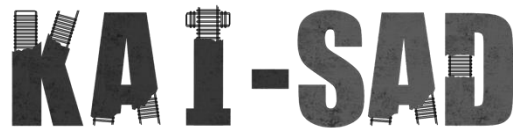


***KAI-SAD SOFTWARE EDUCATIVO PARA EL ANÁLISIS Y ASIGNACIÓN DE
ACEROS A COMPRESIÓN, TENSIÓN Y CORTANTE EN VIGAS RECTANGULARES
DE CONCRETO REFORZADO.***



MONROY ARDILA, KELLY VANESA Autor 01. 1075288004 – 1320026.
BELTRÁN BONILLA, ADRIÁN CAMILO Autor 02. 1024480427 – 1310691.

TRABAJO DE GRADO PARA OBTENER EL TÍTULO DE INGENIERO CIVIL

DIRECTOR: MORA SAMACÁ, JAIME IVÁN

CORPORACIÓN UNIVERSIDAD PILOTO DE COLOMBIA
PROGRAMA DE INGENIERÍA CIVIL
BOGOTÁ D.C – 2019

CONTENIDO

1. INTRODUCCIÓN.	7
1.1. JUSTIFICACIÓN DEL ESTUDIO.	7
1.2. PLANTEAMIENTO DEL PROBLEMA.	8
2. OBJETIVOS.	9
2.1. OBJETIVO GENERAL.	9
2.2. OBJETIVO ESPECÍFICO.	9
3. MARCO TEÓRICO.	10
3.1 CONCEPTOS ESTRUCTURALES	10
3.1.1 CARGAS.	10
3.1.2 APOYOS.	12
3.1.3 MÉTODO DE HARDY CROSS.	14
3.1.4 DISEÑO DE VIGAS RECTANGULARES A FLEXIÓN POR EL MÉTODO DE LA RESISTENCIA ÚLTIMA (FY)	14
3.1.4.1 SECCIÓN BALANCEADA	15
3.1.4.2 . MÉTODO DE LA RESISTENCIA ÚLTIMA PARA EL DISEÑO DE VIGAS RECTANGULARES A FLEXIÓN.	15
3.1.5 DISEÑO A CORTANTE.	16
3.1.6 ASIGNACIÓN DE ACEROS EN UNA VIGA.	17
3.2 CONCEPTOS DE PROGRAMACIÓN.	17
3.2.1 DESARROLLO, ESTRUCTURA DE CODIGOS DE KAI-SAD.	17
3.2.2 CREACIÓN DE VARIABLES.	19
3.2.3 CREACIÓN CAJAS DE TEXTO, CAJAS DE TEXTO, CAJAS NUMÉRICAS, VISTAS DESPLEGABLES E IMÁGENES.	21
3.2.4 GRÁFICA	21
3.3 DIAGRAMAS DE FLUJO	23
3.3.1 ANÁLISIS ESTRUCTURAL.	23
3.3.1. ASIGNACIÓN DE ACERO.	¡ERROR! MARCADOR NO DEFINIDO.
4. MARCO METODOLÓGICO	25
4.1. PESTAÑA MENÚ	25
4.2. ANÁLISIS ESTRUCTURAL (PROCEDIMIENTO 1)	26
4.2.1. PANEL DE “DATOS INICIALES”	26
4.2.2. PANEL DE “CARGAS Y LUCES”	30
4.2.3 PANEL DE “CALCULAR RESULTADO”	33
4.2.4. PANEL DE “GRÁFICA CORTANTE”	33
4.3. ASIGNACIÓN DE ACEROS (PROCEDIMIENTO 2)	34
4.3.1. PANEL DE “DATOS INICIALES”	34

4.3.1.1. CONFIGURACIÓN CARGA MUERTA	35
4.3.1.2. CONFIGURACIÓN CARGA VIVA:	36
4.3.2. PANEL DE “CARGAS Y LUCES”	36
4.3.3. PANEL DE “CALCULAR RESULTADO”	37
4.3.4. PANEL DE “GRÁFICA CORTANTE”	38
4.3.5. PANEL DE “GRÁFICA DE MOMENTO”	38
4.3.6. PANEL DE “REFUERZOS LONGITUDINALES”	38
4.3.7. PANEL DE “REFUERZOS TRANSVERSALES”	38

5. CÁLCULOS **39**

5.1. EJERCICIO DE APLICACIÓN 1, ANÁLISIS ESTRUCTURAL.	39
5.1.1. CÁLCULO DEL MOMENTO DE EMPOTRAMIENTO PERFECTO.	39
5.1.2. CÁLCULO DE MOMENTOS POR EL MÉTODO DE HARDY CROSS.	42
5.1.2.1. CÁLCULO DE LA RIGIDEZ RELATIVA.	43
5.1.2.2. CÁLCULO FACTOR DE DISTRIBUCIÓN	44
5.1.2.3. CÁLCULOS POR EL MÉTODO DE HARDY CROSS.	45
5.1.3 CÁLCULO DE LAS REACCIONES DE CADA APOYO	50
5.1.4. DIAGRAMA DE FUERZA CORTANTE.	52
5.2. EJERCICIO DE APLICACIÓN 2, ASIGNACIÓN DE ACEROS	52
5.2.1. CÁLCULO DEL AVALÚO DE CARGA.	53
5.2.2. DIAGRAMAS DE FUERZA CORTANTE Y MOMENTO FLECTOR	57
5.2.3. CÁLCULO REFUERZOS LONGITUDINALES.	58
5.2.4. CÁLCULO REFUERZOS TRANSVERSALES.	67
5.3. EJERCICIO DE APLICACIÓN 3	78
5.3.1. DIAGRAMAS DE FUERZA CORTANTE Y MOMENTO FLECTOR	79
5.3.2. CÁLCULO REFUERZOS LONGITUDINALES	80
5.3.3. CÁLCULO REFUERZOS TRANSVERSALES.	90

6. COMPARACIÓN ENTRE KAI SAD Y SAP2000. **103**

6.1. EJERCICIO DE APLICACIÓN 1 Y SAP2000	103
6.1.1. DIAGRAMAS DE CARGA.	104
6.1.2. DIAGRAMAS DE FUERZA CORTANTE.	105
6.2. EJERCICIO DE APLICACIÓN 2 Y SAP2000	105
6.2.1. DIAGRAMAS DE CARGA.	105
6.2.2. DIAGRAMAS DE FUERZA CORTANTE.	106
6.2.3. DIAGRAMAS DE MOMENTOS FLECTORES.	107
6.3. EJERCICIO DE APLICACIÓN 3 Y SAP2000	107
6.3.1. DIAGRAMAS DE CARGA.	107
6.3.2. DIAGRAMAS DE FUERZA CORTANTE.	108
6.3.3. DIAGRAMAS DE MOMENTOS FLECTORES.	109
6.3. RANGO DE ERROR	110
6.3.1. RANGO DE ERROR ENTRE EL EJERCICIO DE APLICACIÓN 1 EN KAI SAD Y SAP2000.	110
6.3.2. RANGO DE ERROR ENTRE EL EJERCICIO DE APLICACIÓN 2 EN KAI SAD Y SAP2000.	111
6.3.3. RANGO DE ERROR ENTRE EL EJERCICIO DE APLICACIÓN 3 EN KAI SAD Y SAP2000.	111

7. CONCLUSIONES	112
8. BIBLIOGRAFÍA	114
9. ANEXOS.	115

LISTADO DE IMÁGENES

Imagen 1. Carga uniformemente distribuida

Imagen 2. Carga puntual céntrica.

Imagen 3. Carga puntual excéntrica.

Imagen 4. Carga distribuida variable.

Imagen 5. Método de la resistencia última. Tomando de Diseño Básico de Vigas Vol. 1.

Imagen 6. Falla por cortante. Tomando de Diseño Básico de Vigas Vol. 1.

Imagen 7. Creación del proyecto en Visual Studio

Imagen 8. Selección tipo de plataforma.

Imagen 9. Creación estructura del proyecto.

Imagen 10. Recorte de pantalla del menú de KAI SAD

Imagen 11. Recorte de pantalla de datos iniciales del programa KAI SAD.

Imagen 12. Recorte de pantalla del listado de 9 luces KAI SAD.

Imagen 13. Carga uniformemente distribuida

Imagen 14. Carga triangular descendiente.

Imagen 15. Carga triangular ascendiente

Imagen 16. Carga puntual céntrica. Fuente.

Imagen 17. Carga puntual excéntrica.

Imagen 18. Recorte de pantalla del panel de KAI SAD.

Imagen 19. Recorte de pantalla del panel de asignación de acero de KAI SAD

Imagen 20. Recorte de pantalla de los cuadros de la carga muerta de KAI SAD.

Imagen 21. Recorte de pantalla de listado 9 luces de KAI SAD.

Imagen 22. Recorte de pantalla mostrando la ubicación de los momentos de empotramiento perfecto.

Imagen 23. Recorte de pantalla mostrando los resultados de la primera iteración

Imagen 24. Recorte de pantalla mostrando como se ubica la división en dos de la primera iteración

LISTADO DE DIAGRAMAS Y GRAFICAS.

Diagrama 1. Diagrama de flujo del procedimiento 1: Análisis Estructural

Diagrama 2. Diagrama de flujo del procedimiento 2: Asignación de Aceros

Gráfica 1. Viga rectangular isostática e hiperestática simplemente apoyada.

Gráfica 2. Viga rectangular isostática e hiperestática con inicio apoyo y final voladizo

Gráfica 3. Viga rectangular isostática e hiperestática con inicio voladizo y final apoyo

Gráfica 4. Viga rectangular isostática e hiperestática con inicio voladizo y final voladizo

Gráfica 5. Viga rectangular isostática e hiperestática con inicio empotrado y final voladizo

Gráfica 6. Viga rectangular isostática e hiperestática con inicio voladizo y final empotrado

Gráfica 7. Viga rectangular isostática e hiperestática con inicio empotrado y final apoyo

Gráfica 8. Viga rectangular isostática e hiperestática con inicio apoyo y final empotrado.

Gráfica 9. Diagrama de carga viga hiperestática 1.

Gráfica 10. Diagrama de corte a-a en voladizo de viga hiperestática 1.

Gráfica 11. Diagrama sección transversal de viga hiperestática 1

Gráfica 12. Diagrama de cuerpo libre de viga hiperestática 1.

Gráfica 13. Diagrama de gráfica cortante de viga hiperestática 1

Gráfica 14. Dimensiones de la viga y la vigueta.

Gráfico 15. Diagrama de carga viga hiperestática 2.

Gráfica 16. Diagrama de gráfica cortante de viga hiperestática 2.

Gráfica 17. Diagrama de gráfica momento flector de viga hiperestática 2

Gráfica 18. Diagrama de cortante

Gráfica 19. Diagrama de cortante. Cantidad y separación entre flejes

Gráfico 20. Diagrama de carga viga hiperestática 3.

Gráfica 21. Diagrama de gráfica cortante de viga hiperestática 3.

Gráfica 22. Diagrama de gráfica momento flector de viga hiperestática 3

Gráfico 23. Recorte de pantalla Diagrama de carga de viga hiperestática 1 en el programa SAP2000.

Grafico 24. Recorte de pantalla Diagrama de gráfica cortante de viga hiperestática 1 en el programa SAP2000.

Gráfica 25. Recorte de pantalla Diagrama de carga de viga hiperestática 2 en el programa SAP2000.

Gráfica 26. Recorte de pantalla Diagrama de gráfica cortante de viga hiperestática 2 en el programa SAP2000.

Gráfica 27. Recorte de pantalla Diagrama de gráfica de momento flector de viga hiperestática 2 en el programa SAP2000.

Gráfica 28. Recorte de pantalla Diagrama de carga de viga hiperestática 3 en el programa SAP2000.

Gráfica 29. Recorte de pantalla Diagrama de gráfica cortante de viga hiperestática 3 en el programa SAP2000.

Gráfica 30. Recorte de pantalla Diagrama de gráfica de momento flector de viga hiperestática 3 en el programa SAP2000.

LISTADO DE TABLAS.

Tabla 1. Método iterativo de Hardy Cross

Tabla 2. Avalúo de cargas para la carga muerta

Tabla 3. Avalúo de cargas del ejercicio de aplicación 2.

Tabla 4. Porcentaje del Rango de error del ejercicio de aplicación 1 entre KAI SAD y SAP2000.

Tabla 5. Porcentaje del Rango de error del ejercicio de aplicación 2 entre KAI SAD y SAP2000.

Tabla 6. Porcentaje del Rango de error del ejercicio de aplicación 3 entre KAI SAD y SAP2000.

1. INTRODUCCIÓN.

1.1. JUSTIFICACIÓN DEL ESTUDIO.

Surge la iniciativa a partir de la intención de mejorar la comprensión y dar mayor claridad a los temas vistos en la línea de estructuras de la Universidad Piloto de Colombia que comprende dos materias en específico, Análisis estructural y Estructuras en concreto, definidas en el pensum de la universidad, además de la implementación de herramientas tecnológicas.

Los temas por tratar en el desarrollo del software son basados en conceptos impartidos por los docentes de cátedra y evaluados a partir de la metodología definida por cada uno, este software que se sugiere sea implementado al final de cada una de estas cátedras, ratificará y verificará que los procesos fueron comprendidos y aplicados, para vigas rectangulares isostáticas e hiperestáticas expuestas en el marco teórico.

El software tiene por argumento y soporte el libro “*Diseño básico de concreto reforzado volumen 1 y 2*”, cuyo autor es el ingeniero y actual docente de cátedra de la universidad, Jaime Iván Mora Samacá, basados en el Reglamento Colombiano de Construcción Sismo Resistente NSR-10, y demás autores especificados a lo largo de este informe.

1.2. PLANTEAMIENTO DEL PROBLEMA.

Con base en lo expuesto en la justificación, se correlaciona la problemática en la falencia de la verificación y comprensión de información e implementación de herramientas tecnológicas, ya que este es el siglo del desarrollo tecnológico, se busca la integración de las tecnologías de la información y la comunicación (TIC).

Las TIC buscarán una mayor interacción entre el docente y el alumno brindando una mayor efectividad en el desarrollo de las materias, pues está demostrado que la tecnología ha tenido a lo largo de la historia un impacto positivo para el desarrollo y optimización de procesos.

Pensando en la mejora a la educación y con base en softwares ya integrados como SAP 2000 y AUTOCAD, que han tenido éxito en el futuro no solo de los estudiantes si no de los profesionales, se cree que la implementación de este software llegará a ser factible para cumplir los objetivos propuestos.

KAI – SAD, buscará corresponder a estas necesidades a través de los conceptos definidos a lo largo del desarrollo del informe, ¿se podrá optimizar KAI- SAD para corresponder a las falencias del mismo?

2. OBJETIVOS.

2.1. OBJETIVO GENERAL.

Crear un software educativo, que mejore la comprensión y claridad de temas vistos en la línea de estructuras específicamente a las asignaturas: Análisis Estructural y Estructuras de concreto, con argumento y soporte en los libros “*Diseño básico de concreto reforzado volumen 1 y 2*”.

2.2. OBJETIVOS ESPECÍFICOS.

- Crear un algoritmo específico con una interfaz dinámica para el desarrollo de Análisis Estructural, que dé como resultados los diagramas de fuerza cortante y valores de momento máximo, de las posibles configuraciones de vigas dentro de los parámetros definidos anteriormente en la justificación a partir de las restricciones y normas.
- Crear un algoritmo específico con una interfaz dinámica para el desarrollo de asignación de aceros donde se muestra aceros a cortante y a flexión para las posibles configuraciones de vigas definidas anteriormente en la justificación y en el apartado de análisis estructural a partir de las restricciones y normas.
- Comprobar el software mediante un ejercicio operado manualmente y ejecutado en SAP 2000.

3. MARCO TEÓRICO.

3.1 CONCEPTOS ESTRUCTURALES

Este capítulo está orientado a comprender conceptos referentes a las generalidades respecto a la línea estructural.

3.1.1 Cargas.

Para la elaboración del software es necesario comprender el concepto de cargas y se define como las fuerzas a las que está sometida cada elemento de la estructura. Para este análisis la investigación se centrará en las cargas verticales o gravitacionales y son aquellas que se ejercen sobre los distintos elementos estructurales que integran la construcción debido a su funcionamiento, es decir, las cargas vivas y muertas que en ellas actúan durante la operación usual del edificio (Montero, 2004). Las cargas muertas pueden definirse como aquellas que son constantes en magnitud y fijas durante la vida de la estructura. Se pueden calcular a partir de la configuración de diseño, de las dimensiones de la estructura y densidad del material. (NSR-10, 2010) . Ahora bien, las cargas vivas se refieren a las fuerzas de ocupación en edificios y/o cargas de tráfico. Pueden estar total o parcialmente en su sitio o no estar presentes y pueden cambiar su ubicación. Las cargas vivas mínimas para las cuales debe diseñarse, en este caso, están contempladas dentro del Reglamento Colombiano de Construcción Sismo Resistente NSR-2010. (NSR-10, 2010). Las maneras en que las cargas interactúan con el elemento, en este caso la viga, son de la siguiente forma:

- *carga uniformemente distribuida*, es una magnitud de la fuerza que ha sido distribuida de manera uniforme, en toda una longitud de un elemento estructural o a una parte de este.

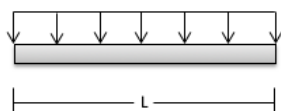


Imagen 1. Carga uniformemente distribuida Fuente. KAI-SAD

- *carga puntual céntrica*, carga que actúa sobre un área concreta, en la mitad de un elemento estructural.

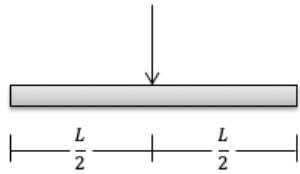


Imagen 2. Carga puntual céntrica. Fuente. KAI-SAD

- *carga puntual excéntrica* carga que actúa sobre un área concreta y que está a fuera del centro observado.

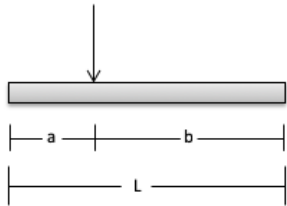


Imagen 3. Carga puntual excéntrica. Fuente. KAI-SAD

- *carga distribuida variablemente (Triangular)* contempla dos casos posibles: triángulos creciente y decreciente (IngMario, s.f.).

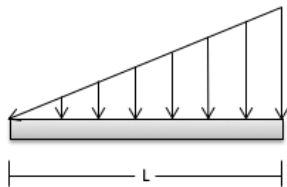


Imagen 4. Carga distribuida variable. Fuente. KAI-SAD

Para la investigación es clave comprender conceptos referentes a elementos estructurales que se integraran en el desarrollo del programa, haciendo énfasis en vigas y apoyos. Las vigas se denominan como un elemento estructural lineal que trabaja principalmente a flexión (Pérez, 1990). En las vigas la longitud predomina sobre las otras dos dimensiones, base y altura que son componentes de una sección transversal, que refiere a la proyección de una sección de

un objeto que se ha realizado mediante un corte perpendicular al eje largo del mismo. (www.parro.com.ar, s.f.).

3.1.2 Apoyos.

Una vez establecido la definición de la viga, es importante determinar sobre que se soporta de aquí nace el concepto de apoyo, que se usa para restringir el movimiento bien sea horizontal (a largo del eje x), vertical (a lo largo del eje y) y rotacional (alrededor del eje z). Según sus características, puede imposibilitar el movimiento en una, en dos, o en tres, de las direcciones detalladas. De esta manera, se clasifican en:

Apoyos De Primer Género, el apoyo de primer orden o comúnmente llamados apoyos simples, permiten el movimiento en dos direcciones, sobre el eje x y su rotación en el eje z , e impidiéndolo en el eje y . Tiene dos grados de libertad (de moverse).

Apoyo De Segundo Género, el apoyo de segundo orden o comúnmente llamados apoyos fijos, permite el movimiento en una dirección, alrededor del eje z e impidiendo el desplazamiento en las otras dos direcciones. Tiene un grado de libertad (de moverse).

Apoyo De Tercer Género, los de tercer orden o empotramientos, impiden el movimiento en todas las direcciones, tanto lineales en ejes x y y como el rotacional alrededor del eje z . Su grado de libertad es 0. (Trujillo, 2007).

Todo para que se cumpla la estática, que es el estado en el que las posiciones relativas de los subsistemas no varían con el tiempo. Según la primera ley de Newton implica que la fuerza neta y el par neto (también conocido como momento de fuerza) de cada organismo en el sistema es igual a cero (Jr, 1992), esto permite hablar de conceptos necesarios como lo son las condiciones de equilibrio; la primera condición de equilibrio es la red de fuerzas de igual a cero (0) y el par neto se conoce como la segunda ley de equilibrio. (Jr, 1992) Estos conceptos permiten identificar si una viga se puede analizar según su comportamiento isostático e hiperestático. Una viga isostática o estáticamente determinada, es aquella que se puede solucionar usando únicamente las ecuaciones de equilibrio de la estática: $\sum f_x=0$, $\sum f_y=0$ y $\sum M_o=0$. (Cuevas, 2003); viga hiperestática es un elemento estructural que, para su

solución, es necesario plantear además de las ecuaciones de equilibrio, ecuaciones de compatibilidad de deformaciones entre los elementos de la estructura, vigas y apoyos. (Cuevas, 2003). Estas condiciones de equilibrio se pueden analizar de manera efectiva con la aplicación del diagrama de cuerpo libre o diagrama de fuerzas, que es una representación gráfica usada a menudo por ingenieros para analizar las fuerzas que actúan sobre un cuerpo. Este diagrama permite identificar las fuerzas y momentos que deben tenerse en cuenta para la solución de un problema. (Hibbeler, 2010)

De lo anterior es importante resaltar el concepto de reacción, ya que, según la tercera ley de Newton, toda fuerza genera una reacción y se puede definir como una fuerza de sujeción de un elemento resistente al suelo o en otro elemento de grandes dimensiones que sirven de soporte al elemento resistente. En sentido general a veces se habla de empotramiento o momentos de reacción, en el caso de enlaces que además impiden el giro de algunas secciones de unión (Thornton, 1995), a estas reacciones para las cargas aplicadas se les conoce como esfuerzo cortante que se define como la resultante de las tensiones paralelas a la sección transversal de un prisma mecánico. Dicho esfuerzo está impidiendo que el objeto se deforme y así pueda tener su rigidez (Berrocal, 1990); momento flector, así definido, dadas las condiciones de equilibrio, coincide con la resultante de fuerzas situadas a uno de los dos lados de la selección en equilibrio en la que pretendemos calcular el momento flector. Debido a que un elemento puede estar sujeto a varias fuerzas, cargas distribuidas y momentos, el diagrama de momento flector varía a lo largo del mismo. Asimismo, las cargas estarán completadas y divididas por tramos de secciones. En una pieza de plano medio, si conoce el desplazamiento vertical del eje vari-céntrico sobre dicho plano de momento flector y puede calcularse a partir de la ecuación de la curva elásticas. (Berrocal, 1990)

Para el tema a tratar, comprender el método justificado en cuanto análisis estructural, para el desarrollo de vigas que contengan más de una luz, es Hardy Cross.

3.1.3 Método De Hardy Cross.

Método creado por el profesor Hardy Cross, usado para el análisis de cualquier viga indeterminada o de un pórtico rígido, que resuelve de manera sencilla y segura, estructuras que se diseñaban con métodos aproximados. Consiste en resolver ecuaciones simultaneas que son resultado de ángulos de giro y deflexión por medio de aproximaciones progresivas. (Jairo, 2002). A continuación, se describen las características que hacen funcional este método:

- Momento de empotramiento perfecto
- Rigidez (k)
- Reacción a la cortante (RV)
- Reacción al momento (RM)

Y para dar como concluido la teoría del Método de Hardy Cross, todo su proceso termina en el resultado de dos diagramas, (de fuerza cortante y momento flector) que arrojan los puntos críticos y valores máximos, parámetros clave para el diseño de una viga, y que se definen como la representación de las variaciones en la magnitud de la fuerza cortante en un elemento estructural para un determinado conjunto de cargas transversales y condiciones de apoyo y de momento flector que se definen como una función a lo largo del eje neutro del elemento donde “x” representa la longitud a lo largo de dicho eje. (Berrocal, 1990).

3.1.4 Diseño De Vigas Rectangulares A Flexión Por El Método De La Resistencia Última (Fy)

Para el diseño de vigas rectangulares a flexión, inicia desde el momento en que se somete a una o a varias cargas gravitacionales (vivas o muertas) incluyendo su propio peso. Esto hace que, desde el punto medio de la sección transversal de la viga, presente en la parte superior un recogimiento como el resultado de la compresión sobre ella y en la parte inferior presente tensión. Al ser concreto reforzado, una combinación entre el concreto y el acero, deben

integrarse de manera balanceada (diseño balanceado), si no se llega a éste balance del diseño, se puede decir que esta sub-reforzado o sobre-reforzado.

3.1.4.1 Sección Balanceada

Es cuando el acero y el concreto alcanzan simultáneamente su máximo esfuerzo de trabajo; se habla de *sección sobre-esforzada* cuando se brinda a la sección transversal una cuantía de acero mayor a la requerida, generando que el concreto llegue a su estado límite de falla, antes que el acero y cause que en la estructura, el concreto no de muestras de fallas y colapse de manera inoportuna, por el contrario una *sección sub-reforzada* es cuando a la sección transversal se le da una cuantía de acero menor a la exigida en la estructura, permitiendo que el acero falle antes que el concreto dando como resultado flexión y grietas sobre la estructura.

3.1.4.2 . Método De La Resistencia Última Para El Diseño De Vigas Rectangulares A Flexión.

Consiste en la utilización del acero longitudinal que son barras que están dispuestas a absorber las tracciones que, por flexión, no está en capacidad de absorber el concreto y debe localizarse en las zonas donde éstas se presenten. Este refuerzo también restringe el desarrollo de grietas originadas por la baja resistencia a la tracción del concreto y mejora su capacidad de deformación. (Awad, 1999); todo esto partiendo de la resistencia máxima del acero y la máxima resistencia a compresión del concreto. Por este método y para que un elemento no falle cuando se aplican las cargas evaluadas, se les debe afectar por coeficientes de mayoración de cargas y de reducción de la resistencia de los materiales. De ésta manera se obtiene un factor de seguridad apropiado (Samacá, 2014).

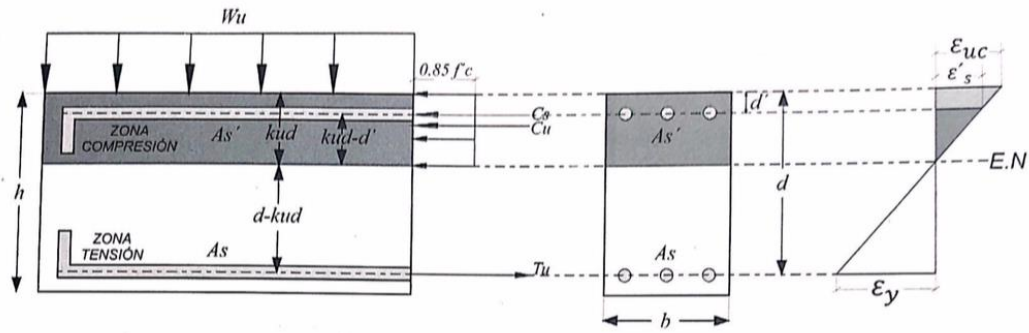


Imagen 5. Método de la resistencia última. Tomando de Diseño Básico de Vigas Vol. 1. **Fuente** (Samacá, 2014)

3.1.5 Diseño A Cortante.

Por este diseño se busca eliminar la falla directa que se puede producir a 45° por la fuerza cortante a la que se encuentra sometida longitudinalmente una viga, mediante un estribo o fleje, el cual envuelve los refuerzos longitudinales, cuya función es soportar tensiones diagonales a causa de las fuerzas cortantes y confinar el concreto proporcionándole restricción a los desplazamientos laterales y más capacidad de deformación (ductilidad). Este refuerzo cumple una función muy importante, su resistencia a la compresión es una función de confinamiento lateral. (Awad, 1999). Este tipo de esfuerzos no se debe limitar a los extremos de la viga y se deben poner a una distancia determinada en toda su longitud. Cabe notar que a lo largo de la viga no todos los flejes se encuentran a la misma longitud y que a medida que se llega al centro la separación es mayor ya que el esfuerzo cortante es menor. (Samacá, 2014). La distancia entre flejes, se determinan de acuerdo a fórmulas que se plantarán en el marco metodológico.

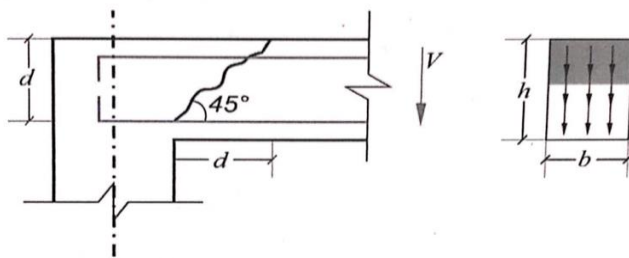


Imagen 6. Falla por cortante. Tomando de Diseño Básico de Vigas Vol. 1. **Fuente** (Samacá, 2014)

3.1.6 Asignación De Aceros En Una Viga.

El concreto al ser un material frágil de buen comportamiento a tensiones de compresión, pero muy baja resistencia a la tracción tiene limitada aplicación estructural y debe combinarse con el acero con el fin de que este material absorba las tracciones que no está en capacidad de soportar. El acero además de soportar las tracciones que el concreto no puede absorber, puede incrementar la resistencia y mejorar la ductilidad de la estructura. (Awad, 1999). Estos se disponen de dos maneras, refuerzos longitudinales y transversales.

3.2 CONCEPTOS DE PROGRAMACIÓN.

3.2.1 Desarrollo, Estructura De Códigos De Kai-Sad.

El programa KAI-SAD, es un software concebido en Visual Studio, siendo esta la plataforma de trabajo, desarrollado en lenguaje C#, C Sharp (Ci Sharp), dado, es uno de lenguajes que en el mercado se considera versátil y de alto rendimiento; y que por su antigüedad en el medio (20 años) permite encontrar en la web diferentes bibliotecas de códigos, que facilitan y agilizan el desarrollo del programa. KAI-SAD registra, analiza y presenta datos para el diseño de una viga en concreto reforzado (de acuerdo a las restricciones definidas en el documento), de hasta nueve luces y para su desarrollo, se emplearon librerías de códigos, bajo dos estructuras básicas, secuencia y selección como *if* y *switch*, que son estructuras para la toma de decisiones múltiples, e iteración con *bucles*, *for* y *while* que son estructuras para ciclos repetitivos, es una sentencia que se ejecuta en repetidas veces de código.

Se familiariza, algunos términos empleados en Visual Studio y códigos utilizados en C#; su tarea dentro de la programación a lo largo del trabajo, para hacer más entendible la descripción del desarrollo de KAI-SAD.

Para crear el proyecto, se elige el vínculo crear nuevo proyecto para abrir el cuadro de diálogo “Nuevo proyecto” en la barra de menú.

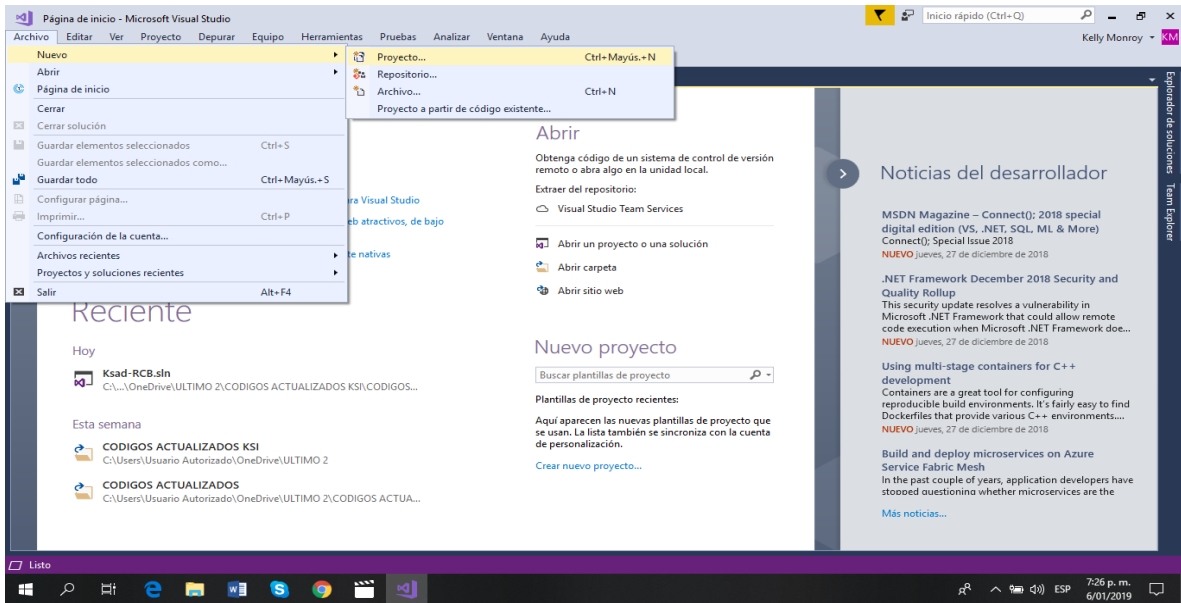


Imagen 7. Creación del proyecto en Visual Studio. Fuente propia.

Luego en el cuadro de diálogo “Nuevo proyecto” aparecen tres plantillas de proyecto, en las plantillas están organizadas por tipo de proyecto y lenguaje de programación, para este caso se selecciona la plantilla “Installed”, como tipo de proyecto se escoge Aplicación de “Windows Forms” (.NET Frameworks), como lenguaje de programación C#; y el nombre con el que se bautiza el programa, KAI-SAD. Como se muestra a continuación en las siguientes imágenes.

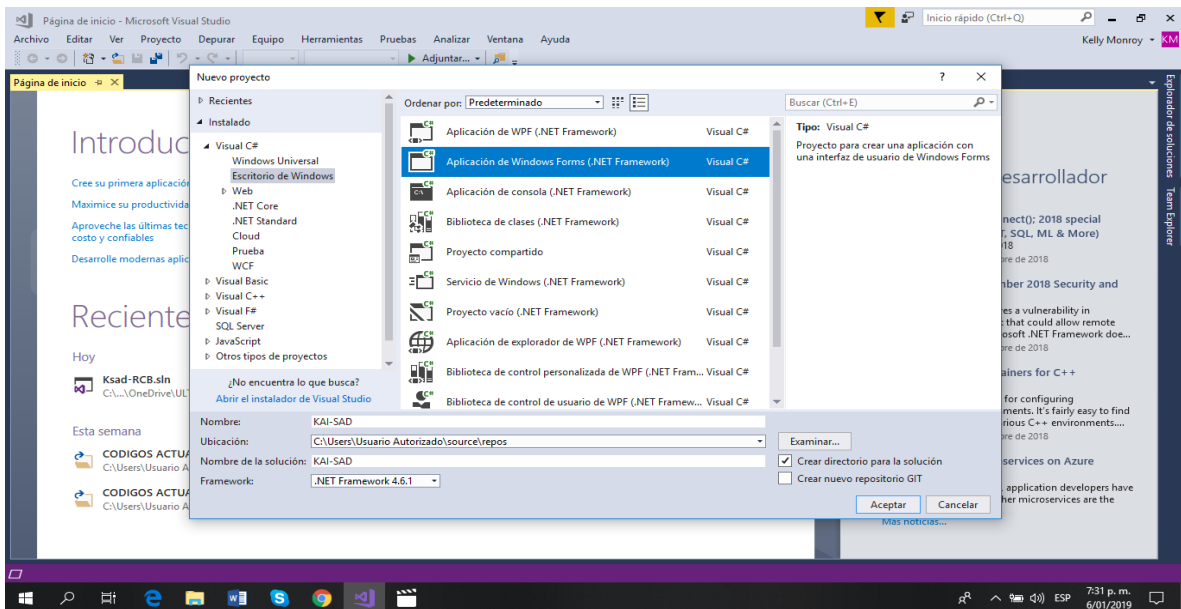


Imagen 8. Selección tipo de plataforma. Fuente Visual Studio.

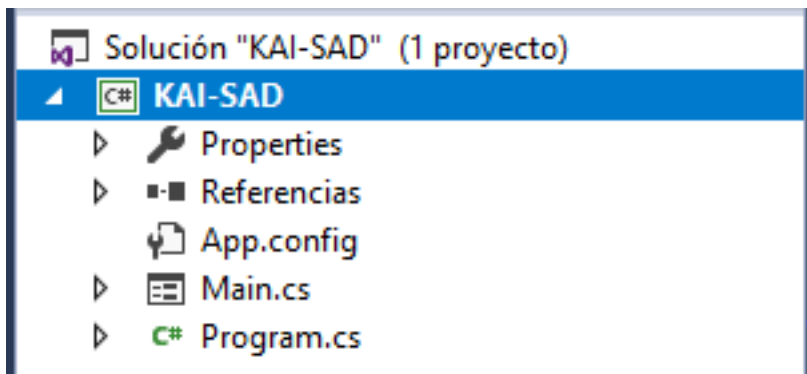


Imagen 9. Creación estructura del proyecto. **Fuente** Visual Studio.

Automáticamente se crea una estructura de proyecto que está conformada por “Properties”, “Referencias” y “Main.cs”; el siguiente código define punto de entrada del programa, con el formulario llamado Main y en él, el código “class” que por defecto Visual Studio inicia el arranque del formulario.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace KAI_SAD
{
    static class Program
    {
        /// <summary>
        /// Punto de entrada principal para la aplicación.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Main());
        }
    }
}
```

3.2.2 Creación De Variables.

Al obtener la estructura de código que define el punto de entrada del programa, se crea por una parte las variables a utilizar, y en el desarrollo del programa se crean más variables a

implementar en el software. ¿Qué es una variable?, en programación, una variable es un espacio de memoria reservado para almacenar un valor determinado que corresponde a un tipo de dato soportado por el lenguaje de programación en el cual se trabaja.

Visual Studio contiene librerías o subcódigos que se crean por defecto y otras se deben crear de acuerdo con el método a usar; “using System” significa la referencia sobre los componentes a utilizar; asimismo, “public partial class Main” es la plantilla que contiene todos los códigos creados durante la ejecución del software. Las variables utilizadas en el almacenamiento son “double” que significa la precisión de los números y se utiliza cuando la precisión de una variable flotante no es suficiente, en otras palabras la precisión de los números es dos veces más que los números “float”; las variables declaradas como tipo “double” pueden contener aproximadamente el doble de dígitos significativos; “int” denota un tipo entero que almacena valores según el tamaño y el intervalo, puede declarar e inicializar una variable mediante la asignación de un literal decimal, un literal hexadecimal o un literal binario; “string” también conocidas como cadenas de caracteres y son de tipo vector, en general, es una sucesión de letras, números, signos o símbolos, y “DataTable” representa un conjunto completo de datos, incluyendo las tablas que contiene, ordenan y restringen los datos, así como las relaciones entre las tablas.

A continuación de la creación de variables, se prosigue a la creación de casillas. Las casillas son las nueve divisiones en el menú de KAI-SAD; para su diseño se recurre a una extensión adicional a Visual Studio, su nombre es Developer Express (DevExpress), es una compañía norteamericana que se dedica al desarrollo de software y su enfoque está en el fabricar controladores de interfaz gráficos de usuario, para plataformas como Visual Studio. El contenido principal al que se recurre para darle orden al proceso en KAI-SAD, está dado por layouts, que permiten distribuir elementos dentro de un diseño, y de esta manera se dividen en “*reinicio, datos iniciales, luces y carga, calcular resultado, gráfica cortante, gráfica momento, refuerzos longitudinales, refuerzos transversales y salir*”, como lo muestra la siguiente imagen. A cada uno de ellos se le asigna un panel, que es el área de trabajo en el que se desarrolla el paso a paso de la ejecución de KAI-SAD.



Imagen 10. Recorte de pantalla del menú de KAI SAD. Fuente Propia.

3.2.3 Creación cajas de texto, cajas numéricas, vistas desplegables e imágenes.

Luego de declarar las variables, el menú y los layouts, para continuar con el algoritmo del software, se llama a la interfaz, a crear y nombrar bien sea cajas de textos, cajas numéricas, vistas desplegables e imágenes que se contemplan en cada uno de los paneles, empleando “InitializeComponent();”, *inicializar componentes*, es un método donde Visual Studio; en cuestión de ideas, se recurre al uso de “ComboBox (cbx)” para crear cuadros de texto que permiten al usuario escribir o seleccionar un elemento, cuadros con una lista de componentes desde la que, el usuario puede seleccionar un ítem; “Label” son controles que se utilizan para mostrar texto o imágenes que no puede editar el usuario y se usan para identificar los objetos en un formulario, “Row Styles” es el estilo de la fila y “Height” es el ancho o alto para crear los espacios en las tablas, así se designan cada espacio que incluye dentro del software un dato o resultado que describe cada componente que contiene un panel.

3.2.4 Gráfica

la programación de la gráfica depende del número de luces que haya, para ello, dentro de la estructura se recurre al código “SerieTipo” para poder Graficar, luego para este código se crean variables instanciadas como “Luz1, Luz2, Luz3” y así sucesivamente hasta llegar a “Luz9”, seguido de esto, para al código “SerieTipo2”, para las uniones de las variables de “SerieTipo”, se utiliza la sintaxis “chart1.ChartAreas(0).AxisX.” evitando que al Graficar muestre valores negativos en el eje “x” y usa un bucle con el que hace un recorrido dentro de las variables “SerieTipo” y “SerieTipo2” del programa en el cual si se obtiene un valor, este se refleje en el área de la gráfica. KAI-SAD reconoce dentro de las variables de cada código, los valores de momentos perfectos y fuerzas cortantes máximas para datos en “y” y para datos en “x”, luego toma las longitudes de las luces que ya creadas. La codificación para Gráficas de cargas rectangulares, toma el “Wu” y lo multiplica por la “longitud

de luz” en la que se aplica esa carga, así se ubica el primer punto en eje “y” de la Gráfica y el punto en el eje “x” es donde comienza la viga y luego las demás luces; si se aplica esta misma carga; para la carga puntual, es una línea constante en sentido en que se aplica la carga y su longitud se ubica desde el apoyo inmediatamente anterior hasta el valor asignado a la variable de longitud, ya sea “a” para puntuales excéntricas o “l/2” para puntuales céntricas.

3.3 DIAGRAMAS DE FLUJO

3.3.1 Análisis Estructural.

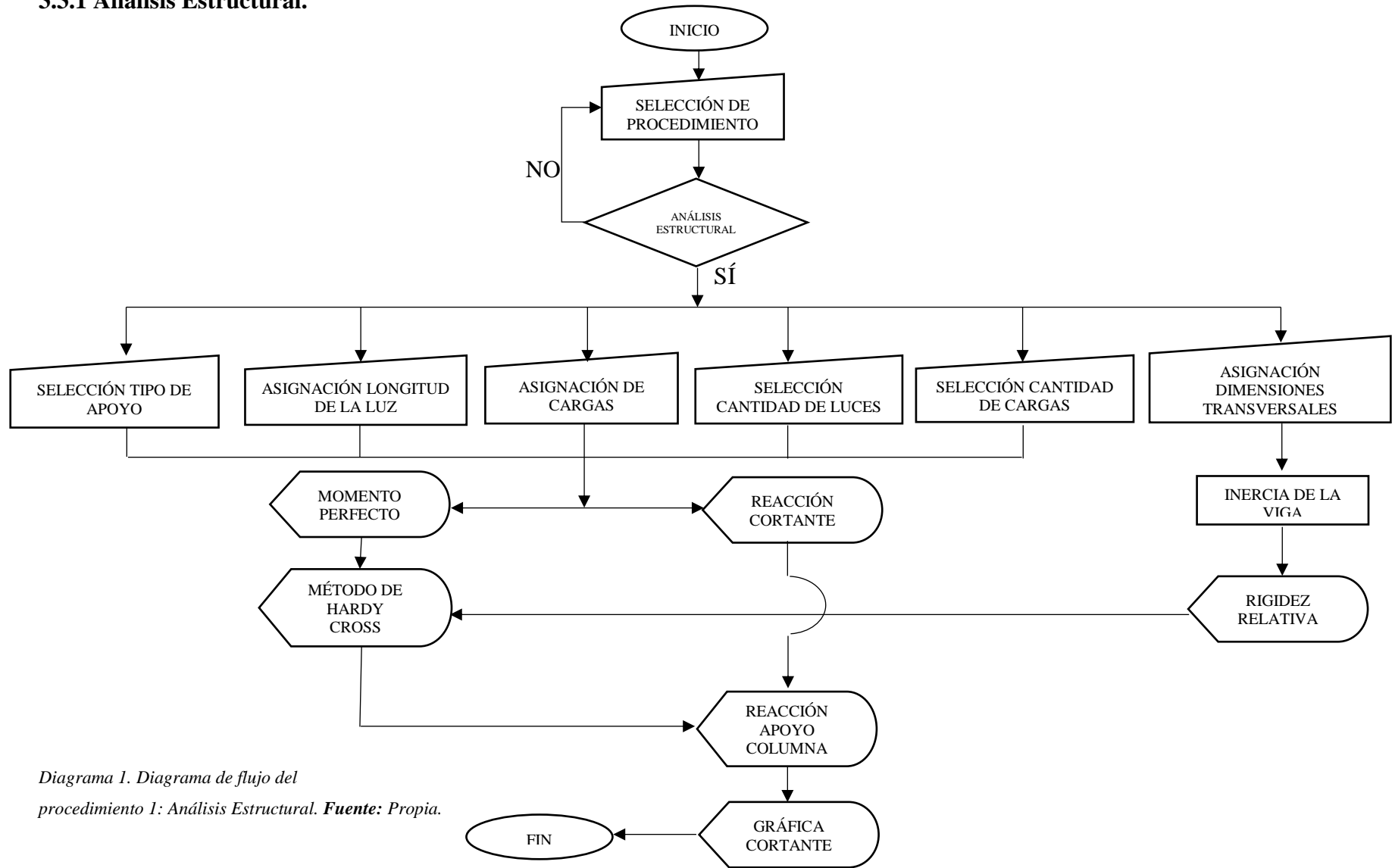


Diagrama 1. Diagrama de flujo del procedimiento 1: Análisis Estructural. Fuente: Propia.

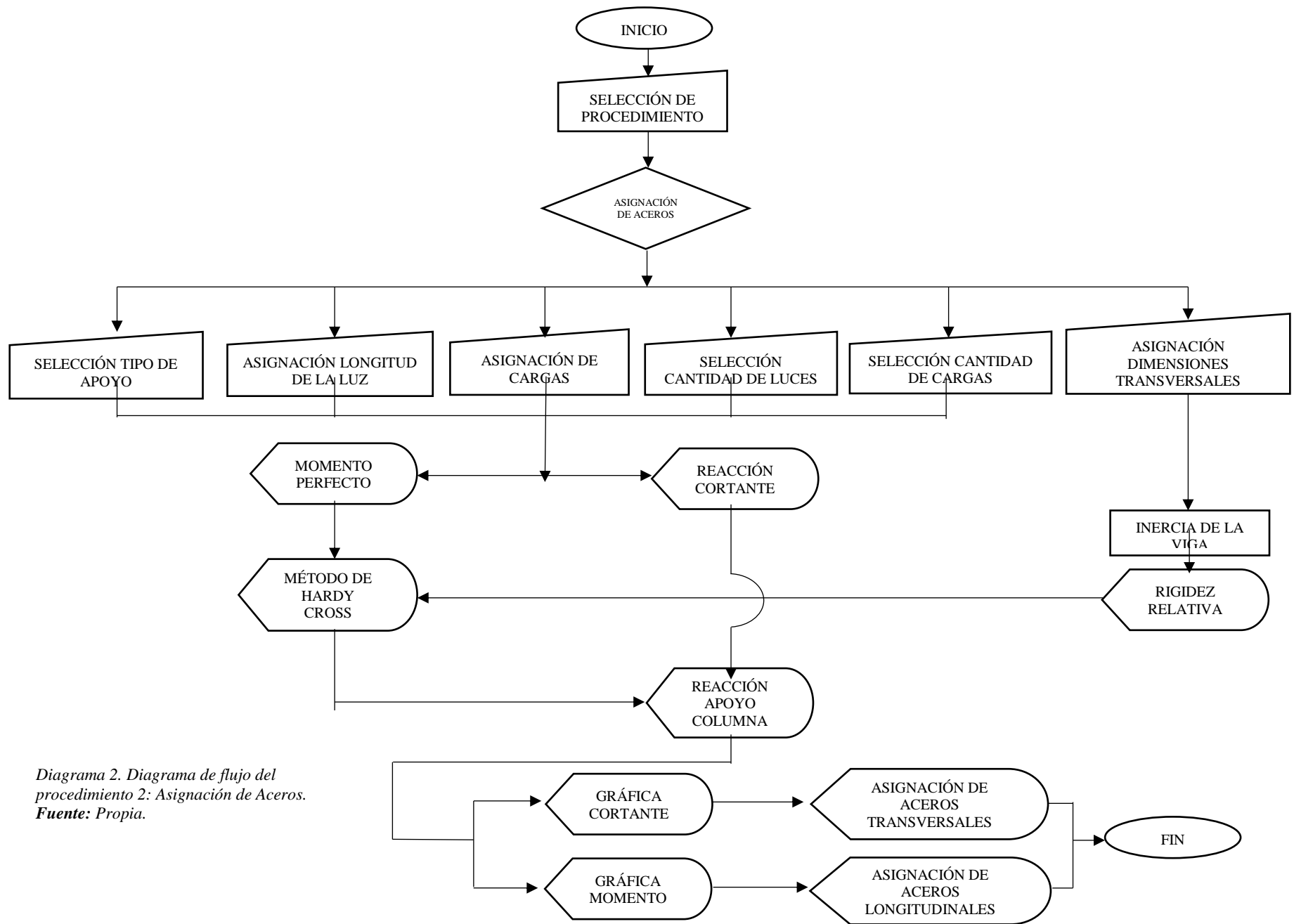


Diagrama 2. Diagrama de flujo del procedimiento 2: Asignación de Aceros.
 Fuente: Propia.

4. MARCO METODOLÓGICO

En este epígrafe, se propone exponer de manera detallada y gráfica, la metodología en que está estructurado el software, a quién se le dio el nombre de **KAI - SAD**. Esta metodología enseña al usuario cada una de las interfaces y como operar de manera correcta el programa de acuerdo a los métodos que opera. Vale resaltar que, de modo eventual, el lector encontrará una serie de notas, las cuales se recomienda darle relevancia, pues son las limitaciones que tiene el programa en su desarrollo.

En primer lugar, al abrir el programa el usuario encontrará dos pestañas que se explican a continuación:

Nota 1. La pestaña “package de menú” se encuentra deshabilitada.

4.1. PESTAÑA MENÚ

Es esta pestaña el usuario encontrará la interfaz inicial de KAI-SAD, en la imagen 10 se visualiza una sucesión de nueve botones que corresponden a cada panel en los que está distribuido el desarrollo operativo del software de la siguiente manera.

- Panel de Datos iniciales.
- Panel de Luces y cargas.
- Panel de Cálculo de resultados.
- Panel de Gráfica de Cortante.
- Panel de Gráfica de momento.
- Panel de Refuerzos a tracción y compresión.
- Panel de Diseño a cortante.

Dentro de la secuencia de botones de los paneles, al comienzo se halla un botón para el reinicio del programa y al final de ellos el botón de salir, este no cerrará el programa, sino minimizará el panel, trasladando a la vista inicial de KAI-SAD.

Cabe aclarar, KAI-SAD ejecuta dos procedimientos que son afines.

Análisis de Estructuras (Procedimiento 1) Por una parte es el primer procedimiento, da como resultado datos de Momentos de Empotramiento Perfecto, Rigidez relativa, método Hardy Cross, Reacción a la Cortante, Reacción al Momento, Reacción en el Apoyo y el Diagrama de Gráfica Cortante.

Asignación de aceros (Procedimiento 2) Por otra parte KAI-SAD ejecuta este procedimiento, que arroja como resultado posibles juegos de varillas como refuerzos longitudinales, en los que el estudiante debe tener criterio para su elección, y el número de unidades que deberá tener como refuerzos transversales la viga en estudio. Para cada método, se explica por panel que se contempla en su operación.

Nota 2. Los paneles “Gráfica de momento”, “Refuerzo longitudinal” y “Refuerzo transversal” están habilitados solo para el procedimiento “Asignación de aceros”

4.2. ANÁLISIS ESTRUCTURAL (Procedimiento 1)

4.2.1. Panel De “Datos Iniciales”

Al seleccionar el botón de datos iniciales, KAI-SAD despliega el panel de datos iniciales en el que usuario debe elegir entre los dos métodos expuestos.

A continuación, se muestra el procedimiento *Análisis Estructural*, explicado minuciosamente y a partir de capturas de pantalla, realizados al programa. No se tiene un orden para la explicación pues la interacción con el programa permite distintas maneras de utilización, seguido de entender esto, se encontrará la explicación del primer panel en el que requiere ingresar datos para el desarrollo de dicho procedimiento.

Al visualizar el panel de datos iniciales, hay un cuadro de opciones en el que, se selecciona el procedimiento *Análisis Estructural*.

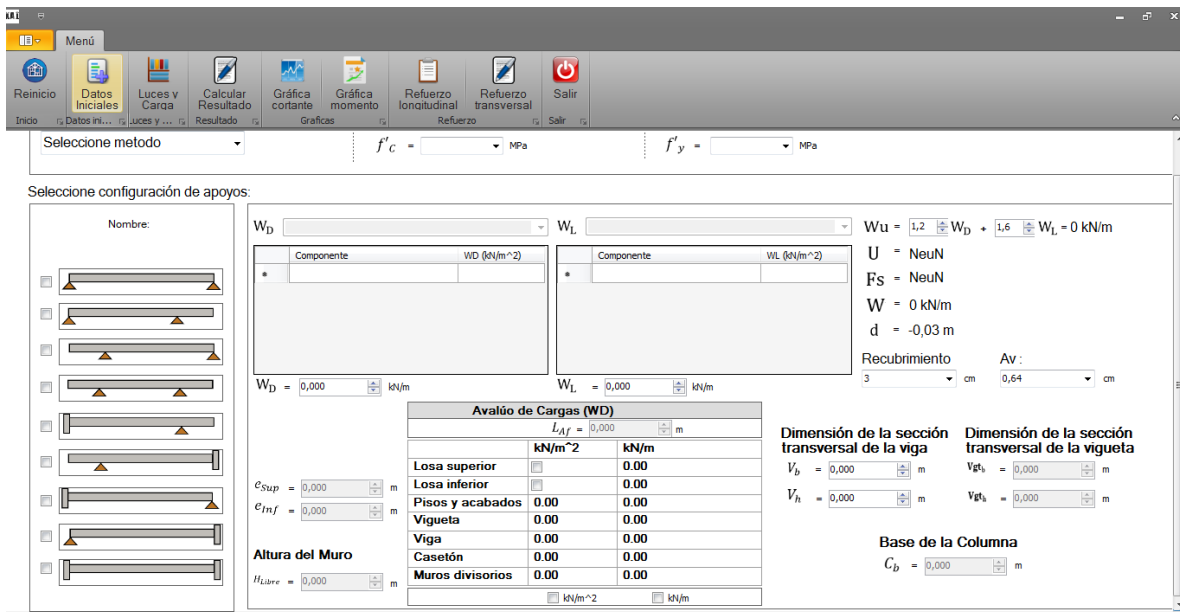


Imagen 11. Recorte de pantalla de datos iniciales del programa KAI SAD. Fuente Propia.

Seguido de esto, el usuario se encuentra con un listado de *configuración de apoyos*, que permitirá al estudiante seleccionar alguna de las siguientes configuraciones:

- Viga rectangular isostática e hiperestática simplemente apoyada.

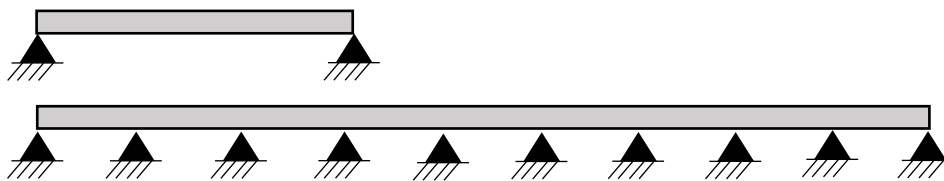


Grafico 1. Viga rectangular isostática e hiperestática simplemente apoyada. Fuente: propia.

- Viga rectangular isostática e hiperestática con inicio apoyo y final voladizo

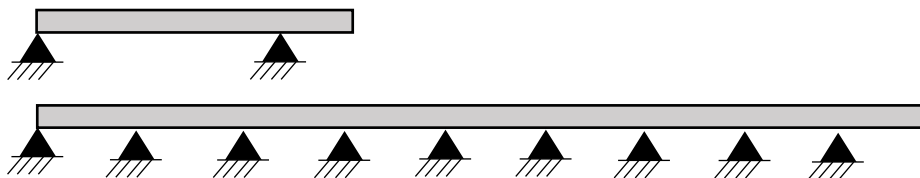


Grafico 2. Viga rectangular isostática e hiperestática con inicio apoyo y final voladizo. Fuente: propia.

- Viga rectangular isostática e hiperestática con inicio voladizo y final apoyo

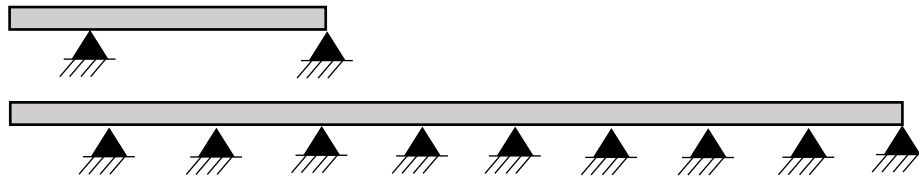


Grafico 3. Viga rectangular isostática e hiperestática con inicio voladizo y final apoyo **Fuente:** propia.

- Viga rectangular isostática e hiperestática con inicio voladizo y final voladizo

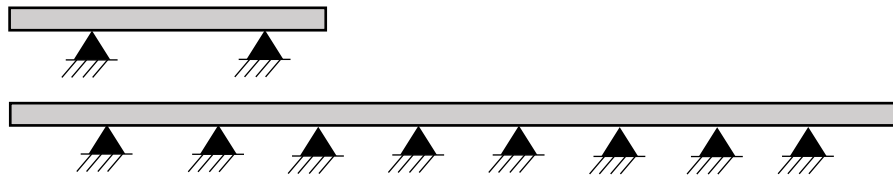


Grafico 4. Viga rectangular isostática e hiperestática con inicio voladizo y final voladizo **Fuente:** propia.

- Viga rectangular isostática e hiperestática con inicio empotrado y final voladizo

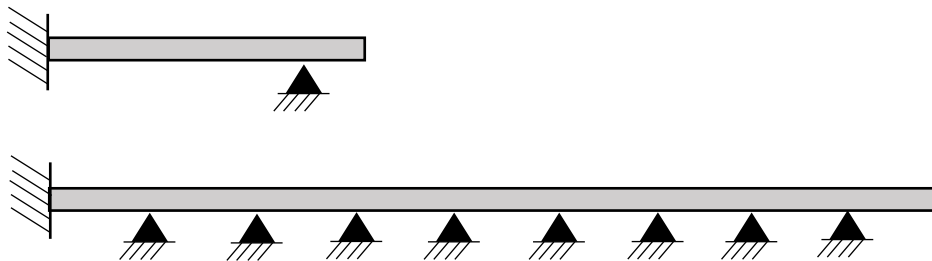


Grafico 5. Viga rectangular isostática e hiperestática con inicio empotrado y final voladizo. **Fuente:** propia

- Viga rectangular isostática e hiperestática con inicio voladizo y final empotrado

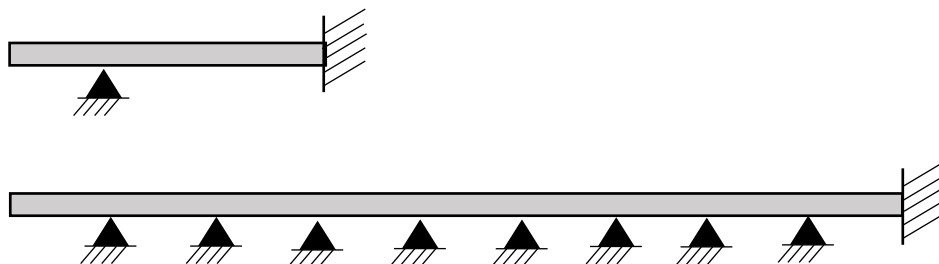


Grafico 6. Viga rectangular isostática e hiperestática con inicio voladizo y final empotrado. **Fuente:** propia

- Viga rectangular isostática e hiperestática con inicio empotrado y final apoyo

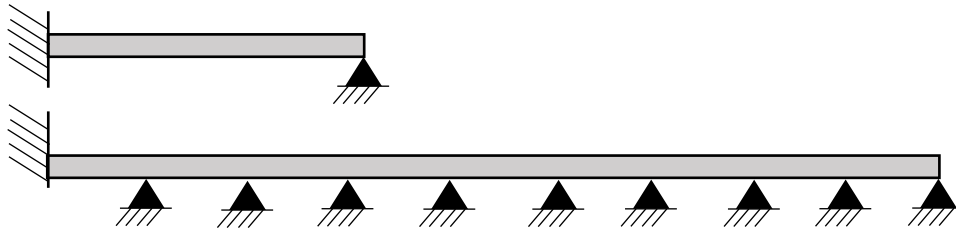


Grafico 7. Viga rectangular isostática e hiperestática con inicio empotrado y final apoyo **Fuente:** propia

- Viga rectangular isostática e hiperestática con inicio apoyo y final empotrado

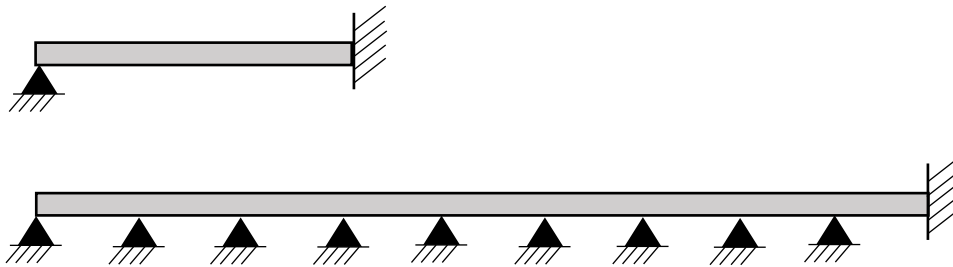


Grafico 8. Viga rectangular isostática e hiperestática con inicio apoyo y final empotrado. **Fuente:** propia

Nota 3. KAI-SAD muestra ilustraciones de una viga una sola luz, no muestra la viga completa en el caso de que tenga más de una luz.

Luego, se deben asignar valores aleatorios a las cajas de texto de la carga muerta W_D , carga viva W_L y las dimensiones de la sección transversal de la viga, base de la viga V_b y altura de la viga V_h , para conseguir el valor deseado de la magnitud de la carga que se verá reflejado en la caja de texto de la carga última W_U .

Nota 4. Para *Análisis Estructural*, se habilitan los campos que se requieren para el desarrollo de este método, en este panel. Los campos deshabilitados en este panel, para este procedimiento son:

- Avalúo de cargas.
- Recubrimiento.
- Área del acero a cortante (A_v).

- Base de la columna.
- Dimensión sección transversal de la vigueta.
- f_c
- f_y

Nota 5. En este procedimiento se encuentra habilitado la caja de texto de la carga última W_U , solo para mostrar visualmente al usuario, el valor de la magnitud que requiere aplicar en la viga de acuerdo al ejercicio que esté planteando en KAI-SAD.

4.2.2. Panel De “Cargas Y Luces”

En este segundo panel, como primera medida el usuario debe seleccionar el número de la cantidad de luces, KAI-SAD permite la asignación de hasta nueve luces en la viga. Al oprimir el triángulo de “*Selección cantidad luces*”, se desplegará una lista de números para asignar el número de luces a la viga, luego de haber contemplado las configuraciones de apoyo ya nombradas.

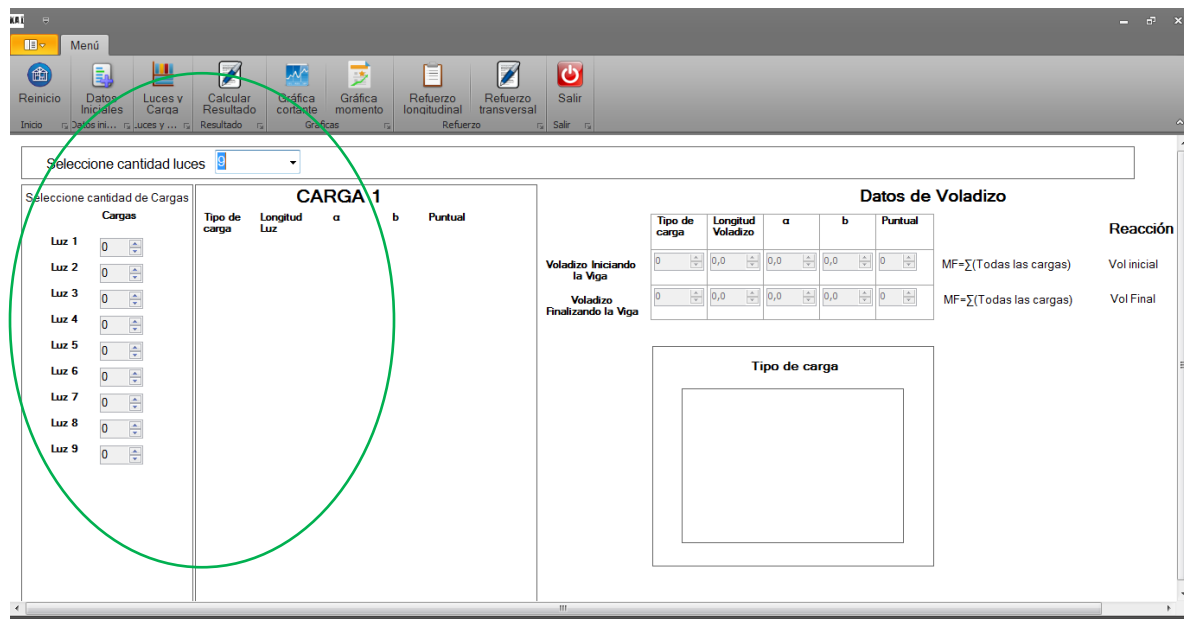


Imagen 12. Recorte de pantalla del listado de 9 luces KAI SAD. **Fuente:** Propia.

Luego el usuario, debe seleccionar la cantidad de cargas que desea aplicar en cada luz de la viga en “*Selección de cantidad de carga*”. Por cada luz KAI-SAD dará la posibilidad de

aplicar hasta seis cargas, bien sea, un solo tipo o combinadas entre ellas de la siguiente manera:

- Tipo 1: Única carga uniformemente distribuida.

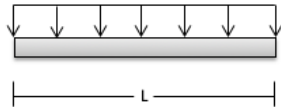


Imagen 13. Carga uniformemente distribuida Fuente. KAI-SAD

- Tipo 2: Única carga triangular descendiente.

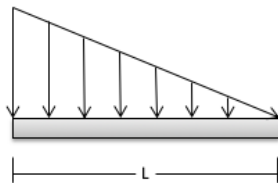


Imagen 14. Carga triangular descendiente. Fuente. KAI-SAD

- Tipo 3: Única carga triangular ascendente.

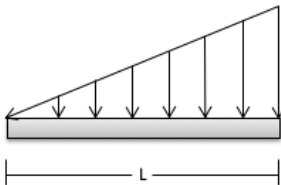


Imagen 15. Carga triangular ascendente. Fuente. KAI-SAD

- Tipo 4: Única puntual céntrica

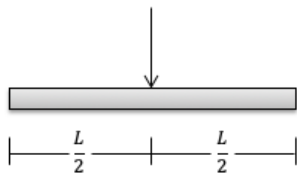


Imagen 16. Carga puntual céntrica. Fuente. KAI-SAD

- Tipo 5: Única puntual excéntrica.

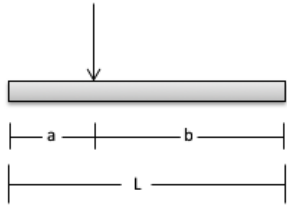


Imagen 17. Carga puntual excéntrica. Fuente. KAI-SAD

Posteriormente debe asignar la longitud de la luz y la magnitud de la fuerza si llegase a ser puntual “P”, tipo 4 o tipo 5. Al seleccionar carga tipo 5, KAI-SAD habilita dos cajas de texto para ingresar dimensiones de la distancia desde apoyo inicial hasta donde se aplique la fuerza “a”, y desde donde se aplica la fuerza hasta el apoyo final “b”.

Luego de entender los tipos con referencia a su carga se muestran las combinaciones cuando se selecciona más de una carga, el programa solo permitirá lo siguiente:

- Carga uniformemente distribuida más cinco puntuales excéntricas,
- Carga uniformemente distribuida más una puntual céntrica.
- Seis puntuales excéntricas.

Datos de voladizo.

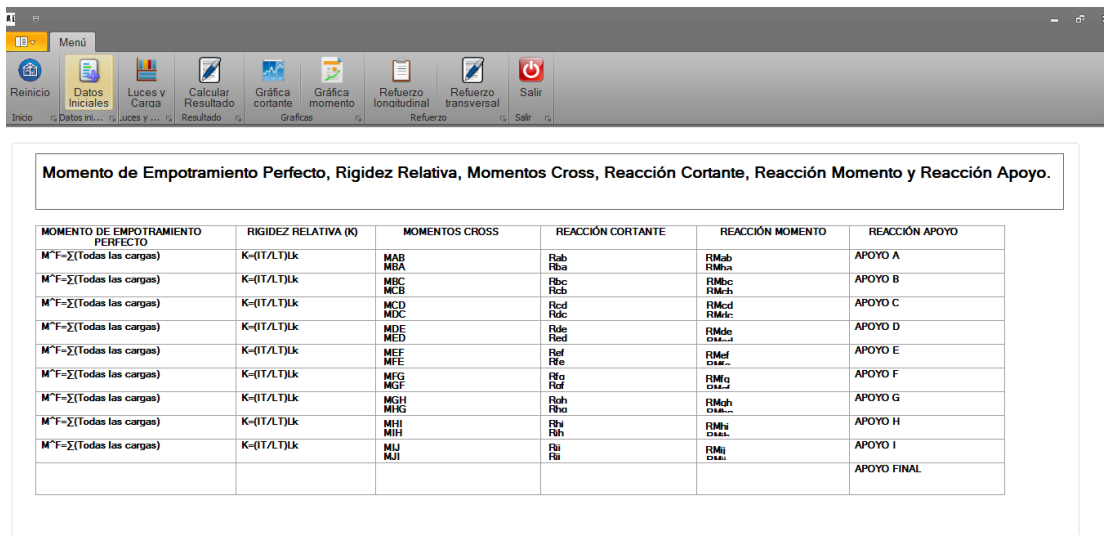
Finalmente, en los voladizos inicial o final, KAI-SAD da la posibilidad de que el usuario aplique solamente, una de los cinco tipos de cargas nombradas anteriormente.

Posteriormente debe asignar la longitud del voladizo y la magnitud de la fuerza si llegase a ser puntual “P”, tipo 4 ó tipo 5. Al seleccionar carga tipo 5, KAI-SAD habilita dos cajas de texto para ingresar dimensiones de la distancia desde apoyo inicial hasta donde se aplique la fuerza “a”, y desde donde se aplica la fuerza hasta el apoyo final “b”.

Nota 6. KAI-SAD no permite la combinación de carga triangular, con ningún otro tipo de carga.

4.2.3 Panel De “Calcular Resultado”

Al seleccionar este tercer panel KAI-SAD automáticamente arrojará una tabla de datos por cada luz, con los resultados de los siguientes conceptos: “*momento de empotramiento perfecto, rigidez relativa, momento Cross, reacción cortante, reacción momento y reacción apoyo.*”



MOMENTO DE EMPOTRAMIENTO PERFECTO	RIGIDEZ RELATIVA (K)	MOMENTOS CROSS	REACCIÓN CORTANTE	REACCIÓN MOMENTO	REACCIÓN APOYO
$M^*F-\Sigma$ (Todas las cargas)	$K-(l/L)lk$	MAB MBA	Rab Rba	RMab RMba	APOYO A
$M^*F-\Sigma$ (Todas las cargas)	$K-(l/L)lk$	MBC MCB	Rbc Rcb	RMbc RMcb	APOYO B
$M^*F-\Sigma$ (Todas las cargas)	$K-(l/L)lk$	MCD MDC	Rcd Rdc	RMcd RMdc	APOYO C
$M^*F-\Sigma$ (Todas las cargas)	$K-(l/L)lk$	MDE MED	Rde Red	RMde RMed	APOYO D
$M^*F-\Sigma$ (Todas las cargas)	$K-(l/L)lk$	MEF MEE	Rfe Rfe	RMef RMef	APOYO E
$M^*F-\Sigma$ (Todas las cargas)	$K-(l/L)lk$	MFG MGF	Rfg Rfg	RMfg RMfg	APOYO F
$M^*F-\Sigma$ (Todas las cargas)	$K-(l/L)lk$	MGH MHG	Rgh Rho	RMgh RMgh	APOYO G
$M^*F-\Sigma$ (Todas las cargas)	$K-(l/L)lk$	MHI MIH	Rhi Rih	RMhi RMhi	APOYO H
$M^*F-\Sigma$ (Todas las cargas)	$K-(l/L)lk$	MJI MIJ	Rji Rji	RMji RMji	APOYO I
					APOYO FINAL

Imagen 18. Recorte de pantalla del panel de calcular resultados de KAI SAD. Fuente: Propia.

Para saber los resultados de momento empotramiento perfecto y reacción en el apoyo generado por los voladizos inicial o final, el usuario debe remitirse al panel de “*Luces y Cargas*”, puesto que, sus resultados se encuentran allí en la parte superior al lado de la tabla de “*datos de voladizo*”.

4.2.4. PANEL DE “GRÁFICA CORTANTE”

En este panel se visualiza el diagrama de fuerza cortante, de los puntos de coordenadas obtenidos por los resultados anteriores en panel de “*calcular resultado*”.

Nota 7. El diagrama de gráfico de momento en el procedimiento de Análisis estructural, está deshabilitada hasta implementar la segunda versión de KAI-SAD.

Y en este orden de ideas, finaliza al procedimiento de *Análisis estructural* dando apertura al procedimiento de *Asignación de acero* donde paso a paso se explica sus parámetros.

4.3.ASIGNACIÓN DE ACEROS (Procedimiento 2)

4.3.1. Panel De “Datos Iniciales”

En este procedimiento, como primera medida es necesario aclarar que, para el desarrollo del mismo, KAI-SAD permite trabajar únicamente con carga uniformemente distribuida. Al igual que en el procedimiento de Análisis Estructural, serán habilitadas todas las cajas de texto que el usuario debe gestionar para su operación.

Al seleccionar este segundo procedimiento en el software, hay que asignar el tipo de configuración de apoyo estipulados en el literal 4.2.1., adicional a ello el usuario debe determinar el valor de referencia de la resistencia a la compresión del concreto “ $f'c$ ” y el valor de referencia de la fluencia del acero “ $f'y$ ”.

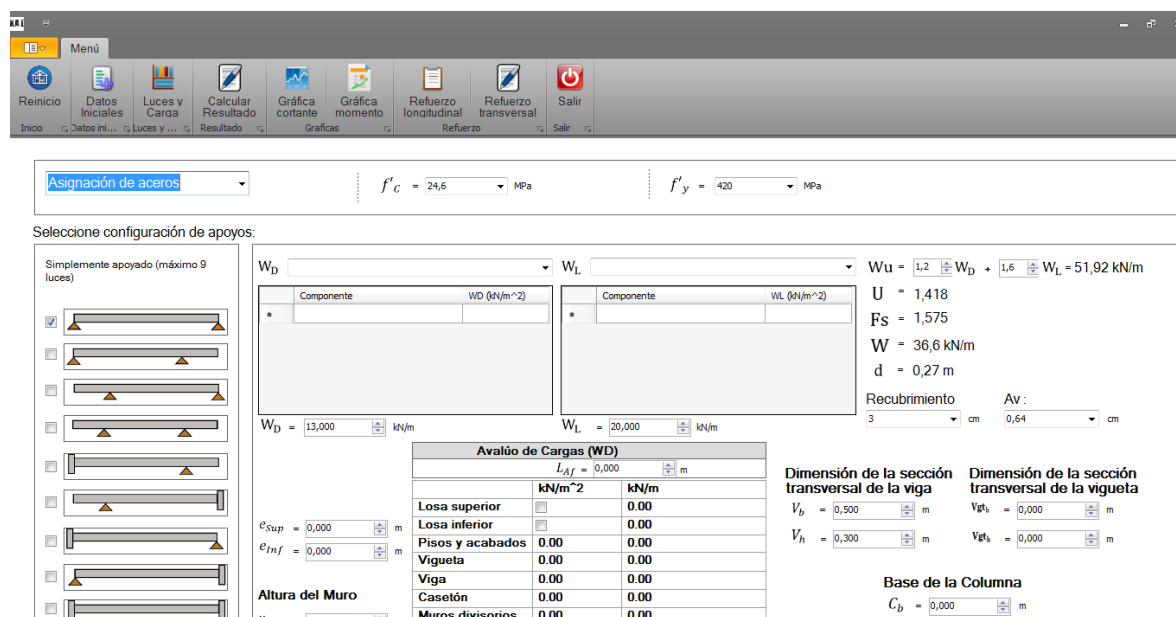


Imagen 19. Recorte de pantalla del panel de asignación de acero de KAI SAD. Fuente: Propia.

4.3.1.1. Configuración Carga muerta: para la carga muerta se contempla dos opciones de manipulación, la primera asignando el valor en la caja de texto “ W_d ”, y la segunda es seleccionando un componente de carga definidos en las tablas de cargas muertas mínimas de elementos no estructurales horizontales por la NSR-10.

The screenshot shows a software interface for configuring dead loads. At the top, there are two dropdown menus for W_D and W_L . Below them are two empty tables with columns for 'Componente' and 'WD (kN/m²)' or 'WL (kN/m²)'. Below these tables are input fields for $W_D = 13,000$ kN/m and $W_L = 20,000$ kN/m. In the center is a table titled 'Avalúo de Cargas (WD)' with a sub-table for 'Avalúo de Cargas (WD)' and a parameter $L_{Af} = 0,000$ m. The table has columns for 'Componente', 'kN/m²', and 'kN/m'. To the right of the table are input fields for $e_{Sup} = 0,000$ m, $e_{Inf} = 0,000$ m, and 'Altura del Muro' $H_{Libre} = 0,000$ m. On the far right, there are input fields for 'Dimensión de transversal d' with $V_b = 0,500$ and $V_h = 0,300$.

Componente	WD (kN/m ²)	WL (kN/m ²)
*		

$W_D = 13,000$ kN/m $W_L = 20,000$ kN/m

Avalúo de Cargas (WD)		
$L_{Af} = 0,000$ m		
	kN/m ²	kN/m
Losa superior	<input type="checkbox"/>	0.00
Losa inferior	<input type="checkbox"/>	0.00
Pisos y acabados	0.00	0.00
Vigueta	0.00	0.00
Viga	0.00	0.00
Casetón	0.00	0.00
Muros divisorios	0.00	0.00

$e_{Sup} = 0,000$ m $e_{Inf} = 0,000$ m

Altura del Muro
 $H_{Libre} = 0,000$ m

Dimensión de transversal d
 $V_b = 0,500$
 $V_h = 0,300$

Imagen 20. Recorte de pantalla de los cuadros de la carga muerta de KAI SAD. Fuente: Propia.

La segunda opción de configuración de carga muerta, se debe escoger entre las componentes recopiladas de la NSR-10 contemplados en las tablas B.3.4.1-2 “Cargas muertas mínimas de elementos no estructurales horizontales - relleno de pisos”, tabla B.3.4.1-3 “Cargas muertas mínimas de elementos no estructurales horizontales pisos” y tabla B.3.4.2-4 “Cargas muertas mínimas de elementos no estructurales horizontales - muros”. Al seleccionar alguno de estos tipos se despliega una lista a detalle, con cada componente específico.

Nota 8. No se debe, dar más de un clic en cada tipo, dado que el programa asimila cada clic como un valor a sumar.

Una vez seleccionado el tipo, se verá reflejado en panel de “avalúo de cargas”, luego se debe asignar un valor a la longitud aferente. Si se opta por tener en cuenta el peso, bien sea

la losa superior o inferior, se debe seleccionar su opción dentro del panel de “*avalúo de cargas*” y posteriormente estipular el espesor de la losa seleccionada. Asimismo, dar valor a la altura de muro “ H_{Libre} ”, la selección de unidades en que el usuario requiere ya sea “ kN/m^2 ” o “ kN/m ” y por último la dimensión de la base de la columna “ C_b ”.

Nota 9. Según (Mora, 2019), de acuerdo al avalúo de cargas contempla dos métodos. El procedimiento a, no contempla la longitud aferente y sus unidades serán en kN/m^2 ; y el procedimiento b sí contempla la longitud aferente y sus unidades son en kN/m para efectos de este programa.

En la mayoración de carga, se permite al usuario determinar sus coeficientes de acuerdo a la configuración que necesite y estipuladas en la NSR-10.

4.3.1.2. Configuración carga viva: esta al igual que la configuración de la carga muerta se puede asignar de manera directa, ingresando un valor a la caja de texto “ W_L ” o seleccionando un componente según su uso. Esto contemplado en la tabla B.4.2.1-1 “Cargas vivas mínimas uniformemente distribuidas” de la NSR-10.

4.3.2. Panel De “Cargas Y Luces”

En este panel, el usuario debe seleccionar el número de la cantidad de luces, KAI-SAD permite al igual que en el anterior método, la asignación de hasta nueve luces en la viga. Al oprimir el triángulo de “*Selección cantidad luces*”, se desplegará una lista de números para asignar el número de luces a la viga, luego de haber contemplado las configuraciones de apoyo ya nombradas.

Seleccione cantidad luces

Seleccione cantidad de Cargas		CARGA 1				
	Cargas	Tipo de carga	Longitud Luz	α	b	Puntual
Luz 1	<input style="width: 30px;" type="text" value="0"/>					
Luz 2	<input style="width: 30px;" type="text" value="0"/>					
Luz 3	<input style="width: 30px;" type="text" value="0"/>					
Luz 4	<input style="width: 30px;" type="text" value="0"/>					
Luz 5	<input style="width: 30px;" type="text" value="0"/>					

Imagen 21. Recorte de pantalla de listado 9 luces de KAI SAD. Fuente: Propia.

Cabe resaltar que para cada luz y también para voladizos, KAI-SAD da la posibilidad de aplicar un solo tipo de carga, siendo esta de tipo 1. Posteriormente debe asignar la longitud de la luz.

4.3.3. Panel De “Calcular Resultado”

Al seleccionar este tercer panel KAI-SAD automáticamente arrojará una tabla de datos por cada luz, con los resultados de los siguientes conceptos: *“momento de empotramiento perfecto, rigidez relativa, momento Cross, reacción cortante, reacción momento y reacción apoyo.”*

Para saber los resultados de momento empotramiento perfecto y reacción en el apoyo generado por los voladizos inicial o final, el usuario debe remitirse al panel de *“Luces y Cargas”*, puesto que, sus resultados se encuentran allí en la parte superior al lado de la tabla de *“datos de voladizo”*.

4.3.4. Panel De “Gráfica Cortante”

En este panel se visualiza el diagrama de fuerza cortante, de los puntos de coordenadas obtenidos por los resultados anteriores en panel de “*calcular resultado*”.

4.3.5. Panel De “Gráfica De Momento”

En este panel se visualiza el diagrama de momento flector, con base a las áreas de las derivadas de las cargas, traducidas como fuerzas cortantes.

4.3.6. Panel De “Refuerzos Longitudinales”

En este panel, se ven reflejados las cuantías de refuerzos longitudinales tanto como para tracción y compresión, de acuerdo al Método de la Resistencia Última, en el que arroja resultados de “*Momento ultimo*”, “ *ρ balanceado*”, “*área de acero a tensión (As)*” y opciones de juego de varillas a aplicar como refuerzo longitudinal en la viga.

4.3.7. Panel De “Refuerzos Transversales”

En este panel KAI-SAD, muestra al usuario las unidades de estribos que requiere la viga en estudio, para cada longitud de desarrollo máxima, medio y baja.

5. CÁLCULOS

Este capítulo muestra los cálculos pertenecientes a cada uno de métodos ejecutados por KAI-SAD, explicado a partir de los siguientes ejercicios, que cuentan con las formulaciones en sus procedimientos.

5.1. EJERCICIO DE APLICACIÓN 1, ANÁLISIS ESTRUCTURAL.

Calcular los diagramas de fuerza cortante de la viga hiperestática 1.

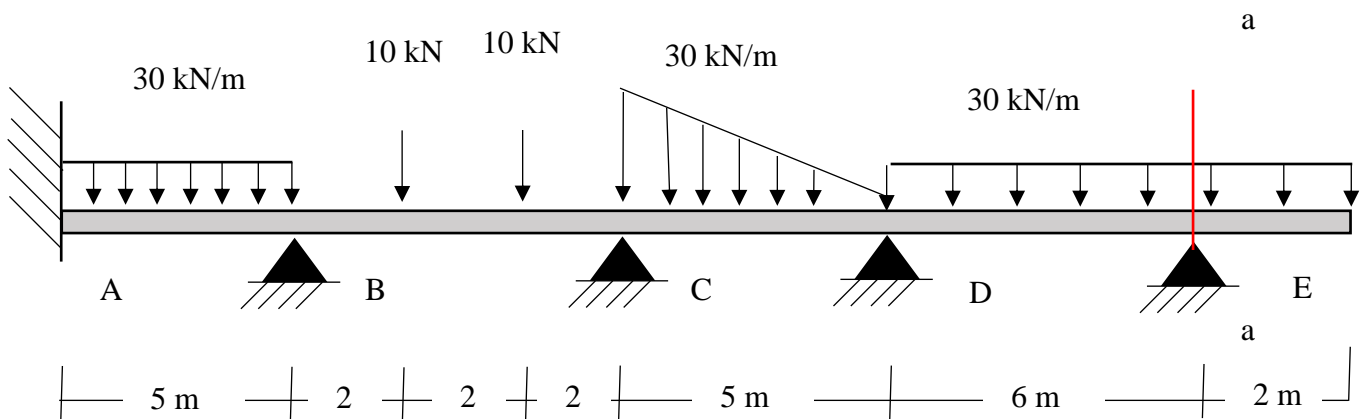


Grafico 9. Diagrama de carga viga hiperestática 1. Fuente: propia.

5.1.1. Cálculo Del Momento De Empotramiento Perfecto.

Como primera medida, se procede al cálculo de momentos de empotramiento perfecto, que se determinan a partir de una serie de ecuaciones que dependen del tipo de carga que se esté aplicando a la viga. Se definen de la siguiente manera:

Donde: M_f es momento de empotramiento perfecto.

w es el valor de la carga.

l es la longitud de la luz.

la Longitud desde el apoyo hasta la carga puntual

lb Longitud desde la carga puntual, hasta el apoyo

- Tipo 1: Única carga uniformemente distribuida.

$$Mf = \frac{wl^2}{12}$$

- Tipo 2: Única carga triangular descendiente.

$$Mf = \frac{wl^2}{20}$$

$$Mf = -\left(\frac{wl^2}{30}\right)$$

- Tipo 3: Única carga triangular ascendente.

$$Mf = \frac{wl^2}{30}$$

$$Mf = -\left(\frac{wl^2}{20}\right)$$

- Tipo 4: Única puntual céntrica

$$Mf = \frac{wl}{8}$$

- Tipo 5: Única puntual excéntrica.

$$Mf = \frac{w * la * lb^2}{lt^2}$$

Asignando esta serie de ecuaciones al ejercicio de aplicación 1, los momentos de empotramientos perfectos se calculan así:

Desde el apoyo A al apoyo B se aplica una carga tipo 1, de manera que.

$$Mf_{AB} \approx \frac{30 * 5^2}{12} = 62.5 \text{ kN} * \text{m}$$

$$Mf_{BA} \approx -\frac{30 * 5^2}{12} = -62.5 \text{ kN} * \text{m}$$

Luz desde el nodo B al nodo C, se aplican dos cargas tipo 5. En este caso, al haber dos cargas puntuales excéntricas dentro de la misma luz, se deben sumar como muestra el ejemplo a continuación.

$$Mf_{BC} \approx \frac{10*2*4^2}{6^2} + \frac{10*4*2^2}{6^2} = 13.33 \text{ kN} * \text{m}$$

$$Mf_{CB} \approx -\frac{10*2^2*4}{6^2} - \frac{10*4^2*2}{6^2} = -13.33 \text{ kN} * \text{m}$$

En la luz desde el apoyo C al apoyo D se aplica una carga tipo 2, que arroja como momentos de empotramiento perfecto:

$$Mf_{CD} \approx \frac{30 * 5^2}{20} = 37.5 \text{ kN} * \text{m}$$

$$Mf_{DC} \approx -\left(\frac{30 * 5^2}{30}\right) = -25 \text{ kN} * \text{m}$$

En la luz desde el apoyo D al apoyo E, se encuentra una carga uniformemente distribuida y se desarrolla de la siguiente manera:

$$Mf_{DE} \approx \frac{30 \cdot 6^2}{12} = 90 \text{ kN} \cdot \text{m}$$

$$Mf_{ED} \approx -\frac{30 \cdot 6^2}{12} = -90 \text{ kN} \cdot \text{m}$$

Para hallar el momento directo en el voladizo, se realiza el corte “a-a” y se hace la segunda ecuación de equilibrio, sumatoria de momentos desde el nodo E y desarrolla de la siguiente manera:

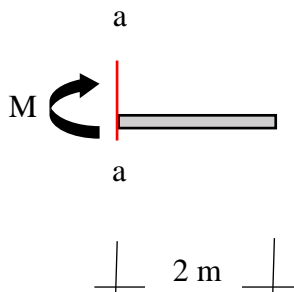


Grafico 10. Diagrama de corte a-a en voladizo de viga hiperestática 1. **Fuente:** propia.

$$\sum M \approx M_{Vol} - (30 \cdot 2) = 0$$

$$M_{Vol} = 60 \text{ kN} \cdot \text{m}$$

5.1.2. Cálculo De Momentos Por El Método De Hardy Cross.

Luego de dar solución a los momentos de empotramiento perfectos, se procede al cálculo del momento por el método de Hardy Cross, para él, se necesita hallar el valor de la rigidez relativa (K) de cada apoyo de la viga en estudio y su factor de distribución (FD).

5.1.2.1. Cálculo de la rigidez relativa.

Para la rigidez relativa (K) se debe conocer el valor de la inercia de la viga, en la que se contemplan las dimensiones de la sección transversal de la misma y el mínimo común múltiplo entre los valores de las dimensiones de cada luz del elemento. Se opera de la siguiente manera.

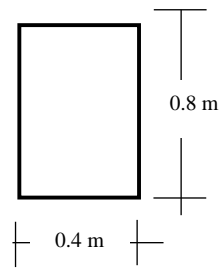


Grafico 11. Diagrama sección transversal de viga hiperestática I. **Fuente:** propia.

- Cálculo de la Inercia de la viga

$$I_v \approx \frac{1}{12} * b * h^3$$

$$I_v \approx \frac{1}{12} * 0.4 * 0.8^3 = 0.01706 \text{ m}^4$$

- Cálculo del mínimo común múltiplo entre dimensiones de las luces de la viga.

$$\text{M.C.M. (5,6,5,6)} = 30$$

- Cálculo de la rigidez relativa (k)

$$K \approx \frac{I_v}{\text{Longitud Luz.}} \text{ M.C.M}$$

$$K_{AB} \approx \frac{0.01706}{5} * 30 = 0.102$$

$$K_{BC} \approx \frac{0.01706}{6} * 30 = 0.085$$

$$K_{CD} \approx \frac{0.01706}{5} * 30 = 0.102$$

$$K_{DE} \approx \frac{0.01706}{6} * 30 = 0.085$$

5.1.2.2. Cálculo Factor De Distribución

El factor de distribución depende de los valores que se hayan asignado a cada K en sus apoyos.

Como ejemplo debe fijarse en el factor de distribución del nodo B, se distribuye para nodos A y C, por ende, se calcula el factor de distribución de la luz BA de la siguiente manera:

$$FD_{BA} = \frac{K_{BA}}{K_{BA} + K_{BC}}$$

Y a su vez el factor de distribución de la luz BC de la siguiente:

$$FD_{BC} = \frac{K_{BC}}{K_{BC} + K_{BA}}$$

En este caso:

$$FD_{BA} \approx \frac{0.102}{0.102 + 0.085} = 0.55$$

$$FD_{BC} \approx \frac{0.085}{0.085 + 0.102} = 0.45$$

La sumatoria de los factores de distribución del nodo B deben tener un equivalente a uno (1); en orden de ideas:

$$FD_B \approx FD_{BA} + FD_{BC} = 1$$

Reemplazando en el ejercicio de aplicación:

$$FD_B \approx 0.55 + 0.45 = 1$$

Este procedimiento se repite con cada nodo y a su vez para cada luz en el que se distribuye el factor.

Nota 10. Vale aclarar, que el valor del factor de distribución cuando la viga está empotrada es igual cero (0), cuando tiene voladizo no genera ningún valor para el factor de distribución, y cuando la viga comienza o termina simplemente apoyada su valor es igual a uno (1). KAI-SAD por defecto reconoce estos valores.

5.1.2.3. Cálculos Por El Método De Hardy Cross.

En el método iterativo de Hardy Cross se debe lograr la convergencia de valores equivalentes de momentos en cada apoyo, con signo opuesto. Ver tabla XX.

ITERACIÓN 1.

Para la primera iteración, los momentos de empotramiento perfecto se deben ubicar según corresponda a su luz.

	A		B		C		D		E	
	AB	BA	BC	CB	CD	DC	DE	ED	Voladizo	
K	0.102	0.102	0.085	0.085	0.1024	0.1024	0.085	0.085	0	
FD	0.00	0.55	0.45	0.45	0.55	0.55	0.45	1.00	0	
Iteración	62.50	-62.50	13.33	-13.33	37.50	-25.00	97.50	-97.50	60.00	
1	0.00	26.82	22.35	-10.98	-13.18	-39.55	-32.95	37.50	0.00	

Imagen 22. Recorte de pantalla mostrando la ubicación de los momentos de empotramiento perfecto. **Fuente:** propia.

En el apoyo A, para la luz AB, al comenzar empotrada, se debe multiplicar el momento de empotramiento perfecto por el factor de distribución, según el caso que corresponda de acuerdo a la *Nota 10*

Apoyo A.

$$AB = Mf_{AB} * FD_{AB}$$

Después de esto, la iteración para el apoyo B compartido en luces en BA y BC, sus momentos de empotramiento perfecto, deben ser sumados y después multiplicados por el factor de distribución según corresponda, y así sucesivamente con los demás apoyos, de la siguiente manera.

Apoyo B.

$$BA = (Mf_{BA} + Mf_{BC}) * FD_{BA}$$

$$BC = (Mf_{BA} + Mf_{BC}) * FD_{BC}$$

Apoyo C.

$$CB = (Mf_{CB} + Mf_{CD}) * FD_{CB}$$

$$CD = (Mf_{CB} + Mf_{CD}) * FD_{CD}$$

Apoyo D.

$$DC = (Mf_{DC} + Mf_{DE}) * FD_{DC}$$

$$DE = (Mf_{DC} + Mf_{DE}) * FD_{DE}$$

En el nodo E, para la luz ED, al ser el último apoyo, simplemente apoyado y como termina en voladizo, al momento de empotramiento perfecto (ED) se le debe sumar el valor del momento directo del voladizo, y esto multiplicado por el factor de distribución según, el caso que corresponda de acuerdo a la *Nota 10*

Apoyo E.

$$ED = (Mf_{ED} + M_{Vol}) * FD_{ED}$$

Nota 11. Si fuese un caso opuesto a este, en que la viga comenzara con voladizo y seguida de un apoyo simplemente apoyado, se debe operar de la misma manera. Se debe sumar el valor del momento directo del voladizo, al momento de empotramiento perfecto de la luz contigua.

Todos los resultados de las iteraciones se deben tomar con signo contrario a su solución.

Adaptando todas estas ecuaciones al ejercicio de aplicación 1, los resultados de la primera iteración son:

	A		B		C		D		E	
	AB	BA	BC	CB	CD	DC	DE	ED	Voladizo	
K	0.102	0.102	0.085	0.085	0.1024	0.1024	0.085	0.085	0	
FD	0.00	0.55	0.45	0.45	0.55	0.55	0.45	1.00	0	
Iteración	62.50	-62.50	19.33	-19.33	37.50	-25.00	97.50	-97.50	60.00	
1	0.00	26.82	22.35	-10.98	-13.18	-39.55	-32.95	37.50	0.00	
Iteración	13.41	0.00	-5.49	11.17	-19.77	-6.59	18.75	-16.48	0.00	

Imagen 23. Recorte de pantalla mostrando los resultados de la primera iteración. **Fuente:** propia

Apoyo A.

$$AB \approx 62.5 * 0 = 0$$

Apoyo B.

$$BA \approx (-62.50 + 13.33) * 0.55 = -26.82$$

$$BC \approx (-62.50 + 13.33) * 0.45 = -22.35$$

Apoyo C.

$$CB \approx (-13.33 + 37.50) * 0.45 = 10.98$$

$$CD \approx (-13.33 + 37.50) * 0.55 = 13.18$$

Apoyo D.

$$DC \approx (-25 + 97.5) * 0.55 = 39.87$$

$$DE \approx (-25 + 97.5) * 0.45 = 32.95$$

APOYO E.

$$ED \approx (-97.5 + 60) * 1 = -37.5$$

Para llevar el a cabo la siguiente iteración, se debe considerar que, a los resultados de la primera iteración, se les debe dividir en dos y ubicar, como se muestra en la siguiente imagen:

	A	B		C		D		E	
	AB	BA	BC	CB	CD	DC	DE	ED	Voladizo
K	0.102	0.102	0.085	0.085	0.1024	0.1024	0.085	0.085	0
FD	0.00	0.55	0.45	0.45	0.55	0.55	0.45	1.00	0
Iteración	62.50	-62.50	13.33	-13.33	37.50	-25.00	97.50	-97.50	60.00
1	0.00	-26.82	22.35	-10.98	-13.18	-39.55	-32.95	37.50	0.00
Iteración	13.41	0.00	-5.49	11.17	-19.77	-6.59	18.75	-16.48	0.00
2	0.00	3.00	2.50	3.91	4.69	-6.63	-5.53	16.48	0.00

Imagen 24. Recorte de pantalla mostrando como se ubica la división en dos de la primera iteración. Fuente: propia.

Y para la segunda iteración, se maneja el mismo procedimiento hasta llegar al objetivo del Cross, lograr la convergencia de los valores de los momentos de las luces compartidas de cada nodo, como se muestra en la siguiente tabla del método de Hardy Cross.

	A	B		C		D		E	
	AB	BA	BC	CB	CD	DC	DE	ED	Voladizo
K	0.102	0.102	0.085	0.085	0.1024	0.1024	0.085	0.085	0
FD	0.00	0.55	0.45	0.45	0.55	0.55	0.45	1.00	0
Iteración 1	62.50	-62.50	13.33	-13.33	37.50	-25.00	97.50	-97.50	60.00
	0.00	26.82	22.35	-10.98	-13.18	-39.55	-32.95	37.50	0.00
Iteración 2	13.41	0.00	-5.49	11.17	-19.77	-6.59	18.75	-16.48	0.00
	0.00	3.00	2.50	3.91	4.69	-6.63	-5.53	16.48	0.00
Iteración 3	1.50	0.00	1.95	1.25	-3.32	2.35	8.24	-2.76	0.00
	0.00	-1.07	-0.89	0.94	1.13	-5.77	-4.81	2.76	0.00
Iteración 4	-0.53	0.00	0.47	-0.44	-2.89	0.56	1.38	-2.41	0.00
	0.00	-0.26	-0.21	1.51	1.82	-1.06	-0.88	2.41	0.00
Iteración 5	-0.13	0.00	0.76	-0.11	-0.53	0.91	1.20	-0.44	0.00
	0.00	-0.41	-0.34	0.29	0.35	-1.15	-0.96	0.44	0.00
Iteración 6	-0.21	0.00	0.14	-0.17	-0.58	0.17	0.22	-0.48	0.00
	0.00	-0.08	-0.07	0.34	0.41	-0.22	-0.18	0.48	0.00
Iteración 7	-0.04	0.00	0.17	-0.03	-0.11	0.20	0.24	-0.09	0.00
	0.00	-0.09	-0.08	0.06	0.08	-0.24	-0.20	0.09	0.00
Iteración 8	-0.05	0.00	0.03	-0.04	-0.12	0.04	0.04	-0.10	0.00
	0.00	-0.02	-0.01	0.07	0.09	-0.05	-0.04	0.10	0.00
Iteración 9	-0.01	0.00	0.04	-0.01	-0.02	0.04	0.05	-0.02	0.00
	0.00	-0.02	-0.02	0.01	0.02	-0.05	-0.04	0.02	0.00
Iteración 10	-0.01	0.00	0.01	-0.01	-0.03	0.01	0.01	-0.02	0.00
	0.00	0.00	0.00	0.02	0.02	-0.01	-0.01	0.02	0.00
Iteración 11	0.00	0.00	0.01	0.00	0.00	0.01	0.01	0.00	0.00
	0.00	0.00	0.00	0.00	0.00	-0.01	-0.01	0.00	0.00
Iteración	0.00	0.00	0.00	0.00	-0.01	0.00	0.00	0.00	0.00

12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Iteración	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Σ	76.67	-34.14	34.14	-7.15	7.15	-75.53	75,53	-60.00	60.00

Tabla 1. Método iterativo de Hardy Cross. **Fuente:** propia.

5.1.3 Cálculo De Las Reacciones De Cada Apoyo

Luego, como siguiente parámetro, en un diagrama de cuerpo libre de la viga, se ubican las reacciones de las fuerzas internas cortantes de cada nodo (RV) y los valores de las reacciones de los momentos internos (RM), para a través del cálculo realizado a continuación, hallar las restricciones en los apoyos y de esta manera realizar el diagrama de la gráfica de fuerza cortante.

Reacción al momento interno (RM), se halla de la siguiente manera:

$$RM = \frac{M_A + M_B}{Longitud_{Luz}}$$

De esta manera:

$$RM_{AB} \approx \frac{76.67 - 34.14}{5} = 8.51$$

$$RM_{BA} \approx \frac{-76.67 + 34.14}{5} = -8.51$$

$$RM_{BC} \approx \frac{34.14 - 7.15}{6} = 4.50$$

$$RM_{CB} \approx \frac{-34.14 + 7.15}{6} = -4.50$$

$$RM_{CD} \approx \frac{7.15 - 75.53}{5} = -13.67$$

$$RM_{DC} \approx \frac{-7.15 + 75.53}{5} = 13.67$$

$$RM_{DE} \approx \frac{75.53 - 60}{6} = 2.59$$

$$RM_{ED} \approx \frac{-75.53 + 60}{6} = -2.59$$

Después de identificar los valores de RV y RM se operan entre ellos y de manera cómo lo indican las flechas rojas, para llegar a los valores de la reacción en cada apoyo de la viga.

Diagrama De Cuerpo Libre.

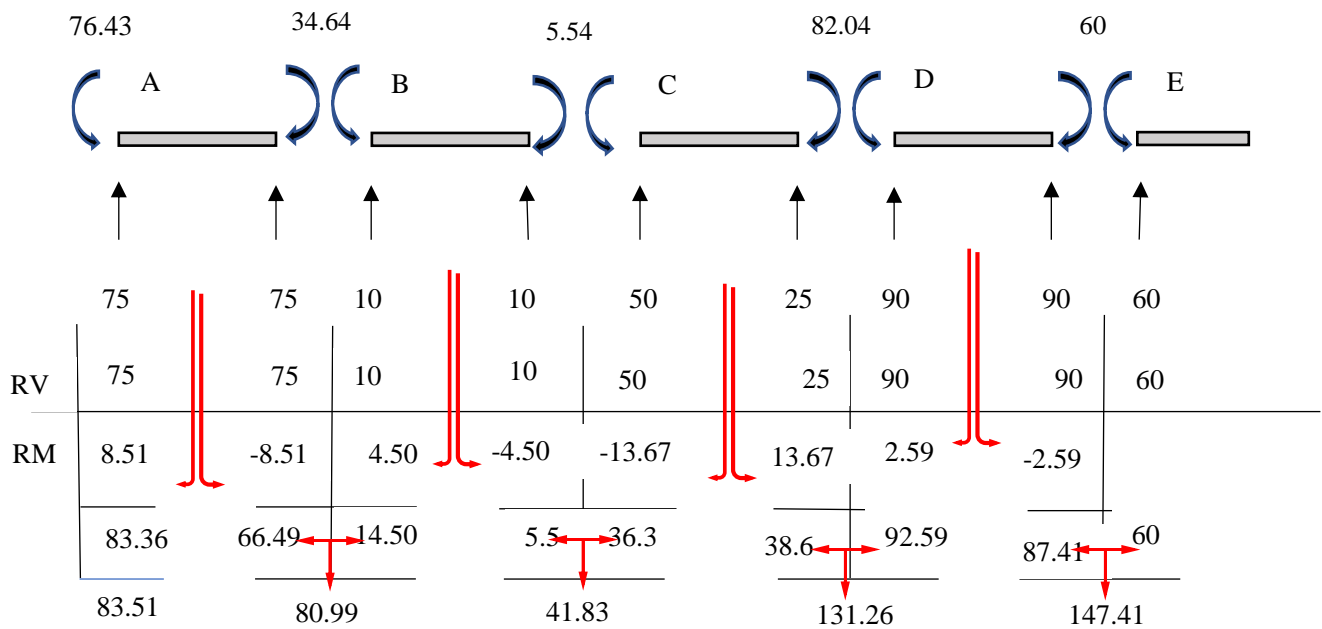


Grafico 12. Diagrama de cuerpo libre de viga hiperestática 1. Fuente: propia

5.1.4. Diagrama De Fuerza Cortante.

Y cómo último paso, con las reacciones de cada apoyo se procede a Graficar el diagrama de fuerza cortante.

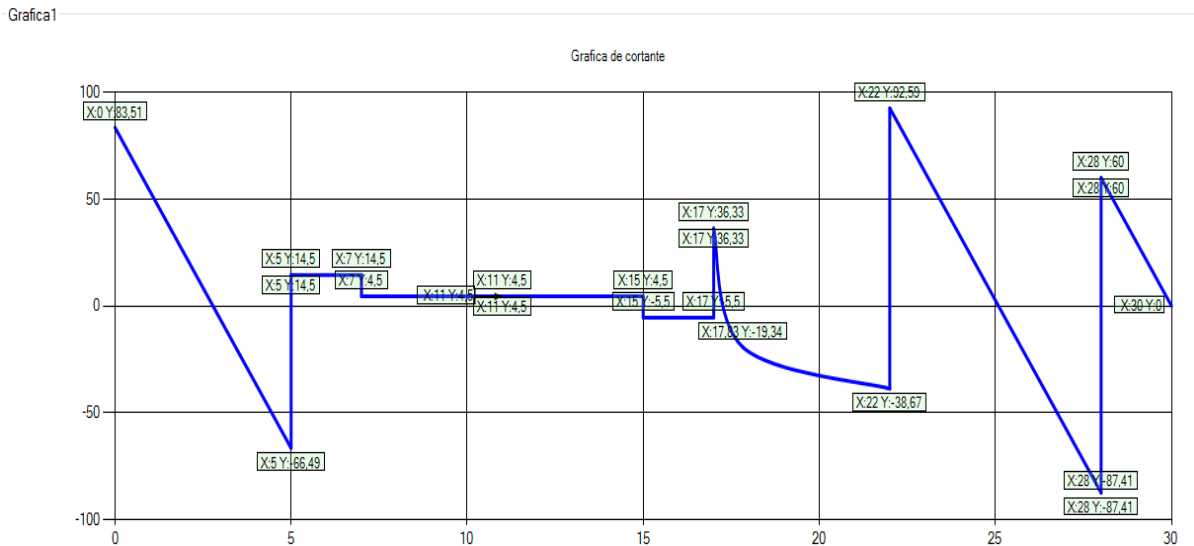


Grafico 13. Diagrama de gráfica cortante de viga hiperestática 1. Fuente: propia

5.2. EJERCICIO DE APLICACIÓN 2, ASIGNACIÓN DE ACEROS

De este punto en adelante se muestra los cálculos correspondientes del procedimiento 2 a partir del ejercicio de aplicación 2.

- Calcular los diagramas de fuerza cortante y momento flector de la viga hiperestática 2.
- Calcular las áreas de acero a compresión y tensión para la viga hiperestática 2.
- Sugerir combinaciones de aceros para la misma viga a partir del numeral b.
- Calcular la cantidad de estribos.

De acuerdo con las aclaraciones en el procedimiento 1, no se requiere para el diseño de una viga comenzar por el análisis de cargas de la estructura, pero para este procedimiento si es

necesario, por ende, se definirá el desarrollo de aplicación para asignación de acero teniendo en cuenta las siguientes variables:

Donde, f'_c : resistencia a la compresión para KAI-SAD se maneja los siguientes valores: 14.1 MPa, 17.6 MPa, 21,1 MPa, 24.6 MPa, 28.1 MPa, 31.6 MPa y 35.2 MPa acorde al Reglamento Colombiano de Construcción Sismo Resistente NSR-10 capítulo C-4. Requisitos de durabilidad.

f'_y : Resistencia a la tracción para KAI-SAD se asignaron valores típicos de 240 MPa y 420 MPa de acuerdo con el Reglamento Colombiano de Construcción Sismo Resistente NSR-10 capítulo C-7. Detalles del refuerzo.

A continuación, se instancian las variables que luego se utilizarán en los cálculos.

- Dimensiones transversales de la viga y vigueta.
 - Base (b)
 - Altura (h)
- Base de la columna (Cb)
- Espesor superior (E_{SUP})
- Espesor inferior (E_{INF})
- Recubrimiento.
- Área del acero a cortante (A_v)
- Altura libre del muro (H_{Libre})
- Longitud aferente (L_{Af})

5.2.1. Cálculo Del Avalúo De Carga.

A continuación, se muestra cómo se desarrolla el cálculo para evaluar las cargas que realiza KAI-SAD a partir del Reglamento Colombiano de Construcción Sismo Resistente NSR-10 y del Diseño Básico de concreto reforzado Vol. 2 del Ingeniero Jaime Iván Mora Samacá.

Avaluó de Cargas (WD)		
	kN/m ²	kN/m
Losa Superior	$e_{Sup} * \gamma$	$e_{Sup} * \gamma * L_{Af}$
Losa Inferior	$e_{Inf} * \gamma$	$e_{Inf} * \gamma * L_{Af}$
Pisos y Acabados (B.3.4.1-2 y B.3.4.1-3)	W_{PP}	$W_{PP} * L_{Af}$
Vigueta	$\frac{Vigueta_{base} * Vigueta_{altura} * \gamma}{L_{Af}}$	$Vigueta_{base} * Vigueta_{altura} * \gamma$
Viga	$\frac{Viga_{base} * Viga_{altura} * \gamma}{L_{Af}}$	$Viga_{base} * Viga_{altura} * \gamma$
Casetón (B.3.4.1-3)	W_{PP}	$W_{PP} * L_{Af}$
Muros Divisorios (B.3.4.2-4)	$\frac{W_{PP} * H_{Libre}}{L_{Af}}$	$W_{PP} * H_{Libre}$
	Σ	Σ

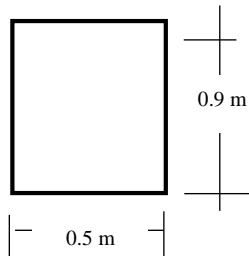
Tabla 2. Avaluó de cargas para la carga muerta **Fuente:** propia

W_{PP} : Peso propio del componente seleccionado dentro de los listados de KAI-SAD.

γ : Densidad del concreto reforzado $24 \frac{kg}{m^3}$

Una vez comprendido esto se muestra en el ejercicio de aplicación 2

Dimensión transversal de la viga



Dimensión transversal de la vigueta

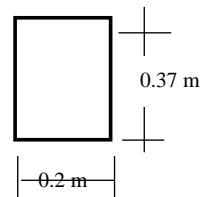


Grafico 14. Dimensiones de la viga y la vigueta. **Fuente:** Propia

Base de la columna C_b : 0.4 m

Recubrimiento: 4.5 cm

A_v : 0.64 cm

H_{Libre} : 2.7 m

e_{Sup} : 0.049 m

e_{Inf} : 0.025 m

L_{Af} : 1.3 m

Avalúo de Cargas (WD)		
	kN/m ²	kN/m
Losa Superior	1.176	1.5288
Losa Inferior	0.6	0.78
Pisos y Acabados (B.3.4.1-2 y B.3.4.1-3)	1.1	1.43
Vigueta	1.36615	1.776
Viga	8.30769	10.8
Casetón (B.3.4.1-3)	0.2	0.26
Muros Divisorios (B.3.4.2-4)	9.13846	1.88
	21.8883	28.4548

Tabla 3. Avalúo de cargas del ejercicio de aplicación 2. **Fuente:** propia

Se hizo uso y ocupación del componen *Baldosa sobre 25mm de mortero* para pisos y acabado, *madera densa 25mm* para casetón y *mampostería de bloque de arcilla – pañetado en ambas caras de espesor 300mm* para muros. Para este ejercicio se escogió $W_D = 28.4548 \frac{kN}{m}$ la cual deberá sumar el peso propio de la viga, conforme a la siguiente formula.

$$PP_{VG} = \gamma * V_b * V_h$$

$$W_D = 28.4548 \frac{kN}{m} + (24 * 0.5 * 0.9) = 39.2548$$

Así mismo, para la carga viva tendrá un uso de $W_L = 6 \text{ kN/m}$ con ocupación en almacenamiento liviano.

Luego de obtener el análisis de carga por medio de la metodología del avalúo de cargas, se procede a la identificación de la carga mayorada (W_U), coeficiente de carga (U), factor de seguridad (F_S), carga sin mayorar (W) y altura efectiva útil (d) como se muestra a continuación:

Carga mayorada (W_U)

$$W_U = (1.2 * W_D) + (1.6 * W_L)$$

$$W_U = (1.2 * 39.2548) + (1.6 * 6) = 56.7058 \frac{\text{kN}}{\text{m}}$$

Carga sin mayorar (W)

$$W = W_D + W_L$$

$$W = 39.2548 + 6 = 45.2548 \text{ kN/m}$$

Coeficiente de carga (U)

$$U = \frac{W_U}{W}$$

$$U = \frac{56.7058}{45.2548} = 1.253$$

Factor de seguridad (F_S)

$$F_S = \frac{U}{\phi}$$

$$F_s = \frac{1.253}{0.9} = 1.392$$

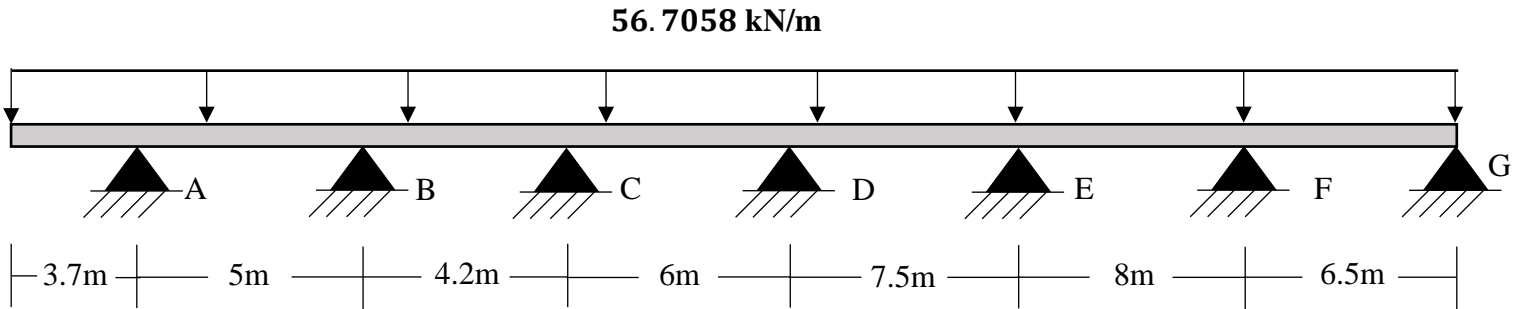


Gráfico 15. Diagrama de carga viga hiperestática 2. Fuente: propia.

Se procede a realizar los diagramas de fuerza cortante y momento flector. Para el procedimiento 1 solamente aplica el diagrama de fuerza cortante, mientras que para procedimiento 2 aplica en conjunto los dos diagramas.

5.2.2. Diagramas De Fuerza Cortante Y Momento Flector

Para el programa se obtuvieron los puntos de coordenadas con los cuales fue posible realizar el diagrama de fuerza cortante a partir del procedimiento 1 pero este es basado solamente para carga uniformemente distribuida. El programa halla las áreas que se generan a partir de estas coordenadas para poder calcular el diagrama de momento flector.

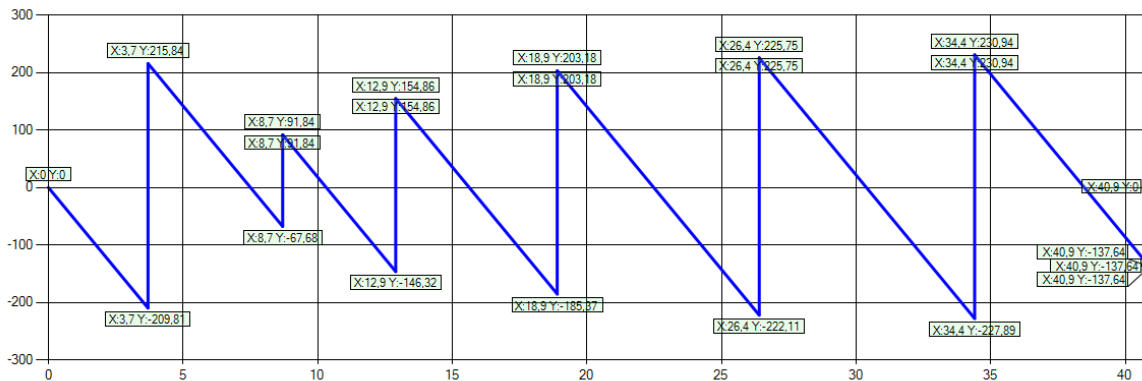


Gráfico 16. Diagrama de gráfica cortante de viga hiperestática 2. Fuente: propia

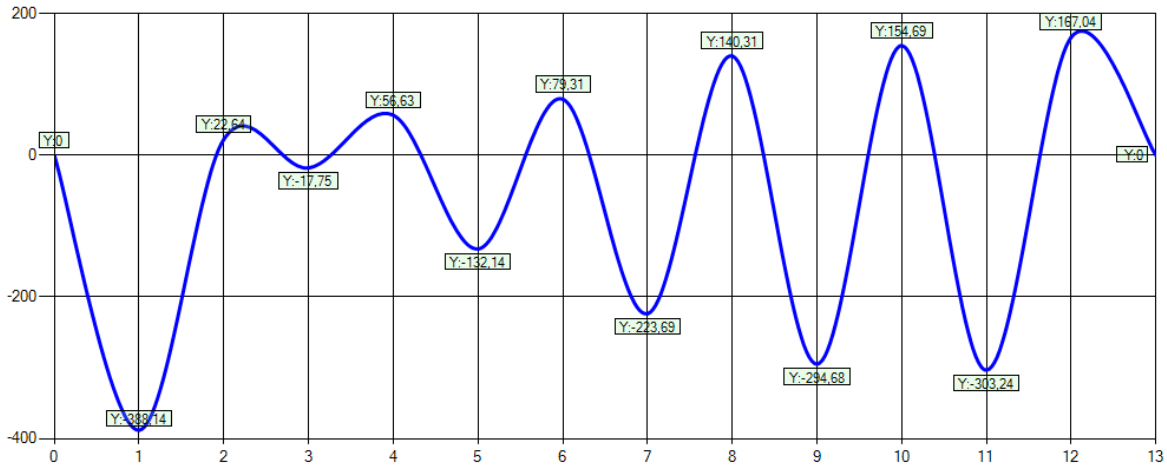


Grafico 17. Diagrama de gráfica momento flector de viga hiperestática 2. **Fuente:** propia

5.2.3. Cálculo Refuerzos Longitudinales.

Dando continuidad al ejercicio de aplicación 2 y con base a los momentos flectores máximos se realiza la asignación de los cálculos de refuerzos longitudinales.

Cálculo De Cuantías De Cada Luz Y Los Nodos

Para este proceso se requiere definir parámetros como los son el momento de diseño “ M_1 ” y momento de requerimiento “ M_2 ”. El momento de diseño hace referencia al momento máximo obtenido en el diagrama de momento flector mientras que el momento de requerimiento se calcula de la siguiente manera:

$$M_2 = \phi * f_y * \rho_{Max} * \left(1 - 0,59 * \frac{\rho_{Max} * f_y}{f'_c}\right) * V_b * d^2$$

M_2 = el momento de diseño

ϕ = Factor de reducción para momentos resistentes 0.9

f_y = resistencia a la tracción de 420 Mpa

ρ_{Max} = cuantía máxima

f'_c = resistencia a la compresión

V_b = base de la viga

$d^2 = \text{altura efectiva útil}$

Aplicado a este ejercicio el momento de diseño es igual a:

$$\rho_{Min} = 0.003333$$

$$\rho_{Max} = 0.01588$$

$$M_2 = 0.9 * 420000 * 0.01588 * \left(1 - 0.59 * \frac{0.01588 * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$M_2 = 1843.08 \text{ kN} * \text{m}$$

Asignación de Cuantía (ρ) Para Cada Luz

- *Cálculo de cuantía en el Voladizo*

$$388.14 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$\rho_{bal} A = \mathbf{0.002894}$$

$M_1 < M_2$ No necesita refuerzo a compresión.

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.003333 * 50 * 85.5 = \mathbf{14.2486 \text{ cm}}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 2I#3

2. 3#8

3. 12#4

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01588 * 50 * 85.5 = \mathbf{67.887 \text{ cm}}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 24#6

2. 18#7

3. 11#9

Nota. 12. Para KAI-SAD, el valor del momento en el voladizo está representado para el nodo A

- *Cálculo de cuantía de la luz del nodo A al nodo B*

$$22.64 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$\rho_{bal} \mathbf{AB} = \mathbf{0.000164}$$

$M_1 < M_2$ No necesita refuerzo a compresión

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.003333 * 50 * 85.5 = \mathbf{14.2486 \text{ cm}}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 21#3

2. 3#8

3. 12#4

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01588 * 50 * 85.5 = \mathbf{67.887 \text{ cm}}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 24#6

2. 18#7

3. 11#9

- *Cálculo de cuantía del nodo B*

$$17.75 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$\rho_{bal} \mathbf{B} = \mathbf{0.000129}$$

$M_1 < M_2$ No necesita refuerzo a compresión

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.003333 * 50 * 85.5 = \mathbf{14.2486 \text{ cm}}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 2I#3

2. 3#8

3. 12#4

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01588 * 50 * 85.5 = \mathbf{67.887 \text{ cm}}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 24#6

2. 18#7

3. 11#9

- *Cálculo de cuantía de la luz del nodo B al nodo C*

$$56.63 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$\rho_{bal} \mathbf{BC} = \mathbf{0.000412}$$

$M_1 < M_2$ No necesita refuerzo a compresión

$$AS' = 0.00333$$

$$AS' = \rho_{Min} * b * d$$

$$3 * 50 * 85.5 = \mathbf{14.2486 \text{ cm}}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 2I#3

2. 3#8

3. 12#4

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01588 * 50 * 85.5 = \mathbf{67.887 \text{ cm}}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 24#6
2. 18#7
3. 11#9

- *Cálculo de cuantía del nodo C*

$$132.14 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$\rho_{bal} C = \mathbf{0.000966}$$

$M_1 < M_2$ No necesita refuerzo a compresión

$$AS' = 0.00333$$

$$AS' = \rho_{Min} * b * d$$

$$3 * 50 * 85.5 = \mathbf{14.2486 \text{ cm}}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 21#3
2. 3#8
3. 12#4

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01588 * 50 * 85.5 = \mathbf{67.887 \text{ cm}}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 24#6
2. 18#7
3. 11#9

- *Cálculo de cuantía de del nodo C al nodo D*

$$79.31 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$\rho_{bal}CD = 0.000577$$

$M_1 < M_2$ No necesita refuerzo a compresión

$$AS' = 0.00333$$

$$AS' = \rho_{Min} * b * d$$

$$3 * 50 * 85.5 = 14.2486 \text{ cm}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 2I#3

2. 3#8

3. 12#4

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01588 * 50 * 85.5 = 67.887 \text{ cm}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 24#6

2. 18#7

3. 11#9

- Cálculo de cuantía de del nodo D

$$223.69 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$\rho_{bal} D = 0.001646$$

$M_1 < M_2$ No necesita refuerzo a compresión.

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.003333 * 50 * 85.5 = 14.2486 \text{ cm}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 2I#3

2. 3#8

3. 12#4

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01588 * 50 * 85.5 = \mathbf{67.887 \text{ cm}}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 24#6
2. 18#7
3. 11#9

- *Cálculo de cuantía de del nodo D al nodo E*

$$140.31 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$\rho_{bal}DE = \mathbf{0.001026}$$

$M_1 < M_2$ No necesita refuerzo a compresión.

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.003333 * 50 * 85.5 = \mathbf{14.2486 \text{ cm}}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 21#3
2. 3#8
3. 12#4

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01588 * 50 * 85.5 = \mathbf{67.887 \text{ cm}}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 24#6
2. 18#7
3. 11#9

- *Cálculo de cuantía del nodo E*

$$294.68 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$\rho_{bal} E = 0.002181$$

$M_1 < M_2$ No necesita refuerzo a compresión.

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.003333 * 50 * 85.5 = 14.2486 \text{ cm}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 2I#3

2. 3#8

3. 12#4

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01588 * 50 * 85.5 = 67.887 \text{ cm}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 24#6

2. 18#7

3. 11#9

- Cálculo de cuantía de la luz del nodo E al nodo F

$$154.69 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$\rho_{bal} EF = 0.001133$$

$M_1 < M_2$ No necesita refuerzo a compresión.

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.003333 * 50 * 85.5 = 14.2486 \text{ cm}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 2I#3

2. 3#8

3. 12#4

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01588 * 50 * 85.5 = \mathbf{67.887 \text{ cm}}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 24#6
2. 18#7
3. 11#9

- Cálculo de cuantía del nodo F

$$303.24 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$\rho_{bal} F = \mathbf{0.002246}$$

$M_1 < M_2$ No necesita refuerzo a compresión.

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.003333 * 50 * 85.5 = \mathbf{14.2486 \text{ cm}}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 21#3
2. 3#8
3. 12#4

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01588 * 50 * 85.5 = \mathbf{67.887 \text{ cm}}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 24#6
2. 18#7
3. 11#9

- Cálculo de cuantía de la luz del nodo F al nodo G

$$167.04 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$\rho_{bal}FG = 0.001224$$

$M_1 < M_2$ No necesita refuerzo a compresión.

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.003333 * 50 * 85.5 = 14.2486 \text{ cm}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 2I#3
2. 3#8
3. 12#4

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01588 * 50 * 85.5 = 67.887 \text{ cm}$$

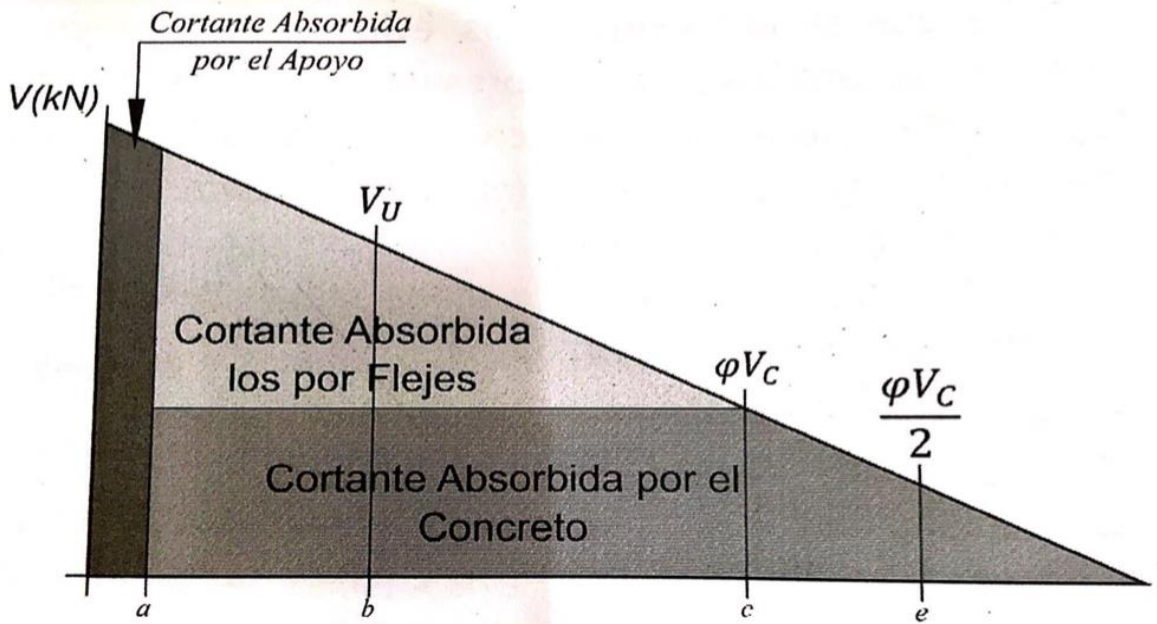
Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 24#6
2. 18#7
3. 11#9

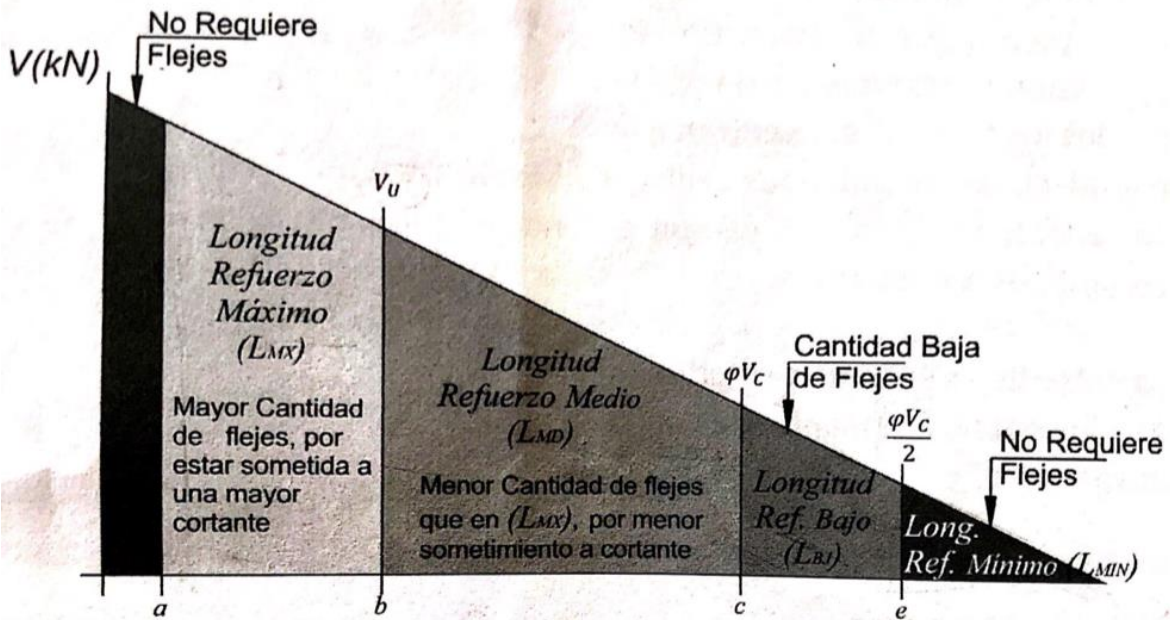
5.2.4. Cálculo Refuerzos Transversales.

En este literal, se calculan los refuerzos transversales a partir del método de Diseño a Cortante, a continuación, se definen las variables:

- V_{MAX} Fuerza cortante máxima de la luz, es el mayor valor de las fuerzas cortantes de una luz.
- V_U Fuerza cortante mayorada.
- ϕV_C Resistencia nominal a la cortante del concreto multiplicada por el factor de reducción.
- d Altura efectiva útil a tensión.
- s Es la separación entre flejes.
- n Número de flejes.
- B_w Base de la viga



Gráfica 18. Diagrama de cortante **Fuente.** (Samacá, 2014)



Gráfica 19. Diagrama de cortante. Cantidad y separación entre flejes. **Fuente.** (Samacá, 2014)

Separación De Flejes

$$S_{Min} = \frac{d}{2}$$
$$Sep. = \frac{\rho * Av * d}{(Vu - \phi V_c)}$$
$$S = \frac{Av * fy}{0.062 * \sqrt{f'_c} * bw}$$

Nota 13. KAI-SAD no ejecuta diseño cortante para voladizos.

- Refuerzo transversal de la luz del nodo A al nodo B

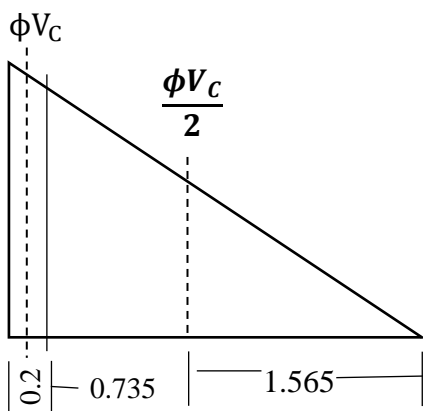
$$V_U = V_{Max} - W_U(0.2 + d)$$

$$V_U = 215.84 - 56.7058 * (0.2 + 0.855) = 156.015 \text{ kN}$$

$$\phi V_c = \phi * 0.17 * \sqrt{f'_c} * b * d$$

$$\phi V_c = 0.75 * 0.17 * \sqrt{24.6} * 0.5 * 0.855 = 270.342 \text{ kN}$$

$$\frac{\phi V_c}{2} = \frac{270.342 \text{ kN}}{2} = 135.171 \text{ kN}$$



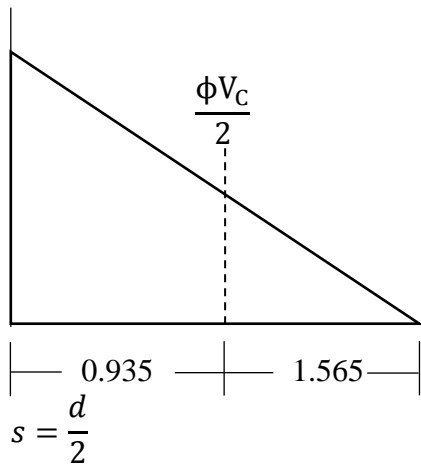
$$V_U > \phi V_c$$

$$x = \frac{135.171 * 2.5}{215.84} = 1.565$$

$$L_{\frac{\phi V_C}{2}} = 2.5 - 0.2 - 1.565$$

$$L_{\frac{\phi V_C}{2}} = 0.735$$

V (kN)



$$s = \frac{85.5}{2} = 42.75 \text{ cm}$$

$$\text{No. Flejes} = \frac{0.735}{0.4275} + 1 = 2.719 \approx 3 \text{ UN}$$

- *Cálculo de refuerzo transversal de la luz del nodo B al nodo C*

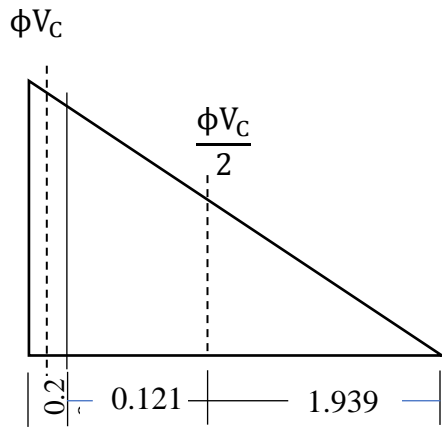
$$V_U = V_{Max} - W_U(0.2 + d)$$

$$V_U = 146.32 - 56.7058 * (0.2 + 0.855) = 86.495 \text{ kN}$$

$$\phi V_C = \phi * 0.17 * \sqrt{f'_c} * b * d$$

$$\phi V_C = 0.75 * 0.17 * \sqrt{24.6} * 0.5 * 0.855 = 270.342 \text{ kN}$$

$$\frac{\phi V_C}{2} = \frac{270.342 \text{ kN}}{2} = 135.171 \text{ kN}$$

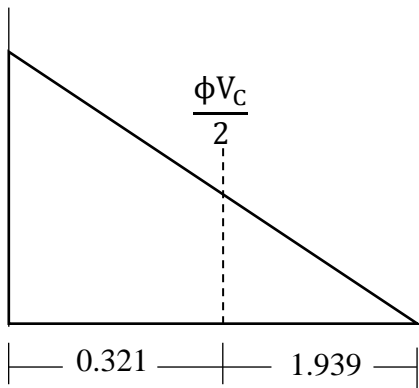


$$V_U > \phi V_C$$

$$x = \frac{135.171 * 2.1}{146.32} = 1.939$$

$$L_{\frac{\phi V_C}{2}} = 4.2 - 0.2 - 1.939$$

V (kN)



$$s = \frac{d}{2}$$

$$s = \frac{85.5}{2} = 42.75 \text{ cm}$$

$$\text{No. Flejes} = \frac{0.121}{0.4275} + 1 = 1.283 \approx 2 \text{ UN}$$

- *Cálculo de refuerzo transversal de la luz del nodo C al nodo D*

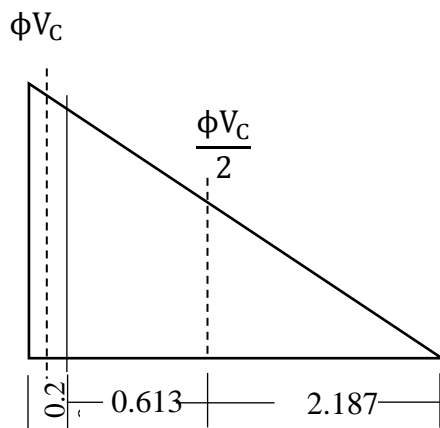
$$V_U = V_{Max} - W_U(0.2 + d)$$

$$V_U = 185.37 - 56.7058 * (0.2 + 0.855) = 125.545 \text{ kN}$$

$$\phi V_C = \phi * 0.17 * \sqrt{f'_c} * b * d$$

$$\phi V_C = 0.75 * 0.17 * \sqrt{24.6} * 0.5 * 0.855 = 270.342 \text{ kN}$$

$$\frac{\phi V_C}{2} = \frac{270.342 \text{ kN}}{2} = 135.171 \text{ kN}$$



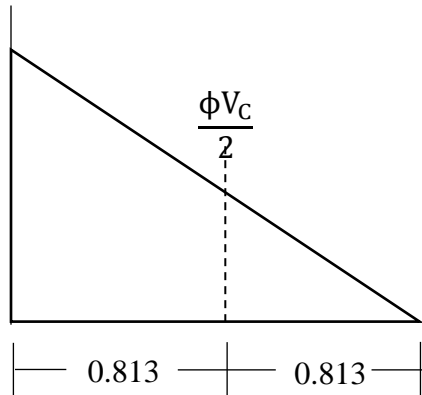
$$V_U > \phi V_C$$

$$x = \frac{135.171 * 3}{185.37} = \mathbf{2.187}$$

$$L_{\frac{\phi V_C}{2}} = 3 - 0.2 - 2.187$$

$$\frac{L\phi V_C}{2} = 0.613$$

V (kN)



$$s = \frac{d}{2}$$

$$s = \frac{85.5}{2} = 42.75 \text{ cm}$$

$$\text{No. Flejes} = \frac{0.613}{0.4275} + 1 = 2.434 \approx 3 \text{ UN}$$

- Refuerzo transversal de la luz del nodo D al nodo E

$$V_U = V_{Max} - W_U(0.2 + d)$$

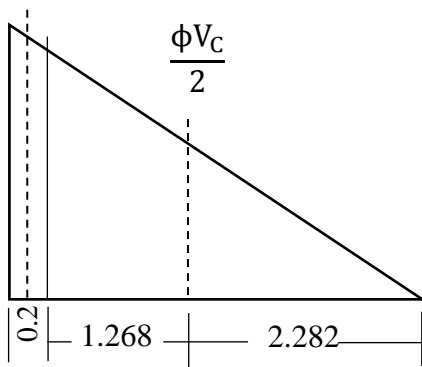
$$V_U = 222.11 - 56.7058 * (0.2 + 0.855) = 162.28 \text{ kN}$$

$$\phi V_C = \phi * 0.17 * \sqrt{f'_c} * b * d$$

$$\phi V_C = 0.75 * 0.17 * \sqrt{24.6} * 0.5 * 0.855 = 270.342 \text{ kN}$$

$$\frac{\phi V_C}{2} = \frac{270.342 \text{ kN}}{2} = 135.171 \text{ kN}$$

ϕV_C



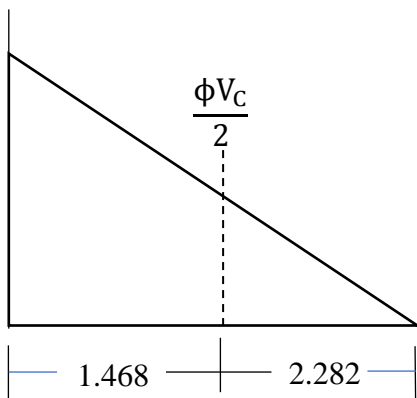
$$V_U > \phi V_c$$

$$x = \frac{135.171 * 3.75}{222.11} = 2.282$$

$$L_{\frac{\phi V_c}{2}} = 3.75 - 0.2 - 2.282$$

$$L_{\frac{\phi V_c}{2}} = 1.268$$

V (kN)



$$s = \frac{d}{2}$$

$$s = \frac{85.5}{2} = 42.75 \text{ cm}$$

$$\text{No. Flejes} = \frac{1.268}{0.4275} + 1 = 3.966 \approx 4 \text{ UN}$$

- Refuerzo transversal de la luz del nodo E al nodo F

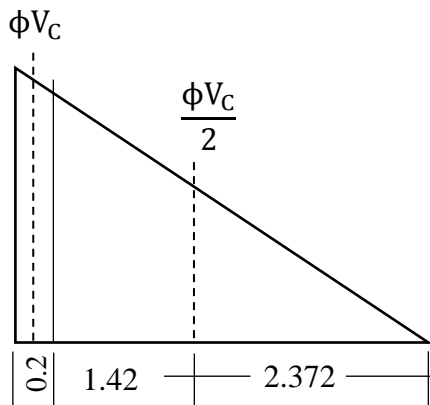
$$V_U = V_{Max} - W_U(0.2 + d)$$

$$V_U = 227.89 - 56.7058 * (0.2 + 0.855) = 168.065 \text{ kN}$$

$$\phi V_C = \phi * 0.17 * \sqrt{f'_c} * b * d$$

$$\phi V_C = 0.75 * 0.17 * \sqrt{24.6} * 0.5 * 0.855 = 270.342 \text{ kN}$$

$$\frac{\phi V_C}{2} = \frac{270.342 \text{ kN}}{2} = 135.171 \text{ kN}$$



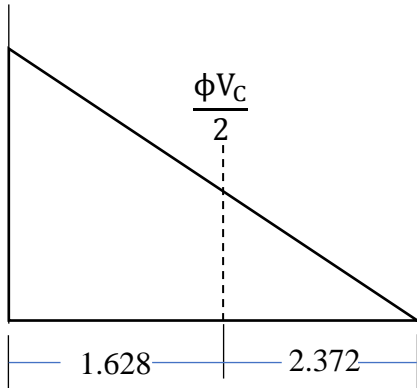
$$V_U > \phi V_C$$

$$x = \frac{135.171 * 4}{227.89} = 2.372$$

$$L_{\frac{\phi V_C}{2}} = 4 - 0.2 - 2.372$$

$$\frac{L\phi V_c}{2} = 1.428$$

V (kN)



$$s = \frac{d}{2}$$

$$s = \frac{85.5}{2} = 42.75 \text{ cm}$$

$$\text{No. Flejes} = \frac{1.428}{0.4275} + 1 = 4.34 \approx 5 \text{ UN}$$

- Refuerzo transversal de la luz del nodo F al nodo G

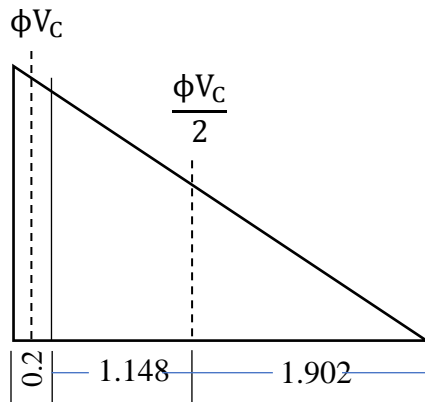
$$V_U = V_{Max} - W_U(0.2 + d)$$

$$V_U = 230.94 - 56.7058 * (0.2 + 0.855) = 171.115 \text{ kN}$$

$$\phi V_c = \phi * 0.17 * \sqrt{f'_c} * b * d$$

$$\phi V_c = 0.75 * 0.17 * \sqrt{24.6} * 0.5 * 0.855 = 270.342 \text{ kN}$$

$$\frac{\phi V_c}{2} = \frac{270.342 \text{ kN}}{2} = 135.171 \text{ kN}$$



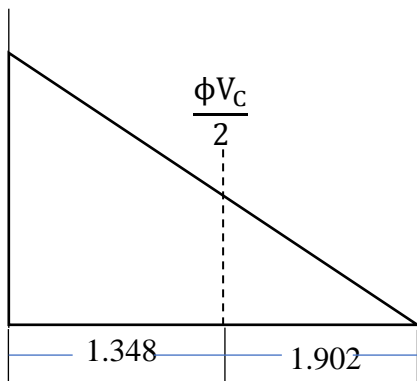
$$V_U > \phi V_C$$

$$x = \frac{135.171 * 3.25}{230.94} = 1.902$$

$$L_{\frac{\phi V_C}{2}} = 3.25 - 0.2 - 1.902$$

$$L_{\frac{\phi V_C}{2}} = 1.148$$

V (kN)



$$s = \frac{d}{2}$$

$$s = \frac{85.5}{2} = 42.75 \text{ cm}$$

$$\text{No. Flejes} = \frac{1.148}{0.4275} + 1 = 3.685 \approx 4 \text{ UN}$$

$$0 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 420}{24.6}\right) * 0.5 * 0.855^2$$

$$\rho_{bal} G = 0$$

$M_1 < M_2$ No necesita refuerzo a compresión

5.3. EJERCICIO DE APLICACIÓN 3

Se cambia dimensión de la viga quedando base:0.35 m y altura: 0.7

Longitud aferente 2.6 m

Esup: 0.55

Einf: 0.4

F'c: 14.1

Fy:240 MPa

WD: 82.196

WD+PP: 115.2912

Recubrimiento 3cm

Carga mayorada (W_U)

$$W_U = (1.2 * W_D) + (1.6 * W_L)$$

$$W_U = (1.2 * 88.076) + (1.6 * 6) = 115.2912 \frac{kN}{m}$$

Carga sin mayorar (W)

$$W = W_D + W_L$$

$$W = 88.076 + 6 = 94.076 \text{ kN/m}$$

Coficiente de carga (U)

$$U = \frac{W_U}{W}$$

$$U = \frac{115.2912}{94.076} = 1.225$$

Factor de seguridad (F_S)

$$F_S = \frac{U}{\phi}$$

$$F_S = \frac{1.253}{0.9} = 1.361$$

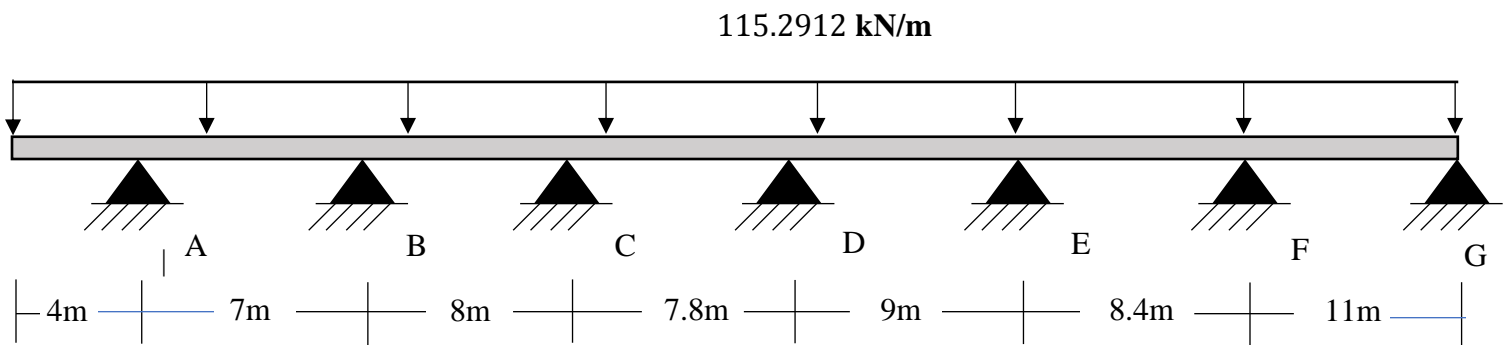
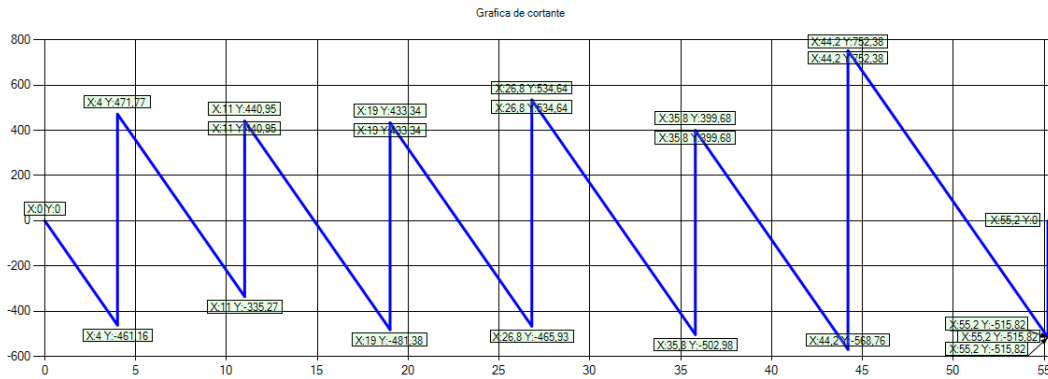


Gráfico 20. Diagrama de carga viga hiperestática 3. Fuente: propia.

5.3.1. Diagramas De Fuerza Cortante Y Momento Flector

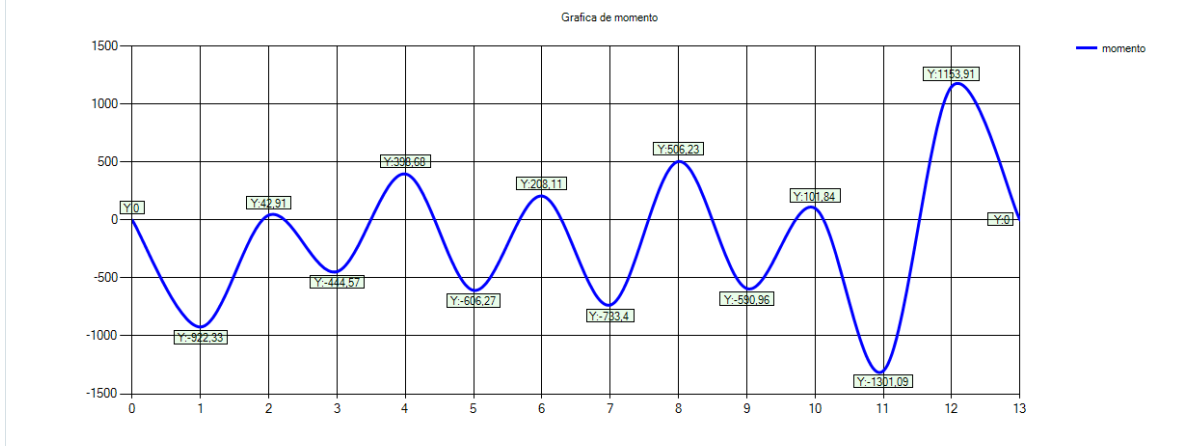
Para el programa se obtuvieron los puntos de coordenadas con los cuales fue posible realizar el diagrama de fuerza cortante a partir del procedimiento 1 pero este es basado solamente para carga uniformemente distribuida. El programa halla las áreas que se generan a partir de estas coordenadas para poder calcular el diagrama de momento flector.

Grafica1



Gráfica 21. Diagrama de gráfica cortante de viga hiperestática 3 en KAI SAD. Fuente: Propia

Grafica2



Gráfica 22. Diagrama de gráfica momento flector de viga hiperestática 3 en KAI SAD. Fuente: Propia

5.3.2. Cálculo Refuerzos Longitudinales

Dando continuidad al ejercicio de aplicación 2 se planteará viceversa al resultado anterior que no necesita refuerzo, siendo este ahora si necesita refuerzo con base a los momentos flectores máximos se realiza la asignación de los cálculos de refuerzos longitudinales.

Cálculo De Cuantías De Cada Luz Y Los Nodos

Para este proceso se requiere definir parámetros como los son el momento de diseño " M_1 " y momento de requerimiento " M_2 ". El momento de diseño hace referencia al momento máximo obtenido en el diagrama de momento flector mientras que el momento de requerimiento se calcula de la siguiente manera:

$$M_2 = \phi * f_y * \rho_{Max} * \left(1 - 0.59 * \frac{\rho_{Max} * f_y}{f'_c}\right) * V_b * d^2$$

$$M_2 = 0.9 * 240000 * 0.01593 * \left(1 - 0.59 * \frac{0.01593 * 240}{14.1}\right) * 0.35 * 0.67^2$$

$$M_2 = 454.128 \text{ kN} * \text{m}$$

Asignación de Cuantía (ρ) Para Cada Luz

- Cálculo de cuantía en el Voladizo

$$922.33 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 240}{14.1}\right) * 0.35 * 0.67^2$$

$M_1 > M_2$ Si necesita refuerzo a compresión.

$$AS' = \frac{M_1 - M_2}{\phi * f_y * (d - d')}$$

$$AS' = \frac{922.33 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS' = 33.868 \text{ cm}^2$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 27#4

2. 17#5

3. 12#6

$$AS = \rho_{max} * b * d + \frac{M_1 - M_2}{\phi * f_y * (d - d')}$$

$$AS = 0.01593 * 0.35 * 0.7 + \frac{922.33 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS = 72.897 \text{ cm}^2$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 26#6

2. 19#7

3. 9#10

Nota. Para KAI-SAD, el valor del momento en el voladizo está representado para el nodo A

- Cálculo de cuantía de la luz del nodo A al nodo B

$$42.91 = 0.9 * 240000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 240}{14.1}\right) * 0.35 * 0.67^2$$

$$\rho_{bal} AB = 0.001281$$

$M_1 < M_2$ No necesita refuerzo a compresión

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.005833 * 35 * 67 = 13.678 \text{ cm}^2$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 20#3
2. 11#4
3. 7#5

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01593 * 35 * 67 = 37.355 \text{ cm}^2$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 29#4
2. 19#5
3. 10#7

- Cálculo de cuantía del nodo B

$$444.57 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 240}{14.1}\right) * 0.35 * 0.67^2$$

$$\rho_{bal} B = 0.015518$$

$M_1 < M_2$ No necesita refuerzo a compresión

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.005833 * 35 * 67 = \mathbf{13.678 \text{ cm}^2}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 20#3
2. 11#4
3. 7#5

$$AS = \rho_{bal} * b * d$$

$$AS = 0.015518 * 35 * 67 = \mathbf{36.389 \text{ cm}^2}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 29#4
2. 19#5
3. 13#6

- *Cálculo de cuantía de la luz del nodo B al nodo C*

$$398.68 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 240}{14.1}\right) * 0.35 * 0.67^2$$

$$\rho_{bal}BC = \mathbf{0.01241}$$

$M_1 < M_2$ No necesita refuerzo a compresión

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.005833 * 35 * 67 = \mathbf{13.678 \text{ cm}^2}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 20#3
2. 11#4
3. 7#5

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01241 * 35 * 67 = \mathbf{29.101 \text{ cm}^2}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 23#4
2. 15#5

3. 6#8

- Cálculo de cuantía del nodo C

$$606.27 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 240}{14.1}\right) * 0.35 * 0.67^2$$

$$\rho_{bal} C = 0.02333$$

$M_1 > M_2$ Si necesita refuerzo a compresión.

$$AS' = \frac{M_1 - M_2}{\phi * f_y * (d - d')}$$

$$AS' = \frac{606.27 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS' = 11 \text{ cm}^2$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 16#3

2. 9#4

3. 4#6

$$AS = \rho_{max} * b * d + \frac{M_1 - M_2}{\phi * f_y * (d - d')}$$

$$AS = 0.01593 * 0.35 * 0.7 + \frac{606.27 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS = 48.361 \text{ cm}^2$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 17#6

2. 13#7

3. 6#10

- Cálculo de cuantía de del nodo C al nodo D

$$208.11 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 240}{14.1}\right) * 0.35 * 0.67^2$$

$$\rho_{bal} CD = 0.006565$$

$M_1 < M_2$ No necesita refuerzo a compresión

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.005833 * 35 * 67 = \mathbf{13.678 \text{ cm}^2}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 20#3

2. 11#4

3. 7#5

$$AS = \rho_{bal} * b * d$$

$$AS = 0.006565 * 35 * 67 = \mathbf{15.394 \text{ cm}^2}$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 22#3

2. 12#4

3. 4#7

- Cálculo de cuantía de del nodo D

$$733.4 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 240}{14.1}\right) * 0.35 * 0.67^2$$

$$\rho_{bal} D = \mathbf{0.031707}$$

$M_1 > M_2$ Si necesita refuerzo a compresión.

$$AS' = \frac{M_1 - M_2}{\phi * f_y * (d - d')}$$

$$AS' = \frac{733.4 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS' = \mathbf{20.201 \text{ cm}^2}$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 29#3

2. 16#4

3. 4#8

$$AS = \rho_{max} * b * d + \frac{M_1 - M_2}{\phi * f_y * (d - d')}$$

$$AS = 0.01593 * 0.35 * 0.7 + \frac{733.4 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS = 57.557 \text{ cm}^2$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 29#5

2. 21#6

3. 15#7

- Cálculo de cuantía de del nodo D al nodo E

$$506.23 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 240}{14.1}\right) * 0.35 * 0.67^2$$

$$\rho_{bal} DE = 0.01826$$

$M_1 > M_2$ Si necesita refuerzo a compresión.

$$AS' = \frac{M_1 - M_2}{\phi * f_y * (d - d')}$$

$$AS' = \frac{506.23 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS' = 3.768 \text{ cm}^2$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 12#2

2. 3#4

3. 1#7

$$AS = \rho_{max} * b * d + \frac{M_1 - M_2}{\phi * f_y * (d - d')}$$

$$AS = 0.01593 * 0.35 * 0.7 + \frac{506.23 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS = 41.124 \text{ cm}^2$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 21#5
2. 15#6
3. 11#7

- Cálculo de cuantía del nodo E

$$590.96 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 240}{14.1}\right) * 0.35 * 0.67^2$$

$$\rho_{bal} E = 0.02249$$

$M_1 > M_2$ Si necesita refuerzo a compresión.

$$AS' = \frac{M_1 - M_2}{\phi * fy * (d - d')}$$

$$AS' = \frac{590.96 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS' = 9.898 \text{ cm}^2$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 14#3
2. 8#4
3. 5#5

$$AS = \rho_{max} * b * d + \frac{M_1 - M_2}{\phi * fy * (d - d')}$$

$$AS = 0.01593 * 0.35 * 0.7 + \frac{590.96 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS = 47.254 \text{ cm}^2$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 24#5
2. 17#6

3. 6#10

- Cálculo de cuantía de la luz del nodo E al nodo F

$$101.84 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 240}{14.1}\right) * 0.35 * 0.67^2$$

$$\rho_{bal} EF = 0.003097$$

$M_1 < M_2$ No necesita refuerzo a compresión

$$AS' = \rho_{Min} * b * d$$

$$AS' = 0.005833 * 35 * 67 = 13.678 \text{ cm}^2$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 20#3

2. 11#4

3. 7#5

$$AS = \rho_{bal} * b * d$$

$$AS = 0.01593 * 35 * 67 = 37.355 \text{ cm}^2$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 29#4

2. 19#5

3. 10#7

- Cálculo de cuantía del nodo F

$$1301.09 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 240}{14.1}\right) * 0.35 * 0.67^2$$

$M_1 > M_2$ Si necesita refuerzo a compresión.

$$AS' = \frac{M_1 - M_2}{\phi * f_y * (d - d')}$$

$$AS' = \frac{1301.09 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS' = 61.267 \text{ cm}^2$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 22#6

2. 16#7

3. 10#9

$$AS = \rho_{max} * b * d + \frac{M_1 - M_2}{\phi * f_y * (d - d')}$$

$$AS = 0.01593 * 0.35 * 0.7 + \frac{1301.09 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS = 98.623 \text{ cm}^2$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 26#7

2. 20#8

3. 16#9

- *Cálculo de cuantía de la luz del nodo F al nodo G*

$$1153.91 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 240}{14.1}\right) * 0.35 * 0.67^2$$

$M_1 > M_2$ Si necesita refuerzo a compresión.

$$AS' = \frac{M_1 - M_2}{\phi * f_y * (d - d')}$$

$$AS' = \frac{1153.91 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS' = 50.62 \text{ cm}^2$$

Opciones de refuerzo longitudinal de compresión KAI-SAD

1. 26#5

2. 18#6

3. 10#8

$$AS = \rho_{max} * b * d + \frac{M_1 - M_2}{\phi * f_y * (d - d')}$$

$$AS = 0.01593 * 0.35 * 0.7 + \frac{1153.91 - 454.128}{0.9 * 240000 * (0.67 - 0.03)}$$

$$AS = 87.976 \text{ cm}^2$$

Opciones de refuerzo longitudinal de tensión KAI-SAD

1. 23#7
2. 18#8
3. 14#9

5.3.3. Cálculo Refuerzos Transversales.

En este literal, se calculan los refuerzos transversales a partir del método de Diseño a Cortante, a continuación se definen las variables:

- V_{MAX} Fuerza cortante máxima de la luz, es el mayor valor de las fuerzas cortantes de una luz.
- V_U Fuerza cortante mayorada.
- ϕV_C Resistencia nominal a la cortante del concreto multiplicada por el factor de reducción.
- d Altura efectiva útil a tensión.
- s Es la separación entre flejes.
- n Número de flejes.
- B_w Base de la viga

Separación De Flejes

$$S_{Min} = \frac{d}{2}$$

$$Sep. = \frac{\rho * A_v * d}{(V_u - \phi V_C)}$$

$$S = \frac{A_v * f_y}{0.062 * \sqrt{f'_c} * b_w}$$

Nota. KAI-SAD no ejecuta diseño cortante para voladizos.

- Refuerzo transversal de la luz del nodo A al nodo B

$$V_U = V_{Max} - W_U(0.2 + d)$$

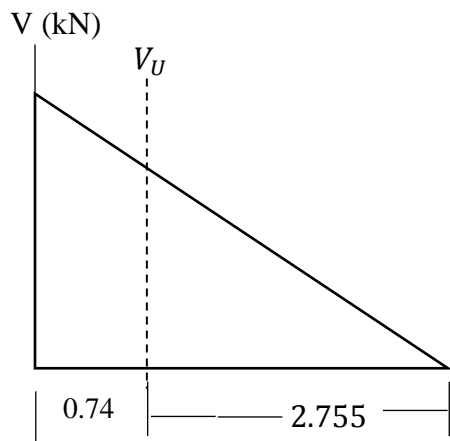
$$V_U = 471.77 - 115.2912 * (0.2 + 0.67) = 371.468 \text{ kN}$$

$$\phi V_C = \phi * 0.17 * \sqrt{f'_c} * b * d$$

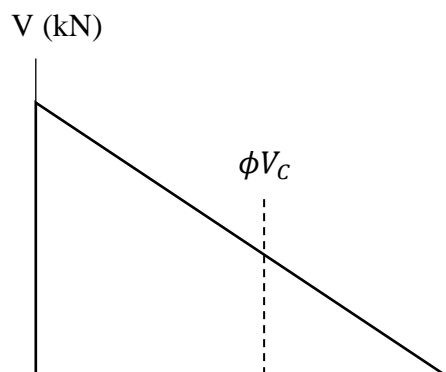
$$\phi V_C = 0.75 * 0.17 * \sqrt{14.1} * 0.35 * 0.67 = 112.27 \text{ kN}$$

$$\frac{\phi V_C}{2} = \frac{112.27 \text{ kN}}{2} = 56.135 \text{ kN}$$

$$V_U < \phi V_C$$



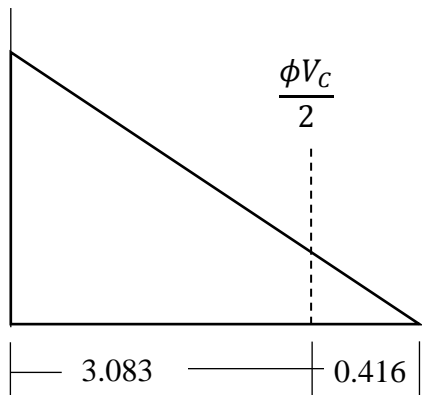
$$x = \frac{371.468 * 3.5}{471.77} = 2.755$$



$$\left| \begin{array}{c} \text{---} 2.665 \text{---} \\ \text{---} 0.832 \text{---} \end{array} \right|$$

$$x = \frac{112.27 * 3.5}{471.77} = 0.832$$

V (kN)



$$x = \frac{56.135 * 3.5}{471.77} = 0.416$$

$$s = \frac{\phi * A_v * f_y * d}{(V_U - \phi V_C)}$$

$$s = \frac{0.75 * 0.000064 * 240000 * 0.67}{(371.468 - 112.27)} = 2.97\text{cm}$$

$$s = \frac{A_v * f_y}{0.062 * \sqrt{f'_c} * V_b}$$

$$s = \frac{0.64 * 240}{0.062 * \sqrt{14.1} * 0.35} = 18.85\text{cm}$$

$$s = \frac{d}{2}$$

$$s = \frac{67}{2} = 33.5\text{cm}$$

$$\text{No. Flejes } L_{\max} = \frac{74}{2.97} + 1 = 25.91 \approx 26 \text{ UN}$$

$$\text{No. Flejes } L_{\text{med}} = \frac{192.5}{2.97} = 64.81 \approx 65 \text{ UN}$$

$$\text{No. Flejes } L_{\text{bajo}} = \frac{41.8}{2.97} = 14.07 \approx 14 \text{ UN}$$

- *Cálculo de refuerzo transversal de la luz del nodo B al nodo C*

$$V_U = V_{\text{Max}} - W_U(0.2 + d)$$

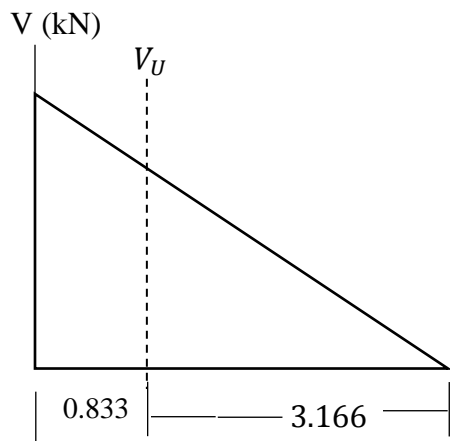
$$V_U = 481.38 - 115.2912 * (0.2 + 0.67) = 381.077 \text{ kN}$$

$$\phi V_C = \phi * 0.17 * \sqrt{f'_c} * b * d$$

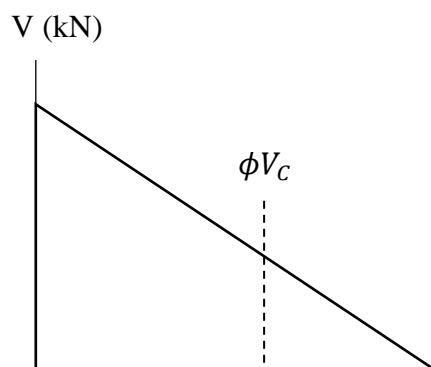
$$\phi V_C = 0.75 * 0.17 * \sqrt{14.1} * 0.35 * 0.67 = 112.27 \text{ kN}$$

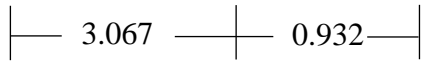
$$\frac{\phi V_C}{2} = \frac{112.27 \text{ kN}}{2} = 56.135 \text{ kN}$$

$$V_U < \phi V_C$$



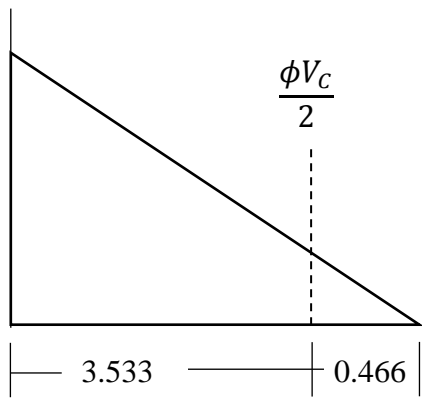
$$x = \frac{381.077 * 4}{481.38} = 3.166$$





$$x = \frac{112.27 * 4}{481.38} = 0.932$$

V (kN)



$$x = \frac{56.135 * 4}{481.38} = 0.466$$

$$s = \frac{\phi * A_V * f_y * d}{(V_U - \phi V_C)}$$

$$s = \frac{0.75 * 0.000064 * 240000 * 0.67}{(481.38 - 112.27)} = 2.091\text{cm}$$

$$s = \frac{A_V * f_y}{0.062 * \sqrt{f'_C} * V_b}$$

$$s = \frac{0.64 * 240}{0.062 * \sqrt{14.1} * 35} = 18.85\text{cm}$$

$$s = \frac{d}{2}$$

$$s = \frac{67}{2} = 33.5 \text{ cm}$$

$$\text{No. Flejes Lmax} = \frac{83.3}{2.091} + 1 = 40.837 \approx 41 \text{ UN}$$

$$\text{No. Flejes Lmed} = \frac{223.4}{2.091} = 106.83 \approx 107 \text{ UN}$$

$$\text{No. Flejes Lbajo} = \frac{46.6}{2.091} = 22.286 \approx 22 \text{ UN}$$

- *Cálculo de refuerzo transversal de la luz del nodo C al nodo D*

$$V_U = V_{Max} - W_U(0.2 + d)$$

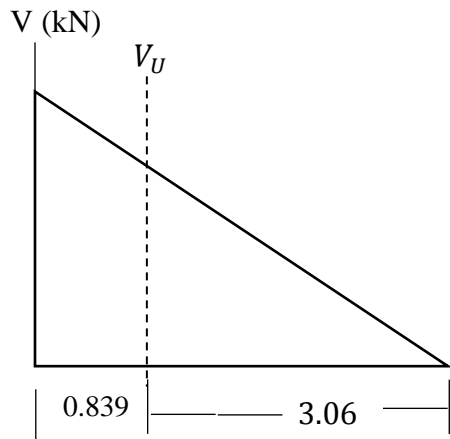
$$V_U = 465.93 - 115.2912 * (0.2 + 0.67) = 365.627 \text{ kN}$$

$$\phi V_C = \phi * 0.17 * \sqrt{f'_c} * b * d$$

$$\phi V_C = 0.75 * 0.17 * \sqrt{14.1} * 0.35 * 0.67 = 112.27 \text{ kN}$$

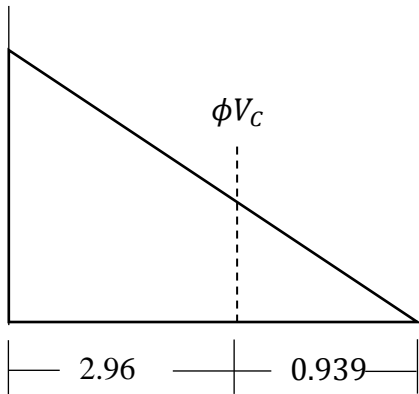
$$\frac{\phi V_C}{2} = \frac{112.27 \text{ kN}}{2} = 56.135 \text{ kN}$$

$$V_U < \phi V_C$$



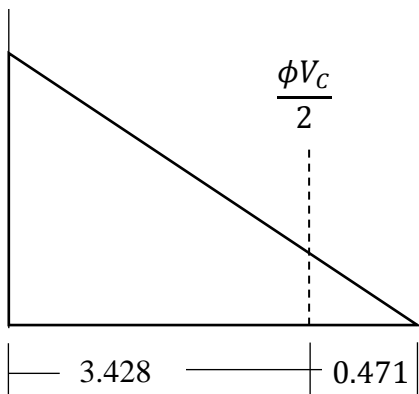
$$x = \frac{365.627 * 3.9}{465.93} = 3.06$$

V (kN)



$$x = \frac{112.27 * 3.9}{465.93} = 0.939$$

V (kN)



$$x = \frac{56.135 * 3.9}{465.93} = 0.471$$

$$s = \frac{\phi * A_v * f_y * d}{(V_U - \phi V_C)}$$

$$s = \frac{0.75 * 0.000064 * 240000 * 0.67}{(465.93 - 112.27)} = 2.182\text{cm}$$

$$s = \frac{A_v * f_y}{0.062 * \sqrt{f'_c} * V_b}$$

$$s = \frac{0.64 * 240}{0.062 * \sqrt{14.1} * 35} = 18.85\text{cm}$$

$$s = \frac{d}{2}$$

$$s = \frac{67}{2} = 33.5 \text{ cm}$$

$$\text{No. Flejes Lmax} = \frac{83.9}{2.182} + 1 = 39.468 \approx 39 \text{ UN}$$

$$\text{No. Flejes Lmed} = \frac{2.121}{2.182} = 97.249 \approx 97 \text{ UN}$$

$$\text{No. Flejes Lbajo} = \frac{46.8}{2.182} = 21.448 \approx 21 \text{ UN}$$

- Refuerzo transversal de la luz del nodo D al nodo E

$$V_U = V_{Max} - W_U(0.2 + d)$$

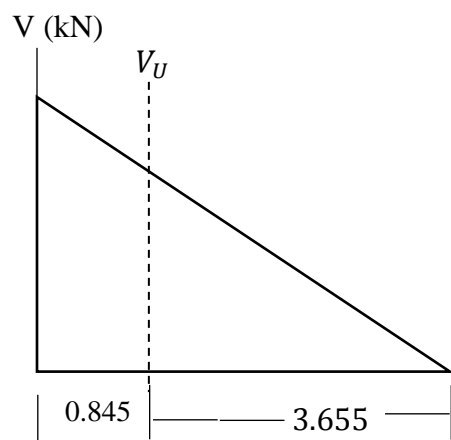
$$V_U = 534.64 - 115.2912 * (0.2 + 0.67) = 434.337 \text{ kN}$$

$$\phi V_c = \phi * 0.17 * \sqrt{f'_c} * b * d$$

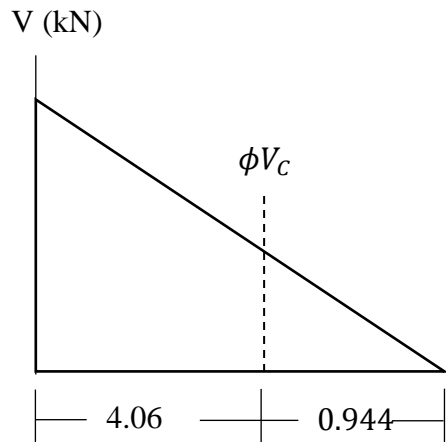
$$\phi V_c = 0.75 * 0.17 * \sqrt{14.1} * 0.35 * 0.67 = 112.27 \text{ kN}$$

$$\frac{\phi V_c}{2} = \frac{112.27 \text{ kN}}{2} = 56.135 \text{ kN}$$

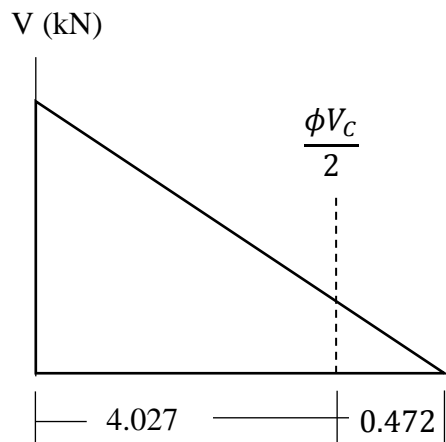
$$V_U < \phi V_c$$



$$x = \frac{434.337 * 4.5}{534.64} = 3.655$$



$$x = \frac{112.27 * 4.5}{534.64} = 0.944$$



$$x = \frac{56.135 * 4.5}{534.64} = 0.472$$

$$s = \frac{\phi * A_v * f_y * d}{(V_u - \phi V_c)}$$

$$s = \frac{0.75 * 0.000064 * 240000 * 0.67}{(534.64 - 112.27)} = 1.827\text{cm}$$

$$s = \frac{A_v * f_y}{0.062 * \sqrt{f'_c} * V_b}$$

$$s = \frac{0.64 * 240}{0.062 * \sqrt{14.1} * 35} = 18.85 \text{ cm}$$

$$s = \frac{d}{2}$$

$$s = \frac{67}{2} = 33.5 \text{ cm}$$

$$\text{No. Flejes } L_{\text{max}} = \frac{84.5}{1.827} + 1 = 47.25 \approx 47 \text{ UN}$$

$$\text{No. Flejes } L_{\text{med}} = \frac{321.5}{1.827} = 175.972 \approx 176 \text{ UN}$$

$$\text{No. Flejes } L_{\text{bajo}} = \frac{3.3}{1.827} = 1.8 \approx 2 \text{ UN}$$

- Refuerzo transversal de la luz del nodo E al nodo F

$$V_U = V_{\text{Max}} - W_U(0.2 + d)$$

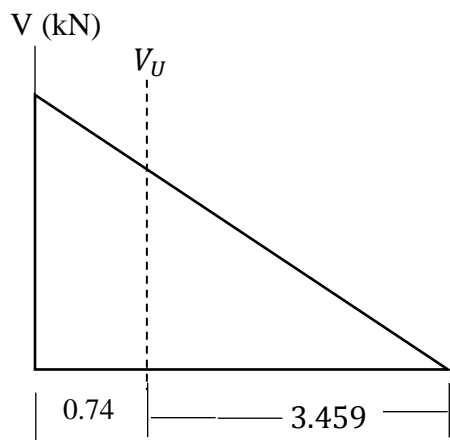
$$V_U = 568.76 - 115.2912 * (0.2 + 0.67) = 468.457 \text{ kN}$$

$$\phi V_C = \phi * 0.17 * \sqrt{f'_c} * b * d$$

$$\phi V_C = 0.75 * 0.17 * \sqrt{14.1} * 0.35 * 0.67 = 112.27 \text{ kN}$$

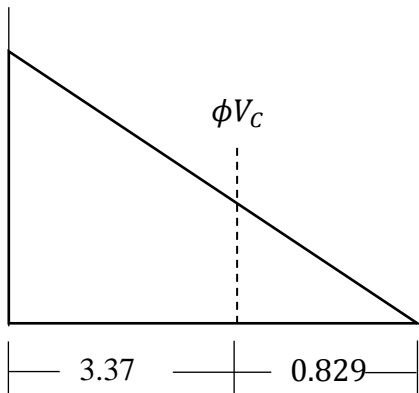
$$\frac{\phi V_C}{2} = \frac{112.27 \text{ kN}}{2} = 56.135 \text{ kN}$$

$$V_U < \phi V_C$$



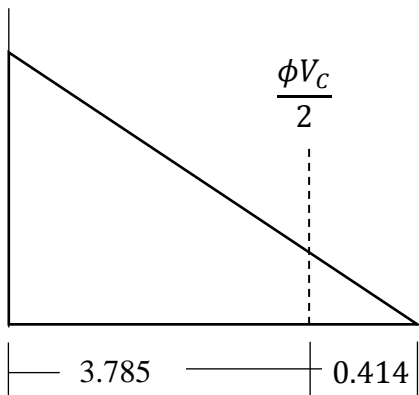
$$x = \frac{468.457 * 4.2}{568.76} = 3.459$$

V (kN)



$$x = \frac{112.27 * 4.2}{568.76} = 0.829$$

V (kN)



$$x = \frac{56.135 * 4.2}{568.76} = 0.414$$

$$s = \frac{\phi * A_v * f_y * d}{(V_u - \phi V_c)}$$

$$s = \frac{0.75 * 0.000064 * 240000 * 0.67}{(568.76 - 112.27)} = 1.690 \text{ cm}$$

$$s = \frac{A_V * f_y}{0.062 * \sqrt{f'_c} * V_b}$$

$$s = \frac{0.64 * 240}{0.062 * \sqrt{14.1} * 35} = 18.85 \text{ cm}$$

$$s = \frac{d}{2}$$

$$s = \frac{67}{2} = 33.5 \text{ cm}$$

$$\text{No. Flejes } L_{\text{max}} = \frac{74}{1.690} + 1 = 44.787 \approx 45 \text{ UN}$$

$$\text{No. Flejes } L_{\text{med}} = \frac{263}{1.690} = 155.621 \approx 156 \text{ UN}$$

$$\text{No. Flejes } L_{\text{bajo}} = \frac{41.5}{1.690} = 24.556 \approx 25 \text{ UN}$$

- Refuerzo transversal de la luz del nodo F al nodo G

$$V_U = V_{\text{Max}} - W_U(0.2 + d)$$

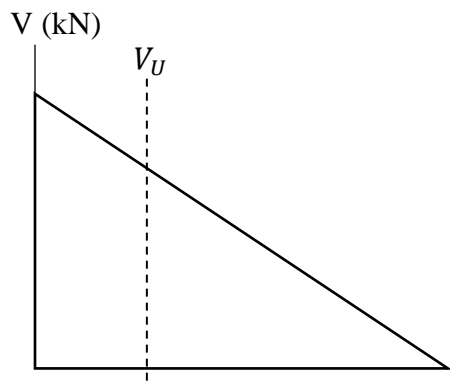
$$V_U = 752.38 - 115.2912 * (0.2 + 0.67) = 652.077 \text{ kN}$$

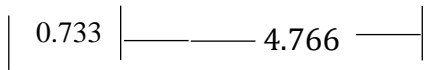
$$\phi V_C = \phi * 0.17 * \sqrt{f'_c} * b * d$$

$$\phi V_C = 0.75 * 0.17 * \sqrt{14.1} * 0.35 * 0.67 = 112.27 \text{ kN}$$

$$\frac{\phi V_C}{2} = \frac{112.27 \text{ kN}}{2} = 56.135 \text{ kN}$$

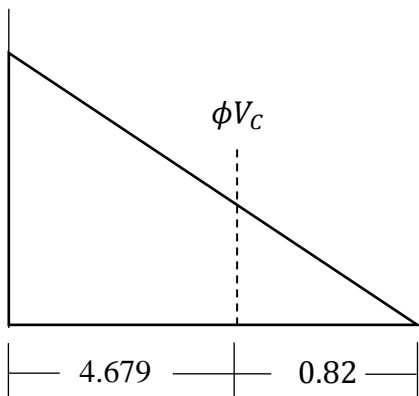
$$V_U < \phi V_C$$





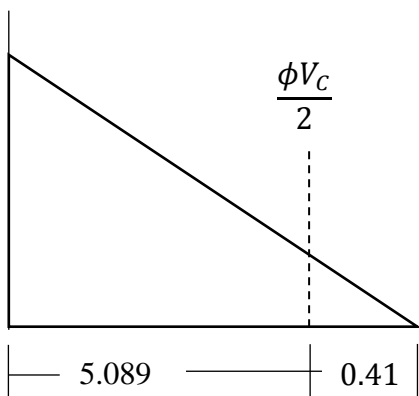
$$x = \frac{652.077 * 5.5}{752.38} = 4.766$$

V (kN)



$$x = \frac{112.27 * 5.5}{752.38} = 0.82$$

V (kN)



$$x = \frac{56.135 * 5.5}{752.38} = 0.41$$

$$s = \frac{\phi * A_V * f_y * d}{(V_U - \phi V_C)}$$

$$s = \frac{0.75 * 0.000064 * 240000 * 0.67}{(752.38 - 112.27)} = 1.205 \text{ cm}$$

$$s = \frac{A_V * f_y}{0.062 * \sqrt{f'_C} * V_b}$$

$$s = \frac{0.64 * 240}{0.062 * \sqrt{14.1} * 35} = 18.85 \text{ cm}$$

$$s = \frac{d}{2}$$

$$s = \frac{67}{2} = 33.5 \text{ cm}$$

$$\text{No. Flejes Lmax} = \frac{73.3}{\frac{1.205}{394.6}} + 1 = 61.829 \approx 62 \text{ UN}$$

$$\text{No. Flejes Lmed} = \frac{394.6}{1.205} = 327.469 \approx 327 \text{ UN}$$

$$\text{No. Flejes Lbajo} = \frac{41}{1.205} = 34.024 \approx 25 \text{ UN}$$

$$0 = 0.9 * 420000 * \rho_{bal} * \left(1 - 0.59 * \frac{\rho_{bal} * 240}{14.1}\right) * 0.35 * 0.67^2$$

$$\rho_{bal} G = 0$$

6. COMPARACIÓN ENTRE KAI SAD Y SAP2000.

En este capítulo se muestra la comparación de los tres ejercicios planteados a lo largo del capítulo 5, entre el programa KAI SAD y SAP2000, para así analizar el rango de error existente entre ambos y poder concluir respecto a esto.

6.1. EJERCICIO DE APLICACIÓN 1 Y SAP 2000

A continuación, se muestran los recortes de pantalla realizados al programa SAP 2000 a partir de la configuración de la viga de este ejercicio en contraste con los gráficos obtenidos en KAI-SAD con la misma configuración de viga, para luego realizar el respectivo rango de error de la comparación de ambos programas, para así poder comprender.

NOTA 13. Para el ejercicio de aplicación 1, con base en las limitaciones planteadas en el documento, la comparación se orientará solamente para el diagrama de fuerza cortante.

6.1.1. Diagramas de Carga.

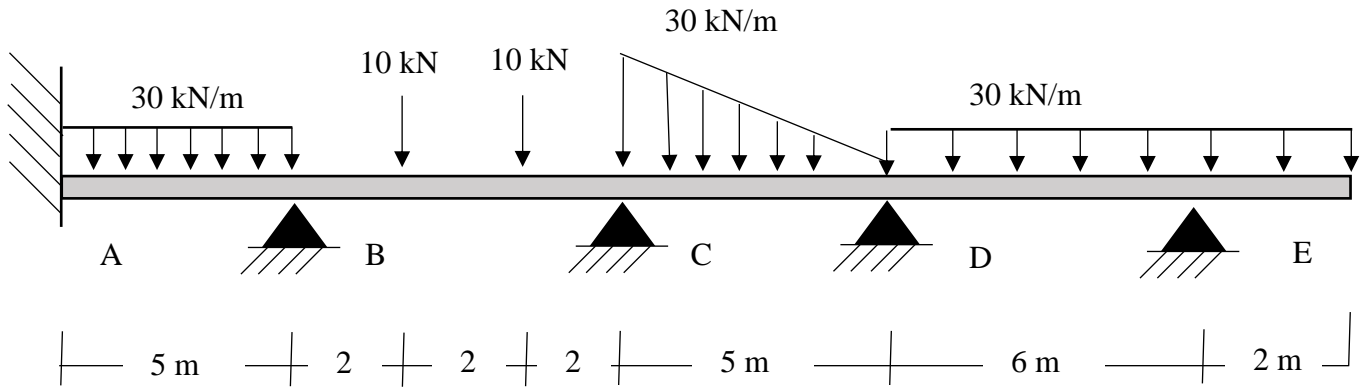


Grafico 9. Diagrama de carga viga hiperestática 1. Fuente: propia.

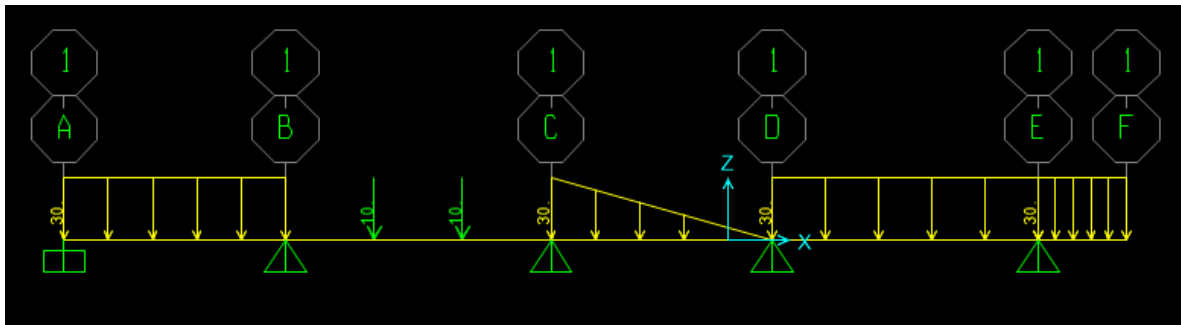


Grafico 23. Recorte de pantalla Diagrama de carga de viga hiperestática 1 en el programa SAP2000. Fuente: propia

6.1.2. Diagramas de Fuerza Cortante.

Grafica1

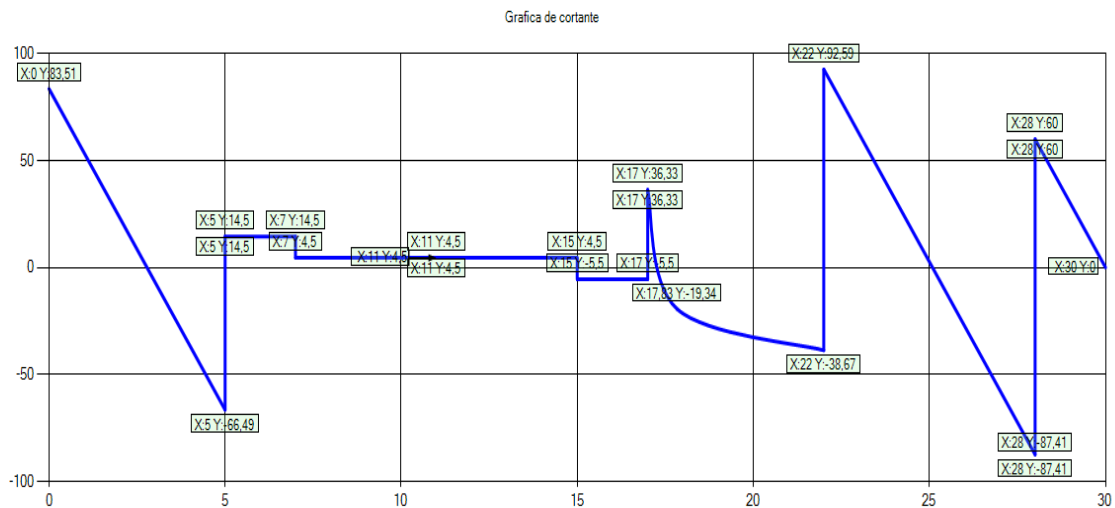


Grafico 13. Diagrama de gráfica cortante de viga hiperestática 1. Fuente: propia

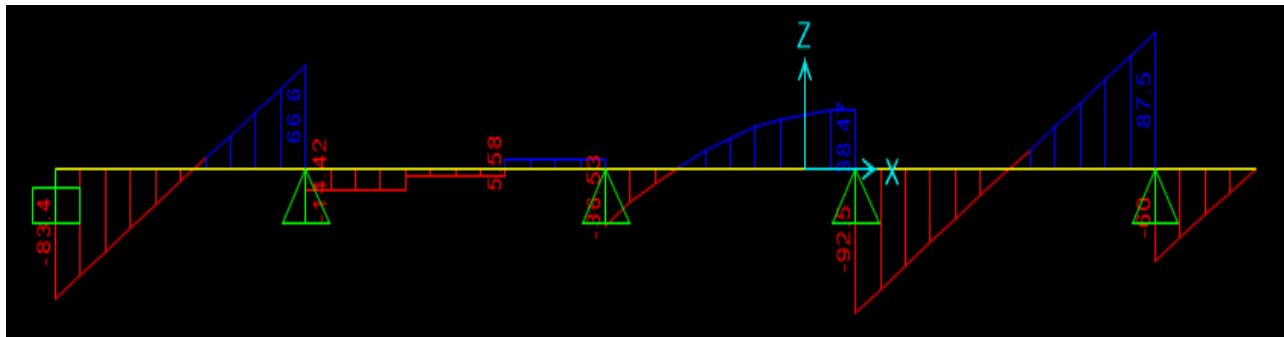
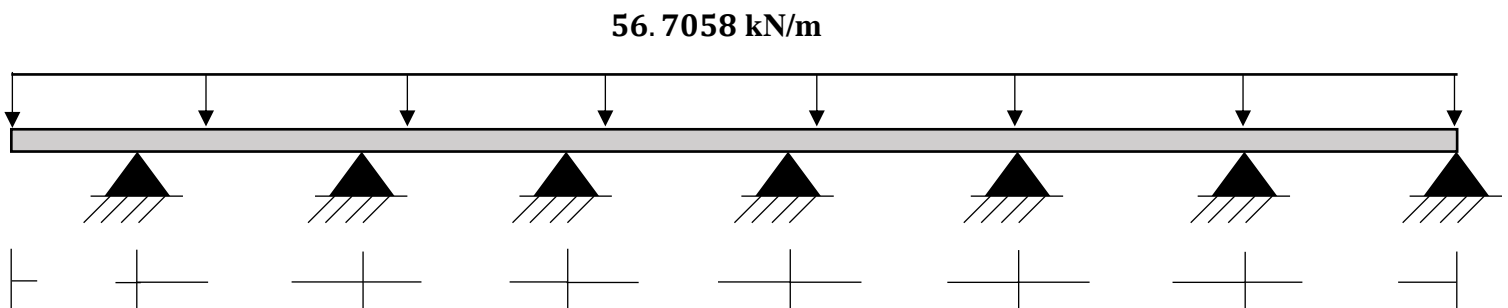


Grafico 24. Recorte de pantalla Diagrama de gráfica cortante de viga hiperestática 1 en el programa SAP2000. Fuente: propia

6.2. EJERCICIO DE APLICACIÓN 2 Y SAP2000

A continuación, se muestran los recortes de pantalla realizados al programa SAP 2000 a partir de la configuración de la viga de este ejercicio en contraste con los gráficos obtenidos en KAI-SAD con la misma configuración de viga.

6.2.1. Diagramas de Carga.



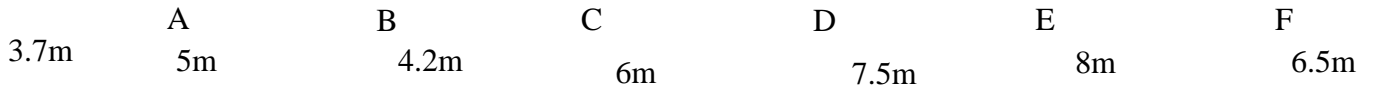
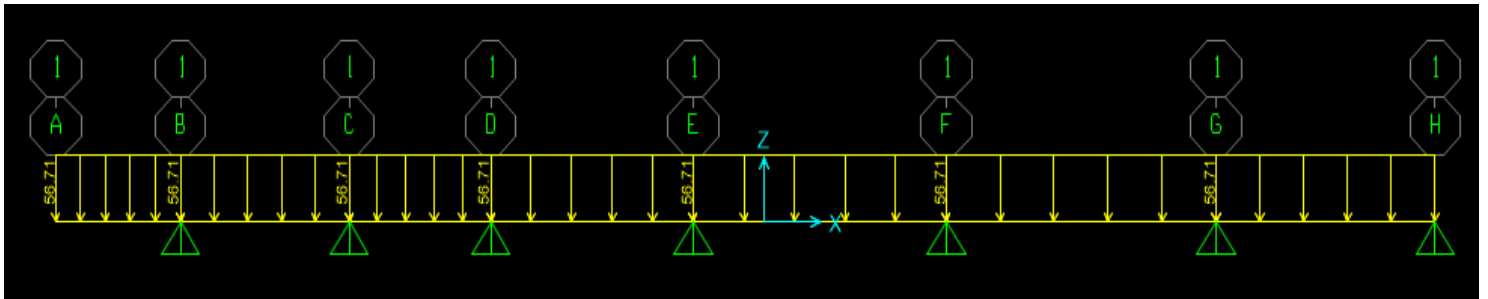
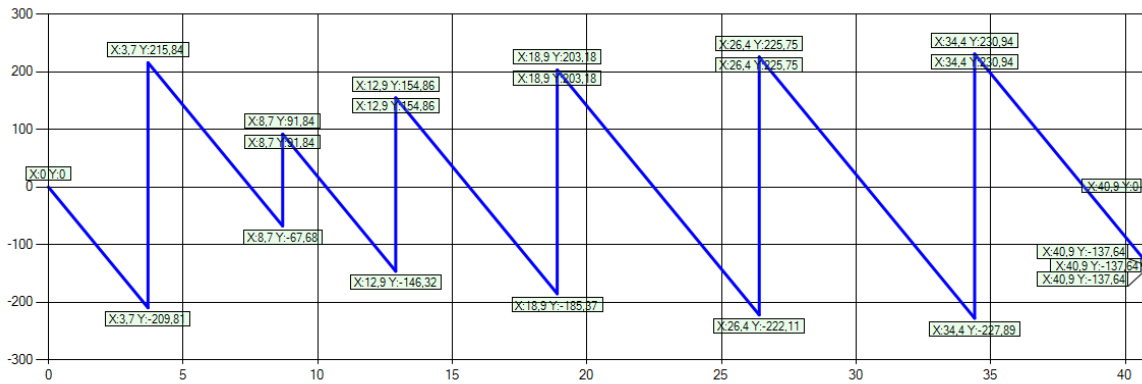


Gráfico 15. Diagrama de carga viga hiperestática 2. Fuente: propia.

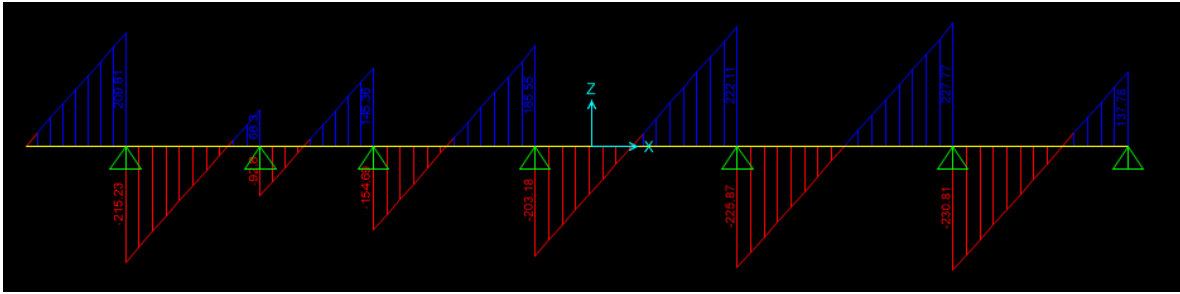


Gráfica 25. Recorte de pantalla Diagrama de carga de viga hiperestática 2 en el programa SAP2000. Fuente: propia

6.2.2. Diagramas de Fuerza Cortante.

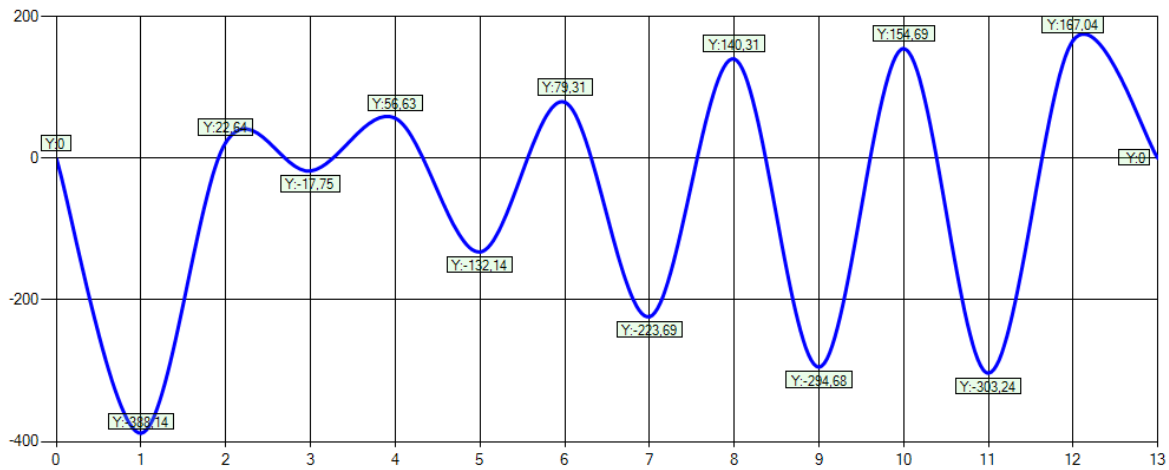


Gráfica 16. Diagrama de gráfica cortante de viga hiperestática 2 programa KAI SAD. Fuente: Propia

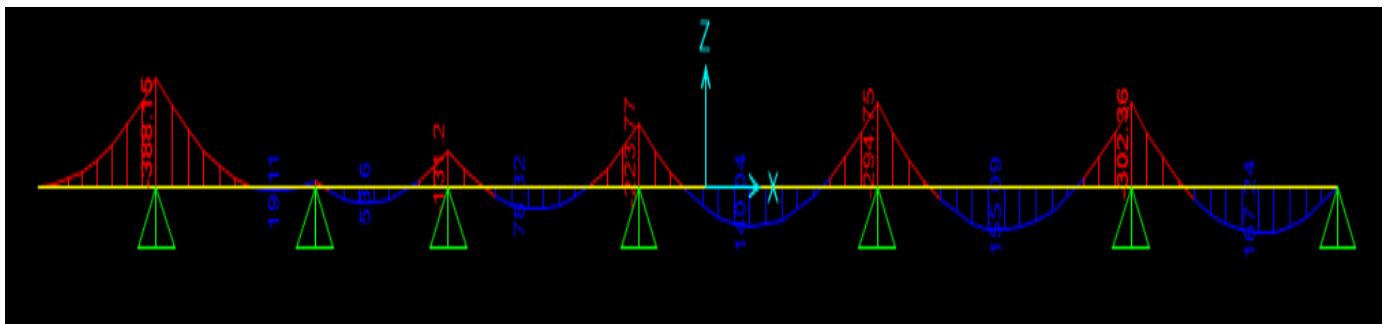


Gráfica 26. Recorte de pantalla Diagrama de gráfica cortante de viga hiperestática 2 en el programa SAP2000.
Fuente: propia

6.2.3. Diagramas de Momentos Flectores.



Gráfica 17. Diagrama de gráfica momento flector de viga hiperestática 2 programa KAI SAD. Fuente: Propia



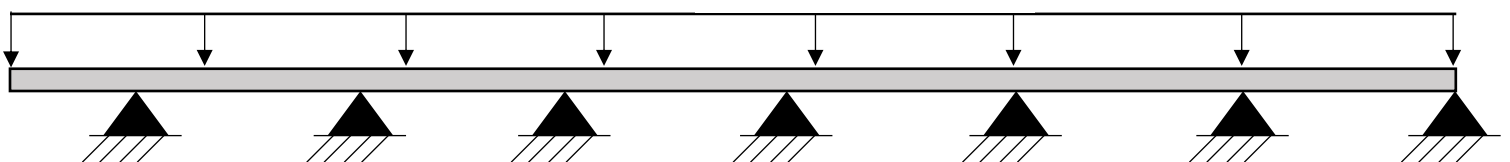
Gráfica 27. Recorte de pantalla Diagrama de gráfica de momento flector de viga hiperestática 2 en el programa SAP2000. Fuente: propia

6.3. EJERCICIO DE APLICACIÓN 3 Y SAP2000

A continuación, se muestran los recortes de pantalla realizados al programa SAP2000 a partir de la configuración de la viga de este ejercicio en contraste con los gráficos obtenidos en KAI SAD con la misma configuración de viga.

6.3.1. Diagramas de Carga.

115.2912 kN/m



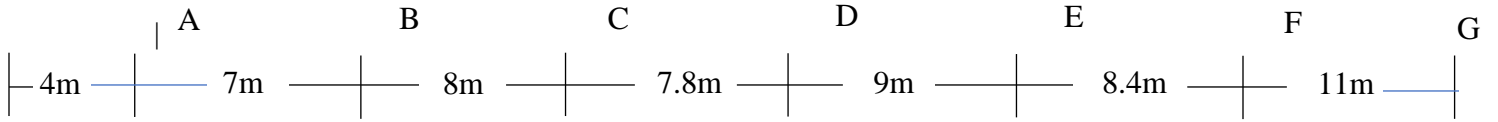
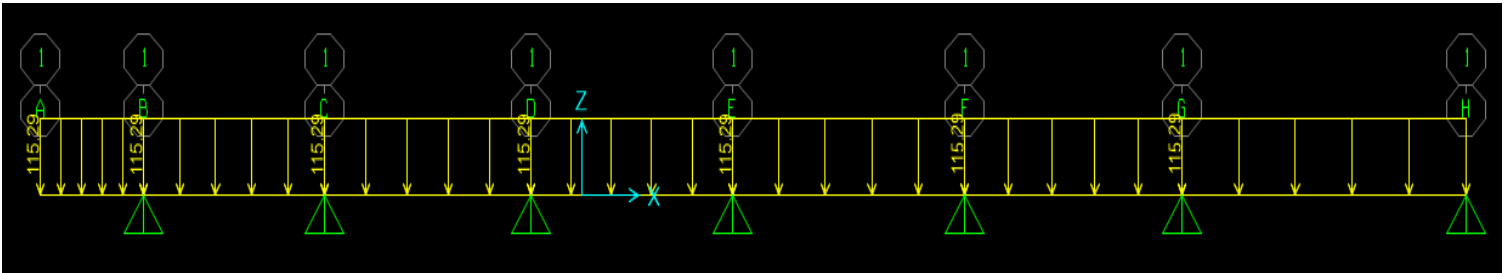


Gráfico 20. Diagrama de carga viga hiperestática 3. Fuente: propia.



Gráfica 28. Recorte de pantalla Diagrama de carga de viga hiperestática 3 en el programa SAP2000. Fuente: propia

6.3.2. Diagramas de Fuerza Cortante.

Grafica1

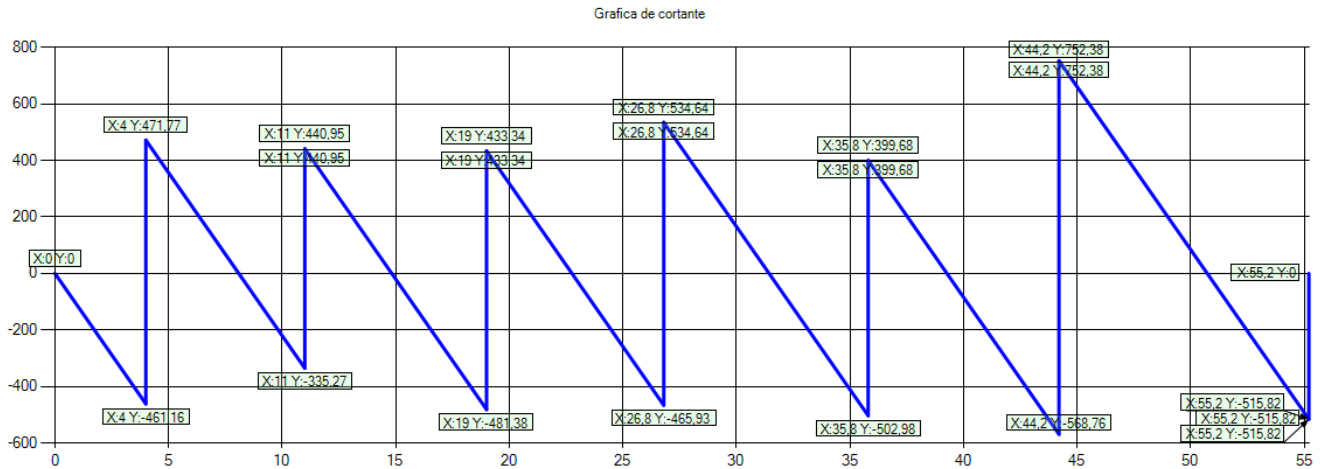
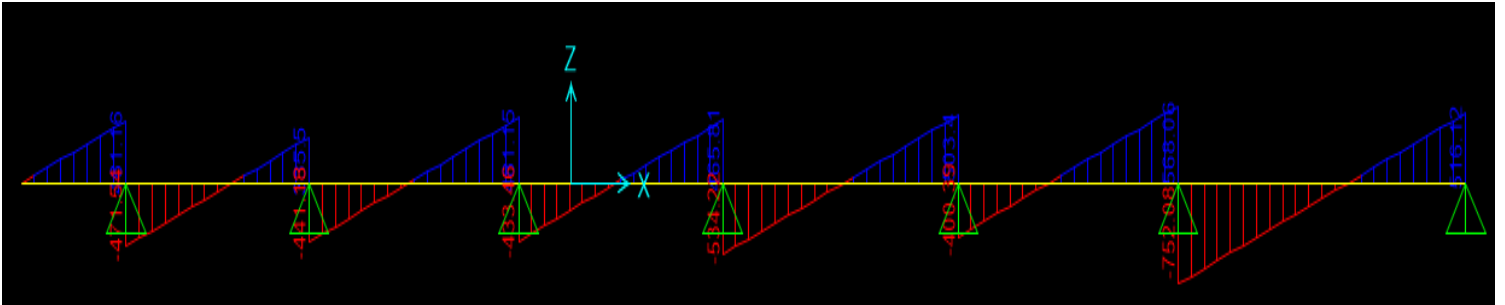


Gráfico 21. Diagrama de fuerza cortante de la viga hiperestática 3 en KAI SAD. Fuente: propia.



Gráfica 29. Recorte de pantalla Diagrama de gráfica cortante de viga hiperestática 3 en el programa SAP2000.
Fuente: propia

6.3.3. Diagramas de Momentos Flectores.

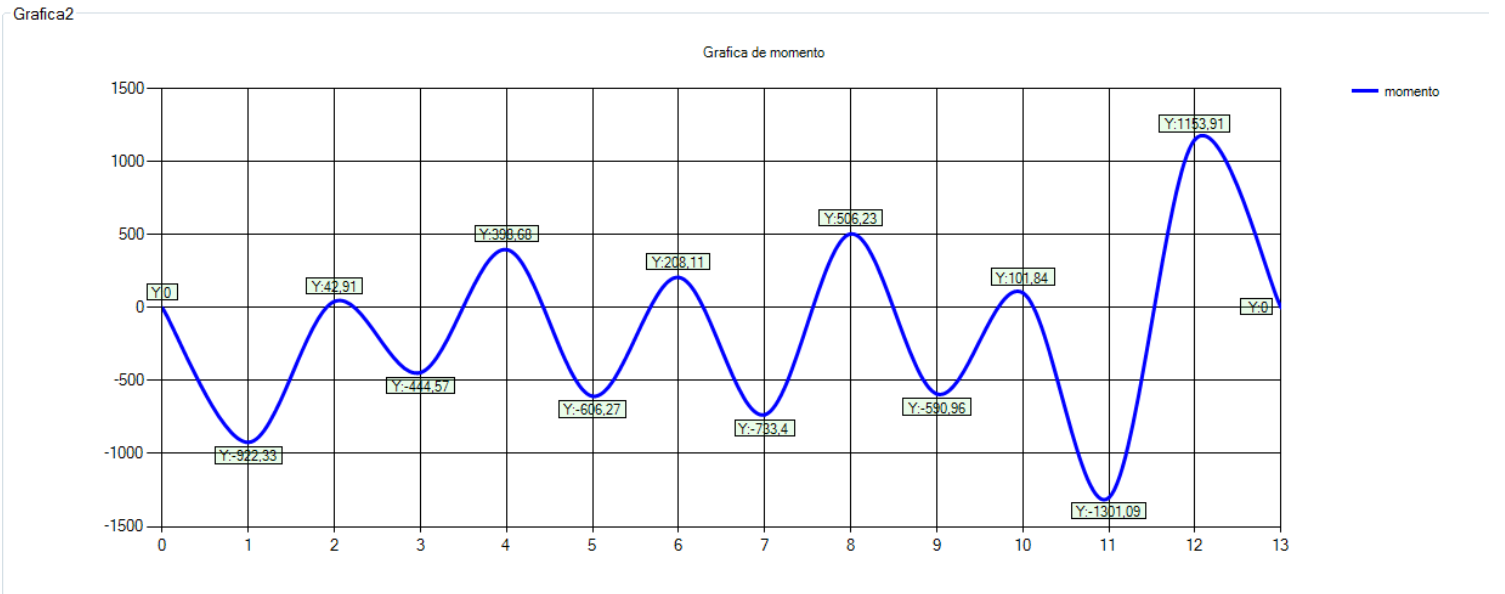
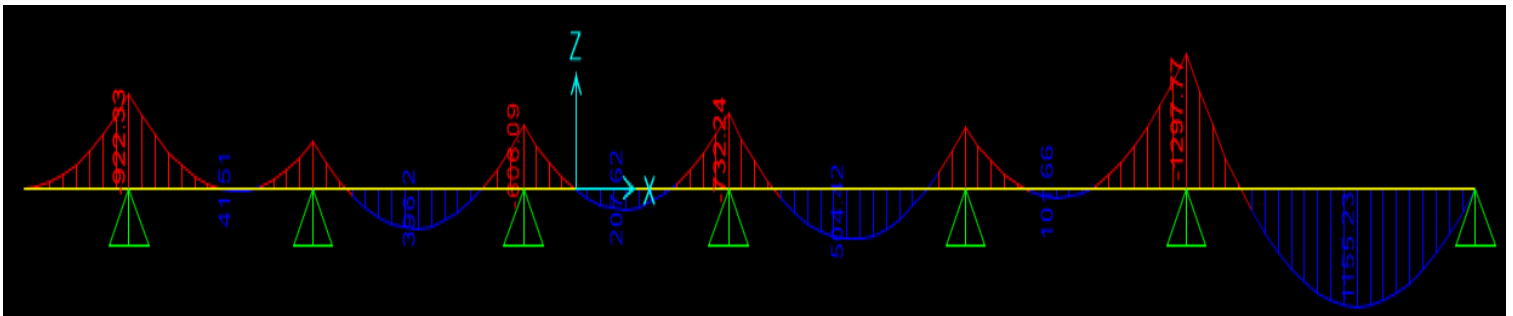


Gráfico 22. Diagrama de momento flector de la viga hiperestática 3 en KAI SAD. Fuente: propia.



Gráfica 30. Recorte de pantalla Diagrama de gráfica de momento flector de viga hiperestática 3 en el programa SAP2000. Fuente: propia

6.3. RANGO DE ERROR

En este subcapítulo se realiza el cálculo del rango de error a partir de los datos obtenidos por ambos programas en los diagramas de momento flector de acuerdo a sus valores máximos, para el ejercicio 2 y 3, para el ejercicio 1 como ya se explicó antes, se realizará el rango de error a partir de los valores obtenidos en los diagramas de fuerza cortante de ambos programas.

Para calcular el % de incertidumbre entre los valores se debe realizar lo siguiente:

E.A: error absoluto

V.E: Valor Experimental: Valores referentes a KAI-SAD

V.T: Valor Teórico: Valores referentes a SAP 2000

$$E.A = |VE - VT|$$

E.R: error relativo:

$$E.R = \frac{E.A}{VT}$$

% de incertidumbre = E.R * 100

Promedio del % de incertidumbre = (sumatoria de % de incertidumbre / número de valores)

6.3.1. Rango de Error Entre el Ejercicio de aplicación 1 en KAI SAD y SAP2000.

Fuerzas cortantes SAP2000	Fuerzas cortantes KAI SAD	Error Absoluto E.A.	Error Relativo E.R	% de incertidumbre	PROMEDIO del % de incertidumbre
83,4	83,51	0,11	0,001318945	0,131894484	0,415 %
66,6	66,49	0,11	0,001651652	0,165165165	
14,42	14,5	0,08	0,00554785	0,554785021	
5,58	5,5	0,08	0,014336918	1,433691756	
36,53	36,33	0,2	0,005474952	0,547495209	
38,4	38,67	0,27	0,00703125	0,703125	
92,5	92,59	0,09	0,000972973	0,097297297	
87,5	87,41	0,09	0,001028571	0,102857143	
60	60	0	0	0	

Tabla 4. Porcentaje del Rango de error del ejercicio de aplicación 1 entre KAI SAD y SAP2000. Fuente: Propia.

La tabla 4 muestra los valores en el diagrama de fuerza cortante obtenidos en los programas de KAI-SAD y SAP 2000, reflejando un porcentaje de error promedio del 0.415 %, considerado como casi nulo, pues si se analiza de manera decimal el error correspondería a 0,00415.

6.3.2. Rango de Error Entre el Ejercicio de aplicación 2 en KAI SAD y SAP2000.

Momentos Maximos de SAP2000	Momentos maximos KAI SAD	Error Absoluto E.A.	Error Relativo E.R	% de incertidumbre	PROMEDIO del % de incertidumbre
388,15	388,14	0,01	2,57632E-05	0,002576324	1,510 %
22,1	22,64	0,54	0,024434389	2,443438914	
19,11	17,75	1,36	0,071166928	7,116692831	
53,6	56,63	3,03	0,056529851	5,652985075	
131,2	132,14	0,94	0,007164634	0,716463415	
78,32	79,31	0,99	0,012640449	1,264044944	
223,77	223,68	0,09	0,000402199	0,040219869	
140,04	140,31	0,27	0,001928021	0,192802057	
294,75	294,68	0,07	0,000237489	0,02374894	
155,09	154,69	0,4	0,002579148	0,257914759	
302,36	303,24	0,88	0,002910438	0,291043789	
167,24	167,04	0,2	0,001195886	0,119588615	

Tabla 5. Porcentaje del Rango de error del ejercicio de aplicación 2 entre KAI SAD y SAP2000. Fuente: Propia.

La tabla 5 muestra los valores en el diagrama de momento flector obtenidos en los programas de KAI-SAD y SAP 2000, reflejando un porcentaje de error promedio del 1,510 %, considerado como casi nulo, pues si se analiza de manera decimal el error correspondería a 0,0151.

6.3.3. Rango de Error Entre el Ejercicio de aplicación 3 en KAI-SAD y SAP 2000.

Momentos Maximos de SAP2000	Momentos maximos KAI SAD	Error Absoluto E.A.	Error Relativo E.R	% de incertidumbre	PROMEDIO del % de incertidumbre
922,33	922,33	0	0	0	0,456 %
41,51	42,91	1,4	0,033726813	3,372681282	
444,4	444,57	0,17	0,000382538	0,038253825	
396,2	398,68	2,48	0,006259465	0,625946492	
606,09	606,27	0,18	0,000296986	0,02969856	
207,62	208,11	0,49	0,002360081	0,236008092	
732,24	733,4	1,16	0,00158418	0,158418005	
504,02	506,23	2,21	0,004384747	0,438474664	
590,8	590,96	0,16	0,000270819	0,027081923	
101,66	101,84	0,18	0,001770608	0,177060791	
1297,77	1301,09	3,32	0,002558235	0,255823451	
1155,23	1153,91	1,32	0,00114263	0,114262961	

Tabla 6. Porcentaje del Rango de error del ejercicio de aplicación 3 entre KAI SAD y SAP2000. Fuente: Propia.

La tabla 6 muestra los valores en el diagrama de momento flector obtenidos en los programas de KAI-SAD y SAP 2000, reflejando un porcentaje de error promedio del 0.456 %, considerado como casi nulo, pues si se analiza de manera decimal el error correspondería a 0,00456.

7. CONCLUSIONES

- Con fundamento en los resultados obtenidos al momento de ejecutar KAI-SAD, con base en las restricciones y conceptos declarados por los autores de este documento en el marco teórico y metodológico, a partir de bibliografías como: “*Diseño básico de concreto reforzado volumen 1 y 2*” cuyo autor es el Ingeniero Jaime Iván Mora Samacá y “*Análisis de estructuras*” de Jairo Uribe Escamilla y basados en el Reglamento Colombiano de Construcción Sismo Resistente de 2010 NSR-10, se llega a la conclusión de que KAI – SAD al ser la primera versión de este software, es apto para ser implementado en las aulas de cátedra para las asignaturas de “Análisis Estructural” y “Estructuras de Concreto”, pues cumple con los lineamientos, conceptos y parámetros que se disponen en cada una de estas asignaturas, con el fin de mejorar la claridad y comprensión en los conceptos impartidos en sus cátedras; no obstante KAI – SAD, representa el origen de lo que podría llegar a ser un software interactivo de ingeniería estructural, si se complementan actualizaciones a esta versión.
- KAI SAD a partir de la necesidad de reflejar parámetros claves para la investigación, tales como los diagramas de fuerza cortante y valores de gráfico momento, de las posibles configuraciones de vigas dentro de los parámetros definidos anteriormente en la justificación, con base en las restricciones y normas, creó dos interfaces dinámicas para el desarrollo de “Análisis estructural” y “Asignación de aceros”, donde se reflejan los valores de las áreas de aceros a cortante y a flexión, que cumplen con la exigencia de todos estos parámetros y que permite inferir y concluir que son funcionales, pues las realiza sin problema, sin embargo existen restricciones que limitan estas funciones pero que se pueden solucionar por medio de la implementación de actualizaciones.

- Lo anteriormente concluido, permite deducir que al ser un software funcional que abarca las cátedras de análisis estructural y estructuras de concreto, para sus temas y conceptos básicos, se puede implementar permitiendo concluir que, para las generaciones futuras a partir del momento de su integración, tendrá una repercusión positiva en la comprensión y uso de las tecnologías de la información y comunicación TIC, pues estas son el futuro y su comprensión es clave para el desarrollo estudiantil y en general ya que es una necesidad que se ha venido dando tanto para alumnos, como para los docentes.
- Al realizar el análisis de la comparación entre los softwares: KAI-SAD desarrollado con las bases de la metodología de aproximación de Hardy Cross, explicada en este informe y en contraste con SAP 2000 cuya metodología de trabajo se basa en el método matricial, con fundamento en sus diagramas de fuerza cortante y momento flector, teniendo en cuenta sus limitaciones y realizando los promedios de los porcentajes de error obtenidos a partir de los valores de fuerza cortante y momentos flectores máximos, se concluye que al ser un porcentaje de error que no supera el 1 % traducido como 0,01 en orden decimal, al realizar el promedio de los promedios del % de incertidumbre, este es casi nulo, por lo tanto KAI SAD, se define como un programa funcional con rango de error mínimo y listo para su implementación en las aulas, también con vistas a seguirle acreditando actualizaciones que garanticen cada vez más la perfección y efectividad del mismo.

8. BIBLIOGRAFÍA

- Mora, J. I. (2019). *Diseño Básico de Concreto Reforzado Vol. 2*. Bogotá: Corporación Universidad Piloto de Colombia.
- NSR-10. (2010). *Reglamento de Construcción Sismo Resistente de 2010*. Obtenido de <https://www.idrd.gov.co/sitio/idrd/sites/default/files/imagenes/2titulo-b-nsr-100.pdf>
- Awad, R. R. (1999). *Hormigón Reforzado*. Medellín : Editorial Universidad Pontificia Bolivariana .
- Berrocal, L. (1990). *Resistencia de los Materiales*. Madrid España : Mc Graw Hill .
- Cuevas, Ó. M. (2003). *Análisis Estructural*. Ciudad de México : Noruega Editores .
- Hibbeler, R. C. (2010). *Ingeniería Mecánica - Estática Décimosegunda Edición*. México: Pearson Educación .
- IngMario. (s.f.). *el rincón del ingeniero*. Obtenido de www.elrincondelingeniero.com/Cargas+triangulares+en+vigas
- Jairo, U. E. (2002). *Análisis de Estructura*. Bogotá : Editorial Escuela Colombiana De Ingeniería .
- Jr, B. a. (1992). *Estática y Mecánica de los Materiales*. McGraw-Hill.
- Montero, L. V. (2004). *Diseño de un edificio con muros de carga y cortante*. Ciudad de México.
- Mora, J. I. (2019). *Diseño Básico de Concreto Reforzado Vol. 2*. Bogotá: Corporación Universidad Piloto de Colombia.
- NSR-10. (2010). *Reglamento de Construcción Sismo Resistente de 2010*. Obtenido de <https://www.idrd.gov.co/sitio/idrd/sites/default/files/imagenes/2titulo-b-nsr-100.pdf>.
- Pérez, R. L. (1990). *Léxico de Arte*. Madrid : Akal .
- Samacá, J. I. (2014). *Diseño Básico de Concreto Reforzado*. Bogotá : Editorial Universidad Piloto de Colombia .
- Thornton, M. &. (1995). *Classical Dynamics of Particles and Systems*. Harcourt Brace & Company.
- Trujillo, J. E. (2007). *MECÁNICA BÁSICA PARA ESTUDIANTES DE INGENIERÍA*. Manizales: Universidad Nacional de Colombia .
- www.parro.com.ar. (s.f.). Obtenido de Diccionario de Arquitectura y Construcción: www.parro.com.ar

9. ANEXOS.

9.1. CODIFICACIÓN DE KAI – SAD

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace KAI_SAD
{
    static class Program
    {
        /// <summary>
        /// Punto de entrada principal para la aplicación.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Main());
        }
    }
}

namespace KAI_SAD
{
    public partial class Main: DevExpress.XtraBars.Ribbon.RibbonForm
    {
        double puntox, puntoy, puntomomentox = 0;
        int Acolum = 0;
        string numerador, denominador;
        DataTable dtreunion, dtoficinas, dteducativos, dtbibliotecas, dtfabricas, dtinstitucional,
dtcomercio, dtresidencial, dtalmacenamiento, dtgarajes, dtcoliseo;
        DataTable dtcieloraso, dtrellenopiso, dtpisosyacabados, dtcubierta, dtrecubrimiento,
dtparticioneslivi, dtenchape, dtventanas, dtmuros;
        double Rarea1_2, Rarea2_1, Rarea2_2, Rarea3_1, Rarea3_2, Rarea4_1, Rarea4_2, Rarea5_1,
Rarea5_2, Rarea6_1, Rarea6_2, Rarea7_1, Rarea7_2, Rarea8_1, Rarea8_2, Rarea9_1, Rarea9_2;
        double PuntoCortanteY1;
        double PuntoCortanteX1;
        double NL = 1;
        double Luz11a1;
        double Luz11b1;
        double Luz21a1;
        double Luz21b1;
        double Luz31a1;
        double Luz31b1;
        double Luz41a1;
        double Luz41b1;
        double Luz51a1;
        double Luz51b1;
        double Luz61a1;
        double Luz61b1;
        double Luz71a1;
        double Luz71b1;
        double Luz81a1;
        double Luz81b1;
        double Luz91a1;
        double Luz91b1;

        double Luz11a2;
        double Luz11b2;
    }
}
```

double Luz21a2;
double Luz21b2;
double Luz31a2;
double Luz31b2;
double Luz41a2;
double Luz41b2;
double Luz51a2;
double Luz51b2;
double Luz61a2;
double Luz61b2;
double Luz71a2;
double Luz71b2;
double Luz81a2;
double Luz81b2;
double Luz91a2;
double Luz91b2;

double Luz11a3;
double Luz11b3;
double Luz21a3;
double Luz21b3;
double Luz31a3;
double Luz31b3;
double Luz41a3;
double Luz41b3;
double Luz51a3;
double Luz51b3;
double Luz61a3;
double Luz61b3;
double Luz71a3;
double Luz71b3;
double Luz81a3;
double Luz81b3;
double Luz91a3;
double Luz91b3;

double Luz11a4;
double Luz11b4;
double Luz21a4;
double Luz21b4;
double Luz31a4;
double Luz31b4;
double Luz41a4;
double Luz41b4;
double Luz51a4;
double Luz51b4;
double Luz61a4;
double Luz61b4;
double Luz71a4;
double Luz71b4;
double Luz81a4;
double Luz81b4;
double Luz91a4;
double Luz91b4;

double Luz11a5;
double Luz11b5;
double Luz21a5;
double Luz21b5;
double Luz31a5;
double Luz31b5;
double Luz41a5;
double Luz41b5;
double Luz51a5;
double Luz51b5;
double Luz61a5;
double Luz61b5;
double Luz71a5;
double Luz71b5;

double Luz81a5;
double Luz81b5;
double Luz91a5;
double Luz91b5;

double Luz11a6;
double Luz11b6;
double Luz21a6;
double Luz21b6;
double Luz31a6;
double Luz31b6;
double Luz41a6;
double Luz41b6;
double Luz51a6;
double Luz51b6;
double Luz61a6;
double Luz61b6;
double Luz71a6;
double Luz71b6;
double Luz81a6;
double Luz81b6;
double Luz91a6;
double Luz91b6;

double LT1Luz1;
double LT1Luz2;
double LT1Luz3;
double LT1Luz4;
double LT1Luz5;
double LT1Luz6;
double LT1Luz7;
double LT1Luz8;
double LT1Luz9;
double LT2Luz1;
double LT2Luz2;
double LT2Luz3;
double LT2Luz4;
double LT2Luz5;
double LT2Luz6;
double LT2Luz7;
double LT2Luz8;
double LT2Luz9;
double LT3Luz1;
double LT3Luz2;
double LT3Luz3;
double LT3Luz4;
double LT3Luz5;
double LT3Luz6;
double LT3Luz7;
double LT3Luz8;
double LT3Luz9;
double LT4Luz1;
double LT4Luz2;
double LT4Luz3;
double LT4Luz4;
double LT4Luz5;
double LT4Luz6;
double LT4Luz7;
double LT4Luz8;
double LT4Luz9;
double LT5Luz1;
double LT5Luz2;
double LT5Luz3;
double LT5Luz4;
double LT5Luz5;
double LT5Luz6;
double LT5Luz7;
double LT5Luz8;
double LT5Luz9;


```

double pmin = 0;
double pmax = 0;
double M1 = 0;
double M2 = 0;
double A1S1 = 0;
double A1S2 = 0;

/// <summary>
/// variables de calculo para MF
/// </summary>
double MFRL = 0;
double MFTIL = 0;
double MFTDL = 0;
double MFCL = 0;
double MFPIL = 0;
double MFPDL = 0;

double[] MFLP = { 0, 0, 0, 0, 0, 0, 0, 0, 0 };
double[] MFLN = { 0, 0, 0, 0, 0, 0, 0, 0, 0 };

/// <summary>
/// variables de tipo de luz
/// </summary>
double Luz1Carga1;
double L1FP1;
double Luz2Carga1;
double L2FP1;
double Luz3Carga1;
double L3FP1;
double Luz4Carga1;
double L4FP1;
double Luz5Carga1;
double L5FP1;
double Luz6Carga1;
double L6FP1;
double Luz7Carga1;
double L7FP1;
double Luz8Carga1;
double L8FP1;
double Luz9Carga1;
double L9FP1;

double Luz1Carga2;
double L1FP2;
double Luz2Carga2;
double L2FP2;
double Luz3Carga2;
double L3FP2;
double Luz4Carga2;
double L4FP2;
double Luz5Carga2;
double L5FP2;
double Luz6Carga2;
double L6FP2;
double Luz7Carga2;
double L7FP2;
double Luz8Carga2;
double L8FP2;
double Luz9Carga2;
double L9FP2;

double Luz1Carga3;
double L1FP3;
double Luz2Carga3;
double L2FP3;
double Luz3Carga3;
double L3FP3;
double Luz4Carga3;
double L4FP3;

```

```
double Luz5Carga3;  
double L5FP3;  
double Luz6Carga3;  
double L6FP3;  
double Luz7Carga3;  
double L7FP3;  
double Luz8Carga3;  
double L8FP3;  
double Luz9Carga3;  
double L9FP3;
```

```
double Luz1Carga4;  
double L1FP4;  
double Luz2Carga4;  
double L2FP4;  
double Luz3Carga4;  
double L3FP4;  
double Luz4Carga4;  
double L4FP4;  
double Luz5Carga4;  
double L5FP4;  
double Luz6Carga4;  
double L6FP4;  
double Luz7Carga4;  
double L7FP4;  
double Luz8Carga4;  
double L8FP4;  
double Luz9Carga4;  
double L9FP4;
```

```
double Luz1Carga5;  
double L1FP5;  
double Luz2Carga5;  
double L2FP5;  
double Luz3Carga5;  
double L3FP5;  
double Luz4Carga5;  
double L4FP5;  
double Luz5Carga5;  
double L5FP5;  
double Luz6Carga5;  
double L6FP5;  
double Luz7Carga5;  
double L7FP5;  
double Luz8Carga5;  
double L8FP5;  
double Luz9Carga5;  
double L9FP5;
```

```
double Luz1Carga6;  
double L1FP6;  
double Luz2Carga6;  
double L2FP6;  
double Luz3Carga6;  
double L3FP6;  
double Luz4Carga6;  
double L4FP6;  
double Luz5Carga6;  
double L5FP6;  
double Luz6Carga6;  
double L6FP6;  
double Luz7Carga6;  
double L7FP6;  
double Luz8Carga6;  
double L8FP6;  
double Luz9Carga6;  
double L9FP6;
```

```
double TCVI = 0;
```

```

double TCVF = 0;
double LTVI = 0;
double LTVF = 0;
double LaVI = 0;
double LbVI = 0;
double FPVI = 0;
double LaVF = 0;
double LbVF = 0;
double FPVF = 0;

/// <summary>
/// variables de inercia
/// </summary>
double IV = 0;
double IC = 0;
double IT = 0;
double[] K = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
double LK = 0;

double MPVI = 0;
double MPVF = 0;

double Recubrimiento = 3;
double BV = 0;
double HV = 0;
double BC = 0;
double HC = 0;

/// <summary>
/// variables para el numero de cargas
/// </summary>
double NCL1;
double NCL2;
double NCL3;
double NCL4;
double NCL5;
double NCL6;
double NCL7;
double NCL8;
double NCL9;

/// <summary>
/// Cross
/// </summary>
double ApoyoColumna = 0;
double ApoyoColumnaRab;
double ApoyoColumnaRba;
double ApoyoColumnaRbc;
double ApoyoColumnaRcb;
double ApoyoColumnaRcd;
double ApoyoColumnaRdc;
double ApoyoColumnaRde;
double ApoyoColumnaRed;
double ApoyoColumnaRef;
double ApoyoColumnaRfe;
double ApoyoColumnaRfg;
double ApoyoColumnaRgf;
double ApoyoColumnaRgh;
double ApoyoColumnaRhg;
double ApoyoColumnaRhi;
double ApoyoColumnaRih;
double ApoyoColumnaRij;
double ApoyoColumnaRji;

/// <summary>
/// w1
/// </summary>
double e_sup = 0;
double e_inf = 0;

```

```

double H_libre = 0;
double Laf = 0;
double[] WdSum = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
double[] WlSum = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
double x = 0;

string Wlstring;

string[] REUNION = { "*Ninguna", "Balcones", "Corredores y escaleras", "Silleteria fija",
"Gimnasios", "Vestibulos", "Silleteria movil", "Areas recreativas", "Plataformas", "Escenarios" };
double[] valueREUNION = { 0, 5, 5, 3, 5, 5, 5, 5, 5, 7.5 };

string[] OFICINAS = { "*Ninguna", "Corredores y escaleras", "Oficinas", "Restaurantes" };
double[] valueOFICINAS = { 0, 3, 2, 5 };

string[] EDUCATIVOS = { "*Ninguna", "Salon de clases", "Corredores y escaleras" };
double[] valueEDUCATIVOS = { 0, 2, 5 };

string[] BIBLIOTECAS = { "*Ninguna", "Salon de lectura", "Estanterias" };
double[] valueBIBLIOTECAS = { 0, 2, 7 };

string[] FABRICAS = { "*Ninguna", "Industrias livianas", "Industrias pesadas" };
double[] valueFABRICAS = { 0, 5, 10 };

string[] INSTITUCIONAL = { "*Ninguna", "Cuartos de cirugia, laboratorios", "Cuartos privados",
"Corredores y escaleras" };
double[] valueINSTITUCIONAL = { 0, 4, 2, 5 };

string[] COMERCIO = { "*Ninguna", "Minorista", "Mayorista" };
double[] valueCOMERCIO = { 0, 5, 6 };

string[] RESIDENCIAL = { "*Ninguna", "Balcones", "Cuartos privados y sus corredores",
"Escaleras" };
double[] valueRESIDENCIAL = { 0, 5, 1.8, 3 };

string[] ALMACENAMIENTO = { "*Ninguna", "Liviano", "Pesado" };
double[] valueALMACENAMIENTO = { 0, 6, 12 };

string[] GARAJES = { "*Ninguna", "Garajes para automoviles de pasajeros", "garajes para
vehiculos de carga de hasta 2000 kg de capacidad" };
double[] valueGARAJES = { 0, 2.5, 5 };

string[] COLISEOYESTADIOS = { "*Ninguna", "Graderias", "Escaleras" };
double[] valueCOLISEOYESTADIOS = { 0, 5, 5 };

string[] CUBIERTASDECARGAMINIMA = { "*Ninguna", "Cubierta, azoteas y terrazas", "Cubiertas
usadas para jardines de cubiertas o para reuniones", "Cubiertas inclinadas con mas de 15° de pendiente
en estructura metalica o de madera", "Cubiertas inclinadas con pendiente de 15° o menos en estructura
metalica o de madera" };
double[] valueCUBIERTASDECARGAMINIMA = { 0, 1.8, 5, 0.35, 0.5 };

string[] comboBoxWlAdd = { "REUNION", "OFICINAS", "EDUCATIVOS", "BIBLIOTECAS", "FABRICAS",
"INSTITUCIONAL", "COMERCIO", "RESIDENCIAL", "ALMACENAMIENTO", "GARAJES", "COLISEO Y ESTADIOS" };

string Wdstring;

string[] CIELORASO = { "*Ninguna", "Canales suspendidas de acero", "Ductos mecanicos",
"Entramado metalico suspendido afinado en cemento", "Entramado metalico suspendido afinado en yeso",
"Fibras acusticas", "Pañete en yeso o concreto", "Pañete en entramado de madera", "Tableros de yeso",
"Sistema de suspension de madera" };
double[] valueCIELORASO = { 0, 0.10, 0.20, 0.70, 0.50, 0.10, 0.25, 0.80, 0.0080, 0.15 };

string[] RELLENODEPISO = { "*Ninguna", "Arena", "Concreto con escoria", "Concreto con piedra",
"Concreto ligero" };
double[] valueRELLENODEPISO = { 0, 0.015, 0.0200, 0.0250, 0.0150 };

```

```
string[] PISOSYACABADOS = { "*Ninguna", "Acabado de piso en concreto", "Afinado (25mm) sobre concreto de agregado petreo", "Baldosa ceramica (20mm) sobre 12 mm de mortero", "Baldosa ceramica (20mm) sobre 25 mm de mortero", "Baldosa sobre 25 mm de mortero", "Bloque de asfalto (50mm), sobre 12 mm de mortero", "Bloque de madera (75mm) sin relleno", "Bloque de madera (75mm) sobre 12 mm de mortero", "Durmientes de madera, 20 mm", "Madera densa, 25 mm", "Marmol y mortero sobre concreto de agregado petreo", "Piso asfaltico o linoleo, 6mm", "Pizarra", "Terrazzo (25 mm), concreto 50 mm", "Terrazzo (40 mm) directamente sobre la losa", "Terrazzo (25mm) sobre afinado en concreto" };
```

```
double[] valuePISOSYACABADOS = { 0, 0.0200, 1.50, 0.80, 1.10, 1.10, 1.50, 0.50, 0.80, 0.15, 0.20, 1.60, 0.05, 0.0300, 1.50, 0.90, 1.50 };
```

```
string[] CUBIERTA = { "*Ninguna", "Cobre o laton", "Fibra de vidrio", "Tablero de fibra", "Perlita", "Espuma de poliestireno", "Espuma depoliuretano", "Cubiertas corrugadas de asbesto - cemento", "Entablado de madera", "Laminas de yeso, 12mm", "Madera laminada (según el espesor)", "Bituminosa, cubierta de grava", "Bituminosa, superficie lisa", "Liquido aplicado", "Tela asfaltica de una capa", "Marquesinas, marco metalico, vidrio de 10 mm", "Tableros defibra, 12 mm", "Tableros de madera, 50 mm", "Tableros de madera, 75 mm", "Tablero metalico, calibre 20 (0,9 mm de espesor nominal)", "Tablero metalico, calibre 18 (1,2 mm de espesor nominal)", "Tablillas (shingles) de asbesto - cemento", "Tablillas (shingles) de asfalto", "Tablillas (shingles) de madera", "Teja de arcilla, incluyendo el mortero" };
```

```
double[] valueCUBIERTA = { 0.05, 0.0020, 0.0030, 0.0015, 0.0005, 0.0010, 0.20, 0.0060, 0.10, 0.0100, 0.25, 0.10, 0.05, 0.03, 0.40, 0.05, 0.25, 0.40, 0.08, 0.08, 0.20, 0.10, 0.15, 0.80 };
```

```
string[] RECUBRIMIENTODEMUROS = { "*Ninguna", "Baldosin de cemento", "Entablado de madera", "Madera laminada (según el espesor)", "Espuma de poli estireno", "Espuma de poliuretano", "fibra o acrilico", "Terlita", "Tablerosdefibra", "Tablero de fibra, 12 mm", "Tableros de yeso, 12 mm" };
```

```
double[] valueRECUBRIMIENTODEMUROS = { 0, 0.80, 0.0060, 0.0100, 0.0005, 0.0010, 0.0020, 0.0015, 0.0030, 0.05, 0.10 };
```

```
string[] PARTICIONESLIVIANAS = { "*Ninguna", "Particiones moviles de acero (altura parcial)", "Particiones moviles de acero (altura total)", "Poste en madera o acero, yeso de 12 mm a cada lado", "Poste en madera, 50 x 100, sin pañetar", "Poste en madera, 50 x 100, pañete por un lado", "Poste en madera, 50 x 100, pañete por ambos lados" };
```

```
double[] valuePARTICIONESLIVIANAS = { 0, 0.50, 0.20, 0.90, 0.30, 0.60, 2.00 };
```

```
string[] ENCHAPE = { "*Ninguna", "Enchape ceramico", "Enchapeen arenisca", "Enchape en caliza", "Enchape en granito" };
```

```
double[] valueENCHAPE = { 0, 0.015, 0.013, 0.015, 0.017 };
```

```
string[] VENTANAS = { "*Ninguna", "Muros cortina de vidrio, entramado y marco", "Ventanas, vidrio, entramado y marco" };
```

```
double[] valueVENTANAS = { 0, 0.5, 0.45 };
```

```
string[] MUROS = { "*Ninguna", "Yeso de 15mm, aislado, entablado de 10mm", "Exteriores con enchape en ladrillo", "Pañetado en ambas caras 100", "Pañetado en ambas caras 150", "Pañetado en ambas caras 200", "Pañetado en ambas caras 250", "Pañetado en ambas caras 300", "Sin pañetar 100", "Sin pañetar 150", "Sin pañetar 200", "Sin pañetar 250", "Sin pañetar 300", "Sin Relleno 100", "Sin Relleno 150", "Sin Relleno 200", "Sin Relleno 250", "Sin Relleno 300", "Relleno cada 1.2m 150", "Relleno cada 1.2m 200", "Relleno cada 1.2m 250", "Relleno cada 1.2m 300", "Relleno cada 1.0m 150", "Relleno cada 1.0m 200", "Relleno cada 1.0m 250", "Relleno cada 1.0m 300", "Relleno cada 0.8m 150", "Relleno cada 0.8m 200", "Relleno cada 0.8m 250", "Relleno cada 0.8m 300", "Relleno cada 0.6m 150", "Relleno cada 0.6m 200", "Relleno cada 0.6m 250", "Relleno cada 0.6m 300", "Relleno cada 0.4m 150", "Relleno cada 0.4m 200", "Relleno cada 0.4m 250", "Relleno cada 0.4m 300", "Todas de celdas llenas 150", "Todas de celdas llenas 200", "Todas de celdas llenas 250", "Todas de celdas llenas 300", "Sin pañetar con arcilla 100", "Sin pañetar con arcilla 150", "Sin pañetar con arcilla 200", "Sin pañetar con arcilla 250", "Sin pañetar con arcilla 300", "Sin pañetar con concreto 100", "Sin pañetar con concreto 150", "Sin pañetar con concreto 200", "Sin pañetar con concreto 250", "Sin pañetar con concreto 300" };
```

```
double[] valueMUROS = { 0, 1, 2.5, 1.8, 2.5, 3.1, 3.8, 4.4, 1.3, 2, 2.6, 3.3, 3.9, 1.4, 1.45, 1.9, 2.25, 2.6, 1.7, 2.25, 2.7, 3.15, 1.8, 2.3, 2.8, 3.3, 1.8, 2.4, 3, 3.45, 2, 2.6, 3.2, 3.75, 2.2, 2.9, 3.6, 4.3, 3, 4, 5, 6.1, 1.9, 2.9, 3.8, 4.7, 5.5, 2, 3.1, 4.2, 5.3, 6.4 };
```

```
string[] NODETALLADOS = { "*Ninguna", "Reunión fachada y particiones", "Reunión piso y cubierta", "Oficinas fachada y particiones móviles", "Oficinas piso y cubierta móviles", "Oficinas fachada y particiones fijas", "Oficinas piso y cubierta fijas", "Educativos fachada y particiones", "Educativos piso y cubierta", "Fabricas fachada y particiones", "Fabricas piso y cubierta", "Institucional internados fachada y particiones", "Institucional internados piso y cubierta", "Institucional centros de detención fachada y particiones", "Institucional centros de detención piso y cubiertas", "Institucional guarderías fachada y particiones", "Institucional guarderías piso y cubiertas", "Comercio fachada y particiones", "Comercio piso y cubierta", "Residencial mampostería fachada y particiones", "Residencial mampostería piso y cubierta", "Residencial livianas fachada y" };
```

```
particiones", "Residencial livianas piso y cubierta", "Almacenamiento fachada y particiones",
"Almacenamiento piso y cubierta", "Garajes fachada y particiones", "Garajes Piso y cubierta" };
double[] valueNODETALLADOS = { 0, 1, 1.8, 1, 1.8, 2, 1.8, 2, 1.5, 0.8, 1.6, 2, 1.6, 2.5, 1.8,
2, 1.6, 1.5, 1.4, 3, 1.6, 2, 1.4, 1.5, 1.5, 0.2, 1 };
```

```
public Main()
{
    InitializeComponent();

    for (double i = 2; i <= 9; i++)
    {
        tableLayoutPanelLuces.RowStyles[Convert.ToInt16(i)].Height = 0;
        tableLayoutPanelLuces.Height = 60;
            tableLayoutPanelLucesC1.RowStyles[Convert.ToInt16(i)].Height = 0;
        tableLayoutPanelLucesC1.Height = 60;
            tableLayoutPanelLucesC2.RowStyles[Convert.ToInt16(i)].Height = 0;
        tableLayoutPanelLucesC2.Height = 60;
            tableLayoutPanelLucesC3.RowStyles[Convert.ToInt16(i)].Height = 0;
        tableLayoutPanelLucesC3.Height = 60;
            tableLayoutPanelLucesC4.RowStyles[Convert.ToInt16(i)].Height = 0;
        tableLayoutPanelLucesC4.Height = 60;
            tableLayoutPanelLucesC5.RowStyles[Convert.ToInt16(i)].Height = 0;
        tableLayoutPanelLucesC5.Height = 60;
            tableLayoutPanelLucesC6.RowStyles[Convert.ToInt16(i)].Height = 0;
        tableLayoutPanelLucesC6.Height = 60;
            tableLayoutPanelLucesR.RowStyles[Convert.ToInt16(i)].Height = 0;
        tableLayoutPanelLucesR.Height = 60;
    }
        tableLayoutPanelLucesR.RowStyles[Convert.ToInt16(10)].Height = 30;
        tableLayoutPanelLucesR.Height = Convert.ToInt16(10 * 30 + 30);
```

```
//-----1/6
```

```
numericUpDownL1FP1.Visible = false;
numericUpDownL2FP1.Visible = false;
numericUpDownL3FP1.Visible = false;
numericUpDownL4FP1.Visible = false;
numericUpDownL5FP1.Visible = false;
numericUpDownL6FP1.Visible = false;
numericUpDownL7FP1.Visible = false;
numericUpDownL8FP1.Visible = false;
numericUpDownL9FP1.Visible = false;
```

```
numericUpDownLuz1LT1.Visible = false;
numericUpDownLuz2LT1.Visible = false;
numericUpDownLuz3LT1.Visible = false;
numericUpDownLuz4LT1.Visible = false;
numericUpDownLuz5LT1.Visible = false;
numericUpDownLuz6LT1.Visible = false;
numericUpDownLuz7LT1.Visible = false;
numericUpDownLuz8LT1.Visible = false;
numericUpDownLuz9LT1.Visible = false;
```

```
numericUpDownLuz1La1.Visible = false;
numericUpDownLuz1Lb1.Visible = false;
numericUpDownLuz2La1.Visible = false;
numericUpDownLuz2Lb1.Visible = false;
numericUpDownLuz3La1.Visible = false;
numericUpDownLuz3Lb1.Visible = false;
numericUpDownLuz4La1.Visible = false;
numericUpDownLuz4Lb1.Visible = false;
numericUpDownLuz5La1.Visible = false;
numericUpDownLuz5Lb1.Visible = false;
numericUpDownLuz6La1.Visible = false;
numericUpDownLuz6Lb1.Visible = false;
numericUpDownLuz7La1.Visible = false;
numericUpDownLuz7Lb1.Visible = false;
numericUpDownLuz8La1.Visible = false;
numericUpDownLuz8Lb1.Visible = false;
numericUpDownLuz9La1.Visible = false;
```

```
numericUpDownLuz9Lb1.Visible = false;
```

```
//-----2/6
```

```
numericUpDownL1FP2.Visible = false;  
numericUpDownL2FP2.Visible = false;  
numericUpDownL3FP2.Visible = false;  
numericUpDownL4FP2.Visible = false;  
numericUpDownL5FP2.Visible = false;  
numericUpDownL6FP2.Visible = false;  
numericUpDownL7FP2.Visible = false;  
numericUpDownL9FP2.Visible = false;
```

```
numericUpDownLuz1LT2.Visible = false;  
numericUpDownLuz2LT2.Visible = false;  
numericUpDownLuz3LT2.Visible = false;  
numericUpDownLuz4LT2.Visible = false;  
numericUpDownLuz5LT2.Visible = false;  
numericUpDownLuz6LT2.Visible = false;  
numericUpDownLuz7LT2.Visible = false;  
numericUpDownLuz8LT2.Visible = false;  
numericUpDownLuz9LT2.Visible = false;
```

```
numericUpDownLuz1La2.Visible = false;  
numericUpDownLuz1Lb2.Visible = false;  
numericUpDownLuz2La2.Visible = false;  
numericUpDownLuz2Lb2.Visible = false;  
numericUpDownLuz3La2.Visible = false;  
numericUpDownLuz3Lb2.Visible = false;  
numericUpDownLuz4La2.Visible = false;  
numericUpDownLuz4Lb2.Visible = false;  
numericUpDownLuz5La2.Visible = false;  
numericUpDownLuz5Lb2.Visible = false;  
numericUpDownLuz6La2.Visible = false;  
numericUpDownLuz6Lb2.Visible = false;  
numericUpDownLuz7La2.Visible = false;  
numericUpDownLuz7Lb2.Visible = false;  
numericUpDownLuz8La2.Visible = false;  
numericUpDownLuz8Lb2.Visible = false;  
numericUpDownLuz9La2.Visible = false;  
numericUpDownLuz9Lb2.Visible = false;
```

```
//-----3/6
```

```
numericUpDownL1FP3.Visible = false;  
numericUpDownL2FP3.Visible = false;  
numericUpDownL3FP3.Visible = false;  
numericUpDownL4FP3.Visible = false;  
numericUpDownL5FP3.Visible = false;  
numericUpDownL6FP3.Visible = false;  
numericUpDownL7FP3.Visible = false;  
numericUpDownL8FP3.Visible = false;  
numericUpDownL9FP3.Visible = false;
```

```
numericUpDownLuz1LT3.Visible = false;  
numericUpDownLuz2LT3.Visible = false;  
numericUpDownLuz3LT3.Visible = false;  
numericUpDownLuz4LT3.Visible = false;  
numericUpDownLuz5LT3.Visible = false;  
numericUpDownLuz6LT3.Visible = false;  
numericUpDownLuz7LT3.Visible = false;  
numericUpDownLuz8LT3.Visible = false;  
numericUpDownLuz9LT3.Visible = false;
```

```
numericUpDownLuz1La3.Visible = false;  
numericUpDownLuz1Lb3.Visible = false;  
numericUpDownLuz2La3.Visible = false;  
numericUpDownLuz2Lb3.Visible = false;  
numericUpDownLuz3La3.Visible = false;  
numericUpDownLuz3Lb3.Visible = false;  
numericUpDownLuz4La3.Visible = false;
```

```
numericUpDownLuz4Lb3.Visible = false;
numericUpDownLuz5La3.Visible = false;
numericUpDownLuz5Lb3.Visible = false;
numericUpDownLuz6La3.Visible = false;
numericUpDownLuz6Lb3.Visible = false;
numericUpDownLuz7La3.Visible = false;
numericUpDownLuz7Lb3.Visible = false;
numericUpDownLuz8La3.Visible = false;
numericUpDownLuz8Lb3.Visible = false;
numericUpDownLuz9La3.Visible = false;
numericUpDownLuz9Lb3.Visible = false;
```

```
//-----4/6
```

```
numericUpDownL1FP4.Visible = false;
numericUpDownL2FP4.Visible = false;
numericUpDownL3FP4.Visible = false;
numericUpDownL4FP4.Visible = false;
numericUpDownL5FP4.Visible = false;
numericUpDownL6FP4.Visible = false;
numericUpDownL7FP4.Visible = false;
numericUpDownL8FP4.Visible = false;
numericUpDownL9FP4.Visible = false;
```

```
numericUpDownLuz1LT4.Visible = false;
numericUpDownLuz2LT4.Visible = false;
numericUpDownLuz3LT4.Visible = false;
numericUpDownLuz4LT4.Visible = false;
numericUpDownLuz5LT4.Visible = false;
numericUpDownLuz6LT4.Visible = false;
numericUpDownLuz7LT4.Visible = false;
numericUpDownLuz8LT4.Visible = false;
numericUpDownLuz9LT4.Visible = false;
```

```
numericUpDownLuz1La4.Visible = false;
numericUpDownLuz1Lb4.Visible = false;
numericUpDownLuz2La4.Visible = false;
numericUpDownLuz2Lb4.Visible = false;
numericUpDownLuz3La4.Visible = false;
numericUpDownLuz3Lb4.Visible = false;
numericUpDownLuz4La4.Visible = false;
numericUpDownLuz4Lb4.Visible = false;
numericUpDownLuz5La4.Visible = false;
numericUpDownLuz5Lb4.Visible = false;
numericUpDownLuz6La4.Visible = false;
numericUpDownLuz6Lb4.Visible = false;
numericUpDownLuz7La4.Visible = false;
numericUpDownLuz7Lb4.Visible = false;
numericUpDownLuz8La4.Visible = false;
numericUpDownLuz8Lb4.Visible = false;
numericUpDownLuz9La4.Visible = false;
numericUpDownLuz9Lb4.Visible = false;
```

```
//-----5/6
```

```
numericUpDownL1FP5.Visible = false;
numericUpDownL2FP5.Visible = false;
numericUpDownL3FP5.Visible = false;
numericUpDownL4FP5.Visible = false;
numericUpDownL5FP5.Visible = false;
numericUpDownL6FP5.Visible = false;
numericUpDownL7FP5.Visible = false;
numericUpDownL8FP5.Visible = false;
numericUpDownL9FP5.Visible = false;
```

```
numericUpDownLuz1LT5.Visible = false;
numericUpDownLuz2LT5.Visible = false;
numericUpDownLuz3LT5.Visible = false;
numericUpDownLuz4LT5.Visible = false;
numericUpDownLuz5LT5.Visible = false;
numericUpDownLuz6LT5.Visible = false;
```

```
numericUpDownLuz7LT5.Visible = false;
numericUpDownLuz8LT5.Visible = false;
numericUpDownLuz9LT5.Visible = false;
```

```
numericUpDownLuz1La5.Visible = false;
numericUpDownLuz1Lb5.Visible = false;
numericUpDownLuz2La5.Visible = false;
numericUpDownLuz2Lb5.Visible = false;
numericUpDownLuz3La5.Visible = false;
numericUpDownLuz3Lb5.Visible = false;
numericUpDownLuz4La5.Visible = false;
numericUpDownLuz4Lb5.Visible = false;
numericUpDownLuz5La5.Visible = false;
numericUpDownLuz5Lb5.Visible = false;
numericUpDownLuz6La5.Visible = false;
numericUpDownLuz6Lb5.Visible = false;
numericUpDownLuz7La5.Visible = false;
numericUpDownLuz7Lb5.Visible = false;
numericUpDownLuz8La5.Visible = false;
numericUpDownLuz8Lb5.Visible = false;
numericUpDownLuz9La5.Visible = false;
numericUpDownLuz9Lb5.Visible = false;
```

```
//-----6/6
```

```
numericUpDownL1FP6.Visible = false;
numericUpDownL2FP6.Visible = false;
numericUpDownL3FP6.Visible = false;
numericUpDownL4FP6.Visible = false;
numericUpDownL5FP6.Visible = false;
numericUpDownL6FP6.Visible = false;
numericUpDownL7FP6.Visible = false;
numericUpDownL9FP6.Visible = false;
```

```
numericUpDownLuz1LT6.Visible = false;
numericUpDownLuz2LT6.Visible = false;
numericUpDownLuz3LT6.Visible = false;
numericUpDownLuz4LT6.Visible = false;
numericUpDownLuz5LT6.Visible = false;
numericUpDownLuz6LT6.Visible = false;
numericUpDownLuz7LT6.Visible = false;
numericUpDownLuz9LT6.Visible = false;
```

```
numericUpDownLuz1La6.Visible = false;
numericUpDownLuz1Lb6.Visible = false;
numericUpDownLuz2La6.Visible = false;
numericUpDownLuz2Lb6.Visible = false;
numericUpDownLuz3La6.Visible = false;
numericUpDownLuz3Lb6.Visible = false;
numericUpDownLuz4La6.Visible = false;
numericUpDownLuz4Lb6.Visible = false;
numericUpDownLuz5La6.Visible = false;
numericUpDownLuz5Lb6.Visible = false;
numericUpDownLuz6La6.Visible = false;
numericUpDownLuz6Lb6.Visible = false;
numericUpDownLuz7La6.Visible = false;
numericUpDownLuz7Lb6.Visible = false;
numericUpDownLuz8La6.Visible = false;
numericUpDownLuz8Lb6.Visible = false;
numericUpDownLuz9La6.Visible = false;
numericUpDownLuz9Lb6.Visible = false;
```

```
//-----
```

```
numericUpDownL1FP2.Visible = false;
numericUpDownL2FP2.Visible = false;
numericUpDownL3FP2.Visible = false;
numericUpDownL4FP2.Visible = false;
numericUpDownL5FP2.Visible = false;
numericUpDownL6FP2.Visible = false;
numericUpDownL7FP2.Visible = false;
```

```
numericUpDownL8FP2.Visible = false;
numericUpDownL9FP2.Visible = false;

numericUpDownLuz1LT2.Visible = false;
numericUpDownLuz2LT2.Visible = false;
numericUpDownLuz3LT2.Visible = false;
numericUpDownLuz4LT2.Visible = false;
numericUpDownLuz5LT2.Visible = false;
numericUpDownLuz6LT2.Visible = false;
numericUpDownLuz7LT2.Visible = false;
numericUpDownLuz8LT2.Visible = false;
numericUpDownLuz9LT2.Visible = false;

numericUpDownLuz1La2.Visible = false;
numericUpDownLuz1Lb2.Visible = false;
numericUpDownLuz2La2.Visible = false;
numericUpDownLuz2Lb2.Visible = false;
numericUpDownLuz3La2.Visible = false;
numericUpDownLuz3Lb2.Visible = false;
numericUpDownLuz4La2.Visible = false;
numericUpDownLuz4Lb2.Visible = false;
numericUpDownLuz5La2.Visible = false;
numericUpDownLuz5Lb2.Visible = false;
numericUpDownLuz6La2.Visible = false;
numericUpDownLuz6Lb2.Visible = false;
numericUpDownLuz7La2.Visible = false;
numericUpDownLuz7Lb2.Visible = false;
numericUpDownLuz8La2.Visible = false;
numericUpDownLuz8Lb2.Visible = false;
numericUpDownLuz9La2.Visible = false;
numericUpDownLuz9Lb2.Visible = false;
//-----

numericUpDownLuz1Carga1.Visible = false;
numericUpDownLuz1Carga2.Visible = false;
numericUpDownLuz1Carga3.Visible = false;
numericUpDownLuz1Carga4.Visible = false;
numericUpDownLuz1Carga5.Visible = false;
numericUpDownLuz1Carga6.Visible = false;

numericUpDownLuz2Carga1.Visible = false;
numericUpDownLuz2Carga2.Visible = false;
numericUpDownLuz2Carga3.Visible = false;
numericUpDownLuz2Carga4.Visible = false;
numericUpDownLuz2Carga5.Visible = false;
numericUpDownLuz2Carga6.Visible = false;

numericUpDownLuz3Carga1.Visible = false;
numericUpDownLuz3Carga2.Visible = false;
numericUpDownLuz3Carga3.Visible = false;
numericUpDownLuz3Carga4.Visible = false;
numericUpDownLuz3Carga5.Visible = false;
numericUpDownLuz3Carga6.Visible = false;

numericUpDownLuz4Carga1.Visible = false;
numericUpDownLuz4Carga2.Visible = false;
numericUpDownLuz4Carga3.Visible = false;
numericUpDownLuz4Carga4.Visible = false;
numericUpDownLuz4Carga5.Visible = false;
numericUpDownLuz4Carga6.Visible = false;

numericUpDownLuz5Carga1.Visible = false;
numericUpDownLuz5Carga2.Visible = false;
numericUpDownLuz5Carga3.Visible = false;
numericUpDownLuz5Carga4.Visible = false;
numericUpDownLuz5Carga5.Visible = false;
numericUpDownLuz5Carga6.Visible = false;

numericUpDownLuz6Carga1.Visible = false;
```

```

numericUpDownLuz6Carga2.Visible = false;
numericUpDownLuz6Carga3.Visible = false;
numericUpDownLuz6Carga4.Visible = false;
numericUpDownLuz6Carga5.Visible = false;
numericUpDownLuz6Carga6.Visible = false;

numericUpDownLuz7Carga1.Visible = false;
numericUpDownLuz7Carga2.Visible = false;
numericUpDownLuz7Carga3.Visible = false;
numericUpDownLuz7Carga4.Visible = false;
numericUpDownLuz7Carga5.Visible = false;
numericUpDownLuz7Carga6.Visible = false;

numericUpDownLuz8Carga1.Visible = false;
numericUpDownLuz8Carga2.Visible = false;
numericUpDownLuz8Carga3.Visible = false;
numericUpDownLuz8Carga4.Visible = false;
numericUpDownLuz8Carga5.Visible = false;
numericUpDownLuz8Carga6.Visible = false;

numericUpDownLuz9Carga1.Visible = false;
numericUpDownLuz9Carga2.Visible = false;
numericUpDownLuz9Carga3.Visible = false;
numericUpDownLuz9Carga4.Visible = false;
numericUpDownLuz9Carga5.Visible = false;
numericUpDownLuz9Carga6.Visible = false;

labelCARGA2.Visible = false;
tableLayoutPanelLucesC2.Visible = false;
labelCARGA3.Visible = false;
tableLayoutPanelLucesC3.Visible = false;
labelCARGA4.Visible = false;
tableLayoutPanelLucesC4.Visible = false;
labelCARGA5.Visible = false;
tableLayoutPanelLucesC5.Visible = false;
labelCARGA6.Visible = false;
tableLayoutPanelLucesC6.Visible = false;
paneldatosI.Visible = false;
pnlCarga.Visible = false;
cbxmetodo.Text = "Seleccione metodo";
cbxRecubrimiento.Text = "3";
cbxunidad.Text = "KN/m^2";
numericUpDownNL.Text = "1";
datosiniciales();
}
void datosiniciales()
{
dtcieloraso = new DataTable();
dtcieloraso.Columns.Add("Ocupacion");
dtcieloraso.Columns.Add("Unidad1");
dtcieloraso.Columns.Add("Unidad2");
dtcieloraso.Rows.Add("Canales suspendidas de acero", "0,10", "10");
dtcieloraso.Rows.Add("Ductos mecanicos", "0,20", "20");
dtcieloraso.Rows.Add("Entramado metalico suspendido afinado en cemento", "0,70", "70");
dtcieloraso.Rows.Add("Entramado metalico suspendido afinado yeso", "0,50", "50");
dtcieloraso.Rows.Add("Fibras acusticas", "0,10", "10");
dtcieloraso.Rows.Add("Peñete en yeso o concreto", "0,25", "25");
dtcieloraso.Rows.Add("Pañete entramado de madera", "0,80", "80");
dtcieloraso.Rows.Add("Tableros de yeso", "0,0080", "8,00");
dtcieloraso.Rows.Add("Sistemas de suspencion madera", "0,15", "15");
dtrellenopiso = new DataTable();
dtrellenopiso.Columns.Add("Ocupacion");
dtrellenopiso.Columns.Add("Unidad1");
dtrellenopiso.Columns.Add("Unidad2");
dtrellenopiso.Rows.Add("Arena", "0,015", "15");
dtrellenopiso.Rows.Add("Concreto con escoria", "0,0200", "20");
dtrellenopiso.Rows.Add("Concreto con piedra", "0,0250", "25");
dtrellenopiso.Rows.Add("Concreto ligero", "0,015", "15");
dtpisosyacabados = new DataTable();

```

```

dtpisosyacabados.Columns.Add("Ocupacion");
dtpisosyacabados.Columns.Add("Unidad1");
dtpisosyacabados.Columns.Add("Unidad2");
dtpisosyacabados.Rows.Add("Acabado de piso en concreto", "0,0200", "20");
dtpisosyacabados.Rows.Add("Afinado(25mm)sobre concreto de agregado petreo", "1,5", "150");
dtpisosyacabados.Rows.Add("Baldosa ceramica(20mm)sobre 12mm de mortero", "0,80", "80");
dtpisosyacabados.Rows.Add("Baldosa ceramica(20mm)sobre 25mm de mortero", "1,10", "110");
dtpisosyacabados.Rows.Add("Baldosa sobre 25mm de mortero", "1,10", "110");
dtpisosyacabados.Rows.Add("Bloque de asfalto(50mm), sobre 12mm de mortero", "1,50", "150");
dtpisosyacabados.Rows.Add("Bloque de madera(75mm) sin relleno", "0,50", "50");
dtpisosyacabados.Rows.Add("Bloque de madera (75mm) sobre 12mm de mortero", "0,80", "80");
dtpisosyacabados.Rows.Add("Durmientes de madera 20mm", "0,15", "15");
dtpisosyacabados.Rows.Add("Madera densa 25mm", "0,20", "20");
dtpisosyacabados.Rows.Add("Marmol y mortero sobre concreto de agregado petreo", "1,60", "160");
dtpisosyacabados.Rows.Add("Piso asphaltico o linoleo,6mm", "0,05", "5");
dtpisosyacabados.Rows.Add("Pizarra", "0,030", "30");
dtpisosyacabados.Rows.Add("Terrazo(25mm), concreto 50mm", "1,5", "150");
dtpisosyacabados.Rows.Add("Terrazo(40mm), directamente sobre la losa", "0,90", "90");
dtpisosyacabados.Rows.Add("Terrazo(25mm), sobre afinado en concreto", "1,50", "150");

```

```

dtcubierta = new DataTable();
dtcubierta.Columns.Add("Ocupacion");
dtcubierta.Columns.Add("Unidad1");
dtcubierta.Columns.Add("Unidad2");
dtcubierta.Rows.Add("Cobre o laton", "0,05", "5");
dtcubierta.Rows.Add("Fibra de vidrio", "0,0020", "2");
dtcubierta.Rows.Add("Tablero de fibra", "0,0030", "3");
dtcubierta.Rows.Add("Perlita", "0,0015", "1,5");
dtcubierta.Rows.Add("Espuma de poliestireno", "0,0005", "0,5");
dtcubierta.Rows.Add("Espuma de poliruretano", "0,0010", "1");
dtcubierta.Rows.Add("Cubiertas corrugadas de asbesto-cemento", "0,20", "20");
dtcubierta.Rows.Add("Entablado madera", "0,0060", "6");
dtcubierta.Rows.Add("Laminas de yeso, 12mm", "0,10", "10");
dtcubierta.Rows.Add("Madera laminada", "0,0100", "10");
dtcubierta.Rows.Add("Bituminosa,cubierta de grava", "0,25", "25");
dtcubierta.Rows.Add("Bituminosa,superficie lisa", "0,10", "10");
dtcubierta.Rows.Add("Acabado de piso en", "0,0200", "20");
dtcubierta.Rows.Add("Liquido aplicado", "0,05", "5");
dtcubierta.Rows.Add("Tela asphaltica de una capa", "0,03", "3");
dtcubierta.Rows.Add("Marquesinas,marco metalico,vidrio de 10mm", "0,40", "40");
dtcubierta.Rows.Add("Tableros de fibra 12mm", "0,05", "5");
dtcubierta.Rows.Add("Tableros de madera 50mm", "0,25", "25");
dtcubierta.Rows.Add("Tableros metalico, calibre 20(0,9mm de espesor)", "0,08", "8");
dtcubierta.Rows.Add("Tableros metalico, calibre 18(1,2mm de espesor)", "0,08", "8");
dtcubierta.Rows.Add("Tablillas(shingles) de asbesto-cemento", "0,20", "20");
dtcubierta.Rows.Add("Tablillas(shingles) de asfalto", "0,10", "10");
dtcubierta.Rows.Add("Tablillas(shingles) de madera", "0,15", "15");
dtcubierta.Rows.Add("Teja de arcilla, incluyendo el mortero", "0,80", "80");

```

```

dtrecubrimiento = new DataTable();
dtrecubrimiento.Columns.Add("Ocupacion");
dtrecubrimiento.Columns.Add("Unidad1");
dtrecubrimiento.Columns.Add("Unidad2");
dtrecubrimiento.Rows.Add("Baldosin de cemento", "0,80", "8");
dtrecubrimiento.Rows.Add("Entablado de madera", "0,0060", "6");
dtrecubrimiento.Rows.Add("Madera laminada", "0,0100", "10");
dtrecubrimiento.Rows.Add("Espuma de poliestireno", "0,0005", "0,5");
dtrecubrimiento.Rows.Add("Espuma de poliuretano", "0,0010", "1");
dtrecubrimiento.Rows.Add("Fibra o acrilico", "0,0020", "2");
dtrecubrimiento.Rows.Add("Perlita", "0,0015", "1,5");
dtrecubrimiento.Rows.Add("Tablero de fibra", "0,0030", "3");
dtrecubrimiento.Rows.Add("Tablero de fibra,12mm", "0,05", "5");
dtrecubrimiento.Rows.Add("Tablero de yeso,12mm", "0,10", "10");

```

```

dtcubierta = new DataTable();
dtcubierta.Columns.Add("Ocupacion");
dtcubierta.Columns.Add("Unidad1");
dtcubierta.Columns.Add("Unidad2");

```

```

dtcubierta.Rows.Add("Cobre o laton", "0,05", "5");
dtcubierta.Rows.Add("Fibra de vidrio", "0,0020", "2");
dtcubierta.Rows.Add("Tablero de fibra", "0,0030", "3");
dtcubierta.Rows.Add("Perlita", "0,0015", "1,5");
dtcubierta.Rows.Add("Espuma de poliestireno", "0,0005", "0,5");
dtcubierta.Rows.Add("Espuma de poliuretano", "0,0010", "1");
dtcubierta.Rows.Add("Cubiertas corrugadas de asbesto-cemento", "0,20", "20");
dtcubierta.Rows.Add("Entablado madera", "0,0060", "6");
dtcubierta.Rows.Add("laminas de yeso, 12mm", "0,10", "10");
dtcubierta.Rows.Add("Madera laminada", "0,0100", "10");
dtcubierta.Rows.Add("Bituminosa,cubierta de grava", "0,25", "25");
dtcubierta.Rows.Add("Bituminosa,superficie lisa", "0,10", "10");
dtcubierta.Rows.Add("Acabado de piso en", "0,0200", "20");
dtcubierta.Rows.Add("Liquido aplicado", "0,05", "5");
dtcubierta.Rows.Add("Tela asfaltica de una capa", "0,03", "3");
dtcubierta.Rows.Add("Marquesinas,marco metalico,vidrio de 10mm", "0,40", "40");
dtcubierta.Rows.Add("Tableros de fibra 12mm", "0,05", "5");
dtcubierta.Rows.Add("Tableros de madera 50mm", "0,25", "25");
dtcubierta.Rows.Add("Tableros metalico, calibre 20(0,9mm de espesor)", "0,08", "8");
dtcubierta.Rows.Add("Tableros metalico, calibre 18(1,2mm de espesor)", "0,08", "8");
dtcubierta.Rows.Add("Tablillas(shingles) de asbesto-cemento", "0,20", "20");
dtcubierta.Rows.Add("Tablillas(shingles) de asfalto", "0,10", "10");
dtcubierta.Rows.Add("Tablillas(shingles) de madera", "0,15", "15");
dtcubierta.Rows.Add("Teja de arcilla, incluyendo el mortero", "0,80", "80");
dtrecubrimiento = new DataTable();
dtrecubrimiento.Columns.Add("Ocupacion");
dtrecubrimiento.Columns.Add("Unidad1");
dtrecubrimiento.Columns.Add("Unidad2");
dtrecubrimiento.Rows.Add("Baldosin de cemento", "0,80", "8");
dtrecubrimiento.Rows.Add("Entablado de madera", "0,0060", "6");
dtrecubrimiento.Rows.Add("Madera laminada", "0,0100", "10");
dtrecubrimiento.Rows.Add("Espuma de poliestireno", "0,0005", "0,5");
dtrecubrimiento.Rows.Add("Espuma de poliuretano", "0,0010", "1");
dtrecubrimiento.Rows.Add("Fibra o acrilico", "0,0020", "2");
dtrecubrimiento.Rows.Add("Perlita", "0,0015", "1,5");
dtrecubrimiento.Rows.Add("Tablero de fibra", "0,0030", "3");
dtrecubrimiento.Rows.Add("Tablero de fibra,12mm", "0,05", "5");
dtrecubrimiento.Rows.Add("Tablero de yeso,12mm", "0,10", "10");
dtparticioneslivi = new DataTable();
dtparticioneslivi.Columns.Add("Ocupacion");
dtparticioneslivi.Columns.Add("Unidad1");
dtparticioneslivi.Columns.Add("Unidad2");
dtparticioneslivi.Rows.Add("Particiones moviles de acero(altura parcial)", "0,50", "50");
dtparticioneslivi.Rows.Add("Particiones moviles de acero(altura total)", "0,20", "20");
dtparticioneslivi.Rows.Add("Poste en madera o acero, yeso de 12mm a cada lado", "0,90", "90");
dtparticioneslivi.Rows.Add("Poste en madera 50x100, sin pañetar", "0,30", "30");
dtparticioneslivi.Rows.Add("Poste en madera 50x100, pañete por un lado", "0,60", "60");
dtparticioneslivi.Rows.Add("Poste en madera 50x100, pañete por ambos lados", "2,0", "200");
dtenchape = new DataTable();
dtenchape.Columns.Add("Ocupacion");
dtenchape.Columns.Add("Unidad1");
dtenchape.Columns.Add("Unidad2");
dtenchape.Rows.Add("Enchape ceramico", "0,015", "15");
dtenchape.Rows.Add("Enchape arenisca", "0,013", "13");
dtenchape.Rows.Add("Enchape en caliza ", "0,015", "15");
dtenchape.Rows.Add("Enchape de granito", "0,017", "17");
dtventanas = new DataTable();
dtventanas.Columns.Add("Ocupacion");
dtventanas.Columns.Add("Unidad1");
dtventanas.Columns.Add("Unidad2");
dtventanas.Rows.Add("Muros cortinas de vidrio,entramado y marco", "0,5", "50");
dtventanas.Rows.Add("Ventanas,vidrio,entramadoy marco", "0,45", "45");

dtmuros = new DataTable();
dtmuros.Columns.Add("Ocupacion");
dtmuros.Columns.Add("Unidad1");
dtmuros.Columns.Add("Unidad2");
dtmuros.Rows.Add("Yeso de 15mm, asilado, entablado de 10mm", "1,00", "100");
dtmuros.Rows.Add("Exteriores con enchape en ladrillo ", "2,50", "250");

```

```

dtmuros.Rows.Add(" ", " 100, 150, 200, 250, 300 ", " 10, 15, 20, 25, 30 ");
dtmuros.Rows.Add("Pañetado en ambas caras", "1,80 2,50 3,10 3,80 4,40 ", " 180 250 310 380 440");
dtmuros.Rows.Add("Sin pañetar", "1,30 2,00 2,60 3,30 3,90 ", " 130 200 260 330 390");
dtmuros.Rows.Add("Sin relleno", "1,40 1,45 1,90 2,25 2,60 ", " 140 145 190 225 260");
dtmuros.Rows.Add("Relleno cada 1,2m ", " 1,70 2,25 2,70 3,15 ", " 170 225 270 315");
dtmuros.Rows.Add("Relleno cada 1,0m", " 1,80 2,30 2,80 3,30 ", " 180 230 280 330");
dtmuros.Rows.Add("Relleno cada 0,8m", " 1,80 2,40 3,00 3,45 ", " 180 240 300 345");
dtmuros.Rows.Add("Relleno cada 0,6m", " 2,00 2,60 3,20 3,75 ", " 200 260 320 375");
dtmuros.Rows.Add("Relleno cada 0,4m", " 2.20 2,90 3,60 4,30 ", " 220 290 360 430");
dtmuros.Rows.Add("todas las celdas llenas ", " 3,00 4,00 5,00 6,10 ", " 300 400 500 610");
dtmuros.Rows.Add("Sin pañetar", "1,90 2,90 3,80, 470 550 ", " 190 290 380 470 550");
dtmuros.Rows.Add("Sin pañetar", "2,00 3,10 4,20 5,30 6,40 ", " 200 310 420 530 640");
//*****WJL*****
dtreunion = new DataTable();
dtreunion.Columns.Add("Ocupacion");
dtreunion.Columns.Add("Unidad1");
dtreunion.Columns.Add("Unidad2");
dtreunion.Rows.Add("Balcones", "5", "500");
dtreunion.Rows.Add("Corredores y escaleras", "5", "500");
dtreunion.Rows.Add("Silleteria fija", "3", "300");
dtreunion.Rows.Add("Gimnasios", "5", "500");
dtreunion.Rows.Add("Vestibulos", "5", "500");
dtreunion.Rows.Add("Silleteria movil", "5", "500");
dtreunion.Rows.Add("Areas recreativas", "5", "500");
dtreunion.Rows.Add("Plataformas", "5", "500");
dtreunion.Rows.Add("Escenarios", "7,5", "750");
dtoficinas = new DataTable();
dtoficinas.Columns.Add("Ocupacion");
dtoficinas.Columns.Add("Unidad1");
dtoficinas.Columns.Add("Unidad2");
dtoficinas.Rows.Add("Corredores y escaleras", "3", "300");
dtoficinas.Rows.Add("Oficinas", "2", "200");
dtoficinas.Rows.Add("Restaurantes", "5", "500");
dteducativos = new DataTable();
dteducativos.Columns.Add("Ocupacion");
dteducativos.Columns.Add("Unidad1");
dteducativos.Columns.Add("Unidad2");
dteducativos.Rows.Add("Salones de clase", "2", "200");
dteducativos.Rows.Add("Corredores y escaleras", "5", "500");
dtbibliotecas = new DataTable();
dtbibliotecas.Columns.Add("Ocupacion");
dtbibliotecas.Columns.Add("Unidad1");
dtbibliotecas.Columns.Add("Unidad2");
dtbibliotecas.Rows.Add("Salones de lectura", "2", "200");
dtbibliotecas.Rows.Add("Estanteria", "7", "700");
dtfabricas = new DataTable();
dtfabricas.Columns.Add("Ocupacion");
dtfabricas.Columns.Add("Unidad1");
dtfabricas.Columns.Add("Unidad2");
dtfabricas.Rows.Add("Industrias livianas", "5", "500");
dtfabricas.Rows.Add("Industrias pesadas", "10", "1000");
dtinstitucional = new DataTable();
dtinstitucional.Columns.Add("Ocupacion");
dtinstitucional.Columns.Add("Unidad1");
dtinstitucional.Columns.Add("Unidad2");
dtinstitucional.Rows.Add("Cuartos de cirujia, laboratorios", "4", "400");
dtinstitucional.Rows.Add("Cuartos privados", "2", "200");
dtinstitucional.Rows.Add("Corredores y escaleras", "5", "500");
dtcomercio = new DataTable();
dtcomercio.Columns.Add("Ocupacion");
dtcomercio.Columns.Add("Unidad1");
dtcomercio.Columns.Add("Unidad2");
dtcomercio.Rows.Add("Minorista", "5", "500");
dtcomercio.Rows.Add("Mayorista", "5", "500");
dtresidencial = new DataTable();
dtresidencial.Columns.Add("Ocupacion");
dtresidencial.Columns.Add("Unidad1");
dtresidencial.Columns.Add("Unidad2");
dtresidencial.Rows.Add("balcones", "5", "500");

```

```

dtresidencial.Rows.Add("Cuartos privados y corredores", "1,8", "180");
dtresidencial.Rows.Add("Escaleras", "3", "300");
dtalmacenamiento = new DataTable();
dtalmacenamiento.Columns.Add("Ocupacion");
dtalmacenamiento.Columns.Add("Unidad1");
dtalmacenamiento.Columns.Add("Unidad2");
dtalmacenamiento.Rows.Add("Liviano", "6", "600");
dtalmacenamiento.Rows.Add("Pesado", "12", "1200");
dtgarajes = new DataTable();
dtgarajes.Columns.Add("Ocupacion");
dtgarajes.Columns.Add("Unidad1");
dtgarajes.Columns.Add("Unidad2");
dtgarajes.Rows.Add("Automoviles pasajeros", "2,5", "250");
dtgarajes.Rows.Add("Vehiculos de carga hasta 2000 kg de capacidad", "5", "500");
dtcoliseo= new DataTable();
dtcoliseo.Columns.Add("Ocupacion");
dtcoliseo.Columns.Add("Unidad1");
dtcoliseo.Columns.Add("Unidad2");
dtcoliseo.Rows.Add("Graderias", "5", "500");
dtcoliseo.Rows.Add("Escaleras", "5", "500");

```

En orden a lo que se ve en la pantalla principal de KAI-SAD, Reinicio con su palabra lo dice, su funcionalidad es hacer un barrido para eliminación de datos en todos los panes datos iniciales, luces y carga, calcular resultado, gráfica cortante, gráfica momento, refuerzo a tracción y compresión, y diseño cortante.

```

/// <summary>
/// inicio
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void btnInicio_ItemClick(object sender, ItemClickEventArgs e)
{
    try
    {
        Main frmmain = new Main();
        frmmain.Show();
        this.Dispose(false);
        plResultado.Visible = false;
        paneldatosI.Dock = DockStyle.Fill;
        pnlCarga.Visible = false;
        paneldatosI.Visible = true;
        plGráfica1.Visible = false;
        plGráfica2.Visible = false;
        plRefuerzo.Visible = false;mm
        pldiseñocortante.Visible = false;
    }
    catch (Exception)
    {
        MessageBox.Show("Error al cargar datos Iniciales", "Error", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}
/*----- validacion de almenos 1 carga-----*/
/*-----*/
if (NCL1 == 0 && NCL2 == 0 && NCL3 == 0 && NCL4 == 0 && NCL5 == 0 && NCL6 == 0 && NCL7 == 0 && NCL8
== 0 && NCL9 == 0)
{
    MessageBox.Show("Almenos debe tener una carga");
    return;
}
/*-----*/
/*----- validacion de ninguna cargas-----*/
/*-----*/
double[,] LuzCarga = {
    { Luz1Carga1, Luz1Carga2, Luz1Carga3, Luz1Carga4, Luz1Carga5, Luz1Carga6 },
    { Luz2Carga1, Luz2Carga2, Luz2Carga3, Luz2Carga4, Luz2Carga5, Luz2Carga6 },
    { Luz3Carga1, Luz3Carga2, Luz3Carga3, Luz3Carga4, Luz3Carga5, Luz3Carga6 },

```

```

    { Luz4Carga1, Luz4Carga2, Luz4Carga3, Luz4Carga4, Luz4Carga5, Luz4Carga6 },
    { Luz5Carga1, Luz5Carga2, Luz5Carga3, Luz5Carga4, Luz5Carga5, Luz5Carga6 },
    { Luz6Carga1, Luz6Carga2, Luz6Carga3, Luz6Carga4, Luz6Carga5, Luz6Carga6 },
    { Luz7Carga1, Luz7Carga2, Luz7Carga3, Luz7Carga4, Luz7Carga5, Luz7Carga6 },
    { Luz8Carga1, Luz8Carga2, Luz8Carga3, Luz8Carga4, Luz8Carga5, Luz8Carga6 },
    { Luz9Carga1, Luz9Carga2, Luz9Carga3, Luz9Carga4, Luz9Carga5, Luz9Carga6 } };
double[] NCL = { NCL1, NCL2, NCL3, NCL4, NCL5, NCL6, NCL7, NCL8, NCL9 };
/// verificar el numero de cargas por luz///
for (int i = 0; i < NL; i++)
{
    for (int l = 0; l <= (NCL[i] - 1); l++)
    {
        if (LuzCarga[i, Convert.ToInt16(NCL[i] - 1)] == 0)
        {
            MessageBox.Show("Valor de entrada tipo de carga " + " perteneciente a la luz " + (i + 1) +
"Carga Numero" + (NCL[i]) + " posee valor 0");
            return;
        }
    }
}
/*-----*/
/*----- validacion solo triangular seleccionada-----*/
/*-----*/
for (int i = 0; i < NL; i++)
{
    if (NCL[i] > 1)
    {
        if (LuzCarga[i, 0] == 2 || LuzCarga[i, 0] == 3)
        {
double[,] LTLuz = {
    { LT1Luz1,LT2Luz1, LT3Luz1, LT4Luz1,LT5Luz1,LT6Luz1},
    { LT1Luz2,LT2Luz2, LT3Luz2, LT4Luz2,LT5Luz2,LT6Luz2},
    { LT1Luz3,LT2Luz3, LT3Luz3, LT4Luz3,LT5Luz3,LT6Luz3},
    { LT1Luz4,LT2Luz4, LT3Luz4, LT4Luz4,LT5Luz4,LT6Luz4},
    { LT1Luz5,LT2Luz5, LT3Luz5, LT4Luz5,LT5Luz5,LT6Luz5},
    { LT1Luz6,LT2Luz6, LT3Luz6, LT4Luz6,LT5Luz6,LT6Luz6},
    { LT1Luz7,LT2Luz7, LT3Luz7, LT4Luz7,LT5Luz7,LT6Luz7},
    { LT1Luz8,LT2Luz8, LT3Luz8, LT4Luz8,LT5Luz8,LT6Luz8},
    { LT1Luz9,LT2Luz9, LT3Luz9, LT4Luz9,LT5Luz9,LT6Luz9}
};
double[,] LFP = {
    { L1FP1, L1FP2, L1FP3, L1FP4, L1FP5, L1FP6 },
    { L2FP1, L2FP2, L2FP3, L2FP4, L2FP5, L2FP6 },
    { L3FP1, L3FP2, L3FP3, L3FP4, L3FP5, L3FP6 },
    { L4FP1, L4FP2, L4FP3, L4FP4, L4FP5, L4FP6 },
    { L5FP1, L5FP2, L5FP3, L5FP4, L5FP5, L5FP6 },
    { L6FP1, L6FP2, L6FP3, L6FP4, L6FP5, L6FP6 },
    { L7FP1, L7FP2, L7FP3, L7FP4, L7FP5, L7FP6 },
    { L8FP1, L8FP2, L8FP3, L8FP4, L8FP5, L8FP6 },
    { L9FP1, L9FP2, L9FP3, L9FP4, L9FP5, L9FP6 },
};
double[,] Luz1a = {
    { Luz11a1,Luz11a2, Luz11a3, Luz11a4,Luz11a5, Luz11a6 },
    { Luz21a1,Luz21a2, Luz21a3, Luz21a4,Luz11a5, Luz21a6 },
    { Luz31a1,Luz31a2, Luz31a3, Luz31a4,Luz11a5, Luz31a6 },
    { Luz41a1,Luz41a2, Luz41a3, Luz41a4,Luz11a5, Luz41a6 },
    { Luz51a1,Luz51a2, Luz51a3, Luz51a4,Luz11a5, Luz51a6 },
    { Luz61a1,Luz61a2, Luz61a3, Luz61a4,Luz11a5, Luz61a6 },
    { Luz71a1,Luz71a2, Luz61a3, Luz71a4,Luz11a5, Luz71a6 },
    { Luz81a1,Luz81a2, Luz71a3, Luz81a4,Luz11a5, Luz81a6 },
    { Luz91a1,Luz91a2, Luz81a3, Luz91a4,Luz11a5, Luz91a6 },
};
double[,] Luz1b = {
    { Luz11b1,Luz11b2, Luz11b3, Luz11b4,Luz11b5, Luz11b6 },
    { Luz21b1,Luz21b2, Luz21b3, Luz21b4,Luz11b5, Luz21b6 },
    { Luz31b1,Luz31b2, Luz31b3, Luz31b4,Luz11b5, Luz31b6 },
    { Luz41b1,Luz41b2, Luz41b3, Luz41b4,Luz11b5, Luz41b6 },
    { Luz51b1,Luz51b2, Luz51b3, Luz51b4,Luz11b5, Luz51b6 },
    { Luz61b1,Luz61b2, Luz61b3, Luz61b4,Luz11b5, Luz61b6 },
};

```

```

{ Luz71b1,Luz71b2, Luz61b3, Luz71b4,Luz11b5, Luz71b6 },
{ Luz81b1,Luz81b2, Luz71b3, Luz81b4,Luz11b5, Luz81b6 },
{ Luz91b1,Luz91b2, Luz81b3, Luz91b4,Luz11b5, Luz91b6 },
};
for (int i = 0; i < NL; i++)
{
  MFRL = 0;
  MFCL = 0;
  MFTIL = 0;
  MFTDL = 0;
  MFPIL = 0;
  MFPDL = 0;
  for (int j = 0; j < NCL[i]; j++)
  {
    if (LuzCarga[i, j] == 1)
    {
      MFRL = MFRL + ((Wu * (LTLuz[i, j] * LTLuz[i, j])) / 12);
    }
    if (LuzCarga[i, j] == 2)
    {
      MFTIL = (Wu * LTLuz[i, j] * LTLuz[i, j]) / 20;
      MFTDL = (Wu * LTLuz[i, j] * LTLuz[i, j]) / 30;

      switch ((i + 1))
      {
        case 1:
          MFLP[0] = MFTIL;
          MFLN[0] = MFTDL;
          break;
        case 2:
          MFLP[1] = MFTIL;
          MFLN[1] = MFTDL;
          break;
        case 3:
          MFLP[2] = MFTIL;
          MFLN[2] = MFTDL;
          break;
        case 4:
          MFLP[3] = MFTIL;
          MFLN[3] = MFTDL;
          break;
        case 5:
          MFLP[4] = MFTIL;
          MFLN[4] = MFTDL;
          break;
        case 6:
          MFLP[5] = MFTIL;
          MFLN[5] = MFTDL;
          break;
        case 7:
          MFLP[6] = MFTIL;
          MFLN[6] = MFTDL;
          break;
        case 8:
          MFLP[7] = MFTIL;
          MFLN[7] = MFTDL;
          break;
        case 9:
          MFLP[8] = MFTIL;
          MFLN[8] = MFTDL;
          break;
      }
    }
  }
  if (LuzCarga[i, j] == 3)
  {
    MFTIL = (Wu * LTLuz[i, j] * LTLuz[i, j]) / 30;
    MFTDL = (Wu * LTLuz[i, j] * LTLuz[i, j]) / 20;
    switch ((i + 1))

```

```

    {
        case 1:
            MFLP[0] = MFTIL;
            MFLN[0] = MFTDL;
            break;
        case 2:
            MFLP[1] = MFTIL;
            MFLN[1] = MFTDL;
            break;
        case 3:
            MFLP[2] = MFTIL;
            MFLN[2] = MFTDL;
            break;
        case 4:
            MFLP[3] = MFTIL;
            MFLN[3] = MFTDL;
            break;
        case 5:
            MFLP[4] = MFTIL;
            MFLN[4] = MFTDL;
            break;
        case 6:
            MFLP[5] = MFTIL;
            MFLN[5] = MFTDL;
            break;
        case 7:
            MFLP[6] = MFTIL;
            MFLN[6] = MFTDL;
            break;
        case 8:
            MFLP[7] = MFTIL;
            MFLN[7] = MFTDL;
            break;
        case 9:
            MFLP[8] = MFTIL;
            MFLN[8] = MFTDL;
            break;
    }
}
if (LuzCarga[i, j] == 4)
{
    MFCL = MFCL + ((LFP[i, j] * LTLuz[i, j]) / 8);
}
if (LuzCarga[i, j] == 5)
{
    MFPIL = MFPIL + ((LFP[i, j] * Luzla[i, j] * Luzlb[i, j] * Luzlb[i, j]) / (LTLuz[i, j] *
LTLuz[i, j]));
    MFPDL = MFPDL + ((LFP[i, j] * Luzla[i, j] * Luzla[i, j] * Luzlb[i, j]) / (LTLuz[i, j] *
LTLuz[i, j]));
}
}
//MessageBox.Show("" + i + ":" + MFPIL);
MFPIL = (MFRL + MFCL + MFPIL);
MFPDL = (-MFRL - MFCL - MFPDL);
if (MFLP[i] != 0)
{
    MFPIL = MFLP[i];
}
if (MFLN[i] != 0)
{
    MFPDL = -MFLN[i];
}
}
switch ((i + 1))
{
    case 1:
        labelMFL1.Text = "+ M ^ F = " + MFPIL.ToString("0.00") + "\n" + "- M ^ F = " +
MFPDL.ToString("0.00");
}

```

```

        MFLP[0] = MFPIIL;
        MFLN[0] = MFPDL;
        break;
    case 2:
        labelMFL2.Text = "+ M ^ F = " + MFPIIL.ToString("0.00") + "\n" + "- M ^ F = " +
MFPDL.ToString("0.00");
        MFLP[1] = MFPIIL;
        MFLN[1] = MFPDL;
        break;
    case 3:
        labelMFL3.Text = "+ M ^ F = " + MFPIIL.ToString("0.00") + "\n" + "- M ^ F = " +
MFPDL.ToString("0.00");
        MFLP[2] = MFPIIL;
        MFLN[2] = MFPDL;
        break;
    case 4:
        labelMFL4.Text = "+ M ^ F = " + MFPIIL.ToString("0.00") + "\n" + "- M ^ F = " + MFPDL.ToString("0.00");
        MFLP[3] = MFPIIL;
        MFLN[3] = MFPDL;
        break;
    case 5:
        labelMFL5.Text = "+ M ^ F = " + MFPIIL.ToString("0.00") + "\n" + "- M ^ F = " +
MFPDL.ToString("0.00");
        MFLP[4] = MFPIIL;
        MFLN[4] = MFPDL;
        break;
    case 6:
        labelMFL6.Text = "+ M ^ F = " + MFPIIL.ToString("0.00") + "\n" + "- M ^ F = " +
MFPDL.ToString("0.00");
        MFLP[5] = MFPIIL;
        MFLN[5] = MFPDL;
        break;
    case 7:
        labelMFL7.Text = "+ M ^ F = " + MFPIIL.ToString("0.00") + "\n" + "- M ^ F = " +
MFPDL.ToString("0.00");
        MFLP[6] = MFPIIL;
        MFLN[6] = MFPDL;
        break;
    case 8:
        labelMFL8.Text = "+ M ^ F = " + MFPIIL.ToString("0.00") + "\n" + "- M ^ F = " +
MFPDL.ToString("0.00");
        MFLP[7] = MFPIIL;
        MFLN[7] = MFPDL;
        break;
    case 9:
        labelMFL9.Text = "+ M ^ F = " + MFPIIL.ToString("0.00") + "\n" + "- M ^ F = " +
MFPDL.ToString("0.00");
        MFLP[8] = MFPIIL;
        MFLN[8] = MFPDL;
        break;
    }
}
/*----- Calculo de K -----*/
IV = 0.083 * (BV * HV * HV * HV);
IC = 0.083 * (BC * HC * HC * HC);
IT = IV / IC;

//if numero de luces != de 1 entonces LK es el minimo como un multiplo
double mcm = 1, vi = 2; // variables iniciales mcm minimi comun multiplo
double in1 = 0, in2 = 0, in3 = 0, in4 = 0, in5 = 0, in6 = 0, in7 = 0, in8 = 0, in9 = 0; // variables
mcm
switch (Convert.ToInt32(NL))
{
    case 1:
        LK = LT1Luz1;
        K[0] = LK * IT / LT1Luz1;
        labelK1.Text = "K = " + K[0].ToString();
        double Maximo = LT1Luz[0, 0];
        for (int i = 0; i < NL; i++)

```

```

{
if (Maximo < LTLuz[i, 0])
{
x = Convert.ToInt16(Maximo);
}
}
break;
case 2:
// while ((x % LTLuz[0, 0] != 0) || (x % LTLuz[1, 0] != 0))
// x++;
in1 = LTLuz[0, 0];
in2 = LTLuz[1, 0];
if (in1 < 1 || in2 < 1)
{
//in2 = in2 * 10;
//in1 = in1 * 10;
//varmenor = 1;
}
while (vi <= in1 || vi <= in2)
{
if (in1 % vi == 0 || in2 % vi == 0)
{
mcm = mcm * vi;
if (in1 % vi == 0)
{
in1 = in1 / vi;
}
if (in2 % vi == 0)
{
in2 = in2 / vi;
}
}
else
{
vi = vi + 1;
}
}
if (varmenor == 1)
{
mcm = mcm / 10;
}
LK = mcm;
K[0] = IT / LT1Luz1 * LK;
labelK1.Text = "K =" + K[0].ToString();
K[1] = IT / LT1Luz2 * LK;
labelK2.Text = "K =" + K[1].ToString();
break;
case 3:
// while ((x % LTLuz[0, 0] != 0) || (x % LTLuz[1, 0] != 0) || (x % LTLuz[2, 0] != 0))
// x++;
in1 = LTLuz[0, 0];
in2 = LTLuz[1, 0];
in3 = LTLuz[2, 0];
if (in1 < 1 || in2 < 1)
{
//in2 = in2 * 10;
//in1 = in1 * 10;
//varmenor = 1;
}
while (vi <= in1 || vi <= in2 || vi <= in3)
{
if (in1 % vi == 0 || in2 % vi == 0 || in3 % vi == 0)
{
mcm = mcm * vi;
if (in1 % vi == 0)
{
in1 = in1 / vi;
}
if (in2 % vi == 0)

```

```

{
in2 = in2 / vi;
}
if (in3 % vi == 0)
{
in3 = in3 / vi;
}
}
else
{
vi = vi + 1;
}
}
if (varmenor == 1)
{
mcm = mcm / 10;
}
LK = mcm;
K[0] = IT / LT1Luz1 * LK;
labelK1.Text = "K =" + K[0].ToString();
K[1] = IT / LT1Luz2 * LK;
labelK2.Text = "K =" + K[1].ToString();
K[2] = IT / LT1Luz3 * LK;
labelK3.Text = "K =" + K[2].ToString();
break;
case 4:
// while ((x % LTLuz[0, 0] != 0) || (x % LTLuz[1, 0] != 0) || (x % LTLuz[2, 0] != 0) || (x % LTLuz[3,
0] != 0))
//     x++;
in1 = LTLuz[0, 0];
in2 = LTLuz[1, 0];
in3 = LTLuz[2, 0];
in4 = LTLuz[3, 0];
if (in1 < 1 || in2 < 1)
{
//in2 = in2 * 10;
//in1 = in1 * 10;
//varmenor = 1;
}
while (vi <= in1 || vi <= in2 || vi <= in3 || vi <= in4)
{
if (in1 % vi == 0 || in2 % vi == 0 || in3 % vi == 0 || in4 % vi == 0)
{
mcm = mcm * vi;
if (in1 % vi == 0)
{
in1 = in1 / vi;
}
if (in2 % vi == 0)
{
in2 = in2 / vi;
}
if (in3 % vi == 0)
{
in3 = in3 / vi;
}
if (in4 % vi == 0)
{
in4 = in4 / vi;
}
}
}
else
{
vi = vi + 1;
}
}
if (varmenor == 1)
{
mcm = mcm / 10;
}

```

```

}
LK = mcm;
K[0] = IT / LT1Luz1 * LK;
labelK1.Text = "K =" + K[0].ToString();
K[1] = IT / LT1Luz2 * LK;
labelK2.Text = "K =" + K[1].ToString();
K[2] = IT / LT1Luz3 * LK;
labelK3.Text = "K =" + K[2].ToString();
K[3] = IT / LT1Luz4 * LK;
labelK4.Text = "K =" + K[3].ToString();
break;
case 5:
//while ((x % Convert.ToInt16(LTLuz[0], 0]) != 0) || (x % LTLuz[1], 0] != 0) || (x % LTLuz[2], 0] != 0)
// (x % LTLuz[3], 0] != 0) || (x % LTLuz[4], 0] != 0))
//while ((x % LTLuz[0], 0] != 0) || (x % LTLuz[1], 0] != 0) || (x % LTLuz[2], 0]
!= 0) || (x % LTLuz[3], 0] != 0) || (x % LTLuz[4], 0] != 0))
//      x++;
in1 = LTLuz[0], 0];
in2 = LTLuz[1], 0];
in3 = LTLuz[2], 0];
in4 = LTLuz[3], 0];
in5 = LTLuz[4], 0];
if (in1 < 1 || in2 < 1)
{
//in2 = in2 * 10;
// in1 = in1 * 10;
// varmenor = 1;
}
while (vi <= in1 || vi <= in2 || vi <= in3 || vi <= in4 || vi <= in5)
{
if (in1 % vi == 0 || in2 % vi == 0 || in3 % vi == 0 || in4 % vi == 0 || in5 % vi == 0)
{
mcm = mcm * vi;
if (in1 % vi == 0)
{
in1 = in1 / vi;
}
if (in2 % vi == 0)
{
in2 = in2 / vi;
}
if (in3 % vi == 0)
{
in3 = in3 / vi;
}
if (in4 % vi == 0)
{
in4 = in4 / vi;
}
if (in5 % vi == 0)
{
in5 = in5 / vi;
}
}
else
{
vi = vi + 1;
}
}
if (varmenor == 1)
{
mcm = mcm / 10;
}
LK = mcm;
K[0] = IT / LT1Luz1 * LK;
labelK1.Text = "K =" + K[0].ToString();
K[1] = IT / LT1Luz2 * LK;
labelK2.Text = "K =" + K[1].ToString();
K[2] = IT / LT1Luz3 * LK;

```

```

labelK3.Text = "K =" + K[2].ToString();
K[3] = IT / LT1Luz4 * LK;
labelK4.Text = "K =" + K[3].ToString();
K[4] = IT / LT1Luz5 * LK;
labelK5.Text = "K =" + K[4].ToString();
break;
case 6:
//while ((x % LTLuz[0, 0] != 0) || (x % LTLuz[1, 0] != 0) || (x % LTLuz[2, 0] != 0) || (x % LTLuz[3,
0] != 0) || (x % LTLuz[4, 0] != 0) || (x % LTLuz[5, 0] != 0))
//  x++;
in1 = LTLuz[0, 0];
in2 = LTLuz[1, 0];
in3 = LTLuz[2, 0];
in4 = LTLuz[3, 0];
in5 = LTLuz[4, 0];
in6 = LTLuz[5, 0];
if (in1 < 1 || in2 < 1)
{
//in2 = in2 * 10;
// in1 = in1 * 10;
// varmenor = 1;
}
while (vi <= in1 || vi <= in2 || vi <= in3 || vi <= in4 || vi <= in5 || vi <= in6)
{
if (in1 % vi == 0 || in2 % vi == 0 || in3 % vi == 0 || in4 % vi == 0 || in5 % vi == 0 || in6 % vi ==
0)
{
mcm = mcm * vi;
if (in1 % vi == 0)
{
in1 = in1 / vi;
}
if (in2 % vi == 0)
{
in2 = in2 / vi;
}
if (in3 % vi == 0)
{
in3 = in3 / vi;
}
if (in4 % vi == 0)
{
in4 = in4 / vi;
}
if (in5 % vi == 0)
{
in5 = in5 / vi;
}
if (in6 % vi == 0)
{
in6 = in6 / vi;
}
}
else
{
vi = vi + 1;
}
}
if (varmenor == 1)
{
mcm = mcm / 10;
}
LK = mcm;
K[0] = IT / LT1Luz1 * LK;
labelK1.Text = "K =" + K[0].ToString();
K[1] = IT / LT1Luz2 * LK;
labelK2.Text = "K =" + K[1].ToString();
K[2] = IT / LT1Luz3 * LK;
labelK3.Text = "K =" + K[2].ToString();

```

```

K[3] = IT / LT1Luz4 * LK;
labelK4.Text = "K =" + K[3].ToString();
K[4] = IT / LT1Luz5 * LK;
labelK5.Text = "K =" + K[4].ToString();
K[5] = IT / LT1Luz6 * LK;
labelK6.Text = "K =" + K[5].ToString();
break;
case 7:
//while ((x % LTLuz[0, 0] != 0) || (x % LTLuz[1, 0] != 0) || (x % LTLuz[2, 0] != 0) || (x % LTLuz[3,
0] != 0) || (x % LTLuz[4, 0] != 0) || (x % LTLuz[5, 0] != 0) || (x % LTLuz[6, 0] != 0))
// x++;
in1 = LTLuz[0, 0];
in2 = LTLuz[1, 0];
in3 = LTLuz[2, 0];
in4 = LTLuz[3, 0];
in5 = LTLuz[4, 0];
in6 = LTLuz[5, 0];
in7 = LTLuz[6, 0];
if (in1 < 1 || in2 < 1)
{
//in2 = in2 * 10;
// in1 = in1 * 10;
// varmenor = 1;
}
while (vi <= in1 || vi <= in2 || vi <= in3 || vi <= in4 || vi <= in5 || vi <= in6 || vi <= in7)
{
if (in1 % vi == 0 || in2 % vi == 0 || in3 % vi == 0 || in4 % vi == 0 || in5 % vi == 0 || in6 % vi ==
0 || in7 % vi == 0)
{
mcm = mcm * vi;
if (in1 % vi == 0)
{
in1 = in1 / vi;
}
if (in2 % vi == 0)
{
in2 = in2 / vi;
}
if (in3 % vi == 0)
{
in3 = in3 / vi;
}
if (in4 % vi == 0)
{
in4 = in4 / vi;
}
if (in5 % vi == 0)
{
in5 = in5 / vi;
}
if (in6 % vi == 0)
{
in6 = in6 / vi;
}
if (in7 % vi == 0)
{
in7 = in7 / vi;
}
}
else
{
vi = vi + 1;
}
}
if (varmenor == 1)
{
mcm = mcm / 10;
}
LK = mcm;

```

```

K[0] = IT / LT1Luz1 * LK;
labelK1.Text = "K =" + K[0].ToString();
K[1] = IT / LT1Luz2 * LK;
labelK2.Text = "K =" + K[1].ToString();
K[2] = IT / LT1Luz3 * LK;
labelK3.Text = "K =" + K[2].ToString();
K[3] = IT / LT1Luz4 * LK;
labelK4.Text = "K =" + K[3].ToString();
K[4] = IT / LT1Luz5 * LK;
labelK5.Text = "K =" + K[4].ToString();
K[5] = IT / LT1Luz6 * LK;
labelK6.Text = "K =" + K[5].ToString();
K[6] = IT / LT1Luz7 * LK;
labelK7.Text = "K =" + K[6].ToString();
break;
case 8:
//while ((x % LTLuz[0, 0] != 0) || (x % LTLuz[1, 0] != 0) || (x % LTLuz[2, 0] != 0) || (x % LTLuz[3,
0] != 0) || (x % LTLuz[4, 0] != 0) || (x % LTLuz[5, 0] != 0) || (x % LTLuz[6, 0] != 0) || (x % LTLuz[7,
0] != 0))
// x++;
in1 = LTLuz[0, 0];
in2 = LTLuz[1, 0];
in3 = LTLuz[2, 0];
in4 = LTLuz[3, 0];
in5 = LTLuz[4, 0];
in6 = LTLuz[5, 0];
in7 = LTLuz[6, 0];
in8 = LTLuz[7, 0];
if (in1 < 1 || in2 < 1)
{
//in2 = in2 * 10;
// in1 = in1 * 10;
// varmenor = 1;
}
while (vi <= in1 || vi <= in2 || vi <= in3 || vi <= in4 || vi <= in5 || vi <= in6 || vi <= in7 || vi
<= in8)
{
if (in1 % vi == 0 || in2 % vi == 0 || in3 % vi == 0 || in4 % vi == 0 || in5 % vi == 0 || in6 % vi ==
0 || in7 % vi == 0 || in8 % vi == 0)
{
mcm = mcm * vi;
if (in1 % vi == 0)
{
in1 = in1 / vi;
}
if (in2 % vi == 0)
{
in2 = in2 / vi;
}
if (in3 % vi == 0)
{
in3 = in3 / vi;
}
if (in4 % vi == 0)
{
in4 = in4 / vi;
}
if (in5 % vi == 0)
{
in5 = in5 / vi;
}
if (in6 % vi == 0)
{
in6 = in6 / vi;
}
if (in7 % vi == 0)
{

```

```

in7 = in7 / vi;
}
if (in8 % vi == 0)
{
in8 = in8 / vi;
}
}
else
{
vi = vi + 1;
}
}
if (varmenor == 1)
{
mcm = mcm / 10;
}
LK = mcm;
K[0] = IT / LT1Luz1 * LK;
labelK1.Text = "K =" + K[0].ToString();
K[1] = IT / LT1Luz2 * LK;
labelK2.Text = "K =" + K[1].ToString();
K[2] = IT / LT1Luz3 * LK;
labelK3.Text = "K =" + K[2].ToString();
K[3] = IT / LT1Luz4 * LK;
labelK4.Text = "K =" + K[3].ToString();
K[4] = IT / LT1Luz5 * LK;
labelK5.Text = "K =" + K[4].ToString();
K[5] = IT / LT1Luz6 * LK;
labelK6.Text = "K =" + K[5].ToString();
K[6] = IT / LT1Luz7 * LK;
labelK7.Text = "K =" + K[6].ToString();
K[7] = IT / LT1Luz8 * LK;
labelK8.Text = "K =" + K[7].ToString();
break;
case 9:
//while ((x % LTLuz[0, 0] != 0) || (x % LTLuz[1, 0] != 0) || (x % LTLuz[2, 0] != 0) || (x % LTLuz[3,
0] != 0) || (x % LTLuz[4, 0] != 0) || (x % LTLuz[5, 0] != 0) || (x % LTLuz[6, 0] != 0) || (x % LTLuz[7,
0] != 0) || (x % LTLuz[8, 0] != 0))
// x++;
in1 = LTLuz[0, 0];
in2 = LTLuz[1, 0];
in3 = LTLuz[2, 0];
in4 = LTLuz[3, 0];
in5 = LTLuz[4, 0];
in6 = LTLuz[5, 0];
in7 = LTLuz[6, 0];
in8 = LTLuz[7, 0];
in9 = LTLuz[8, 0];
if (in1 < 1 || in2 < 1)
{
//in2 = in2 * 10;
// in1 = in1 * 10;
// varmenor = 1;
}
while (vi <= in1 || vi <= in2 || vi <= in3 || vi <= in4 || vi <= in5 || vi <= in6 || vi <= in7 || vi
<= in8 || vi <= in9)
{
if (in1 % vi == 0 || in2 % vi == 0 || in3 % vi == 0 || in4 % vi == 0 || in5 % vi == 0 || in6 % vi ==
0 || in7 % vi == 0 || in8 % vi == 0 || in9 % vi == 0)
{
mcm = mcm * vi;
if (in1 % vi == 0)
{
in1 = in1 / vi;
}
}
if (in2 % vi == 0)
{
in2 = in2 / vi;
}
}
}

```

```

}
if (in3 % vi == 0)
{
in3 = in3 / vi;
}
if (in4 % vi == 0)
{
in4 = in4 / vi;
}
if (in5 % vi == 0)
{
in5 = in5 / vi;
}
if (in6 % vi == 0)
{
in6 = in6 / vi;
}
if (in7 % vi == 0)
{
in7 = in7 / vi;
}
if (in8 % vi == 0)
{
in8 = in8 / vi;
}
if (in9 % vi == 0)
{
in9 = in9 / vi;
}
}
else
{
vi = vi + 1;
}
}
if (varmenor == 1)
{
mcm = mcm / 10;
}
LK = mcm;

```

Los resultados de la rigidez, se muestran en este código "label" y en sus caja de repuestas en el panel.

```

K[0] = IT / LT1Luz1 * LK;
labelK1.Text = "K =" + K[0].ToString();
K[1] = IT / LT1Luz2 * LK;
labelK2.Text = "K =" + K[1].ToString();
K[2] = IT / LT1Luz3 * LK;
labelK3.Text = "K =" + K[2].ToString();
K[3] = IT / LT1Luz4 * LK;
labelK4.Text = "K =" + K[3].ToString();
K[4] = IT / LT1Luz5 * LK;
labelK5.Text = "K =" + K[4].ToString();
K[5] = IT / LT1Luz6 * LK;
labelK6.Text = "K =" + K[5].ToString();
K[6] = IT / LT1Luz7 * LK;
labelK7.Text = "K =" + K[6].ToString();
K[7] = IT / LT1Luz8 * LK;
labelK8.Text = "K =" + K[7].ToString();
K[8] = IT / LT1Luz9 * LK;
labelK9.Text = "K =" + K[8].ToString();
break;
}
void calcular()
{
for (int i = 0; i < K.Length; i++)
{

```

```

K[i] = 0;
}
for (int i = 0; i < MFLP.Length; i++)
{
MFLP[i] = 0;
}
for (int i = 0; i < MFLN.Length; i++)
{
MFLN[i] = 0;
}
Rarea1_2= Rarea2_1= Rarea2_2= Rarea3_1= Rarea3_2= Rarea4_1= Rarea4_2= Rarea5_1= Rarea5_2= Rarea6_1=
Rarea6_2= Rarea7_1= Rarea7_2= Rarea8_1= Rarea8_2= Rarea9_1= Rarea9_2 = 0;

/*-----CROSS-----*/
/*-----*/
//2 luces
double FDAB = 0;
double FDBA = 0;
double FDBC = 0;
double FDCB = 0;
//3 luces
double FD CD = 0;
double FDDC = 0;
//4 luces
double FDDE = 0;
double FEDE = 0;
//5 luces
double FDEF = 0;
double FDFE = 0;
//6 luces
double FDFG = 0;
double FDGF = 0;
//7 luces
double FDGH = 0;
double FDHG = 0;
//8 luces
double FDHI = 0;
double FDIH = 0;
//9 luces
double FDIJ = 0;
double FDJI = 0;
double[] FD = { FDAB, FDBA, FDBC, FDCB, FD CD, FDDC, FDDE, FEDE, FDFE, FDEF, FDFG, FDGF, FDGH, FDHG,
FDHI, FDIH, FDIJ, FDJI };
//MPVoladizo es una variable de entrada
string labell1TDV = lbNombreViga.Text;
for (int i = 0; i < FD.Length; i++)
)
{
FD[i] = 0;
}
//2 luces
FDAB = K[0] / (K[0]);
FDBA = K[0] / (K[0] + K[1]);
FDBC = K[1] / (K[0] + K[1]);
FDCB = K[1] / (K[1] + K[2]);
//3 luces
FD CD = K[2] / (K[1] + K[2]);
FDDC = K[2] / (K[2] + K[3]);
//4 luces
FDDE = K[3] / (K[2] + K[3]);
FEDE = K[3] / (K[3] + K[4]);
//5 luces
FDFE = K[4] / (K[3] + K[4]);
FDEF = K[4] / (K[4] + K[5]);
//6 luces
FDFG = K[5] / (K[4] + K[5]);
FDGF = K[5] / (K[5] + K[6]);
//7 luces

```

```

FDGH = K[6] / (K[5] + K[6]);
FDHG = K[6] / (K[6] + K[7]);
//8 luces
FDHI = K[7] / (K[6] + K[7]);
FDIH = K[7] / (K[7] + K[8]);
//9 luces
FDIJ = K[8] / (K[7] + K[8]);
FDJI = K[8] / (K[8]);
double IAB = 0; // VALOR DE VOLADIZO INICIAL SI TIENE
double IBA = 0;
double IBC = 0;
double ICB = 0;
double ICD = 0;
double IDC = 0;
double IDE = 0;
double IED = 0;
double IFE = 0;
double IEF = 0;
double IFG = 0;
double IGF = 0;
double IGH = 0;
double IHG = 0;
double IHI = 0;
double IIH = 0;
double IIJ = 0;
double IJI = 0;
switch (Convert.ToInt32(NL))

{
case 2:
if (labellTDV == "Empotrado Voladizo" || labellTDV == "Empotrado Empotrado" || labellTDV == "Empotrado
Apoyo")
{
FDAB = 0;
}
if (labellTDV == "Voladizo Empotrado" || labellTDV == "Empotrado Empotrado" || labellTDV == "Apoyo
Empotrado")
{
FDCB = 0;
}
break;
case 3:
if (labellTDV == "Empotrado Voladizo" || labellTDV == "Empotrado Empotrado" || labellTDV == "Empotrado
Apoyo")
{
FDAB = 0;
}
if (labellTDV == "Voladizo Empotrado" || labellTDV == "Empotrado Empotrado" || labellTDV == "Apoyo
Empotrado")
{
FDDC = 0;
}
}
break;
case 4:
if (labellTDV == "Empotrado Voladizo" || labellTDV == "Empotrado Empotrado" || labellTDV == "Empotrado
Apoyo")
{
FDAB = 0;
}
if (labellTDV == "Voladizo Empotrado" || labellTDV == "Empotrado Empotrado" || labellTDV == "Apoyo
Empotrado")
{
FDDE = 0;
}
}
break;
case 5:
if (labellTDV == "Empotrado Voladizo" || labellTDV == "Empotrado Empotrado" || labellTDV == "Empotrado
Apoyo")

```

```

{
FDAB = 0;
}
if (labellTDV == "Voladizo Empotrado" || labellTDV == "Empotrado Empotrado" || labellTDV == "Apoyo
Empotrado")
{
FDFE = 0;
}
break;
case 6:
if (labellTDV == "Empotrado Voladizo" || labellTDV == "Empotrado Empotrado" || labellTDV == "Empotrado
Apoyo")
{
FDAB = 0;
}
if (labellTDV == "Voladizo Empotrado" || labellTDV == "Empotrado Empotrado" || labellTDV == "Apoyo
Empotrado")
{
FDGF = 0;
}
break;
case 7:
if (labellTDV == "Empotrado Voladizo" || labellTDV == "Empotrado Empotrado" || labellTDV == "Empotrado
Apoyo")
{
FDAB = 0;
}
if (labellTDV == "Voladizo Empotrado" || labellTDV == "Empotrado Empotrado" || labellTDV == "Apoyo
Empotrado")
{
FDHG = 0;
}
break;
case 8:
if (labellTDV == "Empotrado Voladizo" || labellTDV == "Empotrado Empotrado" || labellTDV == "Empotrado
Apoyo")
{
FDAB = 0;
}
if (labellTDV == "Voladizo Empotrado" || labellTDV == "Empotrado Empotrado" || labellTDV == "Apoyo
Empotrado")
{
FDIH = 0;
}
break;
case 9:
if (labellTDV == "Empotrado Voladizo" || labellTDV == "Empotrado Empotrado" || labellTDV == "Empotrado
Apoyo")
{
FDAB = 0;
}
if (labellTDV == "Voladizo Empotrado" || labellTDV == "Empotrado Empotrado" || labellTDV == "Apoyo
Empotrado")
{
FDJI = 0;
}
break;
/*****
bool VariableIteracion = true;
int NumeroIteraciones = 0;
while (NumeroIteraciones <= 18)
{
if (NL == 1)
{
IAB = -((MPVI + MFLP[0]) * FDAB);
IBA = -((MFLN[0] + MPVF) * FDAB);
IBC = 0;
ICB = 0;
ICD = 0;

```

```

IDC = 0;
IDE = 0;
IED = 0;
IFE = 0;
IEF = 0;
IFG = 0;
IGF = 0;
IGH = 0;
IHG = 0;
IHI = 0;
IIH = 0;
IIJ = 0;
IJI = 0;
}
if (NL == 2)
{
IAB = -((MPVI + MFLP[0]) * FDAB);
IBA = -((MFLN[0] + MFLP[1]) * FDAB);
IBC = -((MFLN[0] + MFLP[1]) * FDAB);
ICB = -((MFLN[1] + MPVF) * FDAB);
ICD = 0;
IDC = 0;
IDE = 0;
IED = 0;
IFE = 0;
IEF = 0;
IFG = 0;
IGF = 0;
IGH = 0;
IHG = 0;
IHI = 0;
IIH = 0;
IIJ = 0;
IJI = 0;
}
if (NL == 3)
{
IAB = -((MPVI + MFLP[0]) * FDAB);
IBA = -((MFLN[0] + MFLP[1]) * FDAB);
IBC = -((MFLN[0] + MFLP[1]) * FDAB);
ICB = -((MFLN[1] + MFLP[2]) * FDAB);
ICD = -((MFLN[1] + MFLP[2]) * FDAB);
IDC = -((MFLN[2] + MPVF) * FDAB);
IDE = 0;
IED = 0;
IFE = 0;
IEF = 0;
IFG = 0;
IGF = 0;
IGH = 0;
IHG = 0;
IHI = 0;
IIH = 0;
IIJ = 0;
IJI = 0;
}
if (NL == 4)
{
IAB = -((MPVI + MFLP[0]) * FDAB);
IBA = -((MFLN[0] + MFLP[1]) * FDAB);
IBC = -((MFLN[0] + MFLP[1]) * FDAB);
ICB = -((MFLN[1] + MFLP[2]) * FDAB);
ICD = -((MFLN[1] + MFLP[2]) * FDAB);
IDC = -((MFLN[2] + MFLP[3]) * FDAB);
IDE = -((MFLN[2] + MFLP[3]) * FDAB);
IED = -((MFLN[3] + MPVF) * FDAB);
IFE = 0;
IEF = 0;
IFG = 0;

```

```

IGF = 0;
IGH = 0;
IHG = 0;
IHI = 0;
IIH = 0;
IIJ = 0;
IJI = 0;
}
if (NL == 5)
{
IAB = -((MPVI + MFLP[0]) * FDAB);
IBA = -((MFLN[0] + MFLP[1]) * FDAB);
IBC = -((MFLN[0] + MFLP[1]) * FDAB);
ICB = -((MFLN[1] + MFLP[2]) * FDAB);
ICD = -((MFLN[1] + MFLP[2]) * FDAB);
IDC = -((MFLN[2] + MFLP[3]) * FDAB);
IDE = -((MFLN[2] + MFLP[3]) * FDAB);
IED = -((MFLN[3] + MFLP[4]) * FDAB);
IFE = -((MFLN[3] + MFLP[4]) * FDAB);
IEF = -((MFLN[4] + MPVF) * FDAB);
IFG = 0;
IGH = 0;
IHG = 0;
IHI = 0;
IIH = 0;
IIJ = 0;
IJI = 0;
}
if (NL == 6)
{
IAB = -((MPVI + MFLP[0]) * FDAB);
IBA = -((MFLN[0] + MFLP[1]) * FDAB);
IBC = -((MFLN[0] + MFLP[1]) * FDAB);
ICB = -((MFLN[1] + MFLP[2]) * FDAB);
ICD = -((MFLN[1] + MFLP[2]) * FDAB);
IDC = -((MFLN[2] + MFLP[3]) * FDAB);
IDE = -((MFLN[2] + MFLP[3]) * FDAB);
IED = -((MFLN[3] + MFLP[4]) * FDAB);
IFE = -((MFLN[3] + MFLP[4]) * FDAB);
IEF = -((MFLN[4] + MFLP[5]) * FDAB);
IFG = -((MFLN[4] + MFLP[5]) * FDAB);
IGF = -((MFLN[5] + MPVF) * FDAB);
IGH = 0;
IHG = 0;
IHI = 0;
IIH = 0;
IIJ = 0;
IJI = 0;
}
if (NL == 7)
{
IAB = -((MPVI + MFLP[0]) * FDAB);
IBA = -((MFLN[0] + MFLP[1]) * FDAB);
IBC = -((MFLN[0] + MFLP[1]) * FDAB);
ICB = -((MFLN[1] + MFLP[2]) * FDAB);
ICD = -((MFLN[1] + MFLP[2]) * FDAB);
IDC = -((MFLN[2] + MFLP[3]) * FDAB);
IDE = -((MFLN[2] + MFLP[3]) * FDAB);
IED = -((MFLN[3] + MFLP[4]) * FDAB);
IFE = -((MFLN[3] + MFLP[4]) * FDAB);
IEF = -((MFLN[4] + MFLP[5]) * FDAB);
IFG = -((MFLN[4] + MFLP[5]) * FDAB);
IGF = -((MFLN[5] + MFLP[6]) * FDAB);
IGH = -((MFLN[5] + MFLP[6]) * FDAB);
IHG = -((MFLN[6] + MPVF) * FDAB);
IHI = 0;
IIH = 0;
IIJ = 0;
}

```

```

IJI = 0;
}
if (NL == 8)
{
IAB = -((MPVI + MFLP[0]) * FDAB);
IBA = -((MFLN[0] + MFLP[1]) * FDAB);
IBC = -((MFLN[0] + MFLP[1]) * FDAB);
ICB = -((MFLN[1] + MFLP[2]) * FDAB);
ICD = -((MFLN[1] + MFLP[2]) * FDAB);
IDC = -((MFLN[2] + MFLP[3]) * FDAB);
IDE = -((MFLN[2] + MFLP[3]) * FDAB);
IED = -((MFLN[3] + MFLP[4]) * FDAB);
IFE = -((MFLN[3] + MFLP[4]) * FDAB);
IEF = -((MFLN[4] + MFLP[5]) * FDAB);
IFG = -((MFLN[4] + MFLP[5]) * FDAB);
IGF = -((MFLN[5] + MFLP[6]) * FDAB);
IGH = -((MFLN[5] + MFLP[6]) * FDAB);
IHG = -((MFLN[6] + MFLP[7]) * FDAB);
IHI = -((MFLN[6] + MFLP[7]) * FDAB);
IIH = -((MFLN[7] + MPVF) * FDAB);
IJJ = 0;
IJI = 0;
}
if (NL == 9)
{
IAB = -((MPVI + MFLP[0]) * FDAB);
IBA = -((MFLN[0] + MFLP[1]) * FDAB);
IBC = -((MFLN[0] + MFLP[1]) * FDAB);
ICB = -((MFLN[1] + MFLP[2]) * FDAB);
ICD = -((MFLN[1] + MFLP[2]) * FDAB);
IDC = -((MFLN[2] + MFLP[3]) * FDAB);
IDE = -((MFLN[2] + MFLP[3]) * FDAB);
IED = -((MFLN[3] + MFLP[4]) * FDAB);
IFE = -((MFLN[3] + MFLP[4]) * FDAB);
IEF = -((MFLN[4] + MFLP[5]) * FDAB);
IFG = -((MFLN[4] + MFLP[5]) * FDAB);
IGF = -((MFLN[5] + MFLP[6]) * FDAB);
IGH = -((MFLN[5] + MFLP[6]) * FDAB);
IHG = -((MFLN[6] + MFLP[7]) * FDAB);
IHI = -((MFLN[6] + MFLP[7]) * FDAB);
IIH = -((MFLN[7] + MFLP[8]) * FDAB);
IJJ = -((MFLN[7] + MFLP[8]) * FDAB);
IJI = -((MFLN[8] + MPVF) * FDAB);
}
double[] IT = { IAB, IBA, IBC, ICB, ICD, IDC, IDE, IED, IFE, IEF, IFG, IGF, IGH, IHG, IHI, IIH, IJJ,
IJI };
MPVI = 0;
MPVF = 0;
for (int i = 0; i < 9; i++)
{
FD[i * 2] = FD[i * 2] + MFLP[i];
FD[(i * 2) + 1] = FD[(i * 2) + 1] + MFLN[i];
MFLN[i] = (IT[i * 2] / 2);
MFLP[i] = (IT[(i * 2) + 1] / 2);
}
int VariableConfirmacion = 0;
for (int i = 0; i < IT.Length; i++)
{
FD[i] = FD[i] + IT[i];
if (IT[i] < 0.0009)
{
VariableConfirmacion++;
}
}
if (VariableConfirmacion == IT.Length)
{
VariableIteracion = false;
//MessageBox.Show("VariableIteracion Falso");
}
}
}

```

```

NumeroIteraciones++;
//if (NumeroIteraciones > 15) { VariableIteracion = false; }
//MessageBox.Show("Iteracion " + NumeroIteraciones + " ; " + FD[0] + " , " + FD[1] + " , " + FD[2] +
" , " + FD[3] + " , " + FD[4] + " , " + FD[5] + " , " + FD[6] + " , " + FD[7] + " , " + FD[8] + " , "
+ FD[9] + " , " + FD[10] + " , " + FD[11] + " , " + FD[12] + " , " + FD[13] + " , " + FD[14] + " , "
+ FD[15] + " , " + FD[16] + " , " + FD[17]);
}
//MessageBox.Show("SALIO" + " ; " + FD[0] + " , " + FD[1] + " , " + FD[2] + " , " + FD[3] + " , " +
FD[4] + " , " + FD[5] + " , " + FD[6] + " , " + FD[7] + " , " + FD[8] + " , " + FD[9] + " , " + FD[10]
+ " , " + FD[11] + " , " + FD[12] + " , " + FD[13] + " , " + FD[14] + " , " + FD[15] + " , " + FD[16]
+ " , " + FD[17]);

labelMAB.Text = "MAB= " + FD[0].ToString("0.000")+ " "+numerador+"/"+denominador;
labelMBA.Text = "MBA= " + FD[1].ToString("0.000") + " " + numerador + "/" + denominador;
labelMBC.Text = "MBC= " + FD[2].ToString("0.000") + " " + numerador + "/" + denominador;
labelMCB.Text = "MCB= " + FD[3].ToString("0.000") + " " + numerador + "/" + denominador;
labelMCD.Text = "MCD= " + FD[4].ToString("0.000") + " " + numerador + "/" + denominador;
labelMDC.Text = "MDC= " + FD[5].ToString("0.000") + " " + numerador + "/" + denominador;
labelMDE.Text = "MDE= " + FD[6].ToString("0.000") + " " + numerador + "/" + denominador;
labelMED.Text = "MED= " + FD[7].ToString("0.000") + " " + numerador + "/" + denominador;
labelMEF.Text = "MEF= " + FD[8].ToString("0.000") + " " + numerador + "/" + denominador;
labelMFE.Text = "MFE= " + FD[9].ToString("0.000") + " " + numerador + "/" + denominador;
labelMFG.Text = "MFG= " + FD[10].ToString("0.000") + " " + numerador + "/" + denominador;
labelMGF.Text = "MGF= " + FD[11].ToString("0.000") + " " + numerador + "/" + denominador;
labelMGH.Text = "MGH= " + FD[12].ToString("0.000") + " " + numerador + "/" + denominador;
labelMHG.Text = "MHG= " + FD[13].ToString("0.000") + " " + numerador + "/" + denominador;
labelMHI.Text = "MHI= " + FD[14].ToString("0.000") + " " + numerador + "/" + denominador;
labelMIH.Text = "MIH= " + FD[15].ToString("0.000") + " " + numerador + "/" + denominador;
labelMIJ.Text = "MIJ= " + FD[16].ToString("0.000") + " " + numerador + "/" + denominador;
labelMJI.Text = "MJI= " + FD[17].ToString("0.000") + " " + numerador + "/" + denominador;
/*----- Cross -----*/
/*----- Calculo de Apoyo Columna unica-----*/
double[] AC = { ApoyoColumnaRab, ApoyoColumnaRba, ApoyoColumnaRbc, ApoyoColumnaRcb, ApoyoColumnaRcd,
ApoyoColumnaRdc, ApoyoColumnaRde, ApoyoColumnaRed, ApoyoColumnaRef, ApoyoColumnaRfe, ApoyoColumnaRfg,
ApoyoColumnaRgf, ApoyoColumnaRgh, ApoyoColumnaRhg, ApoyoColumnaRhi, ApoyoColumnaRih, ApoyoColumnaRij,
ApoyoColumnaRji };
for (int i = 0; i < NL; i++)
{
for (int j = 0; j < NCL[i]; j++)
{
if (LuzCarga[i, j] == 1)
{
ApoyoColumna = (Wu * LTLuz[i, j] * LTLuz[i, j] / 2) / LTLuz[i, j];
AC[2 * i + 1] = ApoyoColumna + AC[2 * i + 1];
AC[2 * i] = ApoyoColumna + AC[2 * i];
}
if (LuzCarga[i, j] == 2)
{
ApoyoColumna = (Wu * LTLuz[i, j] / 6);
AC[2 * i + 1] = ApoyoColumna + AC[2 * i + 1];
AC[2 * i] = Math.Abs(ApoyoColumna - (Wu * LTLuz[i, j] / 2) + AC[2 * i]);
}
if (LuzCarga[i, j] == 3)
{
ApoyoColumna = (Wu * LTLuz[i, j] / 3);
AC[2 * i + 1] = ApoyoColumna + AC[2 * i + 1];
AC[2 * i] = Math.Abs(ApoyoColumna - (Wu * LTLuz[i, j] / 2) + AC[2 * i]);
}
if (LuzCarga[i, j] == 4)
{
ApoyoColumna = ((LTLuz[i, j] * LFP[i, j] / 2) / LTLuz[i, j]);
AC[2 * i + 1] = ApoyoColumna + AC[2 * i + 1];
AC[2 * i] = ApoyoColumna + AC[2 * i];
}
if (LuzCarga[i, j] == 5)
{
AC[2 * i + 1] = ((LFP[i, j] * Luzla[i, j]) / (Luzla[i, j] + Luzlb[i, j])) + AC[2 * i + 1];
AC[2 * i] = ((LFP[i, j] * Luzlb[i, j]) / (Luzla[i, j] + Luzlb[i, j])) + AC[2 * i];
}
}
}
}

```

```

}
}
labelRab.Text = "Rab:" + AC[0].ToString("0.000")+ " " + numerador;
labelRba.Text = "Rba:" + AC[1].ToString("0.000") + " " + numerador;
labelRbc.Text = "Rbc:" + AC[2].ToString("0.000") + " " + numerador;
labelRcb.Text = "Rcb:" + AC[3].ToString("0.000") + " " + numerador;
labelRcd.Text = "Rcd:" + AC[4].ToString("0.000") + " " + numerador;
labelRdc.Text = "Rdc:" + AC[5].ToString("0.000") + " " + numerador;
labelRde.Text = "Rde:" + AC[6].ToString("0.000") + " " + numerador;
labelRed.Text = "Red:" + AC[7].ToString("0.000") + " " + numerador;
labelRef.Text = "Ref:" + AC[8].ToString("0.000") + " " + numerador;
labelRfe.Text = "Rfe:" + AC[9].ToString("0.000") + " " + numerador;
labelRfg.Text = "Rfg:" + AC[10].ToString("0.000") + " " + numerador;
labelRgf.Text = "Rgf:" + AC[11].ToString("0.000") + " " + numerador;
labelRgh.Text = "Rgh:" + AC[12].ToString("0.000") + " " + numerador;
labelRhg.Text = "Rhg:" + AC[13].ToString("0.000") + " " + numerador;
labelRhi.Text = "Rhi:" + AC[14].ToString("0.000") + " " + numerador;
labelRih.Text = "Rih:" + AC[15].ToString("0.000") + " " + numerador;
labelRij.Text = "Rij:" + AC[16].ToString("0.000") + " " + numerador;
labelRji.Text = "Rji:" + AC[17].ToString("0.000") + " " + numerador;

ApoyoColumnaRab = 0;
ApoyoColumnaRba = 0;
ApoyoColumnaRbc = 0;
ApoyoColumnaRcb = 0;
ApoyoColumnaRcd = 0;
ApoyoColumnaRdc = 0;
ApoyoColumnaRde = 0;
ApoyoColumnaRed = 0;
ApoyoColumnaRef = 0;
ApoyoColumnaRfe = 0;
ApoyoColumnaRfg = 0;
ApoyoColumnaRgf = 0;
ApoyoColumnaRgh = 0;
ApoyoColumnaRhg = 0;
ApoyoColumnaRhi = 0;
ApoyoColumnaRih = 0;
ApoyoColumnaRij = 0;
ApoyoColumnaRji = 0;

double ACA = ((FD[0] + FD[1]) / LT1Luz1) + RVI + AC[0];
double ACB = -((FD[0] + FD[1]) / LT1Luz1) + ((FD[2] + FD[3]) / LT1Luz2) + AC[1] + AC[2];
//double ACCfinal = -((FD[2] + FD[3]) / LT1Luz2) + AC[2] + RVF;//para dos luces
double ACC = -((FD[2] + FD[3]) / LT1Luz2) + ((FD[4] + FD[5]) / LT1Luz3) + AC[3] + AC[4];
double ACD = -((FD[4] + FD[5]) / LT1Luz3) + ((FD[6] + FD[7]) / LT1Luz4) + AC[5] + AC[6];
double ACE = -((FD[6] + FD[7]) / LT1Luz4) + ((FD[8] + FD[9]) / LT1Luz5) + AC[7] + AC[8];
double ACF = -((FD[8] + FD[9]) / LT1Luz5) + ((FD[10] + FD[11]) / LT1Luz6) + AC[9] + AC[10];
double ACG = -((FD[10] + FD[11]) / LT1Luz6) + ((FD[12] + FD[13]) / LT1Luz7) + AC[11] + AC[12];
double ACH = -((FD[12] + FD[13]) / LT1Luz7) + ((FD[14] + FD[15]) / LT1Luz8) + AC[13] + AC[14];
double ACI = -((FD[14] + FD[15]) / LT1Luz8) + ((FD[16] + FD[17]) / LT1Luz9) + AC[15] + AC[16];
double ACfinal = 0;
for (int i = 0; i < 9; i++)
{
if ((NL - 1) == i)
{
ACfinal = -((FD[i * 2] + FD[(i * 2) + 1]) / LT1Luz[i, 0]) + AC[i * 2 + 1] + RVF;
}
}

//FD momentos
labelRVI.Text = "Rvol inicial =" + RVI.ToString("0.00") + " " + numerador;
labelRVF.Text = "Rvol final =" + RVF.ToString("0.00") + " " + numerador;
labelApoyoColumnaA.Text = "ApoyoColumnaA=" + ACA.ToString("0.00")+ " "+numerador;
labelApoyoColumnaB.Text = "ApoyoColumnaB=" + ACB.ToString("0.00") + " " + numerador;
labelApoyoColumnaC.Text = "ApoyoColumnaC=" + ACC.ToString("0.00") + " " + numerador;
labelApoyoColumnaD.Text = "ApoyoColumnaD=" + ACD.ToString("0.00") + " " + numerador;
labelApoyoColumnaE.Text = "ApoyoColumnaE=" + ACE.ToString("0.00") + " " + numerador;
labelApoyoColumnaF.Text = "ApoyoColumnaF=" + ACF.ToString("0.00") + " " + numerador;
labelApoyoColumnaG.Text = "ApoyoColumnaG=" + ACG.ToString("0.00") + " " + numerador;

```

```

labelApoyoColumnaH.Text = "ApoyoColumnaH= " + ACH.ToString("0.00") + " " + numerador;
labelApoyoColumnaI.Text = String.Format("ApoyoColumnaI= {0:0.00} {1}", ACI, numerador);
labelApoyoColumnaFinal.Text = "ApoyoColumnaI= " + ACfinal.ToString("0.00") + " " + numerador;

/// <summary>
/// grafica 1
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void barButtonItem3_ItemClick(object sender, ItemClickEventArgs e)
{
    try
    {
        plResultado.Visible = false;
        plGráfica1.Dock = DockStyle.Fill;
        pnlCarga.Visible = false;
        paneldatosI.Visible = false;
        plGráfica1.Visible = true;
        plGráfica2.Visible = false;
        plRefuerzo.Visible = false;
        pldiseñocortante.Visible = false;
    }
    catch (Exception)
    {
        MessageBox.Show("Error al cargar Gráfica", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

/*****Gráficas*****/
/*****/
double[] ACT = { ACA, ACB, ACC, ACD, ACE, ACF, ACG, ACH, ACI };
puntoy = 0;
puntoy = ACA;
Acolum = 1;
string[] SerieTipo = { "Luz1", "Luz2", "Luz3", "Luz4", "Luz5", "Luz6", "Luz7", "Luz8", "Luz9" };
string[] SerieTipo2 = { "LC1", "LC2", "LC3", "LC4", "LC5", "LC6", "LC7", "LC8", "LC9" };**
/*****Gráfica de cortante*****/
/*****/
chart1.Titles.Clear();
chart1.Series.Clear();
chart1.Titles.Add("Gráfica de cortante");
for (int i = 0; i < SerieTipo.Length; i++)
{
    this.chart1.Series.Add(SerieTipo[i]).BorderWidth = 3;
}
for (int i = 0; i < SerieTipo2.Length; i++)
{
    this.chart1.Series.Add(SerieTipo2[i]).BorderWidth = 3;
}
// chart1.Series[SerieTipo[0]].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
// chart1.Series[SerieTipo[0]].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
// chart1.Series[SerieTipo[0]].LabelBackColor = System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
// chart1.Series[SerieTipo[0]].LabelBorderColor = System.Drawing.Color.Black;
// chart1.Series[SerieTipo[0]].Color = System.Drawing.Color.Blue;
chart1.ChartAreas[0].AxisX.IsMarginVisible = false;
//////////*****
this.chart1.Series.Add("cortante").BorderWidth = 3;
chart1.Series["cortante"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series["cortante"].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series["cortante"].LabelBackColor = System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series["cortante"].LabelBordeColor = System.Drawing.Color.Black;
chart1.Series["cortante"].Color = System.Drawing.Color.Blue;
/*****v**Gráfica de momento*****/
/*****/
chart2.Titles.Clear();
chart2.Series.Clear();

```



```

this.chart1.Series[SerieTipo[0]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
}
this.chart1.Series[SerieTipo[0]].Points.AddXY(PuntoCortanteX2, PuntoCortanteY2);
}*/
}
}
#endregion
double ValorInicialX = 0;
double ValorInicialY = 0;
double PuntoCortanteX1V = 0, PuntoCortanteY1V = 0, PuntoCortanteX1F = 0, PuntoCortanteY1F = 0;
double VariableCargas = 0;
puntomomentox = 0;
double FPParcial = 0;
/*****/
this.chart1.Series["cortante"].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteY1 = punttoy + ACfinal;
punttoy = PuntoCortanteY1;
this.chart1.Series["cortante"].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
if (labell1TDV == "Apoyo Voladizo" || labell1TDV == "Voladizo Voladizo" || labell1TDV == "Empotrado
Voladizo")
{
if (numericUpDownTCVF.Value == 1)
{
this.chart1.Series.Add("Voladizofinal0").BorderWidth = 3;
chart1.Series["Voladizofinal0"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series["Voladizofinal0"].LabelBackColor = System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series["Voladizofinal0"].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series["Voladizofinal0"].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series["Voladizofinal0"].Color = System.Drawing.Color.Blue;
this.chart1.Series["Voladizofinal0"].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteX1V = Convert.ToDouble(numericUpDownLTVF.Value);
PuntoCortanteX1 = PuntoCortanteX1 + PuntoCortanteX1V;
PuntoCortanteY1 = PuntoCortanteY1 + (- Wu * PuntoCortanteX1V);
this.chart1.Series["Voladizofinal0"].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
ValorInicialY = puntoMomentoY1 = PuntoCortanteY1;
//VarAreaX1 = PuntoCortanteX1;
//VarAreaY1 = PuntoCortanteY1;
}
if (numericUpDownTCVF.Value == 2)
{
this.chart1.Series.Add("Voladizofinal0").BorderWidth = 3;
this.chart1.Series.Add("Voladizofinal1").BorderWidth = 3;
chart1.Series["Voladizofinal0"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
chart1.Series["Voladizofinal0"].LabelBackColor = System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series["Voladizofinal0"].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series["Voladizofinal0"].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series["Voladizofinal0"].Color = System.Drawing.Color.Blue;
// chart1.ChartAreas["VoladizoInicial0"].AxisX.IsMarginVisible = false;
this.chart1.Series["Voladizofinal0"].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteX1V = Convert.ToDouble(numericUpDownLTVF.Value);
//PuntoCortanteX1 = PuntoCortanteX1 + PuntoCortanteX1V;
PuntoCortanteY1F = PuntoCortanteY1 / 2;
PuntoCortanteX1F = PuntoCortanteX1 + PuntoCortanteX1V / 4;
this.chart1.Series["Voladizofinal0"].Points.AddXY(PuntoCortanteX1F, PuntoCortanteY1F);
PuntoCortanteX1V = Convert.ToDouble(numericUpDownLTVF.Value);
PuntoCortanteX1F = PuntoCortanteX1 + PuntoCortanteX1V;
PuntoCortanteY1F = (PuntoCortanteY1 + (- Wu * PuntoCortanteX1V / 2));
this.chart1.Series["Voladizofinal0"].Points.AddXY(PuntoCortanteX1F, PuntoCortanteY1F);
}
if (numericUpDownTCVF.Value == 3)
{
this.chart1.Series.Add("Voladizofinal0").BorderWidth = 3;
this.chart1.Series.Add("Voladizofinal1").BorderWidth = 3;
chart1.Series["Voladizofinal0"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

```

```

chart1.Series["Voladizofinal0"].LabelBackColor          =          System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series["Voladizofinal0"].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series["Voladizofinal0"].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series["Voladizofinal0"].Color = System.Drawing.Color.Blue;
// chart1.ChartAreas["Voladizoinitial0"].AxisX.IsMarginVisible = false;
this.chart1.Series["Voladizofinal0"].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteX1V = Convert.ToDouble(numericUpDownLTVF.Value);
//PuntoCortanteX1 = PuntoCortanteX1 + PuntoCortanteX1V;
PuntoCortanteY1F = PuntoCortanteY1 / 2;
PuntoCortanteX1F = PuntoCortanteX1 + ((PuntoCortanteX1V / 4)*3);
this.chart1.Series["Voladizofinal0"].Points.AddXY(PuntoCortanteX1F, PuntoCortanteY1F);
PuntoCortanteX1V = Convert.ToDouble(numericUpDownLTVF.Value);
PuntoCortanteX1F = PuntoCortanteX1 + PuntoCortanteX1V;
PuntoCortanteY1F = (PuntoCortanteY1 + (-Wu * PuntoCortanteX1V / 2));
this.chart1.Series["Voladizofinal0"].Points.AddXY(PuntoCortanteX1F, PuntoCortanteY1F);
}

```

En esta estructura define el tipo de gráfica, si es lineal para la Gráfica cortante o curva para la gráfica de momento, define las variables como áreas, identifica la cantidad de luces, tipo de carga,

```

if (numericUpDownTCVI.Value == 4 || numericUpDownTCVI.Value == 5)
{
chart1.Series[SerieTipo[1]].ChartType
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series[SerieTipo[1]].LabelBackColor          =          System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series[SerieTipo[1]].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series[SerieTipo[1]].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series[SerieTipo[1]].Color = System.Drawing.Color.Blue;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1V, PuntoCortanteY1V);
PuntoCortanteX2 =Convert.ToDouble(numericUpDownLaVI.Value) + ValorInicialX;
// PuntoCortanteY2 = ((Convert.ToDouble(numericUpDownLaVI.Value) * m) + PuntoCortanteY1V);
PuntoCortanteY2 = PuntoCortanteY1V;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX2, PuntoCortanteY2);
PuntoCortanteX1 = PuntoCortanteX2;
PuntoCortanteY1 = PuntoCortanteY2 - (Convert.ToDouble(numericUpDownFPVI.Value));
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
// PuntoCortanteY1 = PuntoCortanteY1 - (Luzlb[1, i] * Wu);
PuntoCortanteX1 = PuntoCortanteX1 + Convert.ToDouble(numericUpDownLbVI.Value);
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteY1 = PuntoCortanteY1 + ACA;
puntoy = PuntoCortanteY1;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
ValorInicialX = puntoMomentoX1 = PuntoCortanteX1;
ValorInicialY = puntoMomentoY1 = PuntoCortanteY1;
VarAreaX1 = PuntoCortanteX1;
VarAreaY1 = PuntoCortanteY1;
PuntoCortanteY2 = 0;
PuntoCortanteX2 = 0;
}
}
}
else
{
chart1.Series[SerieTipo[1]].ChartType
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series[SerieTipo[1]].LabelBackColor          =          System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series[SerieTipo[1]].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series[SerieTipo[1]].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series[SerieTipo[1]].Color = System.Drawing.Color.Blue;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
VarAreaX1 = PuntoCortanteX1;
VarAreaY1 = PuntoCortanteY1;
}
}
}

```

```

if (NCL[1] != 1)
{
if (LuzCarga[1, i + 1] == 4 || LuzCarga[1, i + 1] == 5)
{
PuntoCortanteX2 = Luzla[1, i + 1] + ValorInicialX;
PuntoCortanteY2 = ((Luzla[1, i + 1] * m) + PuntoCortanteY1);
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX2, PuntoCortanteY2);
}
}
}
else
{
FPParcial = FPParcial + LFP[1, i];
PuntoCortanteX1 = Luzla[1, i] + ValorInicialX;
PuntoCortanteY1 = PuntoCortanteY2 - LFP[1, i];
//PuntoCortanteY1 = (Luzla[1, i] * m + ACT[1]) - FPParcial + ValorInicialY;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
if ((NCL[0] + NCL[1] + NCL[2] + NCL[3] + NCL[4] + NCL[5] + NCL[6] + NCL[7] + NCL[8]) != VariableCargas)
{
if (NCL[1] != i + 1)
{
if (NCL[1] == 6)
{
PuntoCortanteX2 = Luzla[1, i] + ValorInicialX;
PuntoCortanteY2 = (Luzla[1, i] * m + ACT[1]) - FPParcial + ValorInicialY;
}
else
{
PuntoCortanteX2 = Luzla[1, i + 1] + ValorInicialX;
PuntoCortanteY2 = PuntoCortanteY1 - ((Luzla[1, i + 1] - Luzla[1, i]) * Wu);
}
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX2, PuntoCortanteY2);
}
}
}
if (NCL[1] == i + 1 && (NCL[0] + NCL[1] + NCL[2] + NCL[3] + NCL[4] + NCL[5] + NCL[6] + NCL[7] + NCL[8])
!= VariableCargas)
{
PuntoCortanteX1 = LTLuz[1, 0] + ValorInicialX;
if (NCL[1] == 1 && LuzCarga[1, i] != 1)
{
PuntoCortanteY1 = ((Luzla[1, i + 1] * m) + ACT[1] - FPParcial + ValorInicialY);
}
else
{
if (NCL[1] == 1)
{
PuntoCortanteY1 = puntoy - (LTLuz[1, 0] * Wu);
puntoy = PuntoCortanteY1;
}
else
{
if (NCL[1] == i + 1)
{
if (1 > 0)
{
PuntoCortanteY1 = PuntoCortanteY1 - (Luzlb[1, i] * Wu);
puntoy = PuntoCortanteY1;
}
else
{
PuntoCortanteY1 = -FPParcial + ACT[1] - (LTLuz[1, i] * Wu);
puntoy = PuntoCortanteY1;
}
}
else
{
PuntoCortanteY1 = PuntoCortanteY1 - (Luzlb[1, i] * Wu);
puntoy = PuntoCortanteY1;
}
}
}
}
}

```

```

}
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
VarAreaX2 = PuntoCortanteX1;
VarAreaY2 = PuntoCortanteY1;
if (Acolum < 9)
{
if (ACT[Acolum] >= 0)
{
PuntoCortanteY1 = puntoy + ACT[Acolum];
vmax = PuntoCortanteY1;
if (Acolum < 8)
{
Acolum++;
}
puntoy = PuntoCortanteY1;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
}
if (LuzCarga[1 + 1, 0] != 1)
{
PuntoCortanteX1 = LTLuz[1, 0] + ValorInicialX;
if (NCL[1] == 1)
{
// PuntoCortanteY1 = PuntoCortanteY1 + ACT[1 + 1];
//PuntoCortanteY2 = PuntoCortanteY1+ ACfinal;
}
else
// PuntoCortanteY1 = PuntoCortanteY1 + ACT[1 + 1];
}
// this.chart1.Series[SerieTipo[0]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
}
}
if ((NCL[0] + NCL[1] + NCL[2] + NCL[3] + NCL[4] + NCL[5] + NCL[6] + NCL[7] + NCL[8]) == VariableCargas)
{
PuntoCortanteX1 = LTLuz[1, 0] + ValorInicialX;
PuntoCortanteX2 = LTLuz[1, 0] + ValorInicialX;
if (NCL[1] == 1)
{
PuntoCortanteY1 = puntoy - (LTLuz[1, 0] * Wu);
puntoy = PuntoCortanteY1;
}
else
{
if (NCL[1] > 1)
{
//PuntoCortanteY1 = PuntoCortanteY1 - (Luzla[1, i] * Wu);
if (Luzla[1, i] <= Luzlb[1, i])
{
if (NCL[1] == i + 1)
{
PuntoCortanteY1 = PuntoCortanteY1 - (Luzlb[1, i] * Wu);
puntoy = PuntoCortanteY1;
}
else
{
PuntoCortanteY1 = PuntoCortanteY1 - (Luzla[1, i] * Wu);
puntoy = PuntoCortanteY1;
}
}
}
if (Luzla[1, i] > Luzlb[1, i])
{
if (NCL[1] == i + 1)
{
PuntoCortanteY1 = -FPParcial + ACT[0] - (LTLuz[1, i] * Wu);
}
else
{
PuntoCortanteY1 = PuntoCortanteY1 - (Luzlb[1, i] * Wu);
}
}
}
}
}

```

```

}
else
{
PuntoCortanteY1 = PuntoCortanteY1 - (Luzla[1, i] * Wu);
}
}
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1); //segundo valor para
todas las Gráficas
VarAreaX2 = PuntoCortanteX1;
VarAreaY2 = PuntoCortanteY1;
if (Acolum < 9)
{
if (ACT[Acolum] >= 0)
{
if (Acolum < 8)
{
PuntoCortanteY1 = puntoy + ACT[Acolum];
vmax = PuntoCortanteY1;
}
Acolum++;
}
puntoy = PuntoCortanteY1;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
}
ValorInicialX = PuntoCortanteX1;
}
if (NCL[1] - 1 == i)
{
ValorInicialX = LTLuz[1, 0] + ValorInicialX;
ValorInicialY = PuntoCortanteY1;
puntoMomentoY1 = ValorInicialY;
}
ValorY0 = (-VarAreaY1) / m;
if (ValorY0 > LTLuz[1, i])
{
AreaX1 = ((LTLuz[1, i] * (VarAreaY1 - PuntoCortanteY1)) / 2) + (PuntoCortanteY1 * LTLuz[1, i]);
AreaX2 = 0;
AreaT = AreaX1 + AreaX2;
}
else
{
// area1p = ACA * (ACA / Wu) / 2;
AreaX1 = ((VarAreaY1 * ValorY0) / 2);
// AreaX1 = area1p;
AreaX2 = (VarAreaY2 * (LTLuz[1, i] - ValorY0) / 2);
AreaT = AreaX1 + AreaX2;
}
.

else
{
if (1 == 0)
{
if (labellTDV == "Voladizo Apoyo" || labellTDV == "Voladizo Voladizo" || labellTDV == "Voladizo
Empotrado")
{
if (numericUpDownTCVI.Value == 1)
{
this.chart1.Series.Add("VoladizoInicial0").BorderWidth = 3;
this.chart1.Series.Add("VoladizoInicial1").BorderWidth = 3;
chart1.Series["VoladizoInicial0"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series["VoladizoInicial0"].LabelBackColor = System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
}
}
}
}

```

```

chart1.Series["Voladizo inicial0"].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series["Voladizo inicial0"].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series["Voladizo inicial0"].Color = System.Drawing.Color.Blue;
this.chart1.Series["Voladizo inicial0"].Points.AddXY(PuntoCortanteX1V, PuntoCortanteY1V);
PuntoCortanteX1V = Convert.ToDouble(numericUpDownLTVI.Value);
PuntoCortanteY1V = -Wu * PuntoCortanteX1V;
this.chart1.Series["Voladizo inicial0"].Points.AddXY(PuntoCortanteX1V, PuntoCortanteY1V);
PuntoCortanteX1=ValorInicialX = PuntoCortanteX1V;
PuntoCortanteY1 = PuntoCortanteY1V + ACA;
puntoy = PuntoCortanteY1;
this.chart1.Series["Voladizo inicial0"].Points.AddXY(PuntoCortanteX1V, PuntoCortanteY1);
ValorInicialY = puntoMomentoY1 = PuntoCortanteY1;
VarAreaX1 = PuntoCortanteX1;
VarAreaY1 = PuntoCortanteY1;
}
if (numericUpDownTCVI.Value == 2)
{
this.chart1.Series.Add("Voladizo inicial0").BorderWidth = 3;
this.chart1.Series.Add("Voladizo inicial1").BorderWidth = 3;
chart1.Series["Voladizo inicial0"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
chart1.Series["Voladizo inicial0"].LabelBackColor = System.Drawing.Color.FromArgb(50, System.Drawing.Color.LightGreen);
chart1.Series["Voladizo inicial0"].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series["Voladizo inicial0"].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series["Voladizo inicial0"].Color = System.Drawing.Color.Blue;
// chart1.ChartAreas["Voladizo inicial0"].AxisX.IsMarginVisible = false;
this.chart1.Series["Voladizo inicial0"].Points.AddXY(PuntoCortanteX1V, PuntoCortanteY1V);
PuntoCortanteX1V = Convert.ToDouble(numericUpDownLTVI.Value);
PuntoCortanteY1V = (-Wu * PuntoCortanteX1V / 2)/2;
PuntoCortanteX1V = PuntoCortanteX1V / 4; ;
this.chart1.Series["Voladizo inicial0"].Points.AddXY(PuntoCortanteX1V, PuntoCortanteY1V);
PuntoCortanteX1V = Convert.ToDouble(numericUpDownLTVI.Value);
PuntoCortanteY1V = -Wu * PuntoCortanteX1V / 2;
this.chart1.Series["Voladizo inicial0"].Points.AddXY(PuntoCortanteX1V, PuntoCortanteY1V);
chart1.Series["Voladizo inicial1"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series["Voladizo inicial1"].LabelBackColor = System.Drawing.Color.FromArgb(50, System.Drawing.Color.LightGreen);
chart1.Series["Voladizo inicial1"].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series["Voladizo inicial1"].Color = System.Drawing.Color.Blue;
this.chart1.Series["Voladizo inicial1"].Points.AddXY(PuntoCortanteX1V, PuntoCortanteY1V);
PuntoCortanteX1 = PuntoCortanteX1V;
PuntoCortanteY1 = PuntoCortanteY1V + ACA;
puntoy = PuntoCortanteY1;
this.chart1.Series["Voladizo inicial1"].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
ValorInicialX = puntoMomentoX1 = PuntoCortanteX1;
ValorInicialY = puntoMomentoY1 = PuntoCortanteY1;
////
VarAreaX1 = PuntoCortanteX1;
VarAreaY1 = PuntoCortanteY1;
}
if (numericUpDownTCVI.Value == 3)
{
this.chart1.Series.Add("Voladizo inicial0").BorderWidth = 3;
this.chart1.Series.Add("Voladizo inicial1").BorderWidth = 3;
chart1.Series["Voladizo inicial0"].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
chart1.Series["Voladizo inicial0"].LabelBackColor = System.Drawing.Color.FromArgb(50, System.Drawing.Color.LightGreen);
chart1.Series["Voladizo inicial0"].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series["Voladizo inicial0"].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series["Voladizo inicial0"].Color = System.Drawing.Color.Blue;
// chart1.ChartAreas["Voladizo inicial0"].AxisX.IsMarginVisible = false;
this.chart1.Series["Voladizo inicial0"].Points.AddXY(PuntoCortanteX1V, PuntoCortanteY1V);
PuntoCortanteX1V = Convert.ToDouble(numericUpDownLTVI.Value);
PuntoCortanteY1V = (-Wu * PuntoCortanteX1V / 2) / 2;
PuntoCortanteX1V = (PuntoCortanteX1V / 4)*3;
this.chart1.Series["Voladizo inicial0"].Points.AddXY(PuntoCortanteX1V, PuntoCortanteY1V);

```

```

PuntoCortanteX1V = Convert.ToDouble(numericUpDownLTVI.Value);
PuntoCortanteY1V = -Wu * PuntoCortanteX1V / 2;
this.chart1.Series["VoladizoInicial0"].Points.AddXY(PuntoCortanteX1V, PuntoCortanteY1V);
chart1.Series["VoladizoInicial1"].ChartType
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series["VoladizoInicial1"].LabelBackColor = System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series["VoladizoInicial1"].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series["VoladizoInicial1"].Color = System.Drawing.Color.Blue;
this.chart1.Series["VoladizoInicial1"].Points.AddXY(PuntoCortanteX1V, PuntoCortanteY1V);
PuntoCortanteX1 = PuntoCortanteX1V;
PuntoCortanteY1 = PuntoCortanteY1V + ACA;
puntoy = PuntoCortanteY1;
this.chart1.Series["VoladizoInicial1"].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
ValorInicialX = puntoMomentoX1 = PuntoCortanteX1;
ValorInicialY = puntoMomentoY1 = PuntoCortanteY1;
////
VarAreaX1 = PuntoCortanteX1;
VarAreaY1 = PuntoCortanteY1;
}
if (numericUpDownTCVI.Value == 4 || numericUpDownTCVI.Value == 5)
{
this.chart1.Series.Add("VoladizoInicial0").BorderWidth = 3;
chart1.Series["VoladizoInicial0"].ChartType
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series["VoladizoInicial0"].LabelBackColor = System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series["VoladizoInicial0"].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series["VoladizoInicial0"].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series["VoladizoInicial0"].Color = System.Drawing.Color.Blue;
this.chart1.Series["VoladizoInicial0"].Points.AddXY(PuntoCortanteX1V, PuntoCortanteY1V);
PuntoCortanteX2 = Convert.ToDouble(numericUpDownLaVI.Value) + ValorInicialX;
// PuntoCortanteY2 = ((Convert.ToDouble(numericUpDownLaVI.Value) * m) + PuntoCortanteY1V);
PuntoCortanteY2 = PuntoCortanteY1V;
this.chart1.Series["VoladizoInicial0"].Points.AddXY(PuntoCortanteX2, PuntoCortanteY2);
PuntoCortanteX1 = PuntoCortanteX2;
PuntoCortanteY1 = PuntoCortanteY2 - (Convert.ToDouble(numericUpDownFPVI.Value));
this.chart1.Series["VoladizoInicial0"].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
// PuntoCortanteY1 = PuntoCortanteY1 - (Luzlb[1, i] * Wu);
PuntoCortanteX1 = PuntoCortanteX1 + Convert.ToDouble(numericUpDownLbVI.Value);
this.chart1.Series["VoladizoInicial0"].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteY1 = PuntoCortanteY1 + ACA;
puntoy = PuntoCortanteY1;
this.chart1.Series["VoladizoInicial0"].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
ValorInicialX = puntoMomentoX1 = PuntoCortanteX1;
ValorInicialY = puntoMomentoY1 = PuntoCortanteY1;
VarAreaX1 = PuntoCortanteX1;
VarAreaY1 = PuntoCortanteY1;
PuntoCortanteY2 = 0;
PuntoCortanteX2 = 0;
}
}
}
for (int i = 0; i < NCL[1]; i++)
{
switch (Convert.ToInt16(LuzCarga[1, i]))
{
case 2://TriangularIzquierda
chart1.Series[SerieTipo[1]].ChartType
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
chart1.Series[SerieTipo[1]].LabelBackColor = System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series[SerieTipo[1]].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series[SerieTipo[1]].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series[SerieTipo[1]].Color = System.Drawing.Color.Blue;
PuntoCortanteX1 = ValorInicialX;
PuntoCortanteY1 = puntoy;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
// PuntoCortanteX1 = LTLuz[1, 0] - Math.Sqrt(2 * LTLuz[1, 0] * (ACT[1] / Wu)) + ValorInicialX;

```

```

PuntoCortanteX1 = (LTLuz[1, 0] / 2) / 3 + ValorInicialX;
PuntoCortanteY1 = (puntoy - (Wu * LTLuz[1, 0]) / 2) / 2;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteX1 = LTLuz[1, 0] + ValorInicialX;
PuntoCortanteY1 = puntoy - (Wu * LTLuz[1, 0]) / 2;
puntoy = PuntoCortanteY1;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
chart1.Series[SerieTipo2[1]].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series[SerieTipo2[1]].LabelBackColor = System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series[SerieTipo2[1]].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series[SerieTipo2[1]].Color = System.Drawing.Color.Blue;
this.chart1.Series[SerieTipo2[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteX1 = LTLuz[1, 0] + ValorInicialX;
if (Acolum < 9)
{
if (ACT[Acolum] >= 0)
{
PuntoCortanteY1 = puntoy + ACT[Acolum];
puntoy = PuntoCortanteY1;
Acolum++;
}
}
this.chart1.Series[SerieTipo2[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
ValorInicialX = puntoMomentoX1 = PuntoCortanteX1;
ValorInicialY = puntoMomentoY1 = PuntoCortanteY1;
break;
case 3://TriangularDerecha
chart1.Series[SerieTipo[1]].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
chart1.Series[SerieTipo[1]].LabelBackColor = System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series[SerieTipo[1]].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series[SerieTipo[1]].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series[SerieTipo[1]].Color = System.Drawing.Color.Blue;
chart1.ChartAreas[0].AxisX.IsMarginVisible = false;
PuntoCortanteX1 = ValorInicialX;
PuntoCortanteY1 = puntoy;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
// PuntoCortanteX1 = Math.Sqrt(2 * LTLuz[1, 0] * (ACT[1] / Wu)) + ValorInicialX;
PuntoCortanteX1 = (LTLuz[1, 0] / 2) + (LTLuz[1, 0] / 4) + ValorInicialX;
PuntoCortanteY1 = (puntoy + (puntoy - (Wu * LTLuz[1, 0]) / 2)) / 2;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteX1 = LTLuz[1, 0] + ValorInicialX;
PuntoCortanteY1 = puntoy - (Wu * LTLuz[1, 0]) / 2;
puntoy = PuntoCortanteY1;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
chart1.Series[SerieTipo2[1]].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series[SerieTipo2[1]].LabelBackColor = System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series[SerieTipo2[1]].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series[SerieTipo2[1]].Color = System.Drawing.Color.Blue;
this.chart1.Series[SerieTipo2[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteX1 = LTLuz[1, 0] + ValorInicialX;
if (Acolum < 9)
{
if (ACT[Acolum] >= 0)
{
PuntoCortanteY1 = puntoy + ACT[Acolum];
Acolum++;
}
}
puntoy = PuntoCortanteY1;
this.chart1.Series[SerieTipo2[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
ValorInicialX = puntoMomentoX1 = PuntoCortanteX1;
ValorInicialY = puntoMomentoY1 = PuntoCortanteY1;
break;

```

```

case 4://PuntualCentrica
{
chart1.Series[SerieTipo[1]].ChartType
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series[SerieTipo[1]].LabelBackColor
= System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series[SerieTipo[1]].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series[SerieTipo[1]].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series[SerieTipo[1]].Color = System.Drawing.Color.Blue;
chart1.ChartAreas[0].AxisX.IsMarginVisible = false;
PuntoCortanteX1 = ValorInicialX;
PuntoCortanteY1 = puntoy;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteX1 = PuntoCortanteX1 + Luzla[1, i];
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteY1 = PuntoCortanteY1 - LFP[1, i];
puntoy = PuntoCortanteY1;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteX1 = PuntoCortanteX1 + Luzla[1, i];
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
chart1.Series[SerieTipo2[1]].ChartType
= System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series[SerieTipo2[1]].LabelBackColor
= System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series[SerieTipo2[1]].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series[SerieTipo2[1]].Color = System.Drawing.Color.Blue;
this.chart1.Series[SerieTipo2[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
if (Acolum < 9)
{
if (ACT[Acolum] >= 0)
{
PuntoCortanteY1 = puntoy + ACT[Acolum];
Acolum++;
}
puntoy = PuntoCortanteY1;
}
this.chart1.Series[SerieTipo2[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
ValorInicialX = puntoMomentoX1 = PuntoCortanteX1;
ValorInicialY = puntoMomentoY1 = PuntoCortanteY1;
// puntoMomentoY1 - LFP[1, i]
}
break;
case 5://PuntualExcentrica
chart1.Series[SerieTipo[1]].ChartType
= System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series[SerieTipo[1]].LabelBackColor
= System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series[SerieTipo[1]].Label = "X:#VALX{0.##} Y:#VALY{0.##}";
chart1.Series[SerieTipo[1]].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series[SerieTipo[1]].Color = System.Drawing.Color.Blue;
chart1.ChartAreas[0].AxisX.IsMarginVisible = false;
PuntoCortanteX1 = ValorInicialX;
PuntoCortanteY1 = puntoy;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteX1 = PuntoCortanteX1 + Luzla[1, i];
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteY1 = PuntoCortanteY1 - LFP[1, i];
puntoy = PuntoCortanteY1;
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
PuntoCortanteX1 = PuntoCortanteX1 + Luzlb[1, i];
this.chart1.Series[SerieTipo[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
chart1.Series[SerieTipo2[1]].ChartType
= System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
chart1.Series[SerieTipo2[1]].LabelBackColor
= System.Drawing.Color.FromArgb(50,
System.Drawing.Color.LightGreen);
chart1.Series[SerieTipo2[1]].LabelBorderColor = System.Drawing.Color.Black;
chart1.Series[SerieTipo2[1]].Color = System.Drawing.Color.Blue;
this.chart1.Series[SerieTipo2[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
if (Acolum < 9)

```

```

{
if (ACT[Acolum] >= 0)
{
PuntoCortanteY1 = puntoy + ACT[Acolum];
Acolum++;
}
puntoy = PuntoCortanteY1;
}
this.chart1.Series[SerieTipo2[1]].Points.AddXY(PuntoCortanteX1, PuntoCortanteY1);
ValorInicialX = puntoMomentoX1 = PuntoCortanteX1;
ValorInicialY = puntoMomentoY1 = PuntoCortanteY1;
// puntoMomentoY1 - LFP[1, i]
break;
}
}
if (l == 0)
{
double AreaV = 0;
if (labellTDV == "Voladizo Apoyo" || labellTDV == "Voladizo Voladizo" || labellTDV == "Voladizo Empotrado")
{
this.chart2.Series["momento"].Points.AddXY(puntomomentox, 0);
AreaV = (PuntoCortanteY1V * PuntoCortanteX1V) / 2;
puntomomentox++;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, AreaV);
puntomomentox++;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, AreaV + AreaX1);
puntomomentox++;
Rarea1_2 = AreaV + AreaX1 + AreaX2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea1_2);
puntomomentox++;
if (AreaX1 > Rarea1_2)
{
labelML1.Text = "M: " + AreaX1.ToString("0.000");

$$\rho = \frac{((-\phi * (FY * 1000) * BV * d * d) - (\text{Math.Sqrt}(((\phi * (FY * 1000) * BV * d * d) * (-\phi * (FY * 1000) * BV * d * d))) - (4 * (\text{Math.Abs}((\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * AreaX1)))) / (2 * (\text{Math.Abs}((\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))))}{2}$$

refuerzoatracion();
}
}
else
{
labelML1.Text = "M: " + Rarea1_2.ToString("0.000");

$$\rho = \frac{((-\phi * (FY * 1000) * BV * d * d) - (\text{Math.Sqrt}(((\phi * (FY * 1000) * BV * d * d) * (-\phi * (FY * 1000) * BV * d * d))) - (4 * (\text{Math.Abs}((\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea1_2)))) / (2 * (\text{Math.Abs}((\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))))}{2}$$

refuerzoatracion();
}
}
}
else
{
this.chart2.Series["momento"].Points.AddXY(puntomomentox, 0);
puntomomentox++;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, AreaX1);
puntomomentox++;
Rarea1_2 = AreaX1 + AreaX2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea1_2);
puntomomentox++;
if (AreaX1 > Rarea1_2)
{
labelML1.Text = "M: " + AreaX1.ToString("0.000");

$$\rho = \frac{((-\phi * (FY * 1000) * BV * d * d) - (\text{Math.Sqrt}(((\phi * (FY * 1000) * BV * d * d) * (-\phi * (FY * 1000) * BV * d * d))) - (4 * (\text{Math.Abs}((\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * AreaX1)))) / (2 * (\text{Math.Abs}((\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))))}{2}$$

refuerzoatracion();
}
}
else
{
labelML1.Text = "M: " + Rarea1_2.ToString("0.000");

```

```

ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d)))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))))) * Rarea1_2))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
refuerzoatracion();
}
}
labelA1L1.Text = "A1: " + AreaX1.ToString("0.000")+ " " +denominador;
labelA2L1.Text = "A2: " + AreaX2.ToString("0.000")+ " " + denominador;
// labelML1.Text = "M: " + AreaT.ToString("0.000");
labelPL1.Text = "ρ: " + ρ.ToString("0.000000")+ "\n" + "pmin: " + pmin.ToString("0.000000");
labelASL1.Text = "AS: " + AS.ToString("0.0000");
labelVL1.Text = "" + NV + "#" + TV;
labelVL1s2.Text = "" + NV1 + "#" + TV1;
labelVL1s3.Text = "" + NV2 + "#" + TV2;
labelM2L1.Text = "M1: " + M1.ToString("0.0000");
labelM2L2.Text = "M2: " + M2.ToString("0.0000");
labelA1s1.Text = "A's: " + A1S1.ToString("0.0000");
labelA2s1.Text = "As: " + A1S1.ToString("0.0000");
if (label1TDV == "Voladizo Apoyo" || label1TDV == "Voladizo Voladizo" || label1TDV == "Voladizo Empotrado")

{
labelA2VI.Text = "A: " + AreaV.ToString("0.000")+ " " + denominador;
double MVIn = -1 * AreaV;
labelMVI.Text = "M: " + (MVIn.ToString("0.000"));
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d)))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))))) * MVIn))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
labelPVI.Text = "ρ: " + ρ.ToString("0.000000")+ "\n" + "pmin: " + pmin.ToString("0.000000");
refuerzoatracion();
labelASVI.Text = "AS: " + AS.ToString("0.0000");
labelVVI.Text = "" + NV + "#" + TV;
labelVVI2s2.Text = "" + NV1 + "#" + TV1;
labelVVI2s3.Text = "" + NV2 + "#" + TV2;
labelM2VI1.Text = "M1: " + M1.ToString("0.0000");
labelM2VI2.Text = "M2: " + M2.ToString("0.0000");
labelA1sVI1.Text = "A's: " + A1S1.ToString("0.0000");
labelA2sVI1.Text = "As: " + A1S1.ToString("0.0000");
}
if (label1TDV == "Apoyo Voladizo" || label1TDV == "Voladizo Voladizo" || label1TDV == "Empotrado Voladizo")
{
if (ACT[Acolum] >= 0)
{
}
else
{
double AreaF = 0;
PuntoCortanteX1F = Convert.ToDouble(numericUpDownLTVF.Value);
AreaF = ((PuntoCortanteY1 + ACfinal) * PuntoCortanteX1F) / 2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, AreaF + Rarea1_2);
puntomomentox++;
//
labelA2VF.Text = "A: " + AreaF.ToString("0.000")+ " " + denominador;
double MVfn = -1 * (Rarea1_2);
labelMVF.Text = "M: " + (MVfn.ToString("0.000"));
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d)))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))))) * MVfn))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
labelPVF.Text = "ρ: " + ρ.ToString("0.000000")+ "\n" + "pmin: " + pmin.ToString("0.000000");
refuerzoatracion();
labelASVF.Text = "AS: " + AS.ToString("0.0000");
labelVVF.Text = "" + NV + "#" + TV;
labelVVF2s2.Text = "" + NV1 + "#" + TV1;
labelVVF2s3.Text = "" + NV2 + "#" + TV2;
labelM2VF1.Text = "M1: " + M1.ToString("0.0000");
labelM2VF2.Text = "M2: " + M2.ToString("0.0000");
}
}
if (1 == 1)

```

```

{
Rarea2_1 = Rarea1_2 + AreaX1;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea2_1);
puntomomentox++;
Rarea2_2 = Rarea2_1 + AreaX2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea2_2);
puntomomentox++;
if (Rarea2_1 > Rarea1_2)
{
if (Rarea2_1 > Rarea2_2)
{
labelML2.Text = "M: " + Rarea2_1.ToString("0.000");
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((-(φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea2_1)))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
refuerzoatracion();
}
else
{
labelML2.Text = "M: " + Rarea2_2.ToString("0.000");
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((-(φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea2_2)))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
refuerzoatracion();
}
}
else
{
labelML2.Text = "M: " + Rarea1_2.ToString("0.000");
}
labelA1L2.Text = "A1: " + AreaX1.ToString("0.000") + " " + denominador;
labelA2L2.Text = "A2: " + AreaX2.ToString("0.000") + " " + denominador;
// labelML2.Text = "M: " + AreaT.ToString("0.000");
labelPL2.Text = "ρ: " + ρ.ToString("0.000000") + "\n" + "ρmin: " + pmin.ToString("0.000000");
labelASL2.Text = "AS: " + AS.ToString("0.0000");

labelVL2.Text = "" + NV + "#" + TV;
labelVL2s2.Text = "" + NV1 + "#" + TV1;
labelVL2s3.Text = "" + NV2 + "#" + TV2;
labelM21L2.Text = "M1: " + M1.ToString("0.0000");
labelM22L2.Text = "M2: " + M2.ToString("0.0000");
labelA1s2.Text = "A's: " + A1S1.ToString("0.0000");
labelA2s2.Text = "As: " + A1S1.ToString("0.0000");
if (labelITDV == "Apoyo Voladizo" || labelITDV == "Voladizo Voladizo" || labelITDV == "Empotrado Voladizo")
{
if (ACT[Acolum] >= 0)
{
}
else
{
double AreaF = 0;
PuntoCortanteX1F = Convert.ToDouble(numericUpDownLTVF.Value);
AreaF = ((PuntoCortanteY1 + ACfinal) * PuntoCortanteX1F) / 2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, AreaF + Rarea2_2);
puntomomentox++;
//
labelA2VF.Text = "A: " + AreaF.ToString("0.000") + " " + denominador;
double MVfn = -1 * (Rarea1_2);
labelMVf.Text = "M: " + (MVfn.ToString("0.000"));
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((-(φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * MVfn)))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
labelPVf.Text = "ρ: " + ρ.ToString("0.000000") + "\n" + "ρmin: " + pmin.ToString("0.000000");
refuerzoatracion();
labelASVF.Text = "AS: " + AS.ToString("0.0000");
labelVVF.Text = "" + NV + "#" + TV;
labelVVF2s2.Text = "" + NV1 + "#" + TV1;
labelVVF3s3.Text = "" + NV2 + "#" + TV2;
}
}
}

```



```

labelA1sVF.Text = "A's: " + A1S1.ToString("0.0000");
labelA2sVF.Text = "As: " + A1S1.ToString("0.0000");
}
}
if (1 == 3)
{
Rarea4_1 = Rarea3_2 + AreaX1;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea4_1);
puntomomentox++;
Rarea4_2 = Rarea4_1 + AreaX2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea4_2);
puntomomentox++;
if (Rarea4_1 > Rarea4_2)
{
labelML4.Text = "M: " + Rarea4_1.ToString("0.000");

$$\rho = \frac{((-\phi * (FY * 1000) * BV * d * d) - (\text{Math.Sqrt}(((\phi * (FY * 1000) * BV * d * d) * (-\phi * (FY * 1000) * BV * d * d))) - (4 * (\text{Math.Abs}((\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea4_1))))}{2 * (\text{Math.Abs}((\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))));$$

refuerzoatracion();
}
else
{
labelML4.Text = "M: " + Rarea4_2.ToString("0.000");

$$\rho = \frac{((-\phi * (FY * 1000) * BV * d * d) - (\text{Math.Sqrt}(((\phi * (FY * 1000) * BV * d * d) * (-\phi * (FY * 1000) * BV * d * d))) - (4 * (\text{Math.Abs}((\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea4_2))))}{2 * (\text{Math.Abs}((\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))));$$

refuerzoatracion();
}
labelA1L4.Text = "A1: " + AreaX1.ToString("0.000") + " " + denominador;
labelA2L4.Text = "A2: " + AreaX2.ToString("0.000") + " " + denominador;
//labelML4.Text = "M: " + AreaT.ToString("0.000");
labelPL4.Text = "ρ: " + ρ.ToString("0.000000") + "\n" + "pmin: " + pmin.ToString("0.000000");

labelASL4.Text = "AS: " + AS.ToString("0.0000");
labelVL4.Text = "" + NV + "#" + TV;
labelVL4s2.Text = "" + NV1 + "#" + TV1;
labelVL4s3.Text = "" + NV2 + "#" + TV2;
labelM21L4.Text = "M1: " + M1.ToString("0.0000");
labelM22L4.Text = "M2: " + M2.ToString("0.0000");
labelA1s4.Text = "A's: " + A1S1.ToString("0.0000");
labelA2s4.Text = "As: " + A1S1.ToString("0.0000");
if (labelITDV == "Apoyo Voladizo" || labelITDV == "Voladizo Voladizo" || labelITDV == "Empotrado Voladizo")
{
if (ACT[Acolum] >= 0)
{
}
else
{
double AreaF = 0;
PuntoCortanteX1F = Convert.ToDouble(numericUpDownLTVF.Value);
AreaF = ((PuntoCortanteY1 + ACfinal) * PuntoCortanteX1F) / 2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, AreaF + Rarea4_2);
puntomomentox++;
labelA2VF.Text = "A: " + AreaF.ToString("0.000") + " " + denominador;
double MVfn = -1 * (Rarea1_2);
labelMVf.Text = "M: " + (MVfn.ToString("0.000"));

$$\rho = \frac{((-\phi * (FY * 1000) * BV * d * d) - (\text{Math.Sqrt}(((\phi * (FY * 1000) * BV * d * d) * (-\phi * (FY * 1000) * BV * d * d))) - (4 * (\text{Math.Abs}((\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * MVfn))))}{2 * (\text{Math.Abs}((\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))));$$

labelPVf.Text = "ρ: " + ρ.ToString("0.000000") + "\n" + "pmin: " + pmin.ToString("0.000000");
refuerzoatracion();
labelASVF.Text = "AS: " + AS.ToString("0.0000");
labelVVF.Text = "" + NV + "#" + TV;
labelVVF2.Text = "" + NV1 + "#" + TV1;
labelVVF3.Text = "" + NV2 + "#" + TV2;
labelM21VF.Text = "M1: " + M1.ToString("0.0000");
labelM22VF.Text = "M2: " + M2.ToString("0.0000");
labelA1sVF.Text = "A's: " + A1S1.ToString("0.0000");

```

```

labelA2sVF.Text = "As: " + A1S1.ToString("0.0000");
}
}
if (1 == 4)
{
Rarea5_1 = Rarea4_2 + AreaX1;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea5_1);
puntomomentox++;
Rarea5_2 = Rarea5_1 + AreaX2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea5_2);
puntomomentox++;
if (Rarea5_1 > Rarea5_2)
{
labelML5.Text = "M: " + Rarea5_1.ToString("0.000");
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((-(φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea5_1)))))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
refuerzoatracion();
}
else
{
labelML5.Text = "M: " + Rarea5_2.ToString("0.000");
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((-(φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea5_2)))))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
refuerzoatracion();
}
labelA1L5.Text = "A1: " + AreaX1.ToString("0.000") + " " + denominador;
labelA2L5.Text = "A2: " + AreaX2.ToString("0.000") + " " + denominador;
//labelML5.Text = "M: " + AreaT.ToString("0.000");
labelPL5.Text = "ρ: " + ρ.ToString("0.000000") + "\n" + "ρmin: " + pmin.ToString("0.000000");
labelASL5.Text = "AS: " + AS.ToString("0.0000");
labelVL5.Text = "" + NV + "#" + TV;
labelVL5s2.Text = "" + NV1 + "#" + TV1;
labelVL5s3.Text = "" + NV2 + "#" + TV2;
labelM21L5.Text = "M1: " + M1.ToString("0.0000");
labelM22L5.Text = "M2: " + M2.ToString("0.0000");
labelA1s5.Text = "A's: " + A1S1.ToString("0.0000");
labelA2s5.Text = "As: " + A1S1.ToString("0.0000");
if (label1TDV == "Apoyo Voladizo" || label1TDV == "Voladizo Voladizo" || label1TDV == "Empotrado Voladizo")
{
if (ACT[Acolum] >= 0)
{
}
else
{
double AreaF = 0;
PuntoCortanteX1F = Convert.ToDouble(numericUpDownLTVF.Value);
AreaF = ((PuntoCortanteY1 + ACfinal) * PuntoCortanteX1F) / 2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, AreaF + Rarea5_2);
puntomomentox++;
labelA2VF.Text = "A: " + AreaF.ToString("0.000") + " " + denominador;
double MVfn = -1 * (Rarea1_2);
labelMVf.Text = "M: " + (MVfn.ToString("0.000"));
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((-(φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * MVfn)))))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
labelPVF.Text = "ρ: " + ρ.ToString("0.000000") + "\n" + "ρmin: " + pmin.ToString("0.000000");
refuerzoatracion();
labelASVF.Text = "AS: " + AS.ToString("0.0000");
labelVVF.Text = "" + NV + "#" + TV;
labelVVF2s2.Text = "" + NV1 + "#" + TV1;
labelVVF2s3.Text = "" + NV2 + "#" + TV2;
labelM21VF.Text = "M1: " + M1.ToString("0.0000");
labelM22VF.Text = "M2: " + M2.ToString("0.0000");
labelA1sVF.Text = "A's: " + A1S1.ToString("0.0000");
labelA2sVF.Text = "As: " + A1S1.ToString("0.0000");
}
}
}

```

```

}
if (1 == 5)
{
Rarea6_1 = Rarea5_2 + AreaX1;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea6_1);
puntomomentox++;
Rarea6_2 = Rarea6_1 + AreaX2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea6_2);
puntomomentox++;
if (Rarea6_1 > Rarea6_2)
{
labelML6.Text = "M: " + Rarea6_1.ToString("0.000");
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((-(φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea6_1)))))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
refuerzoatracion();
}
else
{
labelML6.Text = "M: " + Rarea6_2.ToString("0.000");
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((-(φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea6_2)))))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
refuerzoatracion();
}
labelA1L6.Text = "A1: " + AreaX1.ToString("0.000") + " " + denominador;
labelA2L6.Text = "A2: " + AreaX2.ToString("0.000") + " " + denominador;
//labelML6.Text = "M: " + AreaT.ToString("0.000");
labelPL6.Text = "ρ: " + ρ.ToString("0.000000") + "\n" + "ρmin: " + pmin.ToString("0.000000");
labelASL6.Text = "AS: " + AS.ToString("0.0000");
labelVL6.Text = "" + NV + "#" + TV;
labelVL6s2.Text = "" + NV1 + "#" + TV1;
labelVL6s3.Text = "" + NV2 + "#" + TV2;
labelM21L6.Text = "M1: " + M1.ToString("0.0000");
labelM22L6.Text = "M2: " + M2.ToString("0.0000");
labelA1s6.Text = "A's: " + A1S1.ToString("0.0000");
labelA2s6.Text = "As: " + A1S1.ToString("0.0000");
if (label1TDV == "Apoyo Voladizo" || label1TDV == "Voladizo Voladizo" || label1TDV == "Empotrado Voladizo")
{
if (ACT[Acolum] >= 0)
{
}
else
{
double AreaF = 0;
PuntoCortanteX1F = Convert.ToDouble(numericUpDownLTVF.Value);
AreaF = ((PuntoCortanteY1 + ACfinal) * PuntoCortanteX1F) / 2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, AreaF + Rarea6_2);
puntomomentox++;
labelA2VF.Text = "A: " + AreaF.ToString("0.000") + " " + denominador;
double MVfn = -1 * (Rarea1_2);
labelMVf.Text = "M: " + (MVfn.ToString("0.000"));
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((-(φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * MVfn)))))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
labelPVf.Text = "ρ: " + ρ.ToString("0.000000") + "\n" + "ρmin: " + pmin.ToString("0.000000");
refuerzoatracion();
labelASVF.Text = "AS: " + AS.ToString("0.0000");
labelVVF.Text = "" + NV + "#" + TV;
labelVVF2s2.Text = "" + NV1 + "#" + TV1;
labelVVF2s3.Text = "" + NV2 + "#" + TV2;
labelM21VF.Text = "M1: " + M1.ToString("0.0000");
labelM22VF.Text = "M2: " + M2.ToString("0.0000");
labelA1sVF.Text = "A's: " + A1S1.ToString("0.0000");
labelA2sVF.Text = "As: " + A1S1.ToString("0.0000");
}
}
if (1 == 6)

```

```

{
Rarea7_1 = Rarea6_2 + AreaX1;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea7_1);
puntomomentox++;
Rarea7_2 = Rarea7_1 + AreaX2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea7_2);
puntomomentox++;
if (Rarea7_1 > Rarea7_2)
{
labelML7.Text = "M: " + Rarea7_1.ToString("0.000");

$$\rho = \frac{((-\phi * (FY * 1000) * BV * d * d) - (\text{Math.Sqrt}(((\phi * (FY * 1000) * BV * d * d) * (-\phi * (FY * 1000) * BV * d * d))) - (4 * (\text{Math.Abs}(\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea7_1)))}{2 * (\text{Math.Abs}(\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))}$$
;
refuerzoatracion();
}
else
{
labelML7.Text = "M: " + Rarea7_2.ToString("0.000");

$$\rho = \frac{((-\phi * (FY * 1000) * BV * d * d) - (\text{Math.Sqrt}(((\phi * (FY * 1000) * BV * d * d) * (-\phi * (FY * 1000) * BV * d * d))) - (4 * (\text{Math.Abs}(\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea7_2)))}{2 * (\text{Math.Abs}(\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))}$$
;
refuerzoatracion();
}
labelA1L7.Text = "A1: " + AreaX1.ToString("0.000") + " " + denominador;
labelA2L7.Text = "A2: " + AreaX2.ToString("0.000") + " " + denominador;
//labelML7.Text = "M: " + AreaT.ToString("0.000");
labelPL7.Text = "ρ: " + ρ.ToString("0.000000") + "\n" + "ρmin: " + pmin.ToString("0.000000");
labelASL7.Text = "AS: " + AS.ToString("0.0000");
labelVL7.Text = "" + NV + "#" + TV;
labelVL7s2.Text = "" + NV1 + "#" + TV1;
labelVL7s3.Text = "" + NV2 + "#" + TV2;
labelM21L7.Text = "M1: " + M1.ToString("0.0000");
labelM22L7.Text = "M2: " + M2.ToString("0.0000");
labelA1s7.Text = "A's: " + A1S1.ToString("0.0000");
labelA2s7.Text = "As: " + A1S1.ToString("0.0000");
if (label1TDV == "Apoyo Voladizo" || label1TDV == "Voladizo Voladizo" || label1TDV == "Empotrado Voladizo")
{
if (ACT[Acolum] >= 0)
{
}
else
{
double AreaF = 0;
PuntoCortanteX1F = Convert.ToDouble(numericUpDownLTVF.Value);
AreaF = ((PuntoCortanteY1 + ACfinal) * PuntoCortanteX1F) / 2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, AreaF + Rarea7_2);
puntomomentox++;
labelA2VF.Text = "A: " + AreaF.ToString("0.000") + " " + denominador;
double MVfn = -1 * (Rarea1_2);
labelMVf.Text = "M: " + (MVfn.ToString("0.000"));

$$\rho = \frac{((-\phi * (FY * 1000) * BV * d * d) - (\text{Math.Sqrt}(((\phi * (FY * 1000) * BV * d * d) * (-\phi * (FY * 1000) * BV * d * d))) - (4 * (\text{Math.Abs}(\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * MVfn)))}{2 * (\text{Math.Abs}(\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))}$$
;
labelPVF.Text = "ρ: " + ρ.ToString("0.000000") + "\n" + "ρmin: " + pmin.ToString("0.000000");
refuerzoatracion();
labelASVF.Text = "AS: " + AS.ToString("0.0000");
labelVVF.Text = "" + NV + "#" + TV;
labelVVF2s.Text = "" + NV1 + "#" + TV1;
labelVVF3s.Text = "" + NV2 + "#" + TV2;
labelM21VF.Text = "M1: " + M1.ToString("0.0000");
labelM22VF.Text = "M2: " + M2.ToString("0.0000");
labelA1sVF.Text = "A's: " + A1S1.ToString("0.0000");
labelA2sVF.Text = "As: " + A1S1.ToString("0.0000");
}
}
if (1 == 7)
{
Rarea8_1 = Rarea7_2 + AreaX1;

```

```

this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea8_1);
puntomomentox++;
Rarea8_2 = Rarea8_1 + AreaX2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea8_2);
puntomomentox++;
if (Rarea8_1 > Rarea8_2)
{
labelML8.Text = "M: " + Rarea8_1.ToString("0.000");

$$\rho = \frac{((-\phi * (FY * 1000) * BV * d * d) - (\text{Math.Sqrt}(((\phi * (FY * 1000) * BV * d * d) * (-\phi * (FY * 1000) * BV * d * d))) - (4 * (\text{Math.Abs}(\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea8_1)))}{2 * (\text{Math.Abs}(\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))}$$
;
refuerzoatracion();
}
else
{
labelML8.Text = "M: " + Rarea8_2.ToString("0.000");

$$\rho = \frac{((-\phi * (FY * 1000) * BV * d * d) - (\text{Math.Sqrt}(((\phi * (FY * 1000) * BV * d * d) * (-\phi * (FY * 1000) * BV * d * d))) - (4 * (\text{Math.Abs}(\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea8_2)))}{2 * (\text{Math.Abs}(\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))}$$
;
refuerzoatracion();
}

labelA1L8.Text = "A1: " + AreaX1.ToString("0.000") + " " + denominador;
labelA2L8.Text = "A2: " + AreaX2.ToString("0.000") + " " + denominador;
//labelML8.Text = "M: " + AreaT.ToString("0.000");
labelPL8.Text = "p: " + p.ToString("0.000000") + "\n" + "pmin: " + pmin.ToString("0.000000");
labelASL8.Text = "AS: " + AS.ToString("0.0000");
labelVL8.Text = "" + NV + "#" + TV;
labelVL8s2.Text = "" + NV1 + "#" + TV1;
labelVL8s3.Text = "" + NV2 + "#" + TV2;
labelM21L8.Text = "M1: " + M1.ToString("0.0000");
labelM22L8.Text = "M2: " + M2.ToString("0.0000");
labelA1s8.Text = "A's: " + A1S1.ToString("0.0000");
labelA2s8.Text = "As: " + A1S1.ToString("0.0000");
if (labelITDV == "Apoyo Voladizo" || labelITDV == "Voladizo Voladizo" || labelITDV == "Empotrado Voladizo")
{
if (ACT[Acolum] >= 0)
{
}
else
{
double AreaF = 0;
PuntoCortanteX1F = Convert.ToDouble(numericUpDownLTVF.Value);
AreaF = ((PuntoCortanteY1 + ACfinal) * PuntoCortanteX1F) / 2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, AreaF + Rarea8_2);
puntomomentox++;
labelA2VF.Text = "A: " + AreaF.ToString("0.000") + " " + denominador;
double MVfn = -1 * (Rarea1_2);
labelMVf.Text = "M: " + (MVfn.ToString("0.000"));

$$\rho = \frac{((-\phi * (FY * 1000) * BV * d * d) - (\text{Math.Sqrt}(((\phi * (FY * 1000) * BV * d * d) * (-\phi * (FY * 1000) * BV * d * d))) - (4 * (\text{Math.Abs}(\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * MVfn)))}{2 * (\text{Math.Abs}(\phi * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))}$$
;
labelPVF.Text = "p: " + p.ToString("0.000000") + "\n" + "pmin: " + pmin.ToString("0.000000");
refuerzoatracion();
labelASVF.Text = "AS: " + AS.ToString("0.0000");
labelVVF.Text = "" + NV + "#" + TV;
labelVVF2s2.Text = "" + NV1 + "#" + TV1;
labelVVF2s3.Text = "" + NV2 + "#" + TV2;
labelM21VF.Text = "M1: " + M1.ToString("0.0000");
labelM22VF.Text = "M2: " + M2.ToString("0.0000");
labelA1sVF.Text = "A's: " + A1S1.ToString("0.0000");
labelA2sVF.Text = "As: " + A1S1.ToString("0.0000");
}
}
if (1 == 8)
{
Rarea9_1 = Rarea8_2 + AreaX1;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea9_1);
}

```

```

puntomomentox++;
Rarea9_2 = Rarea9_1 + AreaX2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, Rarea9_2);
puntomomentox++;
if (Rarea9_1 > Rarea9_2)
{
labelML9.Text = "M: " + Rarea9_1.ToString("0.000");
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((-(φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea9_1)))))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
refuerzoatracion();
}
else
{
labelML9.Text = "M: " + Rarea9_2.ToString("0.000");
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((-(φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * Rarea9_2)))))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
refuerzoatracion();
}
labelA1L9.Text = "A1: " + AreaX1.ToString("0.000") + " " + denominador;
labelA2L9.Text = "A2: " + AreaX2.ToString("0.000") + " " + denominador;
//labelML9.Text = "M: " + AreaT.ToString("0.000");
labelPL9.Text = "ρ: " + ρ.ToString("0.000000") + "\n" + "ρmin: " + pmin.ToString("0.000000");
labelASL9.Text = "AS: " + AS.ToString("0.0000");
labelVL9.Text = "" + NV + "#" + TV;
labelVL9s2.Text = "" + NV1 + "#" + TV1;
labelVL9s3.Text = "" + NV2 + "#" + TV2;
labelM21L9.Text = "M1: " + M1.ToString("0.0000");
labelM22L9.Text = "M2: " + M2.ToString("0.0000");
labelA1s9.Text = "A's: " + A1S1.ToString("0.0000");
labelA2s9.Text = "As: " + A1S1.ToString("0.0000");
if (label1TDV == "Apoyo Voladizo" || label1TDV == "Voladizo Voladizo" || label1TDV == "Empotrado Voladizo")
{
double AreaF = 0;
PuntoCortanteX1F = Convert.ToDouble(numericUpDownLTVF.Value);
AreaF = ((PuntoCortanteY1 + ACfinal) * PuntoCortanteX1F) / 2;
this.chart2.Series["momento"].Points.AddXY(puntomomentox, AreaF + Rarea9_2);
puntomomentox++;
labelA2VF.Text = "A: " + AreaF.ToString("0.000") + " " + denominador;
double MVfn = -1 * (Rarea1_2);
labelMVf.Text = "M: " + (MVfn.ToString("0.000"));
ρ = (((-(-(φ * (FY * 1000) * BV * d * d)) - (Math.Sqrt(((-(φ * (FY * 1000) * BV * d * d)) * (-(φ * (FY * 1000) * BV * d * d))) - (4 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC)))) * MVfn)))))) / (2 * (Math.Abs((φ * (FY * 1000) * BV * d * d) * (-0.59 * (FY / FC))))));
labelPVf.Text = "ρ: " + ρ.ToString("0.000000") + "\n" + "ρmin: " + pmin.ToString("0.000000");
refuerzoatracion();
labelASVF.Text = "AS: " + AS.ToString("0.0000");
labelVVF.Text = "" + NV + "#" + TV;
labelVVFf2.Text = "" + NV1 + "#" + TV1;
labelVVFf3.Text = "" + NV2 + "#" + TV2;
labelM21VF.Text = "M1: " + M1.ToString("0.0000");
labelM22VF.Text = "M2: " + M2.ToString("0.0000");
labelA1sVF.Text = "A's: " + A1S1.ToString("0.0000");
labelA2sVF.Text = "As: " + A1S1.ToString("0.0000");
}
//
void refuerzoatracion()
{
AS = ρ * ((BV * 100) * (d * 100));
DiferenciaOld = int.MaxValue;
var1 = var2 = 0;
for (int z = 0; z < VarSA.Length; z++)
{
if (VarSA[z] == AS)
{
TV = TipoVarilla[z];
NV = NumeroVarillas[z];
}
}
}

```

```

return;
}
else
{
Diferencia = VarSA[z] - AS;
Diferencia = Math.Abs(Diferencia);
if (Diferencia < DiferenciaOld)
{
if (VarSA[z] >= AS)
{
DiferenciaOld = Diferencia;
TV = TipoVarilla[z];
NV = NumeroVarillas[z];
var1 = z;
}
}
}
DiferenciaOld = int.MaxValue;
for (int z = 0; z < VarSA.Length; z++)
{
if (VarSA[z] == AS)
{
TV1 = TipoVarilla[z];
NV1 = NumeroVarillas[z];
return;
}
else
{
Diferencia = VarSA[z] - AS;
Diferencia = Math.Abs(Diferencia);
if (Diferencia < DiferenciaOld)
{
if (z != var1)
{
if (VarSA[z] >= AS)
{
DiferenciaOld = Diferencia;
TV1 = TipoVarilla[z];
NV1 = NumeroVarillas[z];
var2 = z;
}
}
}
}
DiferenciaOld = int.MaxValue;
for (int z = 0; z < VarSA.Length; z++)
{
if (VarSA[z] == AS)
{
TV2 = TipoVarilla[z];
NV2 = NumeroVarillas[z];
return;
}
else
{
Diferencia = VarSA[z] - AS;
Diferencia = Math.Abs(Diferencia);
if (Diferencia < DiferenciaOld)
{
if (z != var2 && z != var1)
{
if (VarSA[z] >= AS)
{
DiferenciaOld = Diferencia;
TV2 = TipoVarilla[z];
NV2 = NumeroVarillas[z];
}
}
}
}
}
}
}

```



```

diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz1Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
// diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz1Dvc.Text = Convert.ToString("0.0");
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz1Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - (0.20 - diseñoXvc2);
luz1Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
//maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
//maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
// unmax = (diseñoDvu - 0.2) / smmbm;
// unmax = unmax++;
luz1UNmax.Text = Convert.ToString("0.0");
//unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz1UNmed.Text = Convert.ToString("0.0");
unbaj = ((diseñoDvc2) / smmbm)+1;
luz1UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
diseñoVc2 = diseñoVc / 2;
luz1vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz1xvu.Text = Convert.ToString(diseñoXvu.ToString("0.000"));
diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz1dvu.Text = Convert.ToString(diseñoDvu.ToString("0.000"));
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz1Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz1Dvc.Text = Convert.ToString(diseñoDvc.ToString("0.000"));
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz1Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - diseñoXvc2;
luz1Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
if (maxs1 < maxs2 && maxs1 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs1;
unmax = unmax + 1;
luz1UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs1;
luz1UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs1;
luz1UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else if (maxs2 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs2;
unmax = unmax++;
luz1UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs2;
luz1UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs2;
luz1UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
unmax = (diseñoDvu - 0.2) / smmbm;
unmax = unmax++;
luz1UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz1UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / smmbm;
luz1UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
}

```

```

}
}
}
diseñoVu = vmaxi - (Wu * (0.2 + d));
luz2dvu.Text = Convert.ToString(diseñoVu.ToString("0.000"));
diseñoVc = (0.17 * 0.75 * Math.Sqrt(FC) * BV * d) * 1000;
luz2vc.Text = Convert.ToString(diseñoVc.ToString("0.000"));
if (diseñoVc > diseñoVu)
{
luz1vu.Text = Convert.ToString("0.0");
diseñoVc2 = diseñoVc / 2;
luz2vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
//diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz2xvu.Text = Convert.ToString("0.0");
//diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz2dvu.Text = Convert.ToString("0.0");
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz2Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
// diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz2Dvc.Text = Convert.ToString("0.0");
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz2Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - (0.20 - diseñoXvc2);
luz2Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
//maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
//maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
// unmax = (diseñoDvu - 0.2) / smmbm;
// unmax = unmax++;
luz2UNmax.Text = Convert.ToString("0.0");
//unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz2UNmed.Text = Convert.ToString("0.0");
unbaj = ((diseñoDvc2) / smmbm) + 1;
luz2UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
diseñoVc2 = diseñoVc / 2;
luz2vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz2xvu.Text = Convert.ToString(diseñoXvu.ToString("0.000"));
diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz2dvu.Text = Convert.ToString(diseñoDvu.ToString("0.000"));
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz2Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz2Dvc.Text = Convert.ToString(diseñoDvc.ToString("0.000"));
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz2Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - diseñoXvc2;
luz2Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
if (maxs1 < maxs2 && maxs1 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs1;
unmax = unmax + 1;
luz2UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs1;
luz2UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs1;
luz2UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else if (maxs2 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs2;

```

```

unmax = unmax++;
luz2UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs2;
luz2UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs2;
luz2UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
unmax = (diseñoDvu - 0.2) / smmbm;
unmax = unmax++;
luz2UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz2UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / smmbm;
luz1UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
}
}
}
diseñoVu = vmaxi - (Wu * (0.2 + d));
luz3vu.Text = Convert.ToString(diseñoVu.ToString("0.000"));
diseñoVc = (0.17 * 0.75 * Math.Sqrt(FC) * BV * d) * 1000;
luz3vc.Text = Convert.ToString(diseñoVc.ToString("0.000"));
if (diseñoVc > diseñoVu)
{
luz3dvu.Text = Convert.ToString("0.0");
diseñoVc2 = diseñoVc / 2;
luz3vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
//diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz3xvu.Text = Convert.ToString("0.0");
//diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz3dvu.Text = Convert.ToString("0.0");
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz3Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
// diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz3Dvc.Text = Convert.ToString("0.0");
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz3Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - (0.20 - diseñoXvc2);
luz3Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
//maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
//maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
// unmax = (diseñoDvu - 0.2) / smmbm;
// unmax = unmax++;
luz3UNmax.Text = Convert.ToString("0.0");
//unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz3UNmed.Text = Convert.ToString("0.0");
unbaj = ((diseñoDvc2) / smmbm) + 1;
luz3UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
diseñoVc2 = diseñoVc / 2;
luz3vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz3xvu.Text = Convert.ToString(diseñoXvu.ToString("0.000"));
diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz3dvu.Text = Convert.ToString(diseñoDvu.ToString("0.000"));
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz3Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz3Dvc.Text = Convert.ToString(diseñoDvc.ToString("0.000"));
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz3Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - diseñoXvc2;
luz3Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
}
}
}
}

```

```

/////separadores/////
maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
if (maxs1 < maxs2 && maxs1 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs1;
unmax = unmax + 1;
luz3UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs1;
luz3UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs1;
luz3UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else if (maxs2 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs2;
unmax = unmax++;
luz3UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs2;
luz3UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs2;
luz3UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
unmax = (diseñoDvu - 0.2) / smmbm;
unmax = unmax++;
luz3UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz3UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / smmbm;
luz3UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
}
}
diseñoVu = vmaxi - (Wu * (0.2 + d));
luz4vu.Text = Convert.ToString(diseñoVu.ToString("0.000"));
diseñoVc = (0.17 * 0.75 * Math.Sqrt(FC) * BV * d) * 1000;
luz4vc.Text = Convert.ToString(diseñoVc.ToString("0.000"));
if (diseñoVc > diseñoVu)
{
luz4dvu.Text = Convert.ToString("0.0");
diseñoVc2 = diseñoVc / 2;
luz4vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
//diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz4xvu.Text = Convert.ToString("0.0");
//diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz4dvu.Text = Convert.ToString("0.0");
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz4Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
// diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz4Dvc.Text = Convert.ToString("0.0");
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz4Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - (0.20 - diseñoXvc2);
luz4Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
//maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
//maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
// unmax = (diseñoDvu - 0.2) / smmbm;
// unmax = unmax++;
luz4UNmax.Text = Convert.ToString("0.0");
//unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz4UNmed.Text = Convert.ToString("0.0");
unbaj = ((diseñoDvc2) / smmbm) + 1;
luz4UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
}

```

```

else
{
diseñoVc2 = diseñoVc / 2;
luz4vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz4xvu.Text = Convert.ToString(diseñoXvu.ToString("0.000"));
diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz4dvu.Text = Convert.ToString(diseñoDvu.ToString("0.000"));
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz4Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz4Dvc.Text = Convert.ToString(diseñoDvc.ToString("0.000"));
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz4Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - diseñoXvc2;
luz4Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
if (maxs1 < maxs2 && maxs1 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs1;
unmax = unmax + 1;
luz4UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs1;
luz4UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs1;
luz4UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else if (maxs2 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs2;
unmax = unmax++;
luz4UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs2;
luz4UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs2;
luz4UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
unmax = (diseñoDvu - 0.2) / smmbm;
unmax = unmax++;
luz4UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz4UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / smmbm;
luz4UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
}
}
diseñoVu = vmaxi - (Wu * (0.2 + d));
luz5vu.Text = Convert.ToString(diseñoVu.ToString("0.000"));
diseñoVc = (0.17 * 0.75 * Math.Sqrt(FC) * BV * d) * 1000;
luz5vc.Text = Convert.ToString(diseñoVc.ToString("0.000"));
if (diseñoVc > diseñoVu)
{
luz5dvu.Text = Convert.ToString("0.0");
diseñoVc2 = diseñoVc / 2;
luz5vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
//diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz5xvu.Text = Convert.ToString("0.0");
//diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz5dvu.Text = Convert.ToString("0.0");
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz5Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
// diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz5Dvc.Text = Convert.ToString("0.0");
}
}
}

```

```

diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz5Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - (0.20 - diseñoXvc2);
luz5Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
//maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
//maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
// unmax = (diseñoDvu - 0.2) / smmbm;
// unmax = unmax++;
luz5UNmax.Text = Convert.ToString("0.0");
//unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz5UNmed.Text = Convert.ToString("0.0");
unbaj = ((diseñoDvc2) / smmbm) + 1;
luz5UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
diseñoVc2 = diseñoVc / 2;
luz5vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz5xvu.Text = Convert.ToString(diseñoXvu.ToString("0.000"));
diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz5dvu.Text = Convert.ToString(diseñoDvu.ToString("0.000"));
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz5Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz5Dvc.Text = Convert.ToString(diseñoDvc.ToString("0.000"));
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz5Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - diseñoXvc2;
luz5Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
if (maxs1 < maxs2 && maxs1 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs1;
unmax = unmax + 1;
luz5UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs1;
luz5UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs1;
luz5UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else if (maxs2 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs2;
unmax = unmax++;
luz5UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs2;
luz5UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs2;
luz5UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
}
else
{
unmax = (diseñoDvu - 0.2) / smmbm;
unmax = unmax++;
luz5UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz5UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / smmbm;
luz5UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
}
}
diseñoVu = vmaxi - (Wu * (0.2 + d));

```

```

luz6vu.Text = Convert.ToString(diseñoVu.ToString("0.000"));
diseñoVc = (0.17 * 0.75 * Math.Sqrt(FC) * BV * d) * 1000;
luz6vc.Text = Convert.ToString(diseñoVc.ToString("0.000"));
if (diseñoVc > diseñoVu)
{
luz6vu.Text = Convert.ToString("0.0");
diseñoVc2 = diseñoVc / 2;
luz6vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
//diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz6xvu.Text = Convert.ToString("0.0");
//diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz6dvu.Text = Convert.ToString("0.0");
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz6Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
// diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz6Dvc.Text = Convert.ToString("0.0");
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz6Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - (0.20 - diseñoXvc2);
luz6Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
//maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
//maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
// unmax = (diseñoDvu - 0.2) / smmbm;
// unmax = unmax++;
luz6UNmax.Text = Convert.ToString("0.0");
//unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz6UNmed.Text = Convert.ToString("0.0");
unbaj = ((diseñoDvc2) / smmbm) + 1;
luz6UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
diseñoVc2 = diseñoVc / 2;
luz6vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz6xvu.Text = Convert.ToString(diseñoXvu.ToString("0.000"));
diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz6dvu.Text = Convert.ToString(diseñoDvu.ToString("0.000"));
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz6Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz6Dvc.Text = Convert.ToString(diseñoDvc.ToString("0.000"));
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz6Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - diseñoXvc2;
luz6Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
if (maxs1 < maxs2 && maxs1 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs1;
unmax = unmax + 1;
luz6UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs1;
luz6UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs1;
luz6UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else if (maxs2 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs2;
unmax = unmax++;
luz6UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs2;
luz6UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
}
}

```

```

unbaj = (diseñoDvc - diseñoDvc2) / maxs2;
luz6UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
unmax = (diseñoDvu - 0.2) / smmbm;
unmax = unmax++;
luz6UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz6UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / smmbm;
luz6UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
}
}
diseñoVu = vmaxi - (Wu * (0.2 + d));
luz7vu.Text = Convert.ToString(diseñoVu.ToString("0.000"));
diseñoVc = (0.17 * 0.75 * Math.Sqrt(FC) * BV * d) * 1000;
luz7Dvc.Text = Convert.ToString(diseñoVc.ToString("0.000"));
if (diseñoVc > diseñoVu)
{
luz7vu.Text = Convert.ToString("0.0");
diseñoVc2 = diseñoVc / 2;
luz7vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
//diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz7xvu.Text = Convert.ToString("0.0");
//diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz7dvu.Text = Convert.ToString("0.0");
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz7Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
// diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz7Dvc.Text = Convert.ToString("0.0");
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz7Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - (0.20 - diseñoDvc2);
luz7Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
//maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
//maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
// unmax = (diseñoDvu - 0.2) / smmbm;
// unmax = unmax++;
luz7UNmax.Text = Convert.ToString("0.0");
//unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz7UNmed.Text = Convert.ToString("0.0");
unbaj = ((diseñoDvc2) / smmbm) + 1;
luz7UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
diseñoVc2 = diseñoVc / 2;
luz7vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz7xvu.Text = Convert.ToString(diseñoXvu.ToString("0.000"));
diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz7dvu.Text = Convert.ToString(diseñoDvu.ToString("0.000"));
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz7Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz7Dvc.Text = Convert.ToString(diseñoDvc.ToString("0.000"));
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz7Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - diseñoXvc2;
luz7Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
if (maxs1 < maxs2 && maxs1 < smmbm)

```

```

{
unmax = (diseñoDvu - 0.2) / maxs1;
unmax = unmax + 1;
luz7UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs1;
luz7UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs1;
luz7UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else if (maxs2 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs2;
unmax = unmax++;
luz7UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs2;
luz7UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs2;
luz7UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
unmax = (diseñoDvu - 0.2) / smmbm;
unmax = unmax++;
luz7UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz7UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / smmbm;
luz7UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
}
diseñoVu = vmaxi - (Wu * (0.2 + d));
luz8vu.Text = Convert.ToString(diseñoVu.ToString("0.000"));
diseñoVc = (0.17 * 0.75 * Math.Sqrt(FC) * BV * d) * 1000;
luz8vc.Text = Convert.ToString(diseñoVc.ToString("0.000"));
if (diseñoVc > diseñoVu)
{
luz8vu.Text = Convert.ToString("0.0");
diseñoVc2 = diseñoVc / 2;
luz8vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
//diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz8xvu.Text = Convert.ToString("0.0");
//diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz8dvu.Text = Convert.ToString("0.0");
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz8xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
// diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz8dvc.Text = Convert.ToString("0.0");
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz8xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - (0.20 - diseñoXvc2);
luz8dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
//maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
//maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
// unmax = (diseñoDvu - 0.2) / smmbm;
// unmax = unmax++;
luz8UNmax.Text = Convert.ToString("0.0");
//unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz8UNmed.Text = Convert.ToString("0.0");
unbaj = ((diseñoDvc2) / smmbm) + 1;
luz8UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
diseñoVc2 = diseñoVc / 2;
luz8vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;

```

```

luz8xvu.Text = Convert.ToString(diseñoXvu.ToString("0.000"));
diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz8dvu.Text = Convert.ToString(diseñoDvu.ToString("0.000"));
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz8Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz8Dvc.Text = Convert.ToString(diseñoDvc.ToString("0.000"));
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz8Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - diseñoXvc2;
luz8Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
if (maxs1 < maxs2 && maxs1 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs1;
unmax = unmax + 1;
luz8UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs1;
luz8UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs1;
luz8UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else if (maxs2 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs2;
unmax = unmax++;
luz8UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs2;
luz8UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs2;
luz8UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
unmax = (diseñoDvu - 0.2) / smmbm;
unmax = unmax++;
luz8UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz8UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / smmbm;
luz8UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
}
}
diseñoVu = vmaxi - (Wu * (0.2 + d));
luz9vu.Text = Convert.ToString(diseñoVu.ToString("0.000"));
diseñoVc = (0.17 * 0.75 * Math.Sqrt(FC) * BV * d) * 1000;
luz9vc.Text = Convert.ToString(diseñoVc.ToString("0.000"));
if (diseñoVc > diseñoVu)
{
luz9vu.Text = Convert.ToString("0.0");
diseñoVc2 = diseñoVc / 2;
luz9vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
//diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz9xvu.Text = Convert.ToString("0.0");
//diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz9dvu.Text = Convert.ToString("0.0");
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz9Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
// diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz9Dvc.Text = Convert.ToString("0.0");
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz9Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - (0.20 - diseñoXvc2);
luz9Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////

```

```

//maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
//maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
// unmax = (diseñoDvu - 0.2) / smmbm;
// unmax = unmax++;
luz9UNmax.Text = Convert.ToString("0.0");
//unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz9UNmed.Text = Convert.ToString("0.0");
unbaj = ((diseñoDvc2) / smmbm) + 1;
luz9UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
diseñoVc2 = diseñoVc / 2;
luz9vc2.Text = Convert.ToString(diseñoVc2.ToString("0.000"));
diseñoXvu = ((LTLuz[1, 0] / 2) * diseñoVu) / vmaxi;
luz9xvu.Text = Convert.ToString(diseñoXvu.ToString("0.000"));
diseñoDvu = (LTLuz[1, 0] / 2) - diseñoXvu;
luz9dvu.Text = Convert.ToString(diseñoDvu.ToString("0.000"));
diseñoXvc = ((LTLuz[1, 0] / 2) * diseñoVc) / vmaxi;
luz9Xvc.Text = Convert.ToString(diseñoXvc.ToString("0.000"));
diseñoDvc = (LTLuz[1, 0] / 2) - diseñoXvc;
luz9Dvc.Text = Convert.ToString(diseñoDvc.ToString("0.000"));
diseñoXvc2 = ((LTLuz[1, 0] / 2) * diseñoVc2) / vmaxi;
luz9Xvc2.Text = Convert.ToString(diseñoXvc2.ToString("0.000"));
diseñoDvc2 = (LTLuz[1, 0] / 2) - diseñoXvc2;
luz9Dvc2.Text = Convert.ToString(diseñoDvc2.ToString("0.000"));
/////separadores/////
maxs1 = (0.75 * (Av / (100 * 100)) * (FY * 1000) * d) / (diseñoVu - diseñoVc);
maxs2 = (Av * FY) / (0.062 * Math.Sqrt(FC) * BV);
smmbm = d / 2;
if (maxs1 < maxs2 && maxs1 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs1;
unmax = unmax + 1;
luz9UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs1;
luz9UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs1;
luz9UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else if (maxs2 < smmbm)
{
unmax = (diseñoDvu - 0.2) / maxs2;
unmax = unmax++;
luz9UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / maxs2;
luz9UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / maxs2;
luz9UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"));
}
else
{
unmax = (diseñoDvu - 0.2) / smmbm;
unmax = unmax++;
luz9UNmax.Text = Convert.ToString(unmax.ToString("0.0"));
unmed = (diseñoDvu - diseñoDvc) / smmbm;
luz9UNmed.Text = Convert.ToString(unmed.ToString("0.0"));
unbaj = (diseñoDvc - diseñoDvc2) / smmbm;
luz9UNbaj.Text = Convert.ToString(unbaj.ToString("0.0"))
}
}

```

