Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Estudios de Postgrado
Maestría en Estructuras

DISEÑO GENERATIVO EN LA INGENIERÍA ESTRUCTURAL

Ing. Erick Alexander Morán Romero
Asesorado por el Msc. Oscar Andrés García Valdés

Guatemala, julio de 2022

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO GENERATIVO EN LA INGENIERÍA ESTRUCTURAL

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

ING. ERICK ALEXANDER MORÁN ROMERO
ASESORADO POR EL MSC. OSCAR ANDRÉS GARCÍA VALDÉS

AL CONFERÍRSELE EL TÍTULO DE

MAESTRO EN ESTRUCTURAS

GUATEMALA, JULIO DE 2022

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martinez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Kevin Vladimir Cruz Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANA	Inga. Aurelia Anabela Cordova Estrada
DIRECTOR	Mtro. Ing. Edgar Darío Alvarez Cotí
EXAMINADOR	Mtro. Ing. Armando Fuentes Roca
EXAMINADOR	Mtro. Julio César Escobar Zeceña
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO GENERATIVO EN LA INGENIERÍA ESTRUCTURAL

Tema que me fuera asignado por la Dirección de la Escuela de Estudios de Postgrado, con fecha 12 de octubre de 2020.

Ing. Erick Alexander Morán Romero

Decanato
Facultad de Ingeniería
24189101- 24189102
secretariadecanato@ingenieria.usac.edu.gt

LNG.DECANATO.OI.497.2022

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Estudios de Posgrado, al Trabajo de Graduación titulado: **DISEÑO GENERATIVO EN LA INGENIERÍA ESTRUCTURAL,** presentado por: **Erick Alexander Morán Romero,** que pertenece al programa de Maestría en ciencias en Estructuras después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANA
FACULTAD DE INGENIERÍA

Inga. Aurelia Anabela Cordova Estrada
Decana

Guatemala, julio de 2022

AACE/gaoc

Guatemala, julio de 2022

LNG.EEP.OI.497.2022

En mi calidad de Director de la Escuela de Estudios de Postgrado de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor, verificar la aprobación del Coordinador de Maestría y la aprobación del Área de Lingüística al trabajo de graduación titulado:

"DISEÑO GENERATIVO EN LA INGENIERÍA ESTRUCTURAL"

presentado por **Erick Alexander Morán Romero** correspondiente al programa de **Maestría en ciencias en Estructuras** ; apruebo y autorizo el mismo.

Atentamente,

"Id y Enseñad a Todos"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
DIRECCIÓN
ESCUELA DE POSTGRADO

Mtro. Ing. Edgar Darío Álvarez Cotí
Director
Escuela de Estudios de Postgrado
Facultad de Ingeniería

Guatemala, 23 de octubre de 2021

MSc. Ing Edgar Álvarez Cotí
Director, Escuela de Estudios de Postgrado
Facultad de Ingeniería
Universidad de San Carlos de Guatemala
Presente

Por este medio informo a usted, que se ha revisado y APROBADO la siguiente TESIS DE GRADUACIÓN titulado: "DISEÑO GENERATIVO EN LA INGENIERÍA ESTRUCTURAL" del estudiante Erick Alexander Morán Romero, quien se identifica con numero de carné 999001015, del programa de Maestría en Estructuras.

Con base en la evaluación realizada, se hace constar que se ha evaluado la calidad, validez, pertinencia y coherencia de los resultados obtenidos en el trabajo presentado y según lo establecido en el Normativo de Tesis y Trabajos de Graduación aprobado por Junta Directiva de la Facultad de Ingeniería Punto Sexto inciso 6.10 del Acta 04-2014 de sesión celebrada el 04 de febrero de 2014.

Por lo anterior, se entrega con la presente, la hoja de evaluación aprobada por el docente del curso y toda la documentación administrativa de respaldo, para su aprobación correspondiente por parte de la Escuela de Estudios de Postgrado.

Sin otro particular

Vo. Bo.
Msc. Ing. Armando Fuentes Roca
Coordinador Área Estructuras
Escuela de Estudios de Postgrado
Facultad de Ingeniería

Escuela de Estudios de Postgrado, Edificio S-11
Teléfono: 2418-9142 / 24188000 ext. 1382
WhatsApp: 5746-9323
Email: informacion_eep@ing.usac.edu.gt
https://postgrado.ingenieria.usac.edu.gt

Guatemala, 23 de octubre del 2021

MSc. Ing Edgar Álvarez Cotí
Director, Escuela de Estudios de Postgrado
Facultad de Ingeniería
Universidad de San Carlos de Guatemala
Presente

Por este medio informo a usted, que, como ASESOR, he revisado y aprobado la siguiente **TÉSIS DE GRADUACIÓN** del (la) alumno (a):

Carné:	999001015
Alumno:	Erick Alexander Morán Romero
Maestría:	Estructuras
Título de la Investigación:	Diseño Generativo en la Ingeniería Estructural

En este sentido, extiendo el Visto Bueno correspondiente, para el cumplimiento de requisitos de la Escuela de Estudios de Postgrado, de la Facultad de Ingeniería, de la Universidad de San Carlos de Guatemala.

Asimismo, se hace constar que el estudiante cumplió con el pago de los honorarios profesionales de asesoría de dicha tesis de graduación.

Sin otro particular

Msc. Óscar Andrés García Valdés
Asesor

Ing. Andrés García
Ingeniero Civil
Col. 15,737

ACTO QUE DEDICO A:

Dios	Por darme la vida, regalarme las capacidades cognitivas tan valiosas que poseo y por permitirme llegar a este momento cumbre de mi formación profesional
Mi madre	Por ser el pilar más importante de mi vida y por demostrarme siempre su cariño y apoyo incondicional
Mi hermana y mis sobrinas	Por ser parte de mi vida y representar la unidad familiar.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala y Escuela de Estudios de Postgrado	Por conferirme deferentemente la oportunidad de realizar una investigación que tuviera como propósito dar luz acerca del diseño generativo en la ingeniería estructural y que sentará las bases para la prospección de su desarrollo futuro en los próximos años.
Mi madre	Por el apoyo irrestricto y las muestras de afecto que generosamente me brindó durante el transcurso de los estudios de maestría.
Ing. Andrés García	Por contar con su oportuna asesoría y su apoyo.
Mi familia	A mi hermana, mis sobrinas, mis tíos, mis primos y mis abuelos por brindarme su apoyo incondicional y generosidad.
Ing. Julio Carlos Romero **Juan Rodrigo Argueta** **Josué Emanuel Díaz**	Por ofrecerme su apoyo incondicional, firme y constante.
Ramón Ayala, Benito Antonio Martínez Ocasio Y Gabriel Mora Quintero	Por ofrecerme un refugio a través de la música en los momentos estresantes debido a la gran carga cognitiva asociada a la maestría.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

ACI	Instituto Americano del Concreto (American Concrete Institute).
AEC	Del acrónimo en inglés *Architecture, Engineering and Construction.* Relativo a la industria de la arquitectura, ingeniería y construcción.
AGIES	Asociación Guatemalteca de Ingeniería Estructural y sísmica.
AISC	Instituto Americano de Construcción con Acero (American Institute of Steel Construction).
Algoritmo	Un procedimiento computacional bien definido que toma un conjunto de datos, como entrada y produce algún conjunto de datos, como salida.
ASCE	Sociedad Americana de Ingenieros Civiles (American Society of Civil Engineers).
ASTM	Sociedad Americana de Pruebas y Materiales (American Society for Testing and Materials).

Combinación de Cargas	Conjunto de fuerzas mayoradas por su respectivo factor según la metodología empleada en el análisis.
Deflexión	Elemento plano conformado por lámina metálica, concreto reforzado y miembros horizontales.
Estructura de datos	Forma de organizar los datos para que las operaciones deseables sean rápidas.
Largueros	Vigas de techo con claros entre armaduras.
Nodos	Lugares en una estructura donde se conectan los elementos.
Plugin	Complemento de software que se instala en un programa con el objetivo de mejorar sus capacidades.
Programación	La programación es el proceso de tomar un algoritmo y codificarlo en una notación, un lenguaje de programación, para que pueda ser ejecutado por una computadora.
Python	Lenguaje de programación interpretado, orientado a objetos y de alto nivel con semántica dinámica.

RESUMEN

Comparado con las anteriores décadas, la ingeniería estructural y su práctica cuenta con computadoras con un poder informático cada vez mayor, algoritmos analíticos muy sofisticados, y técnicas de visualización que renderizan los datos analíticos de manera que sean compresibles de inmediato y brindando una retroalimentación visual relevante al ingeniero. El resultado neto es que, en un proyecto dado, con las restricciones típicas de tiempo, es ahora posible evaluar mucho más propuestas de diseño y ganar percepción técnica sobre el trabajo que se está desarrollando.

Esto pone en evidencia que el entorno de tecnológico en la industria de la arquitectura, ingeniería y construcción cambia constantemente y se vuelve cada vez más complejo. Como consecuencia, los profesionales de la ingeniería estructural, en aras de permanecer relevantes en la práctica, no solo deben aclimatarse a este entorno tecnológico, sino que deben estar en constante formación para explorar, evaluar y capacitarse en las herramientas digitales de vanguardia. Además, en algunos casos, se necesita adquirir conceptos nuevos como el *visual scripting,* el pensamiento algorítmico, entre otros.

Este trabajo de investigación, titulado *Diseño Generativo en la Ingeniería Estructural,* aborda tales desafíos que enfrentan los profesionales del diseño en el uso de la tecnología, ofreciendo una clara exposición del tema y, al hacerlo, abriendo un camino hacia su aprendizaje exitoso, mediante la presentación de los procesos y las tecnologías involucradas, así como a través de la exposición de un caso de estudio relevante.

PLANTEAMIENTO DEL PROBLEMA

La falta de conocimiento sobre tecnologías y metodologías para un análisis y diseño estructural optimizado provoca que no se exploren las diversas posibilidades y configuraciones de una estructura.

Como diseñador de la ingeniería estructural, es posible que una persona se piense entre dos y tres opciones al momento de abordar el diseño estructural de un edificio o una parte de éste. Aunado a esta limitación en el número de opciones de diseño, está la falta de tiempo que se requiere al evaluar y analizar cada una de ellas. También, no debe olvidarse el esfuerzo asociado a cada uno de estos procedimientos, ya que el diseño estructural es un proceso cíclico y de refinamiento continuo.

Todos estos factores, que juegan en contra de la exploración, evaluación, simulación, modelado y análisis de configuraciones estructurales distintas, llevan al ingeniero estructural a abordar el tema del diseño con un enfoque limitado en cuanto al número de alternativas a considerar en sus flujos de trabajo. Como resultado, tal profesional, se queda con la primera opción que cumpla con los objetivos y las restricciones del proyecto, sin saber si podría existir una opción mejor o óptima basada en factores económicos, de constructibilidad, de desempeño o cualquier otro que tenga impacto significativo en el proyecto.

Además de la problemática anteriormente expuesta, la dirección que toma la industria lleva a un lugar donde los proyectos son cada vez más complejos y exigentes. Aquí es donde la ingeniería estructural le abre las puertas al diseño generativo, pues, con ayuda de datos y a través de un enfoque algorítmico es

posible definir, explorar y elegir diversas alternativas que conduzcan a una solución óptima y satisfactoria.

Por lo anterior, definimos la pregunta principal:

¿Qué tecnologías, metodologías y/o enfoques deberán adoptarse durante los próximos años para lograr diseños estructurales óptimos, teniendo en cuenta todas las posibles alternativas y opciones de diseño?

Además, pueden surgir algunas interrogantes secundarias, que se desprenden de la pregunta principal, como:

- ¿Qué habilidades, talentos y mentalidades se necesitan para resolver el problema a través del diseño generativo?,
- ¿Qué plataformas de software son necesarias?,
- ¿qué desafíos se presentarán en relación con la interoperabilidad necesaria para intercambiar información en las plataformas de software?,
- ¿qué infraestructura de tecnología de la información será necesaria para permitir el diseño generativo?,
- ¿cómo se abordará la gestión del cambio en la implementación de las tecnologías disruptivas asociadas con el diseño generativo?

OBJETIVOS

General

Presentar un enfoque que ayude a optimizar las soluciones adoptadas por los ingenieros estructurales a través de la generación, exploración, evaluación y elección de varias alternativas de diseño con ayuda de software de vanguardia.

Específicos

1. Describir las habilidades tecnológicas y digitales necesarias para abordar los procesos del diseño computacional y generativo, y así poder aplicar la tecnología de una manera más ágil, integrada y significativa.

2. Exponer la importancia que juegan los nuevos enfoques y plataformas de software en los flujos de trabajo del diseño generativo aplicado a la ingeniería estructural.

3. Presentar directrices y pautas para abordar la tecnología que hace posible la automatización, generación, exploración y selección de configuraciones estructurales óptimas.

4. Presentar los fundamentos sobre técnicas y herramientas de software que permitan agilizar y optimizar el diseño estructural.

HIPÓTESIS

Al adoptar un enfoque de diseño algorítmico y basado en datos, se mejorará la eficiencia de los profesionales de la ingeniería estructural, ya que ahora tienen el potencial para explorar, evaluar y analizar una basta cantidad de alternativas y opciones de diseño.

INTRODUCCIÓN

Actualmente, ser un profesional de la ingeniería estructural es un acto de equilibrio, pues muchos de estos profesionales se encuentran abrumados en términos de disponibilidad de tiempo, recursos, aprendizaje de herramientas digitales y espacio mental. Existe, por tanto, una lucha continua por mantenerse al día con las últimas tecnologías, metodologías, flujos y procesos de trabajo. Al mismo tiempo, también necesitan seguir siendo competitivos para alcanzar un factor diferenciador en el mercado, avanzar proyectos complejos y hacer el trabajo de manera eficiente y efectiva.

Con lo anterior en mente, surge la necesidad de investigar las herramientas digitales actuales en el campo de la ingeniería estructural. Al abordar dicho componente tecnológico, se hace evidente que no ha habido un esfuerzo legítimo pedagógico por abordar las nuevas metodologías y plataformas de software, tales como las que habilitan el diseño paramétrico, diseño asistido por algoritmos o diseño generativo.

Habiendo identificado esta brecha en la ingeniería estructural en Guatemala, este trabajo pretende cubrir una de las metodologías emergentes en este campo más innovadoras: el diseño generativo. Abarcando desde los conceptos subyacentes a éste, como la programación visual, algoritmos evolutivos, entre otros, hasta su aplicación práctica en plataformas de softwares especializadas.

Se presentará la importancia de adoptar un enfoque algorítmico y basado en datos en los flujos de trabajo relacionados con el análisis y diseño de

estructuras. En el primer capítulo se definen todos los conceptos y elementos teóricos inherentes al diseño estructural en general y al diseño generativo, por ejemplo, empleo de parámetros, codificación y algoritmos en la generación de modelos 3D susceptibles de análisis, optimización y solucionadores evolutivos, entre otros.

El capítulo dos profundiza en el tema de visual scripting, presentando la función e interacción entre nodos del sistema de las plataformas de programación visual para generar y explorar geometría parametrizada, construir una configuración estructural, cálculo de elementos finitos y algoritmos de optimización.

A través de una descomposición en actividades principales que conforman el diseño generativo, en el capítulo 3, se busca presentar un flujo de trabajo general que podría usarse al adoptar esta metodología en el proceso de análisis estructural.

Finalmente, en el capítulo cuatro y cinco pretende mostrar cómo los conceptos, procesos y herramientas son aplicados en la ingeniería estructural a través de la presentación de ejemplos prácticos, junto con un análisis, evaluación y explicación de los resultados y beneficios obtenidos.

1. MARCO TEÓRICO

1.1. Fundamentos del análisis y diseño estructural

Dada la variedad de términos que se emplean en el campo de la Ingeniería Estructural, y que podrían generar cierta confusión para los que quieran abordar este material de investigación, se comenzará por discutir qué es *análisis estructural* y qué es *diseño estructural*.

El análisis estructural es un proceso de verificación, utilizando el conocimiento de la mecánica aplicada y sus herramientas técnicas, de las dimensiones de una estructura, que de una manera más o menos definitiva ya ha sido definida por un proceso de diseño. (Nageim, 2017)

En el análisis estructural, lo principal es la predicción del desempeño o el comportamiento de una estructura ante las cargas prescritas y efectos externos, tales como movimientos en los apoyos y cambios de temperatura.

Al predecir el desempeño y la respuesta de la estructura ante las cargas actuantes, debe centrar el interés en el valor numérico de características como:

- Esfuerzos, ya sean axiales, cortantes, flexionantes o combinados.
- Deformaciones y deflexiones
- Reacciones en los apoyos

Figura 1. **Resultados de interés en un análisis estructural**

Fuente: elaboración propia, utilizando Mindmanager.

Por su parte, el diseño estructural implica invención artística y dimensionamiento. La invención es la creación de una forma y elección de un sistema estructural, el dimensionamiento es asignar a cada miembro estructural dimensiones adecuadas para la estabilidad, la facilidad de servicio, la idoneidad y la sostenibilidad. (Nageim, 2017)

El proceso de dimensionamiento se lleva a cabo a través de prueba y error, lo que conlleva a un análisis repetido de la estructura y su sometimiento a una serie de iteraciones. Este procedimiento de prueba y error suele tener, como punto de partida, la asignación de ciertas dimensiones de arranque a los elementos estructurales que conforman el sistema. Estas dimensiones preliminares son derivadas de una serie de proyectos estructurales anteriores con los que la estructura que se pretende analizar guarda cierta similitud. Algunas de las pautas de predimensionamiento han sido plasmadas, por ciertos autores

y expertos, en tablas y fórmulas en las que las dimensiones previas se relacionan con el tipo y material del elemento estructural.

Aquí es donde se comienza a asociar el diseño con el análisis estructural. El diseño estructural está encaminado al dimensionamiento y predicción de formas y tamaños estructurales. Estos, se someten a un proceso de verificación por medio del análisis estructural, cayendo así en un proceso cíclico de refinamiento continuo hasta que las dimensiones del proyecto estructural cumplan con los códigos y estándares correspondientes y, también, con las condiciones de carga impuestas a lo largo de su ciclo de vida.

1.2. Proceso iterativo del diseño estructural

Dependiendo del tipo de proyecto, el arquitecto nos debe proporcionar un *layout* o un esquema arquitectónico, que genera a partir de una serie de necesidades prescritas y requisitos funcionales.

El proyectista encargado de la estructura comienza por definir un sistema estructural, tanto para fuerzas gravitacionales como uno destinado para fuerzas laterales, que se acople a la concepción arquitectónica presentada y que cumpla con una serie de restricciones impuestas por la naturaleza y complejidad del proyecto y, por supuesto, con las solicitaciones de carga esperadas. También, es necesario contar con información que indique la aptitud geotécnica del terreno, así como otras propiedades mecánicas, donde se desplantará la estructura.

Antes de comenzar el proceso de análisis de una estructura se requiere conocer el tamaño de todos sus miembros, estos tamaños se determinan a través del diseño estructural. Lo curioso de esto, es que las decisiones que se toman durante el diseño estructural deben basarse en los valores numéricos de las

fuerzas y deformaciones de la estructura que se obtienen como resultado de llevar a cabo un análisis estructural.

Entonces, ¿por dónde se comienza? Se necesita de un análisis para comenzar a diseñar, pero también se necesita de un diseño para proceder con su análisis.

Lo primero es hacer *estimaciones* iniciales de las dimensiones y la forma de los miembros de la estructura, algunos llaman a este proceso: *predimensionamiento*.

Al hacer esto, se podría decir que se encuentra en una fase de diseño preliminar, en la que se define temporalmente el tamaño de los miembros.

Entonces, ¿de dónde salen estas dimensiones preliminares? Estas dimensiones están basadas en proyectos similares que ya hayan sido diseñados exitosamente, en análisis burdos iniciales o, también, en el criterio y la experiencia del ingeniero.

Una vez que se haya encontrado o propuesto estas dimensiones preliminares y estimado las cargas a las que estará sometida la estructura, se comienza el proceso de análisis para determinar las fuerzas y desplazamientos en cada miembro de la estructura.

Estas cantidades, se emplean para calcular los esfuerzos y las deformaciones en cada miembro. Las que se deben comparar con los esfuerzos y desplazamientos permisibles, los cuales, por lo general, están especificados como requerimientos de seguridad y servicio en los códigos como los publicados

por ACI, AISC, AGIES o cualquier otra reglamento o estándar aplicable en el país donde se esté diseñando el proyecto.

Además, como se dijo antes, deben evaluarse los costos de la estructura propuesta para ver si se está dentro de las limitaciones económicas.

Si se cumplen con todas estas restricciones, se puede continuar definiendo algunos detalles de la estructura y efectuando algunos otros cálculos complementarios.

Esto conduce al proyectista a un diseño final, donde se comienza a documentar el proyecto a través de planos, especificaciones, memorias de cálculo y otros documentos que se requieran para realizar el proceso constructivo.

En caso de que no se satisfagan los requerimientos, se deben introducir modificaciones en los tamaños de los elementos estructurales o en el arreglo de ciertas partes de la estructura y comenzar otra iteración.

En resumen, el diseño estructural proviene de una serie de aproximaciones sucesivas en las que cada ciclo requiere un análisis estructural. Los ciclos de análisis-diseño terminan exitosamente cuando se han cumplido satisfactoriamente con las disposiciones y requisitos impuestos por los códigos, normas y estándares adoptados por la autoridad que tiene jurisdicción en el proyecto.

1.3. Disrupción tecnológica en la industria AEC: cómo abordar el proceso de análisis y diseño asistido por computadora

Comparado con el asombroso ritmo de cambio en la informática, la ingeniería biomédica o la nanotecnología, el campo de la ingeniería estructural parece congelado en el tiempo. Así que es un momento desafiante para la ingeniería estructural. Se pide a los ingenieros que hagan más con menos: ofrecer más opciones de diseño con menores costos y menos impacto ambiental, que trabajen de forma más rápida y que, a la vez, mantengan un alto nivel de innovación y calidad. Y tener menos personas que diseñan proyectos más complejos en menos tiempo. (Hanif, 2016)

Pese a que las innovaciones en ingeniería estructural empalidecen comparadas con otras industrias, se observa, como cada vez más, en los países desarrollados, comienzan a escucharse términos propios de las ciencias informáticas en la comunidad de arquitectura, ingeniería y construcción. Ha empezado a presenciarse como algunos ingenieros y arquitectos pioneros y advenedizos se adentran cada vez en el mundo de la ciencia de datos, transfiriendo y aplicando conceptos propios de esta disciplina a la industria AEC.

En estos últimos años, ya no es sorprendente como en las oficinas de diseño y construcción, los profesionales usan de manera consciente y deliberada términos como *machine learning, data science, big data*, algoritmos, *deep learning,* inteligencia artificial, entre otros. para referirse a ciertos procesos o acciones que forman parte de sus flujos de trabajo. Como resultado, hay mejoras dramáticas en el flujo de datos, información y conocimiento. Esto, como se dijo antes, ocurriendo en países altamente industrializados, que han logrado una avanzada tecnología e innovación.

Por otro lado, en los países que atraviesan un desarrollo incipiente, como Guatemala, aún existe una gran lucha por adoptar de manera *apropiada,* tecnologías medianamente innovadoras. Esto permite visualizar cómo en la industria guatemalteca, no se ha llegado entender y adoptar el tipo de pensamiento que promueve el trabajar de forma inteligente y no más dura. O, incluso, cuando se ha adoptado e implementado alguna tecnología disruptiva, no se le da el enfoque y tratamiento adecuados.

Por ejemplo, en algunas empresas en las que se ha comenzado a emplear la plataforma BIM, Autodesk Revit, se ha hecho un énfasis enorme en su capacidad para documentar de forma rápida y eficiente un proyecto de construcción, y en la manara tan ágil y cautivadora en la genera modelos tridimensionales. Pero poco se ha abordado con la diligencia debida el sorprende poder que tiene este programa como base de datos relacional y su aprovechamiento en la gestión de información de un proyecto de construcción a lo largo de su ciclo de vida.

A pesar del grado de automatización y eficiencia que ofrecen las herramientas digitales actuales, ha habido una creciente tendencia en optar por la utilización de herramientas que sean más fáciles de usar, aunque esto conlleve al ingreso de datos redundantes y duplicados, a la realización tareas anodinas adicionales y, consecuentemente, a dilapidar nuestro valioso tiempo (un activo profesional irrecuperable una vez gastado). Este hecho, frecuente el horizonte actual de la industria AEC en Guatemala, posiblemente se deba a las curvas de aprendizaje tan empinadas asociadas a los softwares, herramientas y nuevas metodologías de análisis y diseño.

En síntesis, muchos profesionales han optado por seguir utilizando tecnología desactualizada, y en algunos casos casi obsoleta, para evitar

embarcarse en el estudio y capacitación de tecnologías disruptivas cuyas curvas de aprendizaje son notablemente más pronunciadas que las de sus predecesoras.

Encarar el aprendizaje de estas nuevas tendencias tecnológicas no solo requiere aprender el uso de nuevas herramientas de software, también se requiere una nueva mentalidad y de la adquisición de nuevas habilidades. Una de estas habilidades, tanto práctica como mental, es el pensamiento computacional. Aunque esta palabra ha logrado acumular una multitud de definiciones, una excelente definición es la que se puede encontrar en el excelente libro *Computational Thinking* de Peter J. Denning y Matti Tedre, en la que se define el pensamiento computacional como las habilidades prácticas y mentales para diseñar cálculos que hagan el trabajo por las personas y para explicar e interpretar el mundo como un complejo de procesos de información. (Tedre, 2019)

Aunque esta definición es muy completa y proviene del campo de la informática, para propósitos de este trabajo de investigación, se acopla bien a los procesos del diseño asistido por algoritmos, en el que se desarrollan algoritmos para que puedan realizar tareas de manera más automática y eficiente.

En lo que respecta al diseño asistido por algoritmos y al diseño generativo, tema central en esta investigación, no basta con aprender a usar su tecnología que la respalda (Autodesk Revit y Dynamo, Rhino3D y Grasshopper), se necesita también comprender conceptos como algoritmo, pensamiento computacional, pensamiento algorítmico, programación visual, entre otros. Eso hace que sea necesario aumentar el *dossier* de conocimientos y, comenzar a empujar limites cognitivos en aras de dominar los puntos más sutiles de estas tecnologías y metodologías disruptivas.

Sin embargo, no solo los profesionales descuidadamente acomodados en herramientas digitales desactualizadas forman parte de la fuerza que nos empuja cada vez a un entorno *ludita.* Los llamados *millenials* y los profesionales de ingeniería pertenecientes a la generación z, han hecho sonar las alarmas respecto al uso indiscriminado e inadecuado las tecnologías de vanguardia. Este grupo de personas no ha tenido reparo en emplear cualquier cantidad de software para resolver problemas, sin detenerse a pensar qué es lo que está haciendo un programa, cómo lo está haciendo y cómo interpretar los resultados que arroja la plataforma utilizada.

En algunos casos, se han empleado las herramientas para resolver un problema, sin detenerse, dar un paso atrás y pensar si realmente era necesario resolver ese problema en primer lugar. Eso ha hecho que quede relegado el buen juicio profesional, la correcta toma de decisiones y el pensamiento crítico a un segundo plano.

Sin embargo, no todos los denominados *millenials* se circunscriben a esta situación. Algunos ya no se limitan al uso de las herramientas creadas y distribuidas por los fabricantes de software, e impacientes con la vieja guardia, los procedimientos estándar, los negocios habituales y el *status quo*, los profesionales jóvenes y emergentes están impulsando este movimiento hacia la convergencia al tomar el asunto en sus propias manos a través de *plug-ins y add-ons* de software creados por ellos mismos y diseminados libremente. Esto, junto con la codificación y la programación visual habilitada por el usuario, mejora los flujos de trabajo y aumenta enormemente la eficiencia. (Deutsch, 2017)

Toda esta vorágine tecnológica debería ponernos en modo reflexivo, y obligarnos a meditar qué deberíamos de hacer para abordar el tema de manera apropiada. Si se aplica la ley de Moore, la idea de que la capacidad de

procesamiento general de las computadoras se duplicará cada dos años, entonces en la próxima década deberíamos estar listos para presenciar cambios dramáticos en la forma en que se vive y se trabaja juntos. (Cremer, 2020) Esto debería inducir a pensar sobre qué curso de acción tomar para abordar pronta y adecuadamente el aprendizaje e implementación de las tecnologías disruptivas, y, de esta manera, adaptar y preparar al personal más fácilmente y anticiparse a ellas, pero también, tratar de razonar cómo hacerla menos amenazante y más manejable.

Se pueden observar dos problemas críticos que adolece la industria AEC en Guatemala en lo que se respecta a la adopción y el uso de herramientas computacionales: la ausencia de un interés genuino por actualizarse, pronta y oportunamente, con las tendencias tecnológicas que pueden impulsar nuestra industria; y la falta del pensamiento crítico al resolver problemas, tanto con la tecnología actualmente usada en la práctica digital de Guatemala, como con la que pueda usarse en un futuro.

1.4. Pensamiento crítico: una necesidad acuciante en la era de turbulencia digital

El pensamiento crítico es muchas cosas, especialmente la capacidad de: mirar los hechos para llegar a conclusiones sensatas que ayuden a tomar decisiones; analizar la forma de pensar y luego presentar pruebas de las propias ideas, en lugar de presentar opiniones personales como evidencia suficiente; ver ambos lados de un problema, estar abierto a nuevas pruebas que desafíen sus ideas, razonar de manera equitativa, esperar que las afirmaciones estén respaldadas por pruebas, deducir conclusiones de los hechos disponibles e identificar y resolver problemas. (Willingham, 2007)

Con la definición anterior en mente, no solo es deseable, sino que totalmente necesario, que el ingeniero proyectista utilice el pensamiento crítico para evaluar las herramientas digitales que empleará dentro de sus flujos de trabajo relacionados con el análisis y diseño estructural. Antes de usar cualquier herramienta, es preciso saber si es la adecuada para realizar el trabajo que se pretende llevar a cabo, debe evaluarla desde el punto de vista de la interoperabilidad para saber si funciona apropiadamente con otras herramientas en el flujo de trabajo.

1.5. Diseño asistido por algoritmos

Antes de sumergirse a detalle en lo que es el diseño asistido por algoritmos conviene presentar cierta terminología para que el lector no esté propenso a ambigüedades, a medida que se desarrolla la totalidad de la temática que atañe a esta investigación.

1.5.1. Concepto de algoritmo

El viaje a través del diseño asistido por algoritmos comienza con la definición de algoritmo, un término que necesita definirse de la manera más clara posible para que se pueda ir comprendiendo el hilo conductor de este trabajo de investigación.

Un algoritmo es cualquier procedimiento computacional bien definido que toma algún valor, o conjunto de valores, como entrada y produce algún valor, o conjunto de valores, como salida. Por tanto, un algoritmo es una secuencia de pasos computacionales que transforman la entrada en la salida. (H. Cormen, 2009)

Los pasos computacionales mencionados anteriormente, pueden verse como una secuencia de operaciones o instrucciones bien definidas para resolver un tipo específico de problemas o para realizar una tarea particular.

Un ejemplo muy ilustrativo de algoritmo sería la preparación de un *parfait* de granola, en el cual, las instrucciones para su preparación serían la siguientes:

- Colocar un sexto de una taza de arándanos en el fondo de un vaso o una copa grande.
- Cubrir los arándanos con media taza de yogurt turco natural.
- Colocar un tercio de taza de granola encima del yogurt.
- Cubrir la granola con media taza de yogurt turco natural.
- Colocar las fresas encima de todo lo que se encuentra en el vaso hasta ahora.
- Cubrirlo todo con crema batida

En el caso de la preparación de un *parfait*, tenemos 6 pasos finitos que especifican operaciones (como colocar en el paso 1 y cubrir en el paso 2) para resolver un tipo específico de problema: esperar tener un *parfait* en su mano o en su paladar. (Tuckfield, 2021)

Además, un algoritmo también es un conjunto inequívoco de instrucciones adecuadamente definidas. Los algoritmos dependen de las instrucciones ingresadas. El resultado será incorrecto si el algoritmo no está correctamente definido. Por el contrario, un algoritmo correcto resolverá el problema computacional dado. Dicho de otra manera, si los pasos en el *parfait* se invierten o se saltan, las posibilidades de un *parfait* bien preparado se reducen.

No solo las instrucciones dentro de un algoritmo deben estar apropiadamente formuladas, sino que también lo deben de estar el conjunto de datos de *entrada* que las instrucciones necesitan para ejecutarse correctamente. Por ejemplo, en el algoritmo del *parfait* presentado anteriormente, en el paso 2, se necesita un tipo y una cantidad específicos de ingredientes como entrada para pueda realizarse la tarea de forma acertada.

Como se indicó en la definición de algoritmo, se espera que éste, transforme las entradas en una salida bien definida, entonces, se dice que un algoritmo es correcto si, para cada instancia de entrada, se detiene con la salida correcta. (Cormen, 2009).

Figura 2. **Algoritmo como un conjunto de reglas**

Fuente: elaboración propia, utilizando Mindmanager.

Para remarcar la importancia que tienen los algoritmos en este trabajo de investigación, conviene entregar otra definición de algoritmo que esté orientada a la resolución de problemas. A este respecto, George Heinneman, en su obra *Learning Algorithms,* ofrece la siguiente definición de algoritmo: “Un algoritmo es un método de resolución de problemas paso a paso implementado como un programa de computadora que devuelve una respuesta correcta en un período de tiempo predecible” (Heineman, 2020, párr. 1).

1.5.2. Programación

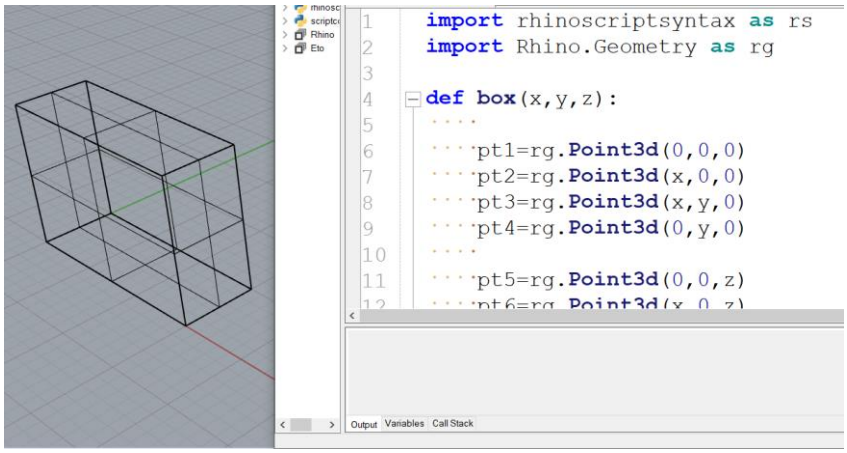
La programación es el proceso de tomar un algoritmo y codificarlo en una notación, un lenguaje de programación, para que pueda ser ejecutado por una computadora. Aunque existen muchos lenguajes de programación y muchos tipos diferentes de computadoras, el primer paso importante es la necesidad de tener la solución. Sin un algoritmo no puede haber programa. (Miller, 2011)

Los algoritmos describen la solución a un problema en términos de los datos necesarios para representar la instancia del problema y el conjunto de pasos necesarios para producir el resultado deseado. Los lenguajes de programación deben proporcionar una forma de notación para representar tanto el proceso como los datos. Con este fin, los lenguajes proporcionan construcciones de control y tipos de datos. (Miller, 2011) Para codificar un algoritmo es necesario un editor específico.

La confección de un algoritmo puede también conducir a la generación de geometría. Para ello, es necesario un editor integrado en un programa de modelado 2D o 3D, como AutoCAD o Rhino3D.

En la figura 3 se puede observar la porción de un algoritmo que genera un prisma rectangular, codificado en el lenguaje de programación Python, creado con ayuda del editor integrado dentro del programa Rhino3D.

Figura 3. **Codificación en el editor de Python de Rhino3D**



Fuente: elaboración propia, utilizando Rhinoceros.

Aunque con los conceptos que se han presentado y definido anteriormente podría abordarse apropiadamente el tema del Visual Scripting, para lo que atañe en esta investigación, se hace necesario definir un término que ha cobrado importancia y popularidad en los últimos años: diseño paramétrico.

1.5.3. Diseño paramétrico

Al hablar de diseño paramétrico, bajo la perspectiva de la industria de la arquitectura, ingeniería y construcción es necesario comenzar la discusión definiendo lo que es BIM. Algunos creen erróneamente que BIM es solo una pieza de software de modelado 3D, o lo que peor, que es sinónimo del empleo del software Autodesk Revit para generar documentos de construcción de forma más rápida y eficiente. Lo que se sabe últimamente es que BIM tiene tantas definiciones como usuarios. Es posible que sea porque el tema ha ido evolucionando y refinándose con el tiempo. Además, como el tema se vuelve cada vez más popular, muchas personas utilizan el acrónimo BIM para hacer lucir más atractivo su trabajo. (Romero, 2020)

Para propósitos de esta investigación, se podría definir BIM como la creación y el empleo de información coordinada y consistente dentro de un proyecto de construcción: información paramétrica utilizada para la toma de decisiones de diseño, producción de documentos de construcción de alta calidad, predicción del rendimiento del edificio, estimación de costos y planificación del proceso constructivo.

BIM es información acerca de la totalidad del edificio y un conjunto completo de documentos de diseño almacenados en una base de datos integrada. Toda la información es paramétrica y, por lo tanto, interconectada. Cualquier cambio en un objeto dentro del modelo se refleja instantáneamente en el resto del proyecto en todas las vistas. Un modelo BIM contiene los elementos de construcción reales del edificio, en lugar de una representación bidimensional de éste que se encuentra comúnmente en los dibujos basados en CAD. BIM ayuda a un equipo de diseño al permitir cambios paramétricos en el diseño de un proyecto de construcción al acelerar el proceso de diseño. Si mueve un muro en una planta, se refleja en las elevaciones, secciones y otras vistas relacionadas en un conjunto de documentación.

Con BIM, se crea un modelo 3D paramétrico utilizado para autogenerar abstracciones de edificios tradicionales, como plantas, secciones, elevaciones, detalles y tablas de planificación. Los dibujos no son conjuntos de segmentos lineales coordinadas manualmente, sino muestras dinámicas e interactivas del modelo. Trabajar en un marco basado en el modelo garantiza que un cambio en una vista se propague a todas las demás vistas del modelo. A medida que se desplaza elementos en la planta, esos cambios aparecen dinámicamente en elevación y sección. Si se elimina una puerta de su modelo, el software elimina simultáneamente la puerta de todas las vistas y se actualiza su tabla de planificación de puertas. (Krygiel, 2008)

Además, los elementos en el modelo paramétrico deben ser tanto inteligentes como semánticos. Inteligentes porque poseen un conjunto de atributos y propiedades que comunican qué son realmente, además de su geometría. Semánticos porque poseen significado y, por tanto, contexto. (Romero, 2020)

En una herramienta de creación BIM, como Revit, el profesional encargado de las características funcionales del proyecto comienza a construir virtualmente los componentes del proyecto, a medida que se lleva a cabo esta acción, Revit crea automáticamente el modelo analítico estructural, el cual, dependiendo del mecanismo de interoperabilidad, puede presentarse un intercambio de información bidireccional entre la plataforma BIM y la herramienta de software que realiza el análisis estructural.

Una vez que se ha expuesto el rol de los parámetros dentro de los flujos de trabajo de la industria AEC, podría definirse el diseño paramétrico como un proceso basado en el pensamiento algorítmico que permite la expresión de parámetros y reglas que, en conjunto, definen, codifican y aclaran la relación entre la intención del diseño y la respuesta del diseño. (Jabi, 2013)

1.5.4. *Visual Scripting* y pensamiento algorítmico

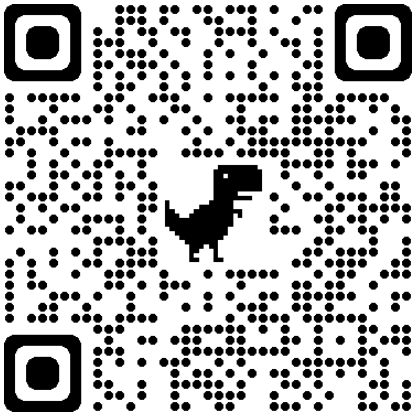
Existen dos conceptos centrales subyacentes al pensamiento algorítmico. El concepto de estructura de datos y el de algoritmo. Una estructura de datos es una forma de organizar los datos para que las operaciones deseables sean rápidas. Como se discutió en la sección anterior, un algoritmo es una secuencia de pasos que resuelve un problema. En ocasiones, se puede hacer algoritmos rápidos sin estructuras de datos sofisticas; otras veces, la estructura de datos correcta puede ofrecer un aumento de la velocidad significativo. (Zingaro, 2020)

Retomando la estructura de datos en el capítulo 3, cuando se está construyendo un algoritmo para la generación de la geometría de una armadura plana.

Como se vio en la sección anterior, existen medios poderosos en los que podemos generar geometría a través de datos y algoritmos. Pero no solo se puede usar un editor de código para ejecutar tales tareas a través de los *scripts* escritos. Herramientas digitales como Dynamo (para Autodesk Revit) y Grasshopper (para Rhino) hacen accesible la interacción con los algoritmos a través de una interfaz visual y más intuitiva. Como resultado, profesionales de la industria con una instrucción, de moderada a avanzada, en programación de computadoras y en algoritmos, pueden aspirar a aprender estas herramientas con poca dificultad.

La figura 4 muestra un código QR en el cual, al acceder desde un dispositivo móvil, se le presentará un video en el que se hace una introducción a Grasshopper y al concepto de algoritmo.

Figura 4. **Código QR para ingresar a un video introductorio acerca de Grasshopper y algoritmos**

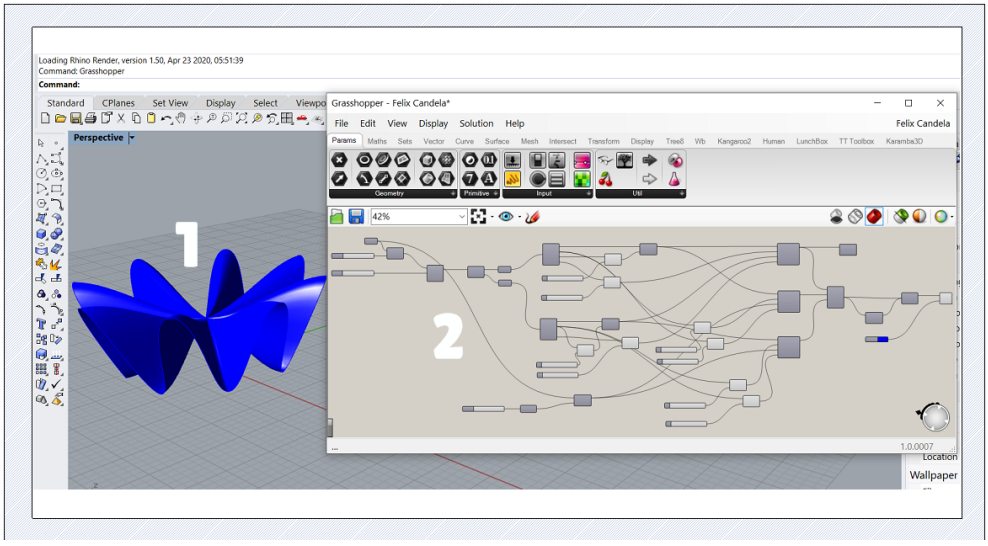


Fuente: elaboración propia, utilizando Google.

Con la programación visual, se definen las instrucciones y relaciones del programa a través de una interfaz de usuario gráfica (o visual). En lugar de escribir texto limitado por sintaxis, conectamos nodos preempaquetados. Los cuales realizan tareas predefinidas. (Dynamo, s.f.)

Para continuar una descripción fluida y comprensible acerca de la programación visual, conviene presentar de forma general los elementos de interfaz que hacen posible la interacción con algoritmos visuales. Para ello, se tomará de ejemplo una captura de pantalla del programa Rhino 3D y Grasshoper. En la Figura 5 se observa un algoritmo visual encaminado a generar una geometría similar a un cascarón de tipo Félix Candela.

Figura 5. **Algoritmo visual para generar una estructura tipo Félix Candela**



Fuente: elaboración propia utilizando Rhinoceros.

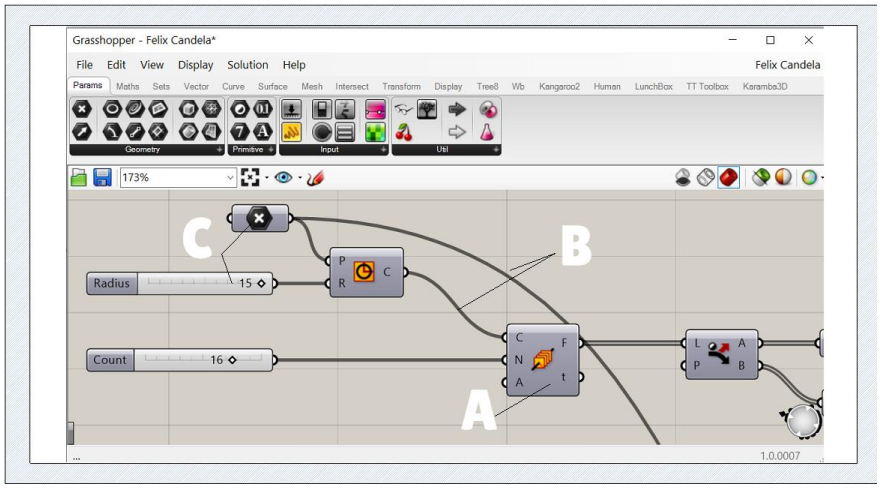
En la figura 5, se muestra el espacio de trabajo del software Rhino y de su herramienta de programación visual Grasshopper, juntos los dos proporcionan el

espacio adecuado para trabajar en el diseño asistido por algoritmos. Este espacio de trabajo, según las etiquetas 1 y 2 colocadas en la imagen, se componen de dos entornos principales:

- El Editor Visual de Algoritmos
- El Entorno de Modelado 3D

Si se observa con más detalle el algoritmo que genera la geometría presentada anteriormente, se puede ver que consta, principalmente, de dos tipos elementos: un elemento tipo nodo A y un elemento tipo cable B (véase la figura 6).

Figura 6. **Configuración de componentes y cables en Grasshopper**



Fuente: elaboración propia, utilizando Rhinoceros y Grasshopper.

Estos nodos o componentes, como se les denomina en algunos softwares) pueden considerarse como un algoritmo predefinido que se encuentra a la espera de datos para poder funcionar y ejecutar una determinada acción; y en algunos

casos, poder generar valiosos datos para que otro(s) nodo(s) puedan hacer uso de éstos.

Los datos de entrada pueden ingresarse conectado un nodo que contiene dichos datos (como el que tiene la etiqueta C en la imagen anterior) con el nodo que ejecuta la tarea. Esta conexión se realiza a través de cables, éstos últimos responsables de transportar la información y los datos de nodo a nodo.

Entonces, una herramienta informática potente y accesible como el visual scripting está destinada a ayudar en la resolución de problemas en ingeniería estructural, por medio de la confección de algoritmos visuales.

En su brillante libro *Think Like a Programer: An Introduction to Creative Problem Solving,* el autor, Spraul (2012) define la resolución de problemas, desde la óptica de la informática, como la capacidad de tomar la descripción de un problema dado y escribir un programa original que realiza un conjunto particular de tareas y cumple con todas las restricciones establecidas para brindarle una adecuada solución a dicho problema. Esta es la tarea que tratamos de llevar a cabo con el diseño asistido por algoritmos. Dado un problema de optimización en ingeniería estructural, tratamos de generar un programa para buscar una solución coherente y satisfactoria.

Sin embargo, además de comprender los fundamentos de la programación, la resolución de problemas, tal y como la define Spraul, requiere adicionalmente otro tipo de habilidades. Esto se debe a que la resolución de problemas es una actividad diferente a la de aprender la sintaxis de programación y, por lo tanto, utiliza un conjunto diferente de músculos mentales. Aprender la sintaxis y la semántica de programación y leer programas, son en su mayoría actividades analíticas del cerebro izquierdo. Escribir un programa original

utilizando herramientas y habilidades aprendidas previamente es una actividad creativa del lado derecho del cerebro. (Spraul, 2012)

En su forma más simple, la creación de *scripts* permite un compromiso significativamente más profundo entre la computadora y el usuario al automatizar aspectos rutinarios y actividades repetitivas, facilitando así una gama mucho mayor de resultados potenciales para la misma inversión de tiempo. (Burr, 2011)

Se debe aclarar que el generar un *script* para abordar un problema no debe tratarse como si fuera una receta de concina. Diferentes problemas requerirán distintos enfoques y distintos *scripts*. Es decir, no debemos «copiar y pegar» códigos de manera irresponsable, aunque se trate con problemas muy similares a otros que ya se han resuelto o que ya han solucionado terceros.

Continuando con la analogía de las recetas de concina, podríamos decir que un excelente chef comprende los ingredientes, los métodos de preparación y los métodos de cocción, y sabe cómo se pueden superponer y mezclar para hacer comidas en las que sus clientes se puedan deleitar. Todo lo que un prominente chef necesita para producir una comida sabrosa y de calidad es una cocina adecuadamente equipada. De la misma manera, un ingeniero que aplica la programación visual con éxito a la resolución de problemas comprende la sintaxis del lenguaje, los marcos de aplicación, los algoritmos y los principios de ingeniería estructural y sabe cómo se pueden combinar para crear grandes programas y, consecuentemente llevar a buen puerto un proyecto estructural.

Podría concluirse, entonces, que adaptarse a este nuevo entorno tecnológico y a sus flujos de trabajo asociados requiere aprender a pensar simultáneamente en varios frentes y desde varios puntos de vista. Lo cual requiere que ampliemos nuestro abanico de conocimientos y habilidades: que

van desde la programación visual hasta el pensamiento computacional, que es precisamente el tema que se discute en las siguientes secciones y en los próximos capítulos

1.6. Software empleado en el diseño asistido por algoritmos

Dependiendo de la industria de la que se esté hablando y los softwares involucrados en aquélla, podrían nombrarse una gran cantidad de plataformas de programación visual. Para lo que nos atañe en este trabajo, mencionaremos dos plataformas de programación visual que han ganado mucha popularidad dentro de la comunidad de usuarios en la industria AEC. Estas plataformas son Dynamo, integrada en el software Autodesk Revit; y Grasshopper, integrada en el programa Rhino 3D. A continuación, se describen a grandes rasgos ambas plataformas.

1.6.1. Dynamo

Dynamo es una herramienta de programación visual accesible tanto para programadores como para no programadores. Brinda a los usuarios la capacidad de guiar visualmente el comportamiento, definir piezas lógicas personalizadas y secuencias de comandos utilizando varios lenguajes de programación textual. (Dynamo, s.f.)

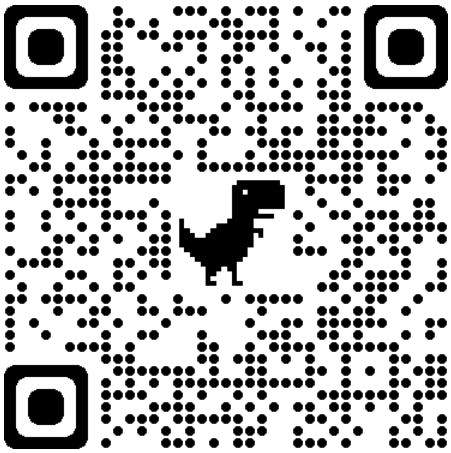
1.6.2. Grasshopper

Grasshopper, una herramienta para el paquete de modelado 3D de Rhinoceros, es un editor de algoritmos gráficos que aprovecha las funciones existentes de Rhino. Grasshopper ofrece nuevas formas de expandir y controlar los procesos de modelado y diseño 3D, incluida la automatización de procesos

repetitivos; generar geometría a través de funciones matemáticas; realizar cambios rápidamente en modelos complejos; y la creación de formas complejas mediante repeticiones de geometría simple. Grasshopper no requiere conocimientos de programación ni de secuencias de comandos, pero aún permite a los diseñadores un alto grado de flexibilidad para crear formas simples y complejas. (ADIT AUTOR, 2018)

La siguiente figura presenta un código QR que lo redirecciona a un video de Youtube en donde se entrega una introducción a la interfaz de usuario de Grasshopper.

Figura 7. **Código QR para ingresar a un video introductorio sobre la interfaz de usuario de Grasshopper**



Fuente: elaboración propia, utilizando Google.

1.6.3. Karamba 3D

Karamba3D es un programa de elementos finitos que está totalmente integrado en el entorno paramétrico de Grasshopper, que es un complemento para la herramienta de modelado 3D Rhinoceros. Esto facilita la combinación de

modelos geométricos parametrizados, cálculos de elementos finitos y algoritmos de optimización como Octopus o Galápagos. (Presigner, s.f.)

En síntesis, para confeccionar un modelo geométrico 3D que pueda emplearse posteriormente para un análisis estructural, se puede emplear la plataforma de software Rhino junto con Grasshopper. Una vez creada la geometría de naturaleza paramétrica, debemos traducirla en un modelo analítico resistente cuyas entidades constituyentes sean susceptibles de análisis por elementos finitos, para alcanzar tal objetivo se hace uso del software Karamba3D.

1.7. Diseño generativo

Lo que hace un enfoque generativo es asignarle a una computadora la tarea de explorar el diseño de forma semiautónoma y luego informar al diseñador qué opciones considera prometedoras para un análisis adicional. Este proceso puede potencialmente revelar partes interesantes del diseño y descubrir soluciones de diseño novedosas que de otro modo estarían ocultas. (Pramod, 2019)

El término diseño generativo, o diseño algorítmico, se ha utilizado de diversas formas para describir procesos geométricos virtuales que están altamente controlados numéricamente y restringidos paramétricamente. Originalmente arraigado en las matemáticas de patrones, el diseño generativo permite al diseñador ingresar una serie de relaciones que se aplican a un componente, o serie de componentes, para generar una variedad formas, en algunos casos complejas, o simplificar los procesos de análisis y modelado mediante la automatización de un conjunto de acciones. Las actividades de diseño generativo se entienden generalmente como novedosas, ya que sería

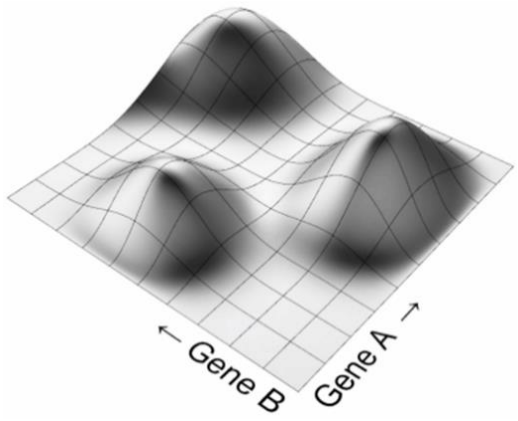
muy difícil para profesionales de la ingeniería y arquitectura llegar a tales formas sin el poder computacional de los paquetes de modelado de información. (Garber, 2014)

1.7.1. Proceso de ejecución de algoritmos evolutivos

Según Nebril, quien se basó en el sitio web oficial de David Rutten las aplicaciones que aplican la lógica evolutiva están destinadas a resolver problemas específicos o son bibliotecas genéricas que permiten a otros programadores llevarlas a cuestas. La herramienta Galápagos, para Rhino y Grasshopper, proporciona una plataforma genérica para la aplicación de algoritmos evolutivos para ser utilizada en una amplia variedad de problemas por los no programadores.

En la figura 8 se muestra el paisaje de aptitud de un modelo particular. El modelo contiene dos variables, es decir, dos valores que pueden cambiar. En Computación Evolutiva nos referimos a las variables como genes. A medida que cambiamos el gen A, el estado del modelo cambia y mejora o empeora (dependiendo de lo que estemos buscando). Entonces, a medida que cambia el gen A, la aptitud de todo el modelo aumenta o disminuye. Pero para cada valor de A, también podemos variar el gen B, lo que da como resultado combinaciones mejores o peores de A y B. Cada combinación de A y B da como resultado una aptitud particular, y esta aptitud se expresa como la altura del paisaje de aptitud. El trabajo del solucionador es encontrar el pico más alto de este paisaje. (Nebril, 2018)

Figura 8. **Paisaje de aptitud para los genes A y B**

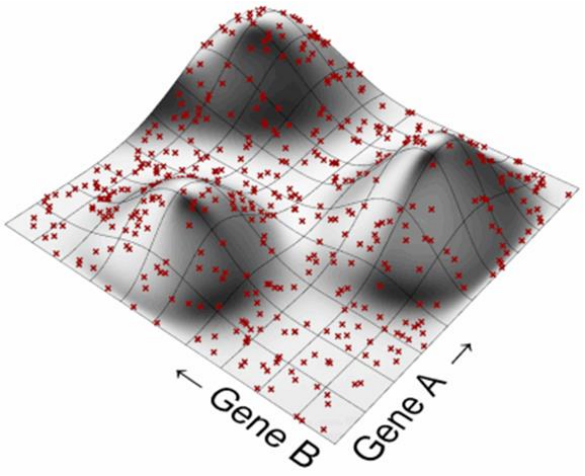


Fuente: Rutten (2011). *Solucionadores Evolutivos.*

Por supuesto, muchos problemas se definen no solo por dos, sino por muchos genes, en cuyo caso ya no podemos hablar de un paisaje en el sentido tradicional. Un modelo con 12 genes sería un volumen de aptitud de 12 dimensiones deformado en 13 dimensiones en lugar de un plano de aptitud bidimensional deformado en 3 dimensiones.

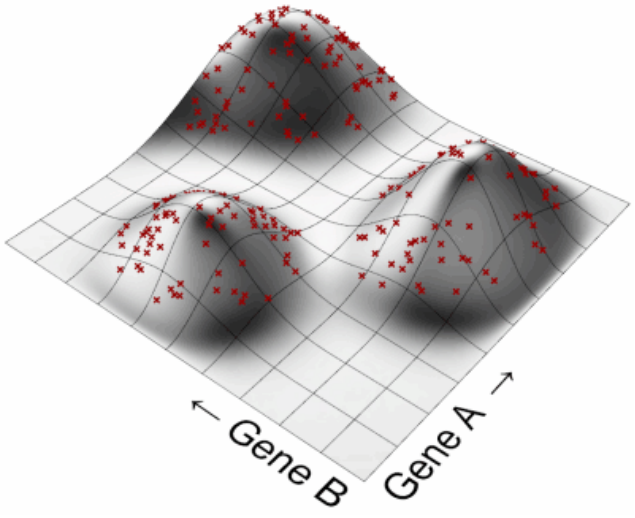
Cuando el solucionador comienza, no tiene idea de la forma real del paisaje de aptitud. Entonces, el paso inicial del solucionador es poblar el paisaje (o espacio modelo) con una colección aleatoria de individuos (o genomas). Un genoma no es más que un valor específico para todos y cada uno de los genes. En el caso anterior, un genoma podría ser, por ejemplo, {A = 0.2 B = 0.5}. El solucionador luego evaluará la idoneidad para todos y cada uno de estos genomas aleatorios, dándonos la distribución mostrada en la figura 9. (Nebril, 2018)

Figura 9. **Distribución de genomas**



Fuente: Rutten (2011). *Solucionadores Evolutivos*.

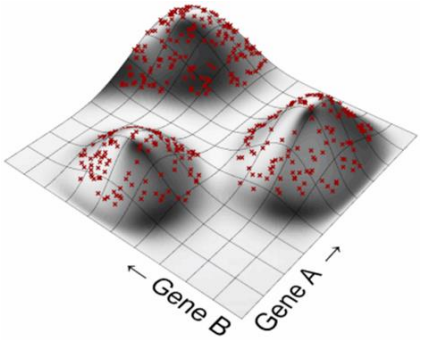
Figura 10. **Énfasis en genomas de mejor rendimiento**



Fuente: Rutten (2011). *Solucionadores Evolutivos*.

No es suficiente simplemente elegir el genoma de mejor rendimiento de la población inicial y dejarlo. Dado que todos los genomas de la Generación Cero se eligieron al azar, en realidad es bastante improbable que alguno de ellos haya ganado el premio gordo. Se deben generar los genomas que exhiban un mejor desempeño en la Generación Cero para crear la Generación Uno. Cuando criamos dos genomas, su descendencia terminará en algún sitio del espacio modelo intermedio, explorando así terreno nuevo como muestra la figura 11. (Nebril, 2018)

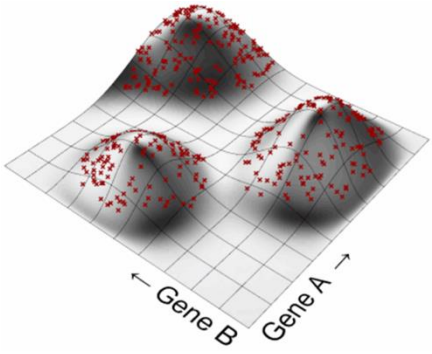
Figura 11. **Generación uno de genomas**



Fuente: Rutten (2011). *Solucionadores Evolutivos*.

Ahora tenemos una nueva población, que ya no es completamente aleatoria y que ya está comenzando a agruparse alrededor de los tres picos de aptitud. Todo lo que tenemos que hacer es repetir los pasos anteriores (eliminar los genomas de peor rendimiento, generar los genomas de mejor rendimiento) hasta que hayamos alcanzado el pico más alto como se ilustra en la figura 12. (Nebril, 2018)

Figura 12. **Distribución de genomas en los picos más altos de aptitud**



Fuente: Rutten (2011). *Solucionadores Evolutivos.*

2. FUNDAMENTOS DEL VISUAL SCRIPTING

2.1. Manejo de datos

En los programas convencionales con los cuales trabajan los profesionales de la ingeniería estructural (ETABS, Sap2000, Autodesk Robot Structural Analysis, Staad Pro, entre otros), la generación de un modelo analítico virtual se basa fuertemente en la presencia pseudo-física del mouse y el teclado en el entorno digital. En contraste, el desarrollo de un modelo analítico dentro del diseño asistido por algoritmos se hace en función del establecimiento de asociaciones lógicas y conceptuales entre geometría y expresiones matemáticas. Esto decir, se tiene poca intervención directa con la manipulación de la geometría mediante el cursor, en su lugar, el procedimiento se basa en el manejo de datos y algoritmos visuales.

De la definición que se dio en la sección anterior sobre el concepto de algoritmo, y dada la importancia que tienen los datos, como se ha visto con anterioridad, para efectos de este trabajo de investigación podríamos volver a definir el concepto de algoritmo como:

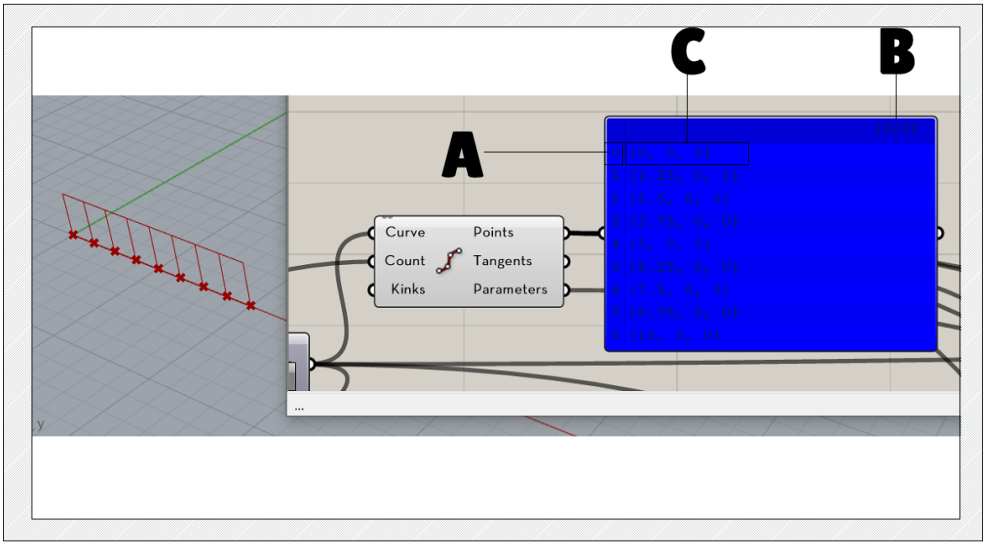
Un procedimiento computacional bien definido que toma un conjunto de datos, como entrada y produce algún conjunto de datos, como salida, obteniéndose la solución a un problema planteado con anterioridad.

Es evidente como desde la óptica de esta definición, la recopilación, ingreso, flujo, procesamiento, modificación y extracción de datos, comienzan a

realzar el protagonismo que tienen los datos en un proceso de diseño asistido por algoritmos.

Esto nos lleva a dar una descripción sobre la forma en que se manipulan los datos dentro del entorno de Grasshopper. Por lo que, en primera instancia, conviene abordar la estructura de datos dentro de los componentes que forman parte de un algoritmo visual y algunos componentes muy recurrentes en la manipulación de datos. Para ello, en la figura 13 se muestra un componente panel conectado a la salida del componente divide curve. Lo que se muestra en el panel son las coordenadas de los nodos del cordón inferior de la armadura tipo *Vierendel* mostrada.

Figura 13. **Datos mostrados en un componente panel**



Fuente: elaboración propia, utilizando Grasshopper.

La parte A de la figura anterior indica el índice de los elementos contenidos dentro de la lista. La parte B corresponde a la ruta, un tipo de nivel jerárquico que contiene listas. La parte C indica el *datum* o elemento de la lista. Éstos se indexan

en la lista con un número natural que va desde 0 hasta i, donde i representa el número total de datos en la salida. En este caso, cada elemento representa la coordenada de un nodo perteneciente al cordón inferior de la armadura analizada.

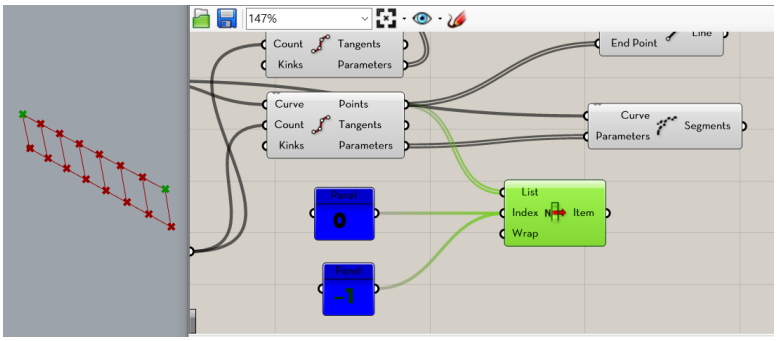
En ocasiones, no es indispensable trabajar con la totalidad de datos que puede arrojar la salida de un componente, por lo que puede ser necesario extraer una porción de estos, de tal manera que obtengamos únicamente los elementos de la lista que necesitamos manipular para ingresar a otro componente y así, realizar una acción en específico. Aquí es donde entran en escena componentes como *List Item*, *Cull Index* y *Cull Pattern.*

2.1.1. Funcionamiento de *List Item*

Con ayuda del componente *List Item* podemos seleccionar de forma rápida y fácil elementos dentro de una lista, indicándole al componente los índices correspondientes a los elementos a tratar.

Por ejemplo, la siguiente imagen muestra un algoritmo para generar una armadura *Vierendel*. Suponiendo que se quiera trabajar con los nodos extremos del cordón superior (por ejemplo, para una asignación de cargas concentradas), necesitamos extraer estos dos elementos de la lista que los contiene, en este caso, la lista correspondiente a la salida *Points* del componente *Divide curve*. Acá es donde figura el componente *List Item*. Conectamos la lista a filtrar a la entrada *List* y especificarlos índices de los elementos que se quiere recuperar (en este caso, 0 y -1, por tratarse de los elementos extremos de la lista). Como resultado, la salida de *List Item*, contiene los datos que se necesitaban.

Figura 14. **Uso del componente *List Item***



Fuente: elaboración propia, utilizando Grasshopper.

2.1.2. Funcionamiento de *Cull Index*

Este componente también realiza una acción de filtrado, pero de forma inversa al componente *List Item*. Es decir, los elementos que indiquemos dentro de la lista, a través de su índice respectivo, quedarán descartados, por lo que la salida del componente *Cull Index* no los tendrá.

Continuando con el ejemplo de los nodos del cordón superior de la armadura, al conectar la salida *Points* del componente *Divide Curve* a la entrada *List* del componente *Cull Index*, y los índices de los elementos a descartar a la entrada *Indices*, lo que se obtendrá como salida del componente *Cull Index* será una lista que contiene todos los elementos asociados a los nodos del cordón superior, a excepción de los nodos extremos. Como se aprecia en la siguiente figura.

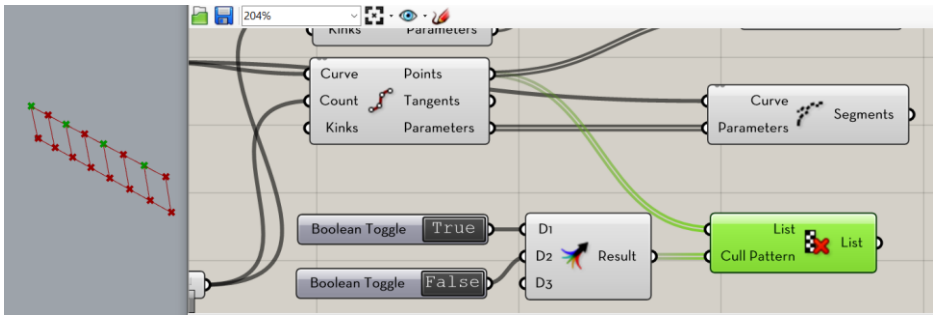
2.1.3. Funcionamiento de *Cull Pattern*

Como los componentes descritos anteriormente, este también ejecuta una acción de filtrado. Éste se hace a través de patrones repetidos de valores

booleanos, los cuales seleccionan y excluyen, según si el valor es verdadero y falso, respectivamente.

Estos valores de verdadero y falso, que fijan los patrones de filtrado, podemos definirlos a través del componente *Boolean Toogle* y el componente *Merge.* La siguiente figura muestra un ejemplo de este tipo de filtrado.

Figura 15. **Uso del componente *Cull Pattern***



Fuente: elaboración propia, utilizando Grasshopper.

2.2. Generación de curvas

Geométricamente, se podría decir que un modelo analítico consta de tres tipos de elementos: puntuales (nodos y apoyos), lineales (vigas, columnas, riostras, entre otros.) y superficies (diafragmas, muros, entre otros.). Por lo tanto, se requiere un conocimiento sólido sobre cómo generar, configurar y aunar estos elementos geométricos para poder darle vida a un modelo 3D susceptible de ser analizado.

Cuando se emplea Rhino-Grasshoper junto a su plugin Karamba3D, primero es necesario generar una geometría paramétrica del modelo. En ocasiones, si ya se ha construido el modelo arquitectónico en Rhino-

Grasshopper, y dependiendo del nivel de parametrización que este posea, podemos utilizar su geometría como excelente punto de partida para comenzar a construir el modelo analítico. Se comienza por ver qué algoritmos son los que dominan la generación de geometría, evaluar su nivel de parametrización, ver qué geometría podría ser reutilizada (junto con su *script* asociado) y por último añadir aquellos elementos que sean únicamente estructurales y que, por obvias razones, no existen en el *script* original, teniendo en cuenta el nivel de control paramétrico que debe concedérseles en función del juicio y criterio del proyectista.

Una vez terminado el proceso anterior, deberá consolidarse toda la geometría, de importancia estructural, en un modelo analítico susceptible de simulación y análisis. Aquí es donde hace su tan esperada aparición el plugin Karamba3D. Éste posee entre sus funcionalidades, una que es capaz de convertir elementos tipo línea en vigas de sección predeterminada. Además, esta función, que manifiesta a través del componente *Line to Beam*, es capaz de detectar que, si dos líneas se encuentran en un punto común, las vigas resultantes estarán conectadas en dicho punto.

Después de haber llevado acciones similares a las descritas en el párrafo anterior en toda la geometría de interés, se hace uso del componente *Assemble Model* para unificar todos los miembros estructurales en un todo que represente lo más fielmente a la estructura a diseñar. En esencia, lo que hace dicho componente es crear un modelo de elementos finitos a partir de entidades dadas (puntos, vigas, soportes, cargas, secciones transversales).

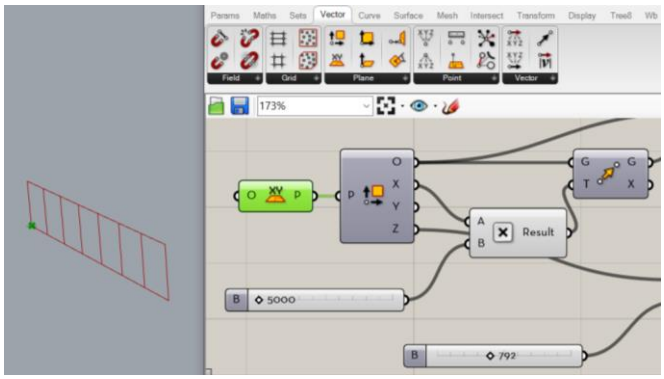
El objetivo de los párrafos anteriores es exponer un panorama general de una parte del flujo de trabajo del diseño paramétrico cuando se utiliza Rhino-Grasshopper de forma conjunta con Karamba 3D. En lo que resta de esta

sección, se describirá la forma de generar la geometría necesaria para crear el modelo analítico. La sección 2.4 está dedicada a descender con más detalle al tema de los componentes que proporciona Karamba 3D para llevar a cabo el análisis y la simulación del modelo estructural paramétrico.

Habitualmente, gran parte de los modelos tridimensionales confeccionados en Rhino-Grasshopper comienzan con la generación de unos cuantos puntos. Éstos pueden generarse con ayuda de algunos de los componentes de Grasshopper como Construct Point; o bien escoger cuidadosamente algunas ubicaciones en el entorno de Rhino y, con sus propias herramientas integradas, dibujar el punto o conjunto de puntos que servirán como partida para el desarrollo de geometría tridimensional. Al seguir esta línea de trabajo, deberá tenerse en cuenta que es imprescindible que todos los puntos creados de esta manera tienen que ser diligentemente almacenados en un componente tipo contenedor en el entorno de Grasshopper.

Otra manera de crear puntos con Grasshopper es a través de la «deconstrucción» de geometría. Por ejemplo, la armadura tipo Vierendeel mostrada en la siguiente figura, empezó a generarse con un punto de origen, que de forma predeterminada se estableció en (0,0,0) al momento de colocar el componente XY Plane en el lienzo. La salida de este componente, un plano, se «deconstruyó» usando el componente *Deconstruct* Plane. La función que tiene este componente es que, cuando se ingresa un plano en su entrada, lo descompone en sus componentes principales (origen, eje x, eje y y eje z). Por lo tanto, el componente ofrece la salida *Origin*, la cual corresponde a un punto en el espacio con sus correspondientes coordenadas. Fue este punto el que se utilizó como nodo y origen de la armadura del ejemplo, y el cual fue utilizado para generar la geometría subsiguiente.

Figura 16. **Uso del componente *Deconstruct Plane***

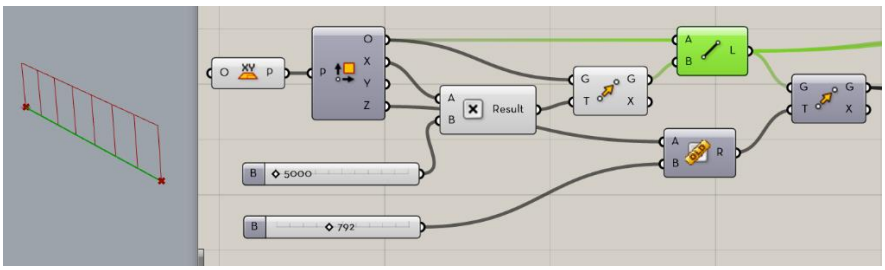


Fuente: elaboración propia, utilizando Grasshopper.

Una vez definidos el origen y otros puntos en ubicaciones convenientes, deberá generarse geometría como rectas o curvas que unan los puntos. Grasshopper cuenta con una amplia gama de componentes para llevar a cabo esta acción, entre los cuales se encuentran *Line, Curve, Nurbs Curve, Polyline, Interpolate,* entre otros.

En la siguiente figura se muestra la creación de un elemento tipo línea empleando el componente Line. Esta línea representa el cordón inferior de la armadura tipo Vierendeel del ejemplo que se ha venido comentado hasta ahora.

Figura 17. **Generación del cordón inferior de la armadura**



Fuente: elaboración propia, utilizando Grasshopper.

En ocasiones, los proyectistas estructurales podrían enfrentarse al diseño de elementos tipo cáscara. Dentro del entorno algorítmico de Grasshopper podemos llevar a cabo el análisis y simulación de estas estructuras generando su geometría por medio de componentes tales como Loft, Extrude, entre otros que tienen como propósito la creación de superficies.

2.3. Transformaciones

En la sección anterior se discutió cómo generar geometría básica previó a transformarla en elementos estructurales y consecuentemente ensamblar el modelo analítico. Pero, bajo determinadas circunstancias, es necesario realizar ciertas transformaciones a la geometría existente para generar nueva geometría y, de esta manera, crear todas las entidades estructurales que se requieren para obtener un modelo analítico.

Existe una ficha en Grasshopper que ofrece una variedad de componentes que ayudan a realizar una diversidad de transformaciones en la geometría previamente desarrollada. Tal ficha posee el nombre de *Transform.*

Dependiendo de la magnitud y complejidad del proyecto arquitectónico pueden sernos de utilidad la inmensa mayoría de los componentes contenidos en la ficha *Transform*. Para una buena parte de las obras estructurales con los que puede enfrentarse un proyectista estructural, las herramientas de transformación a utilizar quedan reducidas a una pequeña porción de componentes, los cuales pueden encontrarse dentro del grupo Euclidian.

La mayoría de los ingenieros estructurales que han empleado cualquier plataforma de software para modelado, análisis o diseño estructural estarán muy

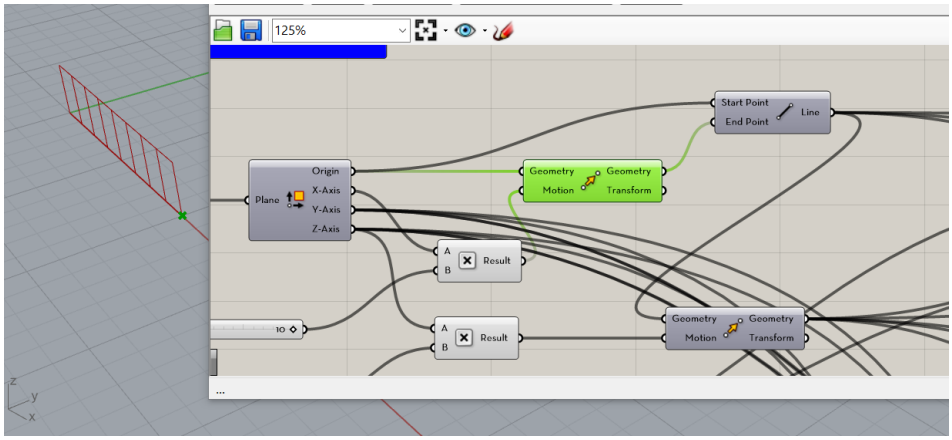
familiarizados con las transformaciones que ofrece el grupo Euclidean, ya que tienen una gran semejanza con las funciones que proveen dichas plataformas.

Otra serie de transformaciones de uso frecuente, son las correspondientes a los grupos Affine y Array.

Dada la simplicidad de estas transformaciones se entregará una explicación sucinta sobre éstas, junto con un ejemplo de aplicación.

Por ejemplo, en la siguiente imagen se muestra un *scricpt* en el que usamos el componente de transformación *Move* para desplazar el punto de origen a partir del cual se creará una armadura Vierendel. Como se puede observar de la figura, además de la entrada de geometría, el componente *Move* necesita de un dato de entrada adicional, el cual corresponde a un vector que entregará la magnitud y dirección del movimiento. En este caso, a través del componente *Multiplication* se proporciona una magnitud de 10 unidades en la dirección *x.*

Figura 18. **Funcionamiento del componente *Move***



Fuente: elaboración propia, utilizando Grasshopper.

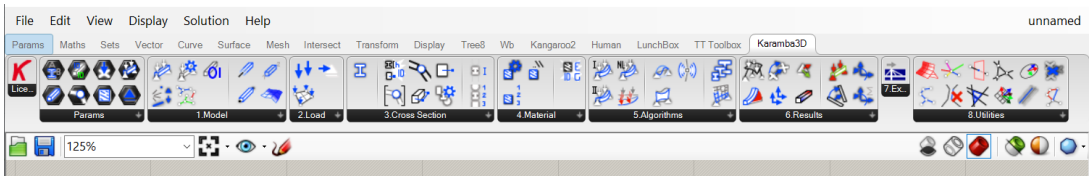
2.4. Componentes de análisis estructurales y de FEM

Al finalizar la confección de un modelo 3D con el grado de parametrización requerido, la siguiente acción a realizar es convertir toda la geometría constituyente de dicho modelo en entidades que puedan tratarse como partes de un modelo analítico resistente. Es acá donde entra en escena el software Karamba3D que, como se mencionó anteriormente, funciona de manera integrada y conjunta con Grasshopper y Rhino.

Al instalar Karamba3D, éste se manifiesta como una nueva ficha dentro de la interfaz de usuario de Grasshopper. Contenidos en esta ficha, están una serie de componentes que nos ayudan con el análisis de elementos finitos de la estructura. Desde la conversión de un modelo arquitectónico 3D a uno analítico, hasta componentes cuya función es calcular esfuerzos y deformaciones.

Se deberá contar con un modelo paramétrico que describa lo más fielmente la geometría de la estructura antes de comenzar a utilizar Karamba3D, ya que es la condición *sine qua non* para realizar una simulación y análisis en dicho software. Como se indicó en unas líneas más atrás, dicho modelo se conforma de puntos, líneas, curva, superficies, entre otros. Ahora bien, para propósitos de la construcción de modelos estructurales, Karamba3D añade siete entidades: Model, Element, Element Set, Joint, Load, Cross Secttion, Material y Support.

Figura 19. **Ficha de Karamab3D en la interfaz de Grasshopper**



Fuente: elaboración propia, utilizando Grasshopper.

2.5. Componentes de solucionadores evolutivos

Una de las metas que persigue un ingeniero estructurista al encontrarse inmerso en el proceso de diseño de una estructura es la de la optimización. El proyectista se empeña en buscar las dimensiones, materiales y detalles de los elementos estructurales, con base en los resultados de esfuerzos y de formaciones, de modo tal que la estructura tenga el mejor desempeño estructurales dentro de las restricciones económicas, y de otra índole, que se encuentren dentro de las necesidades del proyecto.

Además de apoyarse en el uso de algoritmos visuales para realizar un análisis de elementos finitos a una estructura determinada, el proyectista tiene de la oportunidad de llevar a cabo un proceso de optimización con el uso del componente integrado Galápagos Evolutionary Solver.

La herramienta Galápagos, proporciona una plataforma genérica para la aplicación de algoritmos evolutivos para ser utilizada en una amplia variedad de problemas por los no programadores. Esta herramienta y su funcionamiento ya se ha explicado en detalle en la sección 1.7.1.

2.6. Componentes de programación Python

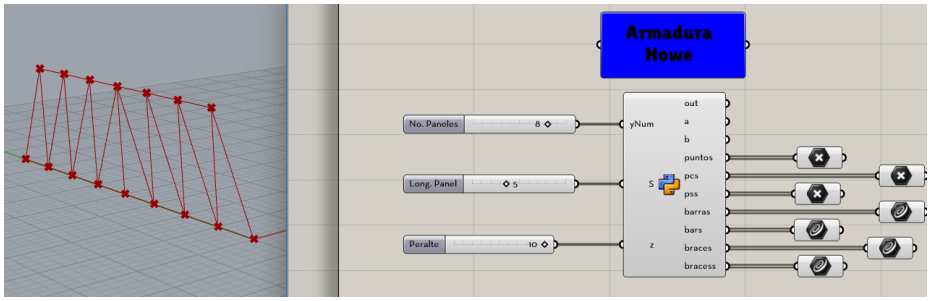
Además de presentar un potente componente que utilice un lenguaje de programación como Python para la automatización de ciertas tareas (como la creación de una armadura), se pretende hacer énfasis en la utilidad de la codificación en la industria AEC. Esto, debido a que la codificación le abre las puertas, no solo a la automatización tareas dentro de una herramienta de software, sino también, a la mejora de éstas y, en el mejor de los casos, a la

creación se software, ya sea de forma individual o a través de la colaboración con desarrolladores calificados, dando lugar así, al software hecho en casa.

De lo anterior, se podría inferir que, a medida que los lenguajes de programación específicos se vuelven menos misteriosos e intimidantes y más fáciles de dominar, el software hecho en casa probablemente se convertirá en una parte familiar de la cultura del diseño. Este movimiento de la comunidad de diseño para tomar un papel activo en la producción de código trasciende las limitaciones de las herramientas de software prefabricadas. (Silver, 2006)

Una forma práctica de aplicar la programación tradicional dentro de Grasshopper es a través de componentes como GhPython Script. Por ejemplo, la Figura 20 muestra uno de estos componentes, el cual se emplea para generar una armadura tipo Howe. Nótese la personalización del componente para incluir *Sliders* como entrada para obtener un control paramétrico sobre la geometría de la armadura (peralte, longitud de panel y número de paneles).

Figura 20. **Componente GhPython**

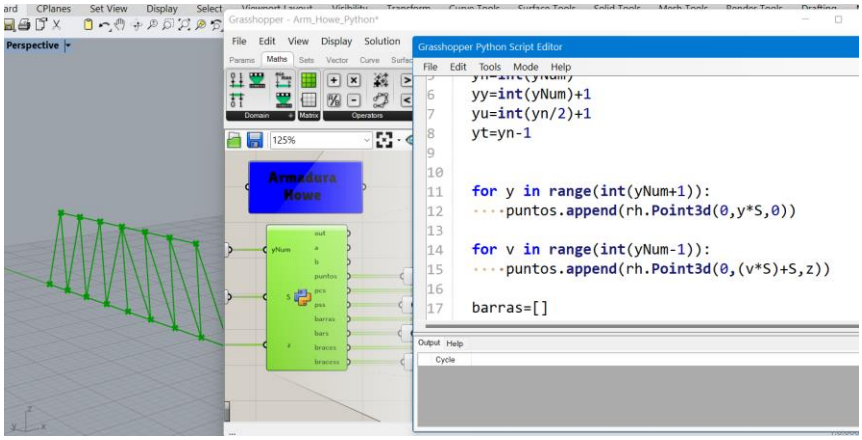


Fuente: elaboración propia, utilizando Grasshopper.

Al dar doble clic dentro del componte GhPython se abrirá una ventana denomina Grasshopper Python Script Editor, dentro del cual se ofrece un espacio para codificar en el lenguaje de programación de Python. La siguiente figura

muestra una porción del script que se desarrolló para generar la armadura paramétrica tipo Howe que se mencionó en el párrafo anterior.

Figura 21. **Editor de *Scripts* de Python y Grasshopper**



Fuente: elaboración propia, utilizando Grasshopper.

2.7. Otros componentes

Los componentes descritos hasta ahora están destinados a ayudar a usuario en el flujo de trabajo básico de diseño generativo. Desde el proceso de modelado, pasando por el planteamiento y análisis estructural, hasta el proceso de optimización a través de algoritmos evolutivos. Pero la realidad, es que existe una gran cantidad de *plugins* que pueden apoyar al proyectista en sus flujos de trabajo de análisis y diseño estructural.

Debido a la proliferación de usuarios de Grasshopper en países como Estados Unidos, Australia e Inglaterra, por mencionar algunos, se ha formado una amplia comunidad de personas que utilizan de forma asidua el programa y que, además, han contribuido de forma deliberada al desarrollo de *plugins* para esta plataforma. Entre algunos de estos encontramos unos de descarga y uso

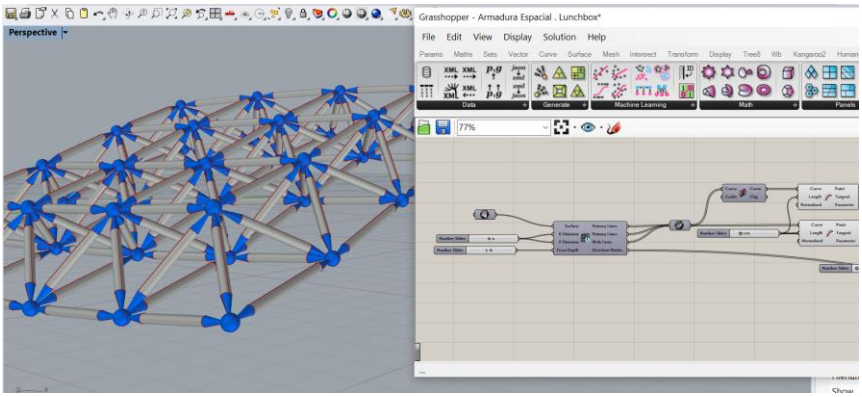
gratuitos. Otros, por el contrario, se ofrecen a través de licencias pagadas. Dada la temática de esta investigación, nos enfocaremos en un reducido grupo de complementos que pueden agilizar y añadir mejoras a nuestros flujos de trabajo.

2.7.1. Lunchbox

Tal y como se define en su sitio web, LunchBox es un complemento para Grasshopper, el cual proporciona componentes simples para realizar operaciones comunes de modelado paramétrico, incluida la teselación de superficies, el análisis de geometría, la exploración de formas matemáticas, estructuras y flujos de trabajo de datos. (LLC, 2020)

Aunque Lunchbox ofrece una gran variedad de componentes para distintas aplicaciones, desde la óptica de un proyectista de estructuras, los que ofrecen una ventaja pasa sus flujos de trabajo son los que se encuentran bajo la categoría Structure, los cuales facilitan enormemente la creación, rejillas de riostras 2D y armaduras espaciales. En la figura 22 se muestra un ejemplo de una armadura espacial creada con el componente Space Truss Structure.

Figura 22. **Ficha de Karamab3D en la interfaz de Grasshopper**



Fuente: elaboración propia, utilizando Grasshopper.

2.7.2. KarambaIDEA

Al hablar del uso de plataformas digitales que forman parte de la caja de herramientas que un ingeniero estructurista usa como parte de su flujo de trabajo en el diseño estructural, una cuestión que tarde o temprano debe abordarse (de preferencia de forma anticipada), es la de la interoperabilidad con otras plataformas de software, ya sea que éstas formen parte exclusiva de sus procesos de análisis, diseño o documentación, o bien, aquellas que utilizan otros profesionales que colaboran con el proyectista estructural.

Dado que la cuestión de la colaboración de equipos interdisciplinarios que participan en un proyecto de la industria AEC se sale del tema tratado en este trabajo de investigación, de acá en adelante, cuando se hable acerca de interoperabilidad, se hará alusión únicamente a las plataformas de software pertenecientes al flujo de trabajo de un ingeniero estructurista. Una de estas plataformas es IDEA Statica, un software para el diseño de conexiones de acero, análisis de miembros y verificación de códigos de detalles de concreto.

KarambaIDEA es un complemento para Grasshopper de código abierto, que puede vincular Karamba3D con la conexión IDEA Statica.

KarambaIDEA facilita la interacción con IDEA StatiCa, lo que permite incluir análisis conjuntos en su flujo de trabajo paramétrico y mejorar la viabilidad y la capacidad de construcción de su diseño.

2.7.3. BIM GeometryGym IFC

Naturalmente, una vez que el proyectista finalice con éxito la fase de diseño estructural, ingresará a la fase de generación de documentos de

construcción, es decir, aquella fase donde debe generar los planos, especificaciones, memorias de cálculo, entre otros. necesarios para documentar su trabajo.

El uso de una plataforma BIM para generar modelos de información de construcción y, como subproducto, los planos de construcción, se está volviendo muy frecuente en las oficinas de arquitectura, ingeniería y construcción. Por lo que se cree importante hablar del complemento BIM GEOMGYM IFC un Complemento OpenBIM para Rhino y Grasshopper que permite generar e intercambiar modelos IFC (*Industry Foundation Class*) con ArchiCAD, Revit, Bently, Tekla y cualquier otro software BIM con capacidad IFC. De esta manera, una vez se tenga el modelo generado en Rhino en una plataforma BIM, se podrá comenzar con la fase de documentación.

3. METODOLOGÍA

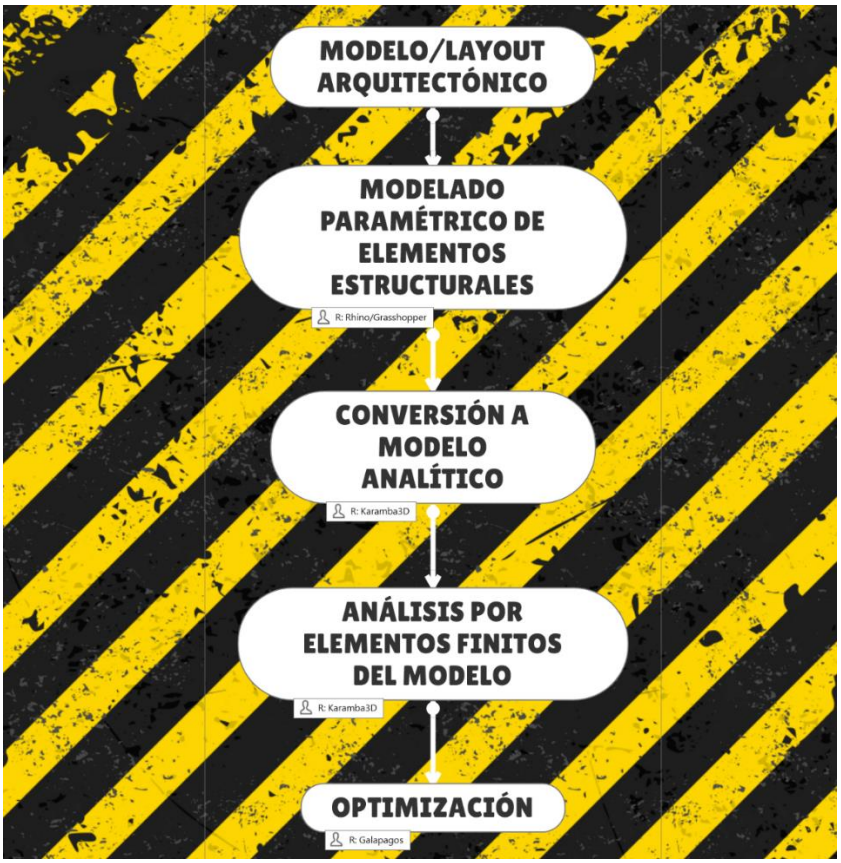
Como todo flujo de trabajo de diseño estructural contemporáneo, el diseño asistido por algoritmos depende fuertemente del uso de plataformas de software. Para lo atinente a esta investigación, el enfoque estará en las herramientas digitales que se detallan a en las siguientes secciones que, para conveniencia, se listan a continuación:

- Rhinoceros 3D
- Grasshopper
- Karamba 3D

Otra característica notable de los flujos de trabajo relacionados con el diseño asistido por algoritmos es la necesidad de contar con fuertes habilidades computacionales, tales como la codificación, conocimientos de estructuras de datos, manejo de datos, pensamiento computacional, entre otros.

En la figura 23 se muestra un diagrama muy simplificado del flujo de trabajo del diseño generativo aplicado a proyectos de ingeniería estructural. En la parte inferior izquierda de los elementos del diagrama se ha colocado una etiqueta que indica la herramienta principal usada en proceso que representa el elemento.

Figura 23. **Diagrama de fujo diseño generativo**



Fuente: elaboración propia, utilizando Mindmanager.

3.1. Modelado paramétrico

Aun cuando la investigación realizada para este trabajo sea eminentemente del campo de la ingeniería estructural conviene dedicar unas líneas al tema de la concepción arquitectónica, tan necesaria para abordar apropiadamente las cuestiones del comportamiento estructural de la obra, y también, para proporcionar mejor contexto y tener una mejor percepción sobre el tema del diseño asistido por algoritmos y su aplicación en la ingeniería estructural.

Como se ha indicado en capítulos anteriores, un proyecto de edificación comienza con un planteamiento arquitectónico encaminado a cumplir requisitos de orden espacial, funcional, estético y, en ocasiones, requerimientos asociados con el desempeño energético y la sostenibilidad.

Con toda esta retahíla de requisitos propios de un proyecto de AEC no es de extrañar que un arquitecto haga un esfuerzo hercúleo para manejar una gran cantidad de variables y datos en su cabeza. De alguna manera, estos datos terminan ingresándose o representándose en un modelo digital. Además de la manipulación de datos, con toda probabilidad, el arquitecto deberá generar, con base en dichos datos, una serie de alternativas arquitectónicas que logren cumplir requisitos funcionales, económicos, reglamentarios, entre otros. Todo este proceso de manipulación de datos y generación de alternativas hace que el diseño asistido por algoritmos sea muy atractivo para el proyectista encargado de la arquitectura de la obra.

El arquitecto puede emplear el diseño paramétrico o diseño asistido por algoritmos de forma tal que pueda generar múltiples formas posibles basadas en parámetros y en ciertas restricciones provistas. Como resultado, el proyectista arquitectónico puede desarrollar formas complejas estéticamente agradables y comenzar a explorar una cantidad significativa de escenarios en lo que respecta la planificación y uso del espacio. También tiene la posibilidad de incorporar, desde una etapa temprana, simulaciones de desempeño energético para diseñar edificios ecológicos y energéticamente eficientes.

Los párrafos anteriores ayudan a sacar una serie de conclusiones respecto al uso del diseño asistido por algoritmos aplicado a proyectos de la industria AEC:

- El diseño asistido por algoritmos es una metodología excepcional que puede aprovechar un arquitecto para la generación de modelos paramétricos y, en consecuencia, el desarrollo de óptimas alternativas arquitectónicas.

- Si existe un buen trabajo coordinado y sinérgico entre el arquitecto y el proyectista estructural, éste puede participar de forma temprana en el proyecto, añadiendo algunos algoritmos relacionados con el análisis estructural y que, con base en los resultados de dicho análisis, pueda darle una mejor asesoría y orientación al arquitecto.

- Relacionado con lo anterior, el ingeniero estructurista podrá reutilizar algunos de los scripts que el arquitecto ya ha creado en la plataforma de programación visual, con lo cual, también se estaría reutilizando y aprovechando la geometría que se genera a partir de estos scripts, traduciéndose en ahorros de esfuerzo y tiempo para el profesional encargado de la disciplina estructural.

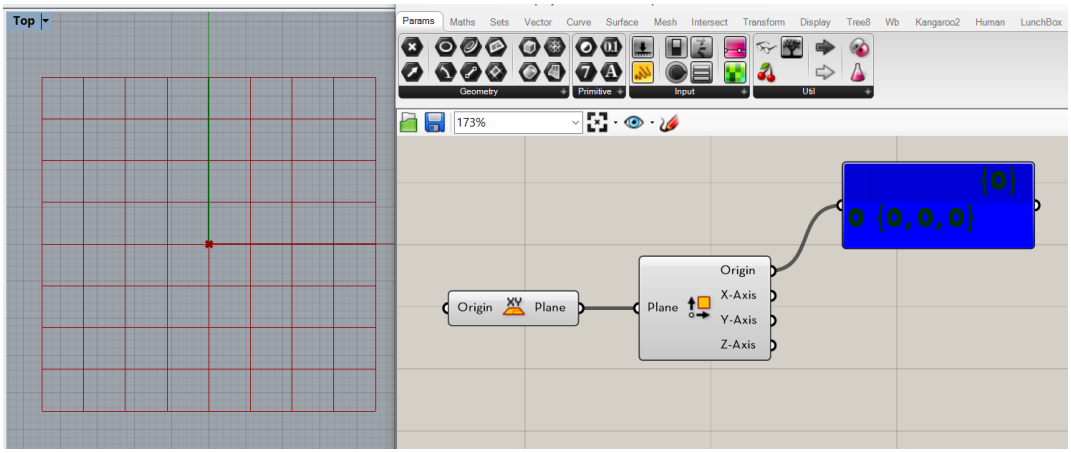
Lo anterior, pone de relieve que si el diseño asistido por algoritmos o el diseño paramétrico comienza a implementarse en etapas tempranas en un proyecto de AEC los arquitectos pueden ahorrarle al proyectista estructural una gran cantidad de tiempo y esfuerzo al generar y compartir la geometría parametrizada confeccionada por ellos. Dado que esta es una condición ideal y no siempre se presenta, las secciones posteriores están desarrolladas asumiendo que el ingeniero estructural es quien genera un modelo geométrico con cierto grado de parametrización y luego lo adapta para darle un tratamiento analítico.

Por lo anterior, en lo sucesivo se abordará el modelado, análisis y simulación de una armadura paramétrica en dos dimensiones.

Dado el carácter bidimensional de la armadura a tratar, es de esperarse que la geometría con capacidades paramétricas que representará a la armadura, este constituida por elementos básicos como líneas, polilíneas y puntos. A continuación, se describe el procedimiento paso a paso de esta construcción digital con características paramétricas.

El primer paso para realizar es la definición de un punto de origen para el modelo. Esto puede hacerse de varias maneras, la que se usará aquí implica el uso de los componentes XY Plane y Deconstruct Plane. El componente XY Plane coloca un plano en la dirección paralela al plano que contiene a los ejes coordenados X y Y, cuyo origen se ubica de forma predeterminada en el conjunto coordenado (0,0,0). Para que se pueda usar este punto como origen del modelo debemos realizar un proceso de deconstrucción del plano XY, eso se hace mediante el componente Deconstruct Plane. Esto se ilustra en la figura 24.

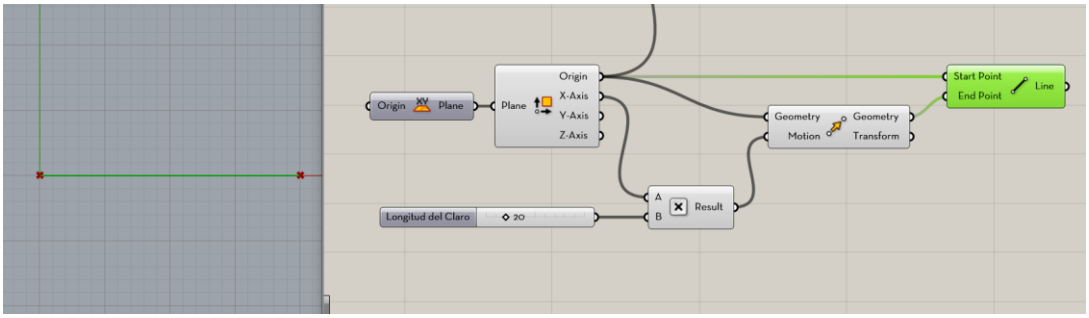
Figura 24. **Uso del componente Deconstruc Plane**



Fuente: elaboración propia, utilizando Grasshopper.

Ahora, lo siguiente por hacer es utilizar las salidas del componente Deconstruct Plane para comenzar a trazar el cordón inferior de la armadura objeto de estudio. Esto se logra trasladando el punto que se marca en el origen del plano XY una distancia paramétrica hacia la dirección de +X y luego usando el componente Line para unir el origen con el punto que se ha movido. La figura 25 ilustra esta serie de operaciones.

Figura 25. **Uso del componente Line**

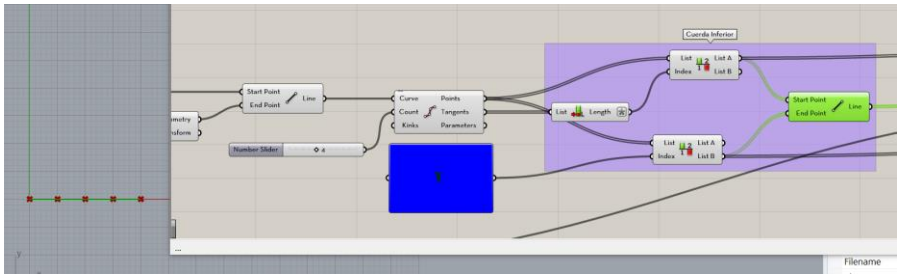


Fuente: elaboración propia, utilizando Grasshopper.

Posterior a la construcción de la línea que representa la longitud total de la cuerda inferior, se colocan en una serie de puntos que dividen la curva (cuerda inferior de la armadura en este caso) de modo tal que representen algunos nodos de la armadura donde concurren los miembros diagonales. Esto se realiza usando el componente Divide Curve. En las entradas de este componente conectamos la línea que representa la cuerda inferior de la armadura y también un Number Slider con el que se pueda manipular paramétricamente la aparición de los puntos en cuestión. Luego para obtener los segmentos lineales en los que se ha dividido la curva utilizamos una configuración de componentes como List Lenght y Split List. Con esto obtenemos dos listas que contienen a los nodos de la cuerda inferior, una de ellas comenzando en el índice 0 original y la otra comenzando en el índice 1. De esta manera al conectar las salidas de los

componentes *Split List* a un componente Line, como se muestra en la figura 26, se estarán creando porciones lineales entre el nodo i y el nodo i+1.

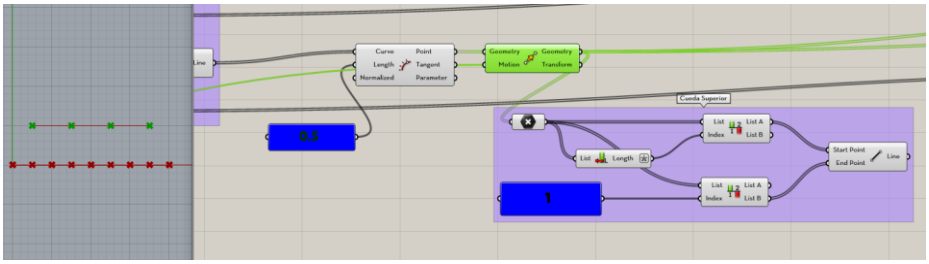
Figura 26. **Generación de cuerda inferior de la armadura**



Fuente: elaboración propia, utilizando Grasshopper.

El siguiente paso es construir la cuerda superior de la armadura partiendo de la geometría ya confeccionada. Para este paso se emplea el componente *Evaluate Length* para localizar los puntos medios de los segmentos lineales inferiores y luego con ayuda del componente *Move* desplazar dichos puntos una distancia igual a la controlada con un *Number Slider*. Por último, se lleva a cabo un procedimiento muy similar al que se usó para la construcción de la cuerda inferior para poder generar los segmentos lineales que componen la cuerda superior. Lo anterior se ilustra gráficamente en la figura 27.

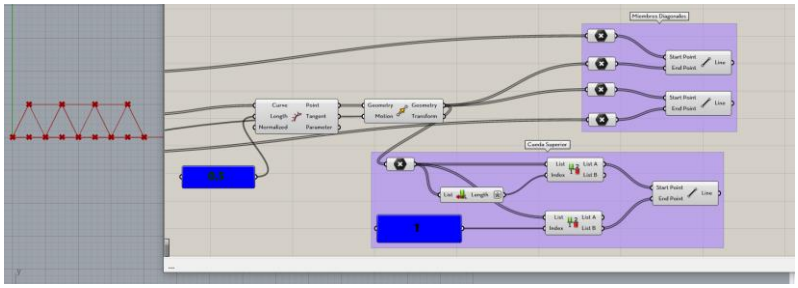
Figura 27. **Construcción cuerda superior**



Fuente: elaboración propia, utilizando Grasshopper.

Para concluir con la generación paramétrica de la armadura, lo que resta por hacer es simplemente utilizar algunos componentes Line para crear los miembros diagonales, en las ubicaciones adecuadas, que unen a los nodos anteriormente construidos. Esto se presenta en la figura 28.

Figura 28. **Generación de elementos diagonales**



Fuente: elaboración propia, utilizando Grasshopper.

3.2. Construcción de modelo analítico en Karamba3D

Construido el modelo geométrico con la capacidad paramétrica deseada, el siguiente paso es convertir la geometría en un modelo analítico representativo de la estructura real. Esto puede realizarse con ayuda del plugin Karamba3D y requiere de un procedimiento general como el siguiente. La lista mostrará en paréntesis los componentes involucrados en la operación:

- Definir los elementos de Modelo. En esencia, esto es convertir elementos tipo «línea» a elementos tipo «viga», de forma tal que el programa reconozca estos últimos como aptos para un análisis (Line to Beam).
- Definir: secciones transversales (Cross Section y Cross Section Selector), materiales (Materials), cargas actuantes (Loads) y soportes (Supp).
- Ensamblar el Modelo (Assemble Model).
- Analizar la estructura (Analyze).

- Definir la visualización del modelo (Model View y Beam View)

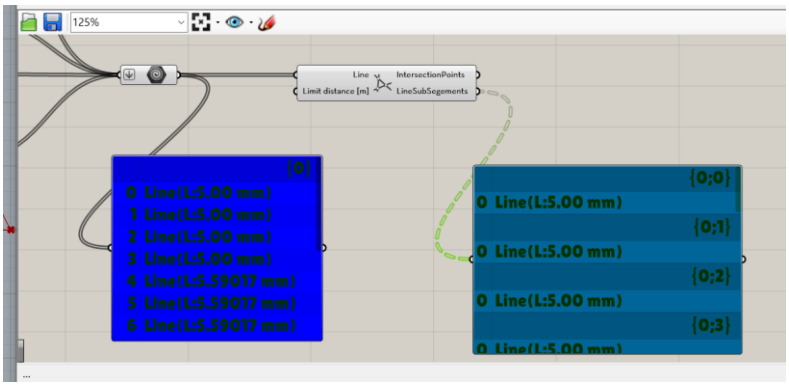
Las siguientes secciones se encargan de guiar al lector en realización de todos los pasos descritos anteriormente para el ejemplo que se ha estado tratando.

Como se discutió anteriormente, se debe llevar a cabo un proceso de transformación en el que los elementos tipo línea del modelo paramétrico pasen a ser elementos tipo viga. De esta forma Karamba3D podrá comenzar a tratarlos como miembros de un modelo analítico resistente.

Para realizar esta conversión, existen unos cuantos caminos por seguir, pero, en esta investigación, se considerará solamente uno de ellos y es el que se describe a continuación. Se debe reunir en una sola lista todos los segmentos lineales que conforman la armadura. Esto se puede hacer uniendo la totalidad de salidas de los componentes Line que generan la armadura a la entrada del componente tipo contenedor *Geometry*. Para obtener unos datos en el formato adecuado de trabajo, debemos activar la opción *Flatten* en la entrada de *Geometry.*

La salida del componente *Geometry* debe conectar al componente Line-Line Intersection. Una de las salidas de este último componente, *Line Sub Segments* en específico, nos entrega una lista con los segmentos lineales en el cual se crea una rama para cada elemento de la lista. Lo que se procede ahora es usar el componente *Line to Beam* que lleva a cabo la tarea principal de convertir elementos tipo «línea» a elementos tipificados como viga. Esto se aprecia en la figura 29.

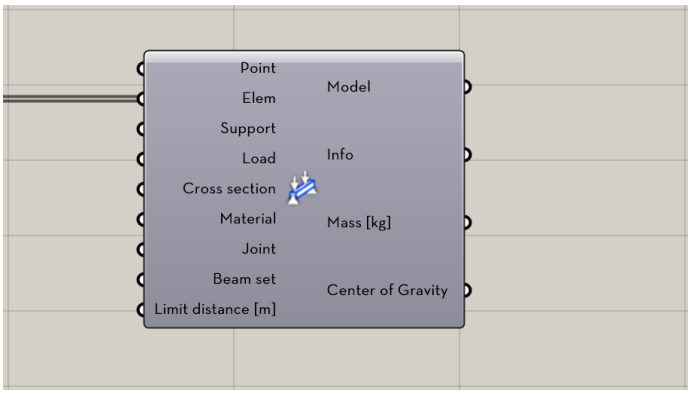
Figura 29. **Componente *Line Intersection***



Fuente: elaboración propia, utilizando Grasshopper.

Aunque el *ensamblaje* del modelo se discutirá en párrafos posteriores, por conveniencia se muestra la figura 30, en la cual se aprecia el componente *Assemble Model* con el fin de ilustrar la necesidad de definir ciertas entradas de datos como secciones transversales, cargas, soportes, materiales, entre otros. Precisamente, estas definiciones son las que se señalan en el paso 2 anteriormente presentado y que se definirán con mayor nivel de detalle a continuación.

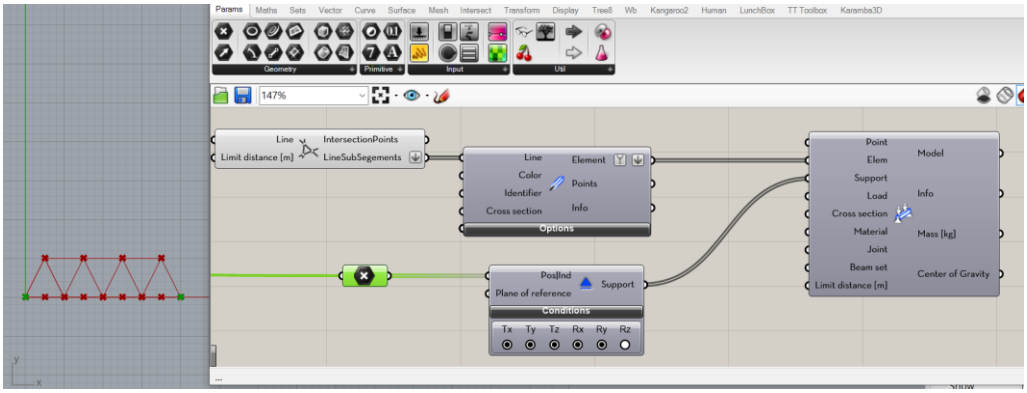
Figura 30. **Componente *Assemble Model***



Fuente: elaboración propia, utilizando Grasshopper.

Definir los soportes están fácil como localizar los puntos en donde se colocarán y luego ingresar los en el puerto Pos|Ind del componente Support. La Figura 31 muestra este paso de forma visual.

Figura 31. **Componente *Support***

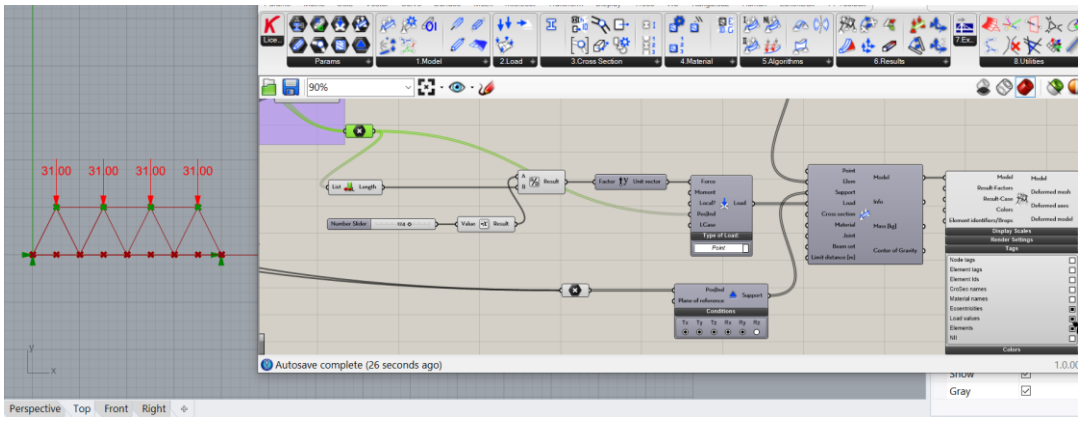


Fuente: elaboración propia, utilizando Grasshopper.

Un proceso similar al anterior se efectúa para la asignación de cargas. Dada la naturaleza vectorial de las cargas, además de indicar un punto de aplicación, se debe indicar tanto la dirección, como la magnitud de las cargas aplicadas. La figura 32 muestra este proceso. Se puede observar que la magnitud de las cargas está contralada por un *Number Slider*, el cual se conecta a un componente negativa para orientar la carga en el eje negativo de las *y* para después utilizar un componente *Divide* para poder repartir de manera uniforme la carga en los nodos de la cuerda superior de la armadura.

Es importante marcar que, para este ejemplo en específico, se utilizó un *tipo de carga puntual* que se puede establecer en el menú que se despliega al picar la parte inferior del componente Load, denominada Tyoe of Load. La naturaleza de esta carga puede obedecer también a distribuciones uniformes, cargas gravitacionales entre otras.

Figura 32. **Asignación de cargas**



Fuente: elaboración propia, utilizando Grasshopper.

Para terminar con la definición de las propiedades estructurales del modelo, el último paso es establecer la sección transversal de los miembros que constituyen la armadura.

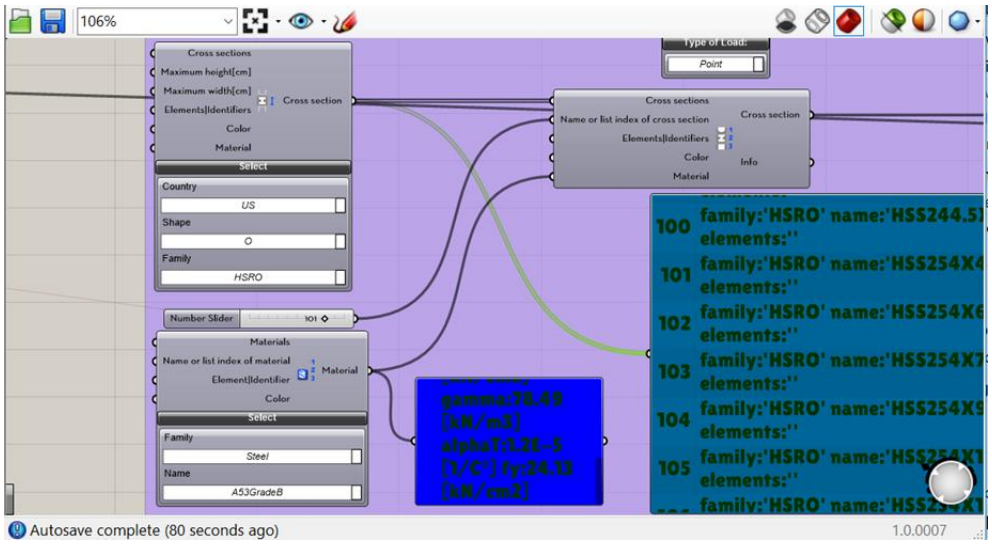
Dado que este ejemplo tiene como fin presentar la esencia de la metodología sea asignará a la totalidad de los miembros estructurales de la armadura la misma sección transversal. Junto con la sección transversal, también se asignará el tipo de material a emplear.

Para indicar con qué sección (o secciones) se desea trabajar entran en juego dos componentes: *Cross Section Range Selector* y *Cross Section Selector*. El primero nos permite secciones transversales por país, forma, familia, o bien, por ancho o peralte máximo.

Dado que, en la república de Guatemala, para los procesos de diseño, fabricación y construcción con acero se toma como una buena referencia los estándares publicados por el Instituto Americano de Construcción en Acero

(AISC por sus siglas en inglés), en este trabajo de investigación se hará referencia y mención a algunas de las publicaciones desarrollados por tal instituto. Con esto en mente, bajo el desplegable *Select*, del componente *Cross Section Range Selector*, establecemos *US* en el renglón Country (Consulte la figura 33). En el renglón *Shape* colocamos *O,* para el programa pueda trabajar con formas de sección transversal circular y, por último, en el renglón *Family* indicamos la opción HSRO.

Figura 33. **Componente *Cross Section Range Selector***



Fuente: elaboración propia, utilizando Grasshopper.

Al establecer el renglón *Family* como HSRO, y tomando en cuenta a las dos opciones anteriormente configuradas, lo que le estamos ordenando al programa es que use los perfiles que en el manual e Construcción en Acero del AISC se denotan como *Round HSS.*

Un componte panel conectado a la salida de *Cross Section Range Selector* puede mostrar la lista de los perfiles de los cuales se dispone para modelar y

analizar la estructura (147 para este caso). Es importante prestar una adecuada atención a este listado, no solo para ver cuáles son los perfiles que se están incluyendo, sino también, para observar qué índice le corresponde a cada perfil en la lista. Al observar detenidamente el componente Panel de la figura 31, el lector puede darse cuenta de que la notación corresponde al sistema internacional SI. Para poder ver la equivalencia de formas *Round* HSS del estándar estadounidense al sistema internacional se recomienda consulta la tabla 17-9 del Manual de Construcción Acero del AISC. (AISC, 2017)

Por otro lado, el componente *Cross Section Selector*, permite seleccionar una sección transversal por nombre o índice de la lista de las secciones transversales. En la figura 34 se observa que un *Number Slider* es el responsable de indicarle al componente *Cross Section Selector.* que sección transversal elegir mediante la indicación de un índice que asociado a ésta.

Figura 34. **Componente *Cross Section Selector***

Fuente: elaboración propia, utilizando Grasshopper.

De la mano con el establecimiento de las secciones transversales, algo que se debe de hacer es indicar el material o materiales de los que se compone

la estructura, ya que éstos deben adherirse a las secciones transversales con las que se está trabajando.

Los materiales se pueden definir configurando manualmente sus propiedades mecánicas o mediante la selección de una biblioteca de materiales predefinidos. (Presigner, s.f.)

Este paso también se muestra en la figura 32, en donde el componente *Material Selection* se conecta al componente *Cross Section Selector*. En el componente en cuestión, debe picarse el botón Select para poder mostrar las opciones *Family y Name*, en las cuales asignamos el tipo de material (concreto, acero, entre otros.) y el estándar que cumple dicho material, respectivamente. Para este ejemplo, hemos definido un Acero de calidad ASTM A53.

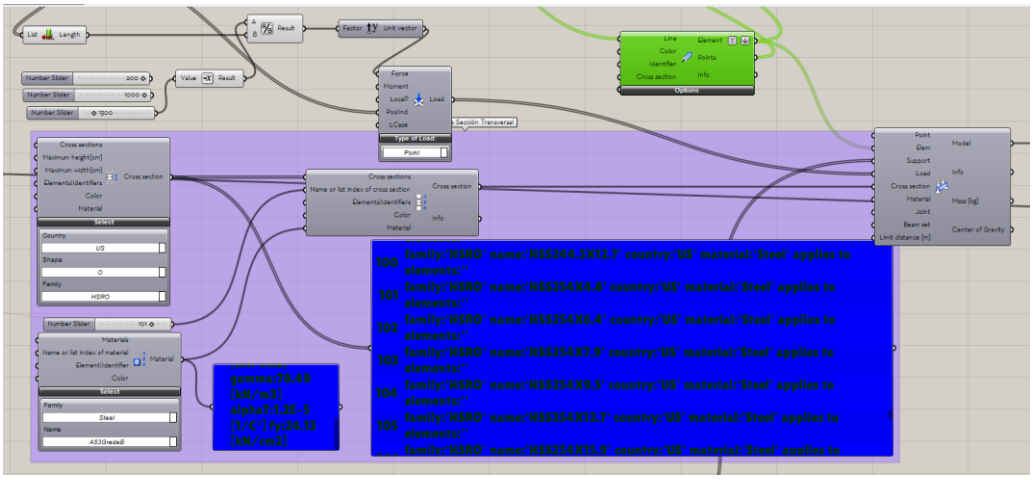
Como se mencionó anteriormente, la definición de materiales está estrechamente relacionada con la fijación y selección de las secciones transversales, ya que el componente *Cross Section Selector* tiene una ranura para conectar allí la salida del componente *Material Selection*.

El otro camino posible para asignar materiales es a través de la entrada Material del componente Assemble, en el que se entrará en detalle en los siguientes párrafos.

Hasta el momento las líneas anteriores se han dedicado a exponer la manera en que se lleva a cabo la definición de los elementos componentes básicos que se requieren para la construcción de un modelo analítico. El paso que se abre a continuación es obvio, se debe interconectar todos los componentes ya definidos de modo tal que juntos, logren darle vida a un modelo susceptible de análisis. Esto se logra conectado las salidas de los componentes

Line to Beam, Loads, Cross Section Selector, Support al componente *Assemble*. Esencialmente, este componente crea un modelo de elementos finitos que pueda utilizarse para procedimientos de análisis estructural. Esto se muestra en la figura 35.

Figura 35. **Componente Assemble**



Fuente: elaboración propia, utilizando Grasshopper.

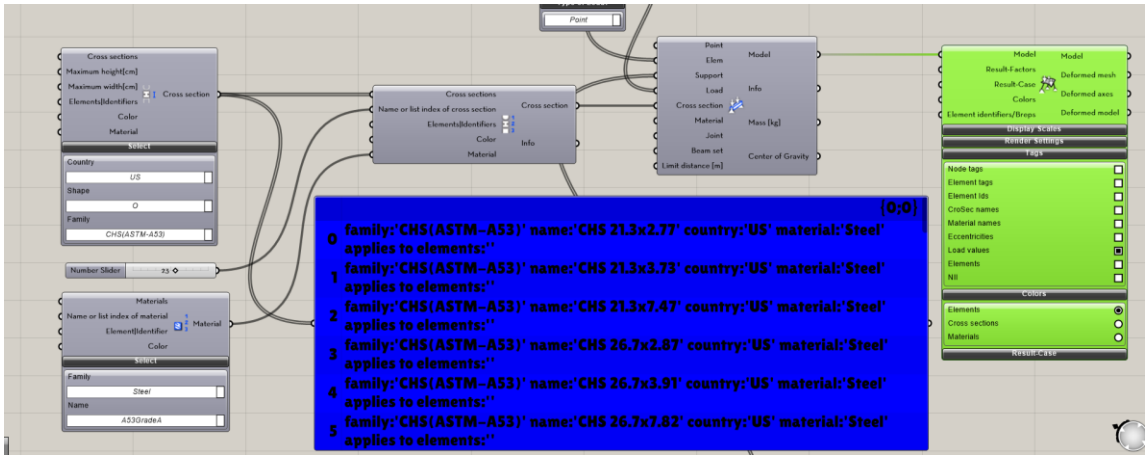
Nótese que en la figura 35 en el componente *Assemble* no hay datos de entrada para la ranura *Material*, esto se debe a que el material se ha asignado *a priori* a través del componente *Cross Section Selector*.

Si bien hemos añadido a la discusión la inclusión del componente *Assemble,* después de la definición de parámetros y propiedades básicas del modelo paramétrico, una buena práctica es insertar este componente en el lienzo de *Grasshopper* una vez se determine la generación de la geometría paramétrica del modelo. También, para tener un control visual, en el entorno de modelado de Rhino (en los *Viewports*, específicamente), sobre lo que pasa con las propiedades que se van definiendo es recomendable conectar la salida *Model* del

componente *Assemble* a la entrada *Model* del componente *Model View*, ya que con ayuda de las opciones proporcionadas en la parte inferior de este componente, podemos ir modificando la visibilidad y gráficos de las definición y asignaciones de propiedades como cargas, soportes, materiales, entre otros.

Ahora que ya se cuenta con el modelo analítico listo para someterse a análisis, se debe ahora colocar en el lienzo el componente *Analyze* para calcular las deflexiones del modelo, el cual, utiliza la teoría de primer orden para este cálculo. En la entrada *Model* de este componente debe conectarse la salida *Model* del componente *Assemble Model*, como se muestra en la Figura 36.

Figura 36. **Uso del componente *Analyze***



Fuente: elaboración propia, utilizando Grasshopper.

El componente de Analyze no solo calcula las deflexiones del modelo, sino que también genera el desplazamiento nodal máximo (en centímetros), la fuerza de gravedad total máxima (en kilo Newton, si se establece la gravedad) y la energía de deformación interna de la estructura para cada caso de carga. Estos valores se pueden utilizar para clasificar estructuras en el curso de un procedimiento de optimización estructural: cuanto más eficiente es una

estructura, menor es la deflexión máxima, la cantidad de material utilizado y el valor de la energía elástica interna. (Presigner, s.f.)

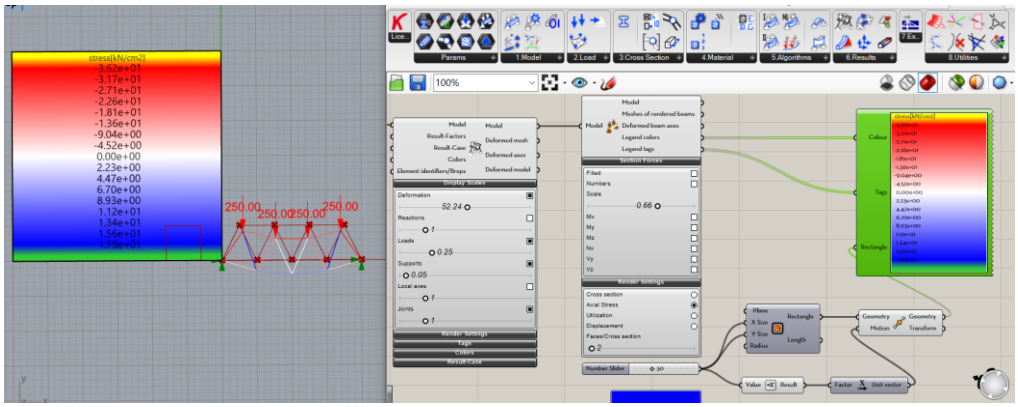
Tanto la deflexión máxima como la energía elástica proporcionan un punto de referencia para la rigidez estructural, pero desde diferentes puntos de vista: el valor de la energía elástica permite juzgar una estructura en su conjunto. El desplazamiento máximo devuelve un valor máximo local. (Presigner, s.f.)

Por último, lo que se debe hacer, es conectar el modelo al componente BeamView para poder visualizar el rango de esfuerzos a los que se está sometiendo la estructura.

Este componente contiene un submenú que se puede expandir haciendo clic en la barra de subtítulo negra. El rango numérico de los controles deslizantes se puede configurar haciendo doble clic en su botón negro. Las propiedades de visualización se adhieren al modelo y siguen siendo válidas hasta que sean anuladas por otro componente de visualización posterior. (Presigner, s.f.)

Junto a este componente se inserta el componente Legend, cuya tarea es mostrar los esfuerzos a través de un rango de colores, donde cada color representa un valor numérico para el esfuerzo que experimenta la estructura. La interacción con el componente Legend es muy sencilla, simplemente se conecta las salidas Legend Colour y Legend Tag, del componente Beam View, a las entradas Colour y Tag, del componente Legend, respectivamente. Esto se muestra en la figura 37.

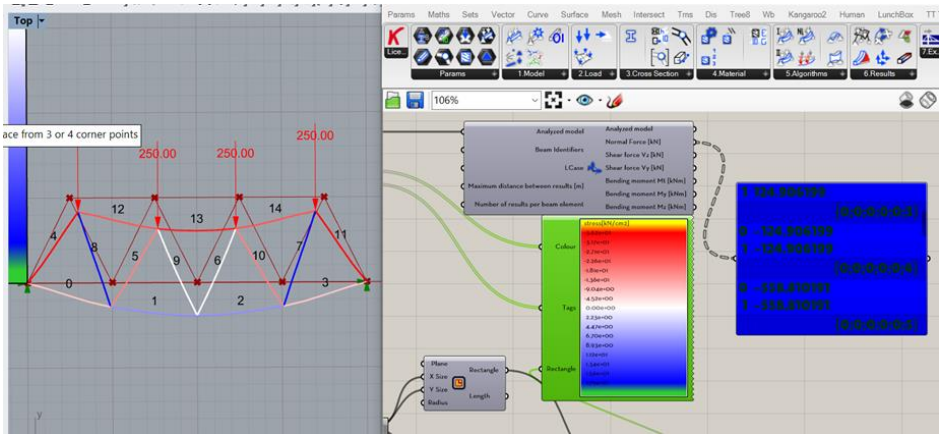
Figura 37. **Uso del componente Analyze**



Fuente: elaboración propia, utilizando Grasshopper.

Por otro lado, si lo que desea es ver la fuerza axial que actúa en cada miembro de la armadura, se debe utilizar el componente Beam Forces. A éste, en su entrada Analized Model, se conecta la salida Analyzed, Model del Componente Beam View. Luego se conecta un cable que va desde la salida Analyzed Model hacia un componente Panel para poder observar una lista con los valores numéricos de las fuerzas, como se muestra en la figura 38.

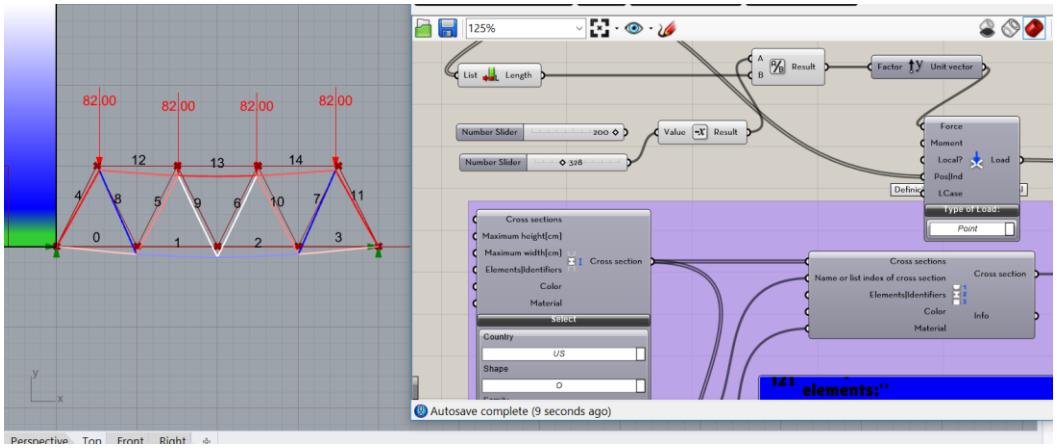
Figura 38. **Uso del componente Analyze**



Fuente: elaboración propia, utilizando Grasshopper.

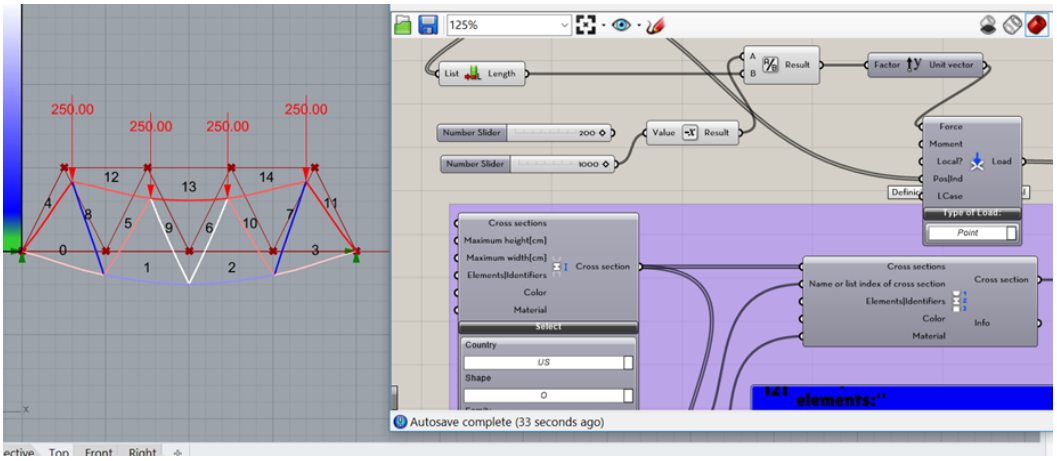
Como lo pone de evidencia la figura 39 y la figura 40, se puede observar en tiempo real los cambios en el estado de esfuerzos y en la configuración deformada de la estructura al cambiar cualquier entrada de datos que esté controlada de forma paramétrica.

Figura 39. **Cambio de estado de esfuerzos por cambios en la carga**



Fuente: elaboración propia, utilizando Grasshopper.

Figura 40. **Cambio de estados de esfuerzo por cambios en la carga**



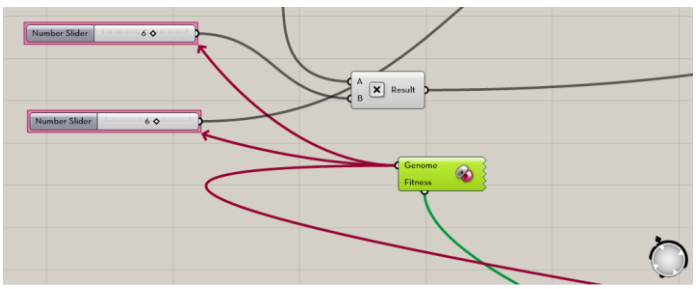
Fuente: elaboración propia, utilizando Grasshopper.

Hasta el momento se ha demostrado el poder y versatilidad de Karamba3D como programa de análisis estructural de elementos finitos utilizando como entrada modelos analíticos con cierto grado de parametrización. Se ha presentado también, como al contar con las características paramétricas del análisis, pueden realizarse cambios en las secciones, cargas, longitudes de claro, número de módulos en la armadura, entre otros. e inmediatamente pueden visualizarse la configuración deformada y los valores de los esfuerzos actuantes, actualizados y en tiempo real, correspondientes a esos cambios.

Sin embargo, aún resta someter al modelo paramétrico a un proceso de optimización, que consiste en encontrar la combinación óptima de peralte, secciones y número de módulos de la armadura de forma tal que el desplazamiento máximo de la armadura no supere el valor límite de 5.00 cm.

Tal procedimiento de optimización requiere que se ingrese al lienzo de Grasshopper el componente Galápagos. Ingresado este componente, desde el puerto *Genome* generamos un cable cuyo extremo final deba conectarse a los componentes tipo *Number Slider* que controlen paramétricamente al tamaño del peralte de la armadura, su número de módulos y las secciones transversales en los elementos de la armadura, como se indica en la figura 41.

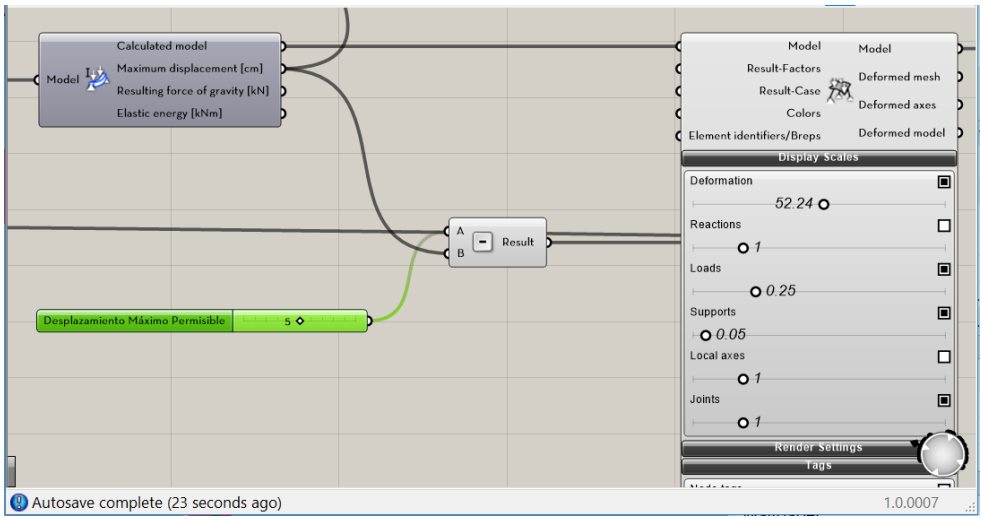
Figura 41. **Uso del componente Galápagos**



Fuente: elaboración propia, utilizando Grasshopper.

Básicamente, los componentes que se conectan a la entrada Genome son las variables que manipulará Galápagos para optimizarlas y obtener un valor que satisfaga la restricción indicada en la entrada *Fitness*. En este caso, la restricción es que la deflexión máxima que experimenta la armadura no supere el valor de 5.00 cm. Para indicar esto, lo primero que se debe hacer es ubicar el componente y su respectiva salida que nos entreguen la deformación de la armadura. En este caso, tal componente es *Analyze* y la salida de interés es *Maximum Displacement.* Luego, con ayuda del componente *Substrac*t debe llegarse a la configuración mostrada en la Figura 42, en donde el *Number Slider* desplazamiento máximo permisible controla el valor numérico del desplazamiento máximo permisible.

Figura 42. **Uso del desplazamiento máximo para Galápagos**

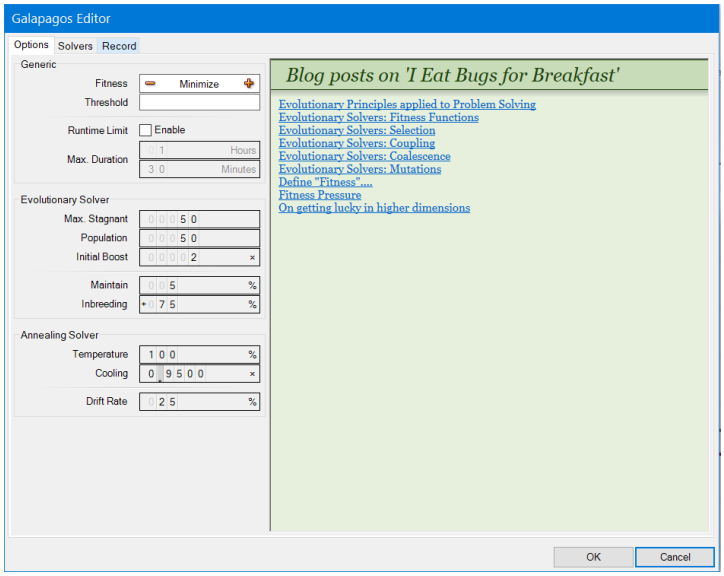


Fuente: elaboración propia, utilizando Grasshopper

Hecho lo anterior, la salida del componente Substract debe conectarse al componente Absolute y luego conectar la salida Fitness del componente Galapagos a la salida del componente Absolute.

Hasta ahora se ha desarrollado todo un algoritmo para analizar un modelo paramétrico a través de elementos finitos y, además, una parte de tal algoritmo, cuya construcción se ha descrito en líneas anteriores, está destinada a someter a algunas variables clave a un proceso de optimización. Lo que resta, es indicarle a las Galápagos cómo se da el proceso de optimización, si se desea encontrar ciertos valores numéricos para algunos parámetros de modo tal otra cantidad importante se límite a un valor máximo o mínimo. Para ello, con doble clic en el componente Galápagos ingresamos al cuadro de diálogo mostrado en la figura 43.

Figura 43. **Galápagos Editor**



Fuente: elaboración propia, utilizando Grasshopper.

Aunque podemos controlar ciertas especificaciones para el solucionador evolutivo que ofrece Galápagos se pondrá énfasis en solo dos acciones que, para fines de esta investigación, son de gran relevancia: (1) en la pestaña *Options*, bajo el apartado *Generic*, en el renglón *Fistness*, debe indicarse *Minimize* (ya que

lo que se desea es conseguir un desplazamiento en el centro de la armadura que sea mínimo), (2) en la pestaña *Solvers* se debe picar el botón *Start Solvers*.

En ese momento Galápagos comienza el proceso de optimización y dependiendo de la dificultad del problema el tiempo que tarde en llegar a una solución podría ser extenso.

Algo importante a tener en cuenta al momento de usar algunos de los componentes de Karamba3D es que prestar atención a las configuraciones y valores predeterminados que estos poseen. Por ejemplo, la configuración predeterminada para los materiales es que sean un acero con un $f_y = 23.5\ kN/m^2$. Para las secciones de vigas, Karamba3D asigna por defecto una sección circular hueca con un diámetro exterior de 114.4mm y un espesor de pared de 4mm.

4. ESTUDIOS DE CASO

4.1. Descripción de la estructura

El estudio de caso atinente a esta investigación será el análisis de una estructura de cubierta conformada por armaduras de acero con espaciamientos entre 6.00 m y cuyas secciones transversales corresponden a perfiles HSS cuadrados de acero calidad ASTM A53 grado B cuyo esfuerzo de cedencia (f_y) tiene un valor numérico de 35.00 KSI. Se probaron varios tipos de armaduras como Warren, Vierendel, Prat, Howe y tipo arqueada, a través de los controles paramétricos del modelo y, con base en los resultados del análisis estructural preliminar, se optó por únicamente un tipo es específico. Las bahías o claros a cubrir corresponden a un área de 25.00 m de ancho por 30.00 m de largo.

4.2. Cargas actuantes

A continuación, se describen las cargas que se consideraron en el análisis de la estructura.

4.2.1. Peso propio de los elementos

Esta carga corresponde al peso propio de los elementos y dada la capacidad paramétrica de cambiar el tipo de sección, tamaño y material los cambios efectuados en estas propiedades se propagan automáticamente a la magnitud de las cargas debido al peso propio que actúan en la estructura.

4.2.2. Peso propio de los elementos

Estas corresponden al contenido permanente del sistema de cubierta, el cual incluye el peso debido a elementos de techado y aislamiento térmico, un *deck* metálico, acabados de cielo raso y componentes mecánicos (tuberías, ductos, entre otros.).

4.2.3. Carga viva

Los valores para las cargas vivas que se supone que actuarán en una estructura se encuentran prescritos en estándares y reglamentos. Para el caso de estudio, el valor adoptado fue de 50 kgf/m^2 según el estándar NSE2 *Demandas Estructurales y Condiciones de Carga*, §3.7, Tabla 3.7.1-1.

En tabla I se muestra un resumen de las cargas que actúan en la estructura:

Tabla I. **Resumen de cargas muertas actuantes**

METRADO DE CARGA MUERTA SUPERPUESTA	
TIPO DE CARGA	**CARGA (Kg/m^2)**
Aislamiento	50.00
Deck Metálico	12.00
Largueros de techo	38.00
Cielo raso y ductos	50.00
TOTAL	150.00

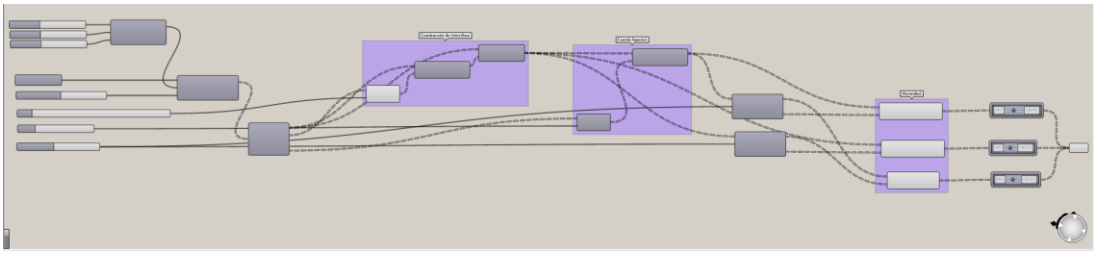
Fuente: elaboración propia.

4.3. Construcción de geometría paramétrica

Los miembros estructurales de la obra descrita anteriormente que modelarán dentro del entrono paramétrico de Rhino y Grasshoper serán el conjunto de armaduras encargadas de soportar la cubierta de modo tal que las cargas provenientes de la cubierta actúen en los nodos de dichas armaduras.

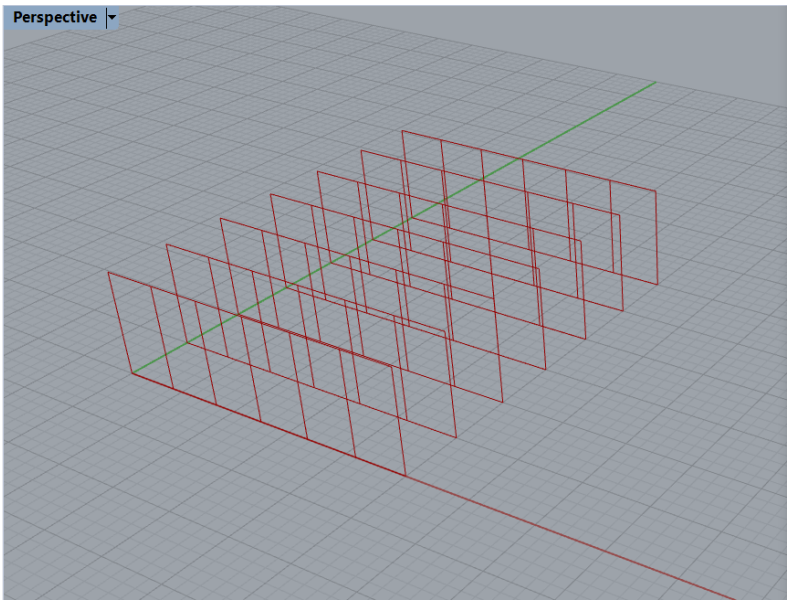
El primer paso, y sobre el cual se fue construyendo las partes restantes del modelo paramétrico, fue modelar un conjunto de armaduras paralelas tipo vierendel que, dada su simplicidad, al ir añadiendo algunos componentes más se pudo agregar elementos diagonales característicos de los otros tipos de armaduras. El algoritmo que genera esto se muestra en la figura 44, mientras que el resulta producido en el entorno de modelado de Rhino se observa en la figura 45. Toda la geometría mostrada se almacena en un componente contenedor *Geometry*, que ha sido antes pasada por un componente *Path Maper*.

Figura 44. **Algoritmo para la generación de un conjunto de armaduras paramétricas**



Fuente: elaboración propia, utilizando Grasshopper.

Figura 45. **Conjunto de armaduras paramétricas**



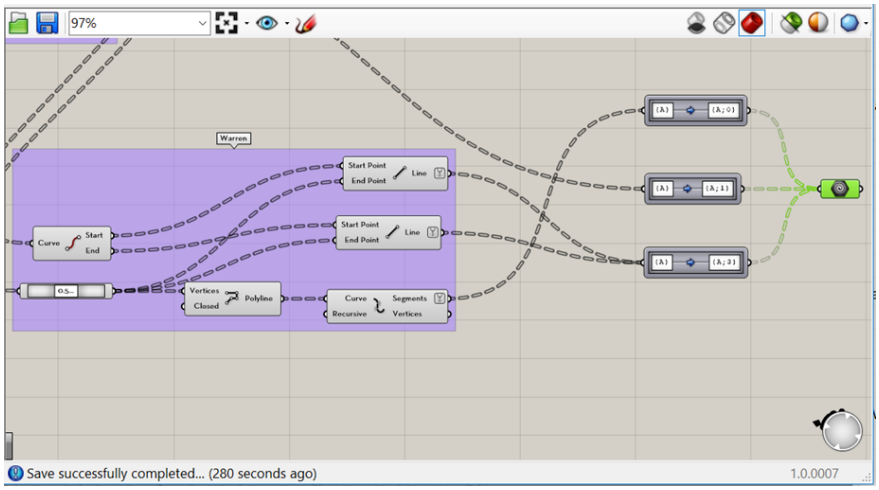
Fuente: elaboración propia, utilizando Grasshopper.

Para construir los demás tipos de armaduras, se reutilizará parte de la geometría generada por los algoritmos ya desarrollados, más específicamente, se usará nuevamente parte del algoritmo que dio vida a la geometría mostrada en la figura 45.

El tipo de armadura que se construyó a partir de lo existente corresponde al tipo Howe. Dado que este tipo de estructura carece de elementos verticales y horizontales superiores en las bahías extremas, se utilizó el componente *Cull Index* para eliminar esta geometría. Por otro lado, las armaduras Howe poseen miembros dispuestos oblicuamente en sus bahías, por lo que se requirió añadir algunos componentes al algoritmo base que pudiesen generar esta geometría y además adaptarse a las capacidades paramétricas ya existentes.

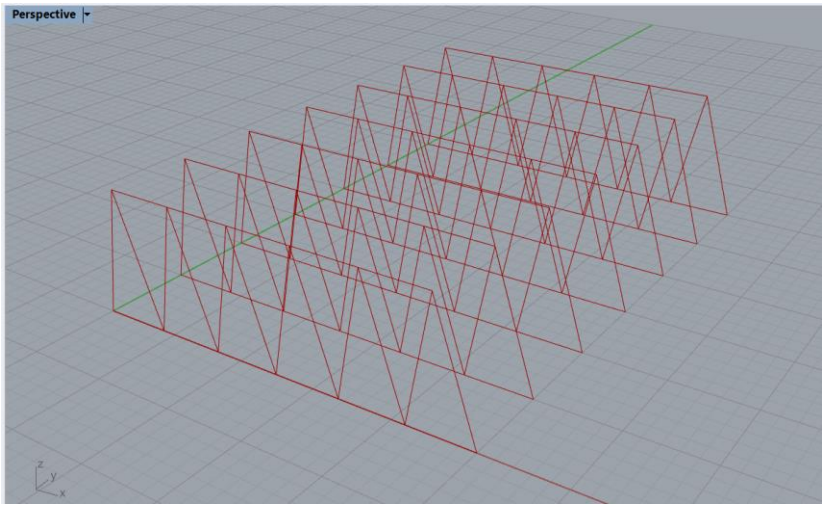
La parte añadida al algoritmo se observa en la figura 46, mientras que la geometría paramétrica asociada se aprecia en la figura 47.

Figura 46. **Algoritmo para la generación de un conjunto de armaduras paramétricas tipo Howe**



Fuente: elaboración propia, utilizando Grasshopper.

Figura 47. **Conjunto de armaduras paramétricas tipo Howe**

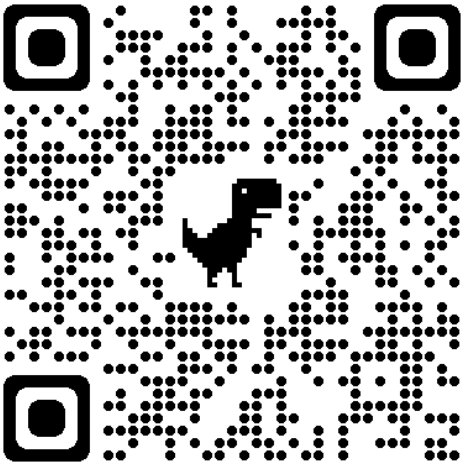


Fuente: elaboración propia, utilizando Grasshopper.

De esta forma, construyendo sobre la geometría preexistente se logró añadir otro tipo de armadura, la arqueada. Como resultado, en el modelo ahora podía elegir entre una cantidad significativa de tipos de armadura, tamaños para el peralte, cantidad de bahías, entre otros.

La figura 48 presenta un código QR que dirige al lector a un video en Youtube en donde se muestran las capacidades paramétricas que alcanzó el modelo considerado en esta investigación.

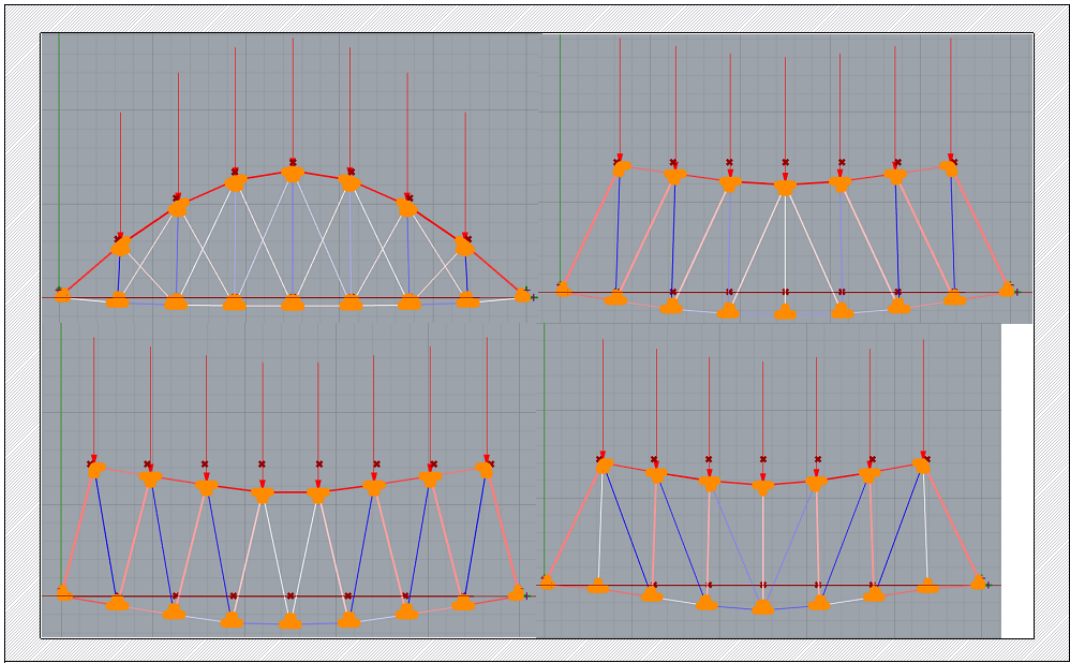
Figura 48. **Código QR que dirige a un video que muestra la capacidad paramétrica del modelo**



Fuente: elaboración propia, utilizando Google.

Del abanico de posibilidades que tienen en términos de tipo de armaduras se optó por elegir una armadura tipo arqueada ya que resultó ser un tipo de armadura que respondía de manera más eficiente al estado de esfuerzos al que se estaba sometiendo la estructura. Lo anterior se ilustra en la figura 49.

Figura 49. **Comparación del estado de esfuerzos en armaduras**



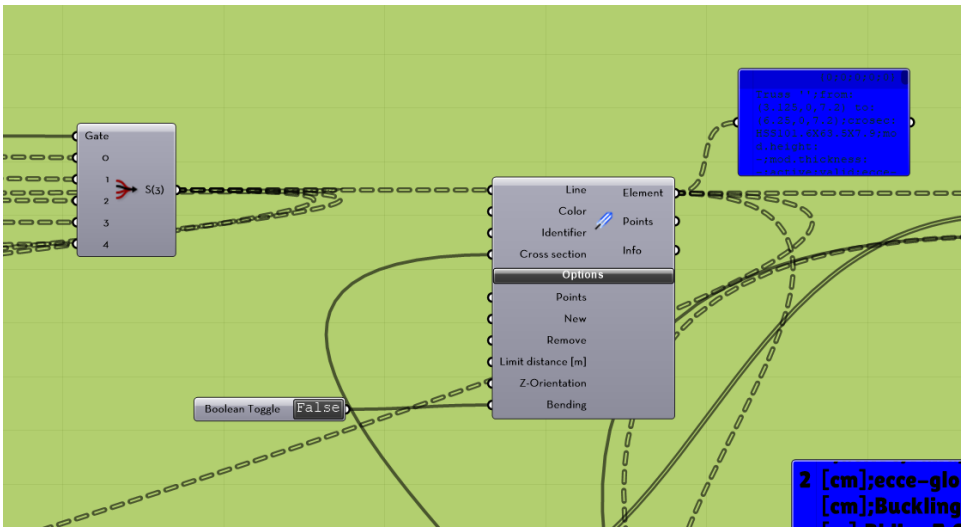
Fuente: elaboración propia, utilizando Rhinoceros.

4.4. Conversión a modelo analítico 3D

Hasta ahora se ha creado un modelo cuya geometría y configuración puede reaccionar en tiempo real a los cambios que puedan efectuarse en los parámetros de entrada: peralte, número de bahías en la armadura, tipo de armadura, entre otros. En este punto, lo que sigue es convertir todo este modelo paramétrico en uno en el que puedan asignarse secciones transversales, cargas, materiales, entre otros, lo que daría como resultado un modelo analítico en el que podrían calcularse los estados de esfuerzo en la estructura, así como sus deflexiones.

Para hacer esta conversión y poder darle un tratamiento analítico al modelo se empleó el componente Line to Beam como se muestra en la figura 50.

Figura 50. **Componente Line to Beam**



Fuente: elaboración propia, utilizando Grasshopper.

Como se aprecia en la figura 48, la geometría proveniente de un Stream Filter, (el cual fue necesario incorporar al algoritmo para poder cambiar entre las distintas opciones de tipos de armadura) se inserta dentro del slot Line del componente Line to Beam. La última configuración que se efectuó, relativa a este componente, fue utilizar un Boolean Toggle, establecido en *False,* e insertarlo en el slot Bending para indicarle al programa que las conexiones entre miembros de la armadura funcionaran como articulaciones, es decir, sin la capacidad de transmitir momentos.

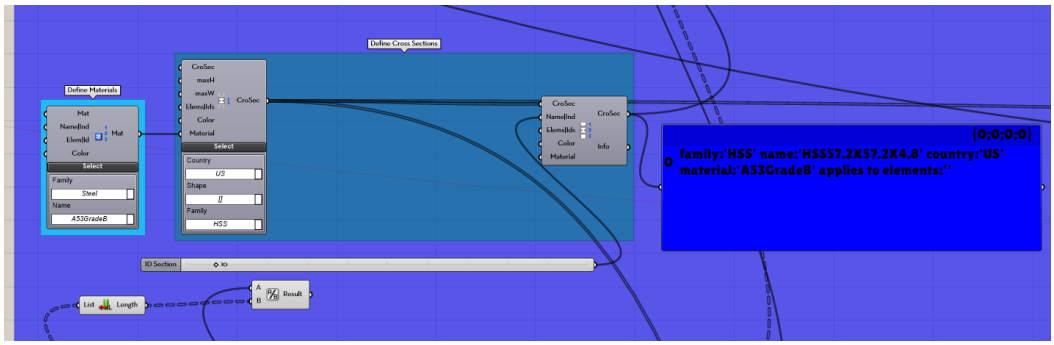
Inmediatamente después de esta conversión de elementos lineales a elementos estructurales, el siguiente paso fue definir todas las propiedades necesarias para llevar a cabo el análisis estructural: materiales y secciones transversales.

El material elegido para la estructura fue un acero estructural de calidad ASTM A53 grado B y la forma estructural establecida fue la HSS cuadrada. Como

con otros atributos de la estructura, a las dimensiones de la sección transversal le fueron concedidos controles paramétricos para poder jugar con una variedad de tamaños. De esta manera, realizando cambios en los números dados por un Number Slider se controlada el ID asociado a las dimensiones de las secciones transversales de una variedad de perfiles HSS.

Por ejemplo, si el *slider* se encontraba posicionado en el número 10, el componente Cross Section Selector suministraba como salida la sección con el ID asignada como 10, en este caso, un perfil HSS 57.2x57.2x4.8, como se observa en la Figura 51. Además, esta salida se conectó al slot Cross Section del componente Line to Beam.

Figura 51. **Componente Cross Section Selector**



Fuente: elaboración propia, utilizando Grasshopper.

Por último, en aras de favorecer el proceso de fabricación y montaje, se optó por un solo tamaño común para la totalidad de miembros en las armaduras.

Hechas estas configuraciones la salida del componente Line to Beam se conectó al componente Assemble Model, en el slot Elem. Se aplicó un *Flatten* y un *Simplfy* a esta entrada para que la estructura de datos fuera coherente con la función de Assemble Model.

Para terminar la generación del modelo analítico se añadió al algoritmo el componente Support para indicar las restricciones de traslación y rotación en los puntos de apoyo de la estructura. La salida de este componente se conectó al componente Assemble Model, en la entrada Support.

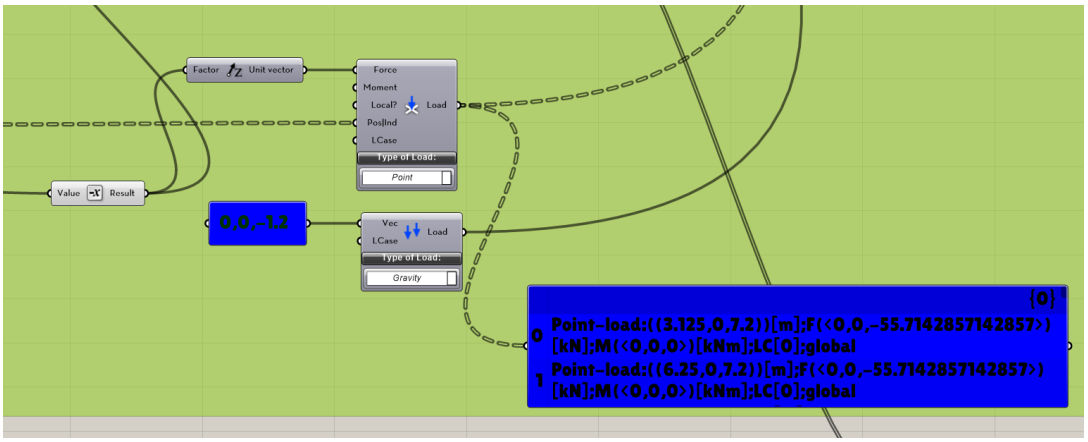
Por último, se asignaron los casos de carga que actuarían en la estructura. Para esto se calculó la carga muerta superpuesta por m^2 y la carga viva de techo por m^2 que soportaría la cubierta, que luego se transfiere a unos largueros (no modelados) que se encuentran simplemente apoyados en los nodos de la armadura.

Las cargas pasaron por un proceso de factorización y combinación para obtener un valor numérico para la carga última que, para este caso, resulto en 2.60 kN/m^2. Dado que esta intensidad está dada por metro cuadrado e idealmente la carga se coloca en los nodos de la armadura, se analizó la ruta de carga gravitacional, asumiendo una condición en que los largueros soportarían esta carga según su área tributaría y que posteriormente estas entregarían las cargas a los nodos de las armaduras.

Estas condiciones de carga se introdujeron al algoritmo de modo tal que fueran consistentes con los cambios en los parámetros que influyen en la magnitud de las cargas actuantes, por ejemplo, el número bahías de la armadura.

Junto a esto, se introdujo otro componente tipo Loads para ingresar el efecto del peso propio de la armadura en las cargas actuantes. Esta configuración se muestra en la figura 52.

Figura 52. **Componente Loads**

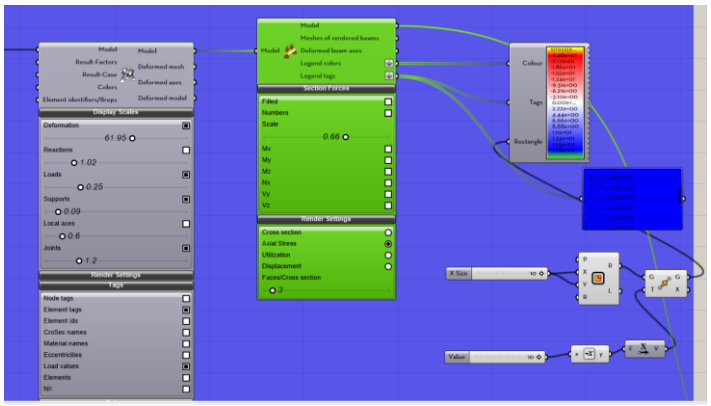


Fuente: elaboración propia, utilizando Grasshopper.

Ambos componentes Loads se conectaron al componente Assemble Model, en la entrada Load. Para que fuera coherente con la estructura de datos, se aplicó un Flatten a los datos provenientes de los dos componentes Loads.

En este punto, el algoritmo ya estaba completo en términos de geometría paramétrica y modelado analítico. Lo que restaba era añadir al algoritmo los componentes encargados del análisis estructural y también aquellos destinados a entregar los resultados visuales en el entorno de modelado 3D de Rhino. Tales componentes son: Analyze, el que se encarga de realizar el análisis estructural *per se* y entregar resultados de máximo desplazamiento, energía elástica y fuerza resultante de gravedad; Model view, destinado a configurar la visualización del modelo analítico; Beam View, que se encarga de la configuración visual de los elementos de la armadura, asignando leyendas en función de desplazamientos, esfuerzo, secciones transversales o utilización. La Figura 53 ilustra lo anterior. En la que se puede ver que el componente Beam View se encuentra resaltado.

Figura 53. **Componente Beam View**

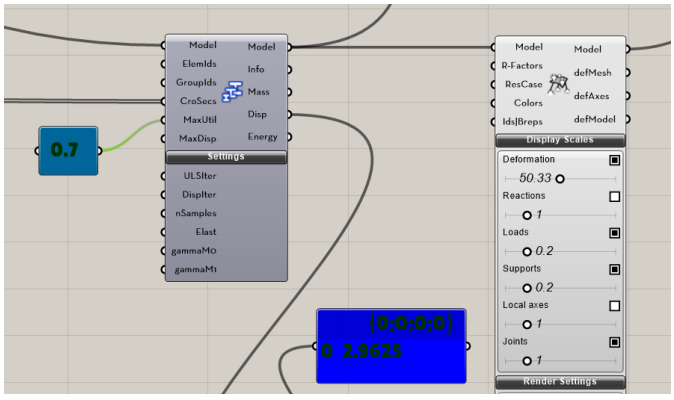


Fuente: elaboración propia, utilizando Grasshopper.

4.5. Optimización de la estructura con Galapagos

Dentro del conjunto de componentes que posee Karamba3D existe el componente Optimize Cross Section, el cuál analiza el modelo y selecciona de una lista de secciones transversales, aquella que cumpla con ciertas restricciones de entrada como, por ejemplo, un valor límite para utilización, o bien, el máximo desplazamiento permisible. Esto se ilustra en la figura 54.

Figura 54. **Optimize Cross Section**



Fuente: elaboración propia, utilizando Grasshopper.

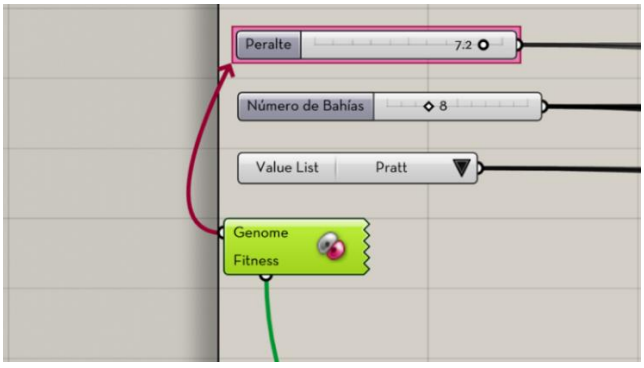
Este componente se empleó en el algoritmo para encontrar una sección transversal de la lista de perfiles HSS considerada de modo tal que la utilización no fuera superior a 0.7.

Además de este proceso de optimización inicial, se usó el componente Galápagos de Grasshopper para poder realizar el proceso de optimización utilizando solucionadores evolutivos.

La variable objetivo sobre la cual recaía el interés en optimizar era el valor del peralte de la cada una de las armaduras bajo estudio, de forma tal que se obtenga la deflexión vertical mínima, regida por la norma NSE 3, §4.4. En el lenguaje utilizado por el componente Galápagos, esto es el genoma del problema es el peralte y la función fitness sería el desplazamiento máximo vertical.

De esta forma, al insertar el componente Galápagos en el lienzo de Grasshopper, se conectó el slot Genome al parámetro del peralte que, el cual estaba gobernado por un componente tipo Number Slider. El *slot* Fitness de las Galápagos se conectó a un contenedor que le que se almacenada el valor numérico de la deflexión vertical resultante del análisis estructural.

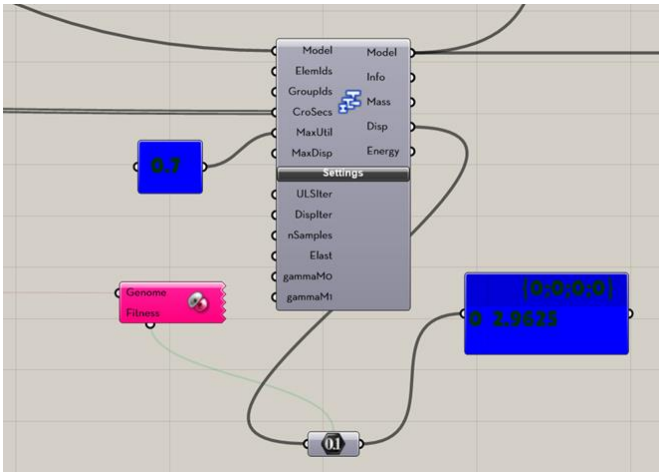
Figura 55. **Componente Galapagos y conexión de Genome**



Fuente: elaboración propia, utilizando Grasshopper.

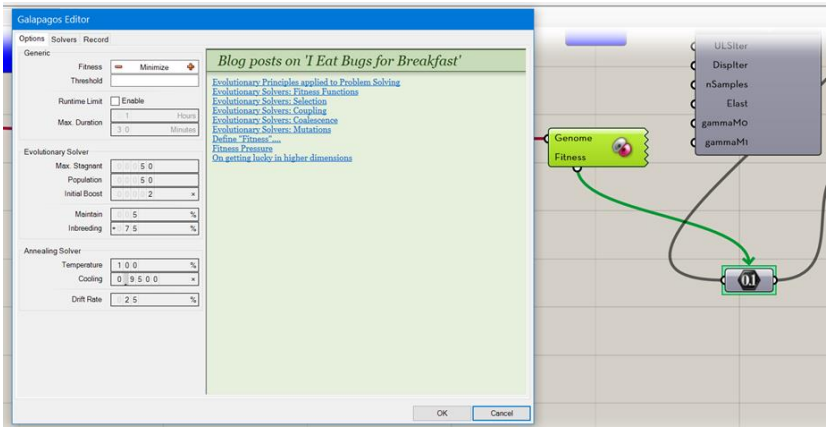
Posteriormente, haciendo doble clic en el componente Galápagos, dentro del cuadro de diálogo Galápagos Editor se ingresó a sus opciones para indicarle al programa que se estaba tratando con un proceso de optimización en que el que se deseaba minimizar una cantidad de respuesta de la estructura. Esto puede observarse en la figura 57.

Figura 56. **Componente Galápagos y conexión de Fitness**



Fuente: elaboración propia, utilizando Grasshopper.

Figura 57. **Componente Galápagos y conexión de Fitness**



Fuente: elaboración propia, utilizando Grasshopper.

5. PRESENTACIÓN Y DISCUSIÓN DE RESULTADOS

5.1. Presentación de resultados

Los resultados de interés para esta investigación fueron tanto cualitativos, como cuantitativos. Por un lado, se pretendía echar luz sobre la tecnología que hace posible es diseño asistido por algoritmos y el diseño generativo, resaltando el alto grado de automatización y parametrización que puede lograrse, no solo en el desarrollo de geometría paramétrica 3D, sino también, en el análisis estructural y optimización de estructuras. En términos generales, estos resultados se fueron demostrando en la totalidad del capítulo 4.

Por otro lado, existen unos resultados de naturaleza cuantitativa que provienen del análisis estructural y del proceso de optimización al que se sometió la estructura en estudio. La figura 56 muestra el resultado del componente Optimize Cross Section, el cual se conecta una componente tipo panel para poder tomar una lectura y desplegar los resultados. Además de un resumen del modelo analizado, el dato de mayor interés en es el de la sección transversal resultante que, para este caso, nos indica que es aquella que obedece al ID 10.

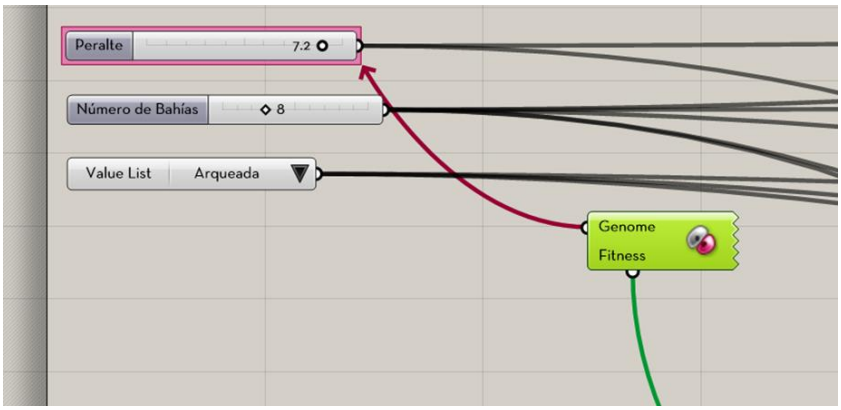
Si se echa un vistazo al componente Cross Section Selector y se establece en el Slider el número 10 (que corresponde al ID de la sección óptima) se puede observar que se trata de un perfil HSS57.2x57.2x4.8. Esto se muestra en la figura 58.

Los resultados del segundo proceso de optimización deben leerse en el componente que se conectó al slot Genome del componente Galápagos.

Después de entrar en el cuadro de diálogo Galápagos Editor y realizar las configuraciones pertinentes, Galápagos empieza el proceso de optimización y después de un tiempo, puede verse el valor óptimo al que ha llegado Galápagos en el Number Silder que maneja el valor del peralte de las armaduras. El valor al que óptimo al que se llegó se muestra en la figura 60, y corresponde a un valor numérico de 7.2 m de peralte.

Figura 58. **Resultados de interés en un análisis estructural**

Continuación figura 58.



Fuente: elaboración propia, utilizando Grasshopper.

5.2. Discusión de resultados

A lo largo de la generación del modelo de la estructura correspondiente al caso de estudio se pudo percibir que las capacidades paramétricas del modelo tridimensional (que después se usaría como modelo analítico) hacían que los cambios en los parámetros de entrada (peralte, número de bahías, secciones transversales, entre otros.) generarán una retroalimentación instantánea del estado del modelo estructural, indicando inestabilidades, estados de esfuerzo muy elevados, deformaciones muy altas, entre otros.

Lo anterior hizo que, antes de comenzar con el proceso de optimización, fuera posible la exploración de una variedad de alternativas para la configuración del conjunto de armaduras, probando una cantidad considerable de escenarios examinando qué pasaba al aumentar y disminuir el número de bahías; cambiar el tipo de armadura de Warren a Howe, de Howe a Pratt; qué sucedía al cambiar la separación entre las armaduras, la cantidad de estas y el número de peraltes. La totalidad del testeo de estos escenarios dependía únicamente de mover componentes en el lienzo de Grasshopper del tipo Number Slider,

consecuentemente estos cambios ocurrían en tiempo real, observando los resultados en el entorno de modelado de Rhino.

Después de una serie de observaciones sobre cómo respondía la estructura cuando se imponían ciertas cargas a ella, y prestando especial atención a su estado de esfuerzos y la configuración deformada adoptada, se optó por elegir una armadura arqueada, cuyo peralte y tamaño de secciones transversales serían objeto de optimización.

Por último, una vez que se probó una cantidad significativa de escenarios y configuraciones geométricas para el conjunto de armaduras bajo estudio, se sometió la estructura a un análisis de optimización, el cual dio como resultado una sección HSS57.2x57.2x4.8 y un peralte de 7.20 m, para las cuales se obtenía una utilización de 0.70 y la deflexión mínima posible.

Si bien el proceso de optimización tiene como objetivo encontrar un peralte óptimo con el que la estructura tuviera una deflexión mínima, también se pudo haber considerado otro genoma, de tal manera que se buscara obtener un número de bahías óptimo con el cual se consiguiera una deflexión mínima.

Aunque el proceso de optimización, como tal, requirió el uso de un componente de Grasshopper y la ayuda de unos cuantos clics y configuraciones, no habría sido posible si antes no se desarrollaba un algoritmo adecuado que gobernara sobre los parámetros y geometría del modelo analítico y que, por supuesto, tuviera una minuciosa consideración sobre las estructuras de datos que se estaban manipulando.

Si bien la estructura asociada al caso de estudio abordado en este trabajo de investigación fue el de un conjunto de armadura que soportan una cubierta, la

aplicación del diseño generativo tiene cabida en cualquier tipo de estructura como puentes carreteros, torres de tanques elevados de almacenamiento, las estructuras tipo *diagrid*, torres de telecomunicaciones o de transmisión eléctrica, entre otros. En suman, toda estructura susceptible de modelarse paramétricamente cae dentro de las aplicaciones del diseño generativo.

Como se ha venido discutiendo desde los primeros capítulos las ventajas de las que se puede gozar aplicando un enfoque generativo en lugar de uno tradicional son las siguientes:

- Evaluar una gran cantidad de alternativas asociadas a la configuración estructural de un proyecto, lo que conduce a una preselección de opciones prometedoras para someterlas a análisis posteriores. En una forma mucho más rápida y eficaz comparada con las tecnologías tradicionales.

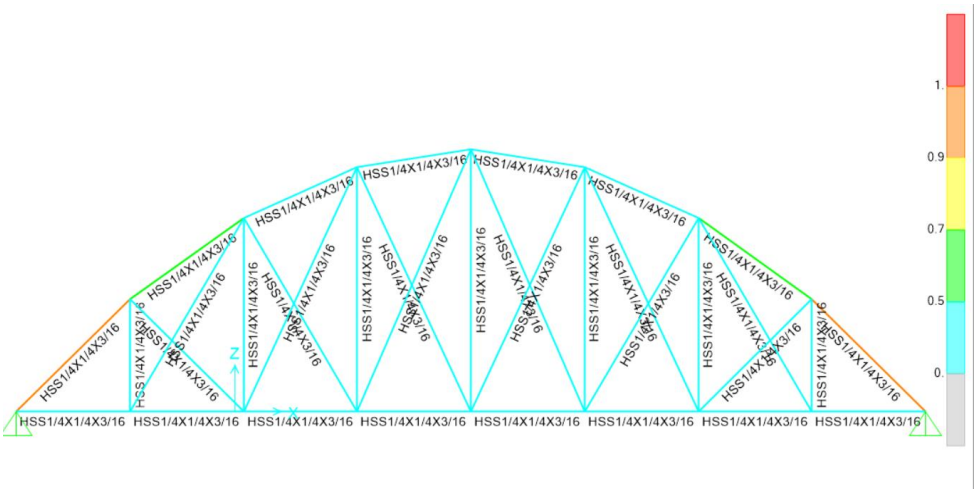
- Probar y explorar una variedad de escenarios en lo que respeta a las condiciones de carga impuestas u otras acciones impuestas a la estructura en una fracción de tiempo que el diseño tradicional.

- Optimizar de manera casi automática una cantidad importante de parámetros críticos dentro del análisis de la estructura al imponerse un número notable de restricciones, superando con creces al enfoque y software tradicionales en los que, de forma iterativa y paso a paso, sólo se logra optimizar una variable (por lo general, el tamaño de una sección transversal) sobre la base de una única restricción (que la estructura se encuentre dentro de los límites aceptables de la relación demanda/capacidad)

5.2.1. Discusión sobre la sección y el peralte obtenido

Como se dijo en los párrafos anteriores, la sección obtenida del proceso generativo fue una cuadrada tipo HSS57.2x57.2x4.8 (2¼x 2¼x3/16, en el sistema inglés). Para ver la congruencia de los resultados obtenidos se modeló la configuración más prometedora arrojada por Karamba3D y Grasshopper en el software tradicional SAP2000, obteniendo unos resultados aceptables dentro de las relaciones demanda/capacidad permisibles y acercándose, en promedio, a la relación de 0.70 impuesta como restricción en el enfoque generativo desarrollado en Grasshopper y Karamba 3D. La figura 61 muestra gráficamente lo descrito en este párrafo.

Figura 59. **Relaciones demanda/capacidad en SAP2000**



Fuente: elaboración propia, utilizando SAP2000.

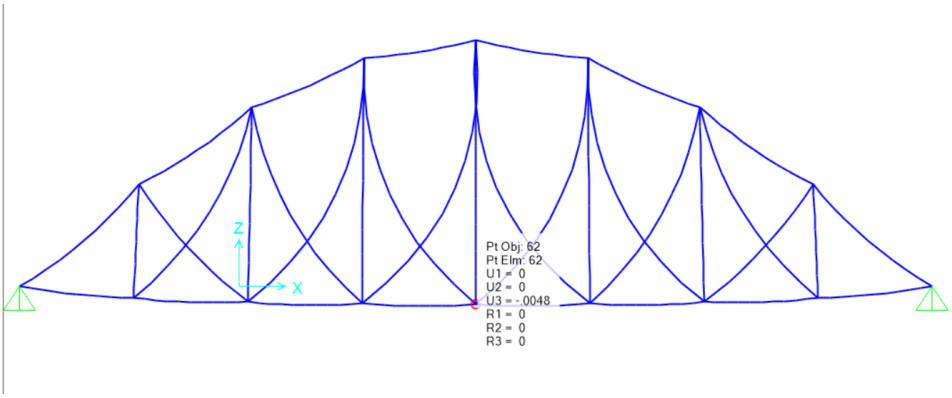
En cuanto al valor numérico del peralte obtenido del proceso basado en solucionadores evolutivos y que corresponde a 7.20 m, se puede tener la sensación de puede ser ligeramente mayor a los obtenidos en proyectos similares, ya que al abordar este tipo de estructuras suele comenzarse un análisis

preliminar con un predimensionamiento basado en relaciones de claro/peralte de 8 a 10. Dando como resultado un valor aproximado de 3.00 m que, dependiendo del estado de cargas, puede llegar a un valor de hasta 5.00 m o 6.00 m.

La razón de esta discrepancia es que en los diseños habituales el estado límite de servicio, evaluado a través de la fórmula L/180 y comparado con la deflexión debido a la carga viva, usualmente se cumple un margen de seguridad no muy grande. Por otro lado, operando bajo un escenario bastante restrictivo en el que el estado límite de servicio fuese de una importancia yugular, es decir, en el que se desease reducir el valor numérico deflexión vertical lo más posible, y tomando en cuenta tanto la carga viva como la carga muerta, se necesitaría de un peralte mayor para cumplir este criterio de aceptación. Por ello, para este hipotético escenario se obtuvo el resultado de un peralte mayor que lo usual, pero con un valor de deflexión muy pequeño.

La siguiente figura muestra el valor de la deflexión vertical experimentada (en metros) en la armadura para el caso de carga viva más carga muerta, en la cual la deflexión se denota por Uz.

Figura 60. **Deformación máxima en SAP2000**



Fuente: elaboración propia, utilizando SAP2000.

CONCLUSIONES

1. El diseño generativo pone en las manos del ingeniero estructural una poderosa herramienta para la automatización y optimización de algunas tareas relacionadas con el análisis estructuras. Sin embargo, es importante resaltar que, con estas bondades, viene una serie de conocimientos y un conjunto de habilidades que el ingeniero deberá aprender, principalmente la capacidad para codificar y pensar computacionalmente.

2. Dentro de la variedad de software que existen en la actualidad para abordar el tema del diseño asistido por algoritmos, se hizo énfasis en Rhino y Grasshopper, demostrando claramente lo importantes que resultan las capacidades paramétricas de estas plataformas, el beneficio de explorar distintas alternativas, probar una variedad de escenarios y sobre todo dar luz sobre el proceso de optimización con solucionadores evolutivos.

3. Al abordar el desarrollo de algoritmos visuales, se observa que hay algunas directrices y buenas prácticas a seguir para construir un código que funcione de manera correcta y eficiente. Además del conocimiento sólido que se debe tener sobre los componentes de Grasshopper y Karamba 3D, debe tenerse especial cuidado en las estructuras de datos asociadas con las entradas y salidas de los componentes, de tal manera que logren acomodarse para funcionar correctamente.

4. Las capacidades paramétricas inherentes a Grasshopper, así como la posibilidad de añadir varias funciones relacionadas con la ingeniería a través de *plugins,* permite al proyectista estructural agilizar y optimizar sus diseños, escogiendo entre la mejor alternativa, una vez que se ha explorado una variedad de escenarios relevantes en una fracción del tiempo que le tomaría hacerlo en programas más tradicionales.

RECOMENDACIONES

1. Los profesionales de la práctica contemporánea de la ingeniería estructural deben incluir dentro de su formación autodirigida, algunos tópicos relacionados a la codificación para poder abordar correctamente los temas del diseño asistido por algoritmos y el diseño generativo y mejorar sus habilidades tecnológicas.

2. Es conveniente evaluar con pensamiento crítico las metodologías y herramientas digitales que pueden utilizarse en el diseño generativo, así como, discernir en qué tipos de problemas puede ser adecuado usar el software asociado con el diseño asistido por algoritmos y sopesar las alternativas de diseño prometedoras.

3. Prestar especial cuidado a las estructuras de datos con las que se tratan en el uso de software de programación visual, ya que un descuido en este sentido podría desencadenar resultados no deseados y en ocasiones, errores imperceptibles en el algoritmo, pero de implicaciones desfavorables en el modelo digital.

4. Es necesario mantenerse al tanto y actualizado con los avances tecnológicos que se hacen en la industria AEC, con particular atención a las herramientas y metodologías en constante evolución, que puedan constituir una oportunidad para agilizar y optimizar diseños dentro de la práctica.

REFERENCIAS

1. ADIT AUTOR, S. S. (25 de Abril de 2018). *STUDIO SEED*. [Mensaje en un blog]. Recuperado de http://www.studioseed.net/blog/software-blog/parametric-generative-design-blog/grasshopper/una-adicion-moderna-en-el-skyline-de-tailandia-la-pixel-tower/

2. AISC. (2017). *Steel Construction Manual.* United State of America: Autor.

3. Burr, M. (2011). *Scripting Cultures: Architectural Design and Programming.* United Kingdom: John Wiley & Sons.

4. Cremer, D. D. (2020). *Leaderhship by Algorith: Who Leads and Who Follows in the AI Era?* Great Britain: Harriman House.

5. Deutsch, R. (2017). *Convergence: The Redesign of Design.* Estados Unidos: John Wiley & Sons.

6. Dynamo, A. (s.f.). *Dynamobim*. [Mensaje en un blog]. Recuperado de https://primer.dynamobim.org/01_Introduction/11_what_is_visual_programming.html

7. Garber, R. (2014). *BIM Design: Realising the Creative Potential of Building Information Modeling.* United Kingdom: Wiley.

8. Hall Cormen, C. E. (2009). *Introduction to Algorithms.* USA: MIT Press.

9. Hanif Cara, D. B. (2016). *Design Engineering Refocused.* United Kingdom: John Wiley & Sons.

10. Heineman, G. (2020). *Learning Algorithms.* United Kingdom: O'Reilly Media.

11. Jabi, W. (2013). *Parametric Design for Architecture.* United Kingdom: Laurence King Publishing.

12. Krygiel, E. (2008). *Green BIM: Successful Sustainable Design with Building Information Modling.* United Kingdom: Wiley.

13. LLC, P. G. (12 de marzo 2020). *PGAPPS*. [Mensaje en un blog]. Recuperado de https://apps.provingground.io/lunchbox/

14. Nag Miller, D. L. (2011). *Problem Solving with Algorithms and Data Structures Using Python.* United Kingdom: Franklin.

15. Nageim, H. A. (2017). *Steel Structures: Practical Design Studies.* Broken Sound Parkway NW: CRC Press, Taylor & Francis Group.

16. Nebril, V. I. (3 de febrero 2018). *Modelización Bioclimática*. [Mensaje en un blog]. Recuperado de http://oa.upm.es/51453/1/TFG_Irala_Nebril_V%C3%ADctorop.pdf

17. Pramod, S. (02 de Marzo de 2019). *Medium - Diseño Generativo y*

Análisis Espacial Métrico. [Mensaje en un blog]. Recuperado de https://towardsdatascience.com/generative-design-metric-space-analysis-e809d949401f

18. Presigner, K. -C. (s.f.). *Karamba3D.com*. [Mensaje en un blog]. Recuperado de https://manual.karamba3d.com/

19. Romero, E. A. (17 de Junio de 2020). *LinkedIn*. [Mensaje en un blog]. Recuperado de https://www.linkedin.com/pulse/qu%25C3%25A9-es-realmente-bim-building-information-modeling-erick-mor%25C3%25A1n/?trackingId=8Ry%2FwuVpQEKCGzwZH0%2FQ7w%3D%3D

20. Rutten, D. (4 de Marzo de 2011). *I Eat Bugs for Breakfast*. Obtenido de https://ieatbugsforbreakfast.wordpress.com/2011/03/04/epatps01/

21. Silver, M. (2006). *Programming Cultures: Ast and Architecture in the Age of Software.* United Kingdom: Wiley-Academy.

22. Spraul, V. A. (2012). *Think Like a Programmer: An Introduction to Creative Problem Solving.* Inglaterra: No Starch Press.

23. Tedre, P. J. (2019). *Computational Thinking.* Estados Unidos: The MIT Press Essential Knwoledge Series.

24. Tuckfield, B. (2021). *Dive into Algorithms: A Python Adventure for the Intrepid Beginner.* Estados Unisdos. Bradford Tuckfield.

25. Willingham, D. (s.f.). *Critical Tinking: Why Is so Hard to Teach?* [Mensaje en un blog]. Recuperado de https://www.readingrockets.org/article/critical-thinking-why-it-so-hard-teach#:~:text=Thinking%20tends%20to%20focus%20on,you%20are%20given%20a%20problem.

26. Zingaro, D. (2020). *Introduction, Algorithmic Thinking: A Problem-Based.* Estados Unisdos. Bradford Tuckfield.