Escuela Técnica Superior de Ingeniería
Grado en Ingeniería en Tecnologías Industriales

Trabajo Fin de Grado

Optimización topológica en el diseño estructural mediante entornos de programación visual y algoritmos genéticos.

Sara Trinidad Quiñones

Tutor: Fernando Medina Reguera

Sevilla, 3 julio de 2024

Quiero mostrar mis agradecimientos a todos aquellos que me han brindado su apoyo incondicional para llegar hasta aquí,

A mis padres, aquellos que han dado todo y un poco más por qué yo cumpla todas mis metas, son ellos los pilares de mi vida, sustentándome cuando todo parece derrumbarse, ayudándome a superar toda adversidad, no sería la persona que soy a día de hoy sin todo lo que me han enseñado.

A mis mejores amigos/as, quienes han estado día a día acompañándome en este camino y no dejándome nunca sola, celebrando las alegrías y llorando las penas, alegrándose por mis logros como si fuesen los suyos y haciéndome sentir orgullosa y afortunada de tenerlos a mi lado.

A mi pareja, Ángel, quién este último año ha sido imprescindible, afrontando mano a mano todo lo que se presentaba y cogiendo la mía para acompañarme toda una vida, no dejándome nunca de dar el lugar y el respeto que merezco, haciéndome sacar la mejor versión de mí y por encima de todo, haciéndome feliz.

A mi familia, quienes no han dejado nunca de creer en mí, haciéndome sentir siempre en casa con ellos e inmensamente orgullosa de tenerlos.

A mi tutor, Fernando, nada habría sido posible sin su increíble apoyo en todo momento, sirviéndome de guía, motivándome y ayudándome a sacar el máximo de mí.

Por último, a mí, por no haberme rendido nunca y gracias a mi esfuerzo y constancia haber conseguido llegar hasta aquí a mis 21 años.

Gracias.

Resumen

Actualmente, la optimización de procesos se ha vuelto extremadamente relevante, especialmente, en el campo del diseño estructural. Además, el auge de técnicas avanzadas para llevar a cabo la optimización, como los algoritmos genéticos, ha abierto muchísimas posibilidades para mejorar la eficiencia de los resultados y obtener significantes reducciones en costes, ambos están estrechamente ligados al tipo de optimización en la topología, aquella relacionada directamente con el mayor aprovechamiento del material.

Esta relevancia y potencial me han llevado a la creación de este proyecto, cuyo objetivo es desarrollar una aplicación para automatizar un proceso de optimización en el diseño estructural, integrando herramientas de programación visual y algoritmos genéticos. El proyecto se ha enfocado en otros numerosos objetivos más detallados, requiriendo un valioso periodo de estudio previo y familiarización con los tres softwares que se integrarán en esta única aplicación; RFEM, herramienta de análisis de elementos finitos; Python, como lenguaje de programación para las líneas de código más complejas e introducción de operadores genéticos; y Grasshopper, que será la base del desarrollo del programa, donde se aplicará la interconexión con los dos anteriormente mencionados.

Grasshopper nos proporcionará el espacio de trabajo, además de multitud de elementos para poder llevar a cabo la programación visual de la geometría, cálculo y optimización de dos tipos distintos de casos estructurales planteados, una viga simple y posteriormente, una celosía bidimensional. Además, este novedoso software permite una sencilla conexión API haciéndolo accesible y multidisciplinar, a su vez, cuenta con numerosos accesorios que instalar, los cuales permiten a los usuarios orientar el uso de la aplicación a sus respectivas áreas. En mi caso, usaré plugins de RFEM Dlubal, entre otros, para orientarlo al ámbito de la ingeniería civil.

Se buscará la integración de RFEM Dlubal para calcular y analizar las estructuras diseñadas en Grasshopper, asegurando que estas cumplan los criterios de resistencia de materiales. Para ello, se creará una interfaz en Grasshopper que permitirá la generación y modificación de estructuras, la configuración de casos de carga y la correcta exportación e importación de datos con RFEM. Luego, implementaremos algoritmos genéticos mediante bucles iterativos sobre los diferentes parámetros estructurales del modelo, como por ejemplo, sobre los perfiles de la sección trasversal de las vigas, buscando la combinación óptima de estos parámetros mediante un proceso automatizado que ahorra tanto tiempo como esfuerzo.

Para facilitar la implementación y uso de la aplicación por otros ingenieros, detallaré de manera precisa el paso a paso de la creación de este proyecto mediante una guía del usuario, de manera visual y explicativa, para llegar al alcance y entendimiento de otros. Con ello pretendo exponer el gran potencial que demuestra la potente combinación de estos softwares, dando a conocer los conceptos clave y mostrando una aplicación que facilitaría el día a día en la ingeniería civil, motivando a otros al uso y desarrollo de esta.

Palabras clave: Optimización topológica, Diseño estructural, Algoritmos Genéticos, Programación Visual, Grasshopper, RFEM, Python.

Abstract

Currently, process optimization has become extremely relevant, especially in the field of structural design. Moreover, the rise of advanced optimization techniques, such as genetic algorithms, has opened up many possibilities to improve efficiency and achieve significant cost reductions, which are directly linked to topology optimization—specifically, the efficient use of materials.

This relevance and potential have led me to create this project, aimed at developing an application to automate the process of optimization in structural design, integrating visual programming tools and genetic algorithms. The project has focused on several detailed objectives, requiring a valuable period of prior study and familiarization with the three software programs to be integrated into this single application: RFEM, a finite element analysis tool; Python, for more complex programming tasks; and Grasshopper, which will serve as the foundation for the program's development, facilitating the interconnection with the other two software programs mentioned.

Grasshopper will provide the workspace and a multitude of elements to enable the visual programming of geometry, calculation, and optimization of two different structural cases: a simple beam and later, a two-dimensional truss. Additionally, this innovative software allows for easy API connection, making it accessible and multidisciplinary. It also has numerous plugins available for installation, allowing users to tailor the application to their specific fields. In my case, I will use RFEM Dlubal plugins, among others, to orient it towards the field of civil engineering.

The integration of RFEM Dlubal will be sought to calculate and analyze the structures designed in Grasshopper, ensuring they meet material resistance criteria. To this end, an interface will be created in Grasshopper that will allow for the generation and modification of structures, the configuration of load cases, and the correct export and import of data with RFEM. Then, genetic algorithms will be implemented through iterative loops over the different structural parameters of the model, such as the cross-sectional profiles of beams, seeking the optimal combination of these parameters through an automated process that saves both time and effort.

To facilitate the implementation and use of the application by other engineers, I will precisely detail the step-by-step creation of this project through a user guide, in a visual and explanatory manner, to reach and be understood by others. With this, I intend to showcase the great potential demonstrated by the powerful combination of these software programs, highlighting key concepts and presenting an application that would simplify daily tasks in civil engineering, motivating others to use and develop it further.

Keywords: Topological Optimization, Structural Design, Genetic Algorithms, Visual Programming, Grasshopper, RFEM, Python.

ÍNDICE

1. Introducción

1.1. Contexto y Motivación.

Con el paso del tiempo, los ingenieros han buscado siempre mejorar las condiciones, procesos y objetivos existentes, creando un mundo ambicioso y con ansias de saber más. De esta búsqueda surge el concepto de optimización: siempre buscamos cómo hacer más con menos, es decir, obtener un modelo o una solución de la manera más eficiente posible, minimizando a su vez el gasto económico, ya que la economía es la base de todo en la actualidad.

Con las herramientas disponibles hoy en día, nos planteamos si podemos ser partícipes de este avance. Realmente, tenemos los medios para ello, si además aportamos nuestro intelecto, podríamos crear soluciones que faciliten la vida de otros mediante análisis y la creación de programas avanzados que combinen los softwares desarrollados por otros, haciendo de este un mundo dinámico y en constante evolución.

Siempre he sido una alumna destacada, con un amplio interés en la mayoría de los retos que se me han presentado. El diseño estructural es algo que ha llamado mi atención desde temprana edad, al igual que un reciente despertar de interés en la programación. Por lo que he llegado a la conclusión de que no hay mejor manera de combinar estas áreas de interés que desarrollando algo que satisfaga necesidades y, posiblemente, inspire a otros a ver la gran capacidad y potencia de los recursos a nuestro alcance. Aplicaciones como Grasshopper, que desconocía antes de este proyecto, han demostrado ser herramientas con gran proyección en el mundo industrial. Para embarcarme en este camino, he realizado estudios intensivos y cursos durante meses, construyendo una sólida base de conocimientos y sintiéndome cómoda en este nuevo entorno de programación visual, que considero tiene un gran futuro.

Por todo esto, me enfrento al reto de crear una aplicación del algoritmo genético para la optimización topológica en el diseño estructural, utilizando este entorno de programación visual en conjunto con programas auxiliares como RFEM, para el cálculo y análisis de estructuras, y Python, para implementar programación avanzada. Trabajar en este proyecto ha sido una experiencia muy gratificante y espero que pueda ayudar o inspirar a otros a encontrar la misma pasión y entusiasmo.

Ilustración 1. Interfaz Grasshopper en diseño estructural.

1.2. Planteamiento del Problema.

En el ámbito de la ingeniería civil y de edificación, los profesionales de estas áreas a menudo tienen que enfrentarse a desafíos de diseño de estructuras, además, se exige que sean tanto funcionales como viables en cuanto a economía respecta. Un ejemplo típico es la necesidad de crear vigas simples o celosías en las que se minimice el uso del material, pero sin poner en riesgo la integridad estructural. Este proceso del que hablamos, en su forma tradicional de proceder implica un gran consumo de tiempo y esfuerzo, ya que requiere la iteración manual de varios perfiles y divisiones estructurales para encontrar la combinación óptima que cumpla con los criterios de resistencia de materiales, como pueden ser el estado límite último (ULS) y de servicio (SLS).

Concretando un problema más específico, el ejecutado en el desarrollo de este proyecto, podríamos empezar a plantearlo cuando a un ingeniero le mandan a diseñar y estudiar una celosía, la cual ha de tener, por supuesto, el menor perfil posible para reducir costos, además de que al estar hablando de una celosía hemos de tener en cuenta la importancia del número de divisiones con la que contará esta. Esto provocaría el tener que realizar a mano sucesivas pruebas combinando de distintas maneras los perfiles y divisiones, teniendo además que realizar el cálculo de las deformaciones resultantes y verificando la capacidad de la estructura para soportar las cargas aplicadas. Para llevar a cabo dichos análisis de manera eficaz, se usan aplicaciones de elementos finitos como es RFEM de Dlubal, y aunque es relativamente sencillo cambiar los perfiles uno por uno, ajustar el número de divisiones implicaría rediseñar una y otra vez la estructura, lo que resulta en un global una pérdida significativa de tiempo y recursos. Este enfoque manual no solo es enrevesado, sino que también ineficiente, ya que esta búsqueda de obtención del perfil óptimo podría llevar a una infinidad de pruebas, es por tanto que lo descrito no garantizaría el encontrar la solución más económica y menos deformable.

Para resolver este problema, he propuesto la combinación de técnicas de optimización, como son los algoritmos genéticos, con el diseño estructural. Los operadores genéticos son adecuados para este tipo de problemas, debido a su gran capacidad en grandes espacios de búsqueda y además, el encontrar soluciones óptimas en un tiempo razonable. De igual modo, es importante tener en cuenta los beneficios de la incorporación de un software de programación visual por bloques, como es Grasshopper, el cual permite generar y manipular los parámetros de estructuras de manera eficiente. Con este software, se puede configurar una estructura inicial y realizar ajustes sobre ella mediante simples acciones como mover un número en una barra o tan solo pulsar un botón para arrancar un proceso completo.

El enfoque propuesto implica construir un diseño general de celosía en Grasshopper, con dos parámetros variables: el perfil de la sección que será del tipo IPE, seleccionado de una lista predefinida, y el número de divisiones dentro también de un rango establecido. La aplicación iterará entre los distintos perfiles y divisiones, buscando la combinación óptima entre ellos, aquella que minimice la deformación y peso de la estructura. Esta iteración se basa en criterios de “Fitness”, típicos de los algoritmos genéticos, que evaluarán la eficacia de cada una de estas configuraciones que se presenten e implementarán operadores para hallar la mejor opción.

La implementación de esta solución no solo reducirá el tiempo y el esfuerzo requeridos para el diseño estructural, sino que también mejorará la precisión y eficiencia del proceso. Al automatizar la optimización de perfiles y divisiones, se espera lograr estructuras más económicas y robustas, contribuyendo a una práctica de ingeniería sostenible y avanzada.

1.3. Objetivos del Trabajo.

Como bien se menciona en varias ocasiones, este proyecto está centrado en la optimización topológica en el diseño estructural mediante la aplicación de algoritmos genéticos y herramientas de programación visual. Por tanto, el objetivo principal es desarrollar una aplicación que automatice todo este proceso, desde el diseño estructural hasta la búsqueda del resultado más eficiente y económico en cuanto a términos de material. A continuación, se establecerán una serie de objetivos específicos del proyecto que van plenamente ligados al objetivo general descrito previamente:

- **Estudiar y comprender los fundamentos básicos de los algoritmos genéticos.**
  Investigar la teoría, realizar cursos de instrucción previos a la realización del trabajo para luego reflejar los aspectos más relevantes y fundamentales, además de las aplicaciones prácticas de los algoritmos genéticos en la optimización de problemas.

- **Explorar herramientas de programación visual como Grasshopper.**
  Nuevamente realización de cursos para comprender y manejar con soltura las funcionalidades y componentes de su interfaz gráfica, pese a su entorno intuitivo, luego adquirir conocimientos para la generación y modificación de estructuras.

- **Integrar el software de análisis estructural RFEM.**
  Configurar y utilizar RFEM para calcular y analizar las estructuras diseñadas, asegurando que cumplan con los criterios de resistencia de materiales.

- **Desarrollar una aplicación que combine Grasshopper y RFEM.**
  Crear una interfaz en Grasshopper que permita la generación, modificación de estructuras y configuración de casos de cargas en estas, que además, se conecte con RFEM para analizarla estructuralmente.

- **Implementar algoritmos genéticos para la optimización estructural.**
  Desarrollar bucles e implementar algoritmos genéticos que iteren sobre diferentes perfiles y otras variables para encontrar la combinación óptima de parámetros estructurales.

- **Automatizar el proceso de diseño de vigas simples y celosías.**
  Configurar la aplicación para poder cambiar los parámetros desde la interfaz visual y generar automáticamente vigas simples y celosías, optimizando el tiempo y esfuerzo requerido.

- **Proporcionar una guía de usuario y documentación técnica.**
  Elaborar una guía detallada del proceso para facilitar la implementación y el uso de la aplicación por otros ingenieros y profesionales del sector.

- **Validar la precisión y eficiencia de la aplicación desarrollada.**
  Realizar pruebas para comprobar que la aplicación cumple con los criterios establecidos.

- **Fomentar la innovación y el uso de nuevas tecnologías en el diseño estructural.**
  Promover la adopción de herramientas avanzadas y técnicas de optimización, demostrando los beneficios de la programación visual y GA en ingeniería civil.

1.4. Estructura del TFG.

El objetivo principal del proyecto es integrar diversas herramientas para crear un programa que optimice el diseño estructural mediante algoritmos genéticos. En este sentido, explicaremos términos clave como optimización topológica, algoritmos genéticos y elementos finitos, que serán esenciales para comprender el desarrollo y aplicación de este trabajo.

Luego, se describirán los softwares y herramientas que se integrarán para formar un solo programa eficiente. **Grasshopper** será nuestra herramienta principal, conocida por su capacidad para facilitar el diseño estructural a través de su interfaz visual de programación. **Python** jugará un papel crucial en la programación avanzada y en la integración de distintas herramientas, gracias a su flexibilidad y la amplia gama de librerías disponibles. **RFEM**, por su parte, es una herramienta esencial para el análisis estructural, que nos permitirá realizar cálculos precisos y detallados sobre nuestras estructuras. Describiremos sus principales funcionalidades y cómo se integrará con **Grasshopper y Python** para garantizar un flujo de trabajo coherente y eficiente.

En la siguiente sección, profundizaremos en los algoritmos y técnicas que utilizaremos, comenzando con la descripción de la conexión **API** entre **Grasshopper, Python y RFEM**, vital para asegurar que los datos se transfieran y procesen correctamente entre las diferentes aplicaciones, permitiendo una integración completa. Además, se estudian los conceptos esenciales de los **algoritmos genéticos (GA)**, que son la base de la metodología de optimización usada. Se detalla cómo estos algoritmos se aplicarán en la optimización estructural, definiendo el proceso iterativo y los mecanismos de selección, mutación, elitismo y cruce que utilizaremos para mejorar continuamente nuestras soluciones.

Caso 1: Viga Simple

En la primera etapa del desarrollo del proyecto, abordaremos un caso simple para probar el funcionamiento del planteamiento y determinar su viabilidad para modelos más complejos. Comenzaremos con una viga simple, explicando cómo crear su geometría y establecer las condiciones necesarias, como los casos de carga, nodos, miembros, materiales y secciones. Posteriormente, exportaremos esta configuración a RFEM para realizar los cálculos necesarios. Una vez obtenidos los resultados en formato CSV, los visualizaremos en Grasshopper y estableceremos las condiciones de contorno para un análisis iterativo. Este bucle recorrerá los perfiles IPE, comprobando los criterios de deformación hasta encontrar la solución óptima que cumpla con los requisitos establecidos.

Caso 2: Celosía en 2D

En la segunda etapa, llevaremos a cabo un caso más complejo: una celosía en 2D. Procedemos de manera similar al caso anterior, comenzando con la creación de una geometría, aunque más elaborada, y estableciendo todas las condiciones necesarias. En este caso, introduciremos algoritmos genéticos para manejar dos variables iterativas: el perfil de la sección y el número de divisiones de la celosía. Generaremos una población inicial y utilizaremos el cálculo del peso de la estructura para definir la función de fitness, que será el criterio de selección del mejor individuo. Implementaremos un bucle iterativo anidado a este para que se lleven a cabo las primeras generaciones del GA y posteriormente incorporaremos los mecanismos de mutación, elitismo y cruce para optimizar el diseño, utilizando código avanzado en módulos de Python para gestionar estos procesos de manera eficiente e introducirnos en la producción de más generaciones de este GA.

Finalmente, evaluaremos los resultados obtenidos en el modelo definitivo de esta aplicación y compararemos la eficiencia y eficacia del nuevo este programa frente a los métodos tradicionales. Discutiremos el posible impacto que pueda tener la aplicación en el campo de la ingeniería civil y exploraremos posibles mejoras y aplicaciones futuras para esta metodología. Con este proyecto no solo busco optimizar estructuras de manera más eficiente, sino también inspirar y facilitar el trabajo de otros ingenieros, demostrando el poder de las herramientas modernas y las técnicas avanzadas de optimización.

2. Términos y definiciones

2.1. Definiciones técnicas

2.1.1. Optimización Estructural

La optimización estructural es un proceso clave de la ingeniería que tiene por objetivo conseguir el cumplimiento de unos requisitos concretos sobre el rendimiento de una estructura de la manera más eficiente y funcional posible, además, ha de tenerse siempre en valor la seguridad de éstas para garantizar a su vez la seguridad de las personas.

Existen diversas técnicas y métodos para poder realizar este proceso, una de ellas y la más estudiada a día de hoy, es el análisis de elementos finitos (FEM), con el que se pueden explorar configuraciones que permitan obtener soluciones óptimas optimizando parámetros como la minimización del peso, maximización de la resistencia, etc.

La optimización estructural es fundamental en sectores como la aeronáutica, la automoción, la arquitectura y la construcción, donde se busca constantemente la mejora del rendimiento, la seguridad y la reducción de costes.

2.1.2. Método de los Elementos Finitos (FEM)

El método de los elementos finitos (FEM) es un método fundamental en la computación muy utilizado en la ingeniería para resolver problemas de distinto tipo. Esta técnica consiste en dividir la geometría del objeto que estamos estudiando en muy pequeñas partes a las que llamaremos “elementos”, estos elementos representarán el espacio continuo del problema. Cada uno de los elementos estará definido por un conjunto finito de parámetros que serán los que se encargan de describir su comportamiento en unos puntos característicos del espacio a los que llamaremos nodos.

Los nodos estarán interconectados como si de una red de pescar se tratase, es por eso por lo que lo llamaremos formación de malla y este será el concepto esencial gracias al cual podremos resolver las ecuaciones matemáticas que modelan el comportamiento físico del sistema. La precisión de los resultados de los resultados obtenidos dependerá de la cantidad y el tipo de elementos que utilicemos en la malla, cuantos más elementos, mayor precisión en el análisis y por tanto mayor eficacia en la obtención de resultados óptimos.

El FEM se aplica en multitud de ocasiones para diagnosticar problemas estructurales mediante simulaciones, consiguiendo con ello información detallada sobre los desplazamientos, deformaciones y tensiones. A su vez, posee la capacidad de adaptarse a una gran variedad de

situaciones, desde análisis térmicos y acústicos hasta simulaciones dinámicas y electromagnéticas. Además, integrarlo con aplicaciones de diseño en herramientas de ingeniería facilita una representación geométrica precisa acompañada de una evaluación eficiente, como por ejemplo en el software RFEM de Dlubal, gracias a ellos conseguimos optimizar procesos en el diseño y análisis en las investigaciones y creaciones diarias.

2.1.3. Análisis de Sensibilidad

El análisis de sensibilidad es crucial en proyectos de ingeniería y optimización, ya que es considerada una técnica utilizada para ver como cambios en diferentes variables afectan al resultado final de un modelo. Gracias a ello, podemos identificar que variables son más importantes y cómo influyen en el resultado final.

Además, es de vital importancia ya que nos garantiza una optimización eficiente, evitando el desperdicio de recursos al centrarse solo en aquellas variables que realmente influyen en los objetivos para la optimización, por otro lado, nos proporcionará información detallada sobre el comportamiento del sistema ante cambios en las variables.

Al indagar en los pequeños cambios que pueden influir significativamente en variables críticas, maximizamos tanto la eficiencia como la capacidad de adaptación frente a condiciones variables y desafíos repentinos. Como subraya Luis Manuel Villa García en 2014, *"El análisis de sensibilidad revela la senda más directa para lograr ajustes efectivos que transformen el comportamiento estructural según nuestras metas*".

2.1.4. Optimización Topológica

La optimización topológica es una metodología avanzada de diseño que emplea algoritmos computacionales para redistribuir material dentro de una geometría, como ejemplo, el que veremos en el desarrollo del proyecto, una celosía donde se redistribuirán las posiciones de sus barras interiores generando un mayor o menor número de divisiones. El objetivo de todo esto será la mejora del rendimiento estructural, caracterizado por la eliminación estratégica de material innecesario, reduciendo el peso sin comprometer la integridad estructural.

La aplicación de esta permite explorar diseños estructurales innovadores y eficientes, no solo minimizando el peso de la estructura sino también mejorando su resistencia y durabilidad, asegurando los estándares de calidad establecidos en la industria.

"The purpose of topology optimization is to find the optimal lay-out of a structure within a specified region. The only known quantities in the problem are the applied loads, the possible support conditions, the volume of the structure to be constructed and possibly some additional design restrictions…", O.Sigmund and M.P.Bendsøe, Topology Optimization, 2003.

La técnica más común para esto es el método de elementos finitos (MEF), este considera el diseño geométrico y divide la estructura en partes, evaluando la rigidez de cada elemento y eliminando material. Además, resuelve desafíos de diseño comunes como esfuerzos térmicos, optimizando el equilibrio entre tamaño, peso y rendimiento.

2.1.5. Modelado Estructural

El modelado estructural es una técnica usada para representar de manera simplificada la estructura de sistemas físicos, tales como puentes, edificios, vehículos o cualquier estructura mecánica. Gracias a esto entendemos, identificamos y optimizamos el comportamiento de estructuras bajo diversas condiciones, para ello, usamos herramientas matemáticas, computacionales, y algoritmos, todos ellos nos ayudan a hacer posible la simulación de fuerzas, tensiones, deformaciones y otros aspectos críticos que afectan de manera directa al rendimiento de la estructura.

Además, es usado para facilitar el diseño de estructuras más eficientes y seguras, mediante el estudio de las zonas y la identificación de aquellas que requieren un esfuerzo mayor, también las zonas de las que se puede reducir material. A su vez facilita la identificación temprana de fallos estructurales antes o durante la construcción de esta y permite realizar simulaciones de casos para poder realizar pruebas y verificaciones, asegurando cumplir normas y especificaciones técnicas muchas veces incluidas en los softwares, como es el caso del que usaremos en este proyecto, RFEM.

A continuación, definiré los elementos que ha de tener un modelo estructural, que son a su vez los que veremos descritos posteriormente en la guía de usuario de creación de los modelos iniciales de estructuras de este propio proyecto, estos son:
Primero crear la geometría, representando las dimensiones de la estructura; luego, tenemos que proceder a la asignación de un material que conlleva una serie de propiedades específicas de este, como por ejemplo su densidad que será relevante para el cálculo del peso de la estructura como veremos luego.
En segundo lugar, estableceremos sus distintos casos de carga (como el peso propio, cargas impuestas), pueden ser cargas estáticas o dinámicas, y también definir como serán las restricciones en sus movimientos, sus puntos de apoyo.
Finalmente, tenemos que llevar a cabo técnicas avanzadas de análisis para evaluar el comportamiento de la estructura y poder posteriormente obtener e interpretar resultados.

Usar este tipo de análisis supone una gran cantidad de ventajas, ya que además de obtener precisión en los diseños creados, podremos reducir costes y mejorar la eficiencia en los procesos de diseño y construcción, podemos encontrarnos ante casos de modelaje en 2D o 3D pero en ambos podemos obtener grandes resultados con esta técnica.

2.1.6. Cargas Estáticas

Las cargas estáticas son aquellas que pese al paso del tiempo, mantiene constante su magnitud, dirección y localización. Estas se dividen en permanentes, aquellas que incluyen los elementos fijos de la estructura, como el peso propio; y accidentales, que resultan del uso de la estructura y condiciones climáticas (como lluvia y nieve, pero no viento).

La comprensión y el análisis de las cargas estáticas aseguran que las estructuras puedan soportar tanto su propio peso como las fuerzas adicionales derivadas del uso y entorno, garantizando su seguridad y estabilidad.

2.1.7. Cargas Dinámicas

A diferencia de las estáticas, son estas las que varían continuamente, como la fuerza del viento o movimientos sísmicos, una de sus cualidades es la variación de un valor nulo al límite causando esto significativas aceleraciones. Estas cargas de las que hablamos pueden ser puntuales o distribuidas a lo largo de todo el elemento estructural.

2.1.8. Deformación Estructural

La deformación estructural ocurre cuando un elemento bajo carga cambia de forma o tamaño debido a fuerzas mecánicas externas, variaciones de temperatura o apoyos que ceden. Esta deformación puede ser elástica, si desaparece cuando finaliza la aplicación de la carga, o permanente, si continúa. En la naturaleza, no existen deformaciones completamente elásticas, siempre hay una pequeña deformación residual. Es crucial estudiar las deformaciones además de los esfuerzos internos y tensiones, para asegurar que se mantengan dentro de los límites aceptables y, a veces, dimensionar piezas para garantizar su rigidez en lugar de su resistencia. La deformación de una barra, por ejemplo, implica un desplazamiento y una rotación, y la evaluación de estos factores es esencial para un diseño estructural seguro y eficiente.

Cuando les aplicamos cargas, todas las estructuras experimentan cambios en su geometría, que aunque a priori parecen imperceptibles, deben medirse para hacer un buen análisis con determinada precisión. Parámetros como el módulo de elasticidad o el área de la sección transversal, como será analizado en este proyecto, son fundamentales para determinar la magnitud de la deformación. Esto permite ajustar la sección de la pieza para aumentar su rigidez y reducir las deformaciones, garantizando así la integridad estructural.

2.1.9. Resistencia de materiales

La resistencia de materiales es una rama de la ingeniería estructural, civil, mecánica y de materiales, centrada en el análisis del efecto de las fuerzas sobre sólidos deformables. Su objetivo principal es comprender como elementos estructurales soportan esfuerzos y fuerzas sin destruirse o estropearse. Además de la ruptura, estos sólidos que se someten a fuerzas exteriores también pueden sufrir deformaciones, y dentro de ellas, diferenciamos dos tipos: elásticas, cuando se deja de aplicar la fuerza el cuerpo vuelve a su forma y dimensión original; o permanentes, donde el cuerpo no regresa a su estado inicial pese al paro de actuación de la fuerza.

Los esfuerzos en los materiales fruto de estas fuerzas aplicadas, se clasifican de distinta manera según como actúan estas. Por ejemplo, tracción es cuando se estira el material al aplicar una fuerza perpendicular a su sección transversal produciendo un alargamiento en el cuerpo. Por otro lado, es también importante la compresión, aplicando una fuerza similar, pero en dirección contraria, acortando el cuerpo, luego, es importante mencionar también los esfuerzos cortantes, estos implican fuerzas opuestas en planos contiguos que intentan deslizar una sección de material sobre otra y finalmente la flexión, que lo que trata es de doblar el cuerpo alargando partes y acortando otras.

Además, la resistencia de materiales también aborda el análisis de tensiones y pandeo, es por eso y más que el análisis profundo de la misma es una parte fundamental en el desarrollo de todo tipo de proyectos diariamente, buscando siempre eficiencia y seguridad.

2.1.10. Análisis Estructural

El análisis estructural es un proceso fundamental en la ingeniería estructural, este implica la evaluación detallada de como cargas y fuerzas afectan a una estructura. Su objetivo principal será asegurar siempre la seguridad y funcionalidad de estas construcciones bajo las cargas que han de soportar, obtenidas mediante previo análisis. El análisis estructural considera factores como los materiales utilizados, la geometría de la estructura y las cargas aplicadas, realizándose este en diversas etapas de los proyectos. En el caso de este proyecto, nunca se deja de realizar ya que cada vez que le damos a inicial la aplicación de nuevo, procede a realizar uno tras otros múltiples análisis estructurales para obtener parámetros importantes en la ejecución del programa.

Durante el análisis se examinan elementos estructurales como vigas y celosías (en mi caso, pero hay muchos más) para entender como responden a cargas como el peso propio, la carga de viento u otras impuestas por nosotros. Este se lleva a cabo mediante varios métodos pero el más usado es el definido previamente, análisis de elementos finitos (FEA) para los problemas más complejos como es el caso presente, además, se usan softwares especializados que facilitan el modelado y simulación de comportamientos. En este proyecto tendremos presente RFEM para FEA y Grasshopper para facilitar el modelado estructural, dos softwares que se describirán posteriormente y que tienen un potente uso en la industria civil.

2.1.11. Estabilidad y Rigidez Estructural

La estabilidad y la rigidez son conceptos fundamentales en el diseño y la ingeniería estructural, cruciales para garantizar la seguridad y durabilidad de las construcciones. Combinadas, pueden asegurar que las estructuras mantienen su integridad bajo cargas variables y condiciones ambientales adversas, cumpliendo altas normas de resistencia y funcionalidad.

La estabilidad estructural se refiere a la capacidad de una estructura para mantenerse en equilibrio soportando cargas externas, resistiendo movimientos como deslizamientos, vuelcos o deformaciones. Esta depende de la distribución adecuada del peso, la forma de la estructura y la calidad de los apoyos usados, su análisis puede clasificar las estructuras en internamente estables o inestables (antes de haber añadido los apoyos estructurales). Para asegurar la estabilidad, se deben usar ecuaciones de equilibrio que consideren las reacciones necesarias en los apoyos y verificar que sean adecuadas las conexiones para soportar cargas.
Los equilibrios de fuerzas son:

$$\Sigma F_x = 0 \quad (sumatorio\ de\ fuerzas\ en\ dirección\ horizontal\ (x))$$

$$\Sigma F_y = 0 \quad (sumatorio\ de\ fuerzas\ en\ dirección\ vertical\ (y))$$

$$\Sigma M = 0 \quad (sumatorio\ de\ momentos)$$

La rigidez estructural se relaciona con la capacidad de un elemento o sistema para resistir las deformaciones bajo cargas. Es esencial para garantizar que una estructura mantenga su forma sin sufrir deformaciones excesivas que comprometan su seguridad. La rigidez depende del módulo de elasticidad del material usado (cuanto mayor, más rígida la estructura), momento de inercia y geometría del elemento. Para aumentar la rigidez se toman medidas como aumentar la sección transversal de los elementos, que es la que veremos y usaremos a lo largo del desarrollo de esta aplicación, este concepto será muy importante en ella.

2.1.12. Análisis de Tensiones y Deformaciones

En el ámbito estructural, tanto la tensión como la deformación juegan roles fundamentales en el comportamiento de materiales sometidos a cargas externas. La tensión se refiere a la fuerza interna que actúa sobre un material debido a una carga aplicada, medida en unidades de fuerza por área (ejemplo: Newton por m^2). Es crucial para determinar cómo los materiales responden a las cargas, por ejemplo, las altas tensiones pueden indicar que el material está cerca de su límite de resistencia.

Por otro lado, la deformación es el cambio de forma o dimensión de un material como respuesta a estas tensiones aplicadas, pueden ser elásticas o plásticas, como mencionamos anteriormente. El análisis de deformaciones examina como la estructura cambia de forma bajo carga, evaluado mediante el tensor de deformaciones, que describe la deformación lineal y distorsión angular sobre un material en un punto dado.

Por tanto, las deformaciones están relacionadas con las tensiones a través de las relaciones tensión-deformación del material, estas relaciones son variables y se expresan normalmente en diagramas o curvas que muestran como la tensión cambia en función de la deformación. Por ejemplo, en materiales elásticos lineales como el acero, la relación tensión-deformación es proporcional y reversible.

El uso de aplicaciones como RFEM es crucial en proyectos de ingeniería, ya que, entre muchos otros, nos permite realizar el cálculo de deformaciones y en el desarrollo de este proyecto cobrarán vital importancia las deformaciones verticales de la estructura ya que formarán parte esencial de la aplicación creada. Gracias al uso de estas, aplicándolas como condición a otras funciones y aspectos, no solo estamos garantizando la seguridad y eficiencia del modelo, sino que también minimizamos riesgos al proporcionarnos esto visiones detalladas y precisas del comportamiento estructural ante diferentes condiciones de carga y escenarios.

2.1.13. Esfuerzo Computacional

Aunque no he encontrado información detallada sobre este concepto, me gustaría ofrecer mi propia definición basada en mi comprensión. El esfuerzo computacional se refiere a los recursos y capacidades necesarios que una aplicación como RFEM, Grasshopper o Python requieren para llevar a cabo análisis complejos y precisos. Utilizando métodos avanzados como el método de los elementos finitos en RFEM o bucles incluyendo códigos de programación visual y escrita que generan geometrías como hace Grasshopper. Esto implica manejar grandes volúmenes de datos y calcular deformaciones y otros parámetros de manera eficiente.

Para mi proyecto, es importante poder extraer rápidamente deformaciones verticales (Uz) y otros datos clave, permitiendo así tomar sacar conclusiones informadas sobre diseño, seguridad estructural y optimización de recursos. Sin embargo, uno de los desafíos más frecuentes que he enfrentado, al igual que muchos otros usuarios, es el tiempo que lleva realizar estas simulaciones. A pesar de contar con un ordenador potente para desarrollar el proyecto, las simulaciones pueden tomar relativamente mucho tiempo (hablamos de minutos, lo cual es considerable en este contexto). Esto se debe al esfuerzo computacional necesario para realizar múltiples pruebas y ajustes durante el proceso de desarrollo de la aplicación. No obstante, para los usuarios finales que utilicen el modelo, obtener los resultados deseados no les llevará más de unos 10 minutos, gracias a la eficiencia del esfuerzo computacional integrado en la aplicación.

3. Materiales y métodos

3.1. Herramientas y Software.

3.1.1. Grasshopper

Grasshopper es un lenguaje de programación visual integrado en Rhinoceros, el cual permite a diseñadores crear algoritmos generativos sin necesidad de recurrir a la programación tradicional únicamente con líneas de código. Es más, su entorno gráfico es muy atractivo e intuitivo por lo que puede llamar la atención de un público más amplio, además de diseñadores e informáticos.

Ilustración 2. Logo aplicación Rhinoceros.

Hemos de comenzar por hablar de Rhinoceros, comúnmente conocido como Rhino, es una herramienta de modelado en superficies de forma libre. Aquello en lo que se basa su diferenciación es la capacidad para albergar una amplia gama de herramientas de diseño computacional, como Grasshopper, lo que la hace relevante en diversos campos del diseño.
Fue inventado en 1980 por la empresa estadounidense Robert McNeel & Associates y lo basaron en un modelo matemático NURBS (Non-Uniform Rational Basis Spline) que permite representar curvas con precisión. Además, Rhino es compatible con la gran mayoría de programas de diseño y programación, por lo tanto es considerada como flexible, permitiendo a los usuarios moverse entre diferentes campos del diseño sin necesidad de aprender múltiples programas.

- **Ventajas de usar Rhino:**

1. Permite modelar formas complejas, integrando todo el campo del diseño en una sola plataforma.
2. Es compatible con muchos formatos de software, facilitando un flujo de trabajo continuo.
3. La licencia tiene una buena relación calidad-precio.
4. Cuenta con una gran cantidad de plugins que amplian sus capacidades.

En cuanto a Grasshopper respecta, es una aplicación de diseño asistido por computadora usada principalmente para modelado paramétrico, donde las geometrías se generan y manipulan a través de parámetros y reglas definidas por los usuarios.

Ilustración 3. Logo aplicación Grasshopper.

- **Aplicaciones industriales de Grasshopper.**

  Este software ha encontrado aplicaciones diversas en la industria debido a su capacidad para manejar diseños complejos y adaptativos de manera eficiente, y algunas de las áreas donde se incluye son:

  1. Arquitectura, ya que permite la creación de estructuras complejas y formas arquitectónicas innovadoras, se pueden diseñar fachadas, estructuras y elementos decorativos que se adaptan a diferentes condiciones y requisitos.
  2. Diseño Industrial, desde mobiliario hasta dispositivos electrónicos ya que Grasshopper facilita la exploración de funciones optimizadas.
  3. Fabricación digital, integrada con tecnologías de fabricación como impresión 3D, permitiendo la creación de piezas personalizadas y estructuras complejas difíciles de producir con métodos tradicionales.
  4. Ingeniería civil, la que fundamenta las bases de este proyecto, ya que podemos aplicar Grasshopper en el diseño de infraestructuras y puentes, donde la optimización estructural y el análisis paramétrico son cruciales.

- **Relación con el diseño estructural.**

  En el ámbito del diseño estructural, Grasshopper se usa para el análisis y la optimización de modelos en función de diferentes parámetros, además, permite a los ingenieros estructurales simular el comportamiento de las estructuras bajo diversas cargas y condiciones. Les permite también optimizar el uso de materiales y mejorar la eficiencia y seguridad de los diseños. La integración de algunos plugins como Dlubal RFEM facilita mucho cualquier proceso relacionado con esto.

- **Proyección de futuro.**

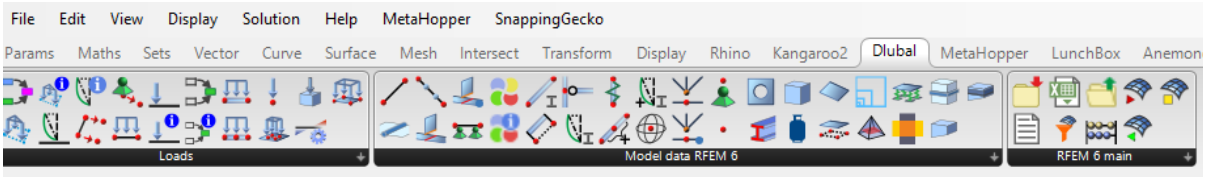
  Aunque Grasshopper es una herramienta poderosa, aun es relativamente desconocida fuera de ciertos círculos profesionales. Sin embargo, su uso está en crecimiento, sobre todo en industrias que buscan innovación en diseño y fabricación, y se espera que a medida que los usuarios se familiaricen con sus capacidades Grasshopper se convierta en un estándar en el diseño paramétrico y la fabricación digital. Su flexibilidad y la comunidad activa de usuarios y desarrolladores aseguran que continuará evolucionando, y esto es algo que podemos ver con sus nuevas versiones ya que cada vez se producen antes.

- **Software y funcionalidad.**

  Grasshopper es gratuito y viene incluido con Rhinoceros, aun que Rhino es un software de pago. Dentro de ella, encontramos una enorme variedad de plugins gratuitos y de código abierto disponibles que extienden las funcionalidades de Grasshopper, permitiendo a los usuarios abordar una amplia gama de problemas de diseño y fabricación. Todo ello disponible en plataformas como "Food4rhino", donde es sencillo buscar, descargar e instalar cualquier tipo de ellos con tan solo tener tu cuenta de usuario en Rhino.
  Además, les permite crear y manipular geometrías complejas a través de una interfaz visual basada en nodos. Los usuarios arrastran y conectan componentes, los cuales representan funciones y datos, en un lienzo digital para construir algoritmos que generan formas y estructuras. Cada componente tiene entradas y salidas que pueden conectarse para formar redes de datos y operaciones, esto recuerda en cierto modo a las lógicas de control de sistemas avanzados ya que también se programan por cajitas, estas vienen también incluidas en bloques de la aplicación. Todo esto permite una gran creatividad y flexibilidad en el diseño, ya que los usuarios pueden experimentar y ajustar parámetros en tiempo real para ver cómo afectan al diseño final.

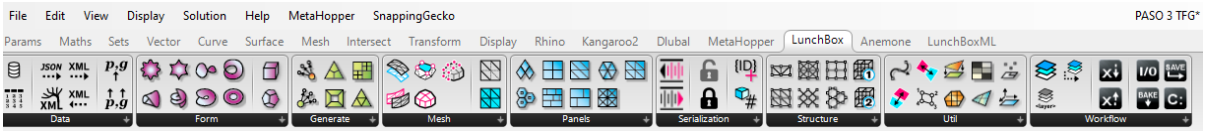
- **Plugins de Grasshopper presentes en este proyecto.**

  1. Dlubal RFEM.



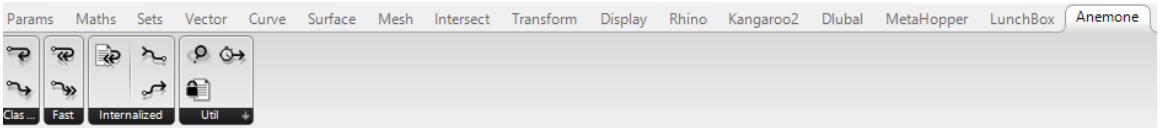
  Es probablemente uno de los más importantes en cuanto a términos de diseño estructural, ya que facilita su análisis avanzado directamente desde Grasshopper. Permite además hacer análisis de elementos finitos (FEM) para evaluar la resistencia y la estabilidad estructural de los diseños paramétricos. Con este plugin veremos como convertir líneas en vigas, aplicar cargas, exportar modelos a RFEM, calcularlos desde GH, establecimiento de parámetros tales como sección o material, apoyos nodales, ajustes estructurales, extracción de parámetros de cálculo a archivos, etc.
  Una multitud de funciones increíble, por tanto, será el plugin que asiente las bases de este proyecto.

  2. Lunchbox.



  Esencial, con él podremos construir celosías "2D Truss" en el caso que desarrollaremos posteriormente, y muchísimos otros elementos de una manera muy sencilla. Este plugin proporciona una serie de herramientas matemáticas y de modelado avanzado, incluyendo funciones para generación de formas complejas, creación de estructuras de paneles y otras herramientas útiles como algoritmos de optimización y análisis de datos.

3. Anemone.



Será el plugin encargado de añadir bucles y controlar flujos iterativos, por ende también crea algoritmos iterativos y recursivos, ampliando las capacidades de programación visual más allá de un flujo de trabajo lineal estándar.
Es ideal para proyectos como este, que requieren de iteraciones y procesos de retroalimentación, como simulaciones dinámicas, optimización de trayectorias y procesos de crecimiento o evolución de formas, que no será este caso pero también es de una gran utilidad.

4. Telepathy.

Es un plugin diseñado para la transferencia de datos en tiempo real entre múltiples instancias de Grasshopper ejecutándose en diferentes máquinas. Facilita la visualización y simplifica los modelos al conectar los cables de manera inalámbrica quitando nudos y enredos que puedan quedar en medio del canvas.

5. Python for GH.

No es un plugin, pero es un componente clave ya que Grasshopper contiene un módulo que integra el lenguaje de programación Python dentro de él, permite a los usuarios escribir scripts para realizar operaciones avanzadas, manipular datos y automatizar tareas complejas.

Cabe destacar que al inicio de este proyecto era Rhino 7 con el que se trabajaba y el bloque de Python tenía algunas limitaciones, es por ello que conseguimos obtener una licencia de Rhino 8 que incluía todas estas mejoras en el bloque y con la que pude continuar el correcto desarrollo del proyecto, aprovechando al máximo el potencial del lenguaje de Python.

- **Elementos Clave de Grasshopper**

1. **Canvas:**
   - **Buscador:** Encuentra y ubica objetos rápidamente.
   - **Barra de herramientas:** Acciones rápidas y ajustes.
   - **Componentes:** Bloques de construcción.
   - **Cinta de unión de componentes.**
   - **Contenedor/Parámetro:** Almacenan datos, pero no realizan transformaciones.
2. **Agarre de Entrada/Salida:**
   - Son medios círculos en los componentes para conectar cables que transportan datos.
3. **Vista Previa:**
   - Los componentes no seleccionados se muestran en rojo, y los seleccionados en verde en la ventana gráfica de Rhino.
4. **Cables:**
   - Indican la estructura de datos transportada: línea simple para un único dato, doble para listas y punteada para árboles de datos.
5. **Árboles de Datos en Grasshopper:**
   - Los árboles de datos son estructuras jerárquicas que organizan datos en listas anidadas, similares a las carpetas de un ordenador. Facilitan la gestión y manipulación de grandes conjuntos de datos y geometrías en Grasshopper, permitiendo un control preciso y eficiente en la creación de algoritmos complejos.

- **Comandos y Parámetros Básicos:**

  - **Point:** Almacena datos persistentes.
  - **Curve:** Representa curvas geométricas.
  - **Surfaces:** Colección de superficies.
  - **Brep:** Superficies y polisuperficies.
  - **Mesh:** Mallas poligonales.
  - **Circle:** Círculos con un dominio de 0 a 2π.
  - **Scribble:** Notas rápidas.
  - **Geometry parameter:** Colección de geometría 3D.
  - **Data:** Colección de datos genéricos.

- **Herramientas de Organización y Manejo de Parámetros**

  - **Lock Solver:** Bloquea componentes para prevenir el crash del programa.
  - **Radial Menu:** Acceso rápido a funciones comunes como preview, group y clusterize.
  - **Wiring:** Añadir, borrar y mover cables entre componentes para organizar el flujo de datos.
  - **SLIDER:** Ajuste rápido de valores numéricos.
  - **PANEL:** Para notas y visualización de datos en tiempo real.
  - **LISTA DE VALORES:** Permite seleccionar entre varios valores predefinidos.

3.1.2. RFEM

RFEM es una herramienta de análisis de elementos finitos desarrollada por Dlubal, utilizada principalmente para el modelado y análisis de estructuras en ingeniería civil. En este caso usaremos la sexta generación de este software, RFEM 6, que permite modelar de manera eficiente estructuras complejas, realizar análisis estructurales y dinámicos, y dimensionar elementos como barras, placas y sólidos. La flexibilidad y facilidad de esta la convierte en una solución ideal para proyectos en sectores como la construcción, además de que podemos incorporar en ella el uso y establecimiento de materiales comúnmente empleados para hacer estos análisis de la manera más precisa posible, materiales como son el hormigón, acero, madera y vidrio.

Ilustración 4. Logo aplicación RFEM de Dlubal.

- **Características principales y algunas nuevas implementadas en esta última versión.**

1. Permite seleccionar parámetros específicos para el análisis y mostrarnos las deformaciones producidas en diferentes tipos de barras y combinaciones de carga, proporcionando una gran flexibilidad y precisión en los cálculos.
2. Mejora el rendimiento al realizar cálculos paralelos de casos y combinaciones de carga mediante el uso de varios procesadores.
3. Proporciona visualizaciones claras de los esfuerzos internos en las tablas de resultados y gráficos, permitiendo una fácil identificación de valores con colores.
4. Ofrecen una muestra configurable de diagramas de resultados, pudiendo definir y ajustar zonas para visualizaciones más específicas y precisas.
5. Muestra fórmulas de diseño aplicadas en cálculos, referenciadas a norma aplicable.
6. Facilita la adecuación de las unidades para los datos de entrada, cargas y resultados.
7. Mejora de los informes: ofrece un entorno interactivo para la modificación de capítulos, permitiendo la importación y exportación de distintos tipos de archivos, permite añadir gráficos e incorpora un editor para adaptarlo a cada usuario.
8. Incorporación de Python, se pueden crear scripts para editar o generar estructuras automáticamente.

RFEM destaca por su interfaz de usuario intuitiva, similar a la del software CAD, que simplifica el modelado y la edición de estructuras mediante menús contextuales. Además, ofrece funcionamiento y configuraciones en inglés, español, alemán y francés, entre otros, lo que aumenta la accesibilidad para usuarios de todos los lugares del mundo. Por todo esto entre muchas más características es por lo que se posiciona como solución completa y adaptable que combina herramientas avanzadas con una interfaz amigable para satisfacer las necesidades de análisis estructural más exigentes.

- **Modelado en RFEM**

Ofrece una amplia variedad de materiales y secciones que facilitan el modelado de estructuras con placas y vigas. Estas bibliotecas incluyen sección transversales estándar y personalizadas, así como distintos tipos de materiales como madera o acero. Ofrece un sistema de filtración en estas bases de datos para satisfacer las necesidades de cada usuario, permitiendo la personalización del modelo.

En el caso que estudiaremos posteriormente, serán de gran importancia las secciones ya que iteraremos entre una serie de ellas ya establecidas en esa base de datos.
Además, permite definir vistas personalizadas desde diferentes ángulos para facilitar la visualización y evaluación de los resultados, esencial en todo trabajo al proporcionar representaciones claras. El software soporta modelos en 1D, 2D y 3D, lo que nos permite manejar vigas y cerchas, pero también complejas estructuras tridimensionales, esta flexibilidad es crucial para adaptarse a todo tipo de escenarios en el diseño estructural.

RFEM también incluye capacidades avanzadas de análisis no lineal para estudiar el comportamiento estructural bajo cargas no lineales, incluyendo el modelizar estas no linealidades en barras y apoyos, para aquellas estructuras que experimentan grandes deformaciones, puede evaluar la estructura bajo condiciones extremas. Por otro lado, las propiedades se pueden definir de manera variable pudiendo hacer numerosos cambios y modificaciones en ellas según vaya siendo necesario para un análisis estructural completo.

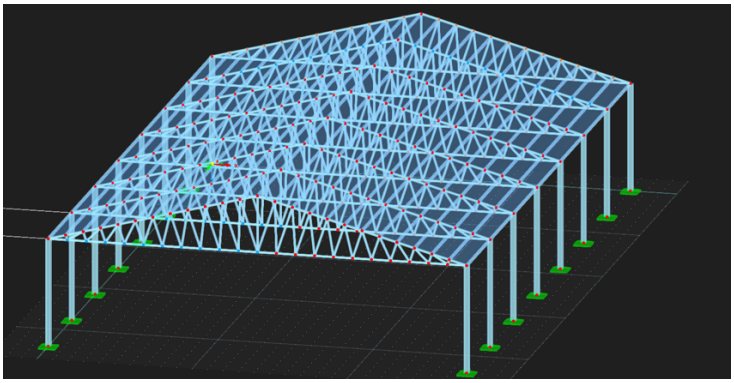


Ilustración 5. Modelado tridimensional realizado por mí en RFEM.

- **Casos de carga y combinaciones, aplicando normativas europeas.**

RFEM permite crear casos de carga personalizados y generar combinaciones de acciones según categorías definidas, asegurando la evaluación de las condiciones de carga para diferentes situaciones de proyecto. Además, han incorporado un asistente para cargas para facilitar la definición de estas en el proceso de modelaje.

1. Generación de cargas de nieve: conforme a normativas como EN 1991-1-3.
2. Generación de cargas de viento: conforme a la norma EN 1991-1-4.
3. Generación de cargas en barras desde cargas superficiales.
4. Generación de cargas de barras desde cargas lineales.
5. Expresiones de combinación: evaluación del estado límite último (ULS) y el estado límite de servicio (SLS). Esta función permite seleccionar varias situaciones de proyecto según normativas específicas, como ULS(STR/GEO) para condiciones permanentes y transitorias, usadas posteriormente en este proyecto.

- **Interfaces.**

  - **RHINO**
    RFEM proporciona una interfaz directa con Rhino, permitiendo a los usuarios intercambiar modelos y datos entre ambos programas de manera fluida, a continuación, mencionar algunas de sus capacidades clave:
    - Importación y exportación de geometría.
    - Transferencia de modelos.
    - Sincronización de datos

  - **GRASSHOPPER**
    Dentro de Rhino, RFEM también se integra con Grasshopper, lo que permite a los usuarios crear algoritmos personalizados para generar y manipular geometría paramétrica, que luego se puede importar a RFEM para el análisis estructural, es lo que veremos en el desarrollo posterior del proyecto.

  - Se conecta también con **PYTHON** y con **EXCEL** a través de la importación y exportación de datos, resultados y la automatización de procesos.

3.1.3. Python

Python es un lenguaje de programación muy avanzado, creado en 1990 por Guido Van Rossum, ha destacado por su simplicidad, legibilidad y versatilidad en una amplia gama de aplicaciones. Este lenguaje se usa en campos tan diversos como la inteligencia artificial, desarrollo web, análisis de datos, automatización y el más reciente, aquel que veremos en el desarrollo de este proyecto es la optimización mediante algoritmos genéticos. Cuenta con una sintaxis clara y concisa, que hace más fácil la escritura de programas complejos con un menor número de líneas de código en comparación con otro tipo de lenguajes.

Además, es un lenguaje de código abierto, lo que quiere decir que es gratuito y su comunidad contribuye constantemente creando nuevas bibliotecas para la aplicación y otros muchos tipos de mejoras, es muy popular también debido a que puede integrarse con una gran capacidad con otros lenguajes y sistemas, por tanto, es ideal para procesos de colaboración.

Ilustración 6. Logo aplicación Python.

- **Trabajar con Python.**

  Para poder trabajar de manera eficaz con Python en entornos industriales, es esencial contar con un adecuado entorno de desarrollo integrado (IDE), una herramienta que facilita la programación combinando varias funciones como escribir, probar y depurar código en un solo lugar. Algunos de estos entornos son PyCharm, Spyder (este es el que empleo para las pruebas), Jupyter Notebook y Visual Studio Code, todos ellos ofrecen herramientas avanzadas para hacer el código de Python más eficiente y no solo facilitar el desarrollo sino proporcionar también capacidades fuertes para la visualización de datos y análisis de rendimiento.
  Además de los IDEs, se puede trabajar también con él en plataformas especializadas en el diseño paramétrico como Grasshopper para Rhino. Esta herramienta permite la integración directa de scripts Python con software de modelado 2D y 3D, facilitando la automatización de tareas complejas y la creación de algoritmos personalizados para diseño y análisis estructural, que es precisamente la base de todo este trabajo que presento.

- **Aplicaciones de Python en el entorno industrial.**

  Python se ha convertido en un recurso indispensable en muchas industrias debido precisamente a esta flexibilidad y capacidad de resolución de problemas, es por eso por lo que estas son algunas de las aplicaciones más comunes en entornos industriales:

  1. Inteligencia Artificial y Machine Learning: usado para el desarrollo de modelos predictivos y reconocimiento de patrones. La biblioteca PyTorch es fundamental para construir y entrenar redes neuronales y algoritmos de aprendizaje autónomo avanzado.
  2. Desarrollo Web: facilita la creación de aplicaciones escalables y seguras.
  3. Procesamiento de Big Data: librerías como Pandas permiten manipular y transformar datos complejos para obtener conclusiones relevantes.
  4. Automatización y control: gran capacidad para interactuar con sistemas y dispositivos.
  5. Optimización y Algoritmos Genéticos: disponibles en la librería DEAP, son los más utilizados para optimizar procesos y diseños.

  No solo ofrece una amplia variedad de aplicaciones industriales, sino que también cuenta con la ventaja de poseer una gran comunidad activa que como mencionaba anteriormente, comparte conocimientos y desarrolla nuevas herramientas para afrontar los desafíos que se van presentando día a día en sectores como la energía, manufactura, salud, etc.

- **Principales librerías usadas en Ingeniería civil y estructural.**

  En Python, las librerías son módulos que extienden la funcionalidad del lenguaje para algunas tareas más específicas, algunas de las más importantes incluyen:

  1. NumPy: fundamental para computación numérica en Python, permite trabajar con operaciones matemáticas avanzadas de manera eficiente.
  2. SciPy: algoritmos y herramientas matemáticas para integración numérica, optimización, álgebra lineal y estadísticas.
  3. Pandas: mencionada anteriormente, facilita la manipulación y análisis de datos estructurados.

4. Matplotlib y Plotly: para la creación de gráficos y visualizaciones de datos en 2D y 3D.
5. **DEAP (Distributed Evolutionary Algorithms in Python):** herramientas para implementar algoritmos genéticos y estrategias de optimización evolutivas, tomará una gran relevancia en el desarrollo final de este proyecyo.
6. Además, usaremos algunas librerías propias para Grasshopper y su tratamiento de árboles y ramificaciones, las veremos implementadas en el desarrollo del proceso del proyecto.

Por otro lado, mencionar la relevancia de los bucles “For” e “If” que constituyen la base de la resolución de cualquier problema sencillo que se presente en los programas de diseño estructural, además, el uso de puertas lógicas a modo de código como por ejemplo, “and”que ayuda a simplificar y resolver muchos problemas.

- **Python en la ingeniería civil y estructural.**

  Este es un campo en continuo, donde son esenciales la innovación y la eficiencia por lo que la programación se ha convertido en una herramienta esencial para los profesionales, ya que es de vital importancia que se adecuen a los cambios tecnológicos.

  Por ejemplo, la programación permite automatizar tareas repetitivas que consumen tiempo, reduciendo errores y liberando tiempo para actividades más complejas. La capacidad de programar también facilita el posterior análisis de grandes conjuntos de datos, permitiendo extraer la información útil y crear visualizaciones que apoyen decisiones más informadas. Además, la programación fomenta un enfoque lógico y sistemático para resolver problemas, ya que cada línea de código que se escribe tiene mucho pensamiento y estudio detrás, y se tiende a generalizarlas para que pueda servir para un número indefinido de casos, permite también optimizar soluciones.

  La sintaxis de Python es accesible para principiantes, permitiendo a todo tipo de ingenieros, sin tener que ser informáticos, poder entender conceptos de programación sin enfrentar una dificultad de aprendizaje. Además, ofrece la posibilidad de colaborar interdisciplinarmente, permitiendo ésta entre profesionales de otros campos y comprender así mejor las diversas aplicaciones de la programación en la ingeniería.

  Python es también compatible con populares softwares de análisis de elementos finitos, como es RFEM, el cual hemos detallado antes, lo que les permite automatizar herramientas dentro de su flujo de trabajo, estrechamente ligado a lo que vengo a mostrar con la ejecución de este proyecto. Esta integración mejora la eficiencia, simplifica procesos y permite abordar simulaciones avanzadas y análisis profundos.

  También puede usarse para desarrollar herramientas que faciliten la gestión de proyectos y automatizar cualquier tipo de proceso de documentación, facilitar la creación de plataformas colaborativas donde puedan intercambiarse datos en tiempo real, entre muchísimas aplicaciones que podemos darle y muchísimas que aún están por descubrirse.

3.2. Algoritmos y Técnicas.

3.2.1. Conexión API.

Las Interfaces de Programación de Aplicaciones, conocidas comúnmente como API, son un concepto que se introduce en nuestras vidas para mejorar los entornos de trabajo en gran medida. Estas son un conjunto de reglas y procedimientos que nos permiten la intercomunicación de softwares, haciendo posible el intercambio y tratamiento de información al tener la función de intermediarias, sin que se deban de dar detalles internos de cada una de las aplicaciones involucradas. Facilitan muchísimo los entornos de colaboración, además, permiten el ahorro de esfuerzo y tiempo al acelerar el desarrollo de softwares que provocan una rápida implementación de nuevas características.

- Tipos:
  - API Privada: mayor control de datos al ser de uso interno.
  - API Pública: fomenta innovación y alcance al público al ser accesible a todos.
  - API de Partners: compartida con socios comerciales específicos.

Este entorno ha supuesto innovaciones tales como la creación o ampliación de canales de ingresos, expansión de alcance y facilita de manera eficiente la innovación mediante la colaboración. Su arquitectura puede ser de dos tipos, Orientada a Servicios (SOA) o a Microservicios, que descompone las aplicaciones en partes más pequeñas y especializadas.

En el entorno de la arquitectura y el diseño, es fundamental la implementación de estas API para facilitar y agilizar muchos flujos de trabajo, concretamente podemos hablar de la aplicación Grasshopper. Gracias a su capacidad para integrar las API, se ha convertido en una herramienta fundamental para la automatización y optimización de trabajos, es por ejemplo el caso de su conexión con Python.

Python es un lenguaje de programación que se integra perfectamente con Grasshopper, y como sabemos, Python es hoy en día una de las principales fuentes en la programación y es usado internacionalmente, lo que le permite a Grasshopper, además de un alcance mucho mayor, aplicaciones tales como:

- Automatización de procesos iterativos.
- Personalización de funcionalidades.

Por otro lado, podemos hablar también de la conexión con RFEM, software de análisis estructural que gracias a su conexión API con Grasshopper, le ofrece múltiples beneficios como:

- Sincronización rápida, sencilla y eficaz de datos entre estas dos aplicaciones.
- Automatizar el proceso de cálculo estructural.
- Optimizar diseños en tiempo real.

Gracias a todo esto, Grasshopper está ganando relevancia en el ámbito del diseño e ingeniería moderna, debido a sus amplias y eficaces capacidades de integración. Esto, permite a sus usuarios aprovechar una amplia gama de herramientas disponibles para descargar e instalar en el entorno gráfico de la aplicación, posibilitan múltiples usos y adaptaciones a muchos sectores industriales, y al ser de código abierto, estar en constante desarrollo.

3.2.2. Algoritmos genéticos.

Los algoritmos genéticos (GA) son modelos de optimización y búsqueda inspirados en los mecanismos de la teoría de la evolución de Charles Darwin y los principios de selección natural. Fueron desarrollados por primera vez por John Holland, profesor de la Universidad de Michigan, en la década de 1960 y 1970, y desde entonces estos algoritmos han sido ampliamente utilizados en diversos campos debido a su capacidad para encontrar soluciones optimas a problemas complejos mediante la simulación de procesos evolutivos.

Las soluciones candidatas se presentan como individuos en una población y se someten a operaciones de selección, cruce y mutación para mejorar sus genes con el tiempo. El proceso iterativo continua hasta que se encuentra una solución satisfactoria o se cumplen otros criterios de parada.

- **Principales elementos y términos para el desarrollo de un Algoritmo Genético.**
  - Individuo: son cada uno de los candidatos para la función de búsqueda.
  - Genotipo/Genoma/Cromosoma: es la estructura de datos que representa a un individuo, es decir, como las características que lo representan.
  - Gen: posición de una característica individual dentro del genoma que las engloba.
  - Alelo: es el valor específico que tiene un gen.
  - Fenotipo/Fitness: es la función que evalúa la calidad del individuo en relación con el problema, con esta función determinaremos como de bueno es un individuo y nos basaremos en ella para determinar su continuación o descarte.

- **Etapas del proceso genético.**

  1. **Inicialización.**

     Se crea una población inicial de individuos, y cada uno de ellos representa una solución potencial al problema y se codifica típicamente en una cadena de bits, enteros, números reales u otras estructuras, todo eso dependerá del tipo de problema que tengamos. La población inicial puede generarse de manera aleatoria o utilizando conocimientos previos y definiciones que puedan orientar la búsqueda desde el inicio. Es decir, en la inicialización se crea una población inicial de individuos con alelos diferentes.

  2. **Selección.**

     Mediante este proceso elegiremos a los individuos de una población para que contribuyan a la creación de la próxima generación. Los individuos con las mejores aptitudes tienen una posibilidad mayor de ser los seleccionados, para ello hay varias técnicas, algunas de las cuales se describen a continuación:

     - Torneo: un grupo de individuos es seleccionado aleatoriamente de la población, y el mejor de este grupo es elegido como padre, se repetiría una y otra vez el proceso hasta obtener todos los padres necesarios para poder proceder a otra fase posterior.

     - Ruleta: en este método la probabilidad de crianza va en función a lo bueno que sea el valor del fitness del individuo, se le asigna un porcentaje de ser elegido en un proceso de selección aleatoria. Pese a ser más probable salir elegidos aquellos con mayor fitness, este método simula una ruleta donde cada segmento es proporcional a la aptitud de cada individuo.

- Ranking: se ordenan los individuos en base a su fitness y se establece un rango de validez, todos aquellos cuyo fitness entre dentro de ese rango serán los seleccionados, esto evita que individuos cuyo fitness sea extremadamente alto dominen la selección.

- Estado estacionario: se elimina a los individuos con peor fitness y se les sustituye por las crianzas de aquellos que tienen los mejores fenotipos.

3. **Cruce (Crossover).**

Es el proceso mediante el cual se combinan genes de dos padres para producir descendencia. Se coge el genoma completo y se establece un criterio de separación de genes, hecho esto se intercambian las partes separadas entre dos individuos, generando nuevos hijos con las características combinadas de ambos padres. A continuación, los métodos más comunes de separación de genes en el genoma:

- Cruce de un punto: se parte en dos los alelos de los padres, y los hijos se llevan un trozo de los alelos de cada padre.

- Cruce de dos puntos: se seleccionan dos puntos y los segmentos entre estos puntos se intercambian entre los padres para formar los descendientes.

- Cruce uniforme: cada valor se elegirá aleatoriamente de los progenitores, creando un descendiente como una selección aleatoria de los valores de los padres.

- Cruce de n puntos: una generalización del cruce de dos puntos, donde se procede igual pero el número n es variable dependiendo del caso.

4. **Mutación.**

Es un proceso que introduce cambios aleatorios en los genes de un individuo, es esencial para mantener la diversidad genética dentro de la población y evitar el estancamiento en algunos puntos. Los tipos más comunes de mutación son:

- Mutación puntual: se cambia un solo gen del individuo, seleccionando aleatoriamente una posición y asignándole un nuevo valor.

- Mutación por intercambio: dos genes en el individuo se intercambiarán entre ellos.

- Mutación de inversión: se selecciona un segmento de genes y se invierte su orden.

- Mutación de desplazamiento: un segmento de genes se mueve a una nueva posición dentro del individuo.

Hay un parámetro crítico dentro de este proceso, se trata de la Tasa de mutación, hemos de mantenerla en un valor intermedio ya que al ser muy alta podría convertirse el algoritmo en una búsqueda aleatoria y si fuese baja, no habría mucha diversidad en los nuevos individuos.

5. **Elitismo.**
   Es una técnica utilizada para asegurar que los mejores individuos de una población pasen a la siguiente generación sin cambios, gracias a este método garantizamos el mantenimiento de la calidad de la población con el paso del tiempo. Hay varios enfoques:

   - Elitismo simple: se selecciona el mejor individuo de la población actual y se copia directamente para la próxima generación.

   - Elitismo porcentual: un cierto porcentaje de los mejores individuos se transfiere directamente a la siguiente generación.

   - Elitismo adaptativo: la cantidad de individuos transferidos dependerá de la variación que haya entre los fitness de la población.

     Este método ayuda a mantener la calidad de las soluciones encontradas y acelera la convergencia del método hasta el resultado óptimo.

- **Aplicaciones de los Algoritmos Genéticos.**

  - Optimización de funciones: resolver problemas matemáticos donde se prioriza la búsqueda del valor máximo o mínimo de una función.
  - Economía y finanzas: en predicción de mercados y carteras de inversión.
  - Diseño de circuitos: optimizar su diseño para minimizar el uso de recursos y maximizar el rendimiento.
  - Inteligencia artificial: entrenamiento de redes neuronales y otros modelos de aprendizaje automático.

- **Vinculación del AG con Optimización Topológica Estructural.**

  El primer paso para vincular AG con optimización topológica sería definir un modelo estructural usando herramientas de elementos finitos, como por ejemplo Grasshopper.
  Para automatizar la evaluación de las configuraciones que se generan con el algoritmo genético se establece una interfaz entre ello y el modelo estructural, permitiendo la evaluación automática de configuraciones generadas por el AG y la retroalimentación a éste.

  Se define una función objetivo para guiar la optimización y el AG trabajará de forma iterativa para ir consiguiendo progresivamente y los resultamos los podemos visualizar mediante herramientas en Python o Grasshopper, por ejemplo.
  Un ejemplo de implementación de esta sería combinando la potencia de los bloques de Grasshopper para desarrollar este algoritmo junto con el bloque de código de Python que se puede usar también en este software, haciendo uso de ambos se podría generar un AG de manera visual y comprensible, es el caso de este proyecto.

3.3. Procedimientos Experimentales.

3.3.1. Caso 1: Viga simple

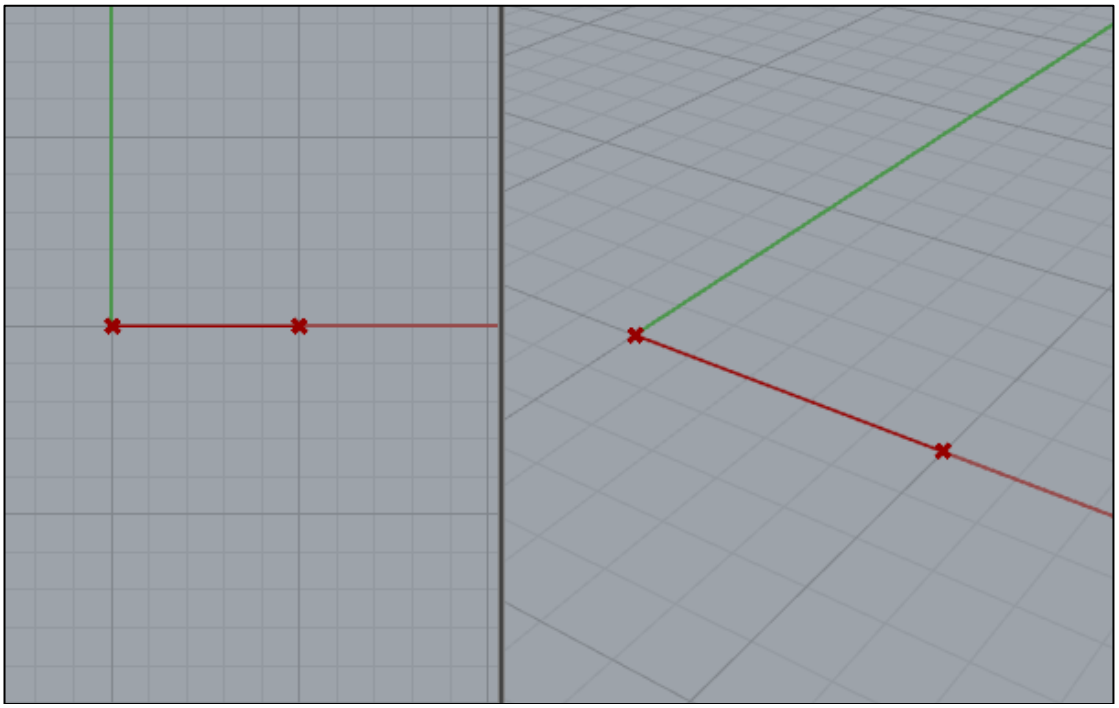


Ilustración 7. Viga Simple en Rhino

1. <u>Planteamiento del Problema</u>

El objetivo es desarrollar un modelo en Grasshopper para analizar la deformación de una viga simple en dos dimensiones sometida a una carga distribuida. La barra tendrá dos puntos, uno en el origen de coordenadas y otro con una longitud predefinida. Se aplicará una carga distribuida a lo largo de ella, y se estudiará la deformación resultante. Para lograr este análisis, se iterará entre diferentes perfiles IPE de la normativa hasta encontrar uno que cumpla con el límite de deformación nodal establecido.

- **Detalles del Planteamiento:**

  - **Geometría de la Barra:** La barra se representará como una línea recta en 2 dimensiones, con un punto en el origen de coordenadas y otro punto con una longitud definida.
  - **Carga Distribuida:** Se aplicará una carga distribuida a lo largo de la barra para simular las fuerzas externas que actúan sobre ella.
  - **Iteración de Perfiles IPE:** Se implementará un bucle para iterar entre los diferentes perfiles IPE especificados en la normativa. Se evaluará la deformación resultante para cada perfil IPE hasta encontrar uno que cumpla con el límite de deformación nodal establecido.
  - **Análisis de Deformación:** Se calculará la deformación en la barra para cada perfil IPE, considerando las propiedades de cada perfil y la distribución de la carga aplicada.
  - **Criterio de Convergencia:** Se establecerá un criterio de convergencia basado en el límite de deformación nodal permitido. Una vez que se encuentre un perfil IPE que cumpla con este criterio, el proceso de iteración se detendrá.

- **Consideraciones Adicionales:**

  - Se utilizarán los componentes disponibles en Grasshopper para modelar la geometría de la barra, aplicar la carga distribuida y realizar el análisis de deformación.
  - Se emplearán plugins como Anemone para implementar el bucle de iteración entre los diferentes perfiles IPE.
  - Se utilizarán los estándares y especificaciones establecidos en la normativa correspondiente para la selección de los perfiles IPE y la evaluación de la deformación nodal.

2. Construcción del Modelo Inicial

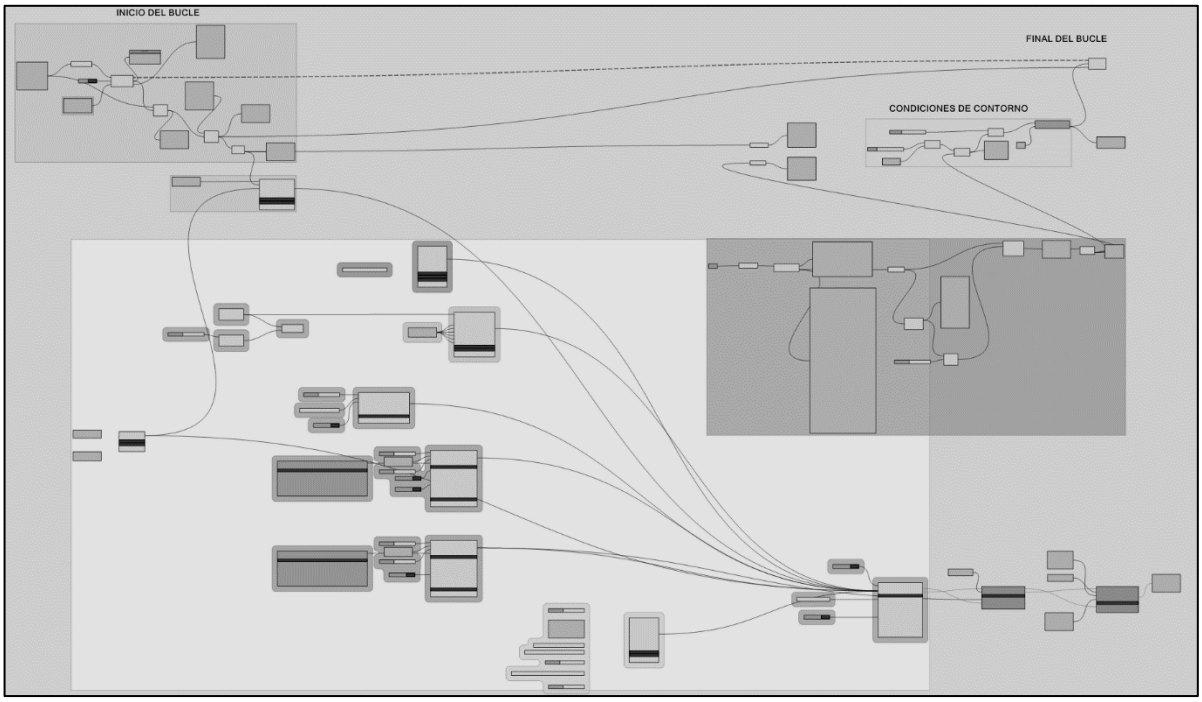


Ilustración 8. Vista general del modelo completo.

2.1. Creación de la Geometría

Al iniciar una nueva hoja de Grasshopper, el primer paso es añadir dos componentes llamados "Construct Point" desde la sección "Vector, Point". Este componente contiene tres inputs para las coordenadas x,y,z, para establecer la ubicación del punto, y un único output que será el punto generado. Estos marcarán el inicio y el final de la barra que queremos establecer. Al segundo punto creado, le añadiremos el componente "Slider", que nos permitirá crear una regla con números en un rango establecido, representando la longitud de la barra, establecida en 5 unidades para este caso pero permitiendo a esta longitud variar de manera rápida y sencilla. Posteriormente, uniremos estos dos puntos en una línea utilizando el componente "Line", donde solo establecemos el punto inicial y final para obtener como output la línea.

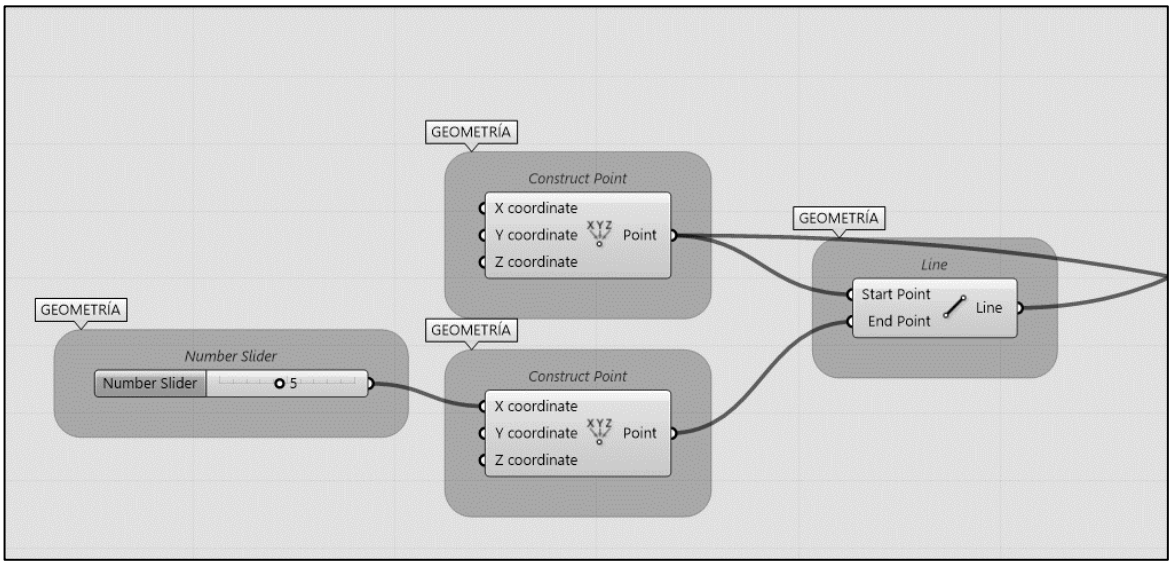


Ilustración 9. Geometría básica.

2.2. Creación del Miembro

Para convertir nuestra línea en una viga, utilizamos el componente "Member".

- **Instalación del plug-in "Dlubal"**: Es necesario agregar este plug-in para continuar el programa, ya que contendrá alto contenido estructural necesario tanto para miembros como para cargas y apoyos nodales, y para la posterior exportación a la aplicación RFEM Dlubal. Para instalarlo, accedemos a la página oficial "Food4Rhino", buscamos "Dlubal", seleccionamos el módulo y procedemos con la instalación.

Dentro del componente "Member", configuramos una serie de parámetros y vinculamos el output de la línea previamente creada al input "GH Line" de este componente. Luego, añadimos un "Value List" vinculado al input "Member Type", que nos ofrece opciones donde seleccionaremos "Type_Beam" para convertirla en viga. Además, para el input "section start", definiremos la sección y material de esta viga.

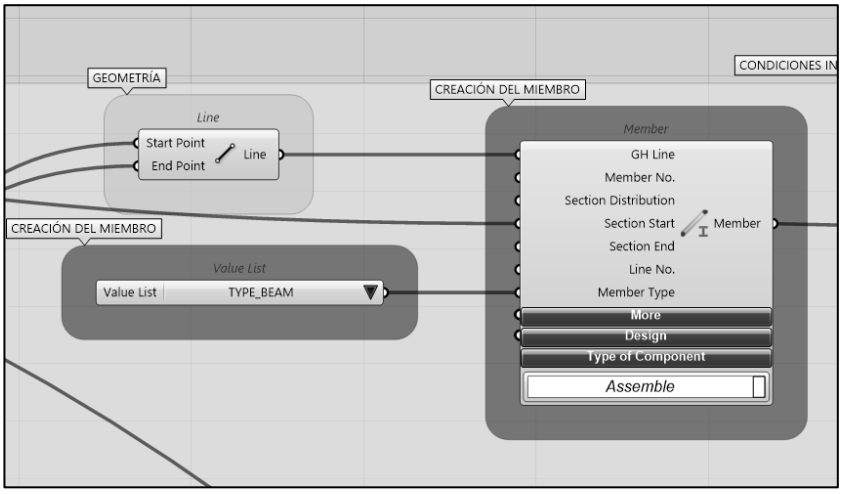


Ilustración 10. Creación del miembro.

2.2.1. Sección Inicial

Aunque iteraremos la sección de la viga posteriormente, es crucial establecer una sección inicial para elaborar el modelo y luego realizar modificaciones. Para ello:

- Agregamos el componente "Section" del módulo de Dlubal.
- Conectamos un panel al input "section name" del componente "Section", definiendo un perfil inicial, por ejemplo, "IPE 400".
- Utilizamos otro panel para establecer el número de sección, denominándolo "1".

No insertaré fotografía puesto que esto se verá modificado alcanzado otro punto del modelo.

2.2.2. Material

Incorporamos un material al módulo de sección siguiendo un procedimiento similar:

- Agregamos el componente "Material" de Dlubal.
- Conectamos su salida al input "Material Nº" del componente "Section".
- Utilizamos dos paneles para definir el nombre del material (por ejemplo, "s235") y el número de material (que será "1"), vinculándolos a sus respectivos inputs.

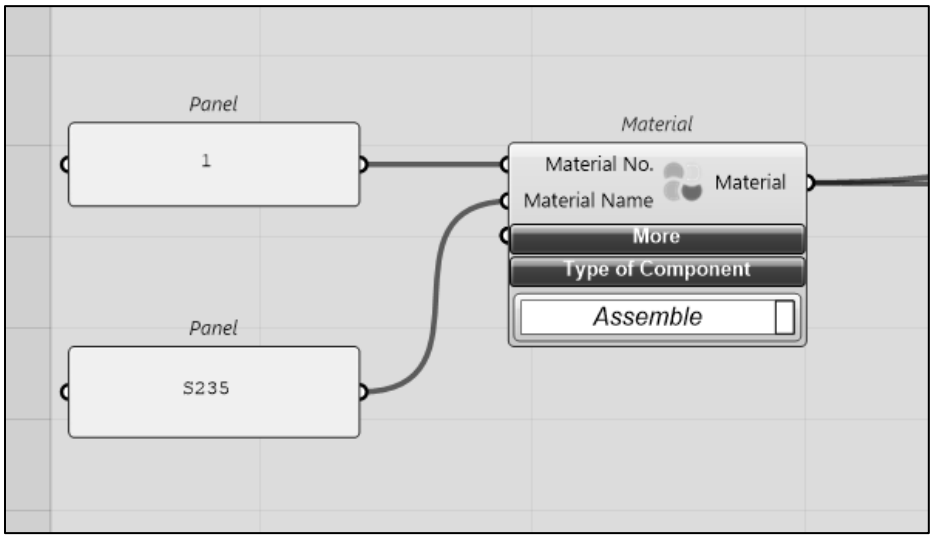


Ilustración 11. Definición del material.

2.3. Creación de los Apoyos Nodales

Los apoyos nodales son cruciales para la estabilidad de la estructura, estos pueden ser establecidos en este caso tanto en el punto inicial como en el final de la estructura, incluso en ambos. Para configurarlos en Grasshopper, primero agregamos el componente "Nodal Support" de Dlubal. Luego, definimos sus inputs:

- **GH node**: Seleccionamos el nodo de Grasshopper donde queremos ubicar el apoyo. En este caso, utilizamos la salida del punto que marca la posición en el origen de coordenadas. {0,0}

- **Restricciones de movimiento**: Conectamos un panel con el término "INF", restringiendo los movimientos de translación y rotación en las coordenadas x y z. Esto se hace para generar un apoyo fijo, ya que vinculamos este panel a todos los inputs de movimiento del componente.

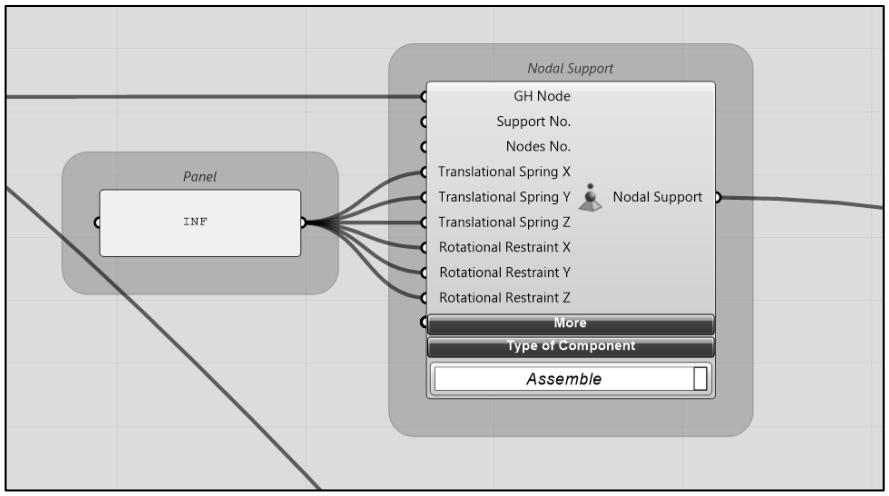


Ilustración 12. Apoyos nodales.

2.4. Configuración de Análisis Estático

La configuración del análisis estático es esencial para evaluar el comportamiento de la estructura bajo cargas. Para ello, añadimos el componente "Static Analysis Settings" de Dlubal. Luego, definimos los siguientes parámetros:

- **Tipo de análisis**: Utilizamos una "Value List" para seleccionar el tipo de análisis. En este caso, elegimos "Geométricamente Lineal" para una evaluación rápida y precisa debido a deformaciones pequeñas y cargas moderadas.
- **Número de análisis estático**: Usamos un slider para marcar el número de análisis estáticos. Establecemos la recta en el número 1 para este caso.
- **Manejo de excepciones**: Utilizamos un botón booleano ("Boolean Toggle") en modo "False" para visualizar los errores directamente en el panel de mensajes de error de Grasshopper, lo que facilita su diagnóstico.

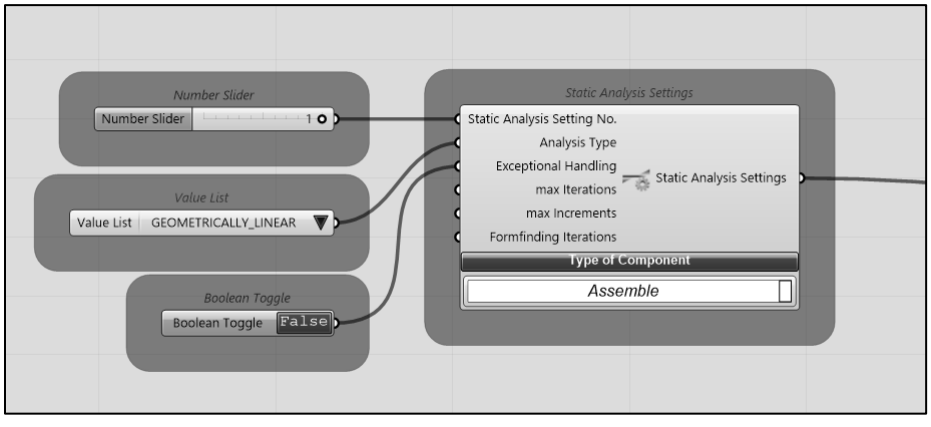


Ilustración 13. Análisis estático.

2.5. Casos de Carga

Para representar dos casos de carga distintos, uno correspondiente al peso propio de la viga y otro a una carga impuesta, seguimos los siguientes pasos:

- Agregamos dos veces el componente "Imposed Load" de Dlubal.
- Para cada caso de carga, añadimos los siguientes componentes:
  - Un slider para determinar el número del caso de carga (1 y 2 respectivamente), conectados al input "Load case nº".
  - Un panel donde definimos el nombre de cada caso de carga ("Self weight" y "Imposed load"), conectados al input "Load case name" respectivamente.
  - Una slider que indica el tipo de ajuste de análisis estático a utilizar (en este caso, ambos sliders establecen el número "1"), conectados al input "Static Analysis Settings" de cada módulo.
  - Un "boolean toggle" con el modo "true" activado, conectado al input "to solve", indicando que queremos tener en cuenta este caso de carga en el cálculo.
  - Un "boolean toggle" extra para el primer caso de carga, conectado al input "Self-Weight active", para activar el peso propio.

Además, añadimos dos componentes "Load Case Classification component" para asociar categorías de carga en el componente "load case", asegurando un análisis estructural preciso.

Para el primer caso de carga:

- Seleccionamos el "Standard group" como "EN 1990", la normativa y estándar europeo.
- La categoría de acción se establece como "Permanent | G", refiriéndose a las cargas constantes sobre la estructura.
- Se elige "ULS (EQU) – Permanent and transient" para el "Design Situation type", abarcando cargas constantes y variables.

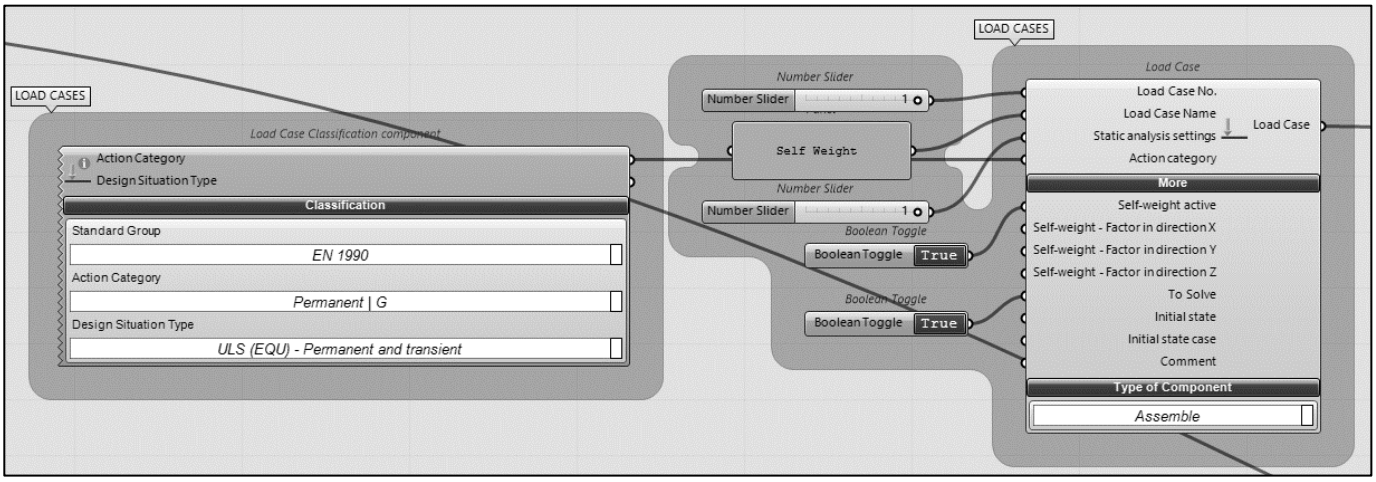


Ilustración 14. Casos de carga (1) Peso propio.

Para el segundo caso de carga:

- Mantenemos el "Standard group" como "EN 1990".
- La categoría de acción se cambia a "Imposed Loads – Category A: domestic, residential áreas | Q/A".
- Se selecciona "ULS (STR/GEO) – Permanent and transient – Eq. 6.10" para el "Design Situation Type", incluyendo estabilidad geotérmica bajo condiciones extremas.

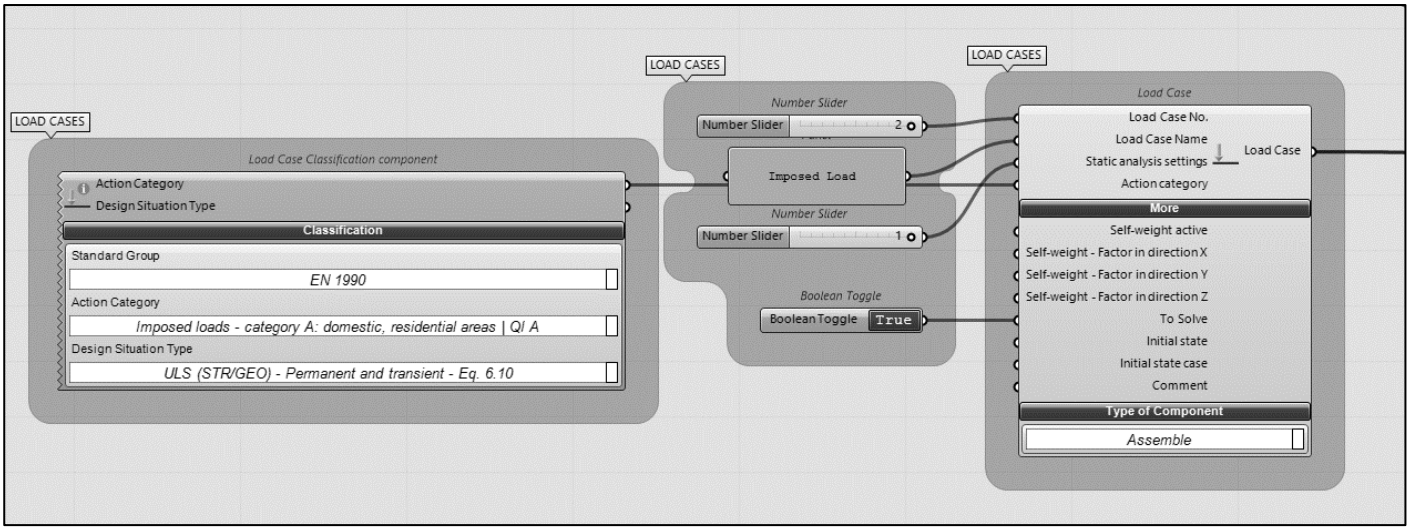


Ilustración 15. Casos de carga (2) Carga impuesta.

2.6. Cargas en Miembros

A continuación, le daremos valor y estableceremos todas las características que de este caso de carga impuesto que definimos previamente, para ello, usamos el componente "Member Load" que quiere decir, carga en un miembro, también forma parte del paquete de Dlubal.
Este módulo se compone de distintos inputs que hemos de especificar individualmente en detalle:

- Member Load Nº: identificaría una carga especifica dentro de un caso de carga, no lo necesitamos definir para este caso.
- Load case Nº: hemos de usar una slider que marque los distintos casos de carga de los que contamos y con la que podamos elegir aquel al que nos referimos, en este caso, la slider marcará "2" que el número asignado previamente al caso de carga impuesta.
- Members Nº: en nuestro caso tenemos un único miembro, por lo tanto nos bastará con poner un panel que indique un número 1.
- Load type: añadiremos una "Value list" que en su desplegable nos mostrará múltiples opciones entre las cuales hemos de seleccionar "Load_type_force" ya que con esta opción el software interpretará esta carga como una fuerza que puede causar deformación y tensión en la viga. Es decir, seleccionamos este para especificar que la carga aplicada es una fuerza.
- Load distribution: seguimos el mismo procedimiento que en el anterior, colocando una "Value List" y seleccionamos "Load_distribution_uniform" del desplegable porque vamos a representar una carga que se aplica de manera constante a lo largo de un miembro estructural, una viga en nuestro caso.

- Coordinate system: simplemente establecemos una slider que marque el número uno y nos limitaremos a hacer esto mismo donde se precise sistema de coordenadas en todo el proceso para así hacer que todo esté en uno único y facilitar los procesos de cálculo.
- Load direction: nuevamente como antes añadimos “Value List” donde seleccionamos “Load_direction_global_z_or_user_defined_w_true”, usaremos este para aplicar cargas en la dirección global Z, que representa la dirección vertical común para cargas gravitacionales, o en una dirección definida por nosotros en caso de ser necesario.
- Load magnitude p: finalmente definimos el valor de la carga, lo haremos con el componente slider, ubicando en él un rango de valores aceptables para lo que queremos representar, en mi caso, llegará hasta los 5 KN/m, que en la slider se verán representados como 5000 N/m.

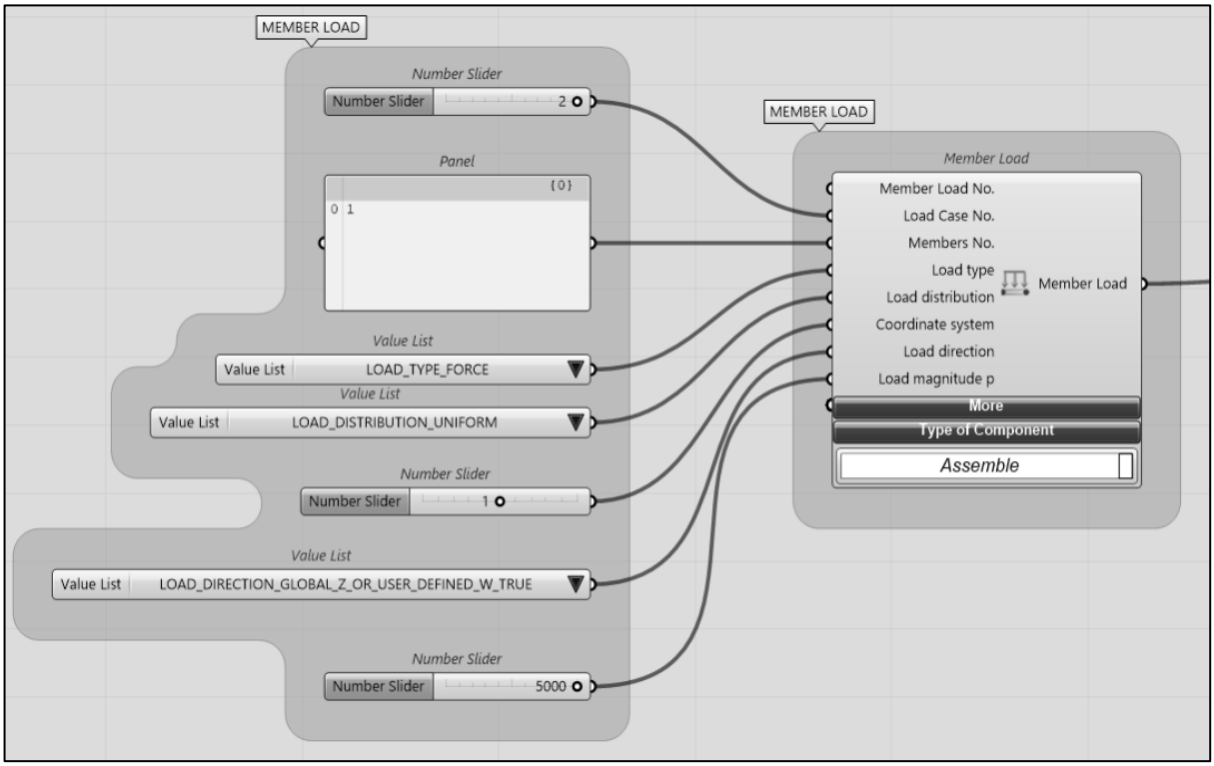


Ilustración 16. Distribución de cargas en el miembro.

2.7. Exportación a RFEM 6

Para realizar una primera exportación a la aplicación RFEM de Dlubal añadiremos en Grasshopper el módulo RFEM 6 Export Component, a través de él haremos todas las conexiones. Detallare a continuación la gestión de los inputs al componente:

- Run: únicamente conectaremos un “Boolean Toggle” que cambiaremos de False a True cuando queramos poner a calcular nuestro modelo completo, exportando a RFEM y regresando a GH de vuelta con los cálculos realizados.

- Input: es el más importante ya que en él hemos de realizar todas las conexiones que queremos importar al programa de cálculo estructural, en este caso hemos realizado 8 conexiones que voy a detallar en una lista, estará compuesta de elementos mencionados en todo el proceso descrito.

Los outputs del modelo que hemos de vincular a este input son:

- Sección
- Material
- Miembros
- Apoyos nodales
- Ajustes de análisis estático
- Dos casos de carga creados, self weight e imposed load.
- Carga en el miembro.

Acto seguido abriremos la pestaña "Additional" del componente para terminar de concretar algunos aspectos necesarios para la correcta exportación del modelo:

- Units: añadimos una "Value list" que usaremos para darle nombre y valor a las unidades usadas en el modelo, de esta lista seleccionaremos metros.

- Overwrite: es muy importante no olvidarnos de esto, ya que es el encargado de controlar si se debe sobrescribir un modelo existente en RFEM durante la exportación y permite, por tanto, la actualización o preservación del modelo según lo consideremos necesario. Para ello usaremos un "Boolean Toggle" que estableceremos en True en este caso ya que no queremos nuevos modelos tras cada iteración del bucle que haremos posteriormente.

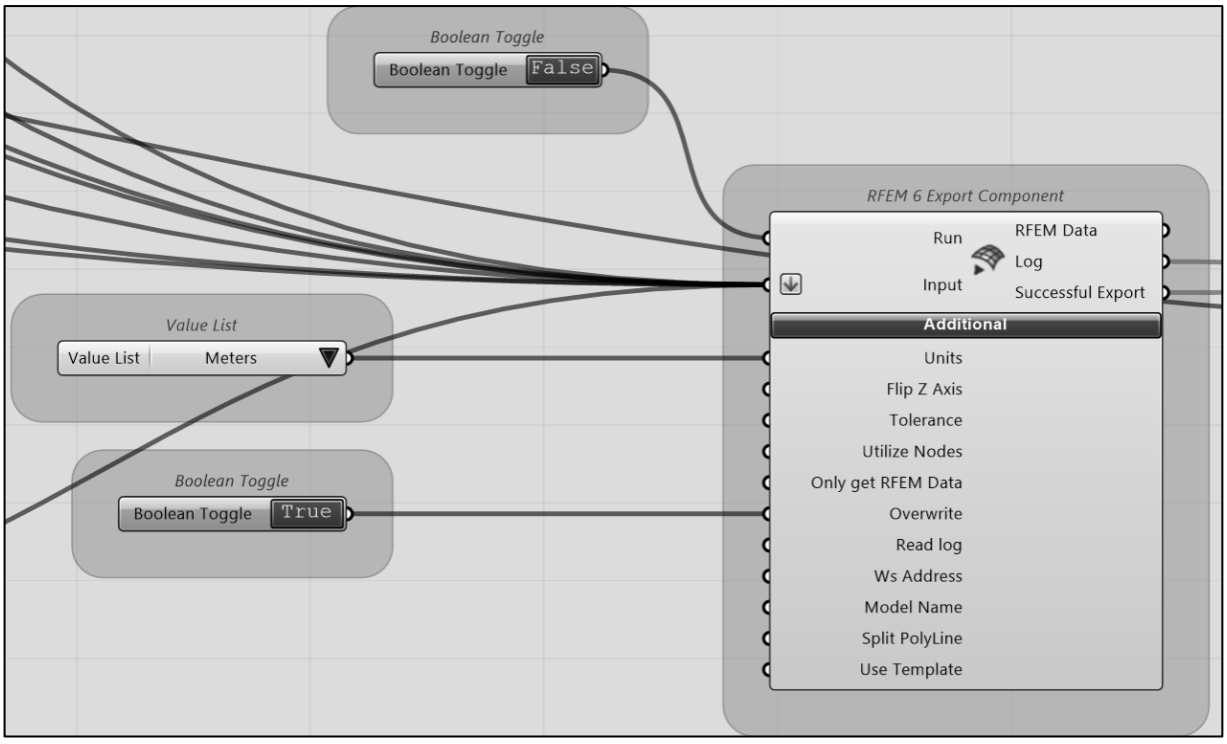


Ilustración 17. Exportación a RFEM.

3. Exportación de Resultados de Grasshopper/RFEM a CSV en Excel.

El objetivo es extraer un archivo CSV del modelo generado en RFEM, el cual es un archivo de texto plano que almacena datos tabulares separados por comas. Se busca extraer un dato específico, como se describe en el siguiente apartado. Por el momento, se pretende extraer los archivos con los datos tras el cálculo del caso de carga impuesta, entre los cuales se incluyen varios archivos que luego serán filtrados según el objetivo.

Para llevar a cabo esta tarea, se siguen los siguientes pasos:

- Se añade el componente "RFEM 6 Calculate Component" para extraer los cálculos del modelo. Sus inputs incluyen:
  - "Run": Conectado al output "Successful Export" del módulo anterior "RFEM 6 Export Component", este input da la orden de iniciar el cálculo una vez que la exportación ha sido completada con éxito.
  - "Model name": Se agrega un panel con el nombre dado al archivo del modelo en RFEM y Grasshopper, el cual debe ser el mismo para ambos.
  - "Cases to calculate": En la pestaña "Additional", se vinculan los casos de carga de los cuales se desean extraer los resultados. En este caso, se vincula el caso de carga 2, "Imposed Load".
  - "Read Log": Se conecta al output "Log" del módulo "RFEM 6 Export Component" para transmitir toda la información relevante.

Posteriormente, se añade el componente "Export Results to Files", que presenta una variedad de inputs que requieren vinculaciones específicas:

- "Run": Conectado a "Successful Calculation", similar al módulo anterior.
- "Model name": Se utiliza un panel para especificar el nombre del modelo, el cual debe coincidir con el utilizado anteriormente. Por ejemplo, "SIMPLE BEAM 2".
- "Path to Export Folder": En este input, se coloca la ubicación deseada para guardar los archivos, especificando la ruta desde la carpeta del ordenador mediante un panel.
- "Type of Export": Se utiliza un panel para especificar "CSV", ya que es el formato deseado para esta exportación.
- "Read Log": Se conecta al output "Log" del módulo mencionado anteriormente para obtener información relevante durante el proceso de exportación.

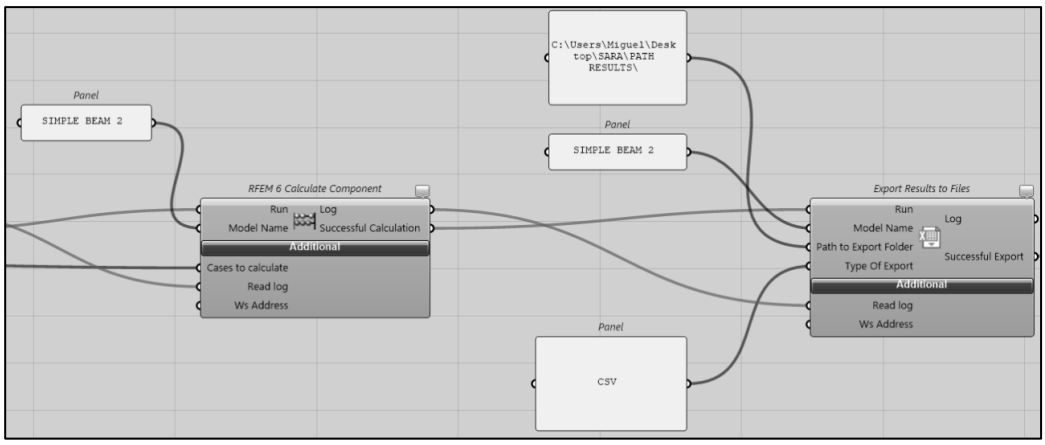


Ilustración 18. Cálculo y exportación a CSV.

4. Lectura de Datos del CSV en Grasshopper.

Se explicará en detalle cómo extraer un dato de uno de los archivos CSV, centrándonos en los análisis estáticos de nudos y las deformaciones asociadas. Específicamente, nos interesará la componente U_Z, que representa el desplazamiento o deformación en la dirección vertical (Z), indicando cuánto se ha movido un punto específico de un elemento estructural bajo las cargas aplicadas.

Para llevar a cabo este proceso, se siguen los siguientes pasos:

1. Se inserta el componente "File Path", donde al hacer clic derecho se selecciona "Select new file location" para especificar la ubicación del archivo del cual se desea extraer el dato. Esto permite que con cada nuevo modelo de cálculo se extraiga automáticamente del archivo correspondiente.

2. Se utiliza el módulo "Read File" para leer el contenido bruto del archivo, antes de pasarlo al componente "Read CSV" para procesar el texto limpio y extraer datos tabulares. De este módulo:
    - "CSV String" se conecta a "Content" del módulo "Read File".
    - "Headers and Values" se conecta a paneles individuales para visualizar sus valores. Además, se utiliza un "Graft Tree" para organizar los datos de manera más estructurada, permitiendo tratar cada elemento por separado.

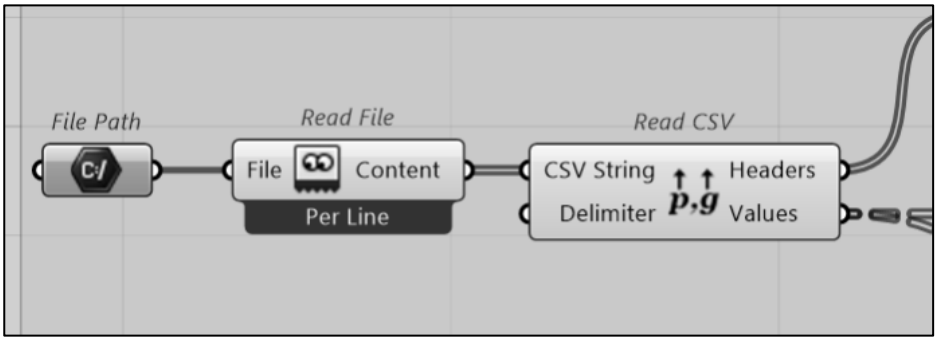


Ilustración 19. Extracción deformación Uz (I)

3. Se añade el componente "Graft Tree" para crear una nueva rama en el árbol de datos, dividiendo la lista en componentes individuales para facilitar operaciones posteriores. Sin embargo, para simplificar la selección del dato de interés, se utiliza el componente "Tree Statistics" para convertir la lista en una lista de índices de posición de los elementos.

4. Se emplea el componente "List Item" para buscar y seleccionar el dato útil de la lista inicial, en este caso, la deformación U_Z. Se utiliza una slider como input "index" para elegir la posición del elemento buscado.

5. Finalmente, se utiliza el componente "Tree Item" para convertir el dato en un elemento de un árbol único. Opcionalmente, se puede utilizar un módulo de Python para nombrar este valor antes de utilizarlo en la definición de las condiciones de contorno.

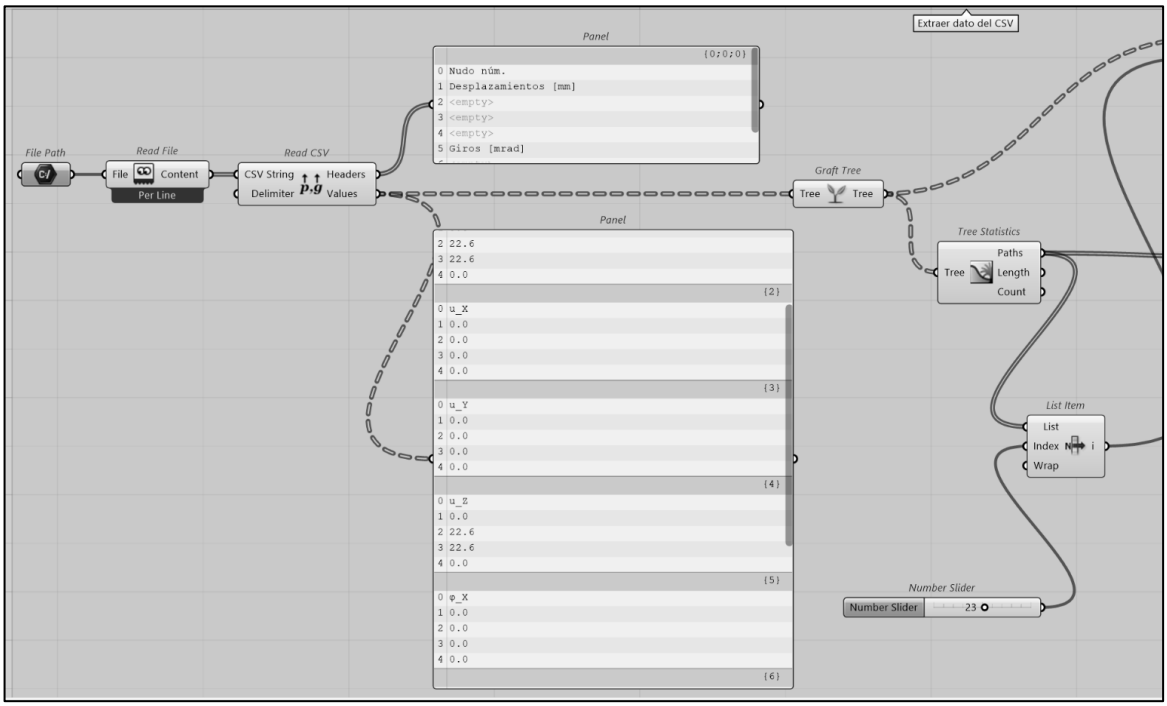


Ilustración 20. Extracción deformación Uz (II)

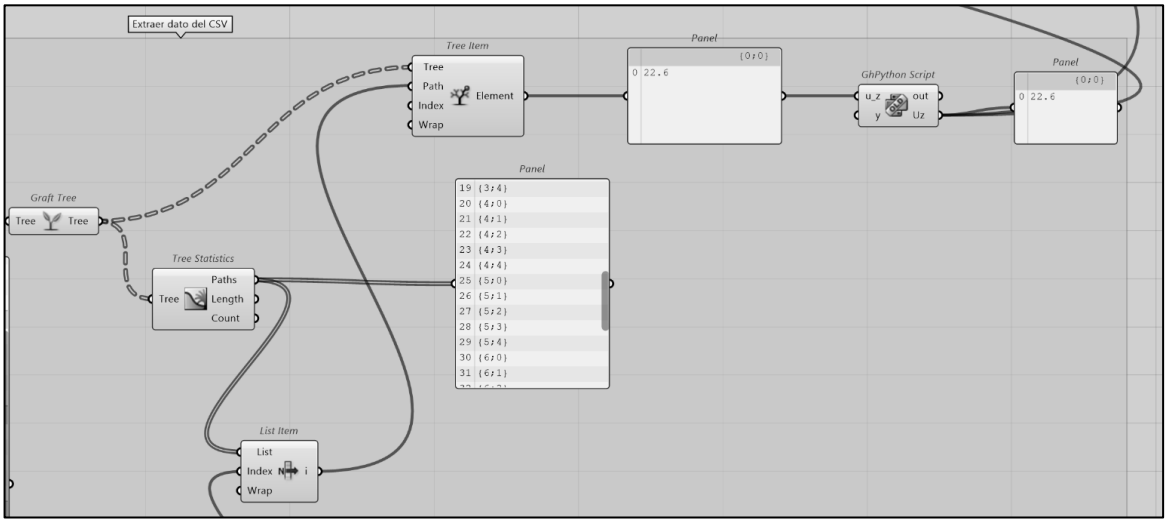


Ilustración 21. Extracción deformación Uz (III)

5. Bucle con Anemone.

Una vez establecidos todos los aspectos relacionados con la creación de la estructura y la definición de sus casos de carga, para finalmente lograr exportar, calcular y extraer un dato de la estructura, implementaremos el objetivo principal de este modelo: probar un modelo de optimización en Grasshopper que genere y calcule un modelo variando su perfil IPE y evaluando la deformación que se produce con cada uno de estos perfiles hasta encontrar el primero que cumpla con la normativa, establecida dentro de unas condiciones de contorno que detallaremos a continuación.

Para lograr esto, utilizaremos el plug-in de Rhino llamado Anemone, que se puede descargar directamente desde la web "Food4Rhino". Este plug-in nos permitirá la creación de bucles con tan solo dos sencillos componentes: LOOP START y LOOP END.

5.1. Componente LOOP START: Variación Automática del Perfil IPE

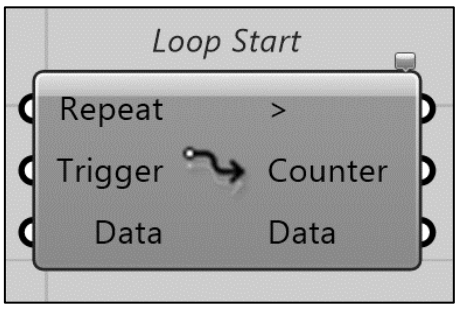


Ilustración 22. Componente Loop Start.

Este componente cuenta con numerosos inputs y outputs que son fundamentales de configurar para hacer funcionar esta optimización. Comenzaremos configurando lo básico conectando ">" al LOOP END.

- **Trigger:** Utilizaremos un simple "Botón" con el que podremos reiniciar el bucle o indicarle que comience de nuevo.
- **Data:** Tanto el input como el output estarán vinculados a paneles. En el input, establecemos un valor inicial del perfil IPE por el cual queremos comenzar a iterar.
- **Repeat:** Configuramos el número de iteraciones que deseamos que haga nuestro bucle. En este caso, para una mayor flexibilidad, usaremos el componente "Counter" como parámetro.
    - Para ello, creamos un panel con una lista separada por saltos de línea, donde cada número representa una iteración. Luego, utilizamos el componente "List Length" para obtener la longitud de la lista, que será el número total de iteraciones.
    - Conectamos el output de "List Length" al input "Repeat".

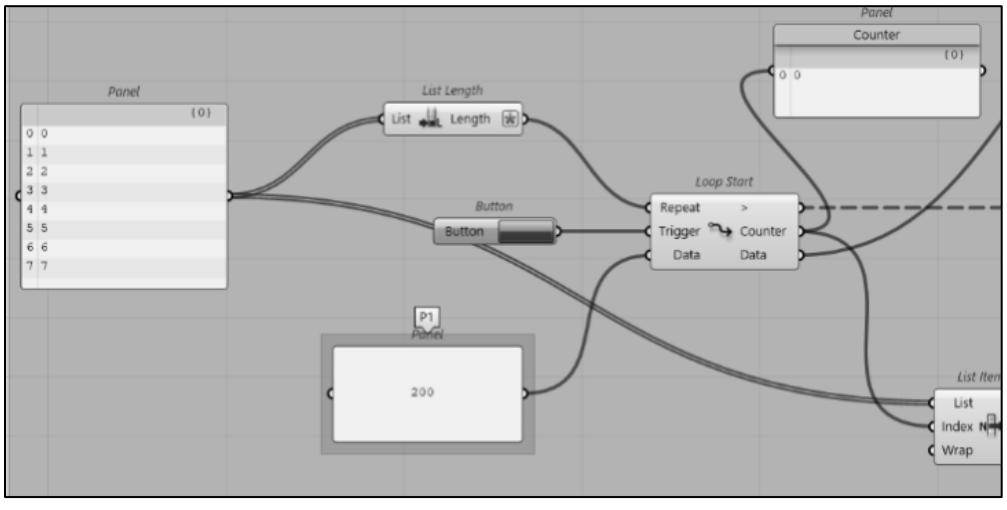


Ilustración 23. Inicio del bucle (I).

- **Counter:** Utilizaremos un "List Item" donde la lista será la creada anteriormente para "Repeat" y el output "Counter" se conectará al input "Index". Esto nos proporcionará un valor "i" que variará con cada iteración y será crucial para utilizarlo como variable en el proceso de selección del perfil IPE.

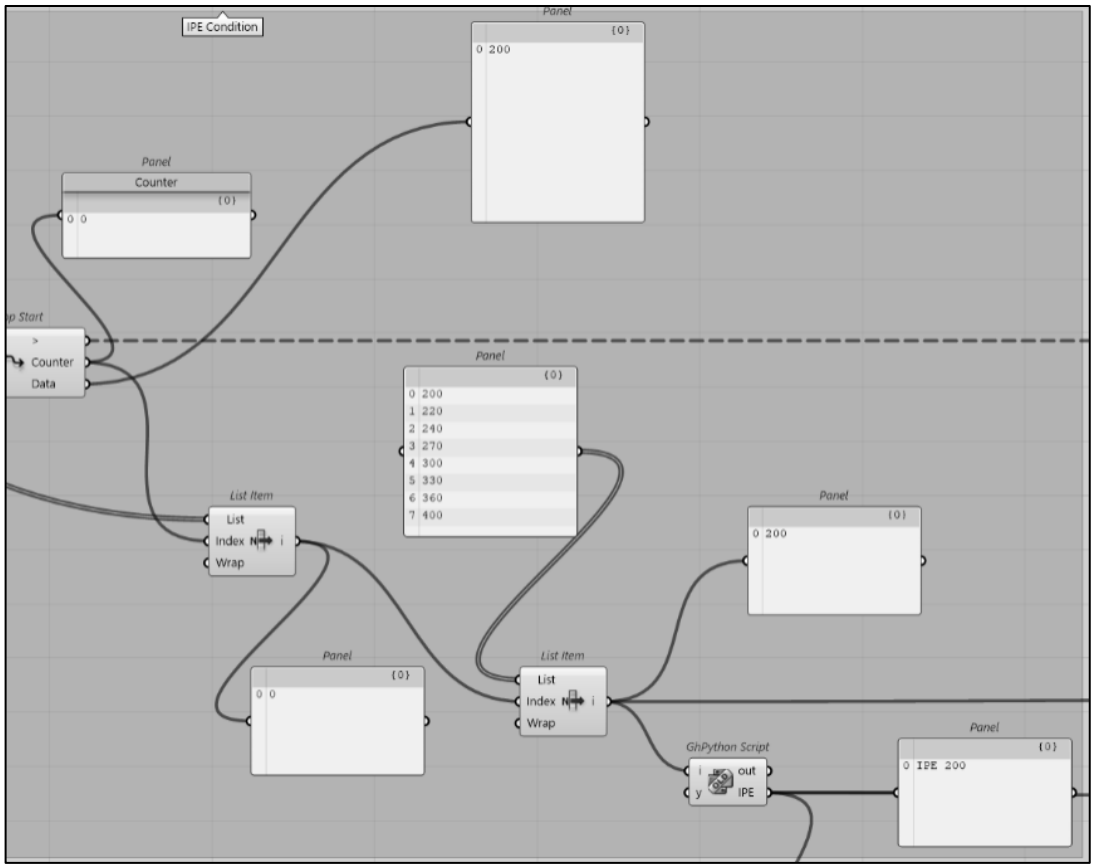


Ilustración 24. Inicio del bucle (II).

  - Para obtener la iteración del perfil IPE, agregamos otro "List Item", donde la lista será un panel con los números de los perfiles (200, 220, 240, 270, 300, 330, 360, 400) y el nuevo "Index" será la salida "i" del componente anterior. Esto asegura que se utilicen todos los perfiles establecidos en las iteraciones.

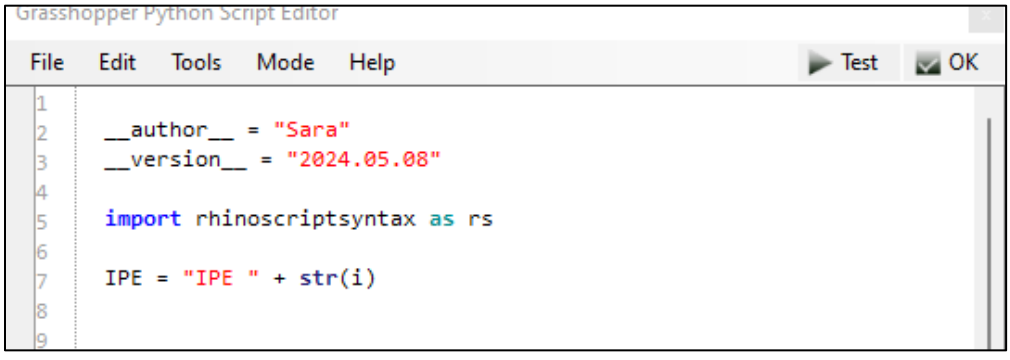


Ilustración 25. Código simple para añadir cadena de texto "IPE"

Para agregar el prefijo "IPE" antes del número de manera segura, utilizaremos un módulo de Python con un código simple que añadirá este texto al número variable. La salida de este módulo, llamada "IPE", se conectará al input "Section Name" del módulo "Section" que definimos previamente en la geometría.

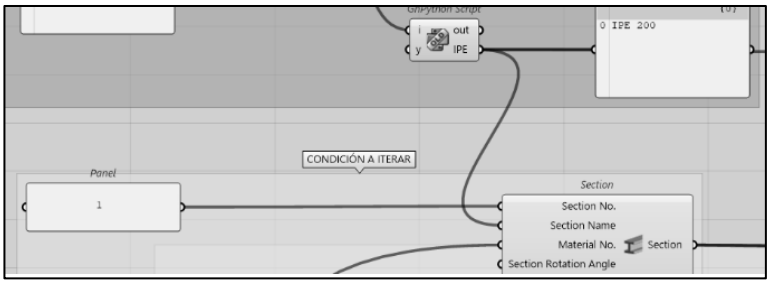


Ilustración 26. Integración al módulo section.

5.2. Condiciones de contorno.

En este caso, solo estableceremos una condición, la cual debe cumplir con el siguiente criterio: $$Radio\ objetivo - \left(\frac{L}{Uz}\right) \leq 0$$

Comenzamos estableciendo la longitud (L) utilizando una slider. Aunque podríamos vincular este valor a la longitud previamente establecida en la geometría para mantener la coherencia, hemos decidido utilizar la slider para obtener una visualización más clara. Inicialmente, la longitud está en metros; por lo tanto, utilizamos un módulo de multiplicación y añadimos la cifra 1000 en un panel para convertir la longitud a milímetros (mm).

Posteriormente, procedemos a dividir esta longitud entre la deformación, también en milímetros (mm). Para ello, utilizamos el módulo de división, ambos localizados en la sección “Maths”.

Luego, realizamos una resta entre el radio objetivo y el resultado de la división anterior. El radio objetivo es un valor crítico que definimos nosotros mismos y que representa un parámetro clave de diseño, puede ser un valor derivado de la relación entre la carga aplicada y la rigidez de la estructura, lo que afecta directamente la estabilidad y el comportamiento estructural bajo cargas. En este caso, el radio objetivo de 122 se expresa en unidades de mm/kN, lo que nos da una medida de la deformación por unidad de carga aplicada.

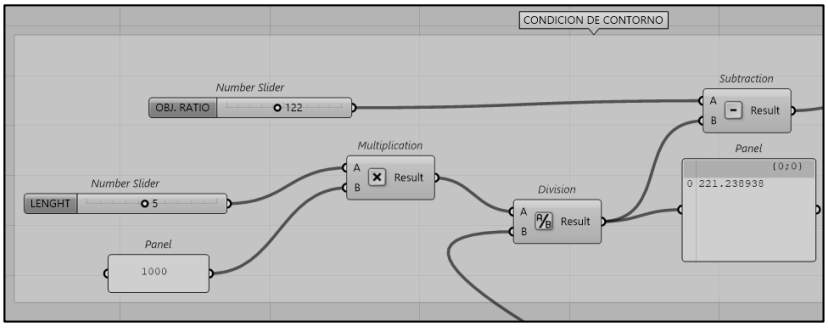


Ilustración 27. Condiciones de contorno (I).

El criterio que seguimos es que el resultado de esta resta debe ser menor o igual a cero. Para verificar esta condición, utilizamos un módulo llamado "smaller than". En este módulo, la primera entrada (first number input) será el resultado de la resta y la segunda entrada (second number) será un panel con un valor de cero.

La salida (output) de este módulo "smaller than" estará conectada a la salida del bucle. Esto nos indica que, una vez que se cumpla esta condición, el bucle se detendrá, señalando así que hemos encontrado el resultado más óptimo.

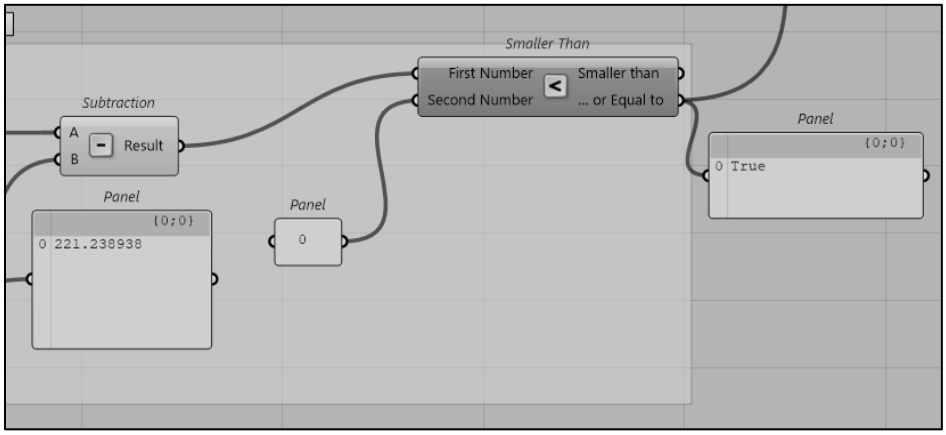


Ilustración 28.Condiciones de contorno (II)

5.3. Componente LOOP END.

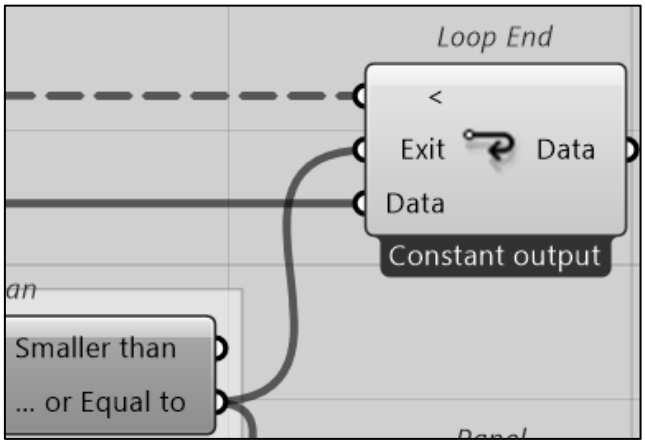


Ilustración 29. Loop end.

Básicamente cerrará nuestro bucle, sus tres inputs han de definirse de manera rápida y sencilla si es que no están definidos ya, el primero “<” se conecta al output “>” del LOOP START; el output “Exit” recién lo acabamos de conectar a la salida de la condición de contorno. Finalmente, el output “Data” será el output donde extraemos el perfil IPE de la lista, dentro de la zona de control del LOOP START, a la salida de “List item”. Nos devolverá el valor del IPE seleccionado como el más óptimo según este criterio.

Y hasta aquí llegaría este sencillo bucle para iniciar el manejo de estructuras con sus respectivos parámetros y optimizaciones en una aplicación de programación visual como es Grasshopper.

6. Registro de resultados y selección del óptimo.

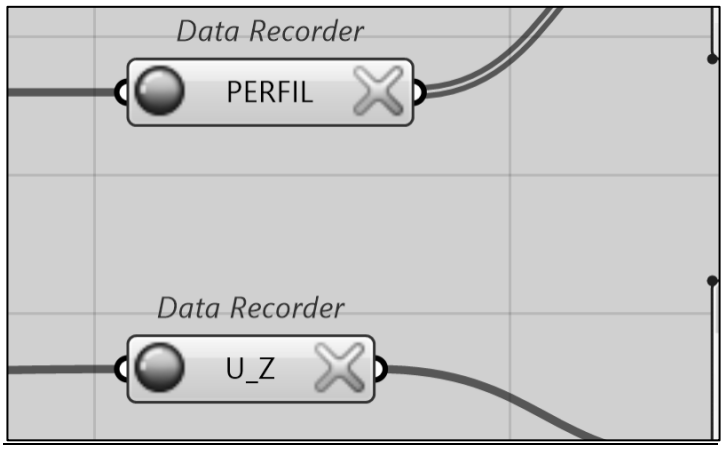


Ilustración 30. Registro de variables.

Usaremos dos módulos "Data Recorder" vinculados a nuestros perfiles y a las deformaciones verticales Uz que va sufriendo nuestra viga para luego poder extraer gracias al componente "List Item" el último valor de ésta, que será aquel que nos muestre el resultado óptimo. Como explicamos anteriormente, el bucle finaliza al alcanzar este resultado, será por tanto que aquel en la última línea será el óptimo.

Para extraerlo, vincularemos como ítem el número de iteración por el que se encuentra el modelo al memento de paro, y como lista cada uno la suya correspondiente, bastaría con ello para obtener los resultados finales de este simple modelo. Como error a destacar, en algunas visualizaciones se repite el valor final de nuevo como reconfirmando la valía de este, no es un problema significativo ya que no altera la validez del resultado y se podría solventar tras un análisis más profundo.

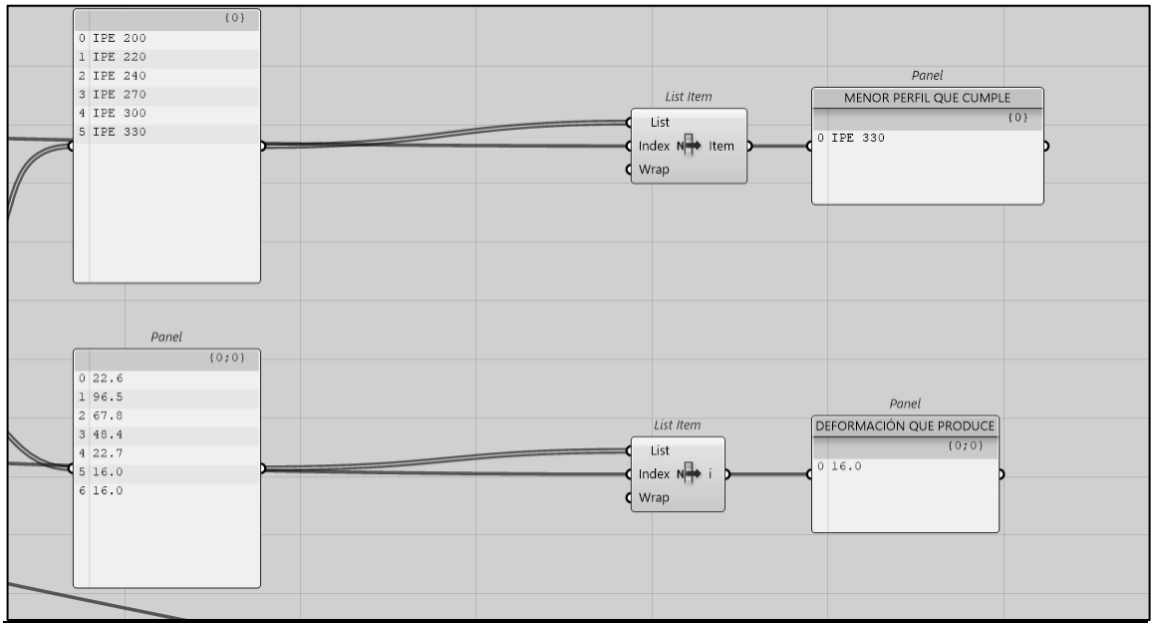


Ilustración 31. Extracción del resultado.

3.3.2. Caso 2.1: Celosía con carga aplicada.

3.3.2.1. Definición de la estructura.

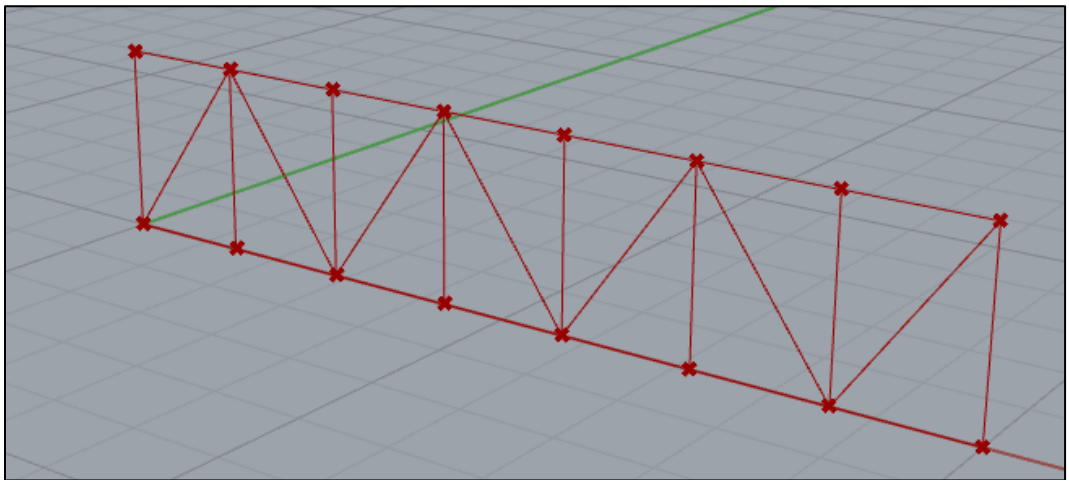


Ilustración 32. Caso 2: Celosía en Rhino.

1. Planteamiento del Problema.

En este nuevo caso, a pesar de que la complejidad aumentará en etapas posteriores, la definición de la estructura inicial, sobre la cual se basarán todos los cálculos y estudios posteriores, resulta bastante sencilla e incluso similar al proceso del modelo anterior. Por esta razón, no repetiré información detallada en el apartado previo si es que es igual que en este, ya que será más sencillo de comprender de este modo.

En este caso, nos proponemos construir una celosía simple con una carga uniformemente distribuida que abarca el conjunto de vigas asignadas a una agrupación denominada miembro. Estas vigas conforman la estructura y sobre ellas estudiaremos y extraeremos su deformación vertical. El objetivo de este modelo será iterar posteriormente entre combinaciones del número de divisiones de la celosía y perfiles IPE de sección, comparando un parámetro que implica una relación entre estos dos al que denominaremos Fitness y será donde comenzaremos a integrar el uso de los Algoritmos Genéticos para así encontrar el modelo más óptimo, eficiente, sostenible y económico.

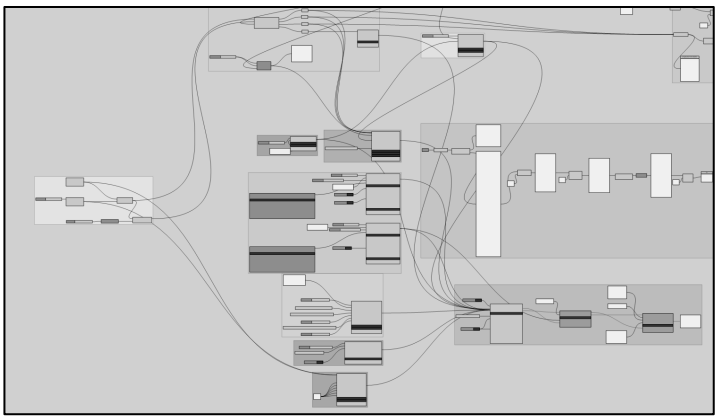


Ilustración 33. Vista general del modelo de definición de la celosía.

2. Construcción del Modelo Inicial

2.1. Creación de la Geometría Básica

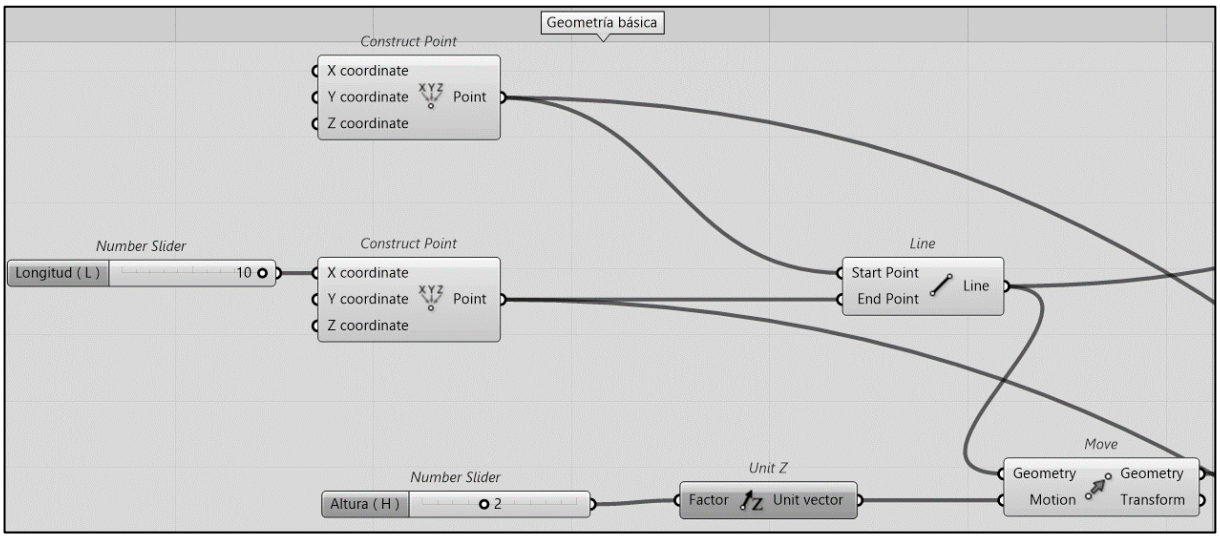


Ilustración 34. Geometría básica celosía.

Comenzamos añadiendo dos componentes "Construct Point" para establecer los puntos inicial y final de la viga. En el segundo punto, definimos la longitud deseada de la viga utilizando un componente "Slider" que nos permita variar esta longitud si es necesario. En este caso, hemos utilizado una longitud de x=10. Luego, para formar una línea uniremos estos dos puntos utilizando el componente "Line".

Dado que nuestro objetivo es replicar esta misma barra dándole una determinada altura para posteriormente crear una celosía, añadimos el componente "Move", que duplicará la línea creada con un determinado movimiento que hemos de definir. En el input "Geometry", conectamos el output del módulo "Line". En el input "Motion", estableceremos la altura a la que se situará la réplica. Para ello, necesitaremos un componente "Slider" que determine el valor deseado, en este caso 2 metros de altura, y el componente "Unit Z" que indica la dirección en la que se realizará este movimiento, queremos que se desplace hacia arriba paralelamente a la línea anterior, por tanto, coordenada (Z+) según nuestro eje. Conectamos el "Slider" al input del "Unit Z" y el output de este módulo al input "Motion" del componente "Move".

2.2. Conversión a Celosía

Para transformar las dos simples líneas en una celosía, utilizaremos el plugin de Dlubal que proporciona componentes específicos para este propósito. Emplearemos el componente "2D Truss", al cual vincularemos como inputs las dos líneas creadas previamente, además de un número de divisiones que inicialmente podemos establecer mediante un panel con un valor arbitrario. Este número de divisiones será uno de los parámetros que variará constantemente en nuestro modelo de optimización topológica.

- Los outputs de este componente incluirán todos los elementos de la celosía separados por grupos, lo que nos permitirá manejar aquellos que necesitemos o particularizar casos específicos.

- Para lograr esto, añadiremos tres componentes "Line" y uno "Point" desde la ventana "Params", y conectaremos los tres outputs que diferencian las líneas horizontales, verticales y diagonales a cada uno de estos componentes, mientras que el output de los nodos estructurales lo conectaremos al componente "Point".

- Posteriormente, emplearemos el componente "Node" también de Dlubal para convertir estos elementos en los nodos estructurales que se establecerán en la celosía.

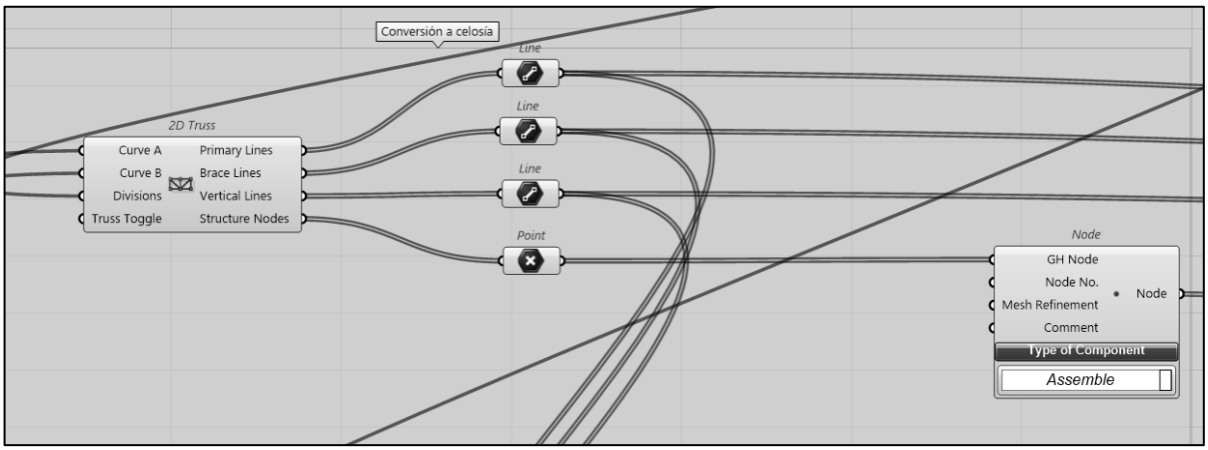


Ilustración 35. Conversión a celosía.

2.3. Creación del Miembro

En este apartado, el proceso será muy similar al del modelo anterior. Comenzaremos conectando los tres módulos "Line" creados previamente al input "GHline" del módulo "Member" de Dlubal para englobar todas las barras de la estructura.

Luego, vincularemos el "Member Number" al módulo "Series", el cual nos permitirá considerar el número de divisiones del miembro independientemente de cualquier variación. Iniciaremos la serie en el input "Start" con una slider que marque 1, y utilizaremos la misma slider para el input "Step", ya que queremos que el paso entre números sea de 1.

Finalmente, conectaremos el "Count" al lugar donde se haya definido el número de divisiones de nuestra estructura. En este caso, inicialmente podemos vincularlo a un panel, pero más adelante veremos cómo vincularlo a otro lugar, ya que el número de divisiones está cambiando en cada cálculo. Para finalizar, utilizaremos una "Value List" para el input "Member Type", donde seleccionaremos el tipo "Beam".

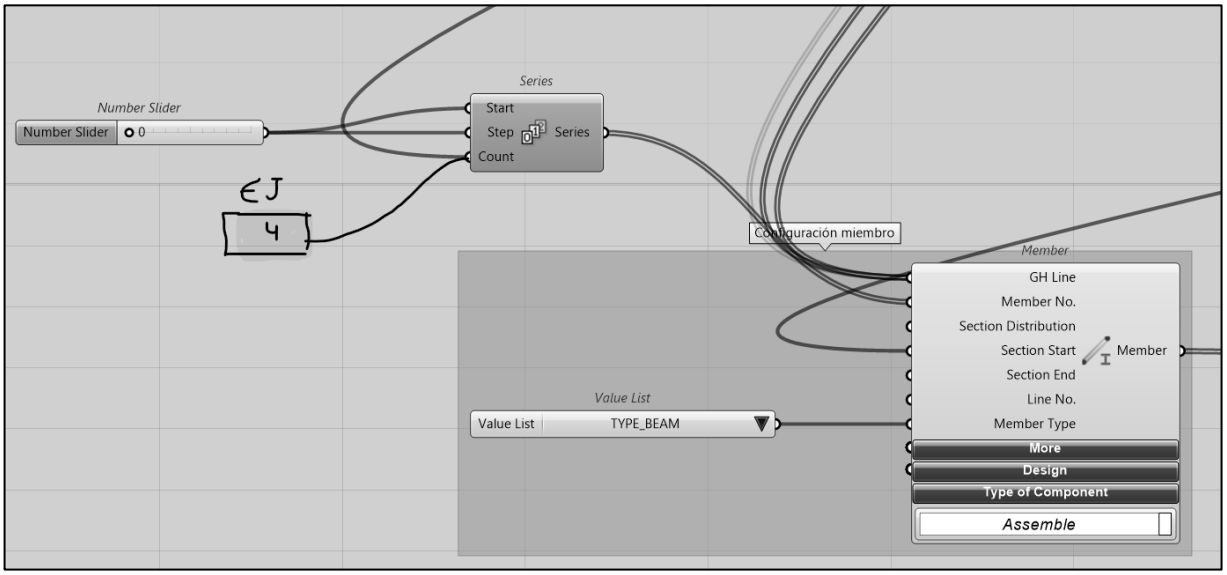


Ilustración 36. Creación del miembro celosía.

Ahora pasaremos a definir las otras dos características de este miembro:

2.3.1. Sección Inicial

Al igual que antes, podemos establecer un panel provisional para el input "Section Name", ya que luego lo cambiaremos para introducir el bucle. Respecto al "Section Number", utilizaremos una slider marcando 1, y vincularemos el "Material Number" a continuación. Esto se realizará mediante el componente "Section" del plugin de Dlubal.

2.3.2. Material

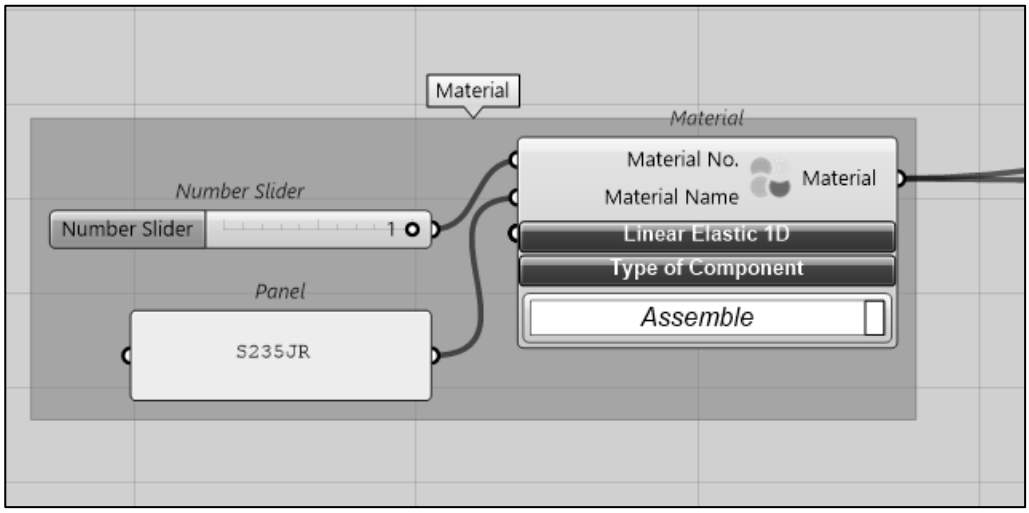


Ilustración 37. Definición del material celosía.

Agregaremos el componente "Material" de Dlubal, utilizando una slider marcando 1 para el "Material Number", y un panel con el nombre del material que será acero " S235JR". Este material se ha seleccionado debido a su amplia utilización en este tipo de estructuras.

2.4. Creación de los Apoyos Nodales y Configuración de Análisis Estático

En este paso, aplicaremos el mismo método utilizado anteriormente en el modelo de la viga simple.

En la creación de los apoyos nodales, establecemos nodos en los puntos inicial y final de la estructura para garantizar su estabilidad. Utilizamos el componente "Nodal Support" de Dlubal, fijando los movimientos de translación y rotación en coordenadas específicas para crear apoyos fijos.

En cuanto a la configuración del análisis estático, empleamos el componente "Static Analysis Settings" de Dlubal. Seleccionamos el tipo de análisis como "Geométricamente Lineal" para evaluaciones rápidas y precisas bajo cargas moderadas. También establecemos el número de análisis estáticos con un slider y manejamos las excepciones con un botón booleano para una fácil identificación de errores.

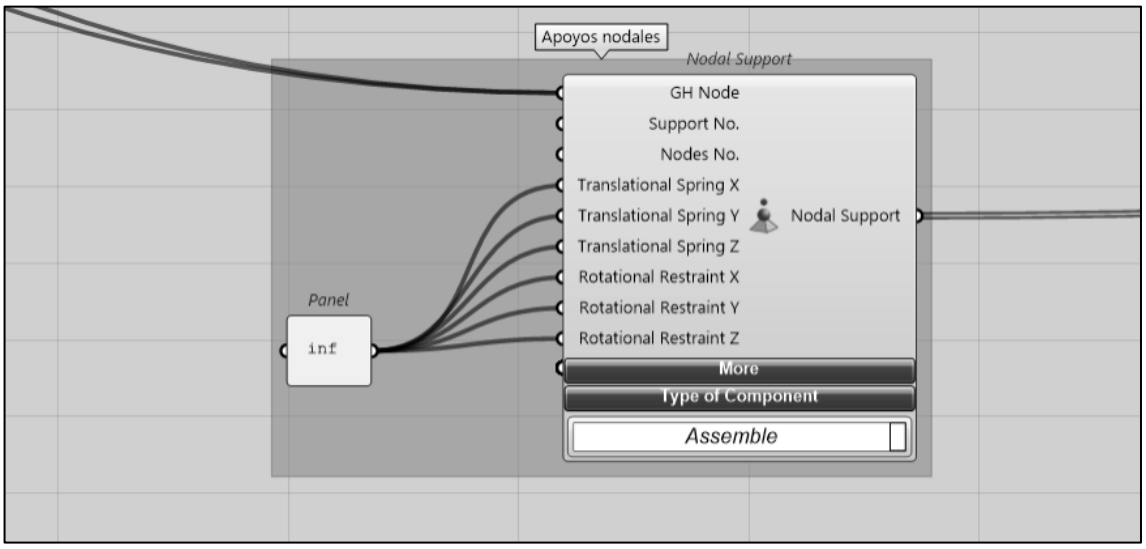


Ilustración 38. Creación de los apoyos nodales celosía.

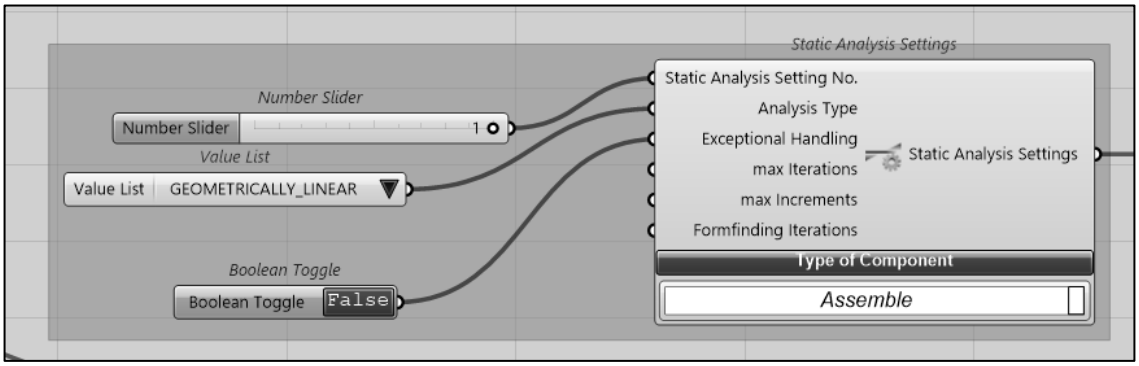


Ilustración 39. Creación del estado de análisis estático celosía.

2.5. Creación de los Casos de carga

En la configuración de los casos de carga, el procedimiento es prácticamente idéntico al modelo anterior. Para representar dos casos de carga distintos, uno correspondiente al peso propio de la viga y otro a una carga impuesta, seguimos los mismos pasos detallados anteriormente. Utilizamos el componente "Load Case" de Dlubal dos veces, configurando cada caso de carga de la siguiente manera:

- Agregamos sliders para determinar el número del caso de carga y paneles para definir el nombre de cada caso.
- Asignamos sliders para establecer el tipo de ajuste de análisis estático, conectados al input correspondiente de cada módulo.
- Uso de boolean toggles para indicar si se activa/desactiva cada caso de carga en cálculo.
- Añadimos componentes "Load Case Classification" para asociar categorías de carga en el componente "Load Case".

Para el primer caso de carga, seguimos la misma configuración que en el modelo anterior, estableciendo la categoría de acción como "Permanent | G" y el tipo de situación de diseño como "ULS (EQU) – Permanent and transient".

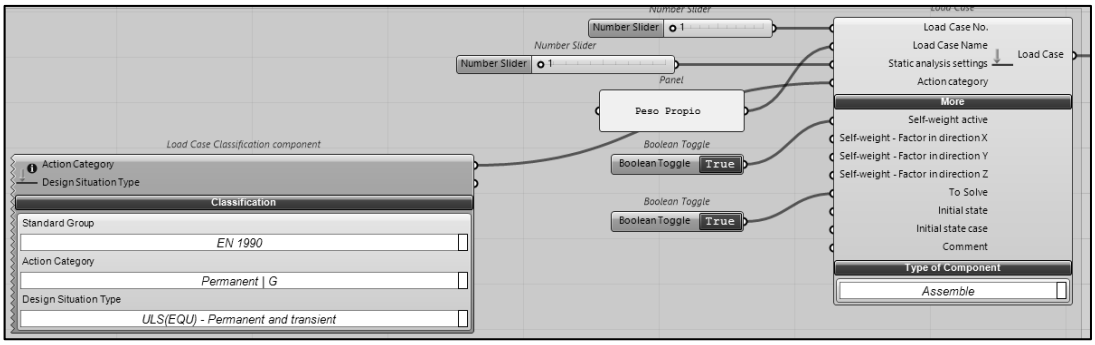


Ilustración 40. Caso de carga (I) Celosía

Para el segundo caso de carga, mantenemos la misma estructura, cambiando la categoría de acción a "Imposed Loads – Category A: domestic, residential áreas | Q/A" y seleccionando el tipo de situación de diseño "ULS (STR/GEO) – Permanent and transient – Eq. 6.10", que abarca la estabilidad geotérmica bajo condiciones extremas.

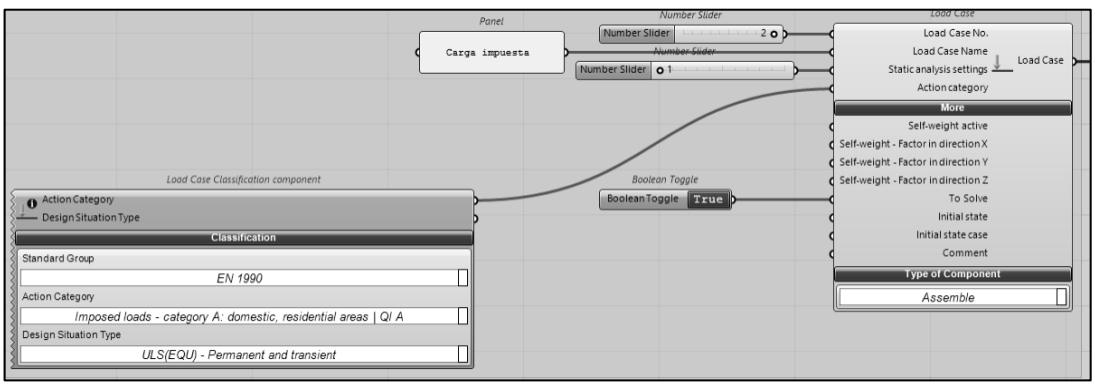


Ilustración 41. Caso de carga (II) Celosía.

2.6. Creación de las cargas en los miembros

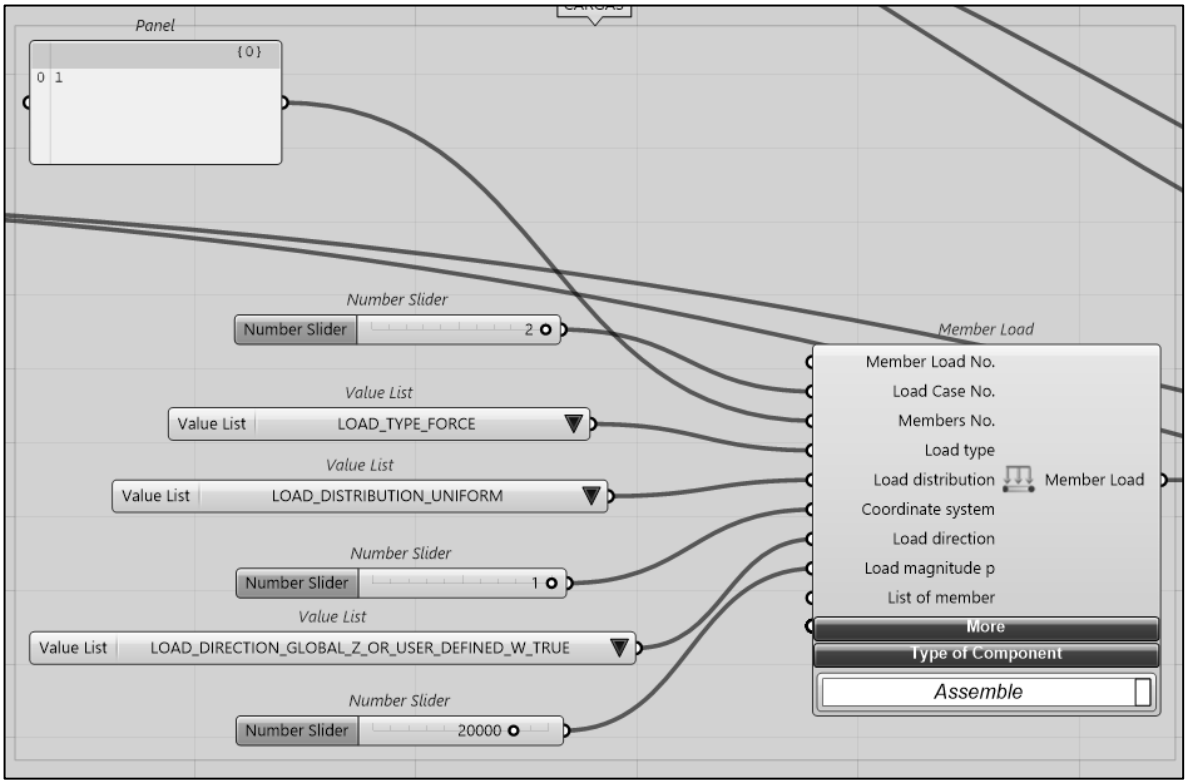


Ilustración 42. Creación de cargas en los miembros celosía.

En cuanto a las cargas en los miembros, se utilizan los mismos componentes y métodos descritos previamente para asignar cargas uniformemente distribuidas a los miembros de la estructura.

La única diferencia notable es que, debido al aumento en la longitud y el número de barras en esta nueva estructura, se aplica una carga mayor de 20 kN en este caso, en contraste con la carga utilizada en el modelo anterior. Este ajuste se realiza para tener en cuenta la mayor complejidad y carga estructural resultante de la ampliación de la estructura.

Para configurar las cargas en los miembros, se utiliza el componente "Member Load" de Dlubal. Este componente requiere la especificación de varios inputs que se definen de manera similar al modelo anterior.

Los pasos clave incluyen:

1. Especificación del caso de carga.
2. Selección del tipo de carga (en este caso, una carga de fuerza).
3. Determinación de la distribución de la carga (uniforme en este caso).
4. Establecimiento del sistema de coordenadas.
5. Definición de la magnitud de la carga.

2.7. Exportación a RFEM 6

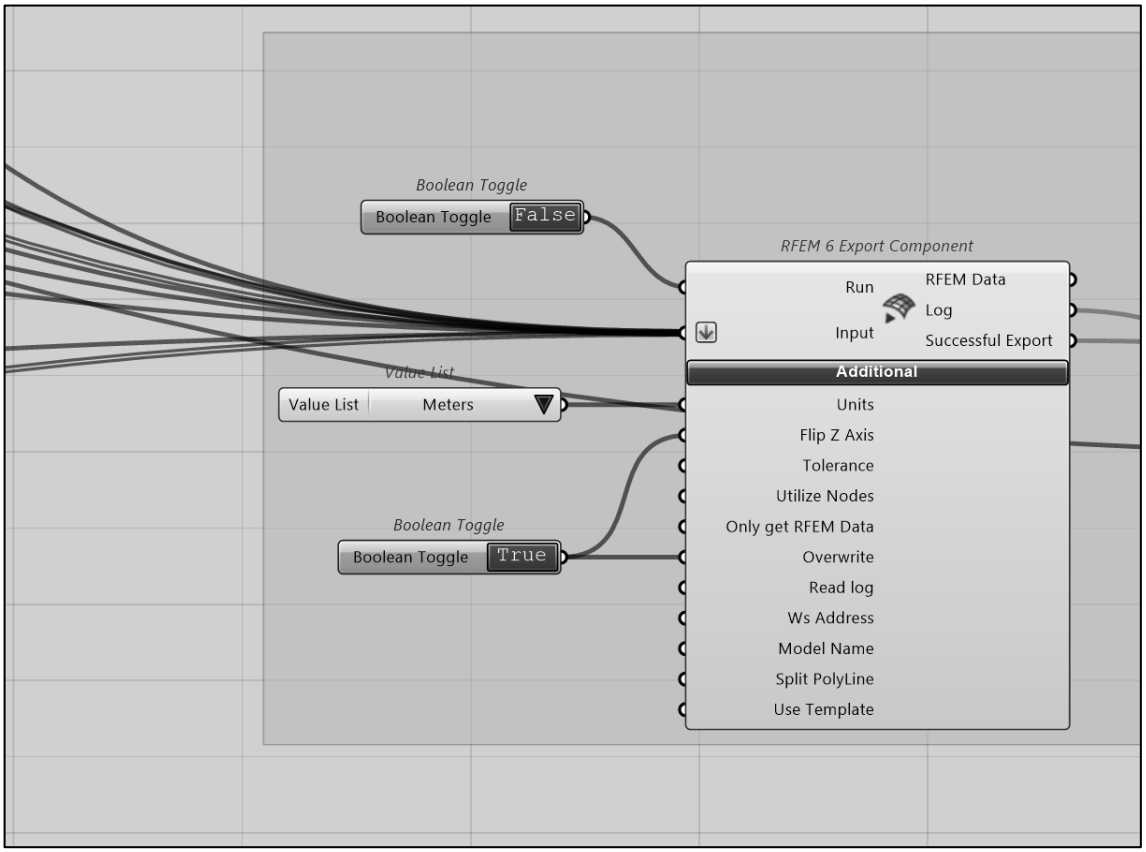


Ilustración 43. Módulo principal exportación a RFEM celosía.

Para exportar el modelo a la aplicación RFEM 6 de Dlubal, se utiliza el módulo RFEM 6 Export Component en Grasshopper. Aquí se detalla de manera resumida la gestión de los inputs al componente ya que se procede de igual modo en este caso:

- **Run**: Se conecta un "Boolean Toggle" que cambia de False a True para iniciar la exportación y cálculos en RFEM.
- **Input**: Se realizan 8 conexiones para importar al programa de cálculo estructural, que incluyen:
  - Sección
  - Material
  - Miembros
  - Apoyos nodales
  - Ajustes de análisis estático
  - Dos casos de carga: self weight e imposed load
  - Carga en el miembro
- **Additional settings**:
  - **Units**: Se selecciona "metros" de una "Value list".
  - **Overwrite**: Se utiliza un "Boolean Toggle" establecido en True para sobrescribir modelos existentes en RFEM y así actualizarlos según sea necesario.

- **Exportación de resultados gh/rfem a csv en Excel**

Para exportar los resultados del modelo de RFEM a archivos CSV en Excel, se siguen los siguientes pasos:

1. Se agrega el componente "RFEM 6 Calculate Component" para extraer los cálculos del modelo, con los siguientes inputs:
   - "Run": Conectado al output "Successful Export" del módulo anterior "RFEM 6 Export Component", indica el inicio del cálculo después de la exportación.
   - "Model name": Un panel con el nombre del archivo del modelo en RFEM y Grasshopper.
   - "Cases to calculate": En la pestaña "Additional", se vincula el caso de carga deseado, como "Imposed Load".
   - "Read Log": Conectado al output "Log" del módulo "RFEM 6 Export Component" para transmitir información relevante.

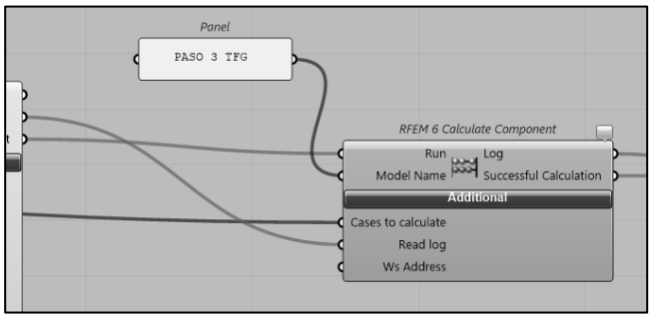


Ilustración 44. Componente de cálculo RFEM celosía.

2. Se añade el componente "Export Results to Files" para realizar la exportación, con los siguientes inputs:
   - "Run": Conectado a "Successful Calculation", similar al módulo anterior.
   - "Model name": Se utiliza un panel para especificar el nombre del modelo, coincidiendo con el anterior.
   - "Path to Export Folder": Se especifica la ubicación para guardar los archivos CSV mediante un panel que define la ruta.
   - "Type of Export": Se elige "CSV" en un panel para exportar en este formato.
   - "Read Log": Conectado al output "Log" del módulo anterior para obtener información relevante durante la exportación.

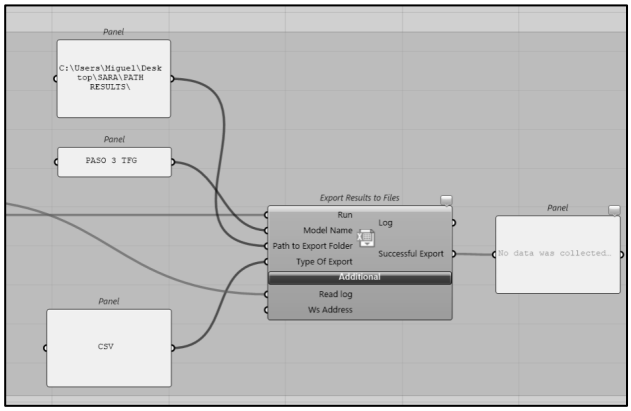


Ilustración 45. Exportación resultados a CSV celosía.

- **Lectura datos del CSV en GrassHopper.**

  Para adaptar la extracción del parámetro de la deformación vertical Uz en este nuevo modelo, donde incorporamos el número de divisiones de la celosía (NDIV), seguimos un nuevo procedimiento detallado, para el que se mostrará en una imagen en la página siguiente.

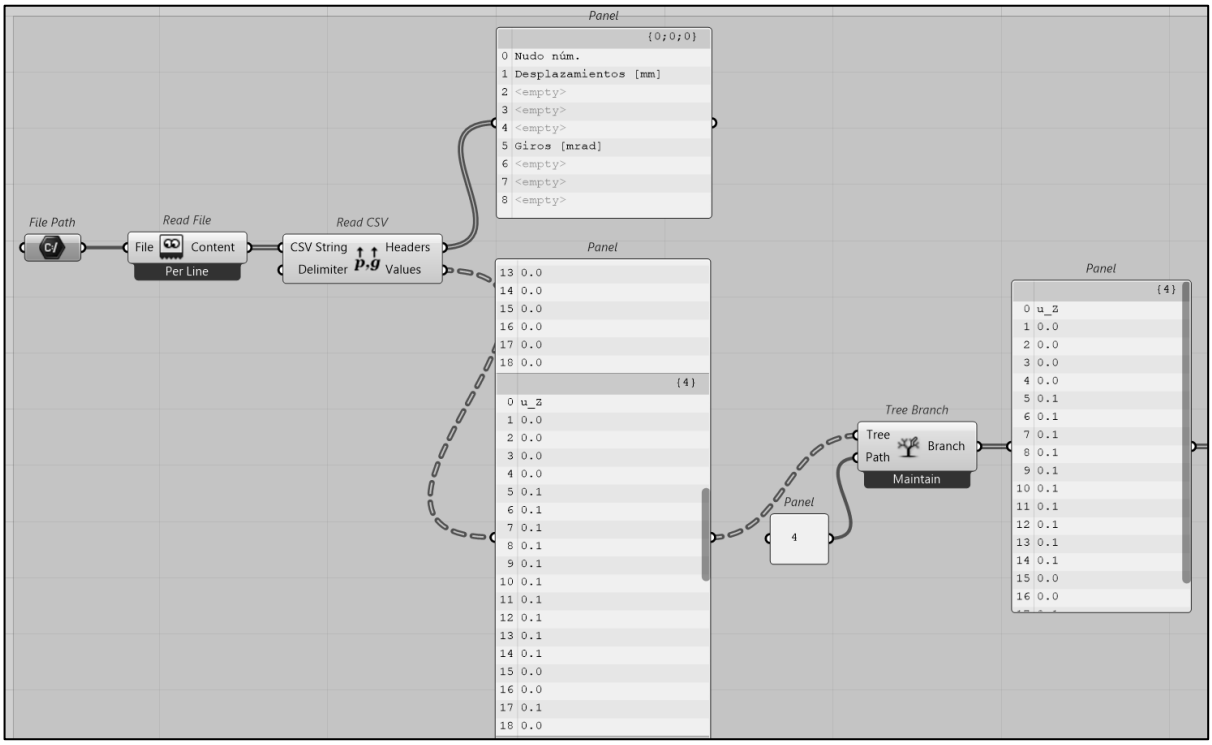


Ilustración 46. Lectura datos CSV (I) celosía.

- Añadimos el módulo "File Path" para vincular la ruta del archivo que contiene las deformaciones nodales, manteniendo la misma ubicación que en el modelo anterior.
- Conectamos el módulo "Read File" al "File Path", seguido del módulo "Read CSV" para obtener los encabezados y el contenido del archivo, como se hizo previamente.
- Utilizamos el módulo "Tree Branch" para extraer la columna deseada (columna 4), indicándolo en el input "Path" con un panel y vinculando el panel con los valores al input "Tree". Conectamos un panel al output para visualizar los resultados.
- Eliminamos la fila 0 que indica el nombre "U_Z" para evitar problemas futuros. Para esto, utilizamos el componente "Cull Index", conectando la lista al input "List" y un panel con el valor "0" al input "Indices". El output de este módulo será la lista depurada.
- Ordenamos la lista para asegurar que el mayor valor esté en el primer lugar. Utilizamos el componente "Sort List" y conectamos su output al input "Reverse List" para invertir el orden y que el mayor valor quede primero. Luego, conectamos un panel al output de "Reverse List" para visualizar los resultados.

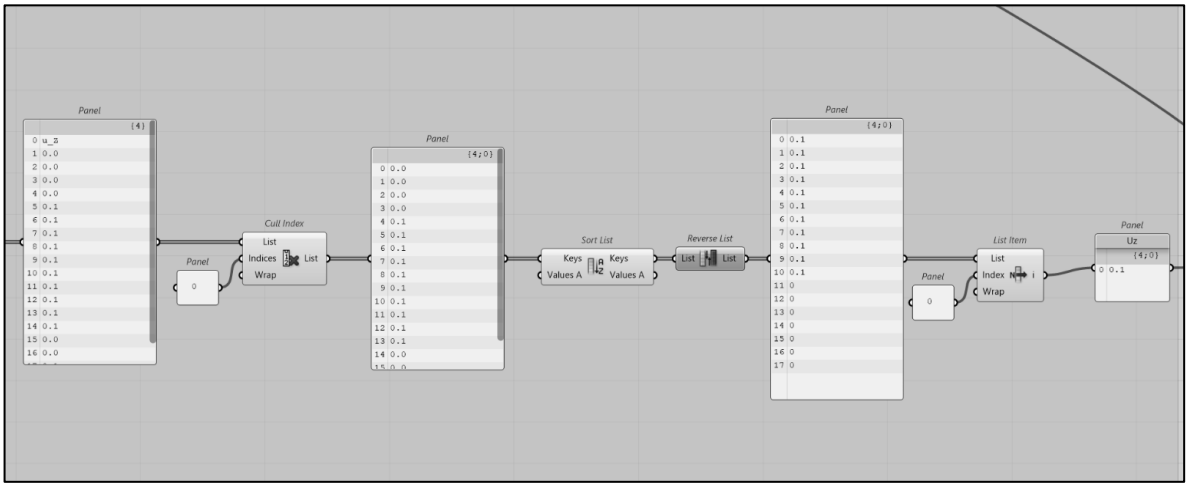


Ilustración 47. Lectura datos CSV (II) celosía.

Finalmente, utilizamos el componente "List Item" para extraer el primer elemento de la lista invertida, que corresponde a la deformación vertical UZ.

3.3.3. Caso 2.2: Celosía e introducción al código de Algoritmo genético.

1. <u>Planteamiento</u>

Una vez concluido el proceso de definición de la estructura y la aplicación de las cargas, así como la extracción del parámetro esencial de deformación vertical Uz, procederemos a implementar el código de algoritmo genético. Esto se logrará mediante el establecimiento de una población inicial y la generación de las primeras iteraciones, con el objetivo de obtener una serie de resultados óptimos comparando varias opciones generadas.

Para este propósito, utilizaremos el plugin Anemone en Grasshopper, que nos permite crear bucles. Implementaremos dos bucles: uno para definir la población inicial y otro anidado dentro del primero, que se encargará de generar las sucesivas generaciones.

A continuación, detallaremos cada uno de estos bucles y el cálculo de los parámetros necesarios para comenzar a ejecutar este algoritmo de manera visual

2. <u>Bucle Interno: Población Inicial</u>

El objetivo de este bucle es generar tres individuos aleatorios, cada uno con dos genes (características): el perfil IPE y el número de divisiones de la celosía (NDIV). A partir de estos genes, se calculará un parámetro denominado fitness, que comprende una relación entre la deformación producida por las cargas aplicadas a la estructura y el peso de esta. Este parámetro permitirá determinar cuál de las tres estructuras evaluadas es la más óptima.

2.1. Inicialización del Bucle

Iniciaremos el bucle con el módulo **Loop Start** de Anemone, al cual configuraremos con los siguientes inputs y outputs:

- **Trigger**: Utilizaremos un botón para iniciar el bucle.
- **IPE**: Añadiremos esta variable tanto en el input como en el output, colocando un panel en la parte de output para visualizar el valor que se obtiene en cada iteración.
- **NDIV**: Seguiremos el mismo procedimiento que para el IPE.

El input **Repeat** estará vinculado con el output **Counter**, ya que los utilizaremos para definir el número de veces que se repetirá el bucle. Procederemos de la siguiente manera:

1. Añadiremos un panel en el que se especifique el número de veces que queremos que se repita el bucle, que corresponderá al número de individuos de la población inicial. Este panel comenzará con el número 0.
2. Usaremos el componente **List Length** para obtener la longitud de esta lista, la cual determinará el número de iteraciones del bucle. El output de este módulo se conectará al input **Repeat**.
3. Utilizaremos la misma lista para convertir el contador en una variable, añadiendo el componente **List Item**. Este componente extraerá elementos de la lista al asociar el valor variable **Counter** a su input **Index**, proporcionando como parámetro "i" el número de iteración correspondiente.

Con el parámetro "i", definiremos la variable clave en el código de Python que generará números aleatorios a partir de las listas especificadas (perfil IPE y NDIV).

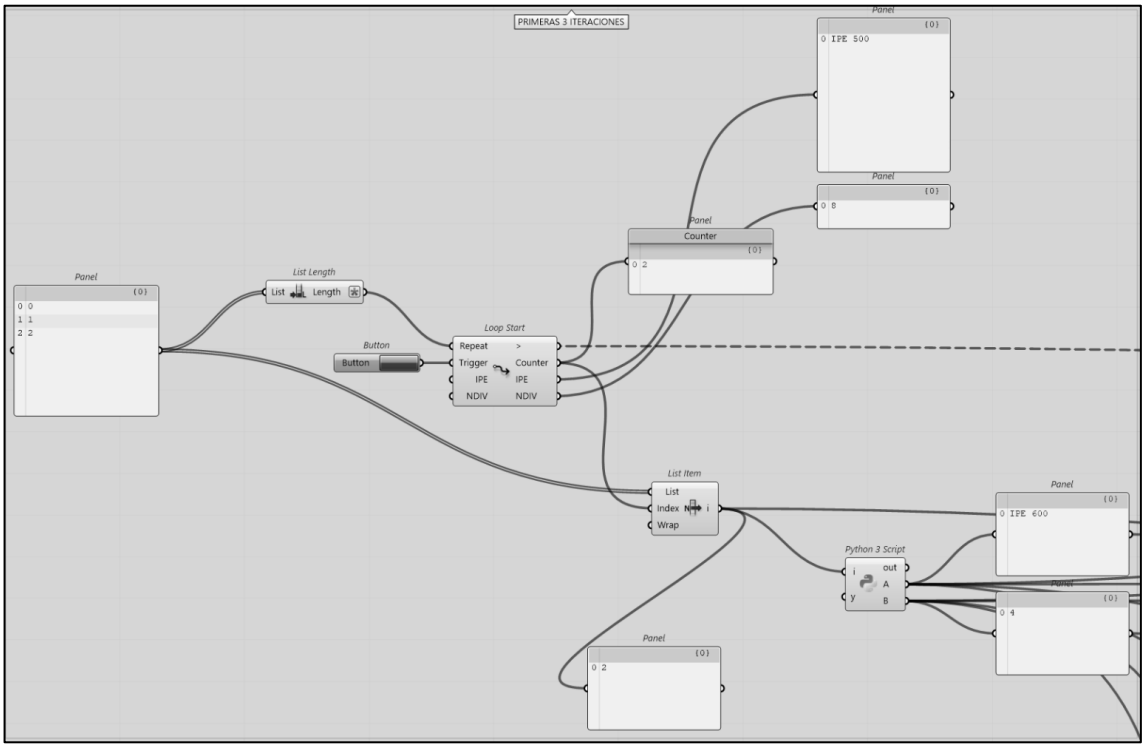


Ilustración 48. Inicialización del bucle celosía.

Para este código, importamos el módulo **random** de Python, que nos facilitará la generación de números aleatorios. Luego, definimos las dos listas correspondientes a perfil IPE y NDIV. Usaremos una estructura condicional **if** para manejar la variable "i":

- Si "i" es igual a 0, se generarán aleatoriamente un IPE y un NDIV, denominados A y B respectivamente.
- Si "i" es igual a 1, se repetirá el proceso generando nuevos valores (probablemente diferentes de los anteriores).
- En cualquier otro caso, el **else** generará nuevamente valores aleatorios.

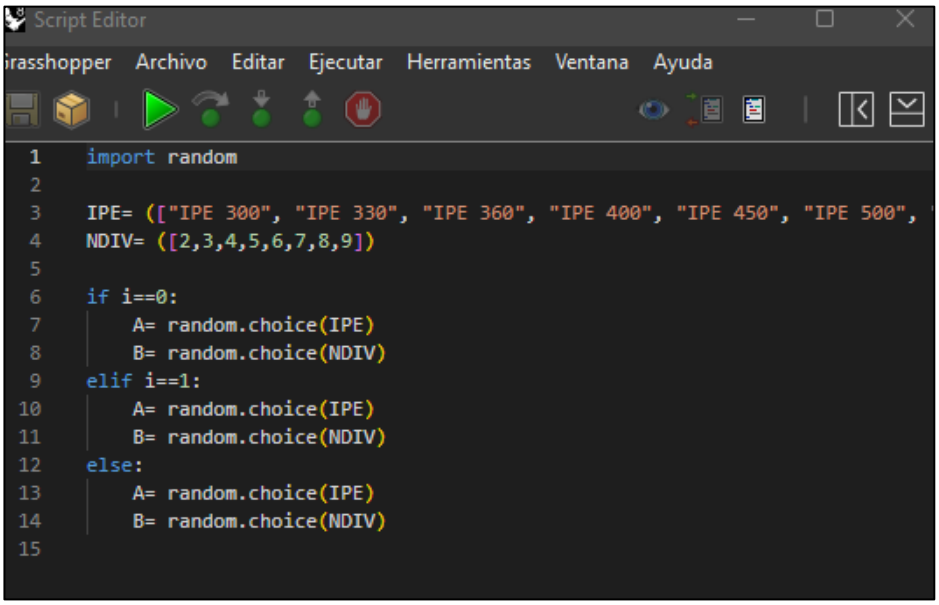


Ilustración 49. Código generación en aleatorio Python celosía.

Este método garantiza que se obtengan parámetros nuevos en cada iteración, proporcionando los valores A y B (IPE y NDIV) necesarios para varias partes del modelo. Los pasos específicos incluyen:

- Vincular A a **Section Name**, convirtiéndolo en el parámetro iterable y eliminando el valor provisional utilizado previamente.
- Vincular B a **Divisions** del módulo 2D Truss en la construcción de la celosía y a **Count** del módulo Series para el número de miembros.
- Añadir dos paneles para visualizar los resultados de estas variables.

2.2. Cálculo del Peso de la Estructura

El cálculo del peso de la estructura es esencial para la posterior definición del parámetro de fitness. Este parámetro se utiliza en la selección de individuos mediante un proceso conocido como torneo, donde se evalúan los individuos y el que tenga el mejor fitness es seleccionado como ganador.

Para realizar este cálculo, que puede parecer complejo sin conocer el modelo resultante, seguiremos los siguientes pasos:

1. Definición de la Posición del IPE:

   - Usaremos un pequeño código en Python para determinar la posición del perfil IPE en una lista previamente definida.
   - La entrada será la variable A, que corresponde al nombre del perfil IPE actual.
   - Mediante una estructura condicional **if**, el código asignará una posición a cada perfil en la lista y devolverá esta posición en una variable llamada "a".

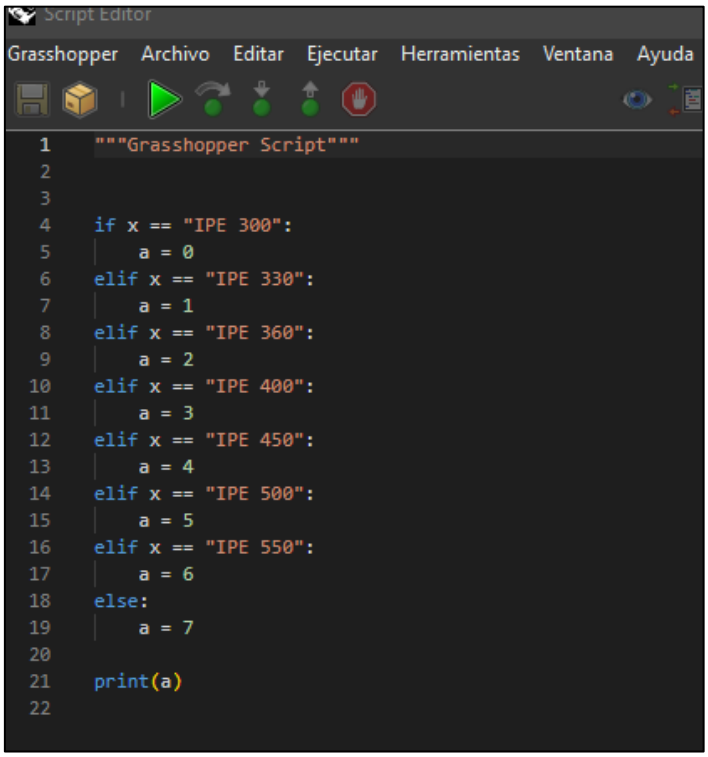


Ilustración 50. Código definición perfil Python celosía.

2. Asignación del Área del Perfil IPE:

   - Definimos una lista con el área en cm^2 correspondiente a cada perfil IPE, extraída de una tabla de propiedades de estos perfiles.
   - Utilizamos el módulo **List Item** para unir la lista al input list y la variable "a" al input index. Esto nos devolverá el área de la estructura según el perfil seleccionado.
   - Convertimos el área de cm^2 a m^2 mediante un cálculo simple.

3. Suma de Longitudes Individuales:

   - Unimos todos los componentes lineales establecidos en la geometría a un componente llamado **Length**, que proporcionará una lista con todas las longitudes individuales.
   - Usamos el componente **Mass Addition** para sumar todas las longitudes individuales. Este parámetro será variable ya que la geometría de la celosía está unida al parámetro variable NDIV, por lo tanto, el número de líneas se modificará con este.

4. **Cálculo del Volumen**:

   - Multiplicamos el área (en m^2) por la longitud total (en metros) para obtener el volumen en m^3.

5. **Cálculo del Peso**:

   - Multiplicamos el volumen por el peso específico del material, en este caso, acero S235JR, que es 7850 kg/m^3.
   - El resultado de esta operación, dado en kilogramos, representará el peso de la estructura.

Siguiendo estos pasos, obtendremos el peso de la estructura de manera precisa, considerando las variables IPE y NDIV que afectan directamente a la geometría y propiedades del modelo estructural.

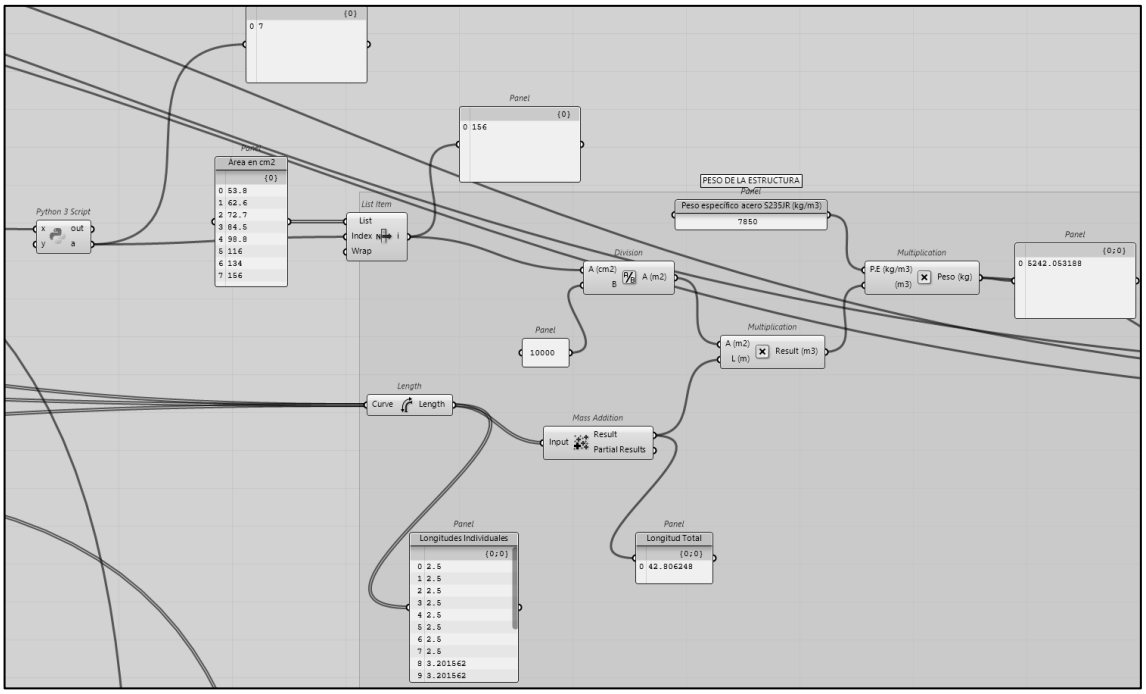


Ilustración 51. Cálculo del peso de la celosía.

2.3. Cálculo del Fitness y Grabación de Individuos de Población Inicial

Una vez obtenido el peso de nuestra estructura, podemos proceder al cálculo del fitness. Este cálculo es sencillo, ya que se trata de una división entre el parámetro Uz y el peso de la estructura. Utilizaremos un módulo **Data Recorder** para registrar el fitness de cada individuo. La selección del mejor individuo se explicará en el bucle exterior, ya que se interconectará y se realizará mediante un procedimiento conjunto.

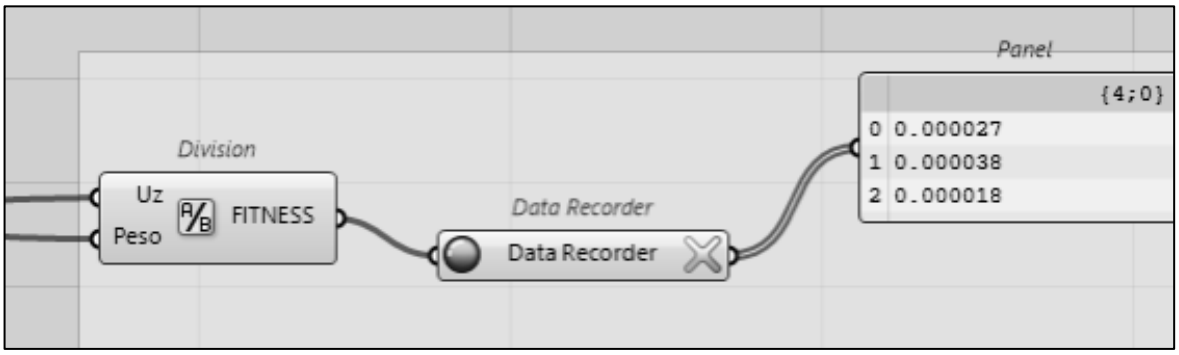


Ilustración 52. Cálculo del fitness de la celosía.

2.4. Cierre del Bucle

Para cerrar el bucle interno, debemos conectar cada uno de sus inputs a la ubicación correspondiente:

- Conectamos el input "<" al output ">" del **Loop Start**.
- Dejamos libre el input **Exit**, ya que no estamos solicitando ninguna salida específica, sino generando números aleatorios y analizando sus relaciones y propiedades para encontrar el óptimo.
- Creamos dos inputs, **IPE** y **NDIV**, que vincularemos a los parámetros "A" y "B" respectivamente, provenientes de la salida de la generación aleatoria.

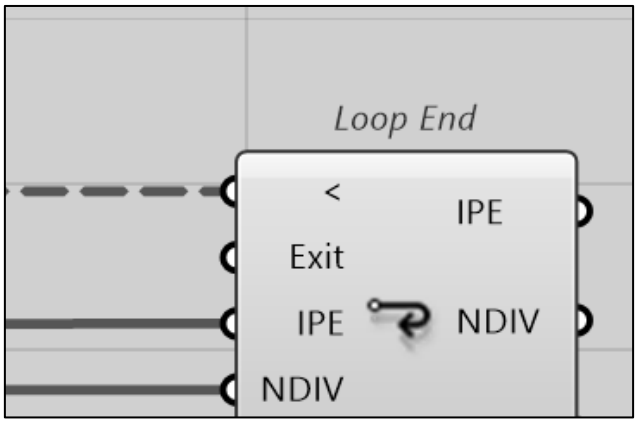


Ilustración 53. Cierre del bucle interno de la celosía.

3. Bucle Externo: Población Inicial y Dos Primeras Generaciones Aleatorias.

El objetivo de este bucle externo es repetir el bucle interno un número determinado de veces, generando así una población inicial y dos generaciones adicionales. A continuación, se describe el proceso detallado.

3.1. Inicialización del Bucle y Anidamiento al Interno

Vamos a repetir un proceso similar para la creación del segundo bucle, pero este será más sencillo porque se basa principalmente en la conexión adecuada con el bucle interior.

- **Definición de Inputs**:
  - **Input Repeat**: Añadimos un panel con el valor que deseamos, en este caso, 3, ya que queremos generar dos generaciones adicionales a la población inicial, lo que totaliza tres iteraciones del bucle interno.
  - **Input Trigger**: Insertamos un botón que activará este bucle.
  - **Input Data**: Este input también será un output del módulo **Loop Start** de Anemone.
- **Conexión con el Bucle Interior**:
  - Usamos el output **Counter** como input trigger del bucle interior. Esto implica eliminar el botón añadido en el bucle anterior y sustituirlo por la variable **Counter** del nuevo bucle externo.
  - Añadimos el parámetro **Data** en el bucle exterior y lo incorporamos también en el bucle interior, tanto en el input como en el output.

Esta configuración nos permitirá que el bucle externo controle la cantidad de veces que se repite el bucle interno, generando nuevos individuos y evaluándolos en cada iteración.

3.2. Cierre del Bucle

Para finalizar la conexión entre los bucles, se debe completar el siguiente proceso:

- Conectar el output **Counter** del módulo **Loop Start** del bucle interior con el input **Data** que se debe crear en el módulo **Loop End**.
- Unir el output **Data** de este mismo módulo al input **Data** del **Loop End** del bucle exterior.

4. Simplificación Visual del Modelo.

Para mejorar la claridad y organización del modelo, hemos utilizado el plugin **Telepathy**, que permite realizar conexiones inalámbricas entre cables de manera sencilla. Este proceso se realiza mediante los siguientes pasos:

- **Conexión con Telepathy**:
  - Insertar el módulo **Sender** en el output de la conexión deseada.
  - Colocar el módulo **Receiver** en el input correspondiente.
  - Vincular el **Sender** al output y el **Receiver** al input de todos los elementos que se desee conectar, evitando la visualización de cables.
  - Se pueden replicar tantos **Receivers** como sea necesario.
  - Es importante recordar nombrar de la misma manera al **Sender** y al **Receiver** para que la conexión funcione correctamente.

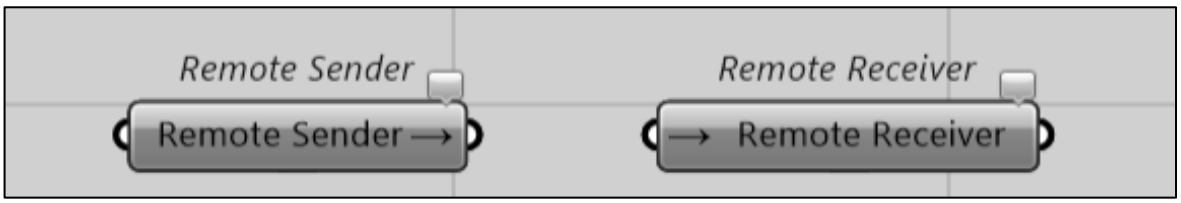


Ilustración 54. Componentes del plug-in telepathy.

A continuación, se muestra una comparación visual del modelo antes y después de la simplificación mediante **Telepathy**. Esta simplificación facilita la comprensión del modelo al eliminar el desorden visual causado por múltiples cables conectados, especialmente en secciones complejas. La imagen representa la organización del apartado que se explicará a continuación. Es importante realizar esta simplificación primero para facilitar la comprensión de los lectores en la próxima explicación detallada del modelo.

La utilización de **Telepathy** no solo mejora la presentación visual del modelo, sino que también facilita la navegación y comprensión de las conexiones entre diferentes componentes, haciendo el proceso de desarrollo y análisis más eficiente y claro.

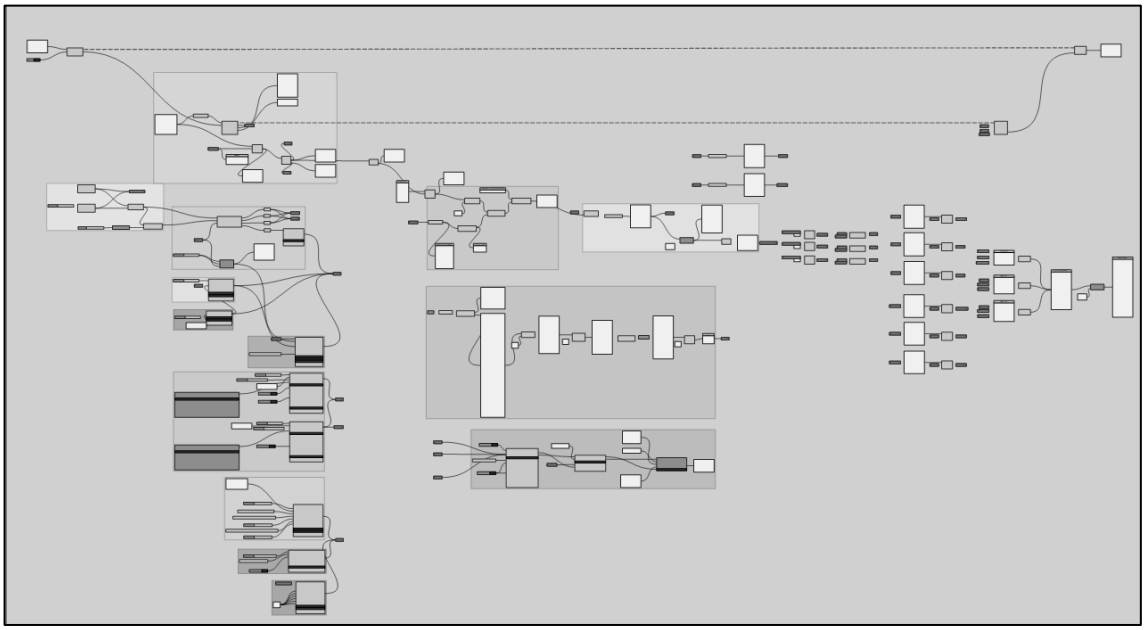


Ilustración 55. Modelo completo con telepathy.

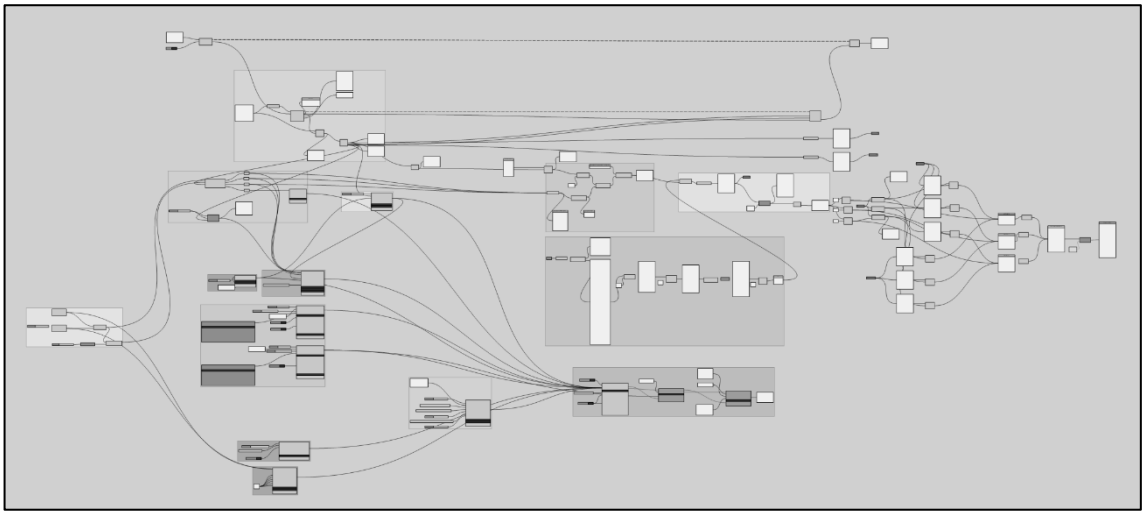


Ilustración 56. Modelo completo sin telepathy.

5. Obtención del Hall of Fame y grabación de resultados completos.

1. **Cálculo del Fitness y Grabación de Resultados**:
   - El cálculo del fitness seguirá el mismo procedimiento descrito anteriormente.
   - Se mantendrá el módulo para grabar resultados, que registrará los valores de fitness obtenidos.

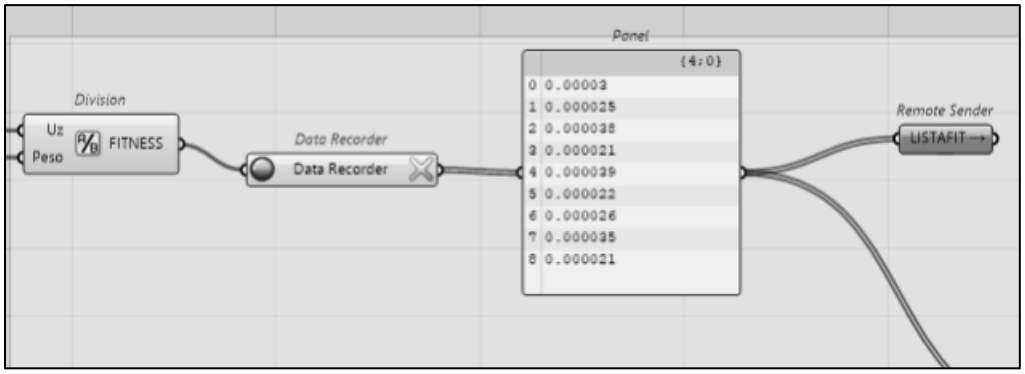


Ilustración 57. Grabación de resultados fitness celosía.

2. **Particionamiento de la Lista de Fitness**:
   - Se utilizará el módulo partition list para dividir la lista completa de fitness en grupos de 3, permitiendo así analizar los datos en bloques.
   - Un panel se configurará para definir el tamaño de cada partición. Este panel puede ser reemplazado por un slider para ajustar dinámicamente el tamaño de los grupos en caso de variar el número de individuos por generación.

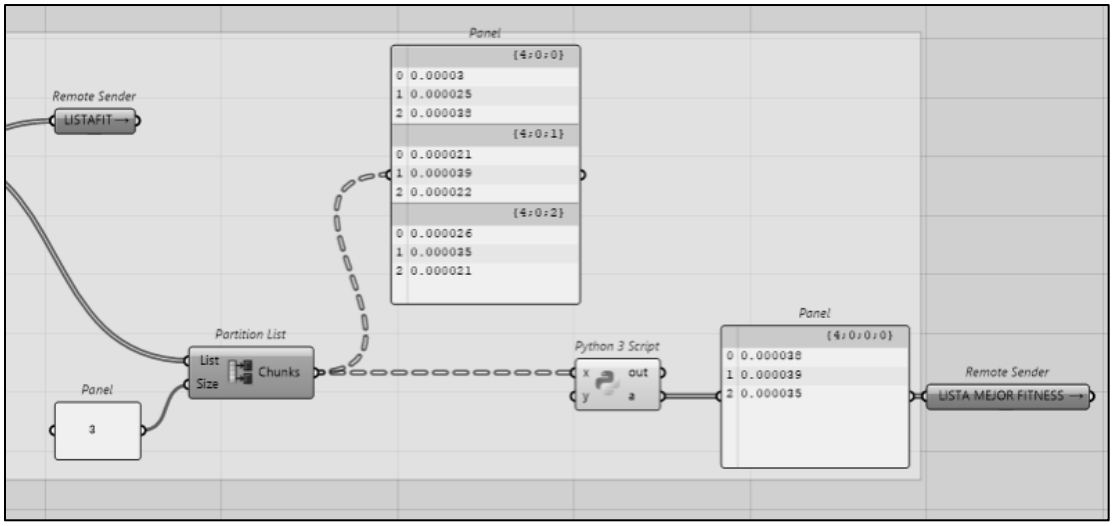


Ilustración 58. Ramificar la lista fitness celosía.

3. **Códigos de Python para Selección del Mejor Individuo**:
   - Un código en Python ayudará a identificar los valores máximos en cada rama del árbol de datos.

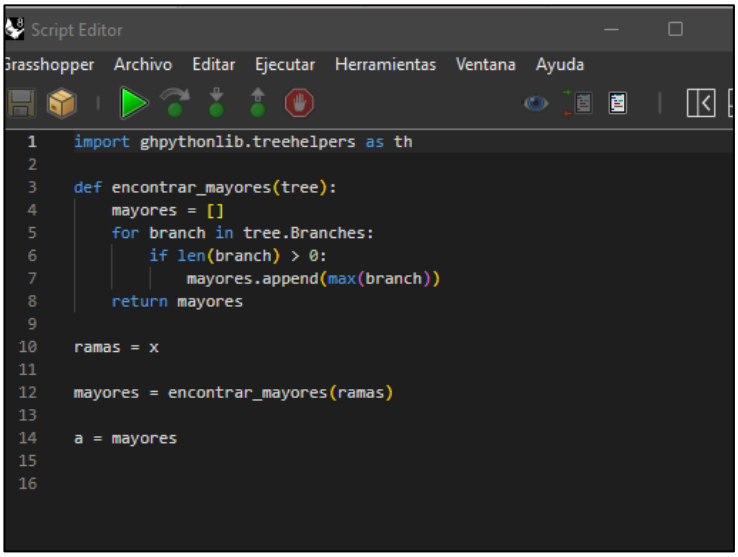


Ilustración 59. Código selección mejor individuo Python celosía.

- **Pasos del Código**:
  1. Importa la biblioteca ghpythonlib.treehelpers.
  2. Define una función encontrar_mayores que recorre cada rama del árbol, encuentra el valor máximo y lo agrega a una lista llamada mayores.
  3. Asigna el árbol de datos de entrada (variable x) a ramas.
  4. Llama a la función encontrar_mayores con ramas y guarda los valores máximos en mayores.
  5. Asigna mayores a la variable de salida "a".

4. **Selección del Mejor Individuo de Cada Grupo**:
   - El objetivo es seleccionar el mejor individuo (con el mejor fitness) de cada grupo de 3.
   - Un módulo telepathy sender denominado "lista mejor fitness" se utilizará para registrar los mejores individuos en una nueva lista.
5. **Grabación de Datos Adicionales (IPE y NDIV)**:
   - Dos módulos grabadores de datos se utilizarán para registrar IPE y NDIV, que serán necesarios para identificar los genes de los individuos con mejor fitness.
   - Las salidas correspondientes al perfil y número de divisiones de cada iteración se simplificarán usando telepathy para facilitar su uso posterior.

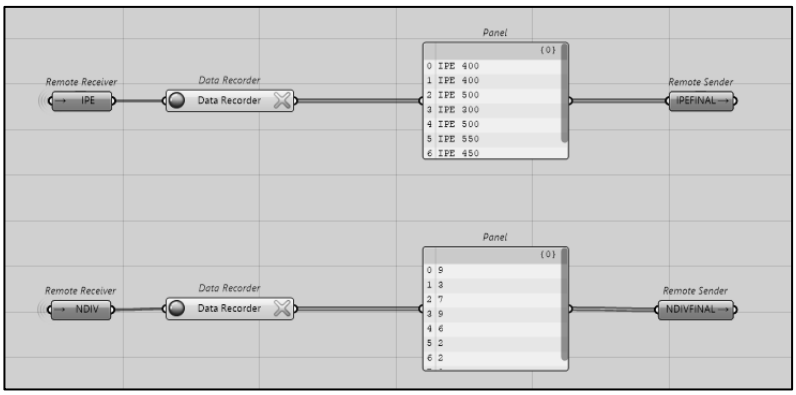


Ilustración 60. Grabación IPE y NDIV celosía.

6. **Separación y Análisis de Fitness Seleccionados**:
   - Los elementos de la lista de mejores fitness se separarán en ítems individuales, según el número de generaciones.
   - Para identificar la posición de estos ítems en la lista global de fitness, se utilizará el componente member index.

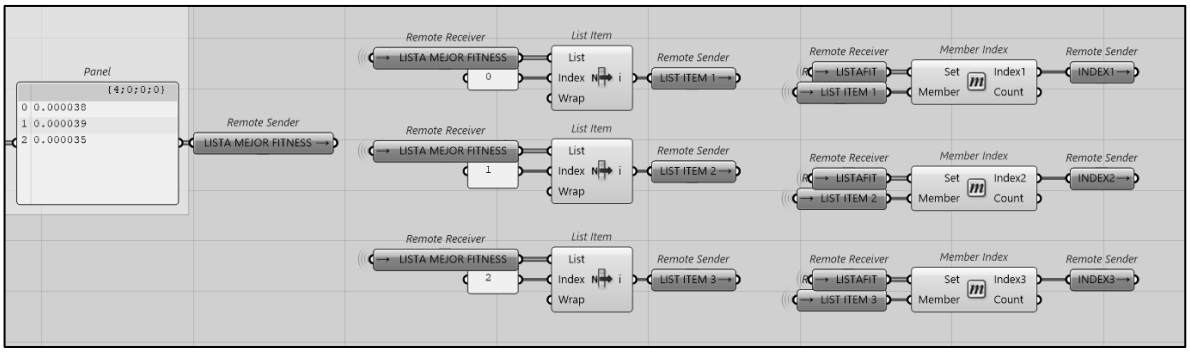


Ilustración 61. Separación y análisis de fitness seleccionados.

7. **Uso de Telepathy para la Selección de IPE y NDIV**:
   - El índice obtenido se enviará a varias listas usando telepathy, según el número de generaciones, y se separará en listas para IPE y NDIV de cada individuo.
   - El componente list item extraerá los valores deseados de las listas totales de perfiles grabados, utilizando el índice previamente obtenido.

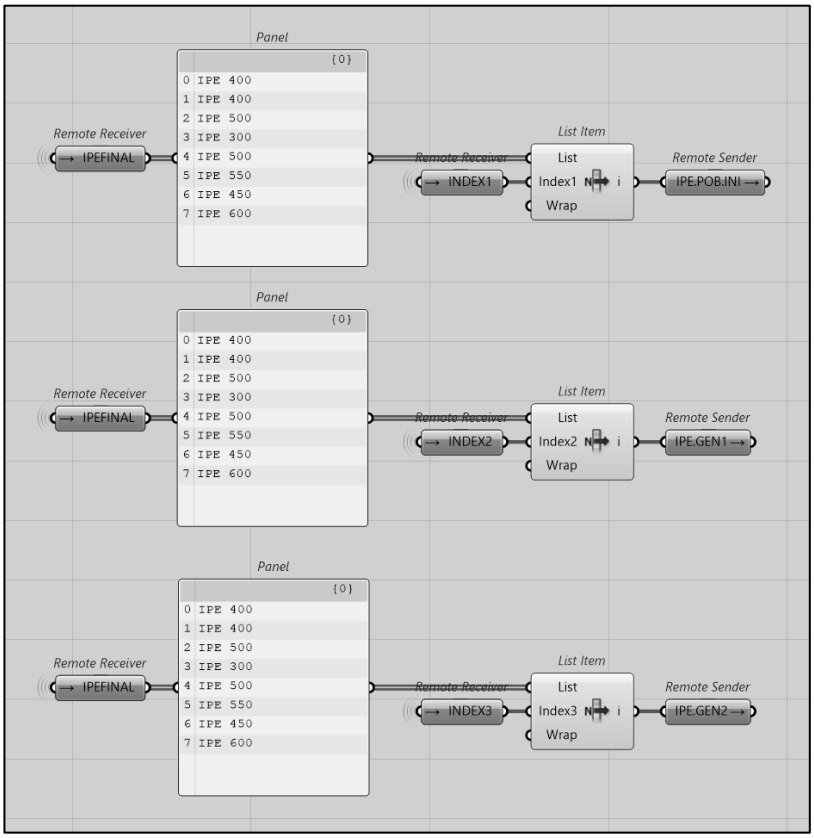


Ilustración 62. Selección IPE final celosía.

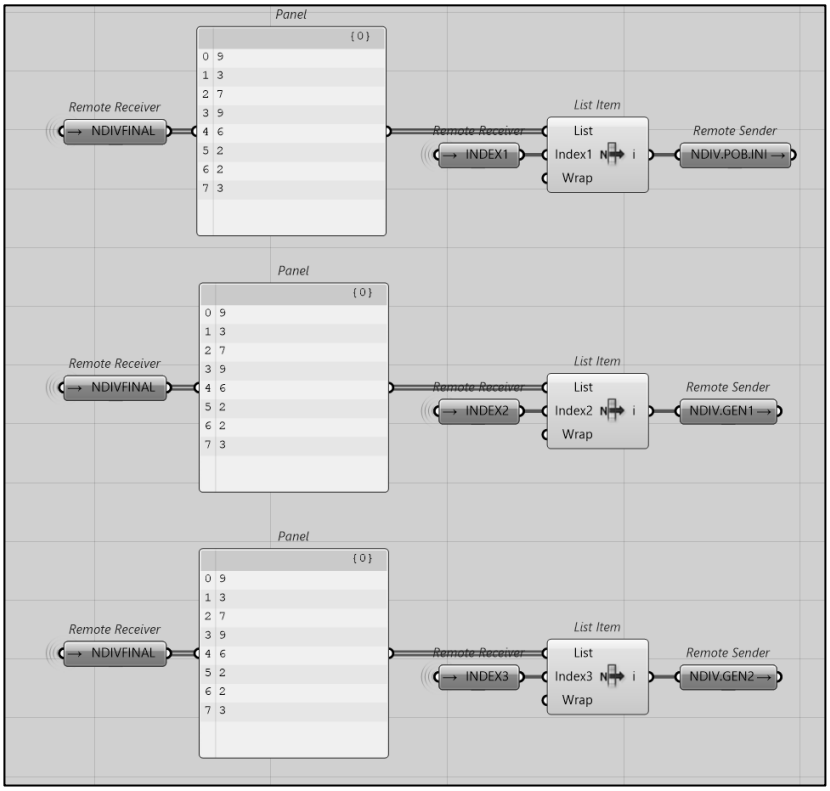


Ilustración 63. Selección NDIV final celosía.

8. **Agrupación de Datos**:
   - Los datos de salida del fitness, IPE y NDIV se agruparán en paneles para proporcionar una descripción completa de cada individuo.
   - Para crear un Hall of Fame, todos los datos se unificarán en una única lista usando los componentes flatten tree y partition list, permitiendo agrupar los elementos por generación.

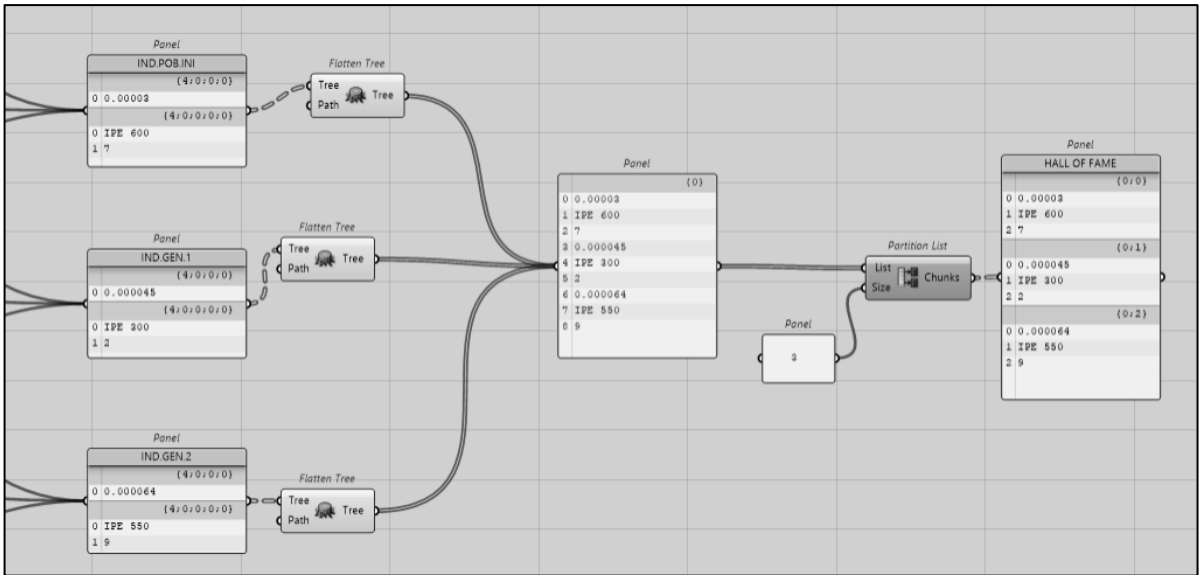


Ilustración 64. Hall of fame.

3.3.4. Caso 3: Optimización con operadores genéticos en la interfaz de Grasshopper.

Para finalizar la aplicación definitiva de optimización de esta celosía haremos una prueba previa de cómo desarrollar los operadores genéticos propios de este algoritmo en un módulo de Python, todo ello dentro de la interfaz gráfica de Grasshopper. Esto nos permitirá dejar las bases asentadas para únicamente replicar este módulo integrándolo en el modelo de la celosía, para luego proceder al cálculo definitivo de resultados y su correspondiente estudio analítico.

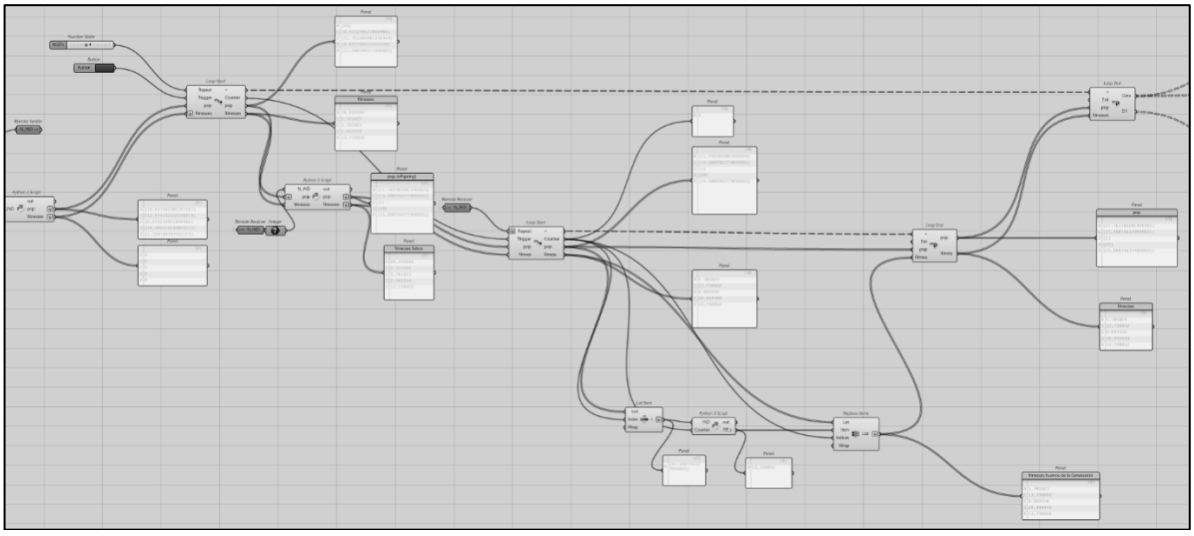


Ilustración 65. Vista general bucles operadores genéticos.

1. **<u>Bucle exterior.</u>**

Crearemos un bucle exterior en el que definiremos con el input “trigger” el número de generaciones de nuestro algoritmo genético mediante el uso de una “slider”, con esto conseguiremos variarlo de manera libre según las necesidades de cada problema. Además, como variables de entrada al módulo “Loop start”, añadiremos población y lista de valores de fitness, en el modelo final, esta población estará ligada a la ya creada (población inicial), mientras tanto, para esta prueba la definiremos como simples números aleatorios entre 1 y 20; al igual que los valores del fitness, los cuales llenaremos de ceros en una lista, correspondiente al número de individuos (en el modelo de la celosía ya quedan establecidos, en este podrán ser elegidos con una “slider”).

Colocaremos dos paneles a la salida de los outputs de las variables en el mismo módulo, para poder visualizar los movimientos y asegurarnos de que todo se ejecute correctamente. Anidado a estas dos variables, incorporaremos un módulo de Python usándolas como inputs de este, además del número de individuos. Este código implementa un algoritmo genético para optimizar la longitud de la barra de una estructura, maximizando así el valor de su flecha. Importaremos varios módulos de Python que facilitarán la incorporación de algoritmos genéticos y cálculos esenciales, luego asignamos los valores de fitness y otras características a cada uno de sus individuos.

Realizaremos las operaciones genéticas de mutación, selección por torneo y elitismo, definiendo parámetros esenciales como porcentajes de mutación y elitismo, así como probabilidad de mutación. La nueva generación de individuos se seleccionará y clonará,

aplicando mutaciones para finalmente, reemplazaremos la población actual con la nueva población optimizada. Este es un resumen de lo que realizará el código de Python creado para esta prueba, que será una gran base para el modelo original, con la diferencia de que el modelo avanzado integrará dos variables.

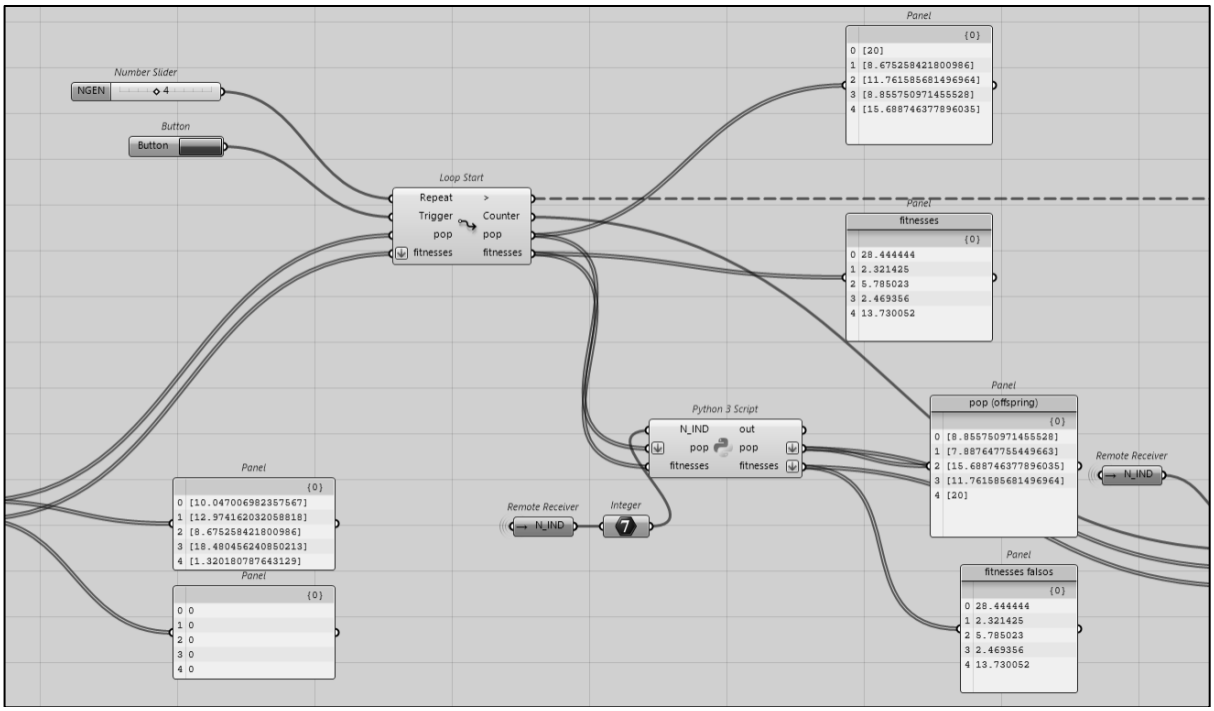


Ilustración 66. Bucle exterior prueba operadores genéticos.

A continuación, voy a proceder a la explicación detallada del módulo con el código de python que mostraré a continuación, ya que será la parte esencial de todo este caso:

1. Importación de Módulos.

2. Asignación de Fitness.
   - f_fitnesses: Convierte los valores de fitness en listas dentro de listas.
   - for ind, fit in zip(pop, f_fitnesses): Asigna los valores de fitness a cada individuo en la población.

3. Definición del Problema:
   - Optimización de la longitud de una barra para maximizar el valor de la flecha.

4. Parámetros del Individuo y Optimización:
   - LMIN y LMAX: Establecen los límites de los valores de los parámetros del individuo.
   - N_POB: Número de individuos en la población.

5. Creación del Problema e Individuo:
   - creator.create("Problema1", base.Fitness, weights=(1.0,)): Define el tipo de problema de optimización.
   - creator.create("Individuo", list, fitness=creator.Problema1): Define el tipo de individuo como una lista con un atributo de fitness.

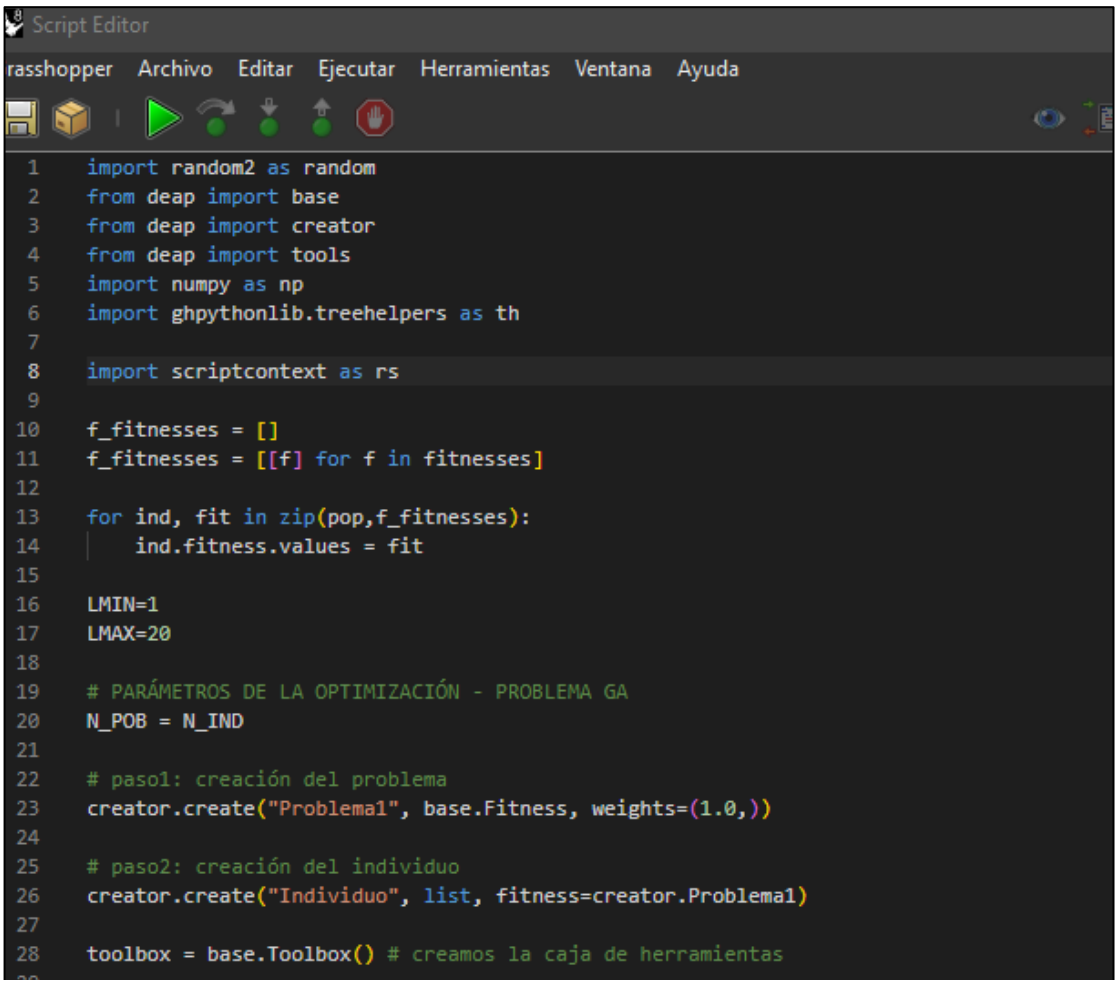


Ilustración 67. Código operadores genéticos (I).

A continuación, la definición de la siguiente fracción de código:

6. Caja de Herramientas y Operaciones Genéticas:
   - toolbox.register: Registra funciones genéticas para la mutación (mut_perc), selección por torneo (selTournament), y elitismo (selBest).

7. Parámetros de la Evolución:
   - mut_porc y elit_porc: Porcentajes de individuos mutados y elitistas.
   - MUTPB: Probabilidad de mutación.

8. Evolución a la Nueva Generación:
   - toolbox.select2 y toolbox.select: Selecciona los individuos para la nueva generación, aplicando elitismo y mutación.
   - toolbox.clone: Clona los individuos seleccionados.
   - Aplicación de Mutación: Mutación de los individuos con una probabilidad definida.
   - pop[:] = offspring: Reemplaza la población actual con la nueva generación.

```
# MUTACIÓN
toolbox.register("mutate", tools.mut_perc, PMIN=LMIN, PMAX=LMAX)
toolbox.register("select", tools.selTournament, tournsize=2) # torneo de dos en dos

# ELITISMO
toolbox.register("select2", tools.selBest) # seleccionamos los mejores

# REPARTO ELITISMO - MUTACIÓN
mut_porc = 0.8
elit_porc = 1-mut_porc

# PROBABILIDAD DE MUTACIÓN
MUTPB = 0.5

# EVOLUCIÓN A LA NUEVA GENERACIÓN - OFFSPRINGS

offspring_E = toolbox.select2(pop, int(len(pop)*elit_porc))
offspring_M = toolbox.select(pop,len(pop)-len(offspring_E))

offspring_E = list(map(toolbox.clone, offspring_E))
offspring_M = list(map(toolbox.clone, offspring_M))

# Aplicación de mutación
for mutant in offspring_M:
    if random.random() < MUTPB:
        toolbox.mutate(mutant)
        del mutant.fitness.values

offspring = offspring_E + offspring_M

pop[:] = offspring
```

Ilustración 68. Código operadores genéticos (II).

2. Bucle interior.

Añadiremos nuevamente un componente “Loop start”, pero en este caso usando el número de individuos como “Trigger”. Esto es lo que se repetirá en cada generación, vinculamos la población de salida del código anterior y los respectivos fitness a las variables de entrada del nuevo “Loop start”, en cuanto al fitness en el caso cero no tendrá valores, pero una vez arranquemos el modelo, dentro de este bucle interior se definirá el cálculo de este, haciendo así posible el funcionamiento del programa. Para el anidamiento de bucles procederemos igual que en el caso anterior, estableceremos conexión entre el output counter del bucle exterior y el input trigger del interior.

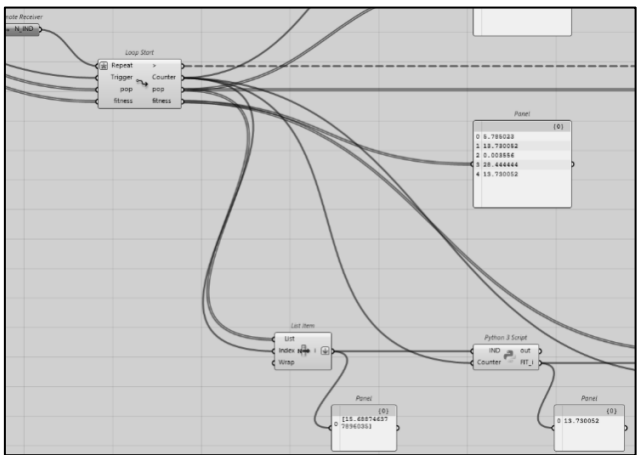


Ilustración 69. Bucle interior operadores genéticos.

En cuanto a las salidas, el objetivo será el cálculo y grabación de este fitness para cada individuo, por lo que con un componente "list item" seleccionaremos elementos de una lista uno a uno, usaremos la población importada del código anterior como input "list" y como índice usaremos el contador, para hacer la selección de los individuos correspondientes. Luego, unido a un código de Python cuyas variables de entrada serán: la salida de "list item" como IND (individuo); y contador. Se evaluará la función calculando individualmente los parámetros físicos que componen el cálculo de su flecha, y graba el resultado de este fitness en una variable de salida que usaremos a continuación.

```
5   P=1000 #kN
6   E=30000E3 #kN/m2
7   I=1/12*0.30*0.5**3 #Inercia sec. rect 30cm*50cm
8
9   def evaluate(individual):
10      fit = P*individual**3/(3*E*I)
11      return fit
12
13
14  FIT_i=evaluate(IND[0])
15
```

Ilustración 70. Cálculo fitness en bucle interior operadores genéticos.

Usamos de manera anidada a este código el componente "replace ítems", con el cual al darle como entrada la lista de fitness directamente del output de "Loop start" (aquella que viene del bucle exterior y que en el caso inicial es cero), como ítem la salida del fitness del código anterior que evalúa la función, y como índices la variable del contador para que los vaya adecuando a la correcta posición en una lista en relación con el individuo al que corresponden, obtendríamos la nueva lista de fitness necesaria para poder realizar una primera evaluación válida en la siguiente vuelta del bucle exterior. Mandándolo ya con esto al cierre del bucle interior como podemos ver a continuación.

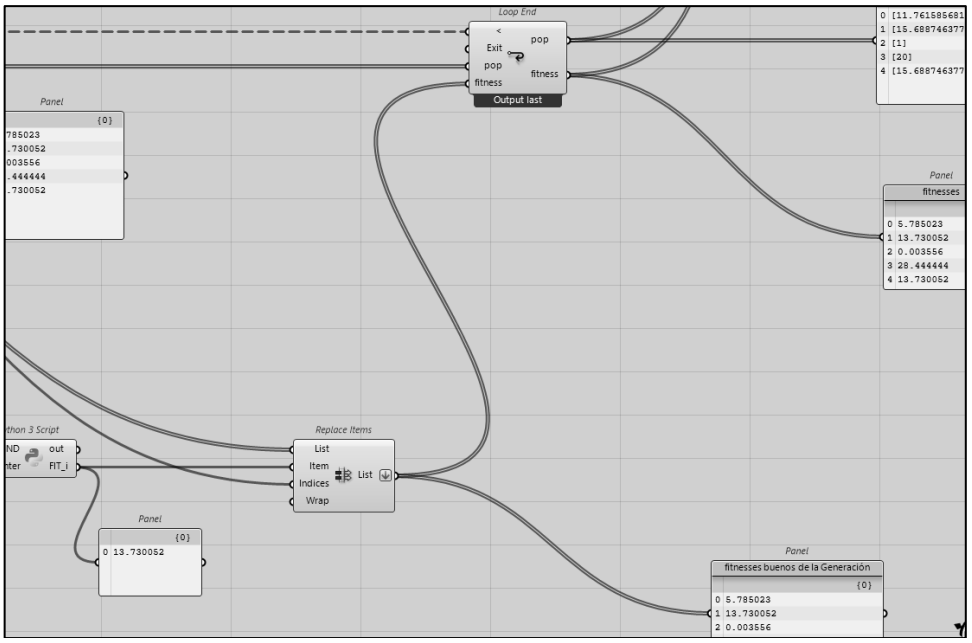


Ilustración 71. Fitness y cierre del bucle interno operadores genéticos.

3. Hall of fame, visualización de resultados.

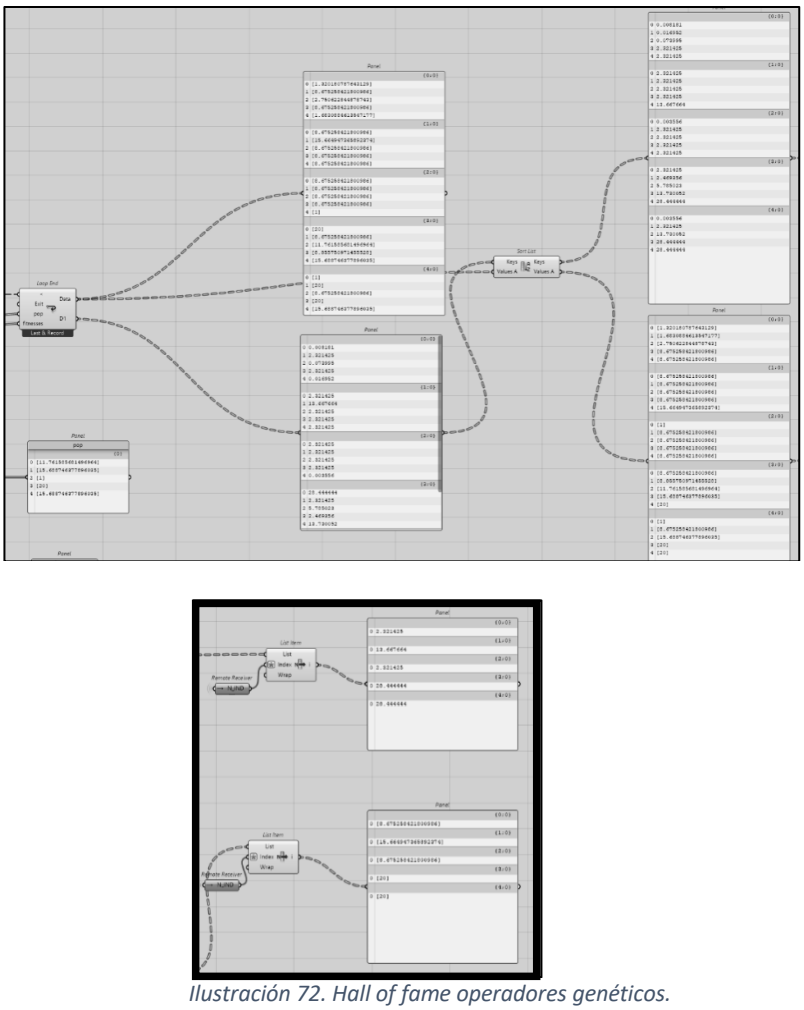


Ilustración 72. Hall of fame operadores genéticos.

Una vez cerrada de manera adecuada la anidación de estos dos bucles siguiendo el protocolo del caso anterior, procederemos a extraer los dos outputs del “Loop end” del bucle exterior como listas con las cuales trabajaremos. Por un lado, tendremos las longitudes de barra y por otro los fitness, es decir, la flecha máxima, ambas listas estarán divididas por ramas según la generación a la que correspondan, ordenaremos estas ramas de menor a mayor con el componente “Sort list”.

Una vez obtenidas las listas ordenadas extraeremos el ultimo componente de cada una de sus ramas usando “list item” y como índice el número total de individuos, ya que esto nos garantizará que nos da el ultimo valor, las grabamos en dos nuevas listas que corresponderán al llamado Hall of fame, lugar donde visualizar los mejores resultados de cada generación y a su vez el mejor resultado de todas las generaciones.

Este modelo nos asentará las bases para poder proceder a la adaptación de la viga del caso anterior, todo ello lo describiremos con más detalle en el siguiente apartado, donde de este modelo de una única variable hemos de ampliarlo a aquel que tiene dos, suponiendo esto un reto mayor. Además, la búsqueda de una integración eficiente será la clave para obtener aquello que se plantea como objetivo final.

3.3.5. Caso 4: Optimización estructural de la celosía del "Caso 2" pero con incorporación de operadores genéticos, obtención del modelo final.

En este cuarto caso del proyecto, combinaremos los conocimientos y técnicas de los dos casos anteriores realizando la ampliación de uno sobre el otro para poder finalmente crear una aplicación completa. Esta aplicación, mediante el uso de operadores genéticos, determinará cuales son las características más optimas que debe tener una celosía según el caso que se establezca. A continuación, detallaré el paso a paso necesario para llevar a cabo esta unificación y lograr una solución final completa y eficiente.

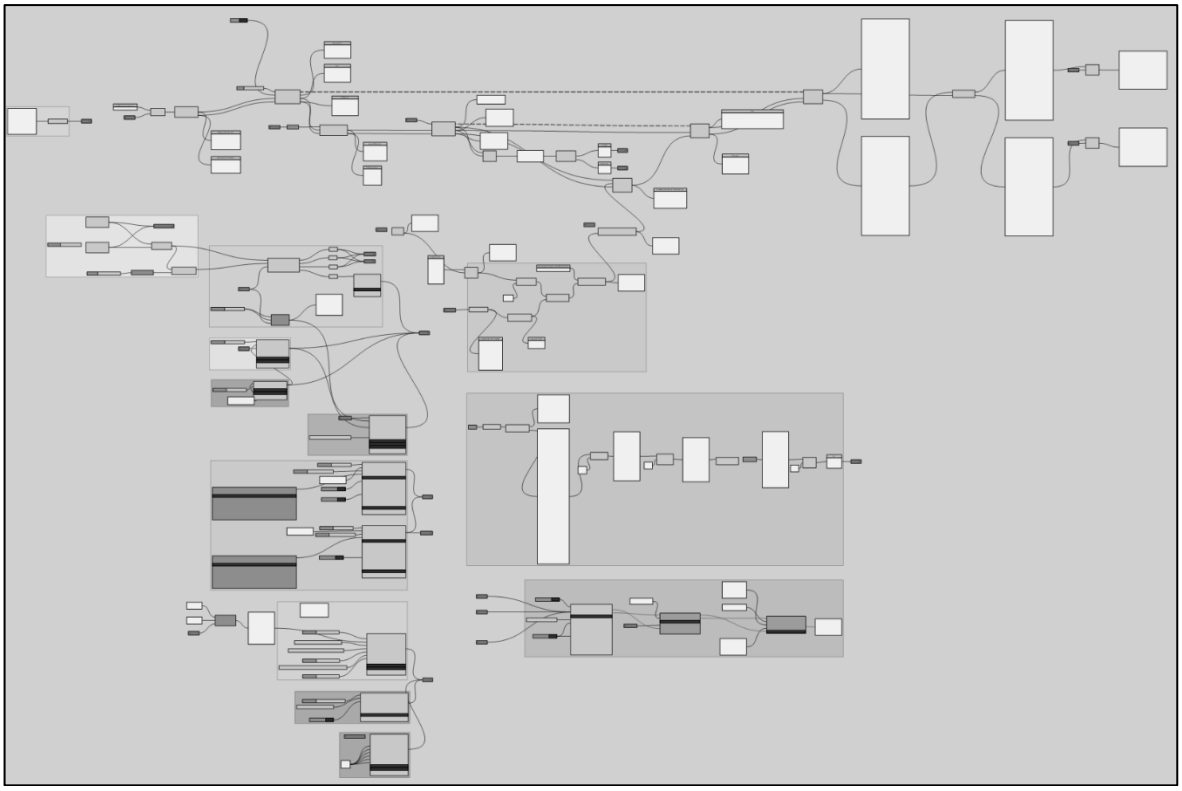


Ilustración 73. Aplicación final vista completa.

Para poder explicar el modelo de manera ordenada y comprensiva separaremos este en módulos a los cuales asignaremos el nombre de la función que cumplen, serán los siguientes:

- Generación de población inicial y lista de fitness.
- Iniciación del bucle exterior e implementación de operadores genéticos.
- Restablecimiento del modelo anterior para la nueva integración en el actual.
- Bucle interior, nueva función de fitness y vinculación al programa.
- Cierre de bucles para la exportación y visualización de resultados en el Hall of Fame.

1. Generación de la población inicial y lista de fitness.

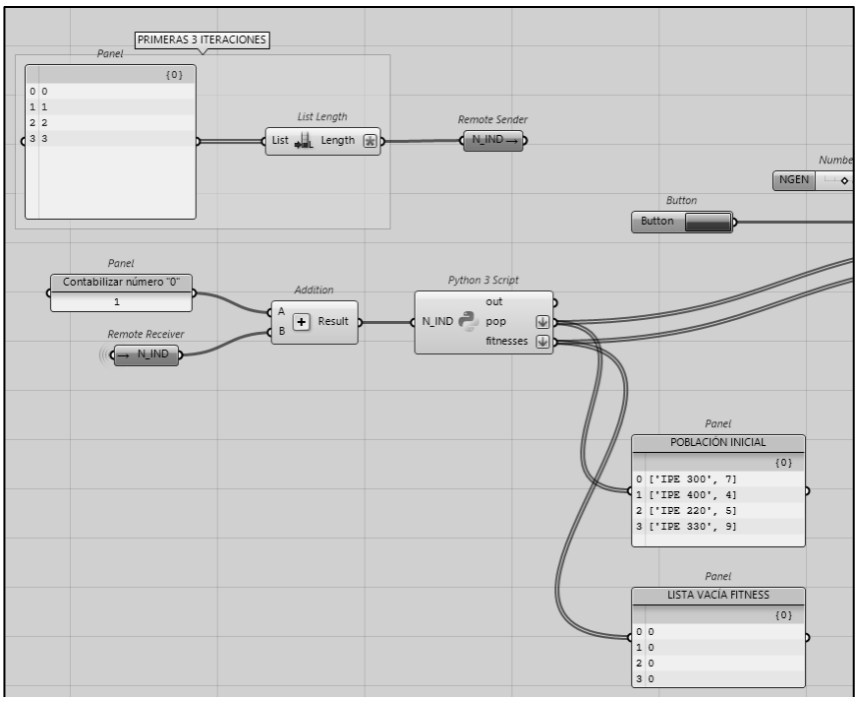


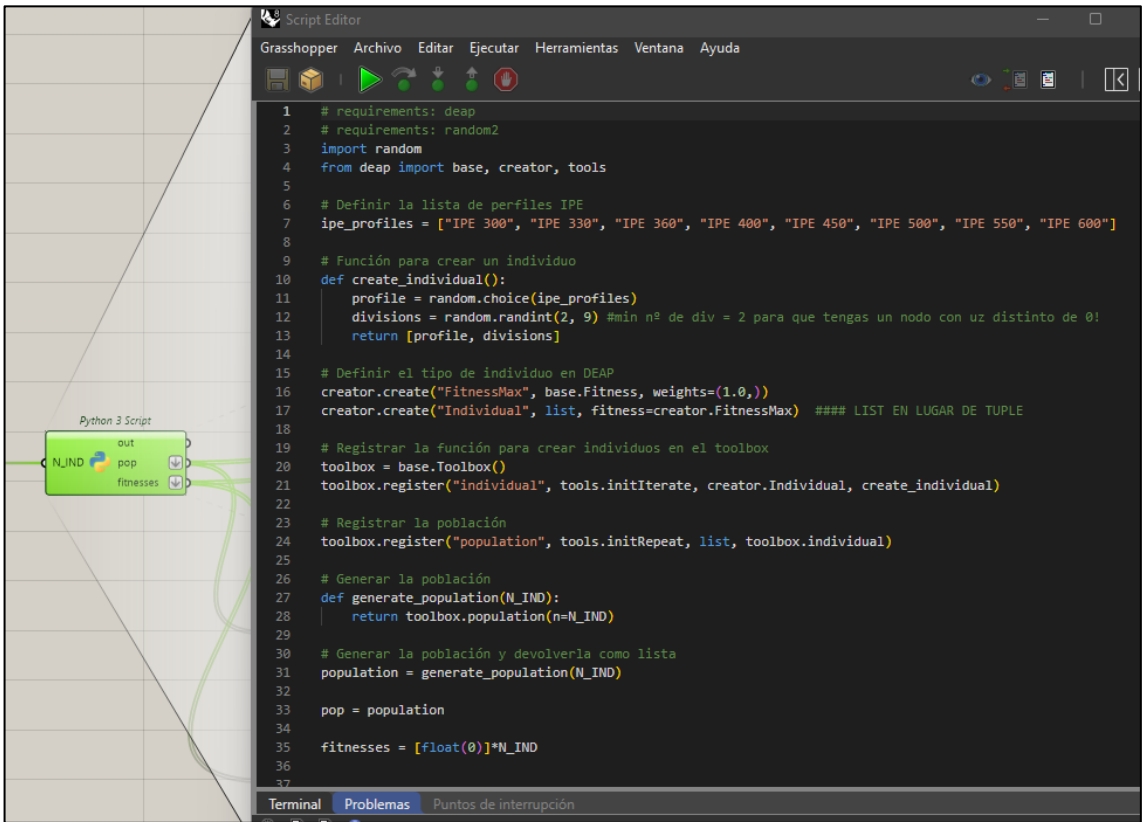
Ilustración 74. Generación de la población inicial y lista de fitness.

Pese a tener en el caso 2 un bucle que nos permitía obtener la población inicial aleatoria, vamos a reestablecer los ajustes para poder llevar a cabo este último caso. Para generar esta población inicial mediante código, primero hemos de establecer el número de individuos que queremos que tenga nuestro caso, este, permitiremos que se modifique de manera libre según el caso que queramos definir, es por esto por lo que usaremos el siguiente método.

- Añadiremos un panel: en él podremos escribir tantos números como deseemos separándolos por renglón, estos determinarán el número de individuos.
- Luego el módulo "List Length": nos devolverá un numero entero que podremos usar para la incorporación en el módulo de python.
- Antes de ello, hemos de tener en cuenta que list length al devolver el resultado no tendrá en cuenta el "0" como número cuando realmente sí que cuenta como individuo más, es por eso que añadiremos de la pestaña "Maths" el módulo "addition" con un panel que marque un "1", para así contabilizar el "0" y de este modo tener el número real.

- EJEMPLO: si nosotros queremos que sean 3 individuos, escribiremos en el panel inicial "0,1,2" separando por líneas. Luego List Length devolverá "2" y con la suma conseguiremos que al módulo de Python llegué "3" siendo este el número real de individuos que queremos en el modelo.

- Ahora, generaremos un código de Python cuya función principal será generar tantos individuos aleatorios como le hayamos indicado de entrada, además de una lista de "0" que tendrá tanta longitud como individuos le hayamos dicho que tiene.
  Por otro lado, aprovecharemos este módulo para definir el tipo de individuos y registrarlos a ellos y a la población en el módulo Deap (de algoritmos genéticos) ya que así nos facilitará vinculación posterior a los operadores genéticos.




```python
# requirements: deap
# requirements: random2
import random
from deap import base, creator, tools

# Definir la lista de perfiles IPE
ipe_profiles = ["IPE 300", "IPE 330", "IPE 360", "IPE 400", "IPE 450", "IPE 500", "IPE 550", "IPE 600"]

# Función para crear un individuo
def create_individual():
    profile = random.choice(ipe_profiles)
    divisions = random.randint(2, 9) #min nº de div = 2 para que tengas un nodo con uz distinto de 0!
    return [profile, divisions]

# Definir el tipo de individuo en DEAP
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)  #### LIST EN LUGAR DE TUPLE

# Registrar la función para crear individuos en el toolbox
toolbox = base.Toolbox()
toolbox.register("individual", tools.initIterate, creator.Individual, create_individual)

# Registrar la población
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

# Generar la población
def generate_population(N_IND):
    return toolbox.population(n=N_IND)

# Generar la población y devolverla como lista
population = generate_population(N_IND)

pop = population

fitnesses = [float(0)]*N_IND
```

Ilustración 75. Código para generar población y lista de fitness vacía.

- Importaciones y Configuración Inicial.
  Importa las bibliotecas necesarias: *random* para la generación de números aleatorios y *deap* para los operadores genéticos.
- Definición de Perfiles IPE y Función para Crear Individuos.
  Define la lista de perfiles IPE disponibles. Define la función *create_individual* que crea un individuo seleccionando un perfil aleatorio de *ipe_profiles* y un número aleatorio de divisiones entre 2 y 9.
- Definición de la Estructura del Individuo en DEAP.
  Utiliza el módulo *creator* para definir nuevos tipos de objetos. La función *creator.create* define un nuevo tipo de fitness llamado *FitnessMax* con un atributo para maximizar (peso 1.0). Define también un nuevo tipo de individuo llamado *Individual* como una lista que incluye un atributo de fitness *FitnessMax.*
- Registro de Funciones en el Toolbox de DEAP.
  Crea una caja de herramientas *(toolbox)* para registrar funciones. Registraremos una función que crea un nuevo individuo usando *create_individual. tools.initIterate* es una función que se utiliza para crear un depósito iterando sobre una función generadora. Registra una función p*opulation* que crea una población de individuos.
- Generación de la Población.
  Define una función *generate_population* que crea una población de tamaño N_IND utilizando la función registrada *population.* Genera una población de individuos de tamaño N_IND y la almacena en *population.* Asigna esta población a pop. Inicializa una lista *fitnesses* con N_IND elementos, todos establecidos a 0, para almacenar los valores de *fitness* de la población.

2. Inicialización del bucle exterior e implementación de operadores genéticos.

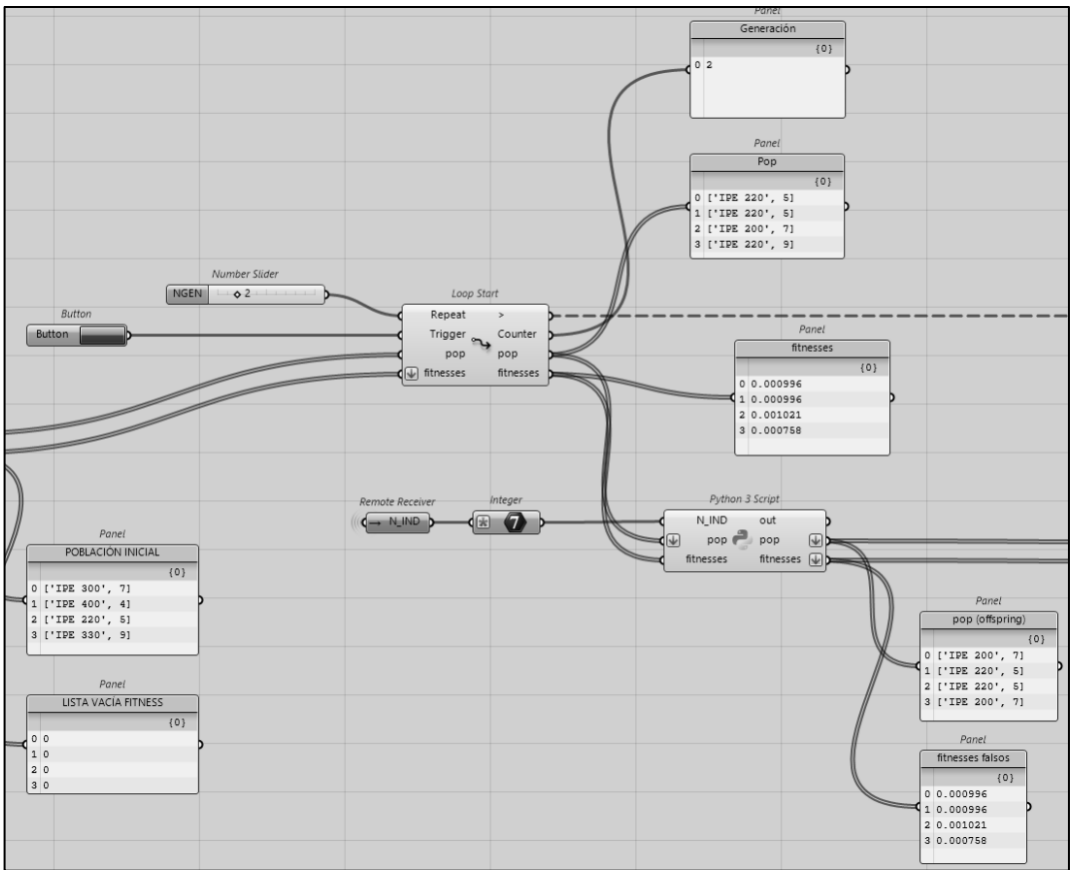
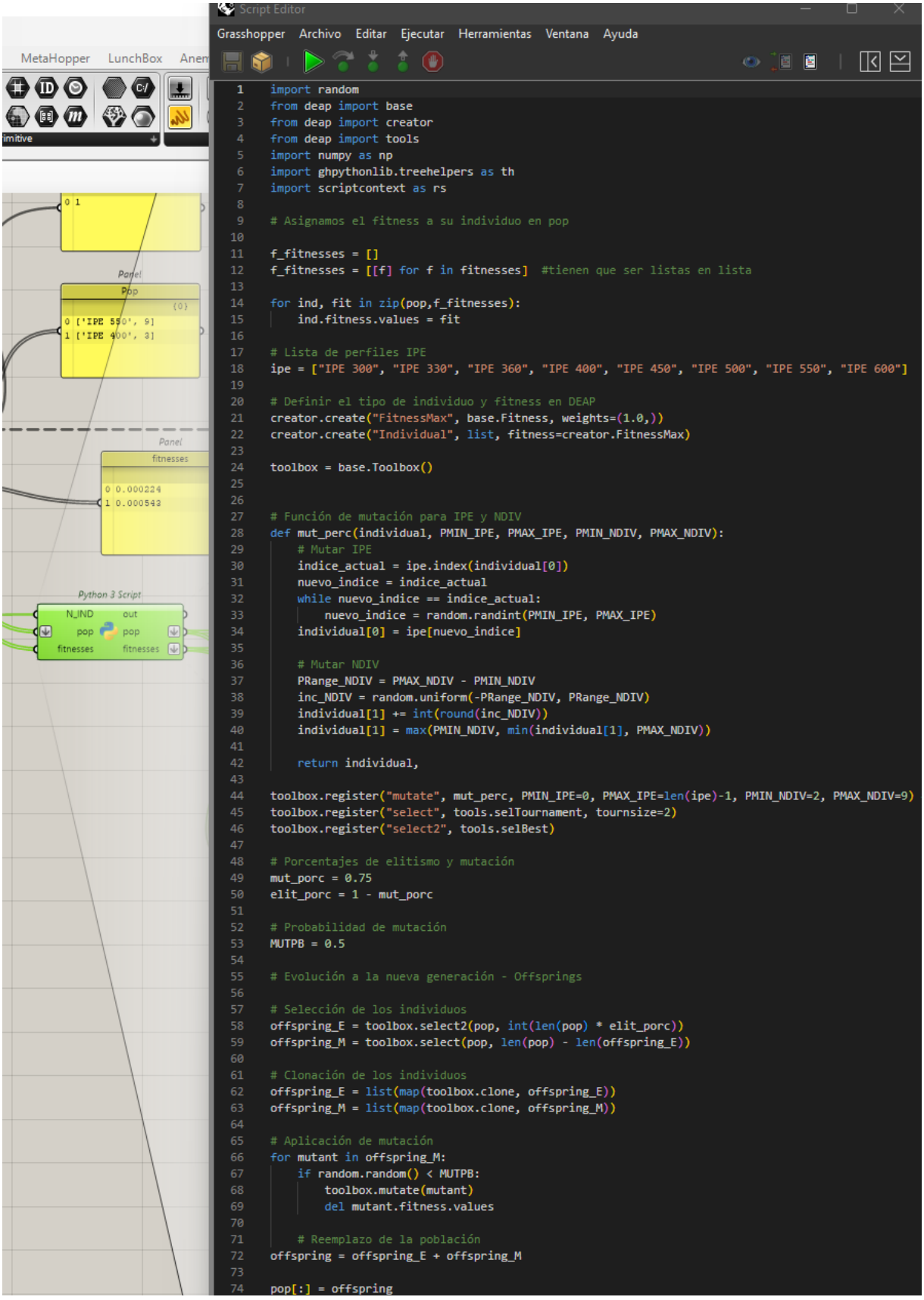


Ilustración 76. Inicialización del bucle exterior e implementación de operadores genéticos.

Iniciaremos con un módulo “Loop Start” de Anemone al que conectaremos como input “pop” la población inicial generada en el módulo anterior y como input “fitnesses” la lista de “0” también generada anteriormente. Luego “Trigger” será para reiniciar la aplicación y que vuelva a ejecutarse por lo que usaremos el componente “button”. Finalmente “Repeat”, determinará el número de generaciones que queremos que tenga nuestro modelo, usaremos una slider con números de 0-10 ya que lo podremos variar a nuestro antojo para así poder realizar diversas pruebas.

Los paneles usados como outputs son simplemente un medio para poder visualizar lo que está sucediendo en nuestro modelo una vez que lo ponemos a correr, y los outputs de “pop” y “fitnesses” serán las entradas de un nuevo módulo de Python donde se encontrará el corazón del modelo, donde aplicaremos estos operadores genéticos.

Este código utilizará la biblioteca DEAP para la evolución de una población de individuos, donde cada individuo representa un perfil IPE y un número de divisiones, luego definirá la estructura y los operadores genéticos necesarios. En este caso vamos a realizar mutaciones, selección (elitismo) y reemplazo de la población, con el objetivo de maximizar una función de fitness que se definirá en procesos posteriores. Además, el proceso incluirá la clonación de los individuos seleccionados y reemplazará el resto de la población antigua por una nueva. Como entrada al módulo, nos traemos el valor nominal de N_IND con el módulo telepathy.



Script Editor

Grasshopper Archivo Editar Ejecutar Herramientas Ventana Ayuda

```python
import random
from deap import base
from deap import creator
from deap import tools
import numpy as np
import ghpythonlib.treehelpers as th
import scriptcontext as rs

# Asignamos el fitness a su individuo en pop

f_fitnesses = []
f_fitnesses = [[f] for f in fitnesses]  #tienen que ser listas en lista

for ind, fit in zip(pop,f_fitnesses):
    ind.fitness.values = fit

# Lista de perfiles IPE
ipe = ["IPE 300", "IPE 330", "IPE 360", "IPE 400", "IPE 450", "IPE 500", "IPE 550", "IPE 600"]

# Definir el tipo de individuo y fitness en DEAP
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)

toolbox = base.Toolbox()


# Función de mutación para IPE y NDIV
def mut_perc(individual, PMIN_IPE, PMAX_IPE, PMIN_NDIV, PMAX_NDIV):
    # Mutar IPE
    indice_actual = ipe.index(individual[0])
    nuevo_indice = indice_actual
    while nuevo_indice == indice_actual:
        nuevo_indice = random.randint(PMIN_IPE, PMAX_IPE)
    individual[0] = ipe[nuevo_indice]

    # Mutar NDIV
    PRange_NDIV = PMAX_NDIV - PMIN_NDIV
    inc_NDIV = random.uniform(-PRange_NDIV, PRange_NDIV)
    individual[1] += int(round(inc_NDIV))
    individual[1] = max(PMIN_NDIV, min(individual[1], PMAX_NDIV))

    return individual,

toolbox.register("mutate", mut_perc, PMIN_IPE=0, PMAX_IPE=len(ipe)-1, PMIN_NDIV=2, PMAX_NDIV=9)
toolbox.register("select", tools.selTournament, tournsize=2)
toolbox.register("select2", tools.selBest)

# Porcentajes de elitismo y mutación
mut_porc = 0.75
elit_porc = 1 - mut_porc

# Probabilidad de mutación
MUTPB = 0.5

# Evolución a la nueva generación - Offsprings

# Selección de los individuos
offspring_E = toolbox.select2(pop, int(len(pop) * elit_porc))
offspring_M = toolbox.select(pop, len(pop) - len(offspring_E))

# Clonación de los individuos
offspring_E = list(map(toolbox.clone, offspring_E))
offspring_M = list(map(toolbox.clone, offspring_M))

# Aplicación de mutación
for mutant in offspring_M:
    if random.random() < MUTPB:
        toolbox.mutate(mutant)
        del mutant.fitness.values

    # Reemplazo de la población
offspring = offspring_E + offspring_M

pop[:] = offspring
```

Ilustración 77. Código de aplicación de operadores genéticos y reemplazo de individuos.

- Importación de Módulos.
  Importa los módulos para la manipulación de datos aleatorios (*random*), las funcionalidades de *DEAP* (*base, creator, tools*), y otras librerías personalizadas (*phpythonlib.treehelpers, scriptcontext).*
- Asignación de fitness a los Individuos.
  Este bloque de código asigna los valores de fitness a los individuos de la población *pop* en *DEAP*. Primero, se crea una lista vacía *f_fitnesses* para almacenar los valores de aptitud ajustados. Luego, cada valor en la lista original *fitnesses* se convierte en una lista anidada dentro de *f_fitnesses*, cumpliendo con el formato requerido por *DEAP.* Después, mediante un *bucle for y zip(),* se asigna a cada individuo en *pop* su respectivo valor de aptitud de *f_fitnesses*, asegurando que esté correctamente estructurado como una lista.
- Lista de Perfiles IPE.
  Define una lista de perfiles IPE, que representan diferentes tamaños de secciones de acero estructural.
- Definición de Tipos de Individuos y Fitness en DEAP.
  Define el tipo de fitness (*FitnessMax)* para maximizar la aptitud y el tipo de individuo (*Individual*) que es una lista con un atributo de fitness asociado.
- Función de Mutación.
  Esta función *mut_perc* realiza mutaciones en un individuo específico, ajustando dos atributos*: IPE y NDIV*. Primero, muta el atributo *IPE* seleccionando aleatoriamente un nuevo índice para la lista *ipe*. Luego, muta el atributo *NDIV* mediante un incremento aleatorio dentro de un rango definido por *PMIN_NDIV* y *PMAX_NDIV*, asegurándose de que el valor resultante esté dentro de esos límites, establecemos "2" como límite inferior ya que este atributo se está refiriendo al número de divisiones dentro de la celosía, siendo 0 y 1 imposibles para obtener un cálculo coherente. Finalmente, devuelve el individuo modificado.
- Registro de la Función de Mutación y Selección en el Toolbox.
  Las funciones registradas en *toolbox* son necesarias para configurar las operaciones fundamentales en un algoritmo genético. Registra la función de mutación (*mut_perc*) y las funciones de selección (*tools.selTournament y tools.selBest*) en el *toolbox*.
- Porcentajes de Elitismo y Mutación.
  Define los porcentajes de individuos que se seleccionarán para mutación y elitismo, dándole a mutación un porcentaje mayor debido que si elitismo es demasiado grande, siendo nuestro caso de pruebas con "pocos" individuos relativamente (0-10) podría darse el caso de que se acabasen seleccionando demasiados por generación y no obtener un análisis bueno ni un buen uso del programa.
- Probabilidad de Mutación.
  Define la probabilidad de que un individuo sea mutado.
- Evolución a la Nueva Generación.
  - Selecciona los individuos elitistas *(offspring_E)* y los individuos para mutación *(offspring_M)* de la población.
  - Clona los individuos seleccionados para crear nuevas copias que se modificarán posteriormente.
  - Aplica la mutación a los individuos seleccionados para mutación con una cierta probabilidad. Elimina el valor de fitness antiguo para recalcularlo después.
  - Combina los individuos elitistas y mutados para formar la nueva población y reemplaza la población antigua con esta nueva generación.

3. Restablecimiento del modelo anterior para la nueva integración en el actual.

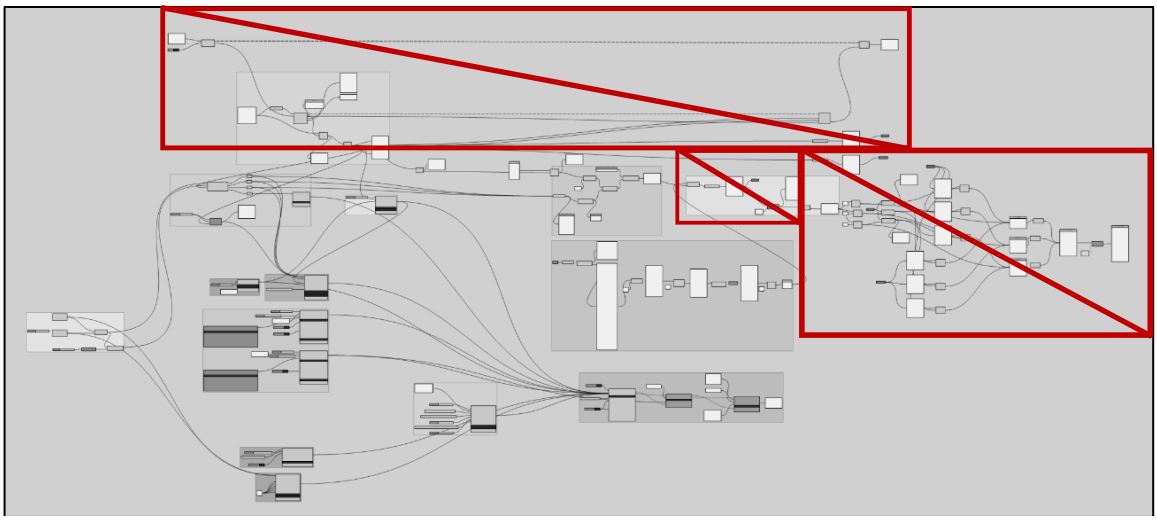


Ilustración 78. Partes que se eliminarán respecto al Caso 2.

Para esta aplicación final, hemos de suprimir los bucles realizados en el caso anterior, además del análisis de sus resultados. Estos se verán remplazados por los nuevos bucles que estamos creando y detallando ahora, siendo únicamente necesaria la creación de la geometría con sus respectivas especificaciones, casos de carga, creación de miembros, proceso de exportación a RFEM y proceso de extracción de datos. Conservamos la extracción de la deformación vertical Uz del archivo CSV extraído de RFEM y el cálculo del peso propio.

Por otro lado, reestableceremos la función de fitness a una nueva ajustada a la precisión requerida por este modelo, por tanto, también eliminamos la establecida en el modelo anterior.

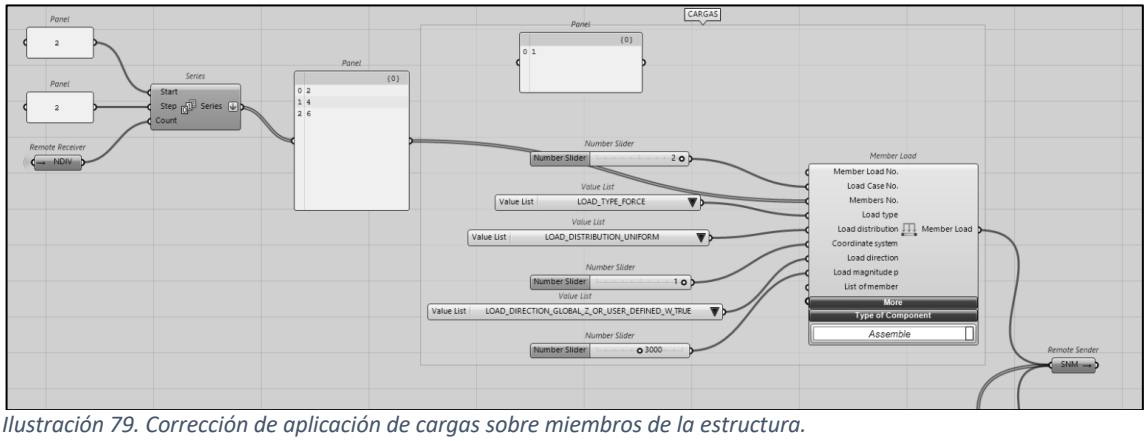


Ilustración 79. Corrección de aplicación de cargas sobre miembros de la estructura.

Además, corregiremos la distribución de cargas sobre los miembros ya que si no es de este modo no se reparte equitativamente entre las barras superiores de la celosía. Analizando en RFEM, vemos que las vigas superiores de la estructura siempre cuentan con los primeros números pares de la secuencia, pese a la variación en su número de divisiones. Entonces, usamos el módulo “series” para crear una lista que comenzando en el Nº2 y saltando de 2 en 2, genere tantos números pares como divisiones tenga la celosía. Y tras varias pruebas, se confirma que de este modo estamos permitiendo la correcta distribución de cargas a diferencia de lo sucedido en modelos previos.

4. Bucle interior, nueva función de fitness y vinculación al programa.

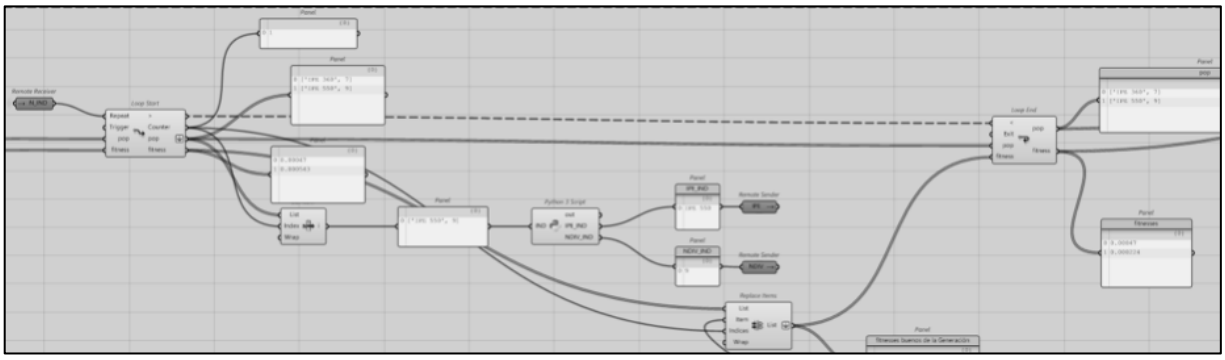


Ilustración 80. Vista bucle interior.

Añadiremos nuevamente el componente “Loop Start”, y traeremos como input las salidas “pop” y “fitnesses” del módulo de Python anterior, en este caso, no necesitaremos botón “trigger” ya que su inicio estará ligado a la ejecución del bucle superior. En cuanto a “Repeat” le pediremos que lo haga según el N_IND definido previamente.

Con este bucle queremos obtener los fitness reales de cada uno de los individuos obtenidos previamente, por tanto, el objetivo es que separe individuo a individuo y meta cada una de sus características (IPE y NDIV) al modelo de generación de geometría para que calcule la estructura, luego extraiga los parámetros y así calcular su fitness, luego grabarlo e iniciar el proceso para el siguiente individuo.

- Añadiremos un componente “List Item”: la lista de entrada al componente la proporcionará el output “pop” del loop start del bucle interno, y el índice será el output “Counter” de este mismo, ya que nos permitirá ir recorriendo todos los individuos de la población y grabando los fitness en esa posición.
- Una vez obtenemos el individuo, usaremos un breve código de Python para poder separar sus componentes y tener por un lado IPE y por otro NDIV, si no, no podemos integrarlas en nuestro modelo donde se usan por separado en distintos lugares, tenemos que usar telepathy en las dos salidas de este módulo y usarlo de entrada en los lugares correspondientes del modelo.
  Usaremos a biblioteca *ast*, la cual permite analizar el código Python para separar sus componentes. Si tenemos varias declaraciones en una entrada usamos *ast* para identificar y extraer componentes específicos del código.

Ilustración 81. Separar componentes del individuo.

- Luego usaremos el componente “replace ítems”: nos servirá para sustituir la lista de “fitnesses” establecida por el módulo anterior al nuevo bucle, para ello, tendremos que vincular esta lista antigua que será el output “fitnesses” establecido en la salida de “Loop Start” al input “List” del componente. Como índices usaremos el contador de nuevo, que nos permitirá grabar el resultado en la posición adecuada en relación al individuo que le corresponde.
- Nueva función de fitness: esta vez multiplicaremos el peso por la deformación producida (a su vez esta deformación la multiplicaremos por 1000 para ajustar un poco la relación de sus órdenes de magnitud), pero como nuestro objetivo es obtener como mejor resultado aquel que tenga mayor fitness, hemos de dividir 1 entre todo esto, si lo dejásemos tal cual sería válido para una función de minimizar, es por eso por lo que hacemos este cambio.
  La vincularemos por tanto como output “Item” al complemento mencionado anteriormente.

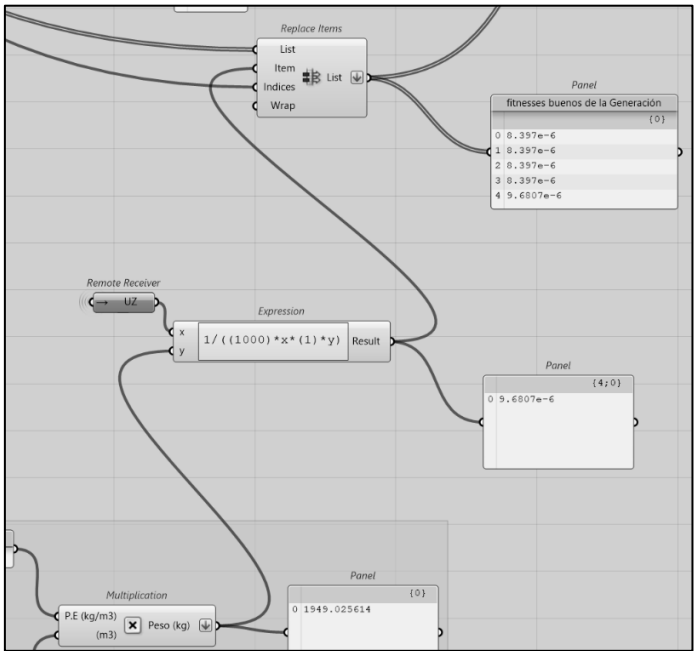


Ilustración 82. Nueva definición de fitness y grabación del mismo en una lista.

5. Cierre de bucles para la exportación y visualización de resultados en el Hall of Fame.

- Para el bucle interior tan solo incorporaremos un componente “Loop end” donde vinculamos con “Loop Start” el método de relación entre ambos “><”, y además la lista “pop” ya que esta no se ha visto modificada durante el bucle, importamos la misma que a la entrada. Luego, vinculamos la nueva lista con los fitnesses actualizados después del cálculo realizado en este modelo como salida a este loop.

- Para el bucle externo repetiremos exactamente el mismo proceso, solo que únicamente hemos de vincular los outputs “pop” y “fitnesses” del loop interno a los inputs de este externo, todos los datos se transferirán automáticamente y volverá a reiniciarse si es que aún quedan generaciones por evaluar. Una vez finalicen de calcularse todos los fitness de los individuos de la última generación, el bucle pasará a cerrarse.

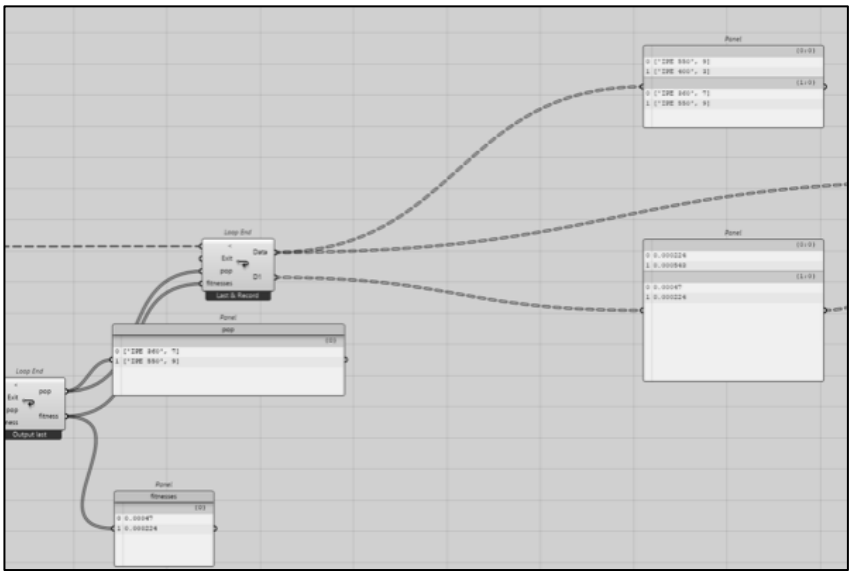


Ilustración 83. Cierre de los bucles interno y externo.

- Por último la grabación de resultados y el Hall of Fame, esto consistirá en que se irán grabando tanto los individuos como los fitness de cada generación y mediante el componente "Sort List" podremos ordenar ambas listas de menor a mayor.

- Como los que nos interesa serán los individuos con mayor fitness de cada generación, usaremos otro "list item" para extraer el elemento con el index "N_IND" que en el momento, deberá estar ubicado en el último elemento de la lista habiendo finalizado todos los bucles. Gracias a ello podremos visualizar en un nuevo panel las soluciones finales que serán las más óptimas y cuyo análisis tendrá lugar en el siguiente apartado del proyecto.

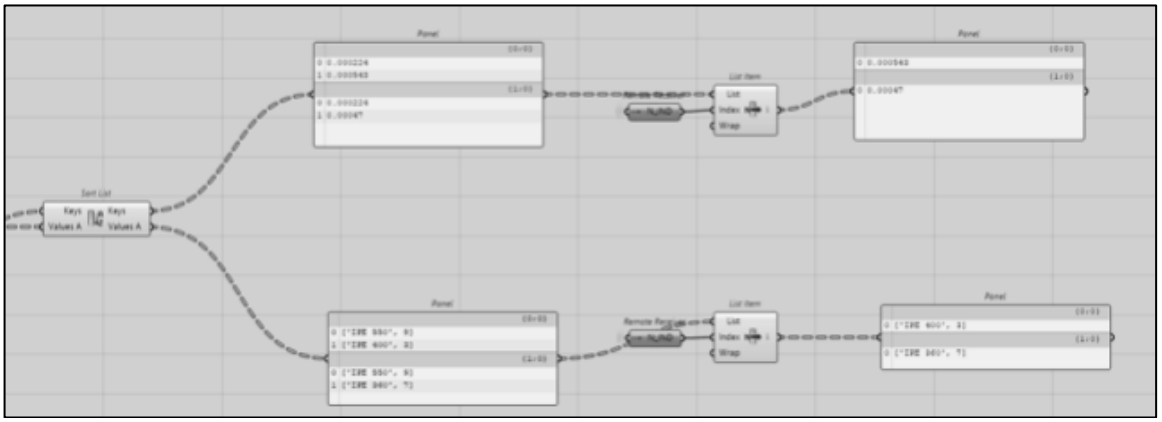


Ilustración 84. Hall of fame definitivo.

Damos por concluida la creación del modelo, habiendo verificado su funcionamiento con multitud de pruebas que serán el objeto de análisis del próximo capítulo.

4. Resultados

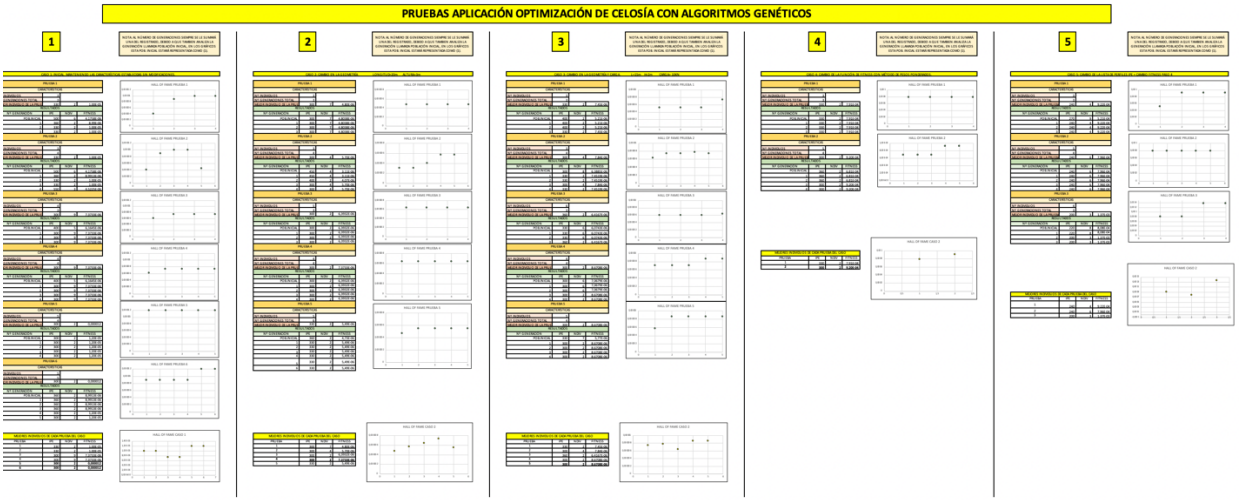


Ilustración 85. Vista general de la toma de pruebas del modelo final.

4.1. Resultados obtenidos y análisis descriptivo.

Para mostrar y analizar resultados ejecutaremos el modelo en numerosas pruebas, realizando cambios en la aplicación entre los distintos casos que plantearé. En algunos, se modificará la geometría del modelo, en otros, la carga aplicada, la función de fitness e incluso combinaciones de entre estos factores. También variaremos los valores de la lista de perfiles IPE.

Este enfoque nos permitirá analizar como cada uno de estos cambios impacta en los resultados obtenidos y determinar las configuraciones óptimas para nuestro objetivo final. A continuación, se presentan los resultados obtenidos y los análisis de cada caso.

4.1.1. Caso 1. Modelo original, sin modificaciones.

En este primer caso, hemos mantenido todo el modelo igual que lo descrito en la guía del usuario anterior. Hemos ejecutado el modelo seis veces para realizar las distintas pruebas, variando el número de generaciones y el número de individuos por generación en cada prueba.

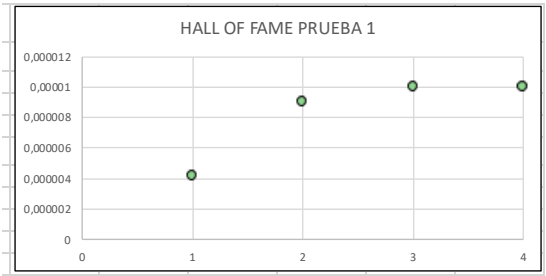
Para cada generación, se registra el mejor individuo obtenido y se crea un gráfico que visualmente muestra la calidad de los individuos obtenidos.

Además, he extraído el mejor individuo de cada prueba y realizado una recolección total, presentándola en un gráfico que muestra las distintas mejores soluciones. Esto nos permite hacernos una idea del resultado óptimo y analizar si un mayor número de generaciones e individuos conduce a mejores resultados, o si ocurre lo contrario.

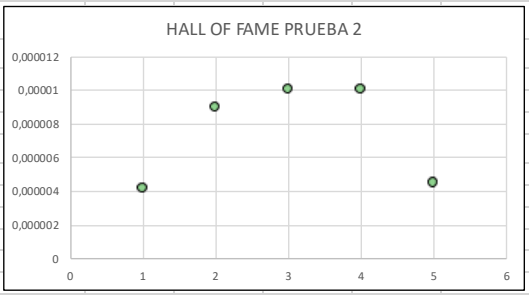
Por otro lado, mostraré también algunas de las capturas de resultados extraídas de los modelos calculados para así dar constancia de la veracidad de estos datos, y se expondrán videos de algunos casos también en la defensa.

CASO 1: INICIAL, MANTENIENDO LAS CARACTERÍSTICAS ESTABLECIDAS SIN MODIFICACIONES.

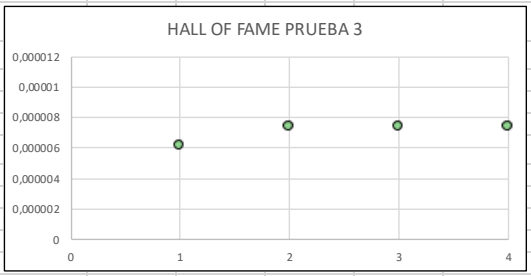
PRUEBA 1			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	330	2	1,00E-05
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	500	6	4,1758E-06
1	360	2	8,99E-06
2	330	2	1,00E-05
3	330	2	1,00E-05



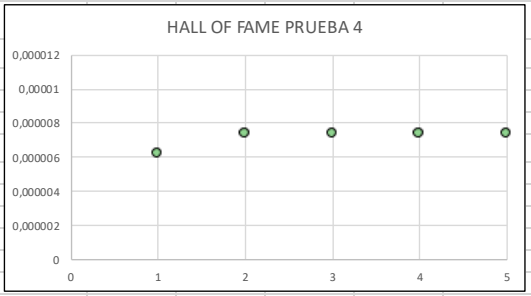
PRUEBA 2			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	330	2	1,00E-05
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	500	6	4,1758E-06
1	360	2	8,9913E-06
2	330	2	1,00E-05
3	330	2	1,00E-05
4	550	3	4,5225E-06



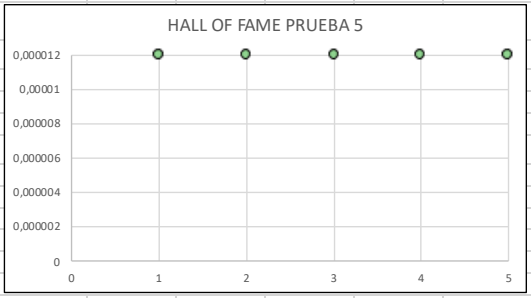
PRUEBA 3			
CARACTERÍSTICAS			
Nº INDIVIDUOS	4		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	300	9	7,3733E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	400	5	6,1645E-06
1	300	9	7,3733E-06
2	300	9	7,3733E-06
3	300	9	7,3733E-06



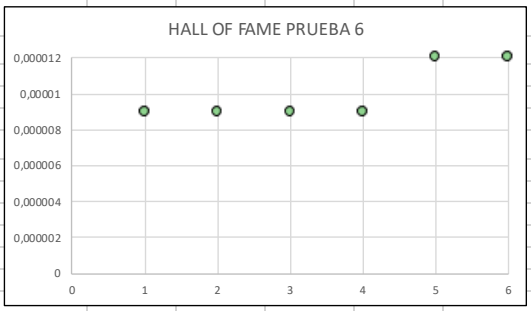
PRUEBA 4			
CARACTERÍSTICAS			
Nº INDIVIDUOS	4		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	9	7,3733E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	400	5	6,1645E-06
1	300	9	7,3733E-06
2	300	9	7,3733E-06
3	300	9	7,3733E-06
4	300	9	7,3733E-06



PRUEBA 5			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	0,000012
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	300	2	1,20E-05
1	300	2	1,20E-05
2	300	2	1,20E-05
3	300	2	1,20E-05
4	300	2	1,20E-05



PRUEBA 6			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	5		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	0,000012
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	360	2	8,9913E-06
1	360	2	8,9913E-06
2	360	2	8,9913E-06
3	360	2	8,9913E-06
4	300	2	1,20E-05
5	300	2	1,20E-05



MEJORES INDIVIDUOS DE CADA PRUEBA DEL CASO			
PRUEBA	IPE	NDIV	FITNESS
1	330	2	1,00E-05
2	330	2	1,00E-05
3	300	9	7,3733E-06
4	300	9	7,3733E-06
5	**300**	**2**	**0,000012**
6	**300**	**2**	**0,000012**

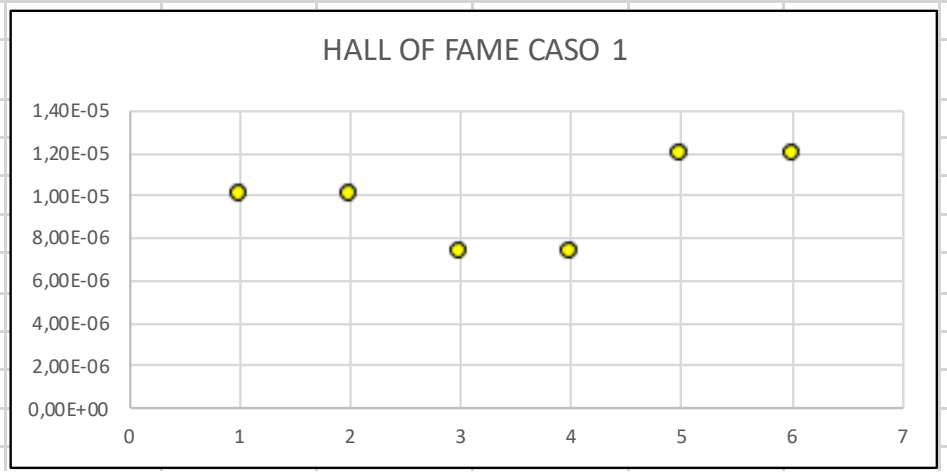


Ilustración 86. Resultados Caso 1.

- Similitudes y patrones:

1. Consistencia del mejor individuo:
   IPE 300 y NDIV 2 se destacan como la combinación óptima en varias pruebas, obtenemos la confirmación en las pruebas 5 y 6 donde vemos que su valor de fitness es 1,20E-05 siendo este el mayor entre todos como también podemos ver reflejado en el Hall of Fame.

2. Rápida convergencia:
   En múltiples pruebas, el individuo se mantiene constante después de las primeras generaciones, indicando una rápida convergencia.

- Variaciones y desarmonías:

1. Generaciones en exceso:
   La Prueba 2 muestra que realizar un mayor número de generaciones no siempre conlleva el encontrar el mejor fitness, por ejemplo, en su generación 4 donde el fitness decrece significativamente.

2. Diversidad en las Poblaciones Iniciales:
   Las diferentes configuraciones iniciales influyen enormemente en la convergencia, por ejemplo, en la prueba 5, donde en la población inicial ya encuentra un resultado tan bueno que no consigue mejorarlo con el paso de las otras generaciones, siempre sobrevive ese individuo gracias al elitismo por permanecer siendo el mejor.

- Conclusión del individuo seleccionado en este caso:

La combinación que se presenta como la óptima después de realizar estas pruebas es la que usa un perfil IPE 300 para la sección trasversal de las vigas y un total de 2 divisiones de celosía, muestra mayor efectividad bajo las condiciones establecidas.

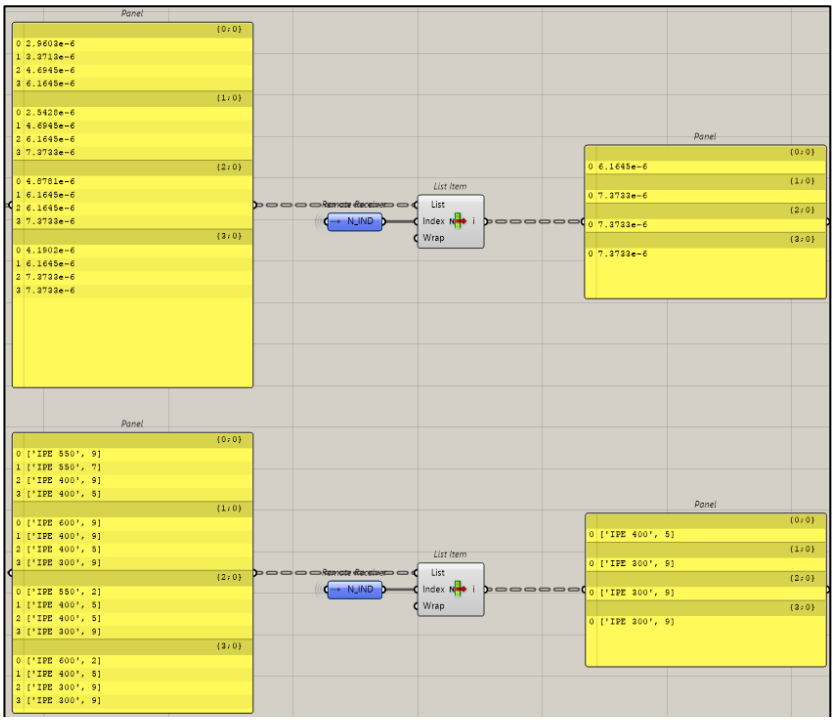


Ilustración 87. Resultados en GH - Caso 1 - Prueba 3

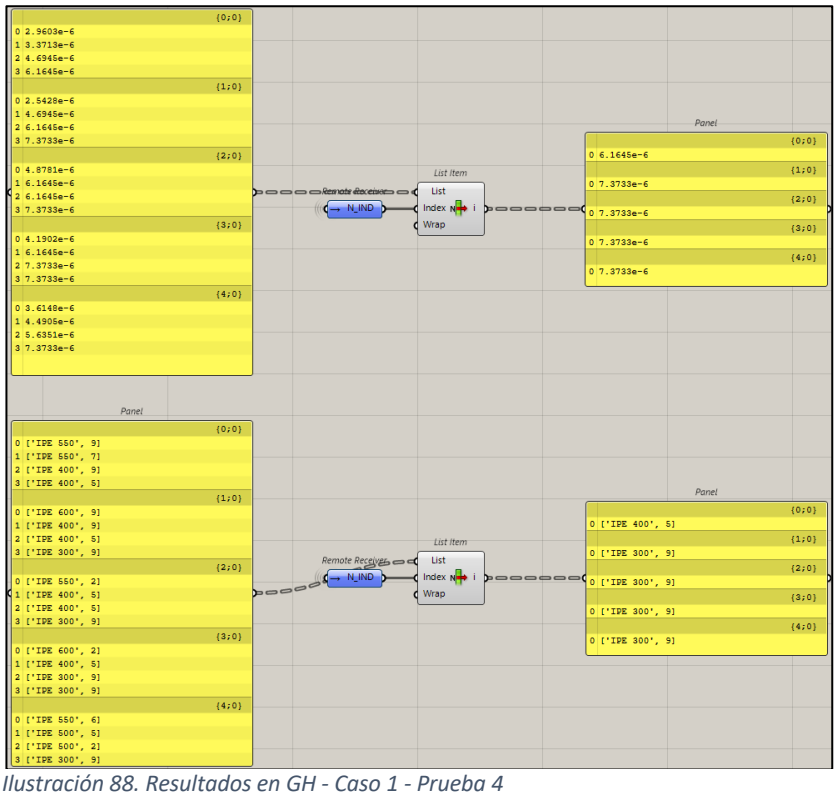


Ilustración 88. Resultados en GH - Caso 1 - Prueba 4

A continuación, muestro tanto en Grashopper como RFEM la celosía que resulta óptima en este caso. Nos servirá de cara a algunos de los casos siguientes, siendo este el resultado óptimo también en la mayoría de ellos es por este motivo que solo lo reflejaré aquí, para que no resulte redundante. Además, en el segundo caso haré otra comparativa para mostrar las estructuras generadas por otras combinaciones; en el tercero, cuarto y quinto no se añaden por no reincidir en la misma información. Dado que la variación en la geometría del modo que realmente se puede apreciar es en los videos grabados que podrían verse a la hora de la exposición presencial y oral de este proyecto.

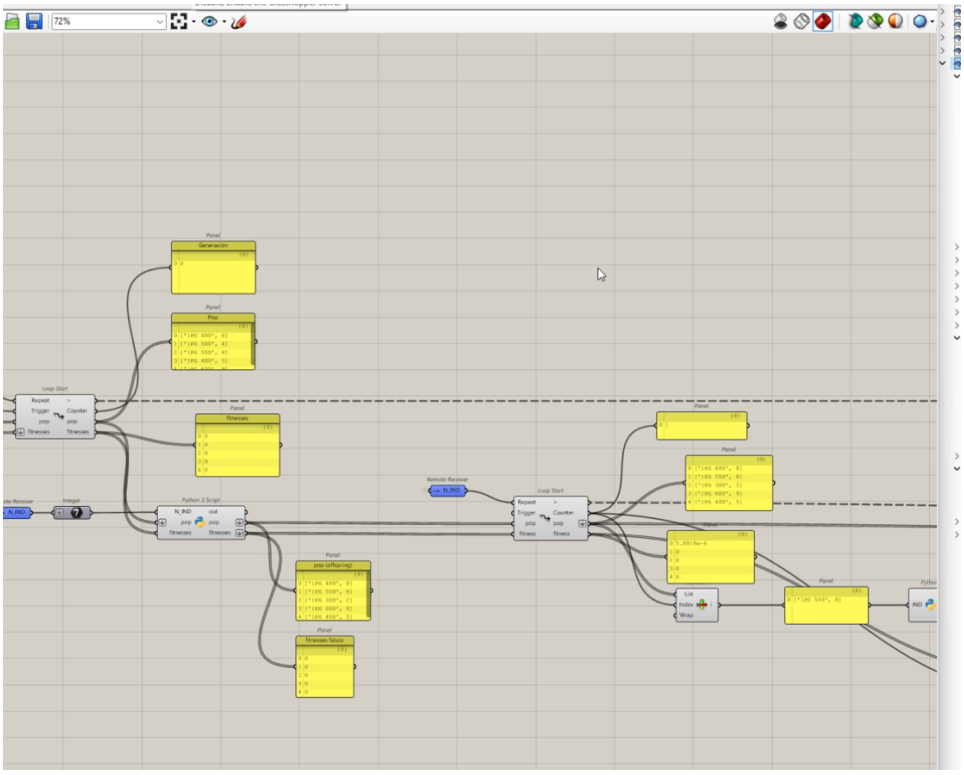


Ilustración 89.Caso 1 celosía en RFEM.

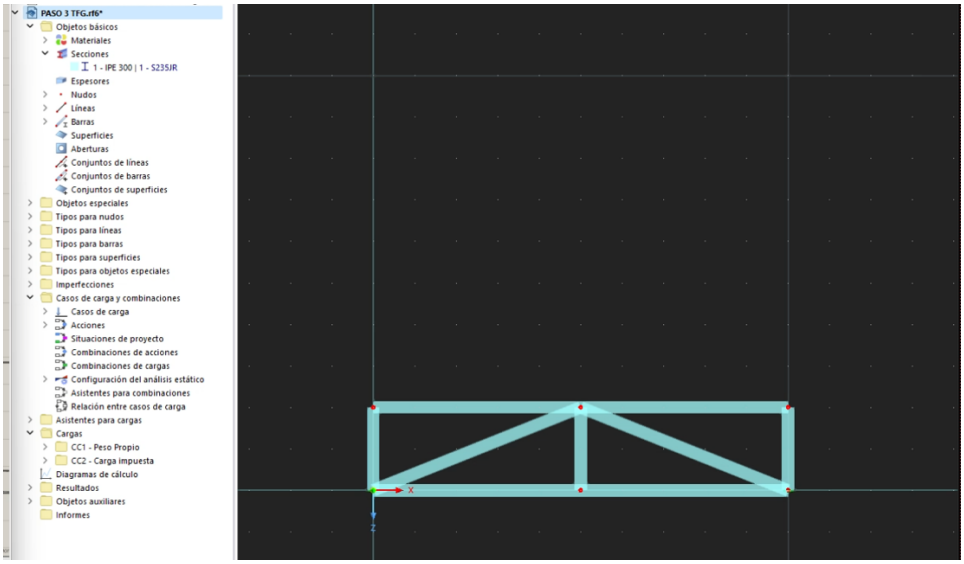


Ilustración 90. Caso 1 celosía en RFEM (II).

4.1.2. Caso 2. Cambio en la geometría.

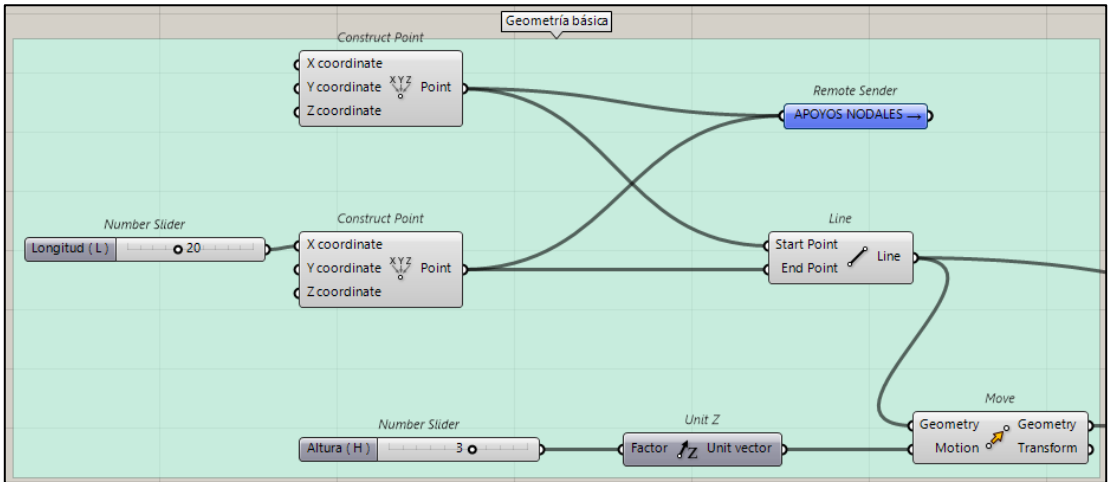


Ilustración 91. Cambios realizados en la geometría Caso 2.

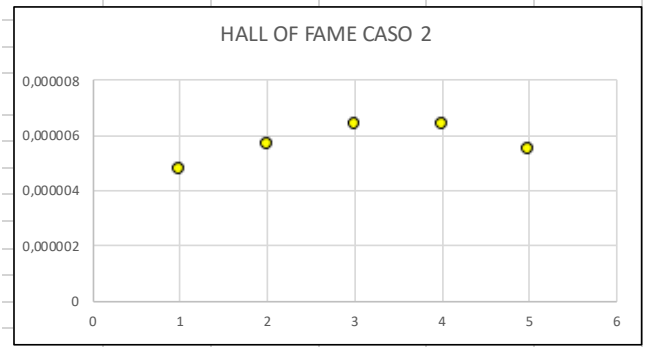
En el Caso 2, se modifica únicamente la geometría del modelo donde, la longitud ha sido aumentada de 10 metros a 20 metros, por otro lado, la altura ha sido incrementada de 2 metros a 3. Por lo demás, el modelo permanecerá igual que en el Caso 1.

Para este caso se seguirán las mismas directrices que en el anterior, realizaremos 5 pruebas esta vez, variando el número de individuos y generaciones en cada una de ellas. En cada prueba se registrará el mejor individuo de cada generación y se mostrará un análisis mediante gráficos. Luego haremos un nuevo Hall of Fame con los mejores resultados de cada una de estas pruebas para evaluar y comparar soluciones.

Este enfoque nos permitirá analizar como las modificaciones geométricas afectan al rendimiento del modelo y determinar si las nuevas dimensiones influyen en la calidad de las soluciones obtenidas.

Por optimización del espacio, mostraré primero en este caso el Hall of Fame y luego los resultados individuales de cada prueba:

MEJORES INDIVIDUOS DE CADA PRUEBA DEL CASO			
PRUEBA	IPE	NDIV	FITNESS
1	300	7	4,80E-06
2	300	4	5,70E-06
3	**300**	**2**	**6,3932E-06**
4	**300**	**2**	**6,3932E-06**
5	330	2	5,49E-06



CASO 2: CAMBIO EN LA GEOMETRÍA LONGITUD=20m ALTURA=3m

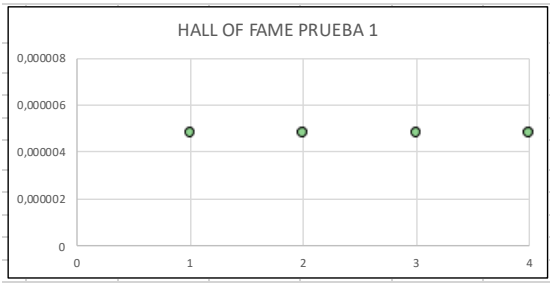
PRUEBA 1			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	300	7	4,80E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	300	7	4,8038E-06
1	300	7	4,8038E-06
2	300	7	4,8038E-06
3	300	7	4,8038E-06

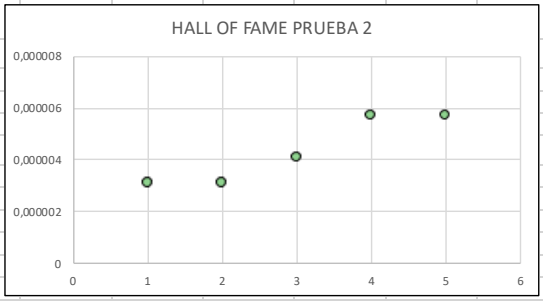
PRUEBA 2			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	4	5,70E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	450	4	3,11E-06
1	450	4	3,11E-06
2	400	2	4,07E-06
3	300	4	5,70E-06
4	300	4	5,70E-06

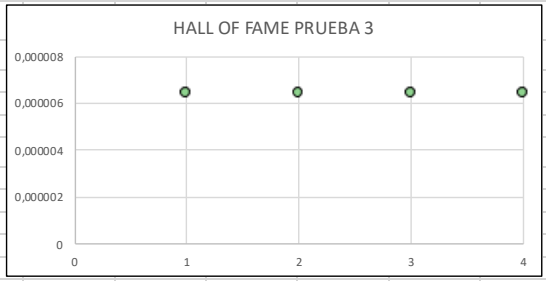
PRUEBA 3			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	6,3932E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	300	2	6,3932E-06
1	300	2	6,3932E-06
2	300	2	6,3932E-06
3	300	2	6,3932E-06

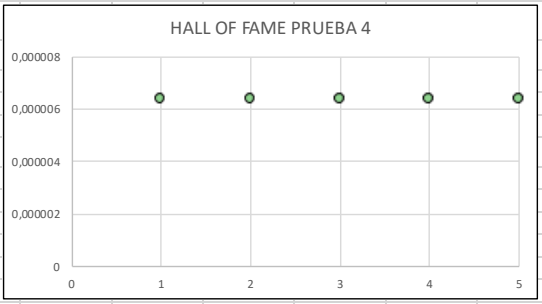
PRUEBA 4			
CARACTERÍSTICAS			
Nº INDIVIDUOS	4		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	6,3932E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	300	2	6,3932E-06
1	300	2	6,3932E-06
2	300	2	6,3932E-06
3	300	2	6,3932E-06
4	300	2	6,3932E-06

PRUEBA 5			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	6		
MEJOR INDIVIDUO DE LA PRUEBA	330	2	5,49E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	360	2	4,73E-06
1	330	2	5,49E-06
2	330	2	5,49E-06
3	330	2	5,49E-06
4	330	2	5,49E-06
5	330	2	5,49E-06
6	330	2	5,49E-06









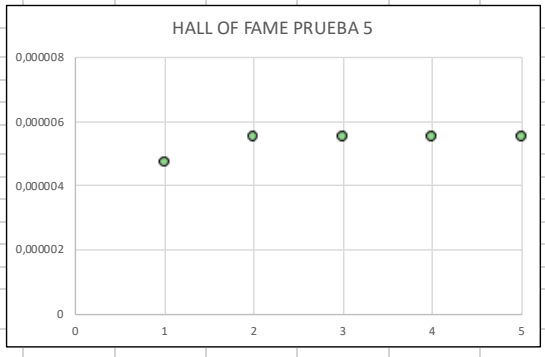


Ilustración 92. Resultados Caso 2.

- Similitudes y patrones:

1. Consistencia de los mejores individuos:
   En todas las pruebas, el perfil IPE aparece constantemente como el mejor individuo en términos de fitness. Esto indica que, independientemente del número de generaciones e individuos, el perfil IPE 300 es una opción favorable bajo las nuevas condiciones geométricas. Además, el número de divisiones (NDIV) también es relativamente bajo en la mayoría de pruebas, con valores comunes de entre 2 y 4.

2. Estabilidad en los resultados:
   En todas las pruebas, una vez que se encuentra un individuo con un buen valor de fitness, este se mantiene constante a través de las generaciones restantes. Esto muestra que las soluciones encontradas son bastante estables y no mejoran ni empeoran significativamente dependiendo del número de generaciones. También nos muestra que el elitismo y la mutación están haciendo bien su función, manteniendo los individuos con mejor fitness de generación en generación y cuando encuentra un parámetro bueno como ha sido el perfil 300, muta en el número de divisiones hasta encontrar aquellas con el máximo valor de fitness.

- Variaciones y desarmonías:

1. Variaciones en el fitness inicial:
   La población inicial muestra una variabilidad considerable en el valor del fitness, con diferencias notables entre las pruebas. Por ejemplo, en la prueba 1, el valor inicial es 4,80E-06 mientras que en otras pruebas es significativamente menor. Esto podría indicar que las condiciones iniciales pueden tener un impacto en el resultado final, dando ventaja o perjudicando en ciertos casos.

2. Resultados divergentes en la prueba 5:
   Se observa que el mejor individuo no sigue la tendencia común de ser un IPE 300. En esta prueba, el mejor individuo tiene un perfil IPE 330 con 2 divisiones, lo que le hace diferente. Además, el valor del fitness es relativamente alto (5,49E-06), lo que indica una buena adaptación pese a la distinción respecto a las demás pruebas.

3. Mejora gradual en la prueba 2:
   Podemos observar gracias a la gráfica como tiene una tendencia positiva con el paso de las generaciones, lo que demuestra que el algoritmo está trabajando en buenas condiciones y presentando resultados eficientes.

- Conclusiones sobre el mejor individuo en este caso:

A lo largo de las pruebas domina el perfil IPE 300 también con dos divisiones, se destaca como el más eficiente, sugiriendo que este sería el perfil óptimo bajo las nuevas condiciones geométricas de 20m de longitud y 3m de altura.

Vemos que, pese al cambio de la geometría, las condiciones óptimas del individuo se han mantenido constantes, que quizás podría decirnos que no es el factor más influyente si no es brusca la variación geométrica y podría haber otros que repercutiesen más. Es lo que estudiaremos a continuación con los otros casos.

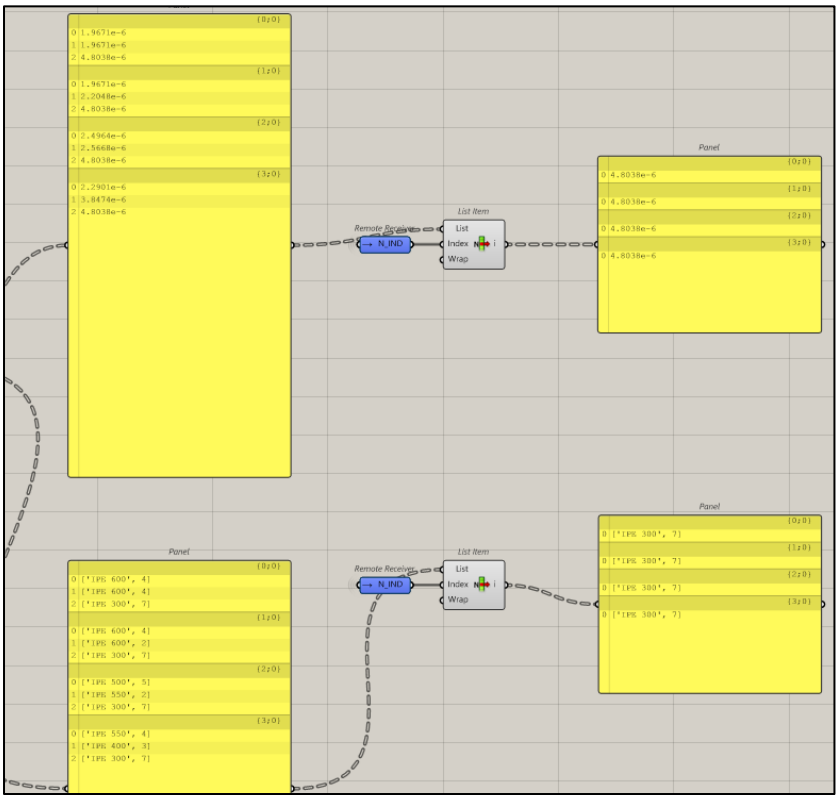


Ilustración 93. Resultados en GH - Caso 2 - Prueba 1

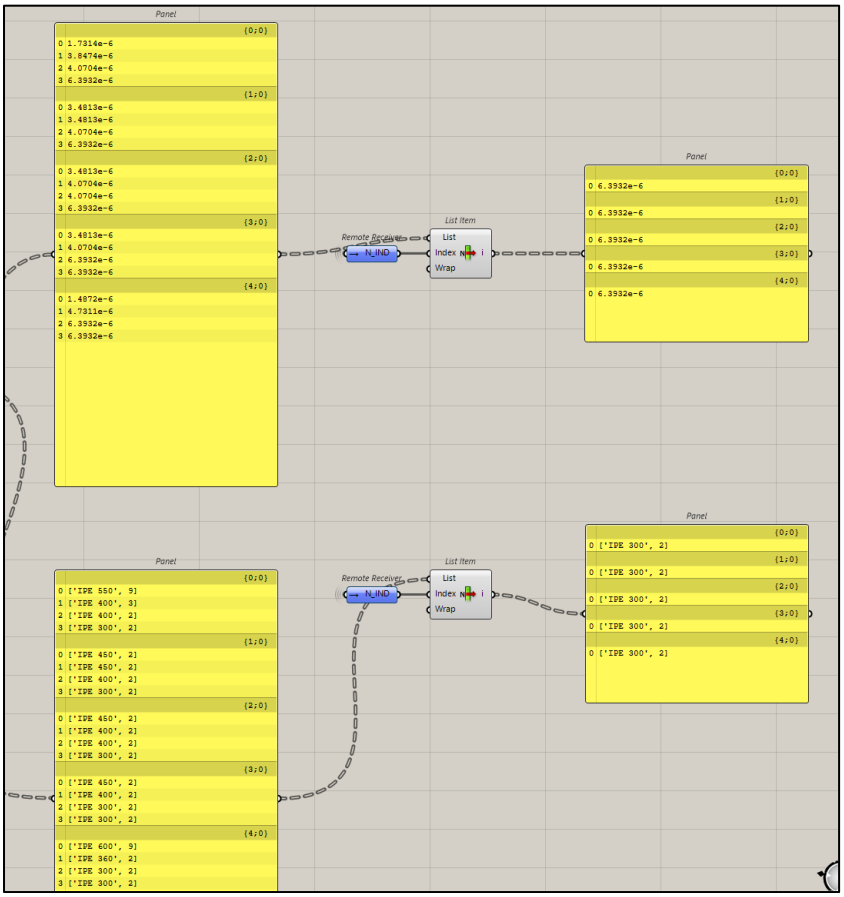
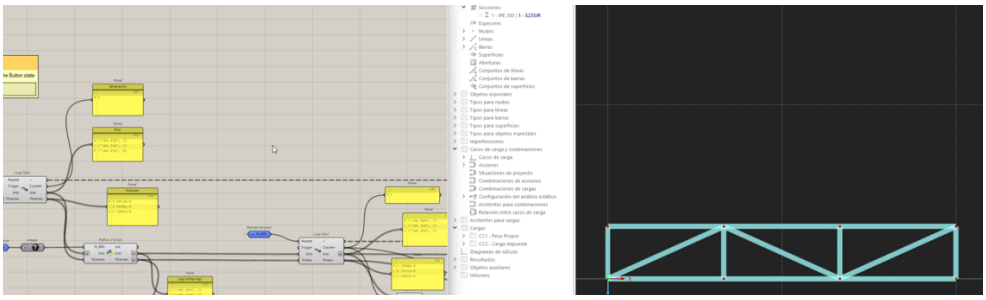
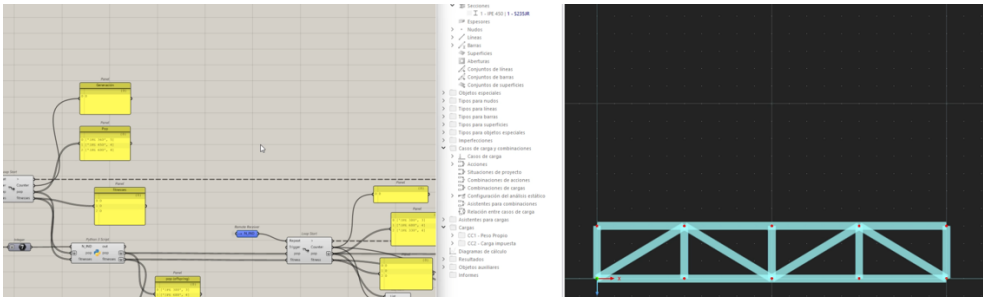


Ilustración 94. Resultados en GH - Caso 2 - Prueba 4

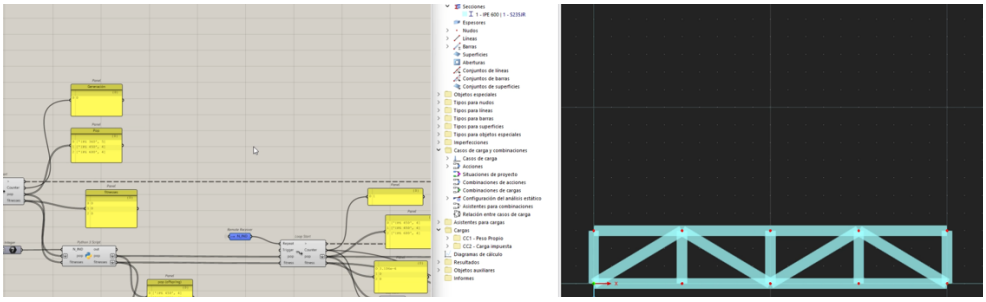
Como el mejor resultado es el mismo que en el caso anterior, voy a mostrar aquí algunas de las celosías estudiadas en estas pruebas para visualizar los cambios entre ellas:



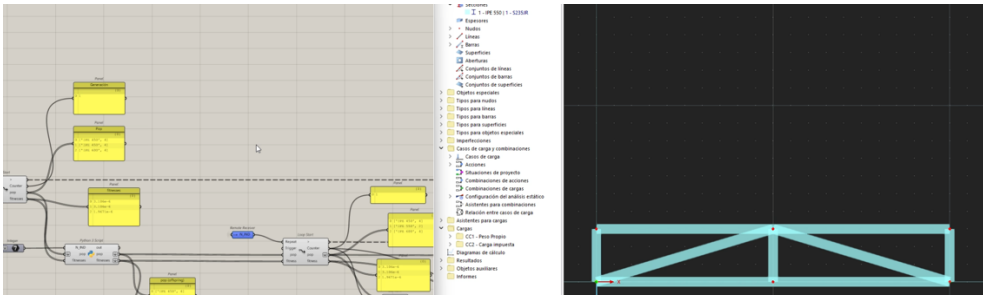
IPE 300, NDIV=3



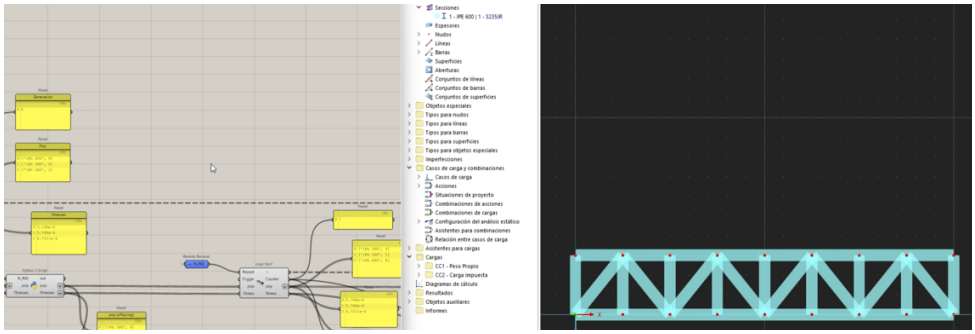
IPE 450, NDIV=4



IPE 600, NDIV=4



IPE 550, NDIV=2



IPE 600, NDIV=8

4.1.3. Caso 3. Cambio en la geometría y en la carga impuesta.

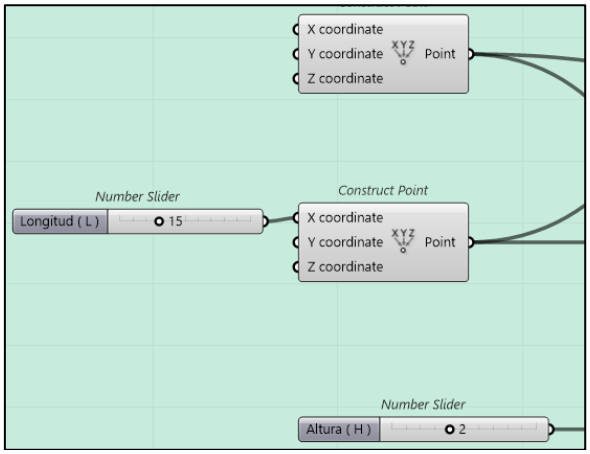


Ilustración 95. Cambio geometría Caso 3.

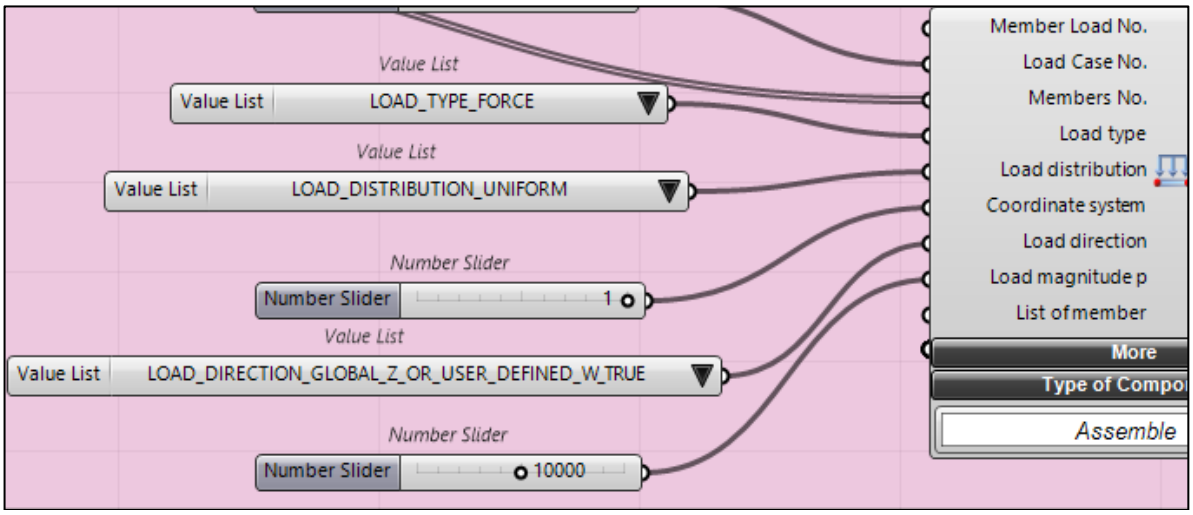


Ilustración 94. Cambio en la carga impuesta en Member Load.

En este tercer caso de pruebas del modelo, realizaremos algunas modificaciones en la geometría y la carga aplicada. La longitud del modelo se ajustará de 10 metros a 15 metros, manteniendo la altura inicial de 2 metros sin cambios. Además, se modificará el valor de la carga en el componente “member load”, aumentando la magnitud de la carga de 3kN a 10 kN.

Los cambios realizados son relativamente moderados en comparación con las magnitudes estructurales involucradas. En casos anteriores, hemos observado una tendencia a mantener como óptima la misma solución, independientemente de las variaciones en geometría. Este caso tiene como objetivo evaluar si esta tendencia persiste o si las soluciones óptimas difieren más significativamente debido a estas nuevas modificaciones.

A continuación, procederemos a visualizar los resultados de las pruebas bajo estas nuevas condiciones y analizaremos los resultados obtenidos para determinar si la solución óptima es persistente o presenta variaciones.

CASO 3: CAMBIO EN LA GEOMETRÍA Y CARGA.	L=15m	H=2m	CARGA= 10KN

PRUEBA 1			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	330	2	7,45E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	400	2	5,21E-06
1	400	2	5,21E-06
2	400	2	5,21E-06
3	330	2	7,45E-06

PRUEBA 2			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	4	7,84E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	300	8	6,3885E-06
1	330	2	7,4519E-06
2	330	2	7,4519E-06
3	300	4	7,84E-06
4	330	2	7,4519E-06

PRUEBA 3			
CARACTERÍSTICAS			
Nº INDIVIDUOS	6		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	360	2	6,4167E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	330	6	6,0743E-06
1	330	6	6,0743E-06
2	330	6	6,0743E-06
3	360	2	6,4167E-06

PRUEBA 4			
CARACTERÍSTICAS			
Nº INDIVIDUOS	4		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	8,6708E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	300	6	7,0679E-06
1	300	6	7,0679E-06
2	300	6	7,0679E-06
3	300	2	8,6708E-06
4	300	2	8,6708E-06

PRUEBA 5			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	8,6708E-06
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	330	7	5,77E-06
1	300	2	8,6708E-06
2	300	2	8,6708E-06
3	300	2	8,6708E-06
4	300	2	8,6708E-06

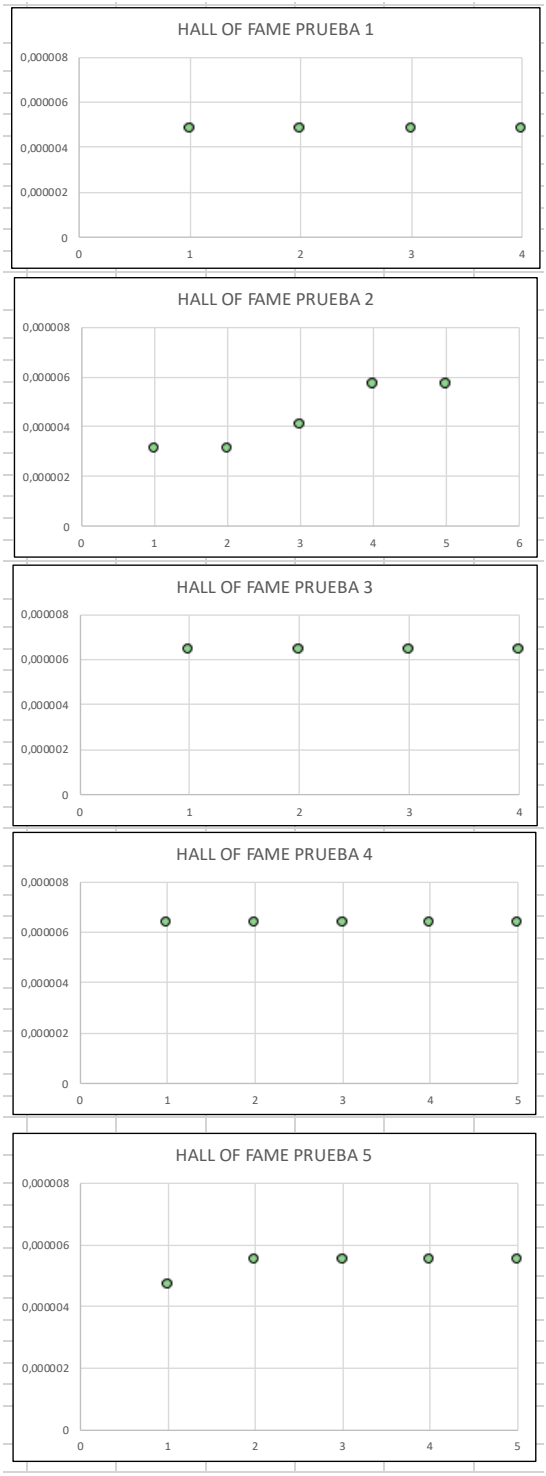
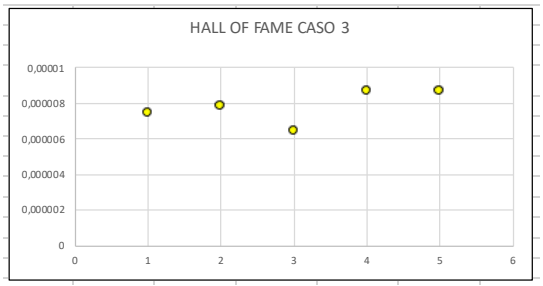


Ilustración 96. Resultados Caso 3.

MEJORES INDIVIDUOS DE CADA PRUEBA DEL CASO			
PRUEBA	IPE	NDIV	FITNESS
1	330	2	7,45E-06
2	300	4	7,84E-06
3	360	2	6,4167E-06
4	**300**	**2**	**8,6708E-06**
5	**300**	**2**	**8,6708E-06**



- <u>Similitudes y tendencias observadas:</u>

1. Consistencia de los mejores individuos:
   En varias pruebas los mejores individuos tienden a converger a un perfil IPE 300 con un NDIV de 2 o 4. Este patrón nos sugiere que, a pesar de las modificaciones en la geometría y en la carga, estos individuos siguen siendo óptimos.

2. Estabilidad del fitness:
   Los valores del fitness muestran una cierta estabilidad dentro de cada prueba, o incrementan para bien su valor. Por ejemplo, en la prueba 1, el fitness mejora desde 5,21E-06 en la población inicial a 7,45E-06 en la última generación, y en la prueba 4, el mejor individuo mantiene un fitness de 8,6708E-06 desde la tercera generación en adelante.

3. Convergencia temprana:
   En varias pruebas se identifica que el mejor individuo se identifica en una etapa temprana de la evolución y se mantiene consistente hasta el final de las generaciones. Esto se muestra en pruebas como la 5 donde el mejor individuo no cambia en las últimas generaciones.

- <u>Variaciones y desarmonías:</u>

1. Variabilidad inicial:
   En la prueba 2, se observa una variabilidad inicial significativa, con el mejor individuo cambiando de IPE 300 y 8 divisiones a IPE 330 y 2 divisiones, antes de obtener el óptimo en IPE 300 y NDIV 4. Esto sugiere una mayor exploración del espacio de soluciones en las primeras generaciones.

2. Mejores individuos diferentes:
   A diferencia de otros casos, en los mejores individuos de cada generación observamos una amplia mayor de perfiles IPE, por ejemplo, en la prueba 3 el IPE 360 es el ganador con 2 divisiones, lo que difiere de la tendencia observada en otras pruebas. Esto podría indicar una influencia de un mayor número de individuos y menor de generaciones.

- Conclusión sobre el individuo final seleccionado:

A través de las pruebas realizadas en este caso con cambios en la geometría y carga, se observa que los mejores individuos siguen presentando ciertas tendencias consistentes, como la preferencia del perfil IPE 300 con bajos números de divisiones. Sin embargo, también se detectan algunas variabilidades y diferencias en los resultados, especialmente al variar número de individuos y generaciones.

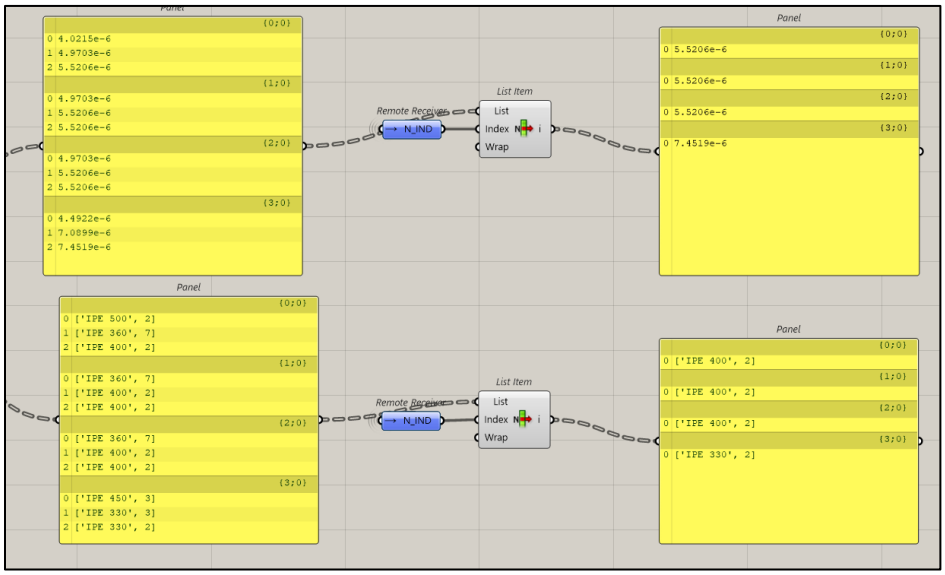


Ilustración 97. Resultados en GH - Caso 3 - Prueba 1

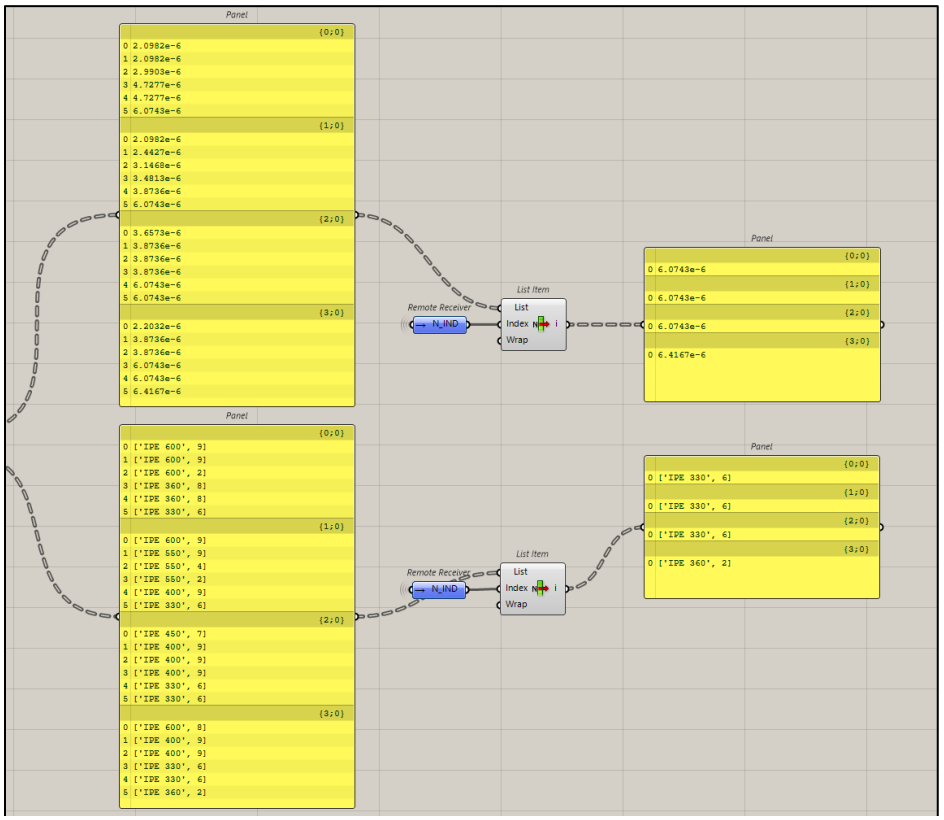


Ilustración 98. Resultados en GH - Caso 3 - Prueba 3

4.1.4. Caso 4. Cambio de la función de fitness a pesos ponderados.

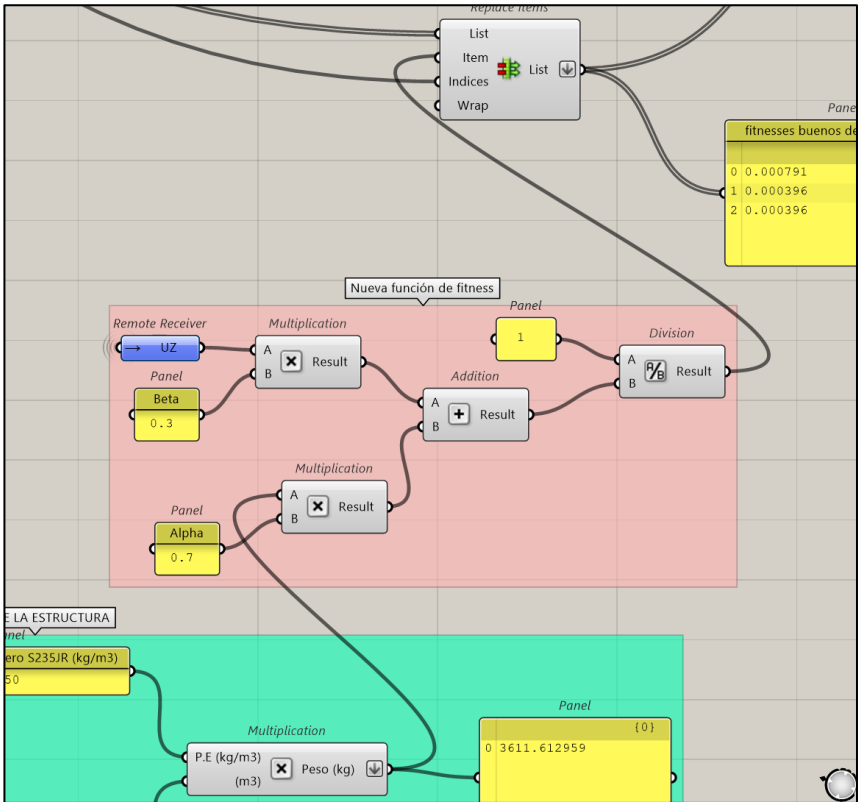


Ilustración 99. Cambio del fitness en Caso 4.

En este cuarto caso de pruebas, vamos a mantener todas las condiciones iniciales del caso 1 de geometría y cargas de la estructura. La longitud será de 10 metros, la altura 2 metros, y la carga aplicada de 3kN. Sin embargo, introduciremos un cambio significativo en la función de fitness, utilizando el método de los pesos ponderados.

El método de los pesos ponderados permite ajustar la relevancia de diferentes parámetros en la evaluación del fitness. En este modelo, los dos parámetros evaluados son la deformación (Uz) y el peso de la estructura según el área de su perfil. Con este enfoque, podemos asignar diferentes ponderaciones a cada parámetro según su importancia relativa, voy a considerar que es más crucial optimizar el peso de la estructura para ahorrar en material, asignándole una ponderación de 0,7. La deformación Uz, aunque también importante, recibirá una ponderación menor de 0,3, ambas deben de sumar un total de 1 unidad.

La fórmula de la nueva función de fitness será la suma ponderada de los parámetros (cada uno multiplicado por su ponderación), ajustada para la maximización. Como buscamos maximizar el valor del fitness hemos de dividir “1” entre el resultado obtenido de la suma ponderada. Esto transformará el fitness de manera que los valores óptimos serán los más altos.

A continuación, muestro la visualización de dos pruebas utilizando esta nueva función, el numero escaso de pruebas realizadas se debe a los resultados vistos en ellas que analizaré abajo.

CASO 4: CAMBIO DE LA FUNCIÓN DE FITNESS CON MÉTODO DE PESOS PONDERADOS.			
PRUEBA 1			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	330	2	7,91E-04
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	330	2	7,91E-04
1	330	2	7,91E-04
2	330	2	7,91E-04
3	330	2	7,91E-04
PRUEBA 2			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	300	2	9,20E-04
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	360	2	6,81E-04
1	360	2	6,81E-04
2	360	2	6,81E-04
3	300	2	9,20E-04
4	300	2	9,20E-04

MEJORES INDIVIDUOS DE CADA PRUEBA DEL CASO			
PRUEBA	IPE	NDIV	FITNESS
1	330	2	7,91E-04
2	**300**	**2**	**9,20E-04**

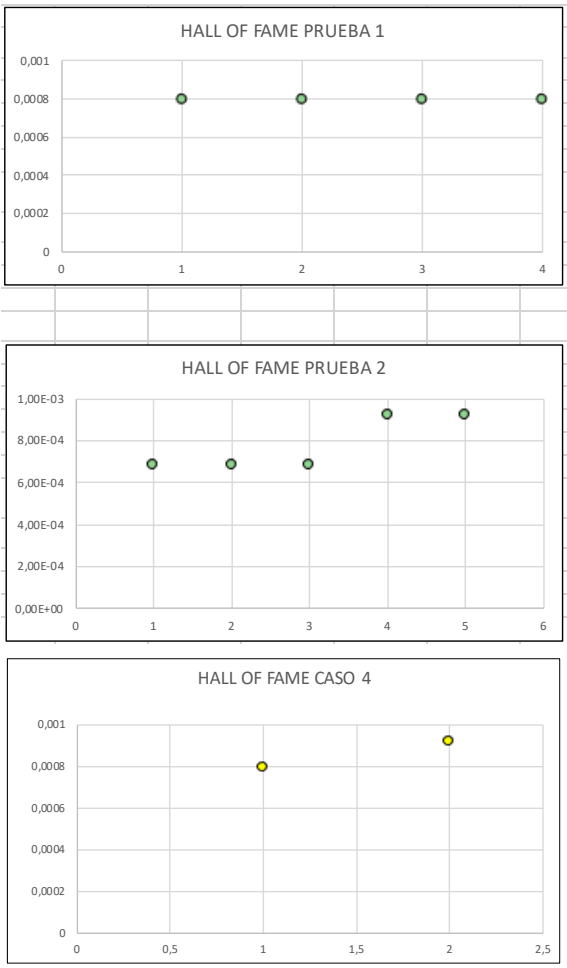


Ilustración 100. Resultados Caso 4.

- <u>Similitudes y patrones:</u>

  1. Consistencia en resultados:
     En ambas pruebas, el mejor individuo final tiende a estabilizarse después de la primera generación. Esto sugiere que el proceso de optimización converge rápidamente hacia una solución óptima en términos de la nueva función de fitness.

  2. En ambas pruebas, los valores de NDIV se mantienen a "2", lo que indica una posible tendencia hacia estructuras menos divididas siendo óptimas con este nuevo método de evaluación.

- <u>Variaciones o desarmonías:</u>

  1. La mejora del fitness en la segunda prueba sugiere que el aumento del número de individuos y generaciones permite encontrar mejores soluciones, pero sin afectar al número de divisiones.

  2. Si encuentra de manera rápida la solución óptima en la población inicial, la mantiene con el tiempo generando monotonía debida al elitismo en los individuos ganadores.

- Conclusiones generales:

  - La rápida convergencia hacia valores más óptimos sugiere que la función de fitness está bien definida y adecuada para la optimización de este modelo estructural.
  - La variación en el número de individuos y generaciones parece influir positivamente en los resultados, permitiendo encontrar mejores soluciones cuando se incrementan estos parámetros.
  - Se han realizado menos pruebas en este caso porque rápidamente se volvía a deducir que obtenía el mejor individuo rápidamente siendo su IPE y NDIV igual que el obtenido en los casos anteriores, por tanto, la ponderación de variables no resulta realmente significativa a la hora de hallar un resultado final. Es por eso que para el siguiente caso se variará directamente la lista de perfiles IPE para generar un cambio brusco de las soluciones y observar cómo se comporta.

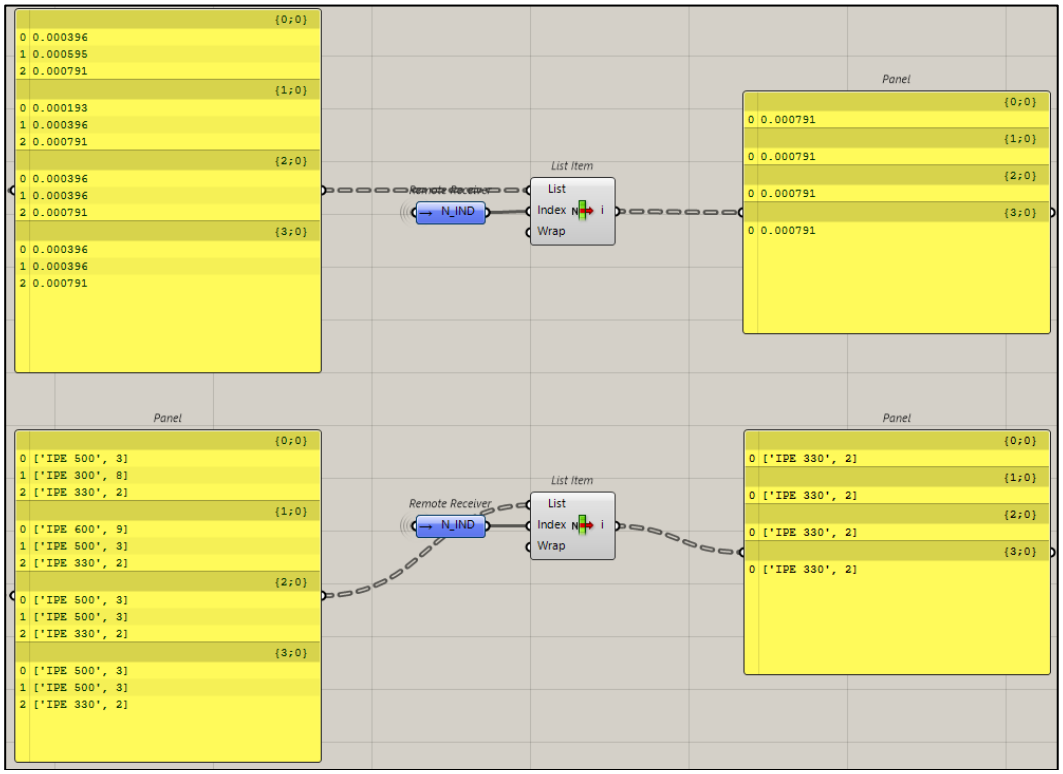


Ilustración 101. Resultados en GH - Caso 4 - Prueba 1

4.1.5. Caso 5. Fitness a pesos ponderados y variación de la lista de perfiles IPE.

En este caso, mantenemos todas las condiciones iguales a las del Caso 4, incluyendo geometría inicial, carga y función de fitness ponderada. Por otro lado, realizaremos una modificación en la lista de perfiles IPE disponibles. En lugar de utilizar perfiles que van de 300 a 600, sustituimos esta lista por una nueva lista que alterna los valores reglados entre 200 y 400. Este cambio no es sencillo ya que implica ajustar los valores en varios lugares del modelo para asegurar que la selección y mutación se realice correctamente en la nueva lista, además, para el cálculo del peso de la estructura estableciendo sus áreas trasversales regladas.

El objetivo de este caso es observar como la restricción de perfiles IPE en un rango más limitado y específico afecta al rendimiento del modelo y los resultados obtenidos. Evaluaremos si el cambio en la lista de perfiles impacta significativamente en la calidad de los individuos.

```python
# requirements: deap
# requirements: random2
import random
from deap import base, creator, tools

# Definir la lista de perfiles IPE
ipe_profiles = ["IPE 200", "IPE 220", "IPE 240", "IPE 270", "IPE 300", "IPE 330", "IPE 360", "IPE 400"]

# Función para crear un individuo
def create_individual():
    profile = random.choice(ipe_profiles)
    divisions = random.randint(2, 9) #min nº de div = 2 para que tengas un nodo con uz distinto de 0!
    return [profile, divisions]

# Definir el tipo de individuo en DEAP
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)  #### LIST EN LUGAR DE TUPLE

# Registrar la función para crear individuos en el toolbox
toolbox = base.Toolbox()
toolbox.register("individual", tools.initIterate, creator.Individual, create_individual)

# Registrar la población
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

# Generar la población
def generate_population(N_IND):
    return toolbox.population(n=N_IND)

# Generar la población y devolverla como lista
population = generate_population(N_IND)

```

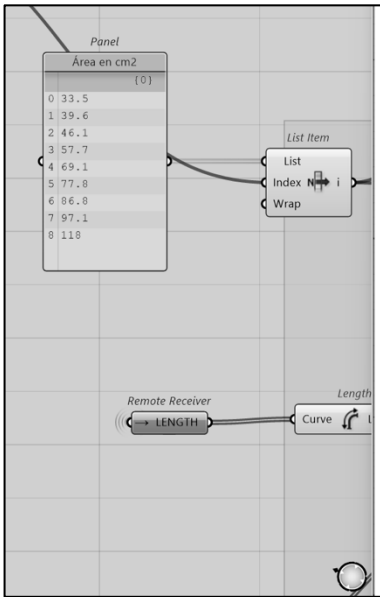
Ilustración 102. Cambio en el script de generación de población inicial.

Script Editor

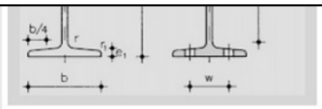
Grasshopper Archivo Editar Ejecutar Herramientas Ventana Ayuda

```python
"""Grasshopper Script"""


if x == "IPE 200":
    a = 0
elif x == "IPE 220":
    a = 1
elif x == "IPE 240":
    a = 2
elif x == "IPE 270":
    a = 3
elif x == "IPE 300":
    a = 4
elif x == "IPE 330":
    a = 5
elif x == "IPE 360":
    a = 6
else:
    a = 7

print(a)

```

Ilustración 103. Cambio en el script de cálculo del peso.



I_y = Momento de inercia de la sección, respe
$W_y = 2I_y : b$. Módulo resistente de la sección, r
$i_y = \sqrt{I_y : A}$. Radio de giro de la sección, resp



Perfil	Dimensiones							Términos de sección						
	h mm	b mm	e=r mm	e_1 mm	r_1 mm	h_1 mm	u mm	A cm^2	S_x cm^3	I_x cm^4	W_x cm^3	i_x cm	I_y cm^4	W cm
IPN 80	80	42	3,9	5,9	2,3	59	304	7,58	11,4	77,8	19,5	3,20	6,29	3,
IPN 100	100	50	4,5	6,8	2,7	75	370	10,60	19,9	171,0	34,2	4,01	12,20	4,
IPN 120	120	58	5,1	7,7	3,1	92	439	14,20	31,8	328,0	54,7	4,81	21,50	7,
IPN 140	140	66	5,7	8,6	3,4	109	502	18,30	47,7	573,0	81,9	5,61	35,20	10,
IPN 160	160	74	6,3	9,5	3,8	125	575	22,80	68,0	935,0	117,0	6,40	54,70	14,
IPN 180	180	82	6,9	10,4	4,1	142	640	27,90	93,4	1.450,0	161,0	7,20	81,30	19,
IPN 200	200	90	7,5	11,3	4,5	159	709	33,50	125,0	2.140,0	214,0	8,00	117,00	26,
IPN 220	220	98	8,1	12,2	4,9	175	775	39,60	162,0	3.060,0	278,0	8,80	162,00	33,
IPN 240	240	106	8,7	13,1	5,2	192	844	46,10	206,0	4.250,0	354,0	9,59	221,00	41,
IPN 260	260	113	9,4	14,1	5,6	208	906	53,40	257,0	5.740,0	442,0	10,40	288,00	51,
IPN 280	280	119	10,1	15,2	6,1	225	966	61,10	316,0	7.590,0	542,0	11,10	364,00	61,
IPN 300	300	125	10,8	16,2	6,5	241	1.030	69,10	381,0	9.800,0	653,0	11,90	451,00	72,
IPN 320	320	131	11,5	17,3	6,9	257	1.090	77,80	457,0	12.510,0	782,0	12,70	555,00	84,
IPN 340	340	137	12,2	18,3	7,3	274	1.150	86,80	540,0	15.700,0	923,0	13,50	674,00	98,
IPN 360	360	143	13,0	19,5	7,8	290	1.210	97,10	638,0	19.610,0	1.090,0	14,20	818,00	114,
IPN 380	380	149	13,7	20,5	8,2	306	1.270	107,00	741,0	24.010,0	1.260,0	15,00	975,00	131,
IPN 400	400	155	14,4	21,6	8,6	323	1.330	118,00	857,0	29.210,0	1.460,0	15,70	1.160,00	149,

Ilustración 104. Tabla de la que se toman las áreas de los perfiles para el cálculo del peso.

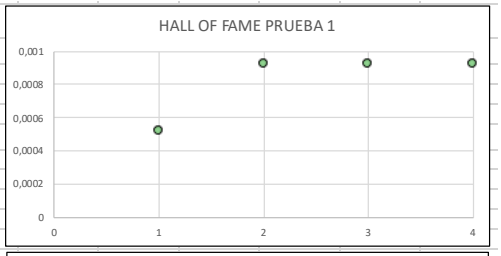
CASO 5: CAMBIO DE LA LISTA DE PERFILES IPE + CAMBIO FITNESS PASO 4

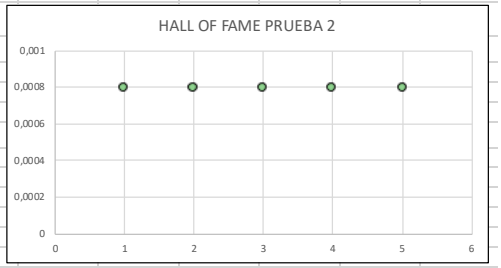
PRUEBA 1			
CARACTERÍSTICAS			
Nº INDIVIDUOS	3		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	240	4	9,22E-04
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	270	9	5,21E-04
1	240	4	9,22E-04
2	240	4	9,22E-04
3	240	4	9,22E-04

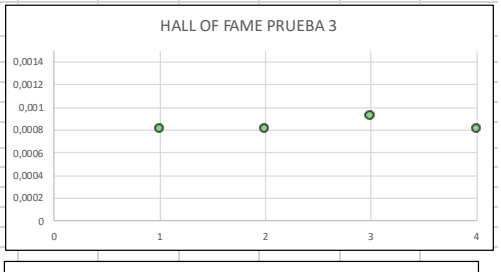
PRUEBA 2			
CARACTERÍSTICAS			
Nº INDIVIDUOS	5		
Nº GENERACIONES TOTAL	4		
MEJOR INDIVIDUO DE LA PRUEBA	240	6	7,96E-04
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	240	6	7,96E-04
1	240	6	7,96E-04
2	240	6	7,96E-04
3	240	6	7,96E-04
4	240	6	7,96E-04

PRUEBA 3			
CARACTERÍSTICAS			
Nº INDIVIDUOS	4		
Nº GENERACIONES TOTAL	3		
MEJOR INDIVIDUO DE LA PRUEBA	240	4	9,22E-04
RESULTADOS			
Nº GENERACIÓN	IPE	NDIV	FITNESS
POB.INICIAL	220	8	8,08E-04
1	220	8	8,08E-04
2	240	4	9,22E-04
3	220	8	8,08E-04

MEJORES INDIVIDUOS DE CADA PRUEBA DEL CASO			
PRUEBA	IPE	NDIV	FITNESS
1	240	4	9,22E-04
2	240	6	7,96E-04
3	240	4	9,22E-04







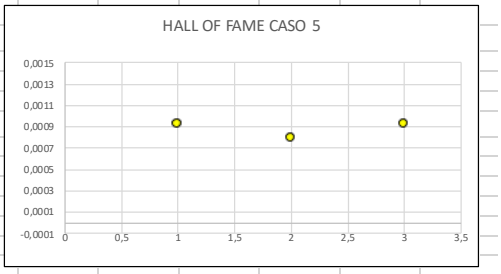


Ilustración 105. Resultados Caso 5

- Similitudes entre las pruebas:

1. Consistencia en el Mejor Individuo:
   En las pruebas 1 y 3, el mejor individuo es consistentemente el perfil IPE 240 con una NDIV de 4 y un fitness de 9.22E-04. Esto sugiere una estabilidad en la selección del mejor individuo bajo las nuevas condiciones.

2. Estabilidad de Fitness:
   En todas las pruebas, los mejores individuos se mantienen constantes a lo largo de las generaciones, indicando que una vez se encuentra un buen individuo, este prevalece.

- Variaciones y Desarmonías:

1. Prueba 2:
   El mejor individuo aquí es también el perfil IPE 240, pero con una NDIV de 6 y un fitness de 7.96E-04. Esta variación en NDIV y fitness indica que aunque el perfil IPE 240 es óptimo, el NDIV puede variar, afectando ligeramente el fitness.

2. Fitness Inicial vs. Final:
   Los valores de fitness iniciales son significativamente más bajos en comparación con los finales en las pruebas 1 y 3, en la 1, el fitness inicial es 5.21E-04, que mejora a 9.22E-04. Esto demuestra que el proceso de selección y mutación está funcionando bien, mejorando la calidad de los individuos.

- Conclusión:

- El cambio en la lista de perfiles IPE ha resultado en una selección más restringida pero efectiva de los mejores individuos.

- Consistencia: A pesar de la variación en la lista de perfiles y la función de fitness ponderada, los resultados muestran una tendencia clara hacia el perfil IPE 240 como solución óptima.

- Impacto de NDIV: La NDIV óptima parece variar entre 4 y 6, lo que indica que hay espacio para optimizar aún más este parámetro dependiendo de las necesidades específicas.

- Mejora con las Generaciones: Los resultados de fitness mejoran significativamente a lo largo de las generaciones, destacando la efectividad del algoritmo en encontrar soluciones óptimas.

Esto nos puede dar a entender que quizás deberíamos haber hecho un estudio más exhaustivo de la selección de los perfiles antes de comenzar con la realización de las pruebas, ya que aquí también observamos perfiles menores (220) que tienen menor fitness que el individuo óptimo del caso (240;4div), esto nos muestra que probablemente es la mejor solución ya que tanto individuos arriba como abajo obtienen peor fitness del que este presenta.

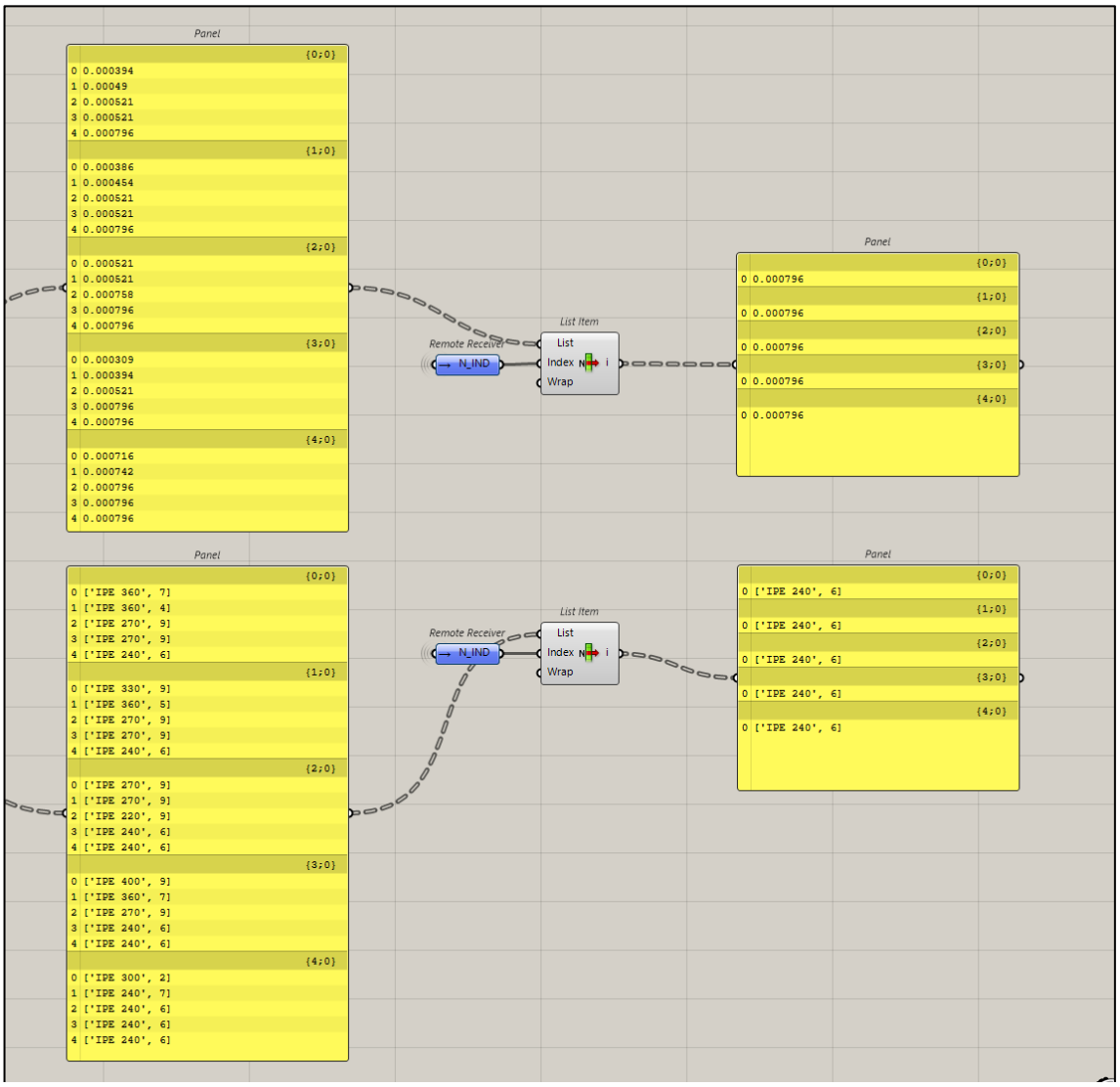


Ilustración 106. Resultados en GH - Caso 5 - Prueba 2

4.1.6. Conclusión general de todos los casos.

En conclusión, a lo largo de los distintos casos analizados, hemos observado que el modelo muestra consistencia y robustez frente a diversas modificaciones en la geometría, carga aplicada y la función de fitness. Los resultados indican una tendencia clara hacia ciertos perfiles IPE como soluciones óptimas siendo estos IPE 300 en los 4 primeros casos e IPE 240 en este último, independientemente de los cambios en las condiciones iniciales.

Aunque se observaron variaciones menores en los valores de NDIV y fitness, los mejores individuos seleccionados permanecieron bastante estables, destacando la efectividad del algoritmo genético en encontrar soluciones óptimas. Esto sugiere que el modelo es adaptable y eficiente, capaz de converger hacia soluciones óptimas a pesar de las variaciones en las condiciones de prueba.

5. Conclusiones

5.1. Resumen de hallazgos.

Durante el desarrollo de esta aplicación de optimización estructural con algoritmos genéticos he sido consciente de que no solo vale un resultado final, sino que el proceso es el todo, he obtenido multitud de conocimientos afrontando los problemas que iban surgiendo en este desarrollo, además, pienso que he realizado importantes hallazgos en esta aplicación los cuales han enriquecido significativamente mi experiencia y habilidades:

- **Evolución desde la Viga Simple hasta Modelos Complejos:** Comenzando con el diseño de una viga simple y avanzando hacia estructuras más complejas como celosías con bucles simples, he podido establecer las bases para la final creación de un modelo complejo y avanzado, utilizando algoritmos genéticos para el diseño de estructuras, he obtenido un profundo entendimiento de cómo diferentes enfoques y técnicas pueden impactar el rendimiento y la eficiencia del diseño estructural.

- **Herramientas para visualizar procesos de optimización e interpretación de resultados**. A través de herramientas de visualización y análisis de datos, he mejorado mi capacidad para identificar patrones, anomalías y tendencias en los resultados obtenidos, lo cual es fundamental para la toma de decisiones informadas en el diseño estructural, pienso que es un hallazgo para que otros puedan seguir el mismo camino y llegar al mismo entendimiento, ya que de este modo la comprensión se facilita.

- **Desarrollo de Habilidades en Programación y APIs:** La implementación de la aplicación ha requerido el uso de programación de módulos y la creación de conexiones API entre diferentes componentes, los cuales no tengo constancia de que se hayan unificado antes, es por lo que pienso también en el logro de establecer esta conexión de manera exitosa. Como hallazgo importante consideraría que ha sido la integración de los algoritmos genéticos, cálculo estructural avanzado y programación visual en un mismo entorno de trabajo. Esta combinación me ha permitido optimizar eficientemente el diseño de estructuras y explorar nuevas posibilidades en la interacción entre el diseño computacional y la ingeniería estructural. Utilizando algoritmos genéticos para la optimización y herramientas de visualización, he podido desarrollar soluciones más innovadoras y de fácil adaptación a muchos modelos y proyectos futuros, mejorando tanto la creatividad como la precisión técnica en proyectos de diseño estructural.

- **Formación Continua y Desarrollo Profesional:** La participación en cursos y la autoformación en estas nuevas tecnologías emergentes y métodos avanzados de optimización han sido fundamentales. Esta previa investigación y formación me ha permitido adquirir las habilidades técnicas y teóricas necesarias para poder proceder al desarrollo de todo este modelo, así como explorar nuevas metodologías que han fortalecido mi capacidad para abordar desafíos técnicos en la ingeniería estructural realizando así cada avance en el proyecto que considero hallazgo tras hallazgo.

En conjunto, estos hallazgos no solo han mejorado y ampliado mi experiencia profesional, sino que también han establecido una base sólida para la innovación continua y el crecimiento en el campo de la optimización estructural mediante algoritmos genéticos y herramientas de análisis avanzado.

5.2. Cumplimiento de objetivos.

En el entorno en el que se desenvuelve este proyecto, hemos recalcado en varias ocasiones que el centro de este es la optimización topológica en el diseño estructural mediante la aplicación de algoritmos genéticos y herramientas de programación visual. Podemos afirmar que el foco objetivo del proyecto ha sido resuelto, implementando un caso complejo en la aplicación de programación visual Grasshopper con la ayuda de la metodología de algoritmos genéticos la cual ha simplificado la obtención del resultado estimado.

Hemos de analizar también cada uno de los objetivos propuestos inicialmente y el modo en el que se han llevado a cabo durante el desarrollo de este proyecto, estos son:

- **Estudiar y comprender los fundamentos básicos de los algoritmos genéticos.**
  Para alcanzar este objetivo, he investigado a fondo con diversas lecturas sobre todo el entorno de los algoritmos genéticos y además, realizado un curso de introducción a ellos durante un mes aproximadamente de investigación de sus técnicas y métodos antes de proceder al inicio del proyecto. Los aspectos más relevantes y fundamentales de los algoritmos genéticos se han reflejado en el proyecto, además de sus aplicaciones prácticas en la optimización de problemas estructurales.

- **Explorar herramientas de programación visual como Grasshopper.**
  De igual modo, gracias a mi tutor pude realizar un curso de más de un mes de duración que me permitió aprender todas las funcionalidades básicas y principales herramientas que ofrece esta aplicación. Permitiéndome esto desenvolverme de manera fluida y eficaz en el entorno de programación al que ya me había familiarizado antes de comenzar este trabajo. La capacidad para generar y modificar estructuras he conseguido dominarla, demostrando la habilidad para ello.

- **Integrar el software de análisis estructural RFEM.**
  Para poder cumplir con este objetivo en el caso complejo de la Celosía, primeramente, diseñé el modelo de la viga simple, donde pude experimentar y tropezar hasta conseguir finalmente el dominio perfecto de la vinculación entre estos dos programas. Como bien acabo de mencionar, me desarrollé en el modelo básico para luego implementarlo en el complejo con soltura y de manera eficaz, es por eso por lo que considero este objetivo como cumplido. Me ha permitido poder calcular y analizar las estructuras diseñadas, asegurándome el cumplimiento con los criterios de resistencia de materiales establecidos.

- **Desarrollar una aplicación que combine Grasshopper y RFEM.**
  Además de todo lo mencionado en el apartado superior, he investigado en profundidad el plug-in de Dlubal en el entorno de Grasshopper, permitiéndome este vincular materiales, miembros, cargas aplicadas, y sobre todo parámetros como la sección que ha sido uno de los aspectos más importantes ya que es uno de los centros de la iteración y búsqueda de la optimización, por tanto, también considero que lo he cumplido, esto me ha permitido obtener análisis integrales y coherentes.

- **Implementar algoritmos genéticos para la optimización estructural.**
  He desarrollado bucles iniciales que representan la población inicial y las dos primeras generaciones entre individuos aleatorios y a su vez he implementado el cálculo del fitness en una etapa inicial de la aplicación creada. Luego, en el modelo final han sido

aplicados de lleno unificando operadores genéticos a estas poblaciones iniciales y cálculo de fitness del que hablábamos, permitiendo crear un algoritmo de código genético completo, tras probarlo en una multitud de casos y pruebas, se ha comprobado que es eficaz y que cumple su función con creces, doy por tanto este objetivo por superado ya que mi aplicación consigue comparar estructuras y usar estos operadores para hallar el mejor individuo.

- **Automatizar el proceso de diseño de vigas simples y celosías.**
  He configurado estas aplicaciones para permitir el cambio de parámetros desde la interfaz visual, generando automáticamente vigas simples y celosías. Esto puede darse tanto siendo modificadas desde las configuraciones iniciales moviendo un par de sliders como dándole al botón que inicie el bucle automático y las genere instantáneamente probando una tras otra, lo que es cierto indistintamente el caso es que ambas técnicas han automatizado el proceso en sus respectivos modos y han optimizado significativamente el tiempo y esfuerzo requeridos para aquel que use el programa, logrando por tanto este objetivo de automatización.

- **Proporcionar una guía de usuario y documentación técnica.**
  Se ha elaborado una guía detallada no solo del proceso el final, sino de ambos. Permitiendo esto el inicio progresivo de aquel que se está introduciendo por primera vez en este entorno visual, con ella podrá seguir indicaciones detalladas desde lo más simple a lo que tiene complejidad de manera llevadera y comprensiva. Pienso que he desarrollado unas buenas guías de usuario, visuales y explicativas, al alcance del entendimiento de todo aquel con unos mínimos conocimientos. Toda este proyecto y guía siempre están basados en sus respectivas documentaciones técnicas y probados, por tanto, puedo confirmar su validez.

- **Validar la precisión y eficiencia de la aplicación desarrollada.**
  Los modelos de viga simple y celosía dan resultados válidos y eficientes pero limitados a la simpleza de sus modelos. El caso final que desarrolla la aplicación completa ha sido el principal caso de estudio y análisis en la descripción de este proyecto. Realizando con él inmensidad de pruebas que me han permitido verificar su correcto funcionamiento y que cumple con las ideas planteadas en el inicio, además, en el análisis de resultados del modelo he ido realizando modificaciones para distintos casos para comprobar de manera más fiable aun si fuese posible la veracidad de este y el cómo utilizarlo eficiente y eficazmente.

- **Fomentar la innovación y el uso de nuevas tecnologías en el diseño estructural.**
  A lo largo de todo el proyecto he promovido activamente la adopción de estas herramientas avanzadas y relativamente novedosas para muchos, además de estas técnicas de optimización. He demostrado los beneficios de la programación visual y de los algoritmos genéticos en la ingeniería civil, animando a otros a adoptar estas innovaciones.

Por tanto, haciendo una recapitulación y vista de todo lo expuesto anteriormente, tengo un alto grado de satisfacción en cuanto a mi cumplimiento de estos objetivos, me habría encantado llevar esto a niveles mucho superiores debido al amplio interés que ha despertado en mí este entorno, pero ha sido una maravillosa base para seguir desarrollándome en un futuro.

5.3. Relevancia y aportaciones.

Para hablar de lo que realmente esta aplicación puede suponer en la actualidad hemos de enfocarnos principalmente en las áreas de la ingeniería estructural y la optimización de procesos de modo general, ya que serán aquellas que pueden ver sus estudios más beneficiados con este proyecto.

Empezando por mencionar una innovación en el propio diseño estructural al introducir el uso de algoritmos genéticos y poder visualizarlo todo de una manera muy intuitiva al alcance de posibles clientes que tuvieran aquellos ingenieros civiles que la usen, facilitando el entendimiento de estos otros, incluso permitiéndoles ajustar algunas variables a su necesidad sin requerir amplios conocimientos informáticos, supone una gran relevancia en negocios.

Además de este entendimiento sencillo que podemos producir en los demás, es de destacar la relevancia de la facilidad que puede suponer al ingeniero trabajar con esta aplicación una vez haya adquirido los conceptos básicos, estos se podrían obtener únicamente de analizar la guía del usuario proporcionada en el proyecto. Permitiéndoles además poder adaptar este único modelo descrito como ejemplo para una gran variedad de casos, personalizándolo una vez adquiridas las bases de este modelo.

Por otro lado, mencionar la relevancia que supone esta automatización del proceso de diseño de vigas simples y celosías, ya que es usual su cálculo para multitud de construcciones hoy en día, facilita el tener que repetir de cálculos en búsqueda de la solución óptima, ahorrando tiempo y reduciendo esfuerzos, además de optimizar recursos existentes; al simplificar parámetros como el tipo de perfil y la división de la celosía, se obtendrán estructuras con menor peso propio y deformaciones verticales, resultando en un modelo que además será económico.

Como sugerencia de aportaciones que podría significar este mismo, mencionaría una mejora en la formación académica y profesional ya que, una vez profesores dominasen esta conexión entre herramientas como Grasshopper y RFEM podrían enseñar a sus alumnos el cálculo de manera más visual y mejorar la comprensión por parte de éstos. Serviría en cursos de ingeniería civil o arquitectura, demostrando además el uso práctico de los algoritmos genéticos en este campo.

La promoción de nuevas tecnologías es otra importante aportación que pretendo reflejar, durante el desarrollo del programa motivo a la adopción de herramientas avanzadas como Grasshopper, RFEM y Python, demostrando sus inmensos beneficios y apostando por el uso de nuevas metodologías, motivando a otros a explorar e integrar estas en cualquier otro tipo de proyectos.

Finalmente, y como detallo en un apartado posterior, proporcionará una base para futuras investigaciones siendo estas sugeridas por mí en el apartado final de las conclusiones del proyecto. Sin más detalle, pienso que hay muchísimas combinaciones de algoritmos y herramientas que explorar. Además de una amplia mejora del proyecto convirtiéndolo al plano tridimensional, es por eso que los resultados obtenidos y las metodologías aplicadas pueden servir de referencia para la realización de estudios más avanzados en esta materia.

5.4. Limitaciones para el estudio.

A lo largo del desarrollo del proyecto, he podido encontrar diversas limitaciones que han influido en el alcance y la ejecución de este. Estas limitaciones incluyen aspectos técnicos, metodológicos y de recursos, a continuación, detallaré las principales encontradas:

- **Capacidad computacional y Compatibilidad de sistemas.**
  Los softwares utilizados en este proyecto, Grasshopper, RFEM y Python ya de por sí de manera individual son muy potentes y demandan una alta capacidad computacional. No todos los sistemas operativos ni todos los ordenadores son capaces de soportar esta carga de trabajo de manera eficiente y eficaz, ni todos podemos estar al alcance de tener uno que lo soporte, fue el caso ya que al disponer únicamente de dispositivos MAC, tuve grandes dificultades para poder acabar consiguiendo el equipo donde he desarrollado este gran proyecto computacional. Todo esto genera limitaciones también en el rendimiento y velocidad de procesamiento de las simulaciones, porque si estábamos hablando de uno solo, en este proyecto se usan simultáneamente las tres, por lo que triplica la capacidad computacional requerida.
  Pese a ello he solventado los problemas lo máximo posible y trabajado con equilibrio para no sobrecargar los equipos.

- **Integración de herramientas y falta de desarrollo en el ámbito.**
  Aunque existe abundante información sobre el uso y funcionalidad de Grasshopper, RFEM y Python de manera aislada, no he conseguido encontrar documentación previa sobre la integración de estos tres programas a la vez como lo planteado en este proyecto. Es por esto que, a diferencia de otros tipos de proyectos, han surgido dudas que he tenido que ir solventando, tirando de ingenio, creatividad y conocimientos, además de contar con el gran soporte de parte de mi tutor, quien siempre estuvo para apoyar en los momentos de dificultad. Todo esto ha supuesto un claro desafío en el día a día del desarrollo del proyecto, pero con todo ello he conseguido avanzar en situaciones complejas y obtener estos resultados de los que me siento contenta.

- **Tiempo de pruebas y cargas iterativas.**
  El tiempo dedicado a las pruebas y cargas iterativas ha sido considerable. Aunque el tiempo requerido al aplicar este modelo aplicado puede ser breve para el usuario final que lo usa y lo aplica siguiendo una guía, el proceso de desarrollo implica realizar numerosas pruebas y ajustes entre ellas. Cada iteración del modelo requiere tiempo para cargar, ejecutar y evaluar, lo que provoca prolongaciones significativas en el desarrollo del proyecto. Esta realidad ha hecho que el proceso sea más lento de lo esperado, requiriendo mucha paciencia y precisión en cada etapa de la optimización.

- **Limitaciones temporales y circunstancias externas.**
  Las restricciones de tiempo y las circunstancias externas también han sido limitantes para mí en este proyecto, la disponibilidad limitada de tiempo me ha impedido explorar algunas de las áreas con mayor profundidad y realizar algunos ajustes y mejoras que me permitiesen refinar multitud de aspectos. Todo ello podría haber mejorado aún más los resultados, sin embargo, con ello he identificado muchas propuestas para nuevas investigaciones que sugeriré posteriormente como continuación del trabajo realizado.

5.5. Sugerencias para futuras investigaciones.

Para poder seguir avanzando en el amplio e inmenso mundo de la optimización estructural, se han ido identificando varios puntos en los cuales se podrían realizar investigaciones y mejoras adicionales con el objetivo de mejorar y perfeccionar el trabajo ya realizado en este proyecto.

- **Ampliar el proyecto aplicándolo a Celosías en 3D.**
  Una de las sugerencias más prometedoras sería avanzar en el proyecto mediante la construcción de celosías en 3D. Actualmente, este proyecto solo se centra en estructuras en dos dimensiones debido a limitaciones de tiempo, sin embargo, habiendo realizado lo más complicado que es iniciarse en la parte conceptual y el manejo de estos softwares, no debería resultar sumamente complicado avanzar realizando esta mejora. Esto permitiría una implementación más precisa y aplicable del algoritmo genético en la optimización de proyectos de construcción o ingeniería civil ya que estas son las que prevalecen hoy en día.

- **Automatizar y mejorar en profundidad el proceso de extracción del peso de la estructura para el cálculo del fitness en la Celosía.**
  Pese a ser factible el procedimiento seguido para el cálculo del peso de la estructura y su posterior evaluación del fitness, bien es cierto que se podría automatizar la obtención del área de éstas ya que, en mi modelo, realicé unos cálculos de manera externa e hice uso de una lista para escoger el adecuado entre ellos según la iteración que fuese. Es por eso que este pequeño proceso podría estudiarse más detalladamente para poder llevar a cabo unas sencillas mejoras que conseguirían optimizarlo del todo.

- **Añadir una optimización topológica más avanzada.**
  Integrar técnicas de optimización topológica que permitan generar formas más complejas y optimizar la distribución de la masa dentro de la estructura. Estos métodos pueden explorar configuraciones que podrían resultar en diseños aún más ligeros, baratos y eficientes.

- **Mejora de la calidad de la función de fitness.**
  Refinar y hacer más restrictiva la función de fitness para obtener con mayor precisión los objetivos de diseño específicos planteados. Esto implicaría ajustar los criterios de evaluación de los individuos para considerar de manera más detallada aspectos como la resistencia estructural, la estabilidad y otros factores críticos de rendimiento.

- **Grabación automática de resultados y análisis en Excel.**
  Desarrollar un sistema automatizado para la grabación de resultados durante estas pruebas de la aplicación. Esto incluiría la captura automática de datos relevantes y la generación de informes en formato Excel, lo cual facilitaría el análisis y la interpretación de los resultados sin pérdida de tiempo.

Espero ser yo quien pueda implementar estas mejoras en un futuro no muy lejano, pero en caso de no poder, animo a todo aquel lector de este proyecto a iniciarse en ello y continuar con las mejoras planteadas habiendo seguido previamente la base descrita para las geometrías planteadas, que serían el inicio de la mejora y los primeros y más importantes pasos a dar en este proyecto.

6. Bibliografía

El desarrollo de este proyecto ha sido una labor completamente ideada y llevada a cabo por mí, bajo la guía y supervisión de mi tutor. La bibliografía que se presenta a continuación ha sido leída y comprendida, sirviendo como base de información para la redacción de la parte teórica y la definición de aplicaciones y conceptos. Sin embargo, mi trabajo como tal es único y, según tengo entendido, nunca se ha realizado antes de esta manera. Es por esta razón que no cito ni referencio de forma explícita durante el desarrollo del proyecto, ya que la bibliografía utilizada ha servido únicamente como apoyo informativo y no como una fuente directa de referencia. También aclaro que previo al inicio del proyecto, realicé un curso básico tanto de RFEM como de Algoritmos Genéticos en Python, por separado, que me dieron soltura y manejo en estas aplicaciones para posteriormente desarrollar el proyecto de manera eficaz.

[I]. 3dnatives. (s. f.). *Rhinoceros: ¿Qué características tiene el software de modelado 3D?* Recuperado 13 de mayo de 2024, de https://www.3dnatives.com/es/rhinoceros-caracteristicas-software-300320202/#!

[II]. *8 Conceptos Básicos del Análisis Estructural para Estudiantes de Ingeniería Civil.* (s. f.). Recuperado 9 de junio de 2024, de https://aceroselectroforjados.com/blog/conceptos-basicos-del-analisis-estructural/

[III]. *Análisis de sensibilidad aplicado a la dinámica de estructuras mediante la modificación de las propiedades inerciales.* (s. f.). Recuperado 9 de junio de 2024, de http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-62302014000100007

[IV]. ANDER FERNÁNDEZ. (s. f.). *Algoritmo Genético en Python desde 0.* Recuperado 3 de diciembre de 2023, de https://anderfernandez.com/blog/algoritmo-genetico-en-python/#C%C3%B3mo-programar-un-algoritmo-gen%C3%A9tico-desde-cero-en-Python

[V]. *BASE FOR AG PYTHON*. (s. f.). Recuperado 3 de diciembre de 2023, de http://deap.gel.ulaval.ca/doc/default/api/base.html

[VI]. Bendsoe & Sigmund. (2003). *Topology Optmization*. https://edisciplinas.usp.br/pluginfile.php/7961076/mod_resource/content/1/Bensoe_Topology_Optimization__Theory__Methods_and_Applications.pdf

[VII]. *Cargas estáticas 1*. (s. f.). Recuperado 9 de junio de 2024, de https://www.ferrovial.com/es/stem/carga-estatica/

[VIII]. *Conceptos básicos de diseño y análisis estructural.* (s. f.). Recuperado 8 de junio de 2024, de https://www.buildsoft.eu/es/blog/conceptos-basicos-de-diseno-y-analisis-estructural

[IX]. *DEFORMACIONES DE LAS ESTRUCTURAS.* (s. f.). Recuperado 9 de junio de 2024, de https://core.ac.uk/download/pdf/268219527.pdf

[X]. DLubal. (s. f.-a). *Análisis tensión-deformación | Características para el cálculo de superficies y sólidos.* Recuperado 2 de julio de 2024, de https://www.dlubal.com/es/soporte-y-formacion/soporte/caracteristicas-de-los-productos/002113

[XI]. DLubal. (s. f.-b). *Nuevas características y herramientas para un trabajo eficaz.* Recuperado 15 de junio de 2024, de https://www.dlubal.com/es/productos/rfem-software-del-mef/rfem/nuevas-caracteristicas

[XII]. *El método de los elementos finitos.* (s. f.). Recuperado 9 de junio de 2024, de https://biblus.us.es/bibing/proyectos/abreproy/60222/fichero/02+-+Capitulo+2.+El+metodo+de+elementos+finitos.pdf

[XIII]. Elena Kosorouva. (s. f.). *¿Para qué se utiliza Python?* Recuperado 16 de junio de 2024, de https://www.datacamp.com/es/blog/what-is-python-used-for

[XIV]. ENRIQUE DE JUSTO MOSCARDÓ, ANTONIO DELGADO TRUJILLO, ANTONIA FERNÁNDEZ SERRANO, MARÍA CONCEPCIÓN, & BASCÓN HURTADO. (s. f.). *DEFORMACIONES*. Recuperado 9 de junio de 2024, de https://personal.us.es/ejem/wp-content/uploads/2016/02/T08-Deformaciones.pdf

[XV]. *Evolutionary Tools.* (s. f.). Recuperado 3 de diciembre de 2023, de http://deap.gel.ulaval.ca/doc/default/api/tools.html

[XVI]. Felix Garofalo. (2024). *Desbloqueando la Excelencia en Ingeniería: El Rol de la Programación y Python en la Ingeniería Civil*. https://www.inesa-tech.com/blog/desbloqueando-la-excelencia-en-ingenieria-el-rol-de-la-programacion-y-python-en-la-ingenieria-civil/

[XVII]. Heliyon. (2017). *TOPOLOGY OPTIMIZATION IN STRUCTURAL DESIGN*. https://www.cell.com/heliyon/pdf/S2405-8440(17)31014-9.pdf

[XVIII]. ishika kapoor. (2022). *Everything You Need to Know About Rhino 3D.* https://www.novatr.com/blog/ultimate-guide-to-rhino-3d

[XIX]. Ivana R. (2024). *¿Por que aprender Python siendo ingeniero civil?* https://insight-construction.com/por-que-aprender-python-siendo-ingeniero-civil/

[XX]. J. Estupiñan, E. Oñate, & B. Suárez. (1999). *Métodos evolutivos en la Optimización Topológica*. https://www.scipedia.com/wd/images/2/2a/Draft_Samper_361841785_5570_M47optimizado.pdf

[XXI]. Joaquín Amat Rodrigo. (2019). *Optimización con algoritmo genético.* https://cienciadedatos.net/documentos/py01_optimizacion_ga

[XXII]. Juan Miquel Canet. (s. f.). *Resistencia de Materiales y Estructuras*. Recuperado 2 de julio de 2024, de https://portal.camins.upc.edu/materials_guia/250120/2012/Resistencia%20de%20materiales%20y%20estructuras.pdf

[XXIII]. Jun Wen Loo. (s. f.). *Thesis*.

[XXIV]. *LAS CARGAS DINÁMICAS EN LAS ESTRUCTURAS*. (s. f.). Recuperado 9 de junio de 2024, de https://e-construir.com/estructuras/cargas-dinamicas.html

[XXV]. *LAS CARGAS ESTÁTICAS DE LAS ESTRUCTURAS*. (s. f.). Recuperado 9 de junio de 2024, de https://e-construir.com/estructuras/cargas-estaticas.html

[XXVI]. Luis Manuel Villa García. (2014). *Análisis de sensibilidad aplicado a la dinámica de estructuras mediante la modificación de las propiedades inerciales.* http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-62302014000100007#:~:text=Dentro%20de%20la%20din%C3%A1mica%20de,deseado%20en%20el%20comportamiento%20estructural.

[XXVII]. *Método de los elementos finitos.* (s. f.). Recuperado 9 de junio de 2024, de https://www.esss.com/es/blog/metodo-de-los-elementos-finitos-que-es/

[XXVIII]. *Modelamiento estructural*. (s. f.). Recuperado 9 de junio de 2024, de https://ingenieriacivilcivil.com/estructura/modelamiento-estructural/#:~:text=Un%20modelo%20estructural%20es%20una,otros%20aspectos%20de%20la%20estructura.

[XXIX]. *Nociones básicas de optimización topológica: Cómo usar modelos algorítmicos para crear un diseño ligero*. (s. f.). Recuperado 26 de junio de 2024, de https://formlabs.com/es/blog/optimizacion-topologica/

[XXX]. *Optimización estructural*. (s. f.). Recuperado 9 de junio de 2024, de https://www.esss.com/es/blog/metodos-y-tecnicas-para-la-optimizacion-estructural-de-proyectos/#:~:text=La%20optimizaci%C3%B3n%20estructural%20de%20proyectos%20es%20un%20proceso%20que%20pretende,al%20mismo%20tiempo%20ciertos%20par%C3%A1metros.

[XXXI]. *PRINCIPIOS DE ESTABILIDAD ESTRUCTURAL.* (s. f.). Recuperado 9 de junio de 2024, de https://e-construir.com/estructuras/estabilidad.html#google_vignette

[XXXII]. *¿Qué es el análisis estructural?* (s. f.). Recuperado 9 de junio de 2024, de https://skyciv.com/es/education/what-is-structural-analysis/

[XXXIII]. *¿Qué es Python?* (s. f.). Recuperado 16 de junio de 2024, de https://www.tokioschool.com/formaciones/cursos-programacion/python/que-es/?utm_source=google&utm_medium=cpc&utm_campaign=tokio_global_es_gsn_dsa_web&MLL=3179&gad_source=1&gclid=Cj0KCQjw97SzBhDaARIsAFHXUWDywVl7taxb9TBYPXCLrsgZcMjsEx7pw9dzHQpMQnPdRIsHqwxdCpgaAryyEALw_wcB

[XXXIV]. Raúl Kaufmann. (s. f.). *Análisis Estructural I Deformaciones y desplazamientos*. Recuperado 9 de junio de 2024, de https://www.studocu.com/es-ar/document/universidad-nacional-de-rosario/analisis-estructural-i/deformaciones-y-desplazamientos/2316572

[XXXV]. *RESISTENCIA DE LOS MATERIALES.* (s. f.). Recuperado 9 de junio de 2024, de https://www.areatecnologia.com/materiales/resistencia-materiales.html#google_vignette

[XXXVI]. *RESISTENCIA DE MATERIALES.* (s. f.). Recuperado 9 de junio de 2024, de https://www.prodel.es/subareas/resistencia-de-materiales/

[XXXVII]. Rhinoceros. (s. f.). *Rhino en Arquitectura, Ingeniería y Construcción.* Recuperado 13 de mayo de 2024, de https://www.rhino3d.com/es/for/architecture/

[XXXVIII]. studioseed.net. (s. f.). *Diseño paramétrico y generativo con grasshopper.* Recuperado 13 de mayo de 2024, de https://studioseed.net/blog/software-blog/parametric-generative-design-blog/diseno-parametrico-y-generativo-con-grasshopper/

[XXXIX]. u-cursos-cl. (s. f.). *INTRODUCCION AL ANALISIS DE TENSIONES Y DEFORMACIONES DE UNA ESTRUCTURA.* Recuperado 2 de julio de 2024, de https://www.u-cursos.cl/ingenieria/2010/2/CI3202/1/material_docente/bajar%3Fid_material%3D306252

[XL]. *Una guía para la determinación estática, indeterminación, e inestabilidad.* (s. f.). Recuperado 9 de junio de 2024, de https://skyciv.com/es/docs/tech-notes/structural-3d/static-determinacy-indeterminacy-and-instability/