros dos botones mencionados no sirven hasta que se hayan ingresado valores en las cajas de texto.

Si se selecciona una sección, se muestra un dibujo de la sección y cajas de texto donde se ingresan los valores correspondientes a los elementos geométricos o donde se muestran los resultados después de que se realiza el cálculo.

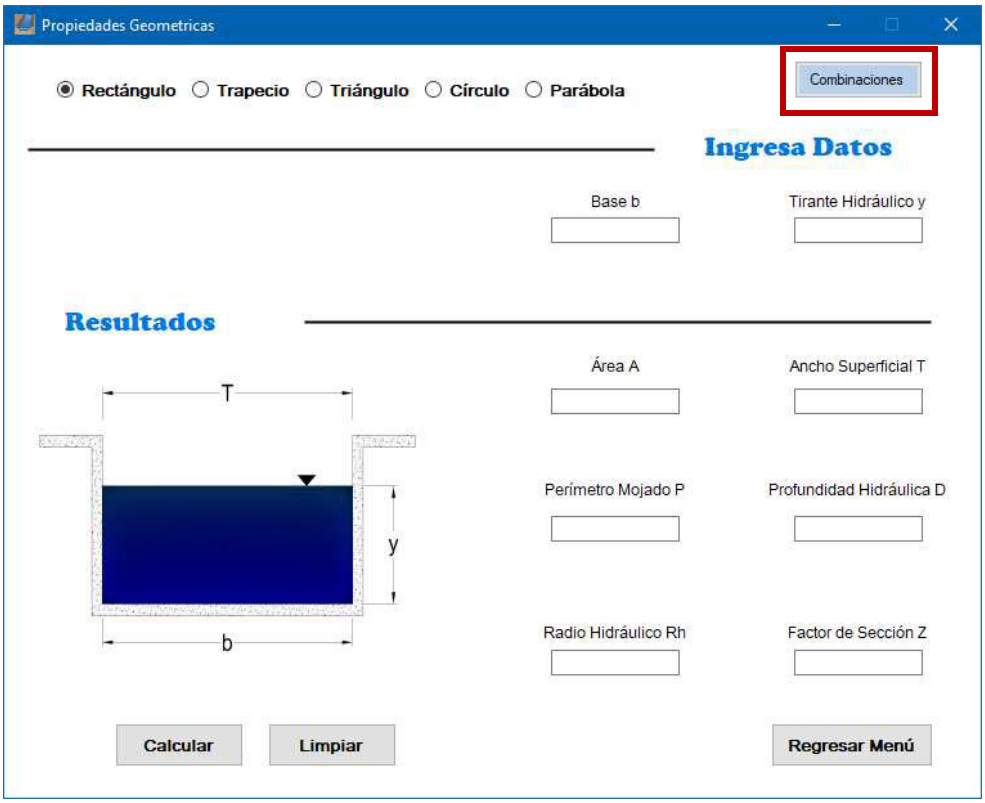

Propiedades Geometricas
Rectángulo
Trapecio
Triángulo
Círculo
Parábola
Combinaciones
Calcular
Limpiar
Regresar Menú

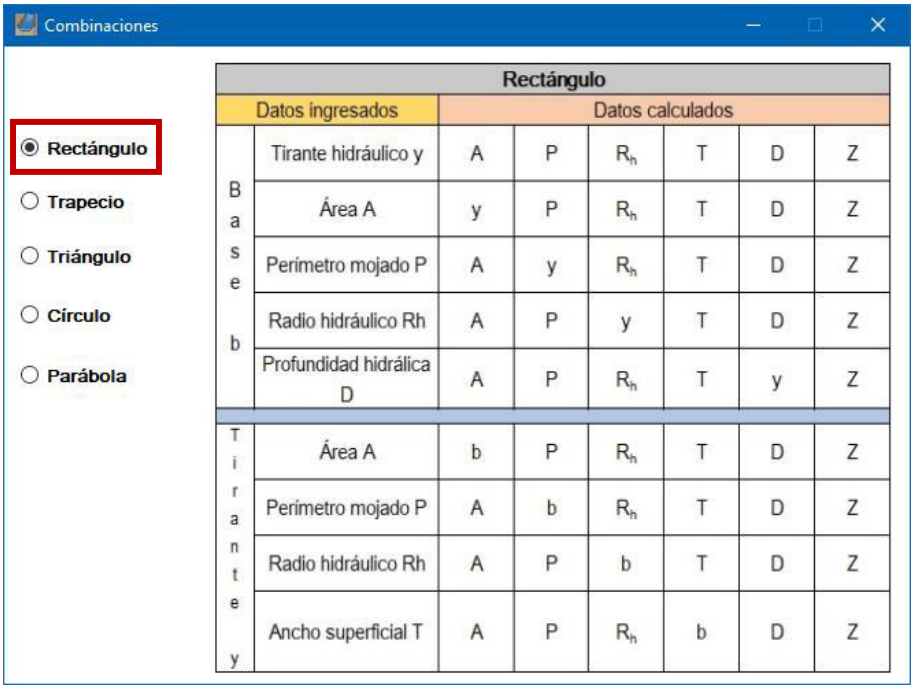

Propiedades Geometricas
Rectángulo
Trapecio
Triángulo
Círculo
Parábola
Combinaciones
Ingresa Datos
Base b
Tirante Hidráulico y
Resultados
T
y
b
Área A
Ancho Superficial T
Perímetro Mojado P
Profundidad Hidráulica D
Radio Hidráulico Rh
Factor de Sección Z
Calcular
Limpiar
Regresar Menú


Propiedades Geometricas
Rectángulo
Trapecio
Triángulo
Círculo
Parábola
Combinaciones
Ingresa Datos
Talud Z1
Talud Z2
Base b
Tirante Hidráulico y
Resultados
T
1
1
y
Z1
Z2
b
Área A
Ancho Superficial T
Perímetro Mojado P
Profundidad Hidráulica D
Radio Hidráulico Rh
Factor de Sección Z
Calcular
Limpiar
Regresar Menú


Propiedades Geometricas
Rectángulo
Trapecio
Triángulo
Círculo
Parábola
Combinaciones
Ingresa Datos
Constante K
Tirante Hidráulico y
Resultados
T
y
Área A
Ancho Superficial T
Perímetro Mojado P
Profundidad Hidráulica D
Radio Hidráulico Rh
Factor de Sección Z
Calcular
Limpiar
Regresar Menú

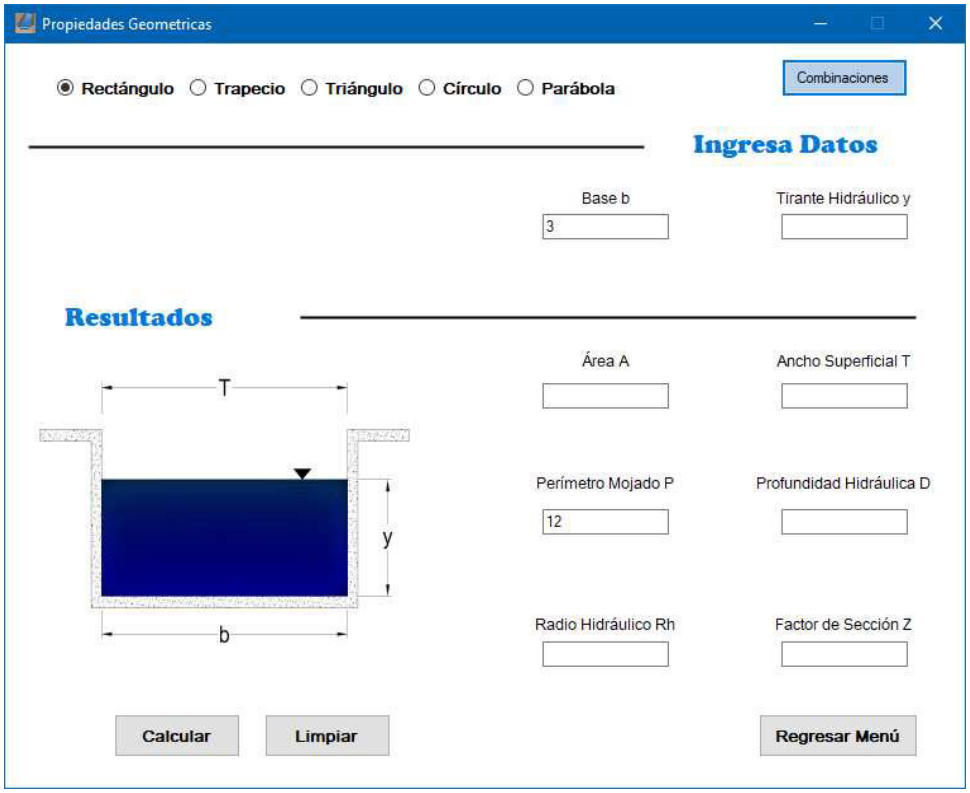
Para las primeras ocasiones en que el usuario maneja este módulo podrá ser que tenga dudas en las combinaciones que existen para ingresar datos. Por ello existe el botón combinaciones que de ser presionado abre una venta con las combinaciones existentes para cada una de las secciones.

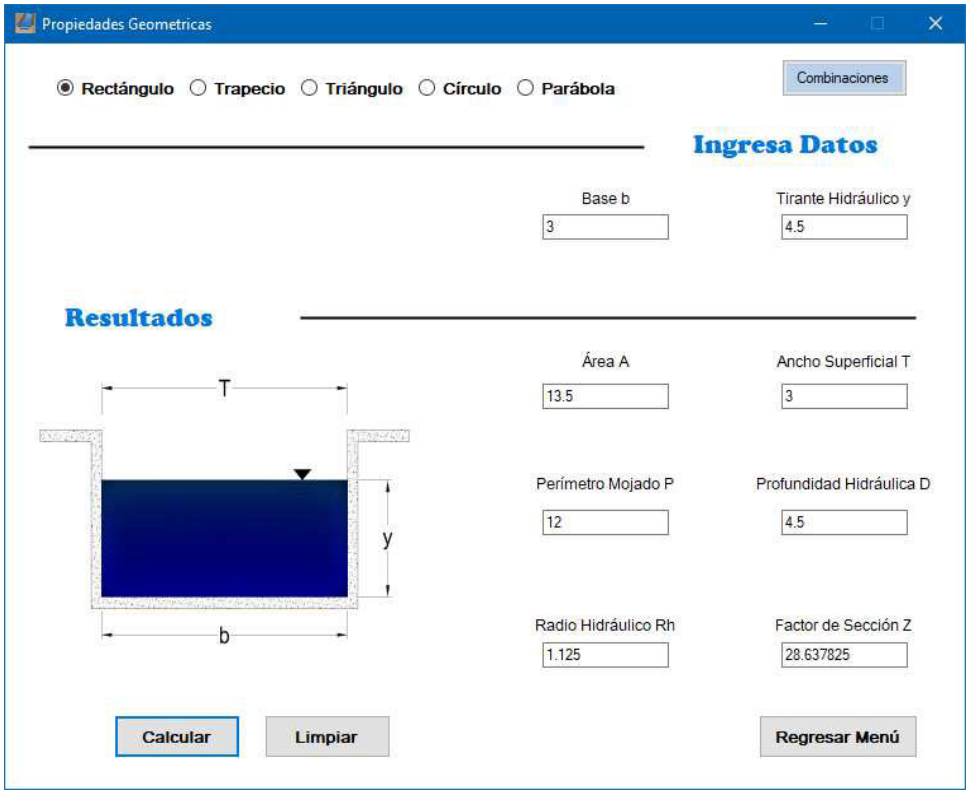




Rectángulo							
Datos ingresados		Datos calculados					
Base b	Tirante hidráulico y	A	P	R_h	T	D	Z
	Área A	y	P	R_h	T	D	Z
	Perímetro mojado P	A	y	R_h	T	D	Z
	Radio hidráulico Rh	A	P	y	T	D	Z
	Profundidad hidrálica D	A	P	R_h	T	y	Z
Tirante y	Área A	b	P	R_h	T	D	Z
	Perímetro mojado P	A	b	R_h	T	D	Z
	Radio hidráulico Rh	A	P	b	T	D	Z
	Ancho superficial T	A	P	R_h	b	D	Z

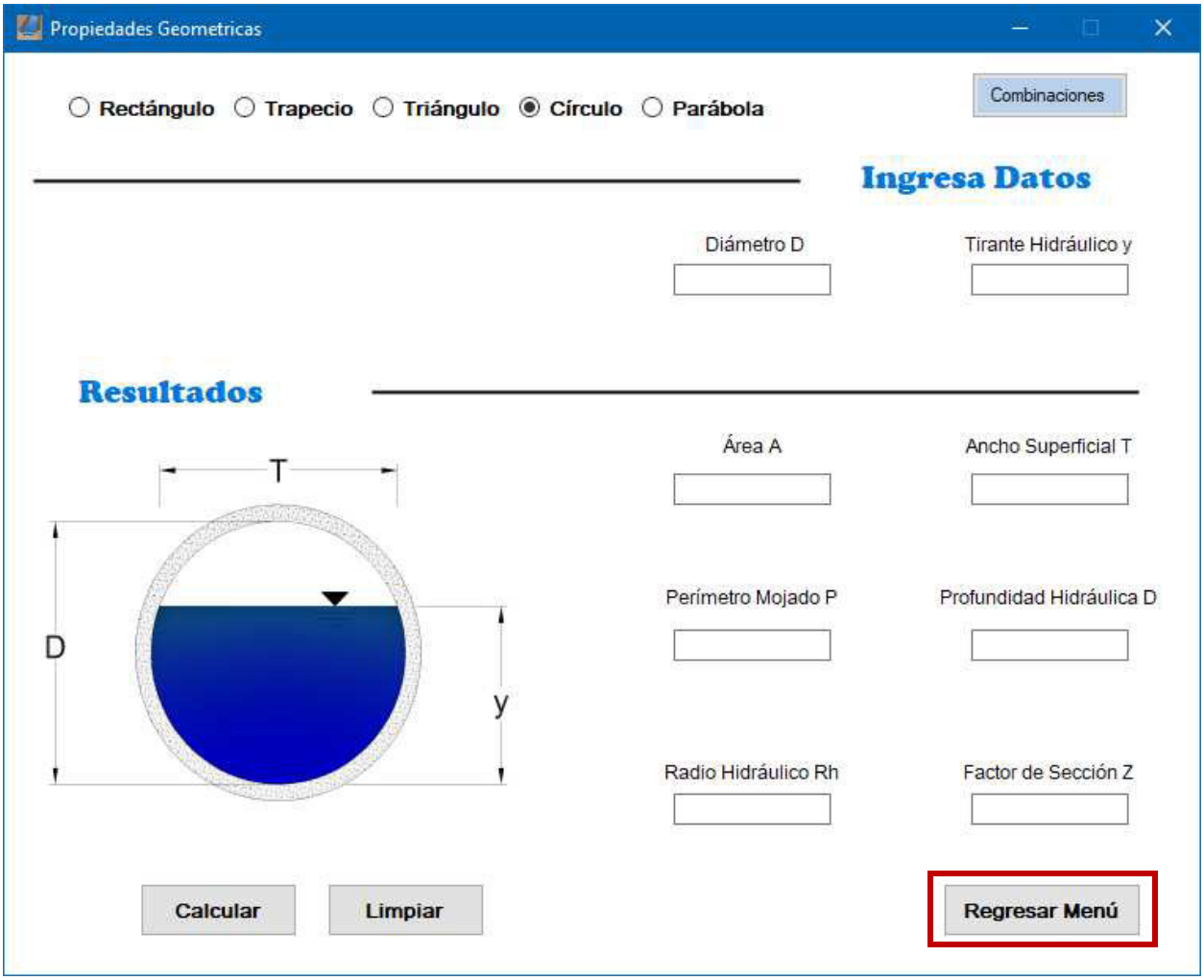
Suponiendo que cierto ejercicio propone una sección circular de 3 metros de base y un radio hidráulico de 12 metros, se tienen los siguientes resultados.



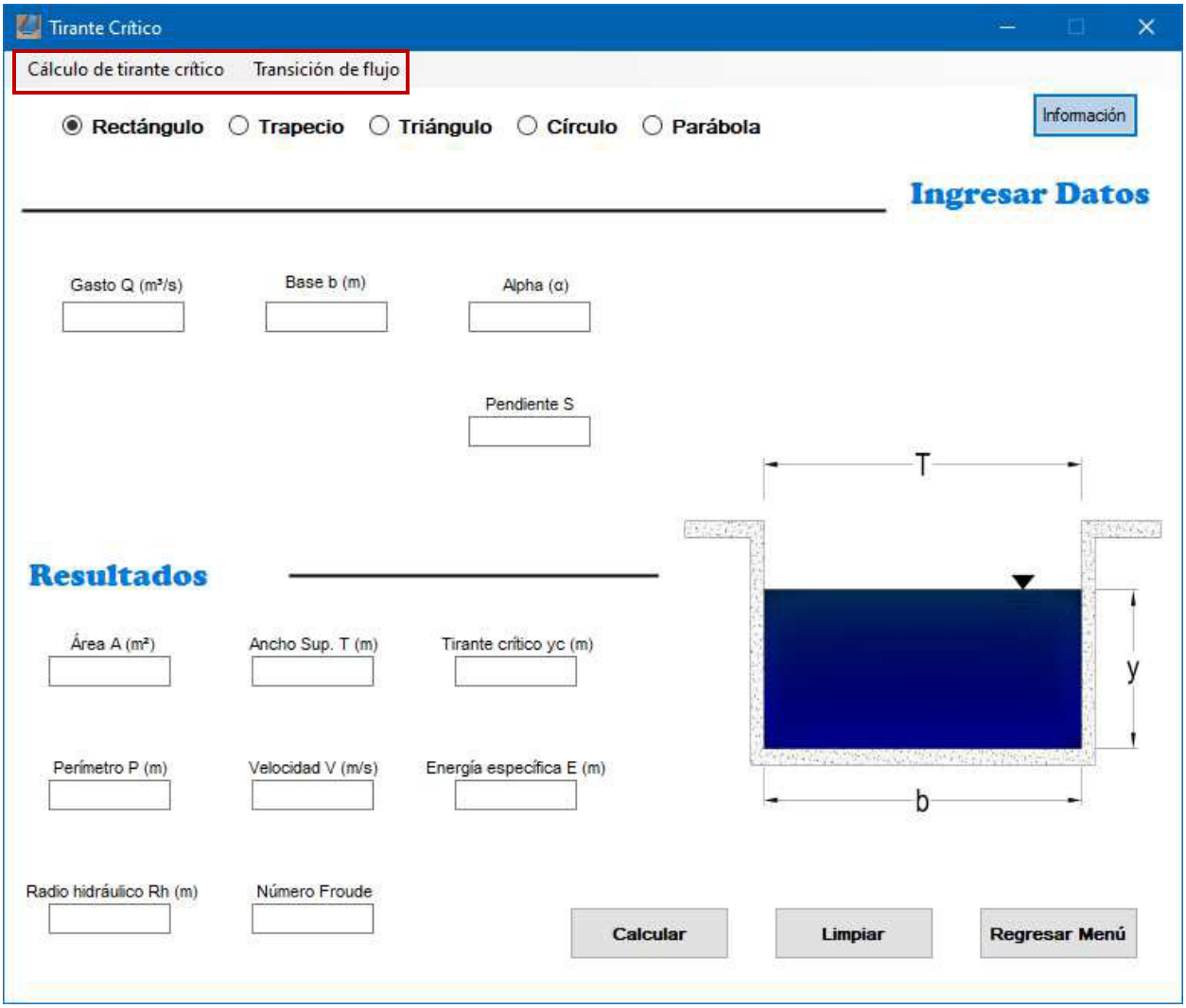


Es importante realizar la aclaración de que, aunque en pantalla se aprecia un apartado denominado “Ingresa Datos” y otro “Resultados”, dentro del segundo apartado es posible ingresar un valor y el otro valor en el primer apartado.

Si el usuario ha finalizado todos sus cálculos en el módulo de propiedades geométricas y desea usar otro módulo, es necesario presionar “Regresar Menú”. En el caso de que ya no se tenga necesidad de ir a otro módulo y no se desea hacer más cálculos, basta con cerrar la venta como en cualquier programa y con ello finaliza completamente la aplicación.



Procediendo al siguiente módulo “Tirante crítico”, este presenta dos subdivisiones, “Cálculo de tirante crítico” que estará seleccionado al iniciar este módulo y “Transición de flujo”.



Similar a “Propiedades geométricas”, en “cálculo de tirante crítico” se debe seleccionar una de las 5 secciones y cada una mostrará el dibujo de una sección de canal, pero distintas cajas de texto en el área “Ingresar Datos”, por ejemplo, la sección trapecial muestra cajas de texto para: Gasto, Base, Talud 1, Talud 2, Alpha y Pendiente. En cambio, una sección circular presenta las cajas: Gasto, Diámetro, Alpha y pendiente. Sin embargo, en el área “Resultados” las cajas de texto se mantienen igual para todas las secciones y en este caso no se han de ingresar valores, sino que sólo se imprimen los datos obtenidos del cálculo.

Tirante Crítico

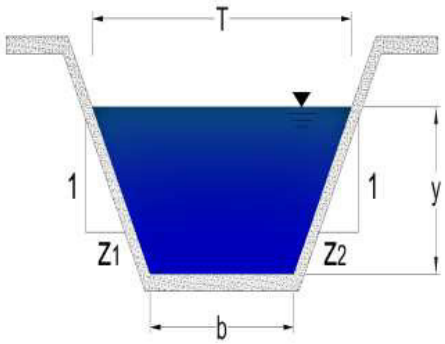
Cálculo de tirante crítico Transición de flujo

○ Rectángulo ◉ Trapecio ○ Triángulo ○ Círculo ○ Parábola Información

Ingresar Datos

Gasto Q (m³/s) Base b (m) Alpha (α)

Talud Z1 Talud Z2 Pendiente S



Resultados

Área A (m²) Ancho Sup. T (m) Tirante crítico yc (m)

Perímetro P (m) Velocidad V (m/s) Energía específica E (m)

Radio hidráulico Rh (m) Número Froude

Calcular Limpiar Regresar Menú

Tirante Crítico

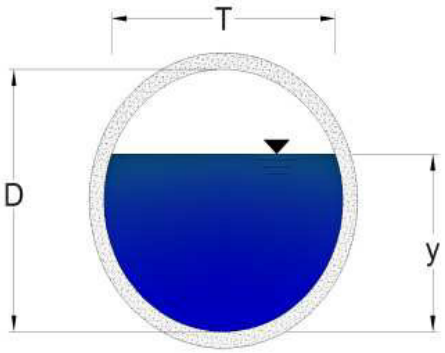
Cálculo de tirante crítico Transición de flujo

○ Rectángulo ○ Trapecio ○ Triángulo ◉ Círculo ○ Parábola Información

Ingresar Datos

Gasto Q (m³/s) Diámetro D (m) Alpha (α)

Pendiente S



Resultados

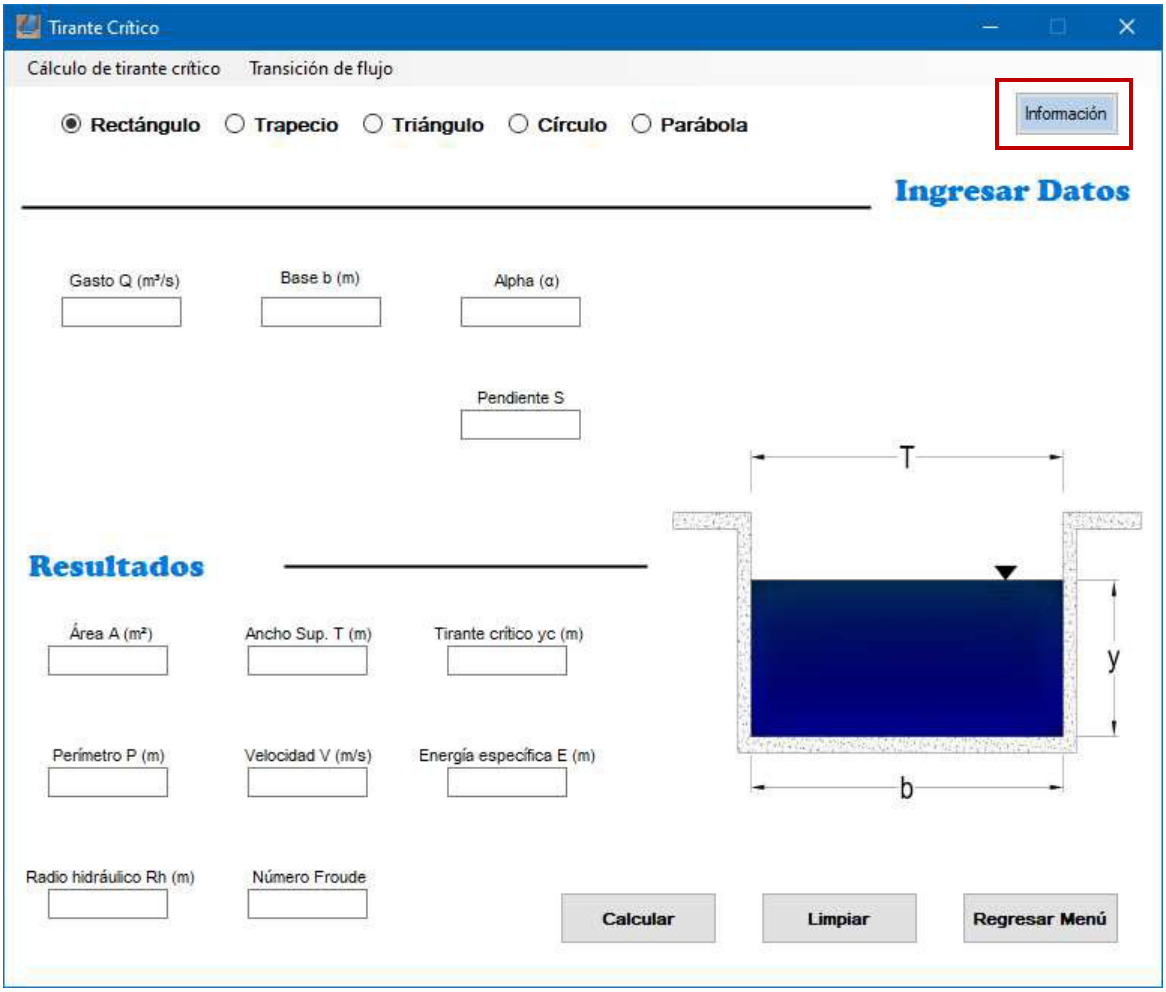
Área A (m²) Ancho Sup. T (m) Tirante crítico yc (m)

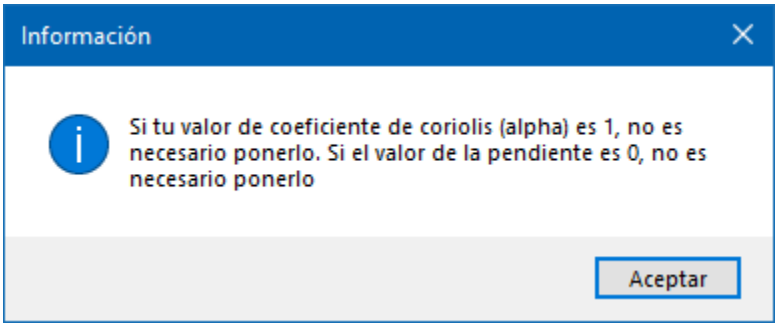
Perímetro P (m) Velocidad V (m/s) Energía específica E (m)

Radio hidráulico Rh (m) Número Froude

Calcular Limpiar Regresar Menú

En la esquina superior derecha existe un botón llamado “Información”, que de ser apretado salta una ventana con un mensaje.





En muchas ocasiones los ejercicios en materias de hidráulica no mencionan un coeficiente de Coriolis ni una pendiente, esto porque se consideran 1 y 0 respectivamente. En ese caso puede dejarse ese para de cajas de texto solas y se efectuará el cálculo o si se ingresan los valores también se realizará sin complicaciones.

En este punto se ha dejado el manual de usuario, para poder revisarlo todo es necesario descargar el programa e instalarlo. Al realizar esta acción se podrá descargar la versión completa de manual de usuario, donde incluye la manera de utilizar cada uno de los módulos y sus respectivos submódulos.

Conclusiones

Día a día el campo de la construcción demanda nuevas herramientas que simplifiquen los análisis, cálculos y diseños. La combinación de ingeniería, programación y métodos numéricos han dado como resultado diferentes programas que en la actualidad son fundamentales para empresas enfocadas al ámbito de obra civil.

En el mercado actual existen diferentes programas que hacen relación a la hidráulica de canales como HEC-RAS, HCanales, Hidra_Bas, entre otros. HYDROGECA se ha creado desde la perspectiva de un estudiante a nivel licenciatura pensando en el cómo apoyar a sus compañeros. Las hojas de cálculo que generalmente son creadas en Excel son útiles y, además, obligan a que se comprenda la teoría de una manera satisfactoria para poder programarlas. Sin embargo, el tener un software que esté enfocado a la geometría de canales, el tirante crítico y el flujo normal hace que muchas hojas de cálculo se unifiquen dando como resultado un programa multifuncional.

Dentro de los beneficios que lo caracterizan se puede mencionar, que es un programa muy estático en cuanto a su interfaz. Aunque al cambiar de tipo de sección se muestra un dibujo de un canal distinto, todas las cajas de texto aparecen al mismo tiempo, tanto cajas para ingresar datos como las que mostrarán resultados. Esto es meramente visual y no afectaría en nada los cálculos realizados.

Con la implementación de esta herramienta computacional, los alumnos del programa educativo de ingeniería civil realizarán diseños y revisiones de casos de estudio en un menor periodo de tiempo, logrando con ello reducir el tiempo destinado en comparación a realizar los mismos análisis por medios de hojas de cálculo en Excel u otro tipo de programación. Es por ello que los ejemplos incluidos en el desarrollo de este trabajo, ayudará de manera significativa en el entendimiento de la interfaz que hace el programa y de la facilidad para poder hacer corridas de diseños.

A partir de esta tesis, la División de Ingenierías del Campus Guanajuato contará con una propuesta de programa computacional propia, que puede ser distribuido sin ninguna restricción a la comunidad estudiantil, debido a que es de código libre y no es necesario invertir en costos de adquisición, además, el programa está abierto a cambios y sugerencias que mejoren la estructura y el alcance de este tipo de estudios.

Futuro del programa

Hay cuatro puntos fundamentales a destacar:

- Localizar posibles fallas o errores del programa: Aunque al crear HYDROGECA se ha tratado de tener la mayor precaución posible, siempre es posible que el usuario inconscientemente encuentre errores en algún submódulo del programa. Una vez que se notifique la falla o defecto, se busca la manera de corregirlo.
- Mejora de interfaz: El creador de cierto producto llega al punto de sacarlo al público. Este cumple con el objetivo por el cual ha sido creado, a pesar de ello, en ocasiones no es muy usado o consumido por su presentación. En un futuro se piensa en trabajar en conjunto con alguien que pueda ayudar a dar un formato más atractivo.
- Añadir nuevos módulos: El tema de canales se puede extender mucho más que sólo tres módulos, por esto se prevé que HYDROGECA cuente con más funciones y que se pueda trabajar a un nivel de maestría incluyendo temas del flujo gradualmente variado, estructuras de control, flujo espacialmente variado, entre otras, que son poco estudiadas inclusive en los textos de referencia.
- Dentro del flujo uniforme aún queda la oportunidad de ampliar los criterios de desarrollo del programa propuesto, incluyendo la parte de las secciones compuestas, que serían de gran ayuda para los casos de revisión, sin embargo, al ser cambiantes las combinaciones de formas que se pueden generar, su planteamiento requiere mayor tiempo de análisis.

Referencias

Aiyesimoju, K. O. (2010). Universal specific energy curve for parabolic open channels. Journal of Science and Technology (Ghana), 30(1).

Attari, M. & Taherian, M. & Mahmood-Hosseini, S. & Bahram-Niazmand, S. & Jeiroodi, M. & Mohammadian, A. (2021) A simple and robust method for identifying the distribution functions of Manning´s roughness coefficient along a natural river. Journal of Hydrology, 595, 1 – 14.

Blalock, M. E. (1980). Minimum specific energy in open channels of compound section (Doctoral dissertation, Georgia Institute of Technology).

Castro-Orgaz, O., & Chanson, H. (2016). Minimum specific energy and transcritical flow in unsteady open-channel flow. Journal of Irrigation and Drainage Engineering, 142(1), 04015030.

Chapra, S., & Canale, R. (2006). Métodos numéricos para ingenieros (Quinta ed.). (J. Enríquez Brito, & M. Roa Hano, Trads.) McGraw-Hill Education.

Chow, V. (1994). Hidráulica de canales abiertos. McGraw-Hill Education.

Costabile, P., & Macchione, F. (2012). Analysis of one-dimensional modelling for flood routing in compound channels. Water resources management, 26(5), 1065-1087.

Cowan, W. (1956). Estimating hydraulic roughness coefficients. Agricultural Engineering, 473-475.

Guerreo-Angulo, J. O. & Arreguín-Cortés, F. I. & Félix-Higuera, J. L. (2003) Ecuaciones Explícitas para el cálculo del tirante crítico en canales trapeciales y circulares. Ingeniería Hidráulica en México. 18 (3), 105 - 110.

Ladino-Moreno, E. O. & García-Ubaque, C. A. & Pineda-Jaimes, J. A. (2021). Development of mobile APP for interactive learning in civil engineering problems: application to open-channel hydraulics. Tecnura, 25(65), 53-70.

Ladino-Moreno, E. O. & García-Vaca, M. C. (2020) Critical flow in open channels: Numerical solution using the Newton-Raphson method for Android 4.0 appplication. Tecnura, 24(63), 88-103.

Lee, J. S., Lee, S. O., Gray, D. D., & Hong, S. H. (2019). Visualization of Specific Energy for Open Channel Flow in Three Dimensions. KSCE Journal of Civil Engineering, 23(6), 2541-2549.

Lee, P. J., Lambert, M. F., & Simpson, A. R. (2002, December). Critical depth prediction in straight compound channels. In Proceedings of the Institution of Civil Engineers-Water and Maritime Engineering (Vol. 154, No. 4, pp. 317-332). Thomas Telford Ltd.

Osman, A. A. (2006). Open Channel Hydraulics. Butterworth-Heinemann.

Sotelo, G. (2002). Hidráulica de canales. Universidad Nacional Autónoma de México, Facultad de ingenierías.

Szymkiewicz, R. (2010) Numerical Modeling in Open Channel Hydraulics, Water Science and Technology. Springer.

Anexos

Código Módulo Propiedades geométricas

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Hidraulica_canales
{
    public partial class Propiedadesgeometricas : Form
    {
        public Propiedadesgeometricas()
        {
            InitializeComponent();
        }

        private void label11_Click(object sender, EventArgs e)
        {

        }
        public void limpiar()
        {
            t_anchosup.Clear();
            t_area.Clear();
            t_base.Clear();
            t_diam.Clear();
            t_factz.Clear();
            t_perimetrom.Clear();
            t_profhid.Clear();
            t_RadioH.Clear();
            t_tirantehid.Clear();
            t_z1.Clear();
            t_z2.Clear();
            textk.Clear();
        }
        private void Propiedadesgeometricas_Load(object sender, EventArgs e)
        {

            radiocirculo.Checked = false;
            radioparabola.Checked = false;
            radiorectan.Checked = false;
            radioTrapecio.Checked = false;
            radiotriang.Checked = false;
            t_anchosup.Hide();
            t_area.Hide();
            t_base.Hide();
            t_diam.Hide();
            t_factz.Hide();
            t_perimetrom.Hide();
            t_profhid.Hide();
            t_RadioH.Hide();
            t_tirantehid.Hide();
            t_z1.Hide();
            t_z2.Hide();
            l_anchosup.Hide();
            l_area.Hide();
            l_base.Hide();
            l_diam.Hide();
            l_Factorsec.Hide();
            l_perimetrom.Hide();
            l_profhid.Hide();
            l_rh.Hide();
            l_tiranteh.Hide();
            l_z1.Hide();
            l_z2.Hide();
            lk.Hide();
            textk.Hide();
            pictureBox1.Hide();
            lresult2.Hide();
            lingres.Hide();
            line2.Hide();
```

```
            line3.Hide();
        }

        private void radiorectan_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            t_anchosup.Show();
            t_area.Show();
            t_base.Show();
            t_diam.Hide();
            t_factz.Show();
            t_perimetrom.Show();
            t_profhid.Show();
            t_RadioH.Show();
            t_tirantehid.Show();
            t_z1.Hide();
            t_z2.Hide();
            l_anchosup.Show();
            l_area.Show();
            l_base.Show();
            l_diam.Hide();
            l_Factorsec.Show();
            l_perimetrom.Show();
            l_profhid.Show();
            l_rh.Show();
            l_tiranteh.Show();
            l_z1.Hide();
            l_z2.Hide();
            textk.Hide();
            lk.Hide();
            pictureBox1.Show();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.rectángulo;
            lresult2.Show();
            lingres.Show();
            line2.Show();
            line3.Show();
        }

        private void radioTrapecio_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            t_anchosup.Show();
            t_area.Show();
            t_base.Show();
            t_diam.Hide();
            t_factz.Show();
            t_perimetrom.Show();
            t_profhid.Show();
            t_RadioH.Show();
            t_tirantehid.Show();
            t_z1.Show();
            t_z2.Show();
            l_anchosup.Show();
            l_area.Show();
            l_base.Show();
            l_diam.Hide();
            l_Factorsec.Show();
            l_perimetrom.Show();
            l_profhid.Show();
            l_rh.Show();
            l_tiranteh.Show();
            l_z1.Show();
            l_z2.Show();
            textk.Hide();
            lk.Hide();
            pictureBox1.Show();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.trapecio;
            lresult2.Show();
            lingres.Show();
            line2.Show();
            line3.Show();
        }

        private void radiotriang_CheckedChanged(object sender, EventArgs e)
```

```
        {
            limpiar();
            t_anchosup.Show();
            t_area.Show();
            t_base.Hide();
            t_diam.Hide();
            t_factz.Show();
            t_perimetrom.Show();
            t_profhid.Show();
            t_RadioH.Show();
            t_tirantehid.Show();
            t_z1.Show();
            t_z2.Show();
            l_anchosup.Show();
            l_area.Show();
            l_base.Hide();
            l_diam.Hide();
            l_Factorsec.Show();
            l_perimetrom.Show();
            l_profhid.Show();
            l_rh.Show();
            l_tiranteh.Show();
            l_z1.Show();
            l_z2.Show();
            textk.Hide();
            lk.Hide();
            pictureBox1.Show();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.triángulo;
            lresult2.Show();
            lingres.Show();
            line2.Show();
            line3.Show();
        }

        private void radiocirculo_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            t_anchosup.Show();
            t_area.Show();
            t_base.Hide();
            t_diam.Show();
            t_factz.Show();
            t_perimetrom.Show();
            t_profhid.Show();
            t_RadioH.Show();
            t_tirantehid.Show();
            t_z1.Hide();
            t_z2.Hide();
            l_anchosup.Show();
            l_area.Show();
            l_base.Hide();
            l_diam.Show();
            l_Factorsec.Show();
            l_perimetrom.Show();
            l_profhid.Show();
            l_rh.Show();
            l_tiranteh.Show();
            l_z1.Hide();
            l_z2.Hide();
            textk.Hide();
            lk.Hide();
            pictureBox1.Show();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.círculo;
            lresult2.Show();
            lingres.Show();
            line2.Show();
            line3.Show();
        }

        private void radioparabola_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            t_anchosup.Show();
            t_area.Show();
```

```
            t_base.Hide();
            t_diam.Hide();
            t_factz.Show();
            t_perimetrom.Show();
            t_profhid.Show();
            t_RadioH.Show();
            t_tirantehid.Show();
            t_z1.Hide();
            t_z2.Hide();
            l_anchosup.Show();
            l_area.Show();
            l_base.Hide();
            l_diam.Hide();
            l_Factorsec.Show();
            l_perimetrom.Show();
            l_profhid.Show();
            l_rh.Show();
            l_tiranteh.Show();
            l_z1.Hide();
            l_z2.Hide();
            textk.Show();
            lk.Show();
            pictureBox1.Show();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.parábola;
            lresult2.Show();
            lingres.Show();
            line2.Show();
            line3.Show();
        }

        private void Propiedadesgeometricas_FormClosing(object sender, FormClosingEventArgs e)
        {
            Application.Exit();
        }

        private void b_Limpiar_Click(object sender, EventArgs e)
        {
            limpiar();
        }

        private void b_calcular_Click(object sender, EventArgs e)
        {
            double bas, tirh, z1, z2, area, perimetro, radiohid, anchosup, profhid, factZ, Diam, radio, tetha↙
    ,tetha2,Diam2,tirh2,x,k;
            double x1 = 0.0000001;
            double x2 = 2 * Math.PI;
            double x3 = 0;
            double x4 = 2*Math.PI;
            double x5 = 0;
            double xold = 0;
            double ea = 0;
            int conteo = 0,conteo2=0;
            int n = 500;
            double epsilon = 1.0e-10;

        ////////////////////////////////////////////////////////////////////////////////////////////////↙
    ///////////
            if (radiorectan.Checked == true)
            {
                //Aquí se comienzan los cálculos para las propiedades geometricas para sección rectangualar
                if (t_base.Text != "" && t_tirantehid.Text != "" && t_area.Text == "" && t_perimetrom.Text ==↙
     "" && t_RadioH.Text == "" && t_anchosup.Text == "" && t_factz.Text == "" && t_profhid.Text == "")
                    //caso donde conocemos base y tirante
                {
                    bas = Convert.ToDouble(t_base.Text);
                    tirh = Convert.ToDouble(t_tirantehid.Text);
                    area = bas * tirh;
                    perimetro = bas + 2 * tirh;
                    radiohid = area / perimetro;
                    anchosup = bas;
                    profhid = tirh;
                    factZ = bas *Math.Pow(tirh,1.5);
                    t_area.Text = Convert.ToString(Math.Round(area,6));
                    t_perimetrom.Text = Convert.ToString(Math.Round(perimetro,6));
```

```
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid,6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup,6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid,6));
                t_factz.Text = Convert.ToString(Math.Round(factZ,6));

            }
            if (t_base.Text != "" && t_area.Text != "" && t_perimetrom.Text == "" && t_RadioH.Text == "" ↙
&& t_anchosup.Text == "" && t_factz.Text == "" && t_profhid.Text == "" && t_tirantehid.Text == "" )
                //caso donde conocemos base y área
            {
                bas = Convert.ToDouble(t_base.Text);
                area = Convert.ToDouble(t_area.Text);
                tirh = area/bas;
                perimetro = bas + 2 * tirh;
                radiohid = area / perimetro;
                anchosup = bas;
                profhid = tirh;
                factZ = bas * Math.Pow(tirh, 1.5);
                t_tirantehid.Text = Convert.ToString(Math.Round(tirh,6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro,6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid,6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup,6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid,6));
                t_factz.Text = Convert.ToString(Math.Round(factZ,6));

            }
            if (t_base.Text != "" && t_perimetrom.Text != "" && t_RadioH.Text == "" && t_anchosup.Text ==↙
 "" && t_area.Text == "" && t_factz.Text == "" &&  t_profhid.Text == "" && t_tirantehid.Text == "")
                //caso donde conocemos base y perímetro
            {
                bas = Convert.ToDouble(t_base.Text);
                perimetro = Convert.ToDouble(t_perimetrom.Text);
                tirh = (perimetro - bas) * 0.5;

                if (perimetro <= bas || perimetro <= tirh)
                {
                    MessageBox.Show("El valor del perímetro debe ser mayor que el valor de la base",     ↙
"Error");
                }
                if (perimetro > bas)
                {

                    area = bas * tirh;
                    radiohid = area / perimetro;
                    anchosup = bas;
                    profhid = tirh;
                    factZ = bas * Math.Pow(tirh, 1.5);
                    t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }
            }

            if (t_base.Text != "" && t_RadioH.Text != "" && t_area.Text == "" && t_perimetrom.Text == "" ↙
&& t_anchosup.Text == "" && t_factz.Text == "" && t_profhid.Text == "" && t_tirantehid.Text == "")
                //caso donde conocemos base y Radio hidráulico
            {
                bas = Convert.ToDouble(t_base.Text);
                radiohid = Convert.ToDouble(t_RadioH.Text);
                tirh = (radiohid * bas) / (bas - 2 * radiohid);
                perimetro = bas + 2 * tirh;

                if (perimetro <= bas || perimetro <= tirh || tirh <= 0)
                {
                    MessageBox.Show("El valor del Radio hidráulico es erroneo ya que no podemos tener un ↙
base menor al perimetro o un tirante menor al perímetro o un tirante menor o igual a cero","Error");
                }

                if (perimetro > bas)
                {
                    area = bas * tirh;
```

```
                    anchosup = bas;
                    profhid = tirh;
                    factZ = bas * Math.Pow(tirh, 1.5);
                    t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                    t_area.Text = Convert.ToString(Math.Round(area,6));
                    t_perimetrom.Text = Convert.ToString(Math.Round(perimetro,6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup,6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid,6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }

            }

            if (t_base.Text != "" && t_area.Text == "" && t_perimetrom.Text == "" && t_RadioH.Text == "" ↙
&& t_anchosup.Text == "" && t_factz.Text == "" && t_profhid.Text != "" && t_tirantehid.Text == "")
            //caso donde conocemos base y profundidad hidráulica
            {
                bas = Convert.ToDouble(t_base.Text);
                profhid = Convert.ToDouble(t_profhid.Text);
                tirh = profhid;
                area = bas * profhid;
                perimetro = bas + 2 * tirh;
                radiohid = area / perimetro;
                anchosup = bas;
                profhid = tirh;
                factZ = bas * Math.Pow(tirh, 1.5);
                t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));

            }

            if (t_tirantehid.Text != "" && t_area.Text != "" && t_base.Text == "" && t_RadioH.Text == "" ↙
&& t_perimetrom.Text == "" && t_anchosup.Text == "" && t_factz.Text == "" && t_profhid.Text == "")
                //caso donde conocemos tirante y área
            {
                area = Convert.ToDouble(t_area.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                bas = area / tirh;
                perimetro = bas + 2 * tirh;
                radiohid = area / perimetro;
                anchosup = bas;
                profhid = tirh;
                factZ = bas * Math.Pow(tirh, 1.5);
                t_base.Text = Convert.ToString(Math.Round(bas,6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro,6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid,6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup,6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid,6));
                t_factz.Text = Convert.ToString(Math.Round(factZ,6));

            }
            if (t_perimetrom.Text != "" && t_tirantehid.Text != "" && t_area.Text == "" && t_base.Text ==↙
 "" && t_RadioH.Text == "" && t_anchosup.Text == "" && t_factz.Text == "" && t_profhid.Text == "")
                //caso donde conocemos tirante y perímetro
            {
                perimetro = Convert.ToDouble(t_perimetrom.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                bas = perimetro-2*tirh;

                if (perimetro <= bas || perimetro <= tirh || bas<=0)
                {
                    MessageBox.Show("Puede que tu perímetro sea menor que tu tirante hidráulico, tu      ↙
perímetro sea menor que tu base, el valor de tu base resulte negativa o una base con valor de cero",     ↙
"Error");
                }

                if (perimetro > tirh && bas>0)
                {
                    area = bas * tirh;
                    radiohid = area / perimetro;
```

```
                    anchosup = bas;
                    profhid = tirh;
                    factZ = bas * Math.Pow(tirh, 1.5);
                    t_base.Text = Convert.ToString(Math.Round(bas, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }

            }
            if (t_base.Text == "" && t_RadioH.Text != "" && t_anchosup.Text == "" && t_area.Text == "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && t_tirantehid.Text != "")
                //caso donde conocemos tirante y Radio hidráulico
            {
                radiohid = Convert.ToDouble(t_RadioH.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                bas = (radiohid * 2 * tirh) / (tirh - radiohid);
                perimetro = bas + 2 * tirh;

                if (perimetro <= bas || perimetro <= tirh || tirh <= 0)
                {
                    MessageBox.Show("Puede que tu perímetro sea menor que tu tirante hidráulico, tu perímetro sea menor que tu base o pude que el valor de tu base resulte negativa","Error");
                }
                if (perimetro > tirh)
                {
                    area = bas * tirh;
                    anchosup = bas;
                    profhid = tirh;
                    factZ = bas * Math.Pow(tirh, 1.5);
                    t_base.Text = Convert.ToString(Math.Round(bas, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }

            }

            if (t_tirantehid.Text != "" && t_area.Text == "" && t_base.Text == "" && t_RadioH.Text == "" && t_perimetrom.Text == "" && t_anchosup.Text != "" && t_factz.Text == "" && t_profhid.Text == "")
            //caso donde conocemos tirante y ancho superfical
            {
                anchosup = Convert.ToDouble(t_anchosup.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                bas = anchosup;
                area = tirh * anchosup;
                perimetro = bas + 2 * tirh;
                radiohid = area / perimetro;
                profhid = tirh;
                factZ = bas * Math.Pow(tirh, 1.5);
                t_base.Text = Convert.ToString(Math.Round(bas, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));

            }
        }
        ////////////////////////////////////////////////////////////////////////////////////////////////////////////
        if (radioTrapecio.Checked== true) //es seleccionado el radiobutton de trapecio
        {
            //Aquí se comienzan los cálculos para las propiedades geometricas para sección Trapecial
            if (t_base.Text != "" && t_tirantehid.Text != "" && t_z1.Text!= "" && t_z2.Text!="" && t_area.Text=="" && t_perimetrom.Text=="" && t_RadioH.Text=="" && t_anchosup.Text=="" && t_profhid.Text=="" && t_factz.Text=="")
                //caso donde conocemos base, tirante y taludes
            {
                bas = Convert.ToDouble(t_base.Text);
```

```
                tirh = Convert.ToDouble(t_tirantehid.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
                area = (bas+tirh*0.5*(z1+z2))*tirh;
                perimetro = bas + tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                radiohid = area / perimetro;
                anchosup = bas+tirh*z1+tirh*z2;
                profhid = area/anchosup;
                factZ =(Math.Pow(area,1.5))/(Math.Pow(anchosup,0.5));
                t_area.Text = Convert.ToString(Math.Round(area,6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro,6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid,6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup,6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid,6));
                t_factz.Text = Convert.ToString(Math.Round(factZ,6));

            }
            if (t_base.Text != "" && t_tirantehid.Text != "" && t_anchosup.Text != "" && t_area.Text ==
"" && t_perimetrom.Text == "" && t_RadioH.Text == "" && t_profhid.Text == "" && t_factz.Text == "" &&
t_z1.Text=="" && t_z2.Text=="")
                //caso donde conocemos base, tirante y Ancho superficial
                // Nota: se considerará que z1 y z2 son iguales
            {
                bas = Convert.ToDouble(t_base.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                anchosup = Convert.ToDouble(t_anchosup.Text);
                z1 = (anchosup - bas) / (2 * tirh);
                z2 = z1;
                area = (bas + (z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                perimetro = bas + tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                radiohid = area / perimetro;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_z1.Text = Convert.ToString(Math.Round(z1, 6));
                t_z2.Text = Convert.ToString(Math.Round(z2, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ,6));

            }
            if (t_base.Text != "" && t_tirantehid.Text != "" && t_area.Text != "" && t_z1.Text == "" &&
t_z2.Text == "" && t_perimetrom.Text == "" && t_RadioH.Text == "" && t_profhid.Text == "" && t_factz.Text
 == "" && t_anchosup.Text == "")
                //caso donde conocemos base, tirante y Área
            // Nota: se considerará que z1 y z2 son iguales
            {
                bas = Convert.ToDouble(t_base.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                area = Convert.ToDouble(t_area.Text);
                z1 = (area/(tirh*tirh)) - (bas/tirh);
                z2 = z1;
                anchosup = bas + tirh * z1 + tirh * z2;
                perimetro = bas + tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                radiohid = area / perimetro;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_z1.Text = Convert.ToString(Math.Round(z1, 6));
                t_z2.Text = Convert.ToString(Math.Round(z2,6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));

            }
            if (t_base.Text != "" && t_tirantehid.Text != "" && t_perimetrom.Text != "" && t_area.Text ==
 "" && t_z1.Text == "" && t_z2.Text == "" && t_RadioH.Text == "" && t_profhid.Text == "" && t_factz.Text
== "" && t_anchosup.Text == "")
                //caso donde conocemos base, tirante y perímetro
            // Nota: se considerará que z1 y z2 son iguales
            {
                bas = Convert.ToDouble(t_base.Text);
```

```
                tirh = Convert.ToDouble(t_tirantehid.Text);
                perimetro = Convert.ToDouble(t_perimetrom.Text);
                z1 = Math.Pow(Math.Pow((perimetro-bas)/(2*tirh),2)-1,0.5);

                if (perimetro <= bas || perimetro <= tirh || z1 < 0)
                {
                    MessageBox.Show("El valor del perímetro mojado debe ser mayor que el tirante hidráulico y la base. A su vez los valores ingresados nos deben arrojar un talud mayor o igual a cero", "Error");
                }
                if (perimetro > bas || perimetro > tirh || z1 >= 0)
                {
                    z2 = z1;
                    anchosup = bas + tirh * z1 + tirh * z2;
                    area = (bas + (z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                    radiohid = area / perimetro;
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                    t_z1.Text = Convert.ToString(Math.Round(z1, 6));
                    t_z2.Text = Convert.ToString(Math.Round(z2, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }

            }
            if (t_base.Text != "" && t_tirantehid.Text != "" && t_RadioH.Text != "" && t_anchosup.Text == "" && t_area.Text == "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && t_z1.Text == "" && t_z2.Text == "")
                //caso donde conocemos base, tirante y Radio hidráulico
            // Nota: se considerará que z1 y z2 son iguales checar detalle
            {
                bas = Convert.ToDouble(t_base.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                radiohid = Convert.ToDouble(t_RadioH.Text);// hay que incluir un método iterativo
                if (((4 * radiohid * radiohid - tirh * tirh) * tirh) == 0)
                {
                    do
                    {
                        xold= x3;
                        x3= ((radiohid*(bas+2*tirh*Math.Sqrt(1+xold*xold))/tirh)-bas)/tirh;
                        conteo++;

                            ea=Math.Abs((x3-xold)/x3)*100;

                    }
                    while (ea>=epsilon || conteo<=100  );


                    z1 = x3;
                    z2 = z1;
                    anchosup = bas + tirh * z1 + tirh * z2;
                    area = (bas + (z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                    perimetro = bas + tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                    t_z1.Text = Convert.ToString(Math.Round(z1, 6));
                    t_z2.Text = Convert.ToString(Math.Round(z2, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }

                if (((4 * radiohid * radiohid - tirh * tirh) * tirh) != 0)
                {

                    z1 = -(2 * radiohid * Math.Sqrt(-radiohid * radiohid * (4 * tirh * tirh - bas * bas) - 2 * bas * bas * radiohid * tirh + tirh * tirh * (tirh * tirh + bas * bas)) + bas * (radiohid - tirh) * tirh) / ((4 * radiohid * radiohid - tirh * tirh) * tirh);
                    z2 = (2 * radiohid * Math.Sqrt(-radiohid * radiohid * (4 * tirh * tirh - bas * bas) -
```

```
 2 * bas * bas * radiohid * tirh + tirh * tirh * (tirh * tirh + bas * bas)) - bas * (radiohid - tirh) * tirh) / ((4 * radiohid * radiohid - tirh * tirh) * tirh);
                    DialogResult vent = MessageBox.Show("El resultado de un posible talud es:  " + Math.Round(z1, 4) + "  El resultado de otro posible talud es:  " + Math.Round(z2, 4) + "  ¿Desea utilizar el primer resultado?", "Elección de Talud", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                    if (vent == DialogResult.Yes)// si pulsa Sí, usaremos Z1
                    {
                        z2 = z1;
                        anchosup = bas + tirh * z1 + tirh * z2;
                        area = (bas + (z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                        perimetro = bas + tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                        profhid = area / anchosup;
                        factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                        t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                        t_area.Text = Convert.ToString(Math.Round(area, 6));
                        t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                        t_z1.Text = Convert.ToString(Math.Round(z1, 6));
                        t_z2.Text = Convert.ToString(Math.Round(z2, 6));
                        t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                        t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                    }
                    if (vent == DialogResult.No)// si pulsa No, usaremos Z2
                    {
                        z1 = z2;
                        anchosup = bas + tirh * z1 + tirh * z2;
                        area = (bas + (z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                        perimetro = bas + tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                        profhid = area / anchosup;
                        factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                        t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                        t_area.Text = Convert.ToString(Math.Round(area, 6));
                        t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                        t_z1.Text = Convert.ToString(Math.Round(z1, 6));
                        t_z2.Text = Convert.ToString(Math.Round(z2, 6));
                        t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                        t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                    }
                }

            }

            if (t_base.Text != "" && t_area.Text != "" && t_z1.Text != "" && t_z2.Text != "" && t_tirantehid.Text == "" && t_RadioH.Text == "" && t_anchosup.Text == "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "")
                //caso donde conocemos base, área  y taludes
            {
                bas = Convert.ToDouble(t_base.Text);
                area = Convert.ToDouble(t_area.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
                tirh = (1.41421356 * (Math.Sqrt(area * (z1 + z2) + 0.5 * bas * bas) - 0.707106781*bas)) / (z1 + z2);
                perimetro = bas + tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                radiohid = area / perimetro;
                anchosup = bas + tirh * z1 + tirh * z2;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));

            }
            if (t_base.Text != "" && t_perimetrom.Text != "" && t_z1.Text != "" && t_z2.Text != "" && t_tirantehid.Text == "" && t_RadioH.Text == "" && t_anchosup.Text == "" && t_factz.Text == "" && t_area.Text == "" && t_profhid.Text == "")
                //caso donde conocemos base, perímetro  y taludes
            {
                bas = Convert.ToDouble(t_base.Text);
                perimetro = Convert.ToDouble(t_perimetrom.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
```

```
                tirh = (perimetro - bas) / (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));

                if (tirh <= 0)
                {
                    MessageBox.Show("El valor del perímetro propuesto nos arrojaría un tirante hidráulico negativo", "Error");
                }
                if (tirh > 0)
                {
                    area = (bas + (z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                    radiohid = area / perimetro;
                    anchosup = bas + tirh * z1 + tirh * z2;
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }
            }
            if (t_base.Text != "" && t_tirantehid.Text == "" && t_RadioH.Text != "" && t_anchosup.Text == "" && t_area.Text == "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && t_z1.Text != "" && t_z2.Text != "")
                //caso donde conocemos base, taludes y Radio hidráulico
            // Nota: se considerará que z1 y z2 pueden ser distintos
            {
                bas = Convert.ToDouble(t_base.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
                radiohid = Convert.ToDouble(t_RadioH.Text);
                tirh = (Math.Sqrt(Math.Pow((Math.Sqrt(z1 * z1 + 1) + Math.Sqrt(z2 * z2 + 1)), 2) * radiohid * radiohid - 2 * bas * (Math.Sqrt(z1 * z1 + 1)-z1 + Math.Sqrt(z2 * z2 + 1)-z2)*radiohid+bas*bas) + (Math.Sqrt(z1 * z1 + 1) + Math.Sqrt(z2 * z2 + 1)) * radiohid - bas) / (z1 + z2);
                area = (bas + (z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                perimetro = bas + tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                anchosup = bas + tirh * z1 + tirh * z2;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
            }
            if (t_base.Text != "" && t_tirantehid.Text == "" && t_RadioH.Text == "" && t_anchosup.Text != "" && t_area.Text == "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && t_z1.Text != "" && t_z2.Text != "")
            //caso donde conocemos base, taludes y ancho superficial
            // Nota: se considerará que z1 y z2 pueden ser distintos
            {
                bas = Convert.ToDouble(t_base.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
                anchosup = Convert.ToDouble(t_anchosup.Text);
                tirh = (anchosup-bas)/(z1+z2);
                area = (bas + (z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                perimetro = bas + tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                radiohid = area / perimetro;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
            }
            if (t_area.Text != "" && t_base.Text == "" && t_tirantehid.Text != "" && t_z1.Text != "" && t_z2.Text != "" && t_perimetrom.Text=="" && t_RadioH.Text=="" && t_anchosup.Text=="" && t_profhid.Text== "" && t_factz.Text=="")
                //caso donde conocemos area, tirante y taludes
```

```
            {
                area = Convert.ToDouble(t_area.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
                bas = (area / tirh) - 0.5 * tirh * (z1 + z2);
                perimetro = bas + tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                radiohid = area / perimetro;
                anchosup = bas + tirh * z1 + tirh * z2;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_base.Text = Convert.ToString(Math.Round(bas, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));

            }
            if (t_area.Text == "" && t_base.Text == "" && t_tirantehid.Text != "" && t_z1.Text != "" &&
t_z2.Text != "" && t_perimetrom.Text != "" && t_RadioH.Text == "" && t_anchosup.Text == "" && t_factz.
Text == "" && t_profhid.Text == "")
                //caso donde conocemos perímetro, tirante y taludes
            {
                perimetro = Convert.ToDouble(t_perimetrom.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
                bas = perimetro - (tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2));

                if (bas <= 0)
                {
                    MessageBox.Show("El valor del perímetro propuesto nos arrojaría una base negativa",
"Error");
                }

                if (bas > 0)
                {
                    area = (bas + (z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                    radiohid = area / perimetro;
                    anchosup = bas + tirh * z1 + tirh * z2;
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_base.Text = Convert.ToString(Math.Round(bas, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }
            }
            if (t_area.Text == "" && t_base.Text == "" && t_tirantehid.Text != "" && t_z1.Text != "" &&
t_z2.Text != "" && t_perimetrom.Text == "" && t_RadioH.Text != "" && t_anchosup.Text == "" && t_factz.
Text == "" && t_profhid.Text == "")
                //caso donde conocemos radio hidráulico, tirante y taludes
            {
                radiohid = Convert.ToDouble(t_RadioH.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
                bas = (-((Math.Sqrt(z1 * z1 + 1) + Math.Sqrt(z2 * z2 + 1)) * radiohid - 0.5 * (z1 + z2) *
 tirh) * tirh)/(radiohid-tirh);
                area = (bas + (z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                perimetro = bas + tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                anchosup = bas + tirh * z1 + tirh * z2;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_base.Text = Convert.ToString(Math.Round(bas, 6));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
```

```
            }
            if (t_area.Text == "" && t_base.Text == "" && t_tirantehid.Text != "" && t_z1.Text != "" &&
t_z2.Text != "" && t_perimetrom.Text == "" && t_RadioH.Text == "" && t_anchosup.Text != "" && t_factz.
Text == "" && t_profhid.Text == "")
            //caso donde conocemos ancho superficial, tirante y taludes
            {
                anchosup = Convert.ToDouble(t_anchosup.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
                bas = anchosup-tirh*(z1+z2);
                area = (bas + (z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                perimetro = bas + tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                radiohid = area / perimetro;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_base.Text = Convert.ToString(Math.Round(bas, 6));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));

            }
        }
        ////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////
        if (radiotriang.Checked == true) //es seleccionado el radiobutton de triángulo
        { //Aquí comienzan los cálculos para las secciones triangular
            if (t_tirantehid.Text != "" && t_z1.Text != "" && t_z2.Text != "" && t_RadioH.Text == "" &&
t_anchosup.Text == "" && t_area.Text == "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.
Text == "" )
            // caso donde conocemos taludes y tirante hidráulico
            {
                tirh = Convert.ToDouble(t_tirantehid.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
                area = ((z1 * tirh * 0.5) + (z2 * tirh * 0.5))*tirh;
                perimetro = tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                radiohid = area / perimetro;
                anchosup = tirh * z1 + tirh * z2;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
            }
            if (t_area.Text != "" && t_z1.Text != "" && t_z2.Text != "" && t_tirantehid.Text == "" &&
t_RadioH.Text == "" && t_anchosup.Text == "" && t_factz.Text == "" && t_perimetrom.Text == "" &&
t_profhid.Text == "")
            // caso donde conocemos taludes y área
            {
                area = Convert.ToDouble(t_area.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
                tirh = Math.Sqrt(area/(0.5*(z1+z2)));
                perimetro = tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                radiohid = area / perimetro;
                anchosup = tirh * z1 + tirh * z2;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
            }
            if (t_perimetrom.Text != "" && t_z1.Text != "" && t_z2.Text != "" && t_RadioH.Text == "" &&
t_anchosup.Text == "" && t_area.Text == "" && t_factz.Text == "" && t_profhid.Text == "" && t_tirantehid.
Text == "")
```

```
                // caso donde conocemos taludes y perímetro
            {
                perimetro = Convert.ToDouble(t_perimetrom.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
                tirh = perimetro / (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                area = ((z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                radiohid = area / perimetro;
                anchosup = tirh * z1 + tirh * z2;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
            }
            if (t_RadioH.Text != "" && t_z1.Text != "" && t_z2.Text != "" && t_anchosup.Text == "" &&
t_area.Text == "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" &&
t_tirantehid.Text == "")
                // caso donde conocemos taludes y radio hidáulico
            {
                radiohid  = Convert.ToDouble(t_RadioH.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
                tirh = (radiohid*(Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2))) /(0.5*z1+0.5*z2) ;
                area = ((z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                perimetro = tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                anchosup = tirh * z1 + tirh * z2;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
            }
            if (t_anchosup.Text != "" && t_z1.Text != "" && t_z2.Text != "" && t_RadioH.Text == "" &&
t_area.Text == "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" &&
t_tirantehid.Text == "")
                // caso donde conocemos taludes y ancho superficial
            {
                anchosup = Convert.ToDouble(t_anchosup.Text);
                z1 = Convert.ToDouble(t_z1.Text);
                z2 = Convert.ToDouble(t_z2.Text);
                tirh = anchosup/(z1+z2);
                area = ((z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                perimetro = tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                radiohid = area / perimetro;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
            }
            if (t_area.Text != "" && t_tirantehid.Text != "" && t_z1.Text == "" && t_z2.Text == "" &&
t_RadioH.Text == "" && t_anchosup.Text == "" && t_factz.Text == "" && t_perimetrom.Text == "" &&
t_profhid.Text == "")
                // caso donde conocemos tirante hidraulico y área
            //Nota: se consideran igual los taludes
            {
                area = Convert.ToDouble(t_area.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                z1 = area / (tirh*tirh);
                z2 = z1;
                perimetro = tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                radiohid = area / perimetro;
                anchosup = tirh * z1 + tirh * z2;
                profhid = area / anchosup;
```

```
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_z1.Text = Convert.ToString(Math.Round(z1, 6));
                t_z2.Text = Convert.ToString(Math.Round(z2, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
            }
            if (t_perimetrom.Text != "" && t_tirantehid.Text != "" && t_z1.Text == "" && t_z2.Text == "" ↙
    && t_RadioH.Text == "" && t_anchosup.Text == "" && t_factz.Text == "" && t_area.Text=="" && t_profhid.   ↙
    Text == "")
                // caso donde conocemos tirante hidraulico y perímetro
            //Nota: se consideran igual los taludes
            {
                perimetro = Convert.ToDouble(t_perimetrom.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);

                if ((Math.Pow(perimetro / (2 * tirh), 2) - 1)<=0)
                {
                    MessageBox.Show("Los valores ingresados nos darían un talud con un valor imaginario o↙
     igual a cero", "Error");
                }

                if ((Math.Pow(perimetro / (2 * tirh), 2) - 1) > 0)
                {

                    z1 = Math.Sqrt(Math.Pow(perimetro / (2 * tirh), 2) - 1);
                    z2 = z1;
                    area = ((z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                    radiohid = area / perimetro;
                    anchosup = tirh * z1 + tirh * z2;
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_z1.Text = Convert.ToString(Math.Round(z1, 6));
                    t_z2.Text = Convert.ToString(Math.Round(z2, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }
            }
            if (t_RadioH.Text != "" && t_tirantehid.Text != "" && t_z1.Text == "" && t_z2.Text == "" &&  ↙
    t_anchosup.Text == "" && t_factz.Text == "" && t_area.Text == "" && t_perimetrom.Text == "" && t_profhid.↙
    Text == "")
                // caso donde conocemos tirante hidraulico y radio hidráulico
            //Nota: se consideran igual los taludes
            {
                radiohid = Convert.ToDouble(t_RadioH.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);

                if ((1 - Math.Pow((2 * radiohid / tirh), 2))<=0)
                {
                    MessageBox.Show("Los valores ingresados nos darían un talud con un valor imaginario o↙
     igual a cero", "Error");
                }

                if ((1 - Math.Pow((2 * radiohid / tirh), 2)) > 0)
                {
                    z1 = Math.Sqrt(Math.Pow((2 * radiohid / tirh), 2) / (1 - Math.Pow((2 * radiohid /    ↙
    tirh), 2)));
                    z2 = z1;
                    area = ((z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                    perimetro = tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                    anchosup = tirh * z1 + tirh * z2;
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_z1.Text = Convert.ToString(Math.Round(z1, 6));
                    t_z2.Text = Convert.ToString(Math.Round(z2, 6));
                    t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
```

```
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }
            }
            if (t_anchosup.Text != "" && t_tirantehid.Text != "" && t_z1.Text == "" && t_z2.Text == "" && t_RadioH.Text == "" && t_factz.Text == "" && t_area.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "")
                // caso donde conocemos tirante hidraulico y ancho superficial
            //Nota: se consideran igual los taludes
            {
                anchosup = Convert.ToDouble(t_anchosup.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                z1 = anchosup/(2*tirh);
                z2 = z1;
                area = ((z1 * tirh * 0.5) + (z2 * tirh * 0.5)) * tirh;
                perimetro = tirh * Math.Sqrt(1 + z1 * z1) + tirh * Math.Sqrt(1 + z2 * z2);
                radiohid = area / perimetro;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_z1.Text = Convert.ToString(Math.Round(z1, 6));
                t_z2.Text = Convert.ToString(Math.Round(z2, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
            }
        }
        ////////////////////////////////////////////////////////////////////////////////////////////////////
        if (radiocirculo.Checked == true) //es seleccionado el radiobutton de círculo
        { //Aquí comienzan los cálculos para la sección circular
            if (t_diam.Text != "" && t_tirantehid.Text != "" && t_RadioH.Text == "" && t_factz.Text == "" && t_area.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && t_anchosup.Text == "")
                //se conoce diametro y tirante hidráulico
            {
                Diam = Convert.ToDouble(t_diam.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);

                if (tirh > Diam)

                {
                    MessageBox.Show("El valor del tirante hidráulico debe ser menor o igual al valor del diametro", "Error");
                }

                if (tirh <= Diam)
                {
                    radio = Diam * 0.5;
                    tetha = 2 * Math.Acos(1 - (tirh / radio));
                    area = ((tetha - Math.Sin(tetha)) * Diam * Diam) / 8;
                    perimetro = 0.5 * tetha * Diam;
                    radiohid = area / perimetro;
                    anchosup = 2 * Math.Sqrt(tirh * (Diam - tirh));
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                    t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }

            }
            if (t_area.Text != "" && t_tirantehid.Text != "" && t_RadioH.Text == "" && t_diam.Text == "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && t_anchosup.Text == "")
            //se conoce tirante hidráulico y área
            {
                area = Convert.ToDouble(t_area.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                /* para hallar el valor de theta usaremos un método iterativo (método de bisección) y conociendo tetha podemos
                 conocer el diametro*/
```

```
                //se lleva a cabo la bisección

                while (!(((((x3 - Math.Sin(x3)) * Math.Pow((-2 * tirh / (Math.Cos(0.5 * x3) - 1)), 2)) / 8) - area) == 0.0 && Math.Abs(x1 - x2) < epsilon))
                {
                    x3 = (x1 + x2) * 0.5;
                    if (((((x1 - Math.Sin(x1)) * Math.Pow((-2 * tirh / (Math.Cos(0.5 * x1) - 1)), 2)) / 8) - area) * ((((x3 - Math.Sin(x3)) * Math.Pow((-2 * tirh / (Math.Cos(0.5 * x3) - 1)), 2)) / 8) - area) < 0) //(((((x3 - Math.Sin(x3)) * Math.Pow((-2 * y / (Math.Cos(0.5 * x3) - 1)), 2)) / 8) - a) < 0)
                    {
                        x2 = x3;
                    }
                    if (((((x1 - Math.Sin(x1)) * Math.Pow((-2 * tirh/ (Math.Cos(0.5 * x1) - 1)), 2)) / 8) - area) * ((((x3 - Math.Sin(x3)) * Math.Pow((-2 * tirh / (Math.Cos(0.5 * x3) - 1)), 2)) / 8) - area) > 0)// ((((x3 - Math.Sin(x3)) * Math.Pow((-2 * y / (Math.Cos(0.5 * x3) - 1)), 2)) / 8) - a) > 0)
                    {
                        x1 = x3;
                    }

                    conteo++;

                    if (conteo > n)
                    {
                        break;
                    }


                }

                tetha = x3;
                Diam = Math.Sqrt(8 * area / (tetha - Math.Sin(tetha)));
                if (Diam < tirh)

                {
                    MessageBox.Show("Los valores ingresados nos arrojan un Diametro menor que nuestro tirante hidráulico", "Error");
                }

                if (Diam >= tirh)
                {
                    perimetro = 0.5 * tetha * Diam;
                    radiohid = area / perimetro;
                    anchosup = 2 * Math.Sqrt(tirh * (Diam - tirh));
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_diam.Text = Convert.ToString(Math.Round(Diam, 6));
                    t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                    t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6)); ;
                }

            }
            if (t_area.Text == "" && t_tirantehid.Text != "" && t_RadioH.Text == "" && t_factz.Text == "" && t_perimetrom.Text != "" && t_profhid.Text == "" && t_anchosup.Text == "" && t_diam.Text == "")
            //se conoce tirante hidráulico y perímetro
            {
                perimetro = Convert.ToDouble(t_perimetrom.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                /* Para este caso es común que existan 2 valores de theta que satisfagan el valor de la ecuación
                por ende existen dos valores de diámetro posible */

                    // Se itera de 0 a 2pi para conocer un posible valor de tetha

                while (!(((0.5 * x3 * (-2 * tirh / (Math.Cos(0.5 * x3) - 1))) - perimetro) == 0.0 && Math.Abs(x1 - x4) < epsilon))
                {
                    x3 = (x1 + x4) * 0.5;
                    if (((0.5 * x1 * (-2 * tirh / (Math.Cos(0.5 * x1) - 1))) - perimetro) * ((0.5 * x3 * (-2 * tirh / (Math.Cos(0.5 * x3) - 1))) - perimetro) < 0) // p=1/2tetha*d de aquí igualamos a cero, d está en función de tetha
```

```
                        {
                            x4 = x3;
                        }
                        if (((0.5 * x1 * (-2 * tirh / (Math.Cos(0.5 * x1) - 1))) - perimetro) * ((0.5 * x3 * (-2 * tirh / (Math.Cos(0.5 * x3) - 1))) - perimetro) > 0)
                        {
                            x1 = x3;
                        }

                        conteo++;

                        if (conteo > n)
                        {
                            break;
                        }


                    }

                    tetha = x3;
                    Diam = (-2 * tirh) / (Math.Cos(0.5 * tetha) - 1);
                    x4 = x3+0.00000001;

                    /*se lleva a cabo la bisección para un rango entre el valor que encontramos en la primera iteración
                     hasta 2pi*/

                    while (!(((0.5 * x5 * (-2 * tirh / (Math.Cos(0.5 * x5) - 1))) - perimetro) == 0.0 && Math.Abs(x4 - x2) < epsilon))
                    {
                        x5 = (x4 + x2) * 0.5;
                        if (((0.5 * x4 * (-2 * tirh / (Math.Cos(0.5 * x4) - 1))) - perimetro) * ((0.5 * x5 * (-2 * tirh / (Math.Cos(0.5 * x5) - 1))) - perimetro) < 0) // p=1/2tetha*d de aquí igualamos a cero, d está en función de tetha
                        {
                            x2 = x5;
                        }
                        if (((0.5 * x4 * (-2 * tirh / (Math.Cos(0.5 * x4) - 1))) - perimetro) * ((0.5 * x5 * (-2 * tirh / (Math.Cos(0.5 * x5) - 1))) - perimetro) > 0)// ((((x3 - Math.Sin(x3)) * Math.Pow((-2 * y / (Math.Cos(0.5 * x3) - 1)), 2)) / 8) - a) > 0)
                        {
                            x4 = x5;
                        }

                        conteo2++;

                        if (conteo2 > n)
                        {
                            break;
                        }


                    }
                    tetha2 = x5;
                    Diam2 = (-2 * tirh) / (Math.Cos(0.5 * tetha2) - 1);

                    DialogResult vent = MessageBox.Show("El resultado de un posible Diámetro es:  " + Math.Round(Diam, 6) + "  El resultado de otro posible Diámetro es:  " + Math.Round(Diam2, 6) + "  ¿Desea utilizar el primer resultado?", "Elección de Diámetro", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                    if (vent == DialogResult.Yes)// si pulsa Sí, usaremos Diam
                    {

                        area = ((tetha - Math.Sin(tetha)) * Diam * Diam) / 8;
                        radiohid = area / perimetro;
                        anchosup = 2 * Math.Sqrt(tirh * (Diam - tirh));
                        profhid = area / anchosup;
                        factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                        t_diam.Text = Convert.ToString(Math.Round(Diam, 6));
                        t_area.Text = Convert.ToString(Math.Round(area, 6));
                        t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                        t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                        t_profhid.Text = Convert.ToString(Math.Round(profhid,6));
                        t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
```

```
                    }

                    if (vent == DialogResult.No)// si pulsa no, usaremos Diam2
                    {

                        area = ((tetha2 - Math.Sin(tetha2)) * Diam2 * Diam2) / 8;
                        radiohid = area / perimetro;
                        anchosup = 2 * Math.Sqrt(tirh * (Diam2 - tirh));
                        profhid = area / anchosup;
                        factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                        t_diam.Text = Convert.ToString(Math.Round(Diam, 6));
                        t_area.Text = Convert.ToString(Math.Round(area, 6));
                        t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                        t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                        t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                        t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                    }




            }
            if (t_area.Text == "" && t_tirantehid.Text != "" && t_RadioH.Text != "" && t_diam.Text == "" ↙
    && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && t_anchosup.Text == "")
            //se conoce tirante hidráulico y radio hidráulico
            {
                radiohid = Convert.ToDouble(t_RadioH.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);


                if (radiohid < 0.25 * tirh || radiohid > ((2 * tirh) / 3))
                {
                    MessageBox.Show("El valor del Radio hidráulico debe ser mayor o igual a y/4 pero     ↙
    menor o igual a 2*y/3","Error");
                }


                if (radiohid >= 0.25 * tirh && radiohid <= ((2 * tirh) / 3))
                {


                    /* para hallar el valor de theta usaremos un método iterativo (método de bisección) y↙
     conociendo tetha podemos
                     conocer el diametro*/


                    //se lleva a cabo la bisección

                    while (!(((0.25 * (1 - (Math.Sin(x3)) / x3) * (-2 * tirh / (Math.Cos(0.5 * x3) - 1)))↙
     - radiohid) == 0.0 && Math.Abs(x1 - x2) < epsilon))
                    {
                        x3 = (x1 + x2) * 0.5;
                        if (((0.25 * (1 - (Math.Sin(x1)) / x1) * (-2 * tirh / (Math.Cos(0.5 * x1) - 1))) ↙
    - radiohid) * ((0.25 * (1 - (Math.Sin(x3)) / x3) * (-2 * tirh / (Math.Cos(0.5 * x3) - 1))) - radiohid) < ↙
    0) // p=1/2tetha*d de aquí igualamos a cero, d está en función de tetha
                        {
                            x2 = x3;
                        }
                        if (((0.25 * (1 - (Math.Sin(x1)) / x1) * (-2 * tirh / (Math.Cos(0.5 * x1) - 1))) ↙
    - radiohid) * ((0.25 * (1 - (Math.Sin(x3)) / x3) * (-2 * tirh / (Math.Cos(0.5 * x3) - 1))) - radiohid) > ↙
    0)// ((((x3 - Math.Sin(x3)) * Math.Pow((-2 * y / (Math.Cos(0.5 * x3) - 1)), 2)) / 8) - a) > 0)
                        {
                            x1 = x3;
                        }

                        conteo++;

                        if (conteo > n)
                        {
                            break;
                        }
```

```
                    }

                    tetha = x3;
                    Diam = (-2 * tirh) / (Math.Cos(0.5 * tetha) - 1);
                    area = ((tetha - Math.Sin(tetha)) * Diam * Diam) / 8;
                    perimetro = 0.5 * tetha * Diam;
                    anchosup = 2 * Math.Sqrt(tirh * (Diam - tirh));
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_diam.Text = Convert.ToString(Math.Round(Diam, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid,6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ,6)); ;
                }
            }
            if (t_area.Text == "" &&t_anchosup.Text!="" && t_tirantehid.Text != "" && t_RadioH.Text == ""↙
 && t_diam.Text == "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "")
            //se conoce tirante hidráulico y ancho superfial
            {
                anchosup = Convert.ToDouble(t_anchosup.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                /* para hallar el valor de theta usaremos un método iterativo (método de bisección) y    ↙
conociendo tetha podemos
                 conocer el diametro*/


                //se lleva a cabo la bisección

                while (!(((Math.Sin(0.5*x3)) * (-2 * tirh / (Math.Cos(0.5 * x3) - 1))) - anchosup) == 0.↙
0 && Math.Abs(x1 - x2) < epsilon))
                {
                    x3 = (x1 + x2) * 0.5;
                    if ((((Math.Sin(0.5 * x1)) * (-2 * tirh / (Math.Cos(0.5 * x1) - 1))) - anchosup) * ((↙
(Math.Sin(0.5 * x3)) * (-2 * tirh / (Math.Cos(0.5 * x3) - 1))) - anchosup) < 0) // T=(sen(1/2tetha)*d de ↙
aquí igualamos a cero, d está en función de tetha
                    {
                        x2 = x3;
                    }
                    if ((((Math.Sin(0.5 * x1)) * (-2 * tirh / (Math.Cos(0.5 * x1) - 1))) - anchosup) * ((↙
(Math.Sin(0.5 * x3)) * (-2 * tirh / (Math.Cos(0.5 * x3) - 1))) - anchosup) > 0)//
                    {
                        x1 = x3;
                    }

                    conteo++;

                    if (conteo > n)
                    {
                        break;
                    }


                }

                tetha = x3;
                Diam = (-2 * tirh) / (Math.Cos(0.5 * tetha) - 1);
                area = ((tetha - Math.Sin(tetha)) * Diam * Diam) / 8;
                perimetro = 0.5 * tetha * Diam;
                radiohid = area / perimetro;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_diam.Text = Convert.ToString(Math.Round(Diam, 6));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6)); ;

            }
            if (t_area.Text != "" && t_tirantehid.Text == "" && t_RadioH.Text == "" && t_diam.Text != "" ↙
&& t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && t_anchosup.Text == "")
            //se conoce diametro y área
```

```
            {
                area = Convert.ToDouble(t_area.Text);
                Diam = Convert.ToDouble(t_diam.Text);


                if (area > 0.25 * (Math.PI * Diam * Diam))
                {
                    MessageBox.Show("El área propuesta es mayor que el área que se puede obtener ↙
considerando un tirante igual al diámetro","Error");
                }

                if (area <= 0.25 * (Math.PI * Diam * Diam))
                {

                    /* para hallar el valor de theta usaremos un método iterativo (método de bisección) y↙
 conociendo tetha podemos
                     conocer todas las propiedades*/


                    //se lleva a cabo la bisección

                    while (!(((((x3 - Math.Sin(x3)) * Diam * Diam) / 8) - area) == 0.0 && Math.Abs(x1 - ↙
x2) < epsilon))
                    {
                        x3 = (x1 + x2) * 0.5;
                        if (((((x1 - Math.Sin(x1)) * Diam * Diam) / 8) - area) * ((((x3 - Math.Sin(x3)) *↙
 Diam * Diam) / 8) - area) < 0) //
                        {
                            x2 = x3;
                        }
                        if (((((x1 - Math.Sin(x1)) * Diam * Diam) / 8) - area) * ((((x3 - Math.Sin(x3)) *↙
 Diam * Diam) / 8) - area) > 0)//
                        {
                            x1 = x3;
                        }

                        conteo++;

                        if (conteo > n)
                        {
                            break;
                        }


                    }

                    tetha = x3;
                    tirh = 0.5 * Diam * (1 - Math.Cos(0.5 * tetha));
                    perimetro = 0.5 * tetha * Diam;
                    radiohid = area / perimetro;
                    anchosup = 2 * Math.Sqrt(tirh * (Diam - tirh));
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                    t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                    t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6)); ;

                }
            }
            if (t_area.Text == "" && t_tirantehid.Text == "" && t_RadioH.Text == "" && t_diam.Text != "" ↙
&& t_factz.Text == "" && t_perimetrom.Text != "" && t_profhid.Text == "" && t_anchosup.Text == "")
            //se conoce diametro y perímetro
            {
                perimetro = Convert.ToDouble(t_perimetrom.Text);
                Diam = Convert.ToDouble(t_diam.Text);
                /* para hallar el valor de theta se puede despejar analíticamente
                 */

                if (perimetro > Diam * Math.PI)
                {
                    MessageBox.Show("EL perímetro ingresado es mayor que el perímetro que podemos obtener↙
```

```
     considerando toda la sección","Error");
                    }


                    tetha = (2*perimetro)/Diam;
                    tirh = 0.5 * Diam * (1 - Math.Cos(0.5 * tetha));
                    area = ((tetha - Math.Sin(tetha)) * Diam * Diam) / 8;
                    radiohid = area / perimetro;
                    anchosup = 2 * Math.Sqrt(tirh * (Diam - tirh));
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));


                }
                if (t_area.Text == "" && t_tirantehid.Text == "" && t_RadioH.Text != "" && t_diam.Text != "" ↙
    && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && t_anchosup.Text=="")
                //se conoce diametro y Radio hidráulico
                {
                    radiohid = Convert.ToDouble(t_RadioH.Text);
                    Diam = Convert.ToDouble(t_diam.Text);
                    /* para hallar el valor de theta usaremos un método iterativo (método de bisección) y    ↙
    conociendo tetha podemos
                     conocer todas las propiedades*/


                    //se lleva a cabo la bisección de 0 a 2pi

                    /*while (!(((0.25 * (1 - (Math.Sin(x3) / (x3))) * Diam) - radiohid) == 0.0 && Math.Abs(x1↙
     - x2) < epsilon))
                    {
                        x3 = (x1 + x2) * 0.5;
                        if (((0.25 * (1 - (Math.Sin(x1) / (x1))) * Diam) - radiohid) * ((0.25 * (1 - (Math.  ↙
    Sin(x3) / (x3))) * Diam) - radiohid) < 0) //
                        {
                            x2 = x3;
                        }
                        if (((0.25 * (1 - (Math.Sin(x1) / (x1))) * Diam) - radiohid) * ((0.25 * (1 - (Math.  ↙
    Sin(x3) / (x3))) * Diam) - radiohid) > 0)//
                        {
                            x1 = x3;
                        }

                        conteo++;

                        if (conteo > n)
                        {
                            break;
                        }


                    }

                    tetha = x3;
                    tirh = 0.5 * Diam * (1 - Math.Cos(0.5 * tetha));

                    x2 = x3+0.0000000001;*/

                    /*se lleva a cabo la bisección para un rango entre el valor que encontramos en la primera↙
     iteración
                         hasta 2pi*/

                   /* while (!(((0.25 * (1 - (Math.Sin(x5) / (x5))) * Diam) - radiohid) == 0.0 && Math.Abs(x2↙
     - x4) < epsilon))
                    {
                        x5 = (x2 + x4) * 0.5;
                        if (((0.25 * (1 - (Math.Sin(x2) / (x2))) * Diam) - radiohid) * ((0.25 * (1 - (Math.  ↙
    Sin(x5) / (x5))) * Diam) - radiohid) < 0) //
                        {
```

```
                        x4 = x5;
                    }
                    if (((0.25 * (1 - (Math.Sin(x2) / (x2))) * Diam) - radiohid) * ((0.25 * (1 - (Math.
Sin(x5) / (x5))) * Diam) - radiohid) > 0)//
                    {
                        x2 = x5;
                    }

                    conteo2++;

                    if (conteo2 > n)
                    {
                        break;
                    }


                }


                tetha2 = x5;
                tirh2 = 0.5 * Diam * (1 - Math.Cos(0.5 * tetha2));*/
                x2 = 4;
                /////////////////
                while (!(((0.25 * (1 - (Math.Sin(2 * Math.Acos(1 - x3 / (0.5 * Diam))) / (2 * Math.Acos(1
 - x3 / (0.5 * Diam))))) * Diam) - radiohid) == 0.0 && Math.Abs(x1 - x2) < 0.000001))
                {
                    x3 = (x1 + x2) * 0.5;
                    if (((0.25 * (1 - (Math.Sin(2 * Math.Acos(1 - x1 / (0.5 * Diam))) / (2 * Math.Acos(1
- x1 / (0.5 * Diam))))) * Diam) - radiohid) * ((0.25 * (1 - (Math.Sin(2 * Math.Acos(1 - x3 / (0.5 *
Diam))) / (2 * Math.Acos(1 - x3 / (0.5 * Diam))))) * Diam) - radiohid) < 0) //
                    {
                        x2 = x3;
                    }
                    if (((0.25 * (1 - (Math.Sin(2 * Math.Acos(1 - x1 / (0.5 * Diam))) / (2 * Math.Acos(1
- x1 / (0.5 * Diam))))) * Diam) - radiohid) * ((0.25 * (1 - (Math.Sin(2 * Math.Acos(1 - x3 / (0.5 *
Diam))) / (2 * Math.Acos(1 - x3 / (0.5 * Diam))))) * Diam) - radiohid) > 0) //
                    {
                        x1 = x3;
                    }

                    conteo++;

                    if (conteo > n)
                    {
                        break;
                    }


                }


                tirh = x3;

                x2 = x3 + 0.0000000001;
                x4 = Diam;
                /*se lleva a cabo la bisección para un rango entre el valor que encontramos en la primera
 iteración
                     hasta 2pi*/

                while (!(((0.25 * (1 - (Math.Sin(2 * Math.Acos(1 - x5 / (0.5 * Diam))) / (2 * Math.Acos(1
 - x5 / (0.5 * Diam))))) * Diam) - radiohid) == 0.0 && Math.Abs(x2 - x4) < 0.000001))
                {
                    x5 = (x2 + x4) * 0.5;
                    if (((0.25 * (1 - (Math.Sin(2 * Math.Acos(1 - x2 / (0.5 * Diam))) / (2 * Math.Acos(1
- x2 / (0.5 * Diam))))) * Diam) - radiohid) * ((0.25 * (1 - (Math.Sin(2 * Math.Acos(1 - x5 / (0.5 *
Diam))) / (2 * Math.Acos(1 - x5 / (0.5 * Diam))))) * Diam) - radiohid) < 0) //
                    {
                        x4 = x5;
                    }
                    if (((0.25 * (1 - (Math.Sin(2 * Math.Acos(1 - x2 / (0.5 * Diam))) / (2 * Math.Acos(1
- x2 / (0.5 * Diam))))) * Diam) - radiohid) * ((0.25 * (1 - (Math.Sin(2 * Math.Acos(1 - x5 / (0.5 *
Diam))) / (2 * Math.Acos(1 - x5 / (0.5 * Diam))))) * Diam) - radiohid) > 0)//
                    {
                        x2 = x5;
```

```
                    }

                    conteo2++;

                    if (conteo2 > n)
                    {
                        break;
                    }


                }



                tirh2 = x5;
                ////////////////

                DialogResult vent = MessageBox.Show("El resultado de un posible tirante hidráulico es:  "↙
 + Math.Round(tirh, 6) + "  El resultado de otro posible tirante hidráulico es:  " + Math.Round(tirh2, 6)↙
 + "  ¿Desea utilizar el primer resultado?", "Elección de tirante hidráulico", MessageBoxButtons.YesNo, ↙
MessageBoxIcon.Question);
                if (vent == DialogResult.Yes)// si pulsa Sí, usaremos tirh
                {

                    tetha = (2 * Math.Acos(1 - tirh / (0.5 * Diam)));
                    perimetro = 0.5 * tetha * Diam;
                    area = ((tetha - Math.Sin(tetha)) * Diam * Diam) / 8;
                    anchosup = 2 * Math.Sqrt(tirh * (Diam - tirh));
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                    t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }

                if (vent == DialogResult.No)// si pulsa No, usaremos tirh2
                {

                    tetha2 = (2 * Math.Acos(1 - tirh2 / (0.5 * Diam)));
                    perimetro = 0.5 * tetha2 * Diam;
                    area = ((tetha2 - Math.Sin(tetha2)) * Diam * Diam) / 8;
                    anchosup = 2 * Math.Sqrt(tirh2 * (Diam - tirh2));
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_tirantehid.Text = Convert.ToString(Math.Round(tirh2, 6));
                    t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }

            }
            if (t_area.Text == "" && t_anchosup.Text !="" && t_tirantehid.Text == "" && t_RadioH.Text == ↙
"" && t_diam.Text != "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "")
            //se conoce diametro y ancho superficial
            {
                anchosup = Convert.ToDouble(t_anchosup.Text);
                Diam = Convert.ToDouble(t_diam.Text);
                tirh = -(Math.Sqrt(Diam * Diam - anchosup * anchosup) - Diam) * 0.5;
                tirh2 = (Math.Sqrt(Diam * Diam - anchosup * anchosup) + Diam) * 0.5;

                DialogResult vent = MessageBox.Show("El resultado de un posible tirante hidráulico es:  "↙
 + Math.Round(tirh, 4) + "  El resultado de otro posible tirante hidráulico es:  " + Math.Round(tirh2, 4)↙
 + "  ¿Desea utilizar el primer resultado?", "Elección de tirante", MessageBoxButtons.YesNo, ↙
MessageBoxIcon.Question);
                if (vent == DialogResult.Yes)// si pulsa Sí, usaremos tirh
                {
                    tetha = 2 * Math.Acos(1 - (tirh / (0.5 * Diam)));
                    perimetro = 0.5 * tetha * Diam;
                    area = ((tetha - Math.Sin(tetha)) * Diam * Diam) / 8;
```

```
                    radiohid = area / perimetro;
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
                    t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6)); ;
                }
                if (vent == DialogResult.No)// si pulsa No, usaremos tirh2
                {
                    tetha = 2 * Math.Acos(1 - (tirh2 / (0.5 * Diam)));
                    perimetro = 0.5 * tetha * Diam;
                    area = ((tetha - Math.Sin(tetha)) * Diam * Diam) / 8;
                    radiohid = area / perimetro;
                    profhid = area / anchosup;
                    factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                    t_tirantehid.Text = Convert.ToString(Math.Round(tirh2, 6));
                    t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                    t_area.Text = Convert.ToString(Math.Round(area, 6));
                    t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                    t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                    t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                }

            }
        }
         ////////////////////////////////////////////////////////////////////////////////////////////////↙
//////////
        if (radioparabola.Checked == true) //es seleccionado el radiobutton de parábola
        { //Aquí comienzan los cálculos para la sección parabólica
            if (t_anchosup.Text != "" && t_tirantehid.Text != "" && t_area.Text == "" && t_RadioH.Text ==↙
 "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && textk.Text=="")
            { // Conocido tirante hidráulico y ancho superficial
                anchosup = Convert.ToDouble(t_anchosup.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                area = 2 * anchosup * tirh / 3;
                perimetro = 0;
                x = 4 * tirh / anchosup;
                if (x>0 && x<=1)
                {
                    perimetro = anchosup + (8 * tirh * tirh) / (3 * anchosup);
                }
                if (x > 1)
                {
                    perimetro = (anchosup * 0.5) * (Math.Sqrt(1 + x * x) + (1 / x) * Math.Log(x + Math.  ↙
Sqrt(1 + x * x)));
                }

                radiohid = area / perimetro;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                k = 4 * tirh / (anchosup * anchosup);
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                textk.Text = Convert.ToString(Math.Round(k, 6));
            }

            if (t_anchosup.Text == "" && t_tirantehid.Text != "" && t_area.Text == "" && t_RadioH.Text ==↙
 "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && textk.Text != "")
            { // Conocido constante k y tirante hidráulico
                k = Convert.ToDouble(textk.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                anchosup = Math.Sqrt(4 * tirh / k);
                area = 2 * anchosup * tirh / 3;
                perimetro = 0;
                x = 4 * tirh / anchosup;
                if (x > 0 && x <= 1)
                {
                    perimetro = anchosup + (8 * tirh * tirh) / (3 * anchosup);
```

```
                }
                if (x > 1)
                {
                    perimetro = (anchosup * 0.5) * (Math.Sqrt(1 + x * x) + (1 / x) * Math.Log(x + Math.
Sqrt(1 + x * x)));
                }

                radiohid = area / perimetro;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
            }

            if (t_anchosup.Text == "" && t_tirantehid.Text == "" && t_area.Text != "" && t_RadioH.Text ==
 "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && textk.Text != "")
            { // Conocido constante k y area
                k = Convert.ToDouble(textk.Text);
                area = Convert.ToDouble(t_area.Text);
                tirh = Math.Pow(k * 9 * area * area / 16, 1 / 3d);
                anchosup = Math.Sqrt(4 * tirh / k);
                perimetro = 0;
                x = 4 * tirh / anchosup;
                if (x > 0 && x <= 1)
                {
                    perimetro = anchosup + (8 * tirh * tirh) / (3 * anchosup);
                }
                if (x > 1)
                {
                    perimetro = (anchosup * 0.5) * (Math.Sqrt(1 + x * x) + (1 / x) * Math.Log(x + Math.
Sqrt(1 + x * x)));
                }

                radiohid = area / perimetro;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
            }

            if (t_anchosup.Text != "" && t_tirantehid.Text == "" && t_area.Text == "" && t_RadioH.Text ==
 "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && textk.Text != "")
            { // Conocido constante k y ancho superficial
                k = Convert.ToDouble(textk.Text);
                anchosup = Convert.ToDouble(t_anchosup.Text);
                tirh = Math.Pow(anchosup, 2) * k * 0.25;
                area = (2 / 3d) * anchosup * tirh;
                perimetro = 0;
                x = 4 * tirh / anchosup;
                if (x > 0 && x <= 1)
                {
                    perimetro = anchosup + (8 * tirh * tirh) / (3 * anchosup);
                }
                if (x > 1)
                {
                    perimetro = (anchosup * 0.5) * (Math.Sqrt(1 + x * x) + (1 / x) * Math.Log(x + Math.
Sqrt(1 + x * x)));
                }

                radiohid = area / perimetro;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_area.Text = Convert.ToString(Math.Round(area, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
```

```
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
            }

            if (t_anchosup.Text == "" && t_tirantehid.Text != "" && t_area.Text != "" && t_RadioH.Text ==↙
 "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && textk.Text == "")
            { // Conocido tirante hidráulico y area
                area = Convert.ToDouble(t_area.Text);
                tirh = Convert.ToDouble(t_tirantehid.Text);
                anchosup = (3 / 2d) * area / tirh;
                perimetro = 0;
                x = 4 * tirh / anchosup;
                if (x > 0 && x <= 1)
                {
                    perimetro = anchosup + (8 * tirh * tirh) / (3 * anchosup);
                }
                if (x > 1)
                {
                    perimetro = (anchosup * 0.5) * (Math.Sqrt(1 + x * x) + (1 / x) * Math.Log(x + Math. ↙
Sqrt(1 + x * x)));
                }

                radiohid = area / perimetro;
                profhid = area / anchosup;
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                k = 4 * tirh / (anchosup * anchosup);
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                t_anchosup.Text = Convert.ToString(Math.Round(anchosup, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                textk.Text = Convert.ToString(Math.Round(k, 6));
            }

            if (t_anchosup.Text != "" && t_tirantehid.Text == "" && t_area.Text != "" && t_RadioH.Text ==↙
 "" && t_factz.Text == "" && t_perimetrom.Text == "" && t_profhid.Text == "" && textk.Text == "")
            { // Conocido area y ancho superficial
                area = Convert.ToDouble(t_area.Text);
                anchosup = Convert.ToDouble(t_anchosup.Text);
                tirh = (3 / 2d) * area / anchosup;
                perimetro = 0;
                x = 4 * tirh / anchosup;
                if (x > 0 && x <= 1)
                {
                    perimetro = anchosup + (8 * tirh * tirh) / (3 * anchosup);
                }
                if (x > 1)
                {
                    perimetro = (anchosup * 0.5) * (Math.Sqrt(1 + x * x) + (1 / x) * Math.Log(x + Math. ↙
Sqrt(1 + x * x)));
                }

                radiohid = area / perimetro;
                profhid = area / anchosup;
                k = 4 * tirh / (anchosup * anchosup);
                factZ = (Math.Pow(area, 1.5)) / (Math.Pow(anchosup, 0.5));
                t_perimetrom.Text = Convert.ToString(Math.Round(perimetro, 6));
                textk.Text = Convert.ToString(Math.Round(k, 6));
                t_RadioH.Text = Convert.ToString(Math.Round(radiohid, 6));
                t_profhid.Text = Convert.ToString(Math.Round(profhid, 6));
                t_factz.Text = Convert.ToString(Math.Round(factZ, 6));
                t_tirantehid.Text = Convert.ToString(Math.Round(tirh, 6));
            }
        }
    }

    private void menu_Click(object sender, EventArgs e)
    {
        Form1 iniciar = new Form1();
        iniciar.Show();
        this.Hide();
    }

    private void comb_Click(object sender, EventArgs e)
```

```
        {
            combinaciones Iniciar = new combinaciones();
            Iniciar.Show();

            MessageBox.Show("A continuación se muestran las posibles combinaciones para ingresar datos de las↙
     distintas secciones", "Información", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
}
```

Código Submódulo Cálculo de tirante crítico

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Hidraulica_canales
{
    public partial class tirante_critico : Form
    {
        public tirante_critico()
        {
            InitializeComponent();
        }

        private void tirante_critico_FormClosing(object sender, FormClosingEventArgs e)
        {
            Application.Exit();
        }

        public void limpiar()
        {
            textQ.Clear();
            textB.Clear();
            textD.Clear();
            textT.Clear();
            textalp.Clear();
            textS.Clear();
            textZ1.Clear();
            textZ2.Clear();
            textA.Clear();
            textP.Clear();
            textRh.Clear();
            textTr.Clear();
            textV.Clear();
            textnf.Clear();
            textyc.Clear();
            textE.Clear();
        }

        private void radiorectangulo_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            lQ.Show();
            lB.Show();
            lD.Hide();
            lT.Hide();
            lalp.Show();
            lz1.Hide();
            lz2.Hide();
            textQ.Show();
            textB.Show();
            textD.Hide();
            textT.Hide();
            textalp.Show();
            textZ1.Hide();
            textZ2.Hide();

            lA.Show();
            lP.Show();
            lrh.Show();
            lTr.Show();
            lk.Hide();
            lv.Show();
            lfr.Show();
```

```
            lyc.Show();
            lE.Show();
            textA.Show();
            textP.Show();
            textRh.Show();
            textTr.Show();
            textV.Show();
            textnf.Show();
            textyc.Show();
            textE.Show();

            pictureBox1.Show();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.rectángulo;
        }

        private void radiotrapecio_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            lQ.Show();
            lB.Show();
            lD.Hide();
            lT.Hide();
            lalp.Show();
            lz1.Show();
            lz2.Show();
            textQ.Show();
            textB.Show();
            textD.Hide();
            textT.Hide();
            textalp.Show();
            textZ1.Show();
            textZ2.Show();

            lA.Show();
            lP.Show();
            lrh.Show();
            lTr.Show();
            lk.Hide();
            lv.Show();
            lfr.Show();
            lyc.Show();
            lE.Show();
            textA.Show();
            textP.Show();
            textRh.Show();
            textTr.Show();
            textV.Show();
            textnf.Show();
            textyc.Show();
            textE.Show();

            pictureBox1.Show();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.trapecio;
        }

        private void radiotriangulo_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            lQ.Show();
            lB.Hide();
            lD.Hide();
            lT.Hide();
            lalp.Show();
            lz1.Show();
            lz2.Show();
            textQ.Show();
            textB.Hide();
            textD.Hide();
            textT.Hide();
```

```
            textalp.Show();
            textZ1.Show();
            textZ2.Show();

            lA.Show();
            lP.Show();
            lrh.Show();
            lTr.Show();
            lk.Hide();
            lv.Show();
            lfr.Show();
            lyc.Show();
            lE.Show();
            textA.Show();
            textP.Show();
            textRh.Show();
            textTr.Show();
            textV.Show();
            textnf.Show();
            textyc.Show();
            textE.Show();

            pictureBox1.Show();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.triángulo;
        }

        private void radiocirculo_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            lQ.Show();
            lB.Hide();
            lD.Show();
            lT.Hide();
            lalp.Show();
            lz1.Hide();
            lz2.Hide();
            textQ.Show();
            textB.Hide();
            textD.Show();
            textT.Hide();
            textalp.Show();
            textZ1.Hide();
            textZ2.Hide();

            lA.Show();
            lP.Show();
            lrh.Show();
            lTr.Show();
            lk.Hide();
            lv.Show();
            lfr.Show();
            lyc.Show();
            lE.Show();
            textA.Show();
            textP.Show();
            textRh.Show();
            textTr.Show();
            textV.Show();
            textnf.Show();
            textyc.Show();
            textE.Show();

            pictureBox1.Show();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.círculo;
        }

        private void radioparabola_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
```

```
            lQ.Show();
            lB.Hide();
            lD.Hide();
            lT.Show();
            lalp.Show();
            lz1.Hide();
            lz2.Hide();
            textQ.Show();
            textB.Hide();
            textD.Hide();
            textT.Show();
            textalp.Show();
            textZ1.Hide();
            textZ2.Hide();

            lA.Show();
            lP.Show();
            lrh.Show();
            lTr.Hide();
            lk.Show();
            lv.Show();
            lfr.Show();
            lyc.Show();
            lE.Show();
            textA.Show();
            textP.Show();
            textRh.Show();
            textTr.Show();
            textV.Show();
            textnf.Show();
            textyc.Show();
            textE.Show();

            pictureBox1.Show();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.parábola;
        }

        private void info_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Si tu valor de coeficiente de coriolis (alpha) es 1, no es necesario ponerlo. Si el ↙
     valor de la pendiente es 0, no es necesario ponerlo", "Información",MessageBoxButtons.OK,MessageBoxIcon.    ↙
    Information);
        }

        private void buttonlimpiar_Click(object sender, EventArgs e)
        {
            limpiar();
        }

        private void buttoncalcular_Click(object sender, EventArgs e)
        {
            //Definimos nuestras variables
            double q, b, d, t, alp, z1, z2, a, p, rh, tr, k, v, nf, yc, E,ap,tp,cte,u,x1,x2,a1,tr1,o,s,ang;
            double g=9.81;
            double ea = 0;
            double xi = 1.0e-6;
            double x = 0,xold=0;
            int conteo=0,it=500;
            double epsilon=1.0e-8;

            if (radiorectangulo.Checked == true)
            {
                try
                {
                    q = Convert.ToDouble(textQ.Text);
                    b = Convert.ToDouble(textB.Text);

                    alp = 0;
                    if (textalp.Text == "")
```

```
                {
                    alp = 1;
                }
                else
                {
                    alp = Convert.ToDouble(textalp.Text);
                }

                s = 0;
                ang = 0;
                if (textS.Text == "")
                {
                    ang = 0;
                }

                if (textS.Text != "")
                {
                    s = Convert.ToDouble(textS.Text);
                    ang = Math.Atan(s);

                    if (ang >= 0 && ang <= (1 / 30d) * Math.PI)
                    {
                        ang = 0;
                    }

                    if (ang > (1 / 30d) * Math.PI)
                    {
                        ang = Math.Atan(s);
                    }
                }

                yc = Math.Pow(alp * q * q / (b * b * g*Math.Cos(ang)), 1 / 3d);
                a = b * yc;
                p = b + 2 * yc;
                rh = a / p;
                tr = b;
                v = q / a;
                nf = v / Math.Sqrt(g * (a / tr)*(Math.Cos(ang)/alp));
                E = yc*Math.Cos(ang) + alp * v * v / (2 * g);
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTr.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textnf.Text = Convert.ToString(Math.Round(nf, 4));
                textE.Text = Convert.ToString(Math.Round(E, 4));
                textyc.Text = Convert.ToString(Math.Round(yc, 4));
            }
            catch
            {
                MessageBox.Show("El cálculo es imposible ya que falta el valor del gasto o el valor de la base","Falta de datos");
            }
        }// finaliza radiorectangulo

        if (radiotrapecio.Checked == true)
        {
            try
            {
                q = Convert.ToDouble(textQ.Text);
                b = Convert.ToDouble(textB.Text);
                z1 = Convert.ToDouble(textZ1.Text);
                z2 = Convert.ToDouble(textZ2.Text);

                alp = 0;
                if (textalp.Text == "")
                {
                    alp = 1;
                }
```

```
                else
                {
                    alp = Convert.ToDouble(textalp.Text);
                }

                s = 0;
                ang = 0;
                if (textS.Text == "")
                {
                    ang = 0;
                }

                if (textS.Text != "")
                {
                    s = Convert.ToDouble(textS.Text);
                    ang = Math.Atan(s);

                    if (ang >= 0 && ang <= (1 / 30d) * Math.PI)
                    {
                        ang = 0;
                    }

                    if (ang > (1 / 30d) * Math.PI)
                    {
                        ang = Math.Atan(s);
                    }
                }

                cte = alp * q * q / (g*Math.Cos(ang));
                //Usamos el método de Newton-Raphson
                do
                {
                    x = xi;
                    a = (b + 0.5 * x * (z1 + z2)) * x;
                    tr = b + (z1 + z2) * x;
                    ap = 3 * Math.Pow(b * x + 0.5 * x * x * (z1 + z2), 2) * (b + x * (z1 + z2));
                    tp = z1 + z2;
                    xi = x - (((a * a * a / tr) - cte) / ((ap * tr - a * a * a * tp) / (tr * tr)));

                    ea = Math.Abs((xi - x) / xi) * 100;
                    conteo++;
                }
                while (ea > epsilon && conteo <= it && ((a * a * a / tr) - cte) != 0);

                if (conteo > it)
                {
                    MessageBox.Show("Es altamente probable que el valor del tirante hidráulico no sea el ↙
correcto", "No Convergencia");
                }

                yc = xi;
                a = (b + 0.5 * yc * (z1 + z2)) * yc;
                p = b + yc * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                rh = a / p;
                tr = b+(z1+z2)*yc;
                v = q / a;
                nf = v / Math.Sqrt(g * (a / tr) * (Math.Cos(ang) / alp));
                E = yc * Math.Cos(ang) + alp * v * v / (2 * g);
                u = (a * a * a / tr) - cte;
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTr.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textnf.Text = Convert.ToString(Math.Round(nf, 4));
                textE.Text = Convert.ToString(Math.Round(E, 4));
                textyc.Text = Convert.ToString(Math.Round(yc, 4));
            }
            catch
```

```
            {
                MessageBox.Show("El cálculo es imposible ya que falta el valor del Gasto o el valor de la Base o algún Talud.", "Falta de datos");
            }
        }// finaliza radiotrapecio

        if (radiotriangulo.Checked == true)
        {
            try
            {
                q = Convert.ToDouble(textQ.Text);
                z1 = Convert.ToDouble(textZ1.Text);
                z2 = Convert.ToDouble(textZ2.Text);

                alp = 0;
                if (textalp.Text == "")
                {
                    alp = 1;
                }
                else
                {
                    alp = Convert.ToDouble(textalp.Text);
                }

                s = 0;
                ang = 0;
                if (textS.Text == "")
                {
                    ang = 0;
                }

                if (textS.Text != "")
                {
                    s = Convert.ToDouble(textS.Text);
                    ang = Math.Atan(s);

                    if (ang >= 0 && ang <= (1 / 30d) * Math.PI)
                    {
                        ang = 0;
                    }

                    if (ang > (1 / 30d) * Math.PI)
                    {
                        ang = Math.Atan(s);
                    }
                }

                yc = Math.Pow(alp*q*q/(0.5*0.5*0.5*g*Math.Cos(ang)*Math.Pow(z1+z2,2)),1/5d);
                a = 0.5 * (z1 + z2) * yc * yc;
                p = yc * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                rh = a / p;
                tr = (z1 + z2) * yc;
                v = q / a;
                nf = v / Math.Sqrt(g * (a / tr) * (Math.Cos(ang) / alp));
                E = yc*Math.Cos(ang) + alp * v * v / (2 * g);
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTr.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textnf.Text = Convert.ToString(Math.Round(nf, 4));
                textE.Text = Convert.ToString(Math.Round(E, 4));
                textyc.Text = Convert.ToString(Math.Round(yc, 4));
            }
            catch
            {
                MessageBox.Show("El cálculo es imposible ya que falta el valor del Gasto o el valor de algún Talud", "Falta de datos");
            }
```

```
            }// finaliza radiotriángulo

            if (radiocirculo.Checked == true)
            {
                try
                {
                    q = Convert.ToDouble(textQ.Text);
                    d = Convert.ToDouble(textD.Text);

                    alp = 0;
                    if (textalp.Text == "")
                    {
                        alp = 1;
                    }
                    else
                    {
                        alp = Convert.ToDouble(textalp.Text);
                    }

                    s = 0;
                    ang = 0;
                    if (textS.Text == "")
                    {
                        ang = 0;
                    }

                    if (textS.Text != "")
                    {
                        s = Convert.ToDouble(textS.Text);
                        ang = Math.Atan(s);

                        if (ang >= 0 && ang <= (1 / 30d) * Math.PI)
                        {
                            ang = 0;
                        }

                        if (ang > (1 / 30d) * Math.PI)
                        {
                            ang = Math.Atan(s);
                        }
                    }

                    cte = alp * q * q / (g * Math.Cos(ang));
                    //Usamos el método de Bisección

                    x1=xi;
                    x2=d;

                    do
                    {
                        x = (x1 + x2) * 0.5;
                        a1 = (1 / 8d) * (2 * Math.Acos(1 - x / (0.5 * d)) - Math.Sin(2 * Math.Acos(1 - x / (0.5 ↙
    * d)))) * d * d;
                        tr1 = 2 * Math.Sqrt(x * (d - x));
                        a = (1 / 8d) * (2 * Math.Acos(1 - x1 / (0.5 * d)) - Math.Sin(2 * Math.Acos(1 - x1 / (0.5↙
     * d)))) * d * d;
                        tr = 2 * Math.Sqrt(x1 * (d - x1));
                        if (((Math.Pow(a, 3) / tr) - cte) * ((Math.Pow(a1, 3) / tr1) - cte) < 0)
                        {
                            x2 = x;
                        }
                        if (((Math.Pow(a, 3) / tr) - cte) * ((Math.Pow(a1, 3) / tr1) - cte) > 0)
                        {
                            x1 = x;
                        }

                        ea = Math.Abs((x - xold) / x) * 100;
                        xold = x;
                        conteo++;
```

```
                }
                while (((Math.Pow(a, 3) / tr) - cte) != 0 && ea > epsilon && conteo <= it);


                yc = x;
                a = (1 / 8d) * (2 * Math.Acos(1 - yc / (0.5 * d)) - Math.Sin(2 * Math.Acos(1 - yc / (0.5 * d)))) * d * d;
                p = 0.5 * (2 * Math.Acos(1 - yc / (0.5 * d)))*d;
                rh = a / p;
                tr = 2 * Math.Sqrt(yc * (d - yc));
                v = q / a;
                nf = v / Math.Sqrt(g * (a / tr) * (Math.Cos(ang) / alp));
                E = yc * Math.Cos(ang) + alp * v * v / (2 * g);
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTr.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textnf.Text = Convert.ToString(Math.Round(nf, 4));
                textE.Text = Convert.ToString(Math.Round(E, 4));
                textyc.Text = Convert.ToString(Math.Round(yc, 4));
            }
            catch
            {
                MessageBox.Show("El cálculo es imposible ya que falta el valor del Gasto o el diámetro", "Falta de datos");
            }
        }// finaliza radiocírculo

        if (radioparabola.Checked == true)
        {
            try
            {
                q = Convert.ToDouble(textQ.Text);
                t = Convert.ToDouble(textT.Text);

                alp = 0;
                if (textalp.Text == "")
                {
                    alp = 1;
                }
                else
                {
                    alp = Convert.ToDouble(textalp.Text);
                }

                s = 0;
                ang = 0;
                if (textS.Text == "")
                {
                    ang = 0;
                }

                if (textS.Text != "")
                {
                    s = Convert.ToDouble(textS.Text);
                    ang = Math.Atan(s);

                    if (ang >= 0 && ang <= (1 / 30d) * Math.PI)
                    {
                        ang = 0;
                    }

                    if (ang > (1 / 30d) * Math.PI)
                    {
                        ang = Math.Atan(s);
                    }
                }
```

```
                    cte = alp * q * q / (g * Math.Cos(ang));


                    yc = Math.Pow((27/8d)*cte/(t*t), 1 / 3d);
                    a = (2 / 3d) * t * yc;
                    p = 0;
                    o = 4 * yc / t;
                    if (o > 0 && o <= 1)
                    {
                        p = t + (8 / 3d) * (yc * yc / t);
                    }
                    if (o > 1)
                    {
                        p = (t * 0.5) * (Math.Sqrt(1 + o * o) + (1 / o) * Math.Log(o + Math.Sqrt(1 + o * o)));

                    }
                    rh = a / p;
                    k = 4*yc/(t*t);
                    v = q / a;
                    nf = v / Math.Sqrt(g * (a / t) * (Math.Cos(ang) / alp));
                    E = yc * Math.Cos(ang) + alp * v * v / (2 * g);
                    u = (a * a * a / t) - cte;
                    textA.Text = Convert.ToString(Math.Round(a, 4));
                    textP.Text = Convert.ToString(Math.Round(p, 4));
                    textRh.Text = Convert.ToString(Math.Round(rh, 4));
                    textTr.Text = Convert.ToString(Math.Round(k, 4));
                    textV.Text = Convert.ToString(Math.Round(v, 4));
                    textnf.Text = Convert.ToString(Math.Round(nf, 4));
                    textE.Text = Convert.ToString(Math.Round(E, 4));
                    textyc.Text = Convert.ToString(Math.Round(yc, 4));
                }
                catch
                {
                    MessageBox.Show("El cálculo es imposible ya que falta el valor del Gasto o el valor de ancho ↙
    superficial.", "Falta de datos");
                }
            }// finaliza radioparabola

        }//finaliza botón calcular

        private void menu_Click(object sender, EventArgs e)
        {
            Form1 Iniciar = new Form1();
            Iniciar.Show();
            this.Hide();
        }

        private void transiciónToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Trans_flujo Iniciar = new Trans_flujo();
            Iniciar.Show();
            this.Hide();
        }




    }
}
```

Código Submódulo Transición de flujo

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Hidraulica_canales
{
    public partial class Trans_flujo : Form
    {
        public Trans_flujo()
        {
            InitializeComponent();
        }

        private void Trans_flujo_FormClosing(object sender, FormClosingEventArgs e)
        {
            Application.Exit();
        }

        public void limpiar()
        {
            textQ.Clear();
            textB.Clear();
            textD.Clear();
            textT.Clear();
            textz1.Clear();
            textz2.Clear();
            textalp.Clear();
            textS.Clear();
            textE.Clear();
            Emin.Clear();

            ty1.Clear();
            ta1.Clear();
            tv1.Clear();
            tfr1.Clear();
            ttf1.Clear();
            tyc.Clear();
            tac.Clear();
            tvc.Clear();
            tfrc.Clear();
            ttfc.Clear();
            ty2.Clear();
            ta2.Clear();
            tv2.Clear();
            tfr2.Clear();
            ttf2.Clear();
            chart1.Series["ey"].Points.Clear();
            chart1.Series["recta"].Points.Clear();
            chart1.Series["horizontal"].Points.Clear();
            chart1.Series["vertical"].Points.Clear();
        }

        private void radiorectangulo_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            textQ.Show();
            textB.Show();
            textD.Hide();
            textT.Hide();
            textz1.Hide();
            textz2.Hide();
            textalp.Show();
            textS.Show();
```

```
            lQ.Show();
            lB.Show();
            lD.Hide();
            lT.Hide();
            lz1.Hide();
            lz2.Hide();
            lalp.Show();
            lS.Show();


            ty1.Show();
            ta1.Show();
            tv1.Show();
            tfr1.Show();
            ttf1.Show();
            tyc.Show();
            tac.Show();
            tvc.Show();
            tfrc.Show();
            ttfc.Show();
            ty2.Show();
            ta2.Show();
            tv2.Show();
            tfr2.Show();
            ttf2.Show();

            ly1.Show();
            la1.Show();
            lv1.Show();
            lfr1.Show();
            ltf1.Show();
            lyc.Show();
            lac.Show();
            lvc.Show();
            lfrc.Show();
            ltfc.Show();
            ly2.Show();
            la2.Show();
            lv2.Show();
            lfr2.Show();
            ltf2.Show();
            textE.Show();
            lE.Show();
        }

        private void radiotrapecio_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            textQ.Show();
            textB.Show();
            textD.Hide();
            textT.Hide();
            textz1.Show();
            textz2.Show();
            textalp.Show();
            textS.Show();

            lQ.Show();
            lB.Show();
            lD.Hide();
            lT.Hide();
            lz1.Show();
            lz2.Show();
            lalp.Show();
            lS.Show();


            ty1.Show();
```

```
            ta1.Show();
            tv1.Show();
            tfr1.Show();
            ttf1.Show();
            tyc.Show();
            tac.Show();
            tvc.Show();
            tfrc.Show();
            ttfc.Show();
            ty2.Show();
            ta2.Show();
            tv2.Show();
            tfr2.Show();
            ttf2.Show();

            ly1.Show();
            la1.Show();
            lv1.Show();
            lfr1.Show();
            ltf1.Show();
            lyc.Show();
            lac.Show();
            lvc.Show();
            lfrc.Show();
            ltfc.Show();
            ly2.Show();
            la2.Show();
            lv2.Show();
            lfr2.Show();
            ltf2.Show();
            textE.Show();
            lE.Show();
        }

        private void radiotriangulo_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            textQ.Show();
            textB.Hide();
            textD.Hide();
            textT.Hide();
            textz1.Show();
            textz2.Show();
            textalp.Show();
            textS.Show();

            lQ.Show();
            lB.Hide();
            lD.Hide();
            lT.Hide();
            lz1.Show();
            lz2.Show();
            lalp.Show();
            lS.Show();


            ty1.Show();
            ta1.Show();
            tv1.Show();
            tfr1.Show();
            ttf1.Show();
            tyc.Show();
            tac.Show();
            tvc.Show();
            tfrc.Show();
            ttfc.Show();
            ty2.Show();
            ta2.Show();
            tv2.Show();
```

```
            tfr2.Show();
            ttf2.Show();

            ly1.Show();
            la1.Show();
            lv1.Show();
            lfr1.Show();
            ltf1.Show();
            lyc.Show();
            lac.Show();
            lvc.Show();
            lfrc.Show();
            ltfc.Show();
            ly2.Show();
            la2.Show();
            lv2.Show();
            lfr2.Show();
            ltf2.Show();
            textE.Show();
            lE.Show();
        }

        private void radiocirculo_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            textQ.Show();
            textB.Hide();
            textD.Show();
            textT.Hide();
            textz1.Hide();
            textz2.Hide();
            textalp.Show();
            textS.Show();

            lQ.Show();
            lB.Hide();
            lD.Show();
            lT.Hide();
            lz1.Hide();
            lz2.Hide();
            lalp.Show();
            lS.Show();


            ty1.Show();
            ta1.Show();
            tv1.Show();
            tfr1.Show();
            ttf1.Show();
            tyc.Show();
            tac.Show();
            tvc.Show();
            tfrc.Show();
            ttfc.Show();
            ty2.Show();
            ta2.Show();
            tv2.Show();
            tfr2.Show();
            ttf2.Show();

            ly1.Show();
            la1.Show();
            lv1.Show();
            lfr1.Show();
            ltf1.Show();
            lyc.Show();
            lac.Show();
            lvc.Show();
            lfrc.Show();
```

```
            ltfc.Show();
            ly2.Show();
            la2.Show();
            lv2.Show();
            lfr2.Show();
            ltf2.Show();
            textE.Show();
            lE.Show();
        }

        private void radioparabola_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            textQ.Show();
            textB.Hide();
            textD.Hide();
            textT.Show();
            textz1.Hide();
            textz2.Hide();
            textalp.Show();
            textS.Show();

            lQ.Show();
            lB.Hide();
            lD.Hide();
            lT.Show();
            lz1.Hide();
            lz2.Hide();
            lalp.Show();
            lS.Show();


            ty1.Show();
            ta1.Show();
            tv1.Show();
            tfr1.Show();
            ttf1.Show();
            tyc.Show();
            tac.Show();
            tvc.Show();
            tfrc.Show();
            ttfc.Show();
            ty2.Show();
            ta2.Show();
            tv2.Show();
            tfr2.Show();
            ttf2.Show();

            ly1.Show();
            la1.Show();
            lv1.Show();
            lfr1.Show();
            ltf1.Show();
            lyc.Show();
            lac.Show();
            lvc.Show();
            lfrc.Show();
            ltfc.Show();
            ly2.Show();
            la2.Show();
            lv2.Show();
            lfr2.Show();
            ltf2.Show();
            textE.Show();
            lE.Show();
        }

        private void lim_Click(object sender, EventArgs e)
        {
```

```
            limpiar();
        }

        private void regresartirante_Click(object sender, EventArgs e)
        {
            tirante_critico Iniciar = new tirante_critico();
            Iniciar.Show();
            this.Hide();
        }

        private void regresarmenu_Click(object sender, EventArgs e)
        {
            Form1 Iniciar = new Form1();
            Iniciar.Show();
            this.Hide();
        }

        private void info_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Si tu valor de coeficiente de coriolis (alpha) es 1, no es necesario ponerlo. Si el ↙
     valor de la pendiente es 0, no es necesario ponerlo", "Información", MessageBoxButtons.OK, MessageBoxIcon. ↙
    Information);
        }

        private void Calcular_Click(object sender, EventArgs e)
        {

            double q, b, d, t, z1, z2, alp, s, ang,E,fx,fxp,ax,bx,cx,dx,cte,a,ap,tr,tp,am1,am3,tr1,o,Ec,r,div,  ↙
    divh,divv;
            double g=9.81;
            double y1, a1, v1, fr1, yc, ac, vc, frc, y2, a2, v2, fr2,y3;
            double ea = 0;
            double xi = 1.0e-6;
            double x = 0, xold = 0;
            int conteo = 0, it = 500;
            double epsilon = 1.0e-8;
            double x1, x2;


            /////////Comienzan los cálculos/////////////

            ////////////////////////Comenzamos con seción rectangular////////

            if (radiorectangulo.Checked == true)
            {
                try
                {
                    q = Convert.ToDouble(textQ.Text);
                    b = Convert.ToDouble(textB.Text);
                    E = Convert.ToDouble(textE.Text);

                    alp = 0;
                    if (textalp.Text == "")
                    {
                        alp = 1;
                    }
                    else
                    {
                        alp = Convert.ToDouble(textalp.Text);
                    }

                    s = 0;
                    ang = 0;
                    if (textS.Text == "")
                    {
                        ang = 0;
                    }
                    else
                    {
```

```
                    s = Convert.ToDouble(textS.Text);
                    ang = Math.Atan(s);

                    if (ang > 0 && ang <= (1 / 30d) * Math.PI)
                    {
                        ang = 0;
                    }
                    else
                    {
                        ang = Math.Atan(s);
                    }
                }

                yc = Math.Pow(alp * q * q / (b * b * g * Math.Cos(ang)), 1 / 3d);
                Ec = yc * Math.Cos(ang) + ((alp * q * q) / (2 * g * b * b * yc * yc));
                tyc.Text = Convert.ToString(Math.Round(yc, 4));
                Emin.Text = Convert.ToString(Math.Round(Ec,4));
                if (Ec >= E)
                {
                    MessageBox.Show("El valor de la energía específica debe ser mayor que la energía mínima ↙
requerida", "Energía no valida");
                }

                if (Ec < E)
                {

                    xi = yc + 1;

                    //Usamos el método de Newton Raphson
                    do
                    {
                        x = xi;
                        fx = x * Math.Cos(ang) + ((alp * q * q) / (2 * g * b * b * x * x)) - E;
                        fxp = Math.Cos(ang) - 2 * ((alp * q * q) / (2 * g * b * b * x * x * x));
                        xi = x - fx / fxp;

                        ea = Math.Abs((xi - x) / xi) * 100;
                        conteo++;

                    }
                    while (ea > epsilon && conteo <= it && fx != 0);

                    y1 = xi;



                    ax = 1;
                    bx = E * -1 / Math.Cos(ang);
                    cx = 0;
                    dx = (alp * q * q) / (2 * g * b * b * Math.Cos(ang));

                    bx = bx + ax * y1;
                    cx = cx + bx * y1;

                    y2 = (-bx + Math.Sqrt(bx * bx - 4 * cx)) * 0.5;
                    y3 = (-bx - Math.Sqrt(bx * bx - 4 * cx)) * 0.5;

                    ty1.Text = Convert.ToString(Math.Round(y1, 4));
                    if (y2 > y3)
                    {
                        ty2.Text = Convert.ToString(Math.Round(y2, 4));
                    }
                    else
                    {
                        y2 = y3;
                        ty2.Text = Convert.ToString(Math.Round(y2, 4));
                    }

                    a1 = b * y1;
```

```
                    v1 = q / a1;
                    fr1 = v1 / Math.Sqrt(g * y1 * Math.Cos(ang) / alp);
                    ac = b * yc;
                    vc = q / ac;
                    frc = vc / Math.Sqrt(g * yc * Math.Cos(ang) / alp);
                    a2 = b * y2;
                    v2 = q / a2;
                    fr2 = v2 / Math.Sqrt(g * y2 * Math.Cos(ang) / alp);


                    ta1.Text = Convert.ToString(Math.Round(a1, 4));
                    tv1.Text = Convert.ToString(Math.Round(v1, 4));
                    tfr1.Text = Convert.ToString(Math.Round(fr1, 4));
                    tac.Text = Convert.ToString(Math.Round(ac, 4));
                    tvc.Text = Convert.ToString(Math.Round(vc, 4));
                    tfrc.Text = Convert.ToString(Math.Round(frc, 4));
                    ta2.Text = Convert.ToString(Math.Round(a2, 4));
                    tv2.Text = Convert.ToString(Math.Round(v2, 4));
                    tfr2.Text = Convert.ToString(Math.Round(fr2, 4));
                    if (fr1 > frc)
                    {
                        ttf1.Text = "Supercrítico";
                        ttf2.Text = "Subcrítico";
                    }
                    else
                    {
                        ttf2.Text = "Supercrítico";
                        ttf1.Text = "Subcrítico";
                    }
                    ttfc.Text = "Crítico";
                    r = y1 * Math.Cos(ang);
                    int u = (int)r + 2;
                    div = Math.Abs((y1 - y2)) / 100;
                    divh = Math.Abs((E - Ec)) / 10;
                    divv = Math.Abs((yc - 0)) / 10;

                    for (int l = 0; l <= u; l++)
                    {
                        chart1.Series["recta"].Points.AddXY(l, l);
                    }

                    chart1.ChartAreas[0].AxisX.Minimum = 0;
                    for (double j = y2; j <= y1; j += div)
                    {
                        chart1.Series["ey"].Points.AddXY(j * Math.Cos(ang) + ((alp * q * q) / (2 * g * b * b * j * j)), j);
                    }

                    for (double ver = 0; ver <= yc; ver += divv)
                    {
                        chart1.Series["vertical"].Points.AddXY(Ec, ver);
                    }
                    for (double hor = Ec; hor <= E; hor += divh)
                    {
                        chart1.Series["horizontal"].Points.AddXY(hor, yc);
                    }
                }
            }

            catch
            {
                MessageBox.Show("Necesitas ingresar al menos un Caudal, una Base y una Energía", "Falta de datos");
            }


        }
```

```
            //////////////////////////Comenzamos con seción trapecial////////

            if (radiotrapecio.Checked == true)
            {
                try
                {
                    q = Convert.ToDouble(textQ.Text);
                    b = Convert.ToDouble(textB.Text);
                    z1 = Convert.ToDouble(textz1.Text);
                    z2 = Convert.ToDouble(textz2.Text);
                    E = Convert.ToDouble(textE.Text);

                    alp = 0;
                    if (textalp.Text == "")
                    {
                        alp = 1;
                    }
                    else
                    {
                        alp = Convert.ToDouble(textalp.Text);
                    }

                    s = 0;
                    ang = 0;
                    if (textS.Text == "")
                    {
                        ang = 0;
                    }
                    else
                    {
                        s = Convert.ToDouble(textS.Text);
                        ang = Math.Atan(s);

                        if (ang > 0 && ang <= (1 / 30d) * Math.PI)
                        {
                            ang = 0;
                        }
                        else
                        {
                            ang = Math.Atan(s);
                        }
                    }

                    //Calculamos el tirante crítico

                    cte = alp * q * q / (g * Math.Cos(ang));
                    //Usamos el método de Newton-Raphson
                    do
                    {
                        x = xi;
                        a = (b + 0.5 * x * (z1 + z2)) * x;
                        tr = b + (z1 + z2) * x;
                        ap = 3 * Math.Pow(b * x + 0.5 * x * x * (z1 + z2), 2) * (b + x * (z1 + z2));
                        tp = z1 + z2;
                        xi = x - (((a * a * a / tr) - cte) / ((ap * tr - a * a * a * tp) / (tr * tr)));

                        ea = Math.Abs((xi - x) / xi) * 100;
                        conteo++;
                    }
                    while (ea > epsilon && conteo <= it && ((a * a * a / tr) - cte) != 0);

                    if (conteo > it)
                    {
                        MessageBox.Show("Es altamente probable que el valor del tirante hidráulico no sea el ↙
correcto", "No Convergencia");
                    }

                    yc = xi;
                    Ec = yc * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math.Pow((b + 0.5 * yc * (z1 + z2)) * yc ↙
```

```
, 2)));
                tyc.Text = Convert.ToString(Math.Round(yc, 4));
                Emin.Text = Convert.ToString(Math.Round(Ec, 4));
                if (Ec >= E)
                {
                    MessageBox.Show("El valor de la energía específica debe ser mayor que la energía mínima requerida", "Energía no valida");
                }

                if (Ec < E)
                {

                    conteo = 0;
                    //fin del cálculo de tirante crítico

                    xi = yc + 1;

                    //Usamos el método de Newton Raphson
                    do
                    {
                        x = xi;
                        fx = x * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math.Pow((b + 0.5 * x * (z1 + z2)) * x, 2))) - E;
                        fxp = Math.Cos(ang) + (alp * q * q / (2 * g)) * (-16 * ((z1 + z2) * x + b) / (x * x * x * Math.Pow((z1 + z2) * x + 2 * b, 3)));
                        xi = x - fx / fxp;

                        ea = Math.Abs((xi - x) / xi) * 100;
                        conteo++;

                    }
                    while (ea > epsilon && conteo <= it && fx != 0);

                    y1 = xi;
                    conteo = 0;

                    //Cálculo de tirante alterno menor con bisección


                    x1 = 0.000001;
                    x2 = yc;

                    do
                    {
                        x = (x1 + x2) * 0.5;
                        am1 = x1 * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math.Pow((b + 0.5 * x1 * (z1 + z2)) * x1, 2))) - E;
                        am3 = x * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math.Pow((b + 0.5 * x * (z1 + z2)) * x, 2))) - E;
                        if (am1 * am3 < 0)
                        {
                            x2 = x;
                        }
                        if (am1 * am3 > 0)
                        {
                            x1 = x;
                        }

                        ea = Math.Abs((x - xold) / x) * 100;
                        xold = x;
                        conteo++;
                    }
                    while (am3 != 0 && ea > epsilon && conteo <= it);

                    y2 = x;



                    ty1.Text = Convert.ToString(Math.Round(y1, 4));
```

```
                    ty2.Text = Convert.ToString(Math.Round(y2, 4));


                    a1 = (b + 0.5 * (z1 + z2) * y1) * y1;
                    v1 = q / a1;
                    fr1 = v1 / Math.Sqrt(g * (a1 / (b + (z1 + z2) * y1)) * Math.Cos(ang) / alp);
                    ac = (b + 0.5 * (z1 + z2) * yc) * yc;
                    vc = q / ac;
                    frc = vc / Math.Sqrt(g * (ac / (b + (z1 + z2) * yc)) * Math.Cos(ang) / alp);
                    a2 = (b + 0.5 * (z1 + z2) * y2) * y2;
                    v2 = q / a2;
                    fr2 = v2 / Math.Sqrt(g * (a2 / (b + (z1 + z2) * y2)) * Math.Cos(ang) / alp);


                    ta1.Text = Convert.ToString(Math.Round(a1, 4));
                    tv1.Text = Convert.ToString(Math.Round(v1, 4));
                    tfr1.Text = Convert.ToString(Math.Round(fr1, 4));
                    tac.Text = Convert.ToString(Math.Round(ac, 4));
                    tvc.Text = Convert.ToString(Math.Round(vc, 4));
                    tfrc.Text = Convert.ToString(Math.Round(frc, 4));
                    ta2.Text = Convert.ToString(Math.Round(a2, 4));
                    tv2.Text = Convert.ToString(Math.Round(v2, 4));
                    tfr2.Text = Convert.ToString(Math.Round(fr2, 4));
                    if (fr1 > frc)
                    {
                        ttf1.Text = "Supercrítico";
                        ttf2.Text = "Subcrítico";
                    }
                    else
                    {
                        ttf2.Text = "Supercrítico";
                        ttf1.Text = "Subcrítico";
                    }
                    ttfc.Text = "Crítico";

                    int u = (int)y1 + 2;
                    div = Math.Abs((y1 - y2)) / 100;
                    divh = Math.Abs((E - Ec)) / 10;
                    divv = Math.Abs((yc - 0)) / 10;

                    for (int l = 0; l <= u; l++)
                    {
                        chart1.Series["recta"].Points.AddXY(l, l);
                    }
                    chart1.ChartAreas[0].AxisX.Minimum = 0;
                    for (double j = y2; j <= y1; j += div)
                    {
                        chart1.Series["ey"].Points.AddXY(j * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math. ↙
Pow((b + 0.5 * j * (z1 + z2)) * j, 2))), j);
                    }
                    for (double ver = 0; ver <= yc; ver += divv)
                    {
                        chart1.Series["vertical"].Points.AddXY(Ec, ver);
                    }
                    for (double hor = Ec; hor <= E; hor += divh)
                    {
                        chart1.Series["horizontal"].Points.AddXY(hor, yc);
                    }
                }

            }

            catch
            {
                MessageBox.Show("Necesitas ingresar al menos un Caudal, una Base, Taludes y una Energía",  ↙
"Falta de datos");
            }
        }
```

```
            ////////////////////////Comenzamos con sección triangular////////

            if (radiotriangulo.Checked == true)
            {
                try
                {
                    q = Convert.ToDouble(textQ.Text);
                    z1 = Convert.ToDouble(textz1.Text);
                    z2 = Convert.ToDouble(textz2.Text);
                    E = Convert.ToDouble(textE.Text);

                    alp = 0;
                    if (textalp.Text == "")
                    {
                        alp = 1;
                    }
                    else
                    {
                        alp = Convert.ToDouble(textalp.Text);
                    }

                    s = 0;
                    ang = 0;
                    if (textS.Text == "")
                    {
                        ang = 0;
                    }
                    else
                    {
                        s = Convert.ToDouble(textS.Text);
                        ang = Math.Atan(s);

                        if (ang > 0 && ang <= (1 / 30d) * Math.PI)
                        {
                            ang = 0;
                        }
                        else
                        {
                            ang = Math.Atan(s);
                        }
                    }

                    //Calculamos el tirante crítico

                    yc = Math.Pow(alp * q * q / (0.5 * 0.5 * 0.5 * g * Math.Cos(ang) * Math.Pow(z1 + z2, 2)), 1 ↙
/ 5d);
                    Ec = yc * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math.Pow((0.5 * yc * (z1 + z2)) * yc,   ↙
2)));
                    tyc.Text = Convert.ToString(Math.Round(yc, 4));
                    Emin.Text = Convert.ToString(Math.Round(Ec, 4));
                    if (Ec >= E)
                    {
                        MessageBox.Show("El valor de la energía específica debe ser mayor que la energía mínima ↙
requerida", "Energía no valida");
                    }

                    if (Ec < E)
                    {
                        xi = yc + 1;

                        //Usamos el método de Newton Raphson
                        do
                        {
                            x = xi;
                            fx = x * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math.Pow((0.5 * x * (z1 + z2)) * ↙
x, 2))) - E;
                            fxp = Math.Cos(ang) + (alp * q * q / (2 * g)) * (-16 / (Math.Pow(z1 + z2, 2) * Math.↙
Pow(x, 5)));
```

```
                    xi = x - fx / fxp;

                    ea = Math.Abs((xi - x) / xi) * 100;
                    conteo++;

                }
                while (ea > epsilon && conteo <= it && fx != 0);

                y1 = xi;
                conteo = 0;

                //Cálculo de tirante alterno menor con bisección


                x1 = 0.000001;
                x2 = yc;

                do
                {
                    x = (x1 + x2) * 0.5;
                    am1 = x1 * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math.Pow((0.5 * x1 * (z1 + z2))↙
* x1, 2))) - E;
                    am3 = x * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math.Pow((0.5 * x * (z1 + z2)) *↙
x, 2))) - E;
                    if (am1 * am3 < 0)
                    {
                        x2 = x;
                    }
                    if (am1 * am3 > 0)
                    {
                        x1 = x;
                    }

                    ea = Math.Abs((x - xold) / x) * 100;
                    xold = x;
                    conteo++;
                }
                while (am3 != 0 && ea > epsilon && conteo <= it);

                y2 = x;



                ty1.Text = Convert.ToString(Math.Round(y1, 4));
                ty2.Text = Convert.ToString(Math.Round(y2, 4));


                a1 = 0.5 * (z1 + z2) * y1 * y1;
                v1 = q / a1;
                fr1 = v1 / Math.Sqrt(g * (a1 / ((z1 + z2) * y1)) * Math.Cos(ang) / alp);
                ac = 0.5 * (z1 + z2) * yc * yc; ;
                vc = q / ac;
                frc = vc / Math.Sqrt(g * (ac / ((z1 + z2) * yc)) * Math.Cos(ang) / alp);
                a2 = 0.5 * (z1 + z2) * y2 * y2;
                v2 = q / a2;
                fr2 = v2 / Math.Sqrt(g * (a2 / ((z1 + z2) * y2)) * Math.Cos(ang) / alp);


                ta1.Text = Convert.ToString(Math.Round(a1, 4));
                tv1.Text = Convert.ToString(Math.Round(v1, 4));
                tfr1.Text = Convert.ToString(Math.Round(fr1, 4));
                tac.Text = Convert.ToString(Math.Round(ac, 4));
                tvc.Text = Convert.ToString(Math.Round(vc, 4));
                tfrc.Text = Convert.ToString(Math.Round(frc, 4));
                ta2.Text = Convert.ToString(Math.Round(a2, 4));
                tv2.Text = Convert.ToString(Math.Round(v2, 4));
                tfr2.Text = Convert.ToString(Math.Round(fr2, 4));
                if (fr1 > frc)
                {
```

```
                        ttf1.Text = "Supercrítico";
                        ttf2.Text = "Subcrítico";
                    }
                    else
                    {
                        ttf2.Text = "Supercrítico";
                        ttf1.Text = "Subcrítico";
                    }
                    ttfc.Text = "Crítico";

                    int u = (int)y1 + 2;
                    div = Math.Abs((y1 - y2)) / 100;
                    divh = Math.Abs((E - Ec)) / 10;
                    divv = Math.Abs((yc - 0)) / 10;

                    for (int l = 0; l <= u; l++)
                    {
                        chart1.Series["recta"].Points.AddXY(l, l);
                    }
                    chart1.ChartAreas[0].AxisX.Minimum = 0;
                    for (double j = y2; j <= y1; j += div)
                    {
                        chart1.Series["ey"].Points.AddXY(j * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math. ↙
Pow((0.5 * j * (z1 + z2)) * j, 2))), j);
                    }
                    for (double ver = 0; ver <= yc; ver += divv)
                    {
                        chart1.Series["vertical"].Points.AddXY(Ec, ver);
                    }
                    for (double hor = Ec; hor <= E; hor += divh)
                    {
                        chart1.Series["horizontal"].Points.AddXY(hor, yc);
                    }
                }

            }

            catch
            {
                MessageBox.Show("Necesitas ingresar al menos un Caudal, Taludes y una Energía", "Falta de   ↙
datos");
            }
        }


        ////////////////////////Comenzamos con seción circular////////

        if (radiocirculo.Checked == true)
        {
            try
            {
                q = Convert.ToDouble(textQ.Text);
                d = Convert.ToDouble(textD.Text);
                E = Convert.ToDouble(textE.Text);

                alp = 0;
                if (textalp.Text == "")
                {
                    alp = 1;
                }
                else
                {
                    alp = Convert.ToDouble(textalp.Text);
                }

                s = 0;
                ang = 0;
                if (textS.Text == "")
                {
```

```
                    ang = 0;
                }
                else
                {
                    s = Convert.ToDouble(textS.Text);
                    ang = Math.Atan(s);

                    if (ang > 0 && ang <= (1 / 30d) * Math.PI)
                    {
                        ang = 0;
                    }
                    else
                    {
                        ang = Math.Atan(s);
                    }
                }

                //Calculamos el tirante crítico

                cte = alp * q * q / (g * Math.Cos(ang));
                //Usamos el método de Bisección

                x1 = xi;
                x2 = d;

                do
                {
                    x = (x1 + x2) * 0.5;
                    a1 = (1 / 8d) * (2 * Math.Acos(1 - x / (0.5 * d)) - Math.Sin(2 * Math.Acos(1 - x / (0.5 * d)))) * d * d;
                    tr1 = 2 * Math.Sqrt(x * (d - x));
                    a = (1 / 8d) * (2 * Math.Acos(1 - x1 / (0.5 * d)) - Math.Sin(2 * Math.Acos(1 - x1 / (0.5 * d)))) * d * d;
                    tr = 2 * Math.Sqrt(x1 * (d - x1));
                    if (((Math.Pow(a, 3) / tr) - cte) * ((Math.Pow(a1, 3) / tr1) - cte) < 0)
                    {
                        x2 = x;
                    }
                    if (((Math.Pow(a, 3) / tr) - cte) * ((Math.Pow(a1, 3) / tr1) - cte) > 0)
                    {
                        x1 = x;
                    }

                    ea = Math.Abs((x - xold) / x) * 100;
                    xold = x;
                    conteo++;
                }
                while (((Math.Pow(a, 3) / tr) - cte) != 0 && ea > epsilon && conteo <= it);


                yc = x;
                o = 2 * Math.Acos(1 - yc / (0.5 * d));
                Ec = yc * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math.Pow((1 / 8d) * (o - Math.Sin(o)) * d * d, 2)));
                tyc.Text = Convert.ToString(Math.Round(yc, 4));
                Emin.Text = Convert.ToString(Math.Round(Ec, 4));
                if (Ec >= E)
                {
                    MessageBox.Show("El valor de la energía específica debe ser mayor que la energía mínima requerida", "Energía no valida");
                }

                if (Ec < E)
                {
                conteo = 0;
                xi = yc + 1;
```

```
                //Usamos el método de Newton Raphson
                do
                {
                    x = xi;
                    o = 2 * Math.Acos(1 - x / (0.5 * d));
                    fx = x * Math.Cos(ang) + ((alp * q * q) / (2*g*Math.Pow((1/8d)*(o-Math.Sin(o))*d*d,2))) ↙
- E;
                    fxp = Math.Cos(ang) + (alp * q * q / (2 * g)) * (-256*(Math.Cos(2*Math.Asin((x-0.5*d)/(0 ↙
.5*d)))+1)/(d*d*d*d*Math.Sqrt(-x*(x-d))*Math.Pow(Math.Sin(2*Math.Asin((x-0.5*d)/(0.5*d)))+2*Math.Asin((x-0.5 ↙
*d)/(0.5*d))+Math.PI,3)));
                    xi = x - fx / fxp;

                    ea = Math.Abs((xi - x) / xi) * 100;
                    conteo++;

                }
                while (ea > epsilon && conteo <= it && fx != 0);

                y1 = xi;
                conteo = 0;

                    //Cálculo de tirante alterno menor con bisección


                    x1 = 0.000001;
                    x2 = yc;

                    do
                    {
                        x = (x1 + x2) * 0.5;
                        am1 = x1 * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math.Pow((1 / 8d) * (2 * Math. ↙
Acos(1 - x1 / (0.5 * d)) - Math.Sin(2 * Math.Acos(1 - x1 / (0.5 * d)))) * d * d, 2))) - E;
                        am3 = x1 * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math.Pow((1 / 8d) * (2 * Math. ↙
Acos(1 - x / (0.5 * d)) - Math.Sin(2 * Math.Acos(1 - x / (0.5 * d)))) * d * d, 2))) - E;
                        if (am1 * am3 < 0)
                        {
                            x2 = x;
                        }
                        if (am1 * am3 > 0)
                        {
                            x1 = x;
                        }

                        ea = Math.Abs((x - xold) / x) * 100;
                        xold = x;
                        conteo++;
                    }
                    while (am3 != 0 && ea > epsilon && conteo <= it);

                    y2 = x;



                    ty1.Text = Convert.ToString(Math.Round(y1, 4));
                    ty2.Text = Convert.ToString(Math.Round(y2, 4));


                    a1 = (1 / 8d) * (2 * Math.Acos(1 - y1 / (0.5 * d)) - Math.Sin(2 * Math.Acos(1 - y1 / (0. ↙
5 * d)))) * d * d;
                    v1 = q / a1;
                    fr1 = v1 / Math.Sqrt(g * (a1 / (2 * Math.Sqrt(y1 * (d - y1)))) * Math.Cos(ang) / alp);
                    ac = (1 / 8d) * (2 * Math.Acos(1 - yc / (0.5 * d)) - Math.Sin(2 * Math.Acos(1 - yc / (0. ↙
5 * d)))) * d * d;
                    vc = q / ac;
                    frc = vc / Math.Sqrt(g * (ac / (2 * Math.Sqrt(yc * (d - yc)))) * Math.Cos(ang) / alp);
                    a2 = (1 / 8d) * (2 * Math.Acos(1 - y2 / (0.5 * d)) - Math.Sin(2 * Math.Acos(1 - y2 / (0. ↙
5 * d)))) * d * d;
                    v2 = q / a2;
                    fr2 = v2 / Math.Sqrt(g * (a2 / (2 * Math.Sqrt(y2 * (d - y2)))) * Math.Cos(ang) / alp);
```

```
                    ta1.Text = Convert.ToString(Math.Round(a1, 4));
                    tv1.Text = Convert.ToString(Math.Round(v1, 4));
                    tfr1.Text = Convert.ToString(Math.Round(fr1, 4));
                    tac.Text = Convert.ToString(Math.Round(ac, 4));
                    tvc.Text = Convert.ToString(Math.Round(vc, 4));
                    tfrc.Text = Convert.ToString(Math.Round(frc, 4));
                    ta2.Text = Convert.ToString(Math.Round(a2, 4));
                    tv2.Text = Convert.ToString(Math.Round(v2, 4));
                    tfr2.Text = Convert.ToString(Math.Round(fr2, 4));
                    if (fr1 > frc)
                    {
                        ttf1.Text = "Supercrítico";
                        ttf2.Text = "Subcrítico";
                    }
                    else
                    {
                        ttf2.Text = "Supercrítico";
                        ttf1.Text = "Subcrítico";
                    }
                    ttfc.Text = "Crítico";

                    int u = (int)y1 + 2;
                    div = Math.Abs((y1 - y2)) / 100;
                    divh = Math.Abs((E - Ec)) / 10;
                    divv = Math.Abs((yc - 0)) / 10;
                    o = 2 * Math.Acos(1 - yc / (0.5 * d));

                    for (int l = 0; l <= u; l++)
                    {
                        chart1.Series["recta"].Points.AddXY(l, l);
                    }
                    chart1.ChartAreas[0].AxisX.Minimum = 0;
                    for (double j = y2; j <= y1; j += div)
                    {
                        o = 2 * Math.Acos(1 - j / (0.5 * d));
                        chart1.Series["ey"].Points.AddXY(j * Math.Cos(ang) + ((alp * q * q) / (2 * g * Math.↙
Pow((1 / 8d) * (o - Math.Sin(o)) * d * d, 2))), j);
                    }
                    for (double ver = 0; ver <= yc; ver += divv)
                    {
                        chart1.Series["vertical"].Points.AddXY(Ec, ver);
                    }
                    for (double hor = Ec; hor <= E; hor += divh)
                    {
                        chart1.Series["horizontal"].Points.AddXY(hor, yc);
                    }
                }

            }

            catch
            {
                MessageBox.Show("Necesitas ingresar al menos un Caudal, Diámetro y una Energía", "Falta de ↙
datos");
            }
        }

        ////////////////////////Comenzamos con seción parábola///////

        if (radioparabola.Checked == true)
        {
            try
            {
                q = Convert.ToDouble(textQ.Text);
                t = Convert.ToDouble(textT.Text);
                E = Convert.ToDouble(textE.Text);
```

```
                alp = 0;
                if (textalp.Text == "")
                {
                    alp = 1;
                }
                else
                {
                    alp = Convert.ToDouble(textalp.Text);
                }

                s = 0;
                ang = 0;
                if (textS.Text == "")
                {
                    ang = 0;
                }
                else
                {
                    s = Convert.ToDouble(textS.Text);
                    ang = Math.Atan(s);

                    if (ang > 0 && ang <= (1 / 30d) * Math.PI)
                    {
                        ang = 0;
                    }
                    else
                    {
                        ang = Math.Atan(s);
                    }
                }

                //Calculamos el tirante crítico

                cte = alp * q * q / (g * Math.Cos(ang));


                yc = Math.Pow((27 / 8d) * cte / (t * t), 1 / 3d);
                Ec = yc * Math.Cos(ang) + ((alp * q * q) / (2 * g * (4 / 9d) * t * t * yc * yc));
                tyc.Text = Convert.ToString(Math.Round(yc, 4));
                Emin.Text = Convert.ToString(Math.Round(Ec, 4));
                if (Ec >= E)
                {
                    MessageBox.Show("El valor de la energía específica debe ser mayor que la energía mínima ↙
requerida", "Energía no valida");
                }

                if (Ec < E)
                {
                    xi = yc + 1;

                    //Usamos el método de Newton Raphson
                    do
                    {
                        x = xi;
                        fx = x * Math.Cos(ang) + ((alp * q * q) / (2 * g * (4 / 9d) * t * t * x * x)) - E;
                        fxp = Math.Cos(ang) + ((alp * q * q) / (2 * g * (4 / 9d) * t * t)) * (-2 / (x * x * ↙
x));
                        xi = x - fx / fxp;

                        ea = Math.Abs((xi - x) / xi) * 100;
                        conteo++;

                    }
                    while (ea > epsilon && conteo <= it && fx != 0);

                    y1 = xi;

                    ax = 1;
                    bx = -E / Math.Cos(ang);
```

```
                        cx = 0;
                        dx = alp * q * q / (2 * g * (4 / 9d) * t * t * Math.Cos(ang));



                        bx = bx + ax * y1;
                        cx = cx + bx * y1;

                        y2 = (-bx + Math.Sqrt(bx * bx - 4 * cx)) * 0.5;
                        y3 = (-bx - Math.Sqrt(bx * bx - 4 * cx)) * 0.5;

                        ty1.Text = Convert.ToString(Math.Round(y1, 4));
                        if (y2 > y3)
                        {
                            ty2.Text = Convert.ToString(Math.Round(y2, 4));
                        }
                        else
                        {
                            y2 = y3;
                            ty2.Text = Convert.ToString(Math.Round(y2, 4));
                        }

                        a1 = (2 / 3d) * t * y1;
                        v1 = q / a1;
                        fr1 = v1 / Math.Sqrt(g * (a1 / t) * Math.Cos(ang) / alp);
                        ac = (2 / 3d) * t * yc;
                        vc = q / ac;
                        frc = vc / Math.Sqrt(g * (ac / t) * Math.Cos(ang) / alp);
                        a2 = (2 / 3d) * t * y2;
                        v2 = q / a2;
                        fr2 = v2 / Math.Sqrt(g * (a2 / t) * Math.Cos(ang) / alp);


                        ta1.Text = Convert.ToString(Math.Round(a1, 4));
                        tv1.Text = Convert.ToString(Math.Round(v1, 4));
                        tfr1.Text = Convert.ToString(Math.Round(fr1, 4));
                        tac.Text = Convert.ToString(Math.Round(ac, 4));
                        tvc.Text = Convert.ToString(Math.Round(vc, 4));
                        tfrc.Text = Convert.ToString(Math.Round(frc, 4));
                        ta2.Text = Convert.ToString(Math.Round(a2, 4));
                        tv2.Text = Convert.ToString(Math.Round(v2, 4));
                        tfr2.Text = Convert.ToString(Math.Round(fr2, 4));
                        if (fr1 > frc)
                        {
                            ttf1.Text = "Supercrítico";
                            ttf2.Text = "Subcrítico";
                        }
                        else
                        {
                            ttf2.Text = "Supercrítico";
                            ttf1.Text = "Subcrítico";
                        }
                        ttfc.Text = "Crítico";

                        int u = (int)y1 + 2;
                        div = Math.Abs((y1 - y2)) / 100;
                        divh = Math.Abs((E - Ec)) / 10;
                        divv = Math.Abs((yc - 0)) / 10;

                        for (int l = 0; l <= u; l++)
                        {
                            chart1.Series["recta"].Points.AddXY(l, l);
                        }
                        chart1.ChartAreas[0].AxisX.Minimum = 0;
                        for (double j = y2; j <= y1; j += div)
                        {
                            chart1.Series["ey"].Points.AddXY(j * Math.Cos(ang) + ((alp * q * q) / (2 * g * (4 / ↙
    9d) * t * t * j * j)), j);
                        }
```

```
                        for (double ver = 0; ver <= yc; ver += divv)
                        {
                            chart1.Series["vertical"].Points.AddXY(Ec, ver);
                        }
                        for (double hor = Ec; hor <= E; hor += divh)
                        {
                            chart1.Series["horizontal"].Points.AddXY(hor, yc);
                        }

                    }

                }

                catch
                {
                    MessageBox.Show("Necesitas ingresar al menos un Caudal, un Anchos Superficial y una Energía"
    , "Falta de datos");
                }
            }

        }

        //Termina botón calculo
    }
}
```

Código Módulo Cálculo Flujo normal

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Hidraulica_canales
{
    public partial class Flujo_normal : Form
    {
        public Flujo_normal()
        {
            InitializeComponent();
        }

        private void Flujo_normal_FormClosing(object sender, FormClosingEventArgs e)
        {
            Application.Exit();
        }

        private void Flujo_normal_Load(object sender, EventArgs e)
        {
            textyd.Hide();
            lyd.Hide();
            lsme.Hide();
            ltc.Hide();
            lsp.Hide();
            lsec1.Hide();
            lA1.Hide();
            lP1.Hide();
            lrh1.Hide();
            ln1.Hide();
            lalp1.Hide();
            lsec2.Hide();
            lA2.Hide();
            lP2.Hide();
            lrh2.Hide();
            ln2.Hide();
            lalp2.Hide();
            ldx.Hide();
            ldy.Hide();
            ldif.Hide();
            lka.Hide();
            lkb.Hide();
            lkr.Hide();
            lkt.Hide();
            lQ0.Hide();
            lva.Hide();
            lvb.Hide();
            lvalp.Hide();
            lvbalp.Hide();
            lS1.Hide();
            lQ11.Hide();
            lQ0Q1.Hide();
            textA1.Hide();
            textP1.Hide();
            textrh1.Hide();
            textn1.Hide();
            textalp1.Hide();
            textA2.Hide();
            textP2.Hide();
            textrh2.Hide();
            textn2.Hide();
            textalp2.Hide();
            textdx.Hide();
```

```
            textdy.Hide();
            textdif.Hide();
            textka.Hide();
            textkb.Hide();
            textkab.Hide();
            textkt.Hide();
            textQ0.Hide();
            textva.Hide();
            textvb.Hide();
            textvalp.Hide();
            textvbalp.Hide();
            textS1.Hide();
            textQ11.Hide();
            textQ0Q1.Hide();
            textct.Hide();
            lct.Hide();
            info1.Hide();
            line1.Show();
            line2.Show();
            lresult.Show();
            lingres.Show();
            info2.Hide();
            line3.Hide();
            line4.Hide();
            lingre2.Hide();
            lresult2.Hide();

        }

        private void factorDeSecciónToolStripMenuItem_Click(object sender, EventArgs e)
        {
            textkt.Enabled = true;
            radiorectangulo.Show();
            radiotrapecio.Show();
            radiotriangulo.Show();
            radiocirculo.Show();
            radioparabola.Show();
            textB.Enabled = true;
            textyd.Hide();
            lyd.Hide();
            texty.Enabled = true;
            lfs.Show();
            lsme.Hide();
            ltc.Hide();
            lsp.Hide();
            //
            lsec1.Hide();
            lA1.Hide();
            lP1.Hide();
            lrh1.Hide();
            ln1.Hide();
            lalp1.Hide();
            lsec2.Hide();
            lA2.Hide();
            lP2.Hide();
            lrh2.Hide();
            ln2.Hide();
            lalp2.Hide();
            ldx.Hide();
            ldy.Hide();
            ldif.Hide();
            lka.Hide();
            lkb.Hide();
            lkr.Hide();
            lkt.Hide();
            lQ0.Hide();
            lva.Hide();
            lvb.Hide();
            lvalp.Hide();
```

```
            lvbalp.Hide();
            lS1.Hide();
            lQ11.Hide();
            lQ0Q1.Hide();
            textA1.Hide();
            textP1.Hide();
            textrh1.Hide();
            textn1.Hide();
            textalp1.Hide();
            textA2.Hide();
            textP2.Hide();
            textrh2.Hide();
            textn2.Hide();
            textalp2.Hide();
            textdx.Hide();
            textdy.Hide();
            textdif.Hide();
            textka.Hide();
            textkb.Hide();
            textkab.Hide();
            textkt.Hide();
            textQ0.Hide();
            textva.Hide();
            textvb.Hide();
            textvalp.Hide();
            textvbalp.Hide();
            textS1.Hide();
            textQ11.Hide();
            textQ0Q1.Hide();
            textct.Hide();
            lct.Hide();
            pictureBox1.Show();
            info1.Hide();
            line1.Show();
            line2.Show();
            lresult.Show();
            lingres.Show();
            info2.Hide();
            line3.Hide();
            line4.Hide();
            lingre2.Hide();
            lresult2.Hide();
        }

        private void secciónMáximaEficienciaToolStripMenuItem_Click(object sender, EventArgs e)
        {

            textkt.Enabled = true;
            radiorectangulo.Show();
            radiotrapecio.Show();
            radiotriangulo.Hide();
            radiocirculo.Hide();
            radioparabola.Hide();
            textB.Enabled = false;
            textyd.Hide();
            lyd.Hide();
            texty.Enabled = true;
            lfs.Hide();
            lsme.Show();
            ltc.Hide();
            lsp.Hide();
            //
            lsec1.Hide();
            lA1.Hide();
            lP1.Hide();
            lrh1.Hide();
            ln1.Hide();
            lalp1.Hide();
            lsec2.Hide();
```

```
            lA2.Hide();
            lP2.Hide();
            lrh2.Hide();
            ln2.Hide();
            lalp2.Hide();
            ldx.Hide();
            ldy.Hide();
            ldif.Hide();
            lka.Hide();
            lkb.Hide();
            lkr.Hide();
            lkt.Hide();
            lQ0.Hide();
            lva.Hide();
            lvb.Hide();
            lvalp.Hide();
            lvbalp.Hide();
            lS1.Hide();
            lQ11.Hide();
            lQ0Q1.Hide();
            textA1.Hide();
            textP1.Hide();
            textrh1.Hide();
            textn1.Hide();
            textalp1.Hide();
            textA2.Hide();
            textP2.Hide();
            textrh2.Hide();
            textn2.Hide();
            textalp2.Hide();
            textdx.Hide();
            textdy.Hide();
            textdif.Hide();
            textka.Hide();
            textkb.Hide();
            textkab.Hide();
            textkt.Hide();
            textQ0.Hide();
            textva.Hide();
            textvb.Hide();
            textvalp.Hide();
            textvbalp.Hide();
            textS1.Hide();
            textQ11.Hide();
            textQ0Q1.Hide();
            textct.Hide();
            lct.Hide();
            pictureBox1.Show();
            info1.Hide();
            line1.Show();
            line2.Show();
            lresult.Show();
            lingres.Show();
            info2.Hide();
            line3.Hide();
            line4.Hide();
            lingre2.Hide();
            lresult2.Hide();

        }

        private void YnconocidoToolStripMenuItem_Click(object sender, EventArgs e)
        {
            textkt.Enabled = true;
            radiorectangulo.Show();
            radiotrapecio.Show();
            radiotriangulo.Show();
            radiocirculo.Show();
            radioparabola.Show();
```

```
            textB.Enabled = true;
            texty.Enabled = false;
            textyd.Show();
            lyd.Show();
            lfs.Hide();
            lsme.Hide();
            ltc.Show();
            lsp.Hide();
            //
            lsec1.Hide();
            lA1.Hide();
            lP1.Hide();
            lrh1.Hide();
            ln1.Hide();
            lalp1.Hide();
            lsec2.Hide();
            lA2.Hide();
            lP2.Hide();
            lrh2.Hide();
            ln2.Hide();
            lalp2.Hide();
            ldx.Hide();
            ldy.Hide();
            ldif.Hide();
            lka.Hide();
            lkb.Hide();
            lkr.Hide();
            lkt.Hide();
            lQ0.Hide();
            lva.Hide();
            lvb.Hide();
            lvalp.Hide();
            lvbalp.Hide();
            lS1.Hide();
            lQ11.Hide();
            lQ0Q1.Hide();
            textA1.Hide();
            textP1.Hide();
            textrh1.Hide();
            textn1.Hide();
            textalp1.Hide();
            textA2.Hide();
            textP2.Hide();
            textrh2.Hide();
            textn2.Hide();
            textalp2.Hide();
            textdx.Hide();
            textdy.Hide();
            textdif.Hide();
            textka.Hide();
            textkb.Hide();
            textkab.Hide();
            textkt.Hide();
            textQ0.Hide();
            textva.Hide();
            textvb.Hide();
            textvalp.Hide();
            textvbalp.Hide();
            textS1.Hide();
            textQ11.Hide();
            textQ0Q1.Hide();
            textct.Hide();
            lct.Hide();
            pictureBox1.Show();
            info1.Show();
            line1.Show();
            line2.Show();
            lresult.Show();
            lingres.Show();
```

```
            info2.Hide();
            line3.Hide();
            line4.Hide();
            lingre2.Hide();
            lresult2.Hide();
        }

        private void secciónPendienteToolStripMenuItem_Click(object sender, EventArgs e)
        {
            textkt.Enabled = false;
            radiorectangulo.Hide();
            radiotrapecio.Hide();
            radiotriangulo.Hide();
            radiocirculo.Hide();
            radioparabola.Hide();
            textQ.Hide();
            textS.Hide();
            textn.Hide();
            textB.Hide();
            textD.Hide();
            textT.Hide();
            textz1.Hide();
            textz2.Hide();
            textyd.Hide();
            texty.Hide();
            textA.Hide();
            textP.Hide();
            textRh.Hide();
            textTR.Hide();
            textV.Hide();
            textTF.Hide();
            textFr.Hide();
            textE.Hide();
            lQ.Hide();
            lpend.Hide();
            ln.Hide();
            lB.Hide();
            lD.Hide();
            lT.Hide();
            lz1.Hide();
            lz2.Hide();
            lyd.Hide();
            ly.Hide();
            lA.Hide();
            lP.Hide();
            lRh.Hide();
            lTR.Hide();
            lk.Hide();
            lV.Hide();
            lTF.Hide();
            lFr.Hide();
            lE.Hide();
            lfs.Hide();
            lsme.Hide();
            ltc.Hide();
            lsp.Show();
            // lo que se mostrará
            lsec1.Show();
            lA1.Show();
            lP1.Show();
            lrh1.Show();
            ln1.Show();
            lalp1.Show();
            lsec2.Show();
            lA2.Show();
            lP2.Show();
            lrh2.Show();
            ln2.Show();
            lalp2.Show();
```

```
            ldx.Show();
            ldy.Show();
            ldif.Show();
            lka.Show();
            lkb.Show();
            lkr.Show();
            lkt.Show();
            lQ0.Show();
            lva.Show();
            lvb.Show();
            lvalp.Show();
            lvbalp.Show();
            lS1.Show();
            lQ11.Show();
            lQ0Q1.Show();
            textA1.Show();
            textP1.Show();
            textrh1.Show();
            textn1.Show();
            textalp1.Show();
            textA2.Show();
            textP2.Show();
            textrh2.Show();
            textn2.Show();
            textalp2.Show();
            textdx.Show();
            textdy.Show();
            textdif.Show();
            textka.Show();
            textkb.Show();
            textkab.Show();
            textkt.Show();
            textQ0.Show();
            textva.Show();
            textvb.Show();
            textvalp.Show();
            textvbalp.Show();
            textS1.Show();
            textQ11.Show();
            textQ0Q1.Show();
            textct.Show();
            lct.Show();
            pictureBox1.Hide();
            info1.Hide();
            line1.Hide();
            line2.Hide();
            lresult.Hide();
            lingres.Hide();
            info2.Show();
            line3.Show();
            line4.Show();
            lingre2.Show();
            lresult2.Show();
        }

        private void menu_Click_1(object sender, EventArgs e)
        {
            Form1 Iniciar = new Form1();
            Iniciar.Show();
            this.Hide();
        }

        private void info1_Click(object sender, EventArgs e)
        {
           MessageBox.Show("En el submódulo Tirante conocido se puede ingresar valor de velocidad. Para mayor ↙
    información consulte el manual de usuario" ,"Información",MessageBoxButtons.OK,MessageBoxIcon.Information);
        }

        private void info2_Click(object sender, EventArgs e)
```

```
        {
            MessageBox.Show("En el submódulo Sección - Pendiente puedes ingresar área y perímetro para que se calcule radio hidráulico o puedes ingresar área y radio hidráulico y el perímetro se calcula. Para mayor información consulte el manual de usuario", "Información", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        public void limpiar()
        {
            textQ.Clear();
            textS.Clear();
            textn.Clear();
            textB.Clear();
            textD.Clear();
            textT.Clear();
            textz1.Clear();
            textz2.Clear();
            texty.Clear();
            textA.Clear();
            textP.Clear();
            textRh.Clear();
            textTR.Clear();
            textV.Clear();
            textFr.Clear();
            textTF.Clear();
            textE.Clear();
            textyd.Clear();
            textA1.Clear();
            textP1.Clear();
            textrh1.Clear();
            textn1.Clear();
            textalp1.Clear();
            textA2.Clear();
            textP2.Clear();
            textrh2.Clear();
            textn2.Clear();
            textalp2.Clear();
            textdx.Clear();
            textdy.Clear();
            textdif.Clear();
            textka.Clear();
            textkb.Clear();
            textkab.Clear();
            textkt.Clear();
            textQ0.Clear();
            textva.Clear();
            textvb.Clear();
            textvalp.Clear();
            textvbalp.Clear();
            textS1.Clear();
            textQ11.Clear();
            textQ0Q1.Clear();
            textct.Clear();

        }

        private void radiorectangulo_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            textQ.Show();
            textS.Show();
            textn.Show();
            textB.Show();
            textD.Hide();
            textT.Hide();
            textz1.Hide();
            textz2.Hide();
            texty.Show();
            textA.Show();
```

```
            textP.Show();
            textRh.Show();
            textTR.Show();
            textV.Show();
            textFr.Show();
            textTF.Show();
            textE.Show();
            lQ.Show();
            lpend.Show();
            ln.Show();
            lB.Show();
            lD.Hide();
            lT.Hide();
            lz1.Hide();
            lz2.Hide();
            ly.Show();
            lA.Show();
            lP.Show();
            lRh.Show();
            lTR.Show();
            lV.Show();
            lFr.Show();
            lTF.Show();
            lE.Show();
            lk.Hide();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.rectángulo;
        }

        private void radiotrapecio_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            textQ.Show();
            textS.Show();
            textn.Show();
            textB.Show();
            textD.Hide();
            textT.Hide();
            textz1.Show();
            textz2.Show();
            texty.Show();
            textA.Show();
            textP.Show();
            textRh.Show();
            textTR.Show();
            textV.Show();
            textFr.Show();
            textTF.Show();
            textE.Show();
            lQ.Show();
            lpend.Show();
            ln.Show();
            lB.Show();
            lD.Hide();
            lT.Hide();
            lz1.Show();
            lz2.Show();
            ly.Show();
            lA.Show();
            lP.Show();
            lRh.Show();
            lTR.Show();
            lV.Show();
            lFr.Show();
            lTF.Show();
            lE.Show();
            lk.Hide();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.trapecio;
        }
```

```
        private void radiotriangulo_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            textQ.Show();
            textS.Show();
            textn.Show();
            textB.Hide();
            textD.Hide();
            textT.Hide();
            textz1.Show();
            textz2.Show();
            texty.Show();
            textA.Show();
            textP.Show();
            textRh.Show();
            textTR.Show();
            textV.Show();
            textFr.Show();
            textTF.Show();
            textE.Show();
            lQ.Show();
            lpend.Show();
            ln.Show();
            lB.Hide();
            lD.Hide();
            lT.Hide();
            lz1.Show();
            lz2.Show();
            ly.Show();
            lA.Show();
            lP.Show();
            lRh.Show();
            lTR.Show();
            lV.Show();
            lFr.Show();
            lTF.Show();
            lE.Show();
            lk.Hide();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.triángulo;
        }

        private void radiocirculo_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            textQ.Show();
            textS.Show();
            textn.Show();
            textB.Hide();
            textD.Show();
            textT.Hide();
            textz1.Hide();
            textz2.Hide();
            texty.Show();
            textA.Show();
            textP.Show();
            textRh.Show();
            textTR.Show();
            textV.Show();
            textFr.Show();
            textTF.Show();
            textE.Show();
            lQ.Show();
            lpend.Show();
            ln.Show();
            lB.Hide();
            lD.Show();
            lT.Hide();
            lz1.Hide();
            lz2.Hide();
```

```
            ly.Show();
            lA.Show();
            lP.Show();
            lRh.Show();
            lTR.Show();
            lk.Hide();
            lV.Show();
            lFr.Show();
            lTF.Show();
            lE.Show();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.círculo;
        }

        private void radioparabola_CheckedChanged(object sender, EventArgs e)
        {
            limpiar();
            textQ.Show();
            textS.Show();
            textn.Show();
            textB.Hide();
            textD.Hide();
            textT.Show();
            textz1.Hide();
            textz2.Hide();
            texty.Show();
            textA.Show();
            textP.Show();
            textRh.Show();
            textTR.Show();
            textV.Show();
            textFr.Show();
            textTF.Show();
            textE.Show();
            lQ.Show();
            lpend.Show();
            ln.Show();
            lB.Hide();
            lD.Hide();
            lT.Show();
            lz1.Hide();
            lz2.Hide();
            ly.Show();
            lA.Show();
            lP.Show();
            lRh.Show();
            lTR.Hide();
            lk.Show();
            lV.Show();
            lFr.Show();
            lTF.Show();
            lE.Show();
            pictureBox1.Image = Hidraulica_canales.Properties.Resources.parábola;
        }

        private void limp_Click(object sender, EventArgs e)
        {
            limpiar();
        }

        private void calcular_Click(object sender, EventArgs e)
        {
            /*declaramos nuestras variables*/
            double q, s, n, b, z1, z2, d, t, yn, a, p, rh, tr, v, fr, en,k,ap,pp,x1,x2,o;
            double ea=0;
            double xi = 1.0e-2;
            double x = 0;
            int it = 1000,conteo=0;
            double epsilon = 1.0e-8;
```

```
        ////////////////////////////////Comienzan los cálculos para factor de sección//////////////////////
////////////////////////////////

        if (radiorectangulo.Checked == true && textB.Enabled == true && texty.Enabled == true && textkt.
Enabled == true)
        {
            try
            {
                q = Convert.ToDouble(textQ.Text);
                s = Convert.ToDouble(textS.Text);
                n = Convert.ToDouble(textn.Text);
                b = Convert.ToDouble(textB.Text);
                k = q*n/Math.Sqrt(s);
                //usaremos método de Newton-Raphson
                do
                {
                    x = xi;
                    a = b * x;
                    p = (b + 2 * x);
                    ap = b;
                    pp = 2;
                    xi = x - (a * Math.Pow(a / p, 2 / 3d) - k)/(ap*Math.Pow(a / p, 2 / 3d)+a*(2/3d)*Math.Pow
(a / p, -1 / 3d)*((ap*p-a*pp)/(p*p)));
                    //xi = x - (b * x * Math.Pow((b * x) / (b + 2 * x), 2 / 3d) - k) / (b * Math.Pow((b * x)
 / (b + 2 * x), 2 / 3d) + (2 * b * b * b * x / (3 * Math.Pow(2 * x + b, 2) * Math.Pow((b * x) / (b + 2 * x),
 1 / 3d))));
                    ea = Math.Abs((xi - x)/xi)*100 ;
                    conteo++;

                }
                while (ea > epsilon && conteo <= it  && ((b * xi * Math.Pow((b * xi) / (b + 2 * xi), 2 / 3d)
 - k)!=0));

                if (conteo > it)
                {
                    MessageBox.Show("Es altamente probable que el valor del tirante hidráulico no sea el
correcto", "No Convergencia");
                }

                yn = xi;
                a = b * yn;
                p = b + 2 * yn;
                rh = a / p;
                tr = b;
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                texty.Text = Convert.ToString(Math.Round(yn,4));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
```

```
                }

            }
            catch
            {
                MessageBox.Show("Faltan datos para realizar el cálculo","Campos Vacíos");
            }
        }
        ////////////////////////////////// cálculos factor de sección trapecio//////////////////////////////
//////

        if (radiotrapecio.Checked == true && textB.Enabled == true && texty.Enabled == true && textkt.
Enabled == true)
        {
            try
            {
                q = Convert.ToDouble(textQ.Text);
                s = Convert.ToDouble(textS.Text);
                n = Convert.ToDouble(textn.Text);
                b = Convert.ToDouble(textB.Text);
                z1 = Convert.ToDouble(textz1.Text);
                z2 = Convert.ToDouble(textz2.Text);
                k = q * n / Math.Sqrt(s);
                // Usamoremos método de Newton-Raphson
                do
                {
                    x = xi;
                    a = (b + 0.5 * x * (z1 + z2)) * x;
                    p= b+x*(Math.Sqrt(1+z1*z1)+Math.Sqrt(1+z2*z2));
                    ap = b + x * (z1 + z2);
                    pp = Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2);
                    rh = a / p;
                    xi = x - ((a * Math.Pow(rh, 2 / 3d)) - k) / (ap * Math.Pow(rh, 2 / 3d) + a * (2 / 3d) *
Math.Pow(rh, -1 / 3d) * ((ap * p - a * pp) / (p * p)));
                    ea = Math.Abs((xi - x) / xi) * 100;
                    conteo++;
                }
                while (ea > epsilon && conteo <= it && (((b + 0.5 * x * (z1 + z2)) * x) * Math.Pow(((b + 0.5
 * x * (z1 + z2)) * x) / (b + x * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2))), 2 / 3d) - k) != 0);

                if (conteo > it)
                {
                    MessageBox.Show("Es altamente probable que el valor del tirante hidráulico no sea el
correcto", "No Convergencia");
                }

                yn = xi;
                a = (b + 0.5 * yn * (z1 + z2)) * x;
                p = (b + yn * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2)));
                rh = a / p;
                tr = b+yn*(z1+z2);
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                texty.Text = Convert.ToString(Math.Round(yn, 4));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
```

```
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }
            catch
            {
                MessageBox.Show("Faltan datos para realizar el cálculo", "Campos Vacíos");
            }
        }
        ////////////////////////////////// cálculos factor de sección triángulo/////////////////////////////
///////

        if (radiotriangulo.Checked == true && texty.Enabled == true && textkt.Enabled == true)
        {
            try
            {
                q = Convert.ToDouble(textQ.Text);
                s = Convert.ToDouble(textS.Text);
                n = Convert.ToDouble(textn.Text);
                z1 = Convert.ToDouble(textz1.Text);
                z2 = Convert.ToDouble(textz2.Text);
                k = q * n / Math.Sqrt(s);
                // Usamoremos método de Newton-Raphson
                do
                {
                    x = xi;
                    xi = x - ((0.5 * x * x * (z1 + z2)) * Math.Pow((0.5 * x * x * (z1 + z2)) / (x * (Math.
Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2))), 2 / 3d) - k)/(((z1+z2)*x)*Math.Pow((0.5 * x * x * (z1 + z2)) /
 (x * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2))), 2 / 3d)+(0.5 * x * x * (z1 + z2))*(2/3d)*Math.Pow(
(0.5 * x * x * (z1 + z2)) / (x * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2))), (-1) / 3d)*(((((z1+z2)*
x)*(x * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2))))-(0.5 * x * x * (z1 + z2))*(Math.Sqrt(1 + z1 *
z1) + Math.Sqrt(1 + z2 * z2)))/Math.Pow((x * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2))),2)));
                    //(((z1+z2)*x)*Math.Pow((0.5 * x * x * (z1 + z2)) / (x * (Math.Sqrt(1 + z1 * z1) + Math.
Sqrt(1 + z2 * z2))), 2 / 3d)+(0.5 * x * x * (z1 + z2))*(2/3d)*Math.Pow((0.5 * x * x * (z1 + z2)) / (x *
(Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2))), (-1) / 3d)*(((((z1+z2)*x)*(x * (Math.Sqrt(1 + z1 * z1) +
 Math.Sqrt(1 + z2 * z2))))-(0.5 * x * x * (z1 + z2))*(Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2)))/Math
.Pow((x * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2))),2))
                        //((((((z1+z2)*x)*(x * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2))))-(0.5 * x *
 x * (z1 + z2))*(Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2)))/Math.Pow((x * (Math.Sqrt(1 + z1 * z1) +
Math.Sqrt(1 + z2 * z2))),2))
                    ea = Math.Abs((xi - x) / xi) * 100;
                    conteo++;
                }
                while (ea > epsilon && conteo <= it );

                if (conteo > it)
                {
                    MessageBox.Show("Es altamente probable que el valor del tirante hidráulico no sea el
correcto", "No Convergencia");
                }

                yn = xi;
                a = (z1+z2)*0.5*yn*yn;
                p = yn * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                rh = a / p;
                tr = yn * (z1 + z2);
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                texty.Text = Convert.ToString(Math.Round(yn, 4));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
```

```
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }
            catch
            {
                MessageBox.Show("Faltan datos para realizar el cálculo", "Campos Vacíos");
            }
        }
        ////////////////////////////////// cálculos factor de sección círculo//////////////////////////////
/////

        if (radiocirculo.Checked == true && texty.Enabled == true && textkt.Enabled == true)
        {
            try
            {
                q = Convert.ToDouble(textQ.Text);
                s = Convert.ToDouble(textS.Text);
                n = Convert.ToDouble(textn.Text);
                d = Convert.ToDouble(textD.Text);
                k = q * n / Math.Sqrt(s);
                x1=xi;
                x2=d;

                //Usamos bisección
                while (((((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x - d) / d)) + 2 * Math.Asin((2 *
x - d) / d) + Math.PI)) * Math.Pow(((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x - d) / d)) + 2 * Math.
Asin((2 * x - d) / d) + Math.PI)) / (0.5 * d * (2 * Math.Asin((2 * x - d) / d) + Math.PI)), 2 / 3d) - k) !=
0 && Math.Abs(x1 - x2) > epsilon))
                {
                    x = (x1 + x2)*0.5;
                    if ((((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x1 - d) / d)) + 2 * Math.Asin((2 *
 x1 - d) / d) + Math.PI)) * Math.Pow(((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x1 - d) / d)) + 2 *
Math.Asin((2 * x1 - d) / d) + Math.PI)) / (0.5 * d * (2 * Math.Asin((2 * x1 - d) / d) + Math.PI)), 2 / 3d) -
 k) * (((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x - d) / d)) + 2 * Math.Asin((2 * x - d) / d) + Math
.PI)) * Math.Pow(((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x - d) / d)) + 2 * Math.Asin((2 * x - d) /
 d) + Math.PI)) / (0.5 * d * (2 * Math.Asin((2 * x - d) / d) + Math.PI)), 2 / 3d) - k) < 0)
                    {
                        x2 = x;
                    }
                    if ((((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x1 - d) / d)) + 2 * Math.Asin((2 *
 x1 - d) / d) + Math.PI)) * Math.Pow(((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x1 - d) / d)) + 2 *
Math.Asin((2 * x1 - d) / d) + Math.PI)) / (0.5 * d * (2 * Math.Asin((2 * x1 - d) / d) + Math.PI)), 2 / 3d) -
 k) * (((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x - d) / d)) + 2 * Math.Asin((2 * x - d) / d) + Math
.PI)) * Math.Pow(((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x - d) / d)) + 2 * Math.Asin((2 * x - d) /
 d) + Math.PI)) / (0.5 * d * (2 * Math.Asin((2 * x - d) / d) + Math.PI)), 2 / 3d) - k) > 0)
                    {
                        x1 = x;
                    }
                    conteo++;
                    if (conteo > it)
                    { break; }

                }


                yn = x;

                if ( (((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x - d) / d)) + 2 * Math.Asin((2 * x -
 d) / d) + Math.PI)) * Math.Pow(((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x - d) / d)) + 2 * Math.
```

```
    Asin((2 * x - d) / d) + Math.PI)) / (0.5 * d * (2 * Math.Asin((2 * x - d) / d) + Math.PI)), 2 / 3d) - k) < -1.0e-6)
                {
                    MessageBox.Show("Cambiar Diámetro ya que el actual no puede transportar el gasto","Error");

                }
                if ((((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x - d) / d)) + 2 * Math.Asin((2 * x - d) / d) + Math.PI)) * Math.Pow(((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * x - d) / d)) + 2 * Math.Asin((2 * x - d) / d) + Math.PI)) / (0.5 * d * (2 * Math.Asin((2 * x - d) / d) + Math.PI)), 2 / 3d) - k) > -1.0e-6)
                {
                    if (yn > 0.8 * d)
                    {
                        MessageBox.Show("Para secciones circulares es recomendable que como máximo se presenten tirantes hidráulicos igual al 80% del Diámetro", "Precaución");
                    }


                    a = (((1 / 8d) * d * d * (Math.Sin(2 * Math.Asin((2 * yn - d) / d)) + 2 * Math.Asin((2 * yn - d) / d) + Math.PI)));
                    p=0.5*d*(2*Math.Acos(1-(yn/(0.5*d))));
                    rh = a / p;
                    tr = 2 * Math.Sqrt(yn * (d - yn));
                    v = q / a;
                    fr = v / Math.Sqrt(9.81 * (a / tr));
                    en = yn + (v * v / (2 * 9.81));
                    texty.Text = Convert.ToString(Math.Round(yn, 4));
                    textA.Text = Convert.ToString(Math.Round(a, 4));
                    textP.Text = Convert.ToString(Math.Round(p, 4));
                    textRh.Text = Convert.ToString(Math.Round(rh, 4));
                    textTR.Text = Convert.ToString(Math.Round(tr, 4));
                    textV.Text = Convert.ToString(Math.Round(v, 4));
                    textFr.Text = Convert.ToString(Math.Round(fr, 4));
                    textE.Text = Convert.ToString(Math.Round(en, 4));
                    if (fr < 1)
                    {
                        textTF.Text = "Subcrítico";
                    }
                    if (fr == 1)
                    {
                        textTF.Text = "Crítico";
                    }
                    if (fr > 1)
                    {
                        textTF.Text = "Supercrítico";
                    }
                }
            }
            catch
            {
                MessageBox.Show("Faltan datos para realizar el cálculo", "Campos Vacíos");
            }
        }
        ////////////////////////////////// cálculos factor de sección parábola//////////////////////////////
    //////

        if (radioparabola.Checked == true && texty.Enabled == true && textkt.Enabled == true)
        {
            try
            {
                q = Convert.ToDouble(textQ.Text);
                s = Convert.ToDouble(textS.Text);
                n = Convert.ToDouble(textn.Text);
                t = Convert.ToDouble(textT.Text);
                k = q * n / Math.Sqrt(s);
                p = 0;
                pp = 0;
                // Usamoremos método de Newton-Raphson
```

```
                do
                {

                    x = xi;
                    a = (2 / 3d) * t * x;
                    o = 4 * x / t;
                    if (o > 0 && o <= 1)
                    {
                        p = t + (8 / 3d) * (x * x / t);
                    }
                    if (o > 1)
                    {
                        p = (t * 0.5) * (Math.Sqrt(1 + o * o) + (1 / o) * Math.Log(o + Math.Sqrt(1 + o *
o)));

                    }
                    ap = (2 / 3d) * t ;
                    if (o > 0 && o <= 1)
                    {
                        pp =  (8 / 3d) * (2 * x / t);
                    }
                    if (o > 1)
                    {
                        pp = 0.5 * t * ((16 * x / (Math.Sqrt(16 * x * x + t * t) * Math.Abs(t))) + ((-t / (4
 * x * x)) * (Math.Log(o + Math.Sqrt(1 + o * o))) + (1 / o) * (4 * 1 / (Math.Sqrt(16 * x * x + t * t)))));
                    }
                    rh = a / p;
                    xi = x - ((a * Math.Pow(rh, 2 / 3d)) - k) / (ap * Math.Pow(rh, 2 / 3d) + a * (2 / 3d) *
Math.Pow(rh, -1 / 3d) * ((ap * p - a * pp) / (p * p)));
                    ea = Math.Abs((xi - x) / xi) * 100;
                    conteo++;
                }
                while (ea > epsilon && conteo <= it );

                if (conteo > it)
                {
                    MessageBox.Show("Es altamente probable que el valor del tirante hidráulico no sea el
correcto", "No Convergencia");
                }

                yn = xi;


                rh = a / p;
                tr = 4 * yn / (t * t);
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / ((3/2d)*(a/yn))));
                en = yn + (v * v / (2 * 9.81));
                texty.Text = Convert.ToString(Math.Round(yn, 4));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }
```

```
            catch
            {
                MessageBox.Show("Faltan datos para realizar el cálculo", "Campos Vacíos");
            }
        }
        ////////////////////////////////Comienzan los cálculos sección máxima eficiencia//////////////////// ↙
////////////////////////////////////

        if (radiorectangulo.Checked == true && textB.Enabled == false && texty.Enabled == true && textkt.    ↙
Enabled == true)
        {
            try
            {
                q = Convert.ToDouble(textQ.Text);
                s = Convert.ToDouble(textS.Text);
                n = Convert.ToDouble(textn.Text);
                k = q * n / Math.Sqrt(s);
                //usaremos método de Newton-Raphson
                do
                {
                    x = xi;
                    a = 2*x * x;
                    p = (2*x + 2 * x);
                    ap = 4*x;
                    pp = 4;
                    xi = x - (a * Math.Pow(a / p, 2 / 3d) - k) / (ap * Math.Pow(a / p, 2 / 3d) + a * (2 /    ↙
3d) * Math.Pow(a / p, -1 / 3d) * ((ap * p - a * pp) / (p * p)));
                    //xi = x - (b * x * Math.Pow((b * x) / (b + 2 * x), 2 / 3d) - k) / (b * Math.Pow((b * x) ↙
 / (b + 2 * x), 2 / 3d) + (2 * b * b * b * x / (3 * Math.Pow(2 * x + b, 2) * Math.Pow((b * x) / (b + 2 * x), ↙
 1 / 3d))));
                    ea = Math.Abs((xi - x) / xi) * 100;
                    conteo++;
                    b = 2 * xi;

                }
                while (ea > epsilon && conteo < it && ((b * xi * Math.Pow((b * xi) / (b + 2 * xi), 2 / 3d) - ↙
 k) != 0));

                if (conteo > it)
                {
                    MessageBox.Show("Es altamente probable que el valor del tirante hidráulico no sea el     ↙
correcto", "No Convergencia");
                }

                yn = xi;
                b = 2 * yn;
                a = b * yn;
                p = b + 2 * yn;
                rh = a / p;
                tr = b;
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textB.Text = Convert.ToString(Math.Round(b, 4));
                texty.Text = Convert.ToString(Math.Round(yn, 4));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
```

```
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }

            }
            catch
            {
                MessageBox.Show("Faltan datos para realizar el cálculo", "Campos Vacíos");
            }
        }
        ////////////////////////////////// cálculos Máxima eficiencia trapecio//////////////////////////////
//////

        if (radiotrapecio.Checked == true && textB.Enabled == false && texty.Enabled == true && textkt.
Enabled == true)
        {
            try
            {
                q = Convert.ToDouble(textQ.Text);
                s = Convert.ToDouble(textS.Text);
                n = Convert.ToDouble(textn.Text);
                z1 = Convert.ToDouble(textz1.Text);
                z2 = Convert.ToDouble(textz2.Text);
                k = q * n / Math.Sqrt(s);
                // Usamoremos método de Newton-Raphson
                do
                {
                    x = xi;
                    a = (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z1 * z1)-0.5*(z1+z2)) * x * x;
                    p = x * (2 * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z1 * z1)) - (z1 + z2));
                    ap = (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z1 * z1) - 0.5 * (z1 + z2)) *2* x ;
                    pp =  (2 * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z1 * z1)) - (z1 + z2));
                    rh = a / p;
                    xi = x - ((a * Math.Pow(rh, 2 / 3d)) - k) / (ap * Math.Pow(rh, 2 / 3d) + a * (2 / 3d) *
Math.Pow(rh, -1 / 3d) * ((ap * p - a * pp) / (p * p)));
                    ea = Math.Abs((xi - x) / xi) * 100;
                    conteo++;
                    b = x * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z1 * z1) - (z1 + z2));
                }
                while (ea > epsilon && conteo <= it && (((b + 0.5 * x * (z1 + z2)) * x) * Math.Pow(((b + 0.5
 * x * (z1 + z2)) * x) / (b + x * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2))), 2 / 3d) - k) != 0);

                if (conteo > it)
                {
                    MessageBox.Show("Es altamente probable que el valor del tirante hidráulico no sea el
correcto", "No Convergencia");
                }

                yn = xi;
                a = (b + 0.5 * yn * (z1 + z2)) * x;
                p = (b + yn * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2)));
                rh = a / p;
                tr = b + yn * (z1 + z2);
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textB.Text = Convert.ToString(Math.Round(b, 4));
                texty.Text = Convert.ToString(Math.Round(yn, 4));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));

                if (fr < 1)
```

```
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }
            catch
            {
                MessageBox.Show("Faltan datos para realizar el cálculo", "Campos Vacíos");
            }
        }

        ////////////////////////////////Comienzan los cálculos tirante conocido//////////////////////////// ↙
    ////////////////////////

        // radio rectangulo/////

        if (radiorectangulo.Checked == true && textB.Enabled == true && texty.Enabled==false)
        {
            // caso donde conocemos gasto,pendiente, base y tirante
            if (textQ.Text != "" && textS.Text != "" && textB.Text != "" && textyd.Text != "" && textn.Text ↙
    == "")
            {
                q = Convert.ToDouble(textQ.Text);
                s = Convert.ToDouble(textS.Text);
                b = Convert.ToDouble(textB.Text);
                yn = Convert.ToDouble(textyd.Text);
                a = b * yn;
                p = b + 2 * yn;
                rh = a / p;
                n = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / q;
                tr = b;
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textn.Text = Convert.ToString(Math.Round(n, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos gasto,rugosidad, base y tirante
            if (textQ.Text != "" && textS.Text == "" && textB.Text != "" && textyd.Text != "" && textn.Text ↙
    != "")
            {
                q = Convert.ToDouble(textQ.Text);
                n = Convert.ToDouble(textn.Text);
```

```
                b = Convert.ToDouble(textB.Text);
                yn = Convert.ToDouble(textyd.Text);
                a = b * yn;
                p = b + 2 * yn;
                rh = a / p;
                s = Math.Pow(q * n / (a * Math.Pow(rh, 2 / 3d)), 2);
                tr = b;
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textS.Text = Convert.ToString(Math.Round(s, 6));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos pendiente,rugosidad, base y tirante
            if (textQ.Text == "" && textS.Text != "" && textB.Text != "" && textyd.Text != "" && textn.Text ↙
!= "")
            {
                s = Convert.ToDouble(textS.Text);
                n = Convert.ToDouble(textn.Text);
                b = Convert.ToDouble(textB.Text);
                yn = Convert.ToDouble(textyd.Text);
                a = b * yn;
                p = b + 2 * yn;
                rh = a / p;
                q = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / n;
                tr = b;
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }
```

```
            ////////////////////////////////// Aquí involucramos el valor de la velocidad//////////////////// ↙
    ////

            // caso donde conocemos pendiente, base, tirante y velocidad
            if (textQ.Text == "" && textS.Text != "" && textB.Text != "" && textyd.Text != "" && textn.Text ↙
    == "" && textV.Text!="")
            {
                s = Convert.ToDouble(textS.Text);
                v = Convert.ToDouble(textV.Text);
                b = Convert.ToDouble(textB.Text);
                yn = Convert.ToDouble(textyd.Text);
                a = b * yn;
                p = b + 2 * yn;
                rh = a / p;
                q = v * a;
                tr = b;
                n = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / q;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textn.Text = Convert.ToString(Math.Round(n, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos rugosidad, base, tirante y velocidad
            if (textQ.Text == "" && textS.Text == "" && textB.Text != "" && textyd.Text != "" && textn.Text ↙
    != "" && textV.Text != "")
            {
                n = Convert.ToDouble(textn.Text);
                v = Convert.ToDouble(textV.Text);
                b = Convert.ToDouble(textB.Text);
                yn = Convert.ToDouble(textyd.Text);
                a = b * yn;
                p = b + 2 * yn;
                rh = a / p;
                q = v * a;
                tr = b;
                s = Math.Pow(q * n / (a * Math.Pow(rh, 2 / 3d)), 2);
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textS.Text = Convert.ToString(Math.Round(s, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
```

```
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

        }// termina radiorectangulo yn conocido

        // radio trapecio/////
        if (radiotrapecio.Checked == true && textB.Enabled == true && texty.Enabled == false)
        {
            // caso donde conocemos gasto,pendiente, base,taludes y tirante
            if (textQ.Text != "" && textS.Text != "" && textB.Text != "" && textyd.Text != "" && textz1.Text
!= "" && textz2.Text != "" && textn.Text == "")
            {
                q = Convert.ToDouble(textQ.Text);
                s = Convert.ToDouble(textS.Text);
                b = Convert.ToDouble(textB.Text);
                yn = Convert.ToDouble(textyd.Text);
                z1 = Convert.ToDouble(textz1.Text);
                z2 = Convert.ToDouble(textz2.Text);
                a = (b+(z1+z2)*0.5*yn)*yn;
                p = b + yn * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                rh = a / p;
                n = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / q;
                tr = b+(z1+z2)*yn;
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textn.Text = Convert.ToString(Math.Round(n, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos gasto, rugosidad, base,taludes y tirante
            if (textQ.Text != "" && textS.Text == "" && textB.Text != "" && textyd.Text != "" && textz1.Text
!= "" && textz2.Text != "" && textn.Text != "")
            {
                q = Convert.ToDouble(textQ.Text);
                n = Convert.ToDouble(textn.Text);
                b = Convert.ToDouble(textB.Text);
                yn = Convert.ToDouble(textyd.Text);
                z1 = Convert.ToDouble(textz1.Text);
                z2 = Convert.ToDouble(textz2.Text);
                a = (b + (z1 + z2) * 0.5 * yn) * yn;
                p = b + yn * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                rh = a / p;
```

```
                s = Math.Pow(q * n / (a * Math.Pow(rh, 2 / 3d)), 2);
                tr = b + (z1 + z2) * yn;
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textS.Text = Convert.ToString(Math.Round(s, 6));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos pendiente, rugosidad, base,taludes y tirante
            if (textQ.Text == "" && textS.Text != "" && textB.Text != "" && textyd.Text != "" && textz1.Text ↙
 != "" && textz2.Text != "" && textn.Text != "")
            {
                s = Convert.ToDouble(textS.Text);
                n = Convert.ToDouble(textn.Text);
                b = Convert.ToDouble(textB.Text);
                yn = Convert.ToDouble(textyd.Text);
                z1 = Convert.ToDouble(textz1.Text);
                z2 = Convert.ToDouble(textz2.Text);
                a = (b + (z1 + z2) * 0.5 * yn) * yn;
                p = b + yn * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                rh = a / p;
                q = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / n;
                tr = b + (z1 + z2) * yn;
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            ////////////////////////////////// Aquí involucramos el valor de la velocidad////////////////////↙
 ////
```

```
            // caso donde conocemos pendiente, base, tirante, taludes y velocidad
            if (textQ.Text == "" && textS.Text != "" && textB.Text != "" && textyd.Text != "" && textn.Text ↙
== "" && textV.Text != "" && textz1.Text!="" && textz2.Text!="")
            {
                s = Convert.ToDouble(textS.Text);
                v = Convert.ToDouble(textV.Text);
                b = Convert.ToDouble(textB.Text);
                yn = Convert.ToDouble(textyd.Text);
                z1 = Convert.ToDouble(textz1.Text);
                z2 = Convert.ToDouble(textz2.Text);
                a = (b + (z1 + z2) * 0.5 * yn) * yn;
                p = b + yn * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                rh = a / p;
                q = v * a;
                tr = b+(z1+z2)*yn;
                n = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / q;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textn.Text = Convert.ToString(Math.Round(n, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos rugosidad, base, tirante, taludes y velocidad
            if (textQ.Text == "" && textS.Text == "" && textB.Text != "" && textyd.Text != "" && textn.Text ↙
!= "" && textV.Text != "" && textz1.Text != "" && textz2.Text != "")
            {
                n = Convert.ToDouble(textn.Text);
                v = Convert.ToDouble(textV.Text);
                b = Convert.ToDouble(textB.Text);
                yn = Convert.ToDouble(textyd.Text);
                z1 = Convert.ToDouble(textz1.Text);
                z2 = Convert.ToDouble(textz2.Text);
                a = (b + (z1 + z2) * 0.5 * yn) * yn;
                p = b + yn * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                rh = a / p;
                q = v * a;
                tr = b + (z1 + z2) * yn;
                s = Math.Pow(q * n / (a * Math.Pow(rh, 2 / 3d)), 2);
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textS.Text = Convert.ToString(Math.Round(s, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
```

```
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

        }// termina radiotrapecio yn conocido

        // radiotriangulo/////
        if (radiotriangulo.Checked == true && texty.Enabled == false)
        {
            // caso donde conocemos gasto, pendiente, taludes y tirante
            if (textQ.Text != "" && textS.Text != "" && textyd.Text != "" && textz1.Text != "" && textz2.
Text != "" && textn.Text == "")
            {
                q = Convert.ToDouble(textQ.Text);
                s = Convert.ToDouble(textS.Text);
                yn = Convert.ToDouble(textyd.Text);
                z1 = Convert.ToDouble(textz1.Text);
                z2 = Convert.ToDouble(textz2.Text);
                a =  (z1 + z2) * 0.5 * yn * yn;
                p =  yn * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                rh = a / p;
                n = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / q;
                tr = (z1 + z2) * yn;
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textn.Text = Convert.ToString(Math.Round(n, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos gasto, rugosidad, taludes y tirante
            if (textQ.Text != "" && textS.Text == "" && textyd.Text != "" && textz1.Text != "" && textz2.
Text != "" && textn.Text != "")
            {
                q = Convert.ToDouble(textQ.Text);
                n = Convert.ToDouble(textn.Text);
                yn = Convert.ToDouble(textyd.Text);
                z1 = Convert.ToDouble(textz1.Text);
                z2 = Convert.ToDouble(textz2.Text);
                a = (z1 + z2) * 0.5 * yn * yn;
                p = yn * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                rh = a / p;
                s = Math.Pow(q * n / (a * Math.Pow(rh, 2 / 3d)), 2);
```

```
                tr = (z1 + z2) * yn;
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textS.Text = Convert.ToString(Math.Round(s, 6));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos Pendiente, rugosidad, taludes y tirante
            if (textQ.Text == "" && textS.Text != "" && textyd.Text != "" && textz1.Text != "" && textz2.
Text != "" && textn.Text != "")
            {
                s = Convert.ToDouble(textS.Text);
                n = Convert.ToDouble(textn.Text);
                yn = Convert.ToDouble(textyd.Text);
                z1 = Convert.ToDouble(textz1.Text);
                z2 = Convert.ToDouble(textz2.Text);
                a = (z1 + z2) * 0.5 * yn * yn;
                p = yn * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                rh = a / p;
                q = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / n;
                tr = (z1 + z2) * yn;
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            ////////////////////////////////// Aquí involucramos el valor de la velocidad//////////////////
////

            // caso donde conocemos pendiente, tirante, taludes y velocidad
```

```
            if (textQ.Text == "" && textS.Text != "" && textyd.Text != "" && textn.Text == "" && textV.Text ↙
!= "" && textz1.Text != "" && textz2.Text != "")
            {
                s = Convert.ToDouble(textS.Text);
                v = Convert.ToDouble(textV.Text);
                yn = Convert.ToDouble(textyd.Text);
                z1 = Convert.ToDouble(textz1.Text);
                z2 = Convert.ToDouble(textz2.Text);
                a = (z1 + z2) * 0.5 * yn * yn;
                p = yn * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                rh = a / p;
                q = v * a;
                tr = (z1 + z2) * yn;
                n = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / q;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textn.Text = Convert.ToString(Math.Round(n, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos rugosidad, tirante, taludes y velocidad
            if (textQ.Text == "" && textS.Text == "" && textyd.Text != "" && textn.Text != "" && textV.Text ↙
!= "" && textz1.Text != "" && textz2.Text != "")
            {
                n = Convert.ToDouble(textn.Text);
                v = Convert.ToDouble(textV.Text);
                yn = Convert.ToDouble(textyd.Text);
                z1 = Convert.ToDouble(textz1.Text);
                z2 = Convert.ToDouble(textz2.Text);
                a = (z1 + z2) * 0.5 * yn * yn;
                p = yn * (Math.Sqrt(1 + z1 * z1) + Math.Sqrt(1 + z2 * z2));
                rh = a / p;
                q = v * a;
                tr = (z1 + z2) * yn;
                s = Math.Pow(q * n / (a * Math.Pow(rh, 2 / 3d)), 2);
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textS.Text = Convert.ToString(Math.Round(s, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
```

```
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

        }// termina radiotriángulo yn conocido

        if (radiocirculo.Checked == true && texty.Enabled == false)
        {
            // caso donde conocemos gasto,pendiente, diámetro y tirante
            if (textQ.Text != "" && textS.Text != "" && textyd.Text != "" && textD.Text!="" && textn.Text == ↙
 "")
            {
                q = Convert.ToDouble(textQ.Text);
                s = Convert.ToDouble(textS.Text);
                yn = Convert.ToDouble(textyd.Text);
                d = Convert.ToDouble(textD.Text);
                o = 2 * Math.Acos(1 - yn / (0.5 * d));
                a = (1 / 8d) * (o - Math.Sin(o)) * d * d;
                p = 0.5 * o * d;
                rh = a / p;
                n = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / q;
                tr = 2*Math.Sqrt(yn*(d-yn));
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textn.Text = Convert.ToString(Math.Round(n, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos gasto, Rugosidad, diámetro y tirante
            if (textQ.Text != "" && textS.Text == "" && textyd.Text != "" && textD.Text != "" && textn.Text ↙
 != "")
            {
                q = Convert.ToDouble(textQ.Text);
                n = Convert.ToDouble(textn.Text);
                yn = Convert.ToDouble(textyd.Text);
                d = Convert.ToDouble(textD.Text);
                o = 2 * Math.Acos(1 - yn / (0.5 * d));
                a = (1 / 8d) * (o - Math.Sin(o)) * d * d;
                p = 0.5 * o * d;
                rh = a / p;
                s = Math.Pow(q * n / (a * Math.Pow(rh, 2 / 3d)), 2);
                tr = 2 * Math.Sqrt(yn * (d - yn));
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
```

```
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textS.Text = Convert.ToString(Math.Round(s, 6));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos Pendiente, Rugosidad, diámetro y tirante
            if (textQ.Text == "" && textS.Text != "" && textyd.Text != "" && textD.Text != "" && textn.Text ↙
!= "")
            {
                s = Convert.ToDouble(textS.Text);
                n = Convert.ToDouble(textn.Text);
                yn = Convert.ToDouble(textyd.Text);
                d = Convert.ToDouble(textD.Text);
                o = 2 * Math.Acos(1 - yn / (0.5 * d));
                a = (1 / 8d) * (o - Math.Sin(o)) * d * d;
                p = 0.5 * o * d;
                rh = a / p;
                q = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / n;
                tr = 2 * Math.Sqrt(yn * (d - yn));
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            ////////////////////////////////// Aquí involucramos el valor de la velocidad//////////////////// ↙
////

            // caso donde conocemos pendiente, tirante, diámetro y velocidad
            if (textQ.Text == "" && textS.Text != "" && textyd.Text != "" && textn.Text == "" && textV.Text ↙
!= "" && textD.Text!="")
            {
                s = Convert.ToDouble(textS.Text);
                v = Convert.ToDouble(textV.Text);
```

```
                yn = Convert.ToDouble(textyd.Text);
                d = Convert.ToDouble(textD.Text);
                o = 2 * Math.Acos(1 - yn / (0.5 * d));
                a = (1 / 8d) * (o - Math.Sin(o)) * d * d;
                p = 0.5 * o * d;
                rh = a / p;
                q = v * a;
                tr = 2 * Math.Sqrt(yn * (d - yn));
                n = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / q;
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textn.Text = Convert.ToString(Math.Round(n, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos rugosidad, tirante, diámetro y velocidad
            if (textQ.Text == "" && textS.Text == "" && textyd.Text != "" && textn.Text != "" && textV.Text ↙
!= "" && textD.Text!="")
            {
                n = Convert.ToDouble(textn.Text);
                v = Convert.ToDouble(textV.Text);
                yn = Convert.ToDouble(textyd.Text);
                d = Convert.ToDouble(textD.Text);
                o = 2 * Math.Acos(1 - yn / (0.5 * d));
                a = (1 / 8d) * (o - Math.Sin(o)) * d * d;
                p = 0.5 * o * d;
                rh = a / p;
                q = v * a;
                tr = 2 * Math.Sqrt(yn* (d - yn));
                s = Math.Pow(q * n / (a * Math.Pow(rh, 2 / 3d)), 2);
                fr = v / Math.Sqrt(9.81 * (a / tr));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textS.Text = Convert.ToString(Math.Round(s, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
```

```
                }
            }

        }// Termina radio círculo yn conocido

        if (radioparabola.Checked == true && texty.Enabled == false)
        {
            // caso donde conocemos gasto,pendiente, ancho  superficial y tirante
            if (textQ.Text != "" && textS.Text != "" && textyd.Text != "" && textT.Text != "" && textn.Text ↙
== "")
            {
                q = Convert.ToDouble(textQ.Text);
                s = Convert.ToDouble(textS.Text);
                yn = Convert.ToDouble(textyd.Text);
                t = Convert.ToDouble(textT.Text);
                a = (2 / 3d) * t * yn;
                x = 4 * yn / t;
                p = 0;
                if (x > 0 && x <= 1)
                {
                    p = t + (8 / 3d) * (yn * yn / t);
                }
                if (x > 1)
                {
                    p = (t * 0.5) * (Math.Sqrt(1 + x * x) + (1 / x) * Math.Log(x + Math.Sqrt(1 + x * x)));
                }

                rh = a / p;
                n = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / q;
                tr = (4*yn)/(t*t);
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / t));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textn.Text = Convert.ToString(Math.Round(n, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos gasto, rugosidad, ancho  superficial y tirante
            if (textQ.Text != "" && textS.Text == "" && textyd.Text != "" && textT.Text != "" && textn.Text ↙
!= "")
            {
                q = Convert.ToDouble(textQ.Text);
                n = Convert.ToDouble(textn.Text);
                yn = Convert.ToDouble(textyd.Text);
                t = Convert.ToDouble(textT.Text);
                a = (2 / 3d) * t * yn;
                x = 4 * yn / t;
                p = 0;
                if (x > 0 && x <= 1)
                {
                    p = t + (8 / 3d) * (yn * yn / t);
```

```
                }
                if (x > 1)
                {
                    p = (t * 0.5) * (Math.Sqrt(1 + x * x) + (1 / x) * Math.Log(x + Math.Sqrt(1 + x * x)));
                }

                rh = a / p;
                s = Math.Pow(q * n / (a * Math.Pow(rh, 2 / 3d)), 2);
                tr = (4 * yn) / (t * t);
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / t));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textS.Text = Convert.ToString(Math.Round(s, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos pendiente, rugosidad, ancho  superficial y tirante
            if (textQ.Text == "" && textS.Text != "" && textyd.Text != "" && textT.Text != "" && textn.Text ↙
!= "")
            {
                s = Convert.ToDouble(textS.Text);
                n = Convert.ToDouble(textn.Text);
                yn = Convert.ToDouble(textyd.Text);
                t = Convert.ToDouble(textT.Text);
                a = (2 / 3d) * t * yn;
                x = 4 * yn / t;
                p = 0;
                if (x > 0 && x <= 1)
                {
                    p = t + (8 / 3d) * (yn * yn / t);
                }
                if (x > 1)
                {
                    p = (t * 0.5) * (Math.Sqrt(1 + x * x) + (1 / x) * Math.Log(x + Math.Sqrt(1 + x * x)));
                }

                rh = a / p;
                q = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / n;
                tr = (4 * yn) / (t * t);
                v = q / a;
                fr = v / Math.Sqrt(9.81 * (a / t));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textV.Text = Convert.ToString(Math.Round(v, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
```

```
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            ////////////////////////////////// Aquí involucramos el valor de la velocidad//////////////////// ↙
    ////

            // caso donde conocemos pendiente, tirante, ancho superficial y velocidad
            if (textQ.Text == "" && textS.Text != "" && textyd.Text != "" && textn.Text == "" && textV.Text ↙
    != "" && textT.Text != "")
            {
                s = Convert.ToDouble(textS.Text);
                v = Convert.ToDouble(textV.Text);
                yn = Convert.ToDouble(textyd.Text);
                t = Convert.ToDouble(textT.Text);
                a = (2 / 3d) * t * yn;
                x = 4 * yn / t;
                p = 0;
                if (x > 0 && x <= 1)
                {
                    p = t + (8 / 3d) * (yn * yn / t);
                }
                if (x > 1)
                {
                    p = (t * 0.5) * (Math.Sqrt(1 + x * x) + (1 / x) * Math.Log(x + Math.Sqrt(1 + x * x)));
                }

                rh = a / p;
                q = v * a;
                tr = (4 * yn) / (t * t);
                n = (a * Math.Pow(rh, 2 / 3d) * Math.Sqrt(s)) / q;
                fr = v / Math.Sqrt(9.81 * (a / t));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textn.Text = Convert.ToString(Math.Round(n, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

            // caso donde conocemos rugosidad, tirante, ancho superficial y velocidad
            if (textQ.Text == "" && textS.Text == "" && textyd.Text != "" && textn.Text != "" && textV.Text ↙
    != "" && textT.Text != "")
            {
                n = Convert.ToDouble(textn.Text);
```

```
                v = Convert.ToDouble(textV.Text);
                yn = Convert.ToDouble(textyd.Text);
                t = Convert.ToDouble(textT.Text);
                a = (2 / 3d) * t * yn;
                x = 4 * yn / t;
                p = 0;
                if (x > 0 && x <= 1)
                {
                    p = t + (8 / 3d) * (yn * yn / t);
                }
                if (x > 1)
                {
                    p = (t * 0.5) * (Math.Sqrt(1 + x * x) + (1 / x) * Math.Log(x + Math.Sqrt(1 + x * x)));
                }

                rh = a / p;
                q = v * a;
                tr = (4 * yn) / (t * t);
                s = Math.Pow(q * n / (a * Math.Pow(rh, 2 / 3d)), 2);
                fr = v / Math.Sqrt(9.81 * (a / t));
                en = yn + (v * v / (2 * 9.81));
                textA.Text = Convert.ToString(Math.Round(a, 4));
                textP.Text = Convert.ToString(Math.Round(p, 4));
                textRh.Text = Convert.ToString(Math.Round(rh, 4));
                textTR.Text = Convert.ToString(Math.Round(tr, 4));
                textS.Text = Convert.ToString(Math.Round(s, 4));
                textFr.Text = Convert.ToString(Math.Round(fr, 4));
                textE.Text = Convert.ToString(Math.Round(en, 4));
                textQ.Text = Convert.ToString(Math.Round(q, 4));
                if (fr < 1)
                {
                    textTF.Text = "Subcrítico";
                }
                if (fr == 1)
                {
                    textTF.Text = "Crítico";
                }
                if (fr > 1)
                {
                    textTF.Text = "Supercrítico";
                }
            }

        }// termina radioparabola


        ////////////////////////////////////////////// Cálculos para sección pendiente//////////////////// ↙
    /////////////////

        double a1, a2, p1, p2, rh1, rh2, n1, n2, alp1, alp2, dx, dy, dif, ka, kb, kr, kt, Q0, va, vb, valp, ↙
    vbalp, s1, Q1, dif2;

        if( textkt.Enabled==false )
        {
            a1 = 0;
            rh1 = 0;
            a2 = 0;
            rh2 = 0;
            // Comprobación de
            if (textP1.Text != "" && textP2.Text != "" && textrh1.Text == "" && textrh2.Text == "")
            {
                a1 = Convert.ToDouble(textA1.Text);
                p1 = Convert.ToDouble(textP1.Text);
                rh1 = a1 / p1;
                a2=  Convert.ToDouble(textA2.Text);
                p2 = Convert.ToDouble(textP2.Text);
                rh2 = a2 / p2;
                textrh1.Text = Convert.ToString(Math.Round(rh1, 4));
                textrh2.Text = Convert.ToString(Math.Round(rh2, 4));
```

```
            }

            if (textP1.Text == "" && textP2.Text == "" && textrh1.Text != "" && textrh2.Text != "")
            {
                a1 = Convert.ToDouble(textA1.Text);
                rh1 = Convert.ToDouble(textrh1.Text);
                p1 = a1 / rh1;
                a2 = Convert.ToDouble(textA2.Text);
                rh2 = Convert.ToDouble(textrh2.Text);
                p2 = a2 / rh2;
                textP1.Text = Convert.ToString(Math.Round(p1, 4));
                textP2.Text = Convert.ToString(Math.Round(p2, 4));
            }

            n1 = Convert.ToDouble(textn1.Text);
            n2 = Convert.ToDouble(textn2.Text);
            alp1 = Convert.ToDouble(textalp1.Text);
            alp2 = Convert.ToDouble(textalp2.Text);
            dx = Convert.ToDouble(textdx.Text);
            dy = Convert.ToDouble(textdy.Text);
            dif = Convert.ToDouble(textdif.Text);
            ka = (a1 * Math.Pow(rh1, 2 / 3d)) / n1;
            kb = (a2 * Math.Pow(rh2, 2 / 3d)) / n2;
            kr = Math.Sqrt(ka * kb);
            Q0 = kr * Math.Sqrt(dy / dx);
            va = Q0 / a1;
            vb = Q0 / a2;
            kt = 0;

            if (a1 < a2 && va > vb)
            {
                DialogResult vent = MessageBox.Show("Expansión Brusca:  " + 0.5 + "  Expansión Gradual:  " + ↙
 0.7 + ". Presione Sí para 0.5 y presiones No para 0.7 ", "Elección de k testada ", MessageBoxButtons.YesNo, ↙
 MessageBoxIcon.Question);
                if (vent == DialogResult.Yes)// si pulsa Sí, usaremos Diam
                {
                    kt = 0.5;
                }

                if (vent == DialogResult.No)// si pulsa Sí, usaremos Diam
                {
                    kt = 0.7;
                }
            }

            if (a1 > a2 && va < vb)
            {
                DialogResult vent = MessageBox.Show("Contracción Gradual:  " + 1.2 + "  Contracción Brusca: ↙
  " + 1.5 + ". Presione Sí para 1.2 y presiones No para 1.5 ", "Elección de k testada ", MessageBoxButtons.  ↙
 YesNo, MessageBoxIcon.Question);
                if (vent == DialogResult.Yes)// si pulsa Sí, usaremos Diam
                {
                    kt = 1.2;
                }

                if (vent == DialogResult.No)// si pulsa Sí, usaremos Diam
                {
                    kt = 1.5;
                }
            }

            if (a1 == a2 && va == vb)
            {
                kt = 1;
            }

            valp = alp1 * (va * va) / (2 * 9.81);
            vbalp = alp2 * (vb * vb) / (2 * 9.81);
            s1 = (dy + kt * (valp - vbalp)) / (dx);
```

```
                Q1 = kr * Math.Sqrt(s1);
                if (Q0 == Double.NaN)
                {

                }
                dif2 = Math.Abs(Q0 - Q1);
                conteo = 1;

                do
                {
                    if (dy < 0.15)
                    {
                        MessageBox.Show("La diferencia de altura debe ser mayor o igual a 15 cm", "Error");
                        break;
                    }
                Q0 = (Q0+Q1)*0.5;
                va = Q0 / a1;
                vb = Q0 / a2;
                valp = alp1 * (va * va) / (2 * 9.81);
                vbalp = alp2 * (vb * vb) / (2 * 9.81);
                s1 = (dy + kt * (valp - vbalp)) / (dx);
                Q1 = kr * Math.Sqrt(s1);
                dif2 = Math.Abs(Q0 - Q1);
                conteo++;
                }

                while(dif2>dif && dif2>0.0000001 && conteo<it );

                textka.Text = Convert.ToString(Math.Round(ka, 6));
                textkb.Text = Convert.ToString(Math.Round(kb, 6));
                textkab.Text = Convert.ToString(Math.Round(kr, 6));
                textkt.Text = Convert.ToString(kt);
                textQ0.Text = Convert.ToString(Math.Round(Q0, 6));
                textva.Text = Convert.ToString(Math.Round(va, 6));
                textvb.Text = Convert.ToString(Math.Round(vb, 6));
                textvalp.Text = Convert.ToString(Math.Round(valp, 6));
                textvbalp.Text = Convert.ToString(Math.Round(vbalp, 6));
                textct.Text = Convert.ToString(conteo);
                textS1.Text = Convert.ToString(Math.Round(s1, 6));
                textQ11.Text = Convert.ToString(Math.Round(Q1, 6));
                textQ0Q1.Text = Convert.ToString(Math.Round(dif2, 6));

            }

        }// final del botón calcular




    }
}
```

Campus Guanajuato | División de Ingenierías

Guanajuato, Gto., 26 de octubre de 2021.

Asunto: se notifica liberación de tesis.

A QUIEN CORRESPONDA:

En seguimiento al proceso de titulación por la **Modalidad de Trabajo de Tesis**, le informo que, de acuerdo con el cumplimiento de los requisitos de egreso de los programas educativos de Licenciatura de la División de Ingenierías, el trabajo de tesis denominado:

"ANÁLISIS HIDRODINÁMICO DEL FLUJO UNIFORME A SUPERFICIE LIBRE"

Desarrollado por el estudiante **Martín Estrada Vaca, PE INGENIERÍA CIVIL;** ha sido revisado por los sinodales, los cuales de común acuerdo con el Director de Trabajo han otorgado la liberación de tesis.

Lo anterior, a fin de proceder al ***cumplimiento de validación de Biblioteca para trámite de titulación*** y continuar con el proceso para la **obtención del grado de Ingeniero Civil.**

Sin más por el momento, reciba un cordial saludo.

Atentamente,

"La verdad os hará libres"

El Director

Universidad de Guanajuato
División de Ingenierías
Campus Guanajuato
DIRECCIÓN

Dr. Gilberto Carreño Aguilera.

DR.GCA/LASR.