APLICACIONES DE REDES NEURONALES ARTIFICIALES PARA EL DISEÑO ÓPTIMO A FLEXIÓN DE VIGAS DE HORMIGÓN ARMADO

POR

Sebastián Nicolás Parra Vergara

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero Civil

Profesor Guía
Patricio Cendoya H.

Profesionales Supervisores
Víctor Aguilar V.

Marzo, 2024
Concepción (Chile)

DEDICATORIA

Dedico este trabajo a mi querida familia, quienes siempre me han apoyado,
en especial a madre Cecilia y a mi hermana Javiera.

AGRADECIMIENTOS

A mis profesores, por haber transmitido sus conocimientos, experiencias y enseñanzas entregadas durante estos años de carrera. En especial a mi profesor guía Patricio Cendoya H., por su compromiso, compañía y buena disposición en todo momento.

A mi familia, en especial a mí mamá y hermana por todo el apoyo, cariño, sacrificio y compañía, por haber sido ejemplos de superación y haberme trasmitido esa motivación desde siempre, y en especial durante esta importante etapa en mi vida.

A mis amigos de la infancia, por su fraternal y grata compañía, además de ser un lugar seguro en donde pertenecer (TBS). A mis amigos del Discord (PC) y liceo MDCH, con quienes he compartido incontables horas juntos, los ARAMS y por haber hecho más ameno el periodo de pandemia.

A mis amigos de la carrera, en especial a Sebastián, Francisco y Brian, por todas la alegrías y angustias que compartimos, por las noches que nos quedamos calculado hasta el amanecer y por la amistad sincera que me han entregado.

RESUMEN

La aplicación de algoritmos de inteligencia artificial (IA) se ha masificado de manera exponencial en los últimos años, encontrando aplicación de estos modelos a un amplio espectro de problemas en diversas áreas de la ciencia, incluyendo la ingeniería estructural. Nos encontramos en una era de revolución tecnológica en donde las nuevas metodologías basadas en algoritmos de IA requieren de un manejo de nuevas herramientas teóricas y computacionales desarrolladas con la finalidad de agilizar los procesos de toma de decisión y optimizar recursos. En el ámbito de la IA, el uso de técnicas como los algoritmos de Redes Neuronales Artificiales (RNA) se convierte en un campo de exploración interesante.

Este trabajo aborda la formulación de un algoritmo de IA basado en RNA para el diseño a flexión de vigas rectangulares rectas de hormigón armado. Este proceso utiliza datos de vigas determinados analíticamente, empleando un modelo optimización del costo de la viga y asegurando la conformidad con las normativas vigentes. Para el entrenamiento de las redes se proporciona un total de 1302 vigas considerando diferentes condiciones de apoyo, hormigón y longitud.

Finalmente, los resultados demuestran que este algoritmo posee una satisfactoria capacidad para aprender a diseñar vigas a flexión a partir de los datos proporcionados, logrando generalizar este proceso de diseño y obteniendo errores de predicción del orden de 10^{-4} para RNA con arquitecturas de no más de siete neuronas en la capa oculta. Además, se ha implementado una interfaz gráfica de usuario que facilita el uso de estas redes neuronales artificiales, posibilitando el diseño a flexión de vigas rectas de hormigón armados sin la necesidad de poseer conocimientos en programación o estructuras.

SUMMARY

The application of artificial intelligence (AI) algorithms has experienced exponential growth in recent years, finding application of these models across a broad spectrum of problems in various areas of science and engineering. We are in an era of technological revolution where new methodologies based on AI algorithms require the handling of new theoretical and computational tools developed to streamline decision-making processes and optimize resources. In the field of AI, the use of techniques such as Artificial Neural Network (ANN) algorithms becomes an interesting area of exploration.

This work addresses the formulation of an AI algorithm based on Artificial Neural Networks (ANNs) applied to flexural design of straight rectangular reinforced concrete beams. The process involves using analytically determined beam data, employing a cost optimization model for the beam, and ensuring compliance with current regulations. A total of 1302 beams, considering different restraint conditions, concrete strength, and lengths, are provided for training the networks.

Ultimately, the results demonstrate that this algorithm possesses satisfactory capability to learn how to design beams subjected to flexure from the provided data. It successfully generalizes the flexural design process even for diverse initial conditions not considered during the training phase, achieving prediction errors on the order of 10^{-4} for architectures with no more than seven neurons in the hidden layer.

Furthermore, a user-friendly graphical interface has been implemented to facilitate the use of these artificial neural networks, enabling the flexural design of straight reinforced concrete beams without the need for programming or structural knowledge.

TABLA DE CONTENIDOS

Lista de tablas

LISTA DE FIGURAS

CAPÍTULO 1: INTRODUCCIÓN

1.1. Motivación

En los últimos años ha aumentado de crecimiento exponencial la investigación y aplicación de algoritmos de Inteligencia Artificial (IA), marcando una era de revolución tecnológica. Esto ha potenciado la implementación computacional de estos algoritmos, impulsando su uso en la realización de predicciones, clasificaciones y agrupamientos de información en diversas disciplinas científicas incluyendo la ingeniería estructural.

Este fenómeno ha permitido el desarrollo de múltiples algoritmos dentro de los que se destacan las Redes Neurales Artificiales (RNA), los cuales son modelos de inteligencia artificial capaces de abordar tanto problemas lineales como no lineales (Goodfellow, 2016). Las RNA construyen sistemas complejos compuestos por múltiples neuronas, lo que facilita la minimización de los procesos de decisión humana (Vasquez, 2022). En el área de la Ingeniería Estructural diversos investigadores han reportado aplicaciones para el diseño estructural (Senouci, 2000; Zarringol, 2021), resistencia de materiales (Navarro, 2016; Chopra, 2018), detección de daño en elementos (Thai, 2022), etc.

El presente trabajo busca desarrollar y aplicar una herramienta de IA basadas en algoritmos de RNA en entornos MATLAB® para el diseño a flexión de vigas rectangulares, estimando dimensiones y cuantías de armaduras longitudinales considerando un costo mínimo de acuerdo con los requerimientos del código ACI 318-19 (comité ACI 318, 2019).

Temas Claves: Algoritmos de ML, RNA, código ACI 318-19, diseño a flexión de vigas de hormigón armado.

1.2. Objetivos

1.2.1 Objetivo general

Aplicar algoritmos de redes neuronales artificiales para el diseño óptimo a flexión de vigas de hormigón armado de acuerdo con el código ACI 318-19.

1.2.2 Objetivos específicos

1. Revisar los alcances de las RNA en aplicaciones a la Ingeniería Estructural.

2. Estudiar los principales fundamentos matemáticos de las RNA y su implementación computacional en entornos MATLAB®.

3. Construir y entrenar RNA que permitan estimar las dimensiones óptimas de vigas rectangulares a flexión de hormigón armado de acuerdo con lo indicado en ACI 318-19.

4. Realizar un análisis de sensibilidad de las RNA con mejor rendimiento para evaluar su desempeño en diferentes condiciones.

5. Desarrollar una interfaz gráfica capaz de aplicar las RNA entrenadas.

1.3. Plan de trabajo

En el desarrollo de esta memoria inicialmente se realiza una revisión del estado del arte de las redes neuronales artificiales, en el contexto de la inteligencia artificial, y su aplicación a problemas de ingeniería estructural, con especial énfasis en algoritmos para el diseño a flexión de vigas de H-A.

Posteriormente se desarrolla un base de datos que 1302 vigas diseñadas a flexión que consideran diferentes condiciones de apoyo y longitud para cargas uniformemente distribuidas, las cuales son

utilizadas para entrenar diferentes redes neuronales. Logrando obtener RNA capaces de realizar diseños a flexión eficientes de vigas rectangulares de hormigón armado, obteniendo las dimensiones de la sección transversal y sus respectivas armaduras longitudinales. Como resultado de este entrenamiento se determina la arquitectura que mejor representa a cada caso de estudio para analizar la sensibilidad de sus variables de salida en función de las variables de entrada.

Finalmente, se confecciona una interfaz gráfica en MATLAB® en donde un usuario sea capaz de ingresar los valores de las variables de entrada y obtenga las dimensiones, capacidad y costo de la viga óptima.

1.4. Principales resultados

Como resultado del entrenamiento de 63 redes neuronales artificiales con 1302 vigas, se identifican las arquitecturas que exhiben una mayor calidad en la predicción de datos, lo que implica una representación más precisa del problema.

Posteriormente, se lleva a cabo un análisis de sensibilidad de las variables de salida más relevantes, tales como Alto, Área de acero longitudinal, resistencia de diseño y costo en relación con las variables de entrada, como la carga distribuida y el largo de la viga demostrando la robustez del modelo.

Finalmente, se concreta la aplicación de estas RNA a través de una interfaz gráfica.

1.5. Organización de la memoria

Este trabajo se estructura en base a seis capítulos, de los cuales el primero es introductorio, presentando la motivación, el objetivo general y los específicos, la metodología de trabajo y los principales resultados obtenidos. El segundo capítulo abarca una revisión del estado del arte de las redes neuronales artificiales y sus aplicaciones en la Ingeniería Estructural. En el capítulo tres se desarrollan los fundamentos teóricos y matemáticos de las redes neuronales artificiales. El capítulo cuatro presenta la formulación de la base de datos de vigas rectangulares de hormigón armado

diseñadas a flexión según un modelo de optimización del costo de la viga. Luego en el quinto capitulo se desarrolla una aplicación de RNA para diseñar a flexión vigas de hormigón armado a partir de los resultados obtenidos en el capítulo cuatro. Finalmente, en el capítulo seis se exponen las principales conclusiones de este trabajo.

CAPÍTULO 2: ESTADO DEL ARTE DE LAS REDES NEURONALES ARTIFICIALES

2.1. Introducción

El presente capítulo abarca una revisión general del estado del arte de las Redes Neuronales Artificiales, su clasificación dentro de los algoritmos de Machine Learning y sus principales aplicaciones en la Ingeniería Estructural.

En los últimos años ha crecido de manera exponencial la cantidad de investigaciones con directrices en la aplicación de algoritmos de inteligencia artificial para realizar predicciones y clasificar información. Esto ha permitido impulsar significativamente la implementación computacional de dichos algoritmos, de los cuales se destaca las Redes Neuronales Artificiales (RNA), las cuales son capaces de llevar a cabo ambas tareas, brindando soluciones a una amplia diversidad de tópicos.

2.2. Aplicaciones del Machine Learning en la Ingeniería Estructural

Los algoritmos de Machine Learning (ML) constituyen una categoría de Inteligencia Artificial (IA) que se centra en enseñar a la computadora a realizar tareas a partir de conjuntos de datos. Esta capacidad otorga al sistema la habilidad de aprender y mejorar de forma autónoma, evitando la necesidad de una programación explícita (Thai, 2022).

Los algoritmos de ML se clasifican según su tipo de entrenamiento (ver Figura 2.1). En primer lugar, está el aprendizaje supervisado en donde una base de datos posee valores de entradas y salidas previamente etiquetadas, este método de aprendizaje se utiliza para problemas de regresión (determinar valores numéricos de salida) y clasificación (determinar una etiqueta a las variables de salida). En segundo lugar, se encuentra el aprendizaje no supervisado el cual posee una base de datos con entradas, pero sin salidas, en donde los algoritmos se encargan de asignar una etiqueta a partir del reconocimiento de patrones, estos algoritmos se utilizan para problemas de agrupamiento de datos. En tercer lugar, se encuentra el aprendizaje por refuerzo el cual se basa en el aprendizaje “prueba y error” para determinar algún patrón principalmente de optimización (Thai, 2022; Vásquez, 2022).

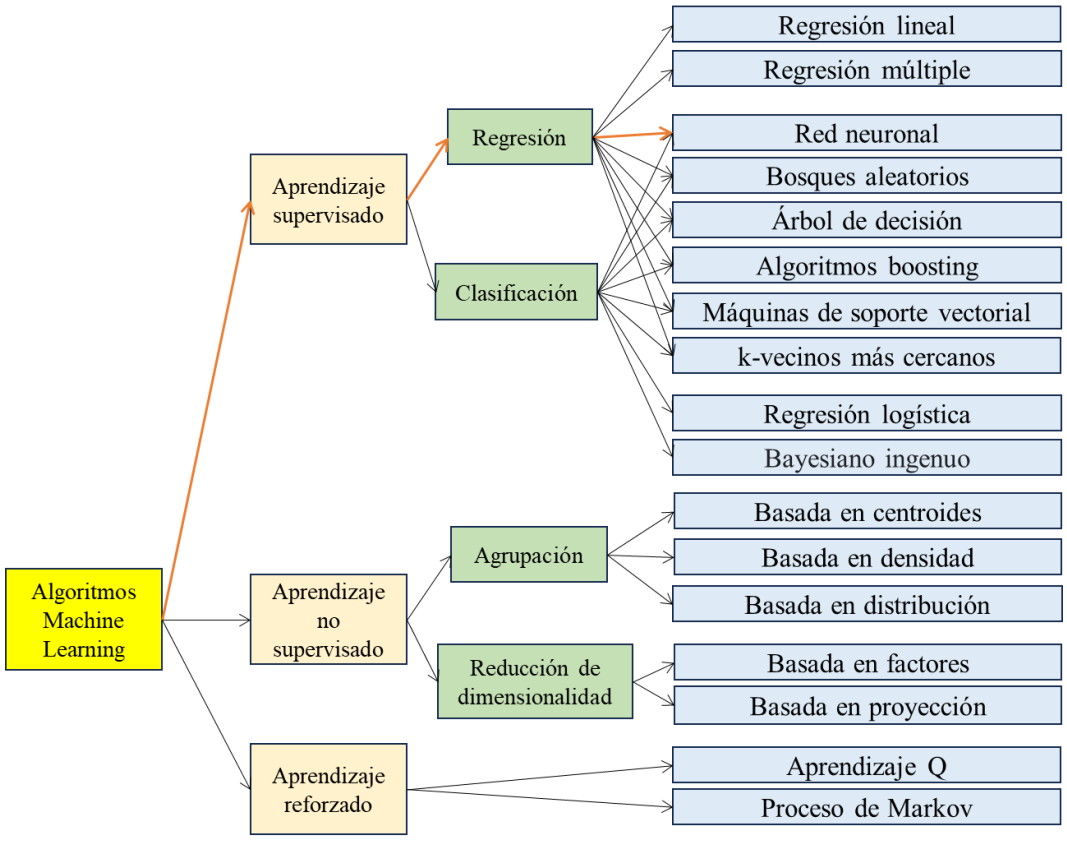

Figura 2.1 Algoritmos de ML según su tipo de aprendizaje (Thai, 2022)

En los últimos años se ha masificado la aplicación de algoritmos de ML en problemas de Ingeniería Estructural (ver Figura 2.2), en donde se demuestra un incremento exponencial en la cantidad de publicaciones desde el año 2016 asociadas a estas aplicaciones. Se destaca los algoritmos basados en Redes Neuronales (RN) (ver Figura 2.3 a)), de los cuales las Redes Neuronales Artificiales (RNA) y Convolucionales (RNC) figuran como los más aplicados (Thai, 2022).

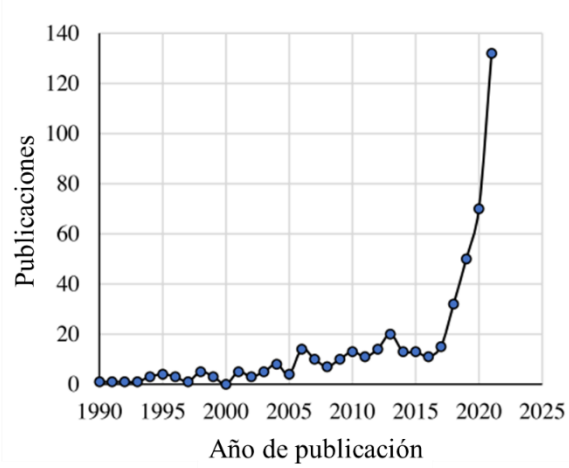

Figura 2.2 Publicaciones relacionadas a aplicaciones de ML en la Ingeniería Estructural (Thai, 2022)

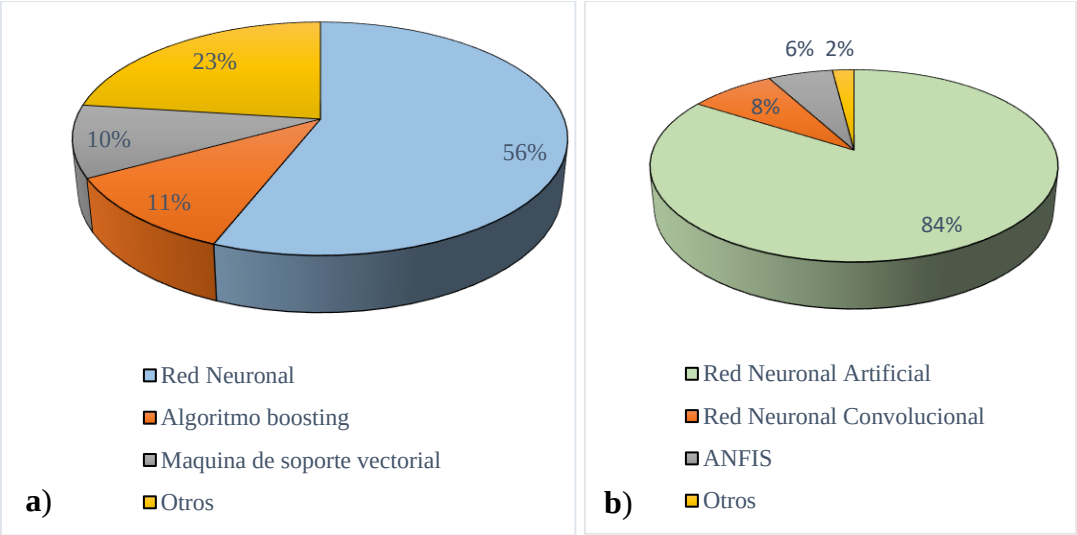

Figura 2.3 a) Algoritmos de ML aplicados en la Ingeniería Estructural, b) Algoritmos de Redes Neuronales más relevantes. (Thai, 2022)

2.3. Aplicaciones de redes neuronales artificiales en la Ingeniería Estructural

Thai (2022) menciona diversas publicaciones en las cuales se estudian diferentes algoritmos de ML en problemas de la Ingeniería Estructural y resistencia de materiales (ver Tabla 2.1), agrupándolas según el elemento estructural, material base y condición de estudio.

Tabla 2.1 Aplicaciones de RNA a elementos y sistemas estructurales

Elemento / Sistema	Material	Caso de estudio
Columna	Hormigón y Acero	Resistencia al fuego
		Resistencia compresión axial
		Resistencia al pandeo
Losa	Hormigón	Resistencia al fuego
		Detección de daño
		Resistencia al corte
		Resistencia a la flexión
Muros	Hormigón y Albañilería	Resistencia al corte
		Resistencia a la compresión
		Resistencia a la flexión
Vigas	Hormigón y Acero	Resistencia al fuego
		Detección de daño
		Resistencia al corte

		Resistencia al pandeo
		Resistencia a la flexión
		Serviciabilidad
Unión Viga-Columna	Hormigón y Acero	Resistencia al corte
		Detección de daño
Enrejados	Acero	Resistencia al fuego
		Detección de daño
Marcos	Acero y Albañilería	Resistencia al fuego
		Detección de daño
Puentes	Hormigón y Acero	Detección de daño

Otro caso de aplicación de las RNA en el contexto de la ingeniería estructural corresponde al estudio de la resistencia de los materiales (ver Tabla 2.2) siendo la predicción de la resistencia a compresión axial de hormigones basado en modelos de redes neuronales artificiales uno de los temas más citados. (Thai, 2022).

Tabla 2.2 Aplicaciones de RNA a problemas de resistencia y propiedades de materiales

Material	Caso de estudio
Hormigón de alto desempeño	Resistencia a la compresión axial
	Resistencia a la tensión axial
	Diseño de mezcla
Hormigón de alta resistencia	Resistencia a la compresión axial
	Módulo de Young
Hormigón de resistencia normal	Resistencia a la compresión axial
	Módulo de Young
	Diseño de mezcla
Hormigón ligero	Resistencia a la compresión axial
	Módulo de Young
Hormigón reforzado con fibra	Resistencia a la compresión axial
	Resistencia a la tensión axial
	Resistencia a la flexión
Hormigón con geopolímeros	Resistencia a la compresión axial
	Resistencia a la flexión
Hormigón auto-compactante	Resistencia a la compresión axial
Hormigón con agregado reciclado	Resistencia a la compresión axial
	Resistencia a la tensión
	Módulo de Young
	Diseño de mezcla

Cabe señalar que en este trabajo se desarrolla la aplicación de RNA al diseño a flexión de vigas de hormigón armado. En este contexto, una de las primeras aplicaciones a este tipo de elementos corresponde a los trabajos de Mukherjee y Deshpande (1995), en donde se presenta el diseño preliminar a flexión de vigas simplemente apoyadas a partir de una base de datos analítica. Posteriormente Senouci (2000), replicaría esta investigación en el año 2000, detallando más sus resultados. En 2019 Kotsovou y colaboradores (2019) publican un estudio comparativo del comportamiento de vigas sometidas a flexión con datos experimentales y teóricos, utilizando una base de datos asociada a vigas conectadas a columnas.

2.4. Conclusiones

En los últimos años se ha presenciado un incremento significativo en la cantidad de estudios de aplicación de los algoritmos de Machine Learning en campos de la Ingeniería Estructural, predominando las Redes Neuronales Artificiales por sobre otros algoritmos, esto puede atribuirse a que una gran cantidad de problemas de interés pueden ser generalizados en función de una cantidad finita de variables de entrada y variables de salida. Adicionalmente, muchos criterios de diseño provienen de la interpretación de resultados experimentales y es aquí donde la capacidad de regresión de las Redes Neuronales Artificiales toma relevancia pues, estos algoritmos llegan como un potencial reemplazo de las técnicas clásicas de regresión.

CAPÍTULO 3: MARCO TEÓRICO DE LAS REDES NEURONALES ARTIFICIALES

3.1. Introducción

Las RNA son un algoritmo estructurado por neuronas y enlaces capaces de realizar regresiones o clasificaciones de información mediante un proceso de aprendizaje basado en algoritmos iterativos que calibran parámetros de ajuste en función de una base de datos provista para el entrenamiento de la red.

El presente capítulo introduce el concepto teórico de Red Neuronal Artificial, sus capacidades, su arquitectura, además del algoritmo de aprendizaje y de entrenamiento. Por último, su implementación computacional mediante el Toolbox Neural Network Fitting de la versión R2020a de MATLAB®

3.2. Principio de las redes neuronales artificiales

Las RNA corresponden a uno de los algoritmos de Aprendizaje Automático (Machine Learning) que se inspira en el comportamiento y representación gráfica de una red neuronal biológica (Bojórquez et al., 2016).

Las RNA son adecuadas para la clasificación y la regresión de datos, en donde, la clasificación corresponde a la asignación de una etiqueta o categoría a una entrada (ej. Clasificar si un mensaje es spam o no, etiquetar una radiografía con su diagnóstico o asignar el mecanismo de falla de un elemento estructural a partir del tipo de grieta que posea), por otra parte, la regresión se enfoca en la predicción de valores numéricos a partir de información de entrada descrita en una base de datos (Thai, 2022).

La base de datos es la columna vertebral de las RNA, esta posee la responsabilidad de generalizar la situación de estudio y de proporcionar la información de entrenamiento a la red hasta que esta sea capaz de predecir adecuadamente los datos provenientes a las entradas y salidas, obteniendo así predicciones coherentes para el problema en cuestión. Las bases de datos en el contexto de la

Ingeniería Civil se clasifican según su origen, entre estas se identifican bases empíricas las cuales provienen de resultados obtenidos de manera experimental en una condición de laboratorio (ej. Ensayos de elementos estructurales y ensayos de resistencia de materiales), y las bases analíticas provenientes de resultados matemáticos obtenidos tras la aplicación de ecuaciones provenientes de la teoría, códigos o normas que gobiernan la física del problema.

3.3. Arquitectura de una red neuronal artificial

Las RNA se constituyen como una red de nodos o neuronas artificiales las cuales se encuentran interconectadas mediante pesos sinápticos, agrupando a las neuronas en "capas" donde cada neurona perteneciente a una capa se encuentra conectada con todas las neuronas de la capa anterior y siguiente, sin embargo, no se establece conexión entre las neuronas de una misma capa (ver Figura 3.1).

Se identifican tres tipos de capas, la capa de entrada encargada de recibir los datos, la capa oculta que procesa la información obteniendo valores en función de pesos, sesgo y función de activación (ver Figura 3.2) y la capa de salida responsable de entregar los resultados de predicción (ver Figura 3.1).

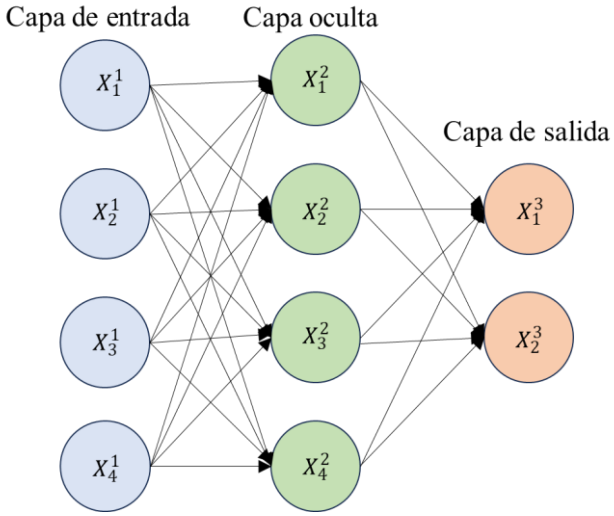

Figura 3.1 Arquitectura de una Red Neuronal Artificial

La información proveniente de las neuronas de la capa de entrada se transmite hacia las neuronas de las capas ocultas a través de las funciones de propagación y activación. La función de propagación se

define como la sumatoria del producto entre las salidas de la capa anterior y los pesos conectados a la neurona, y la función de activación determina el valor de salida de cada neurona. Si la información se propaga "hacia adelante" desde una capa a otra la estructura se conoce como "Feed-Forward Propagation" (ver Figura 3.2).

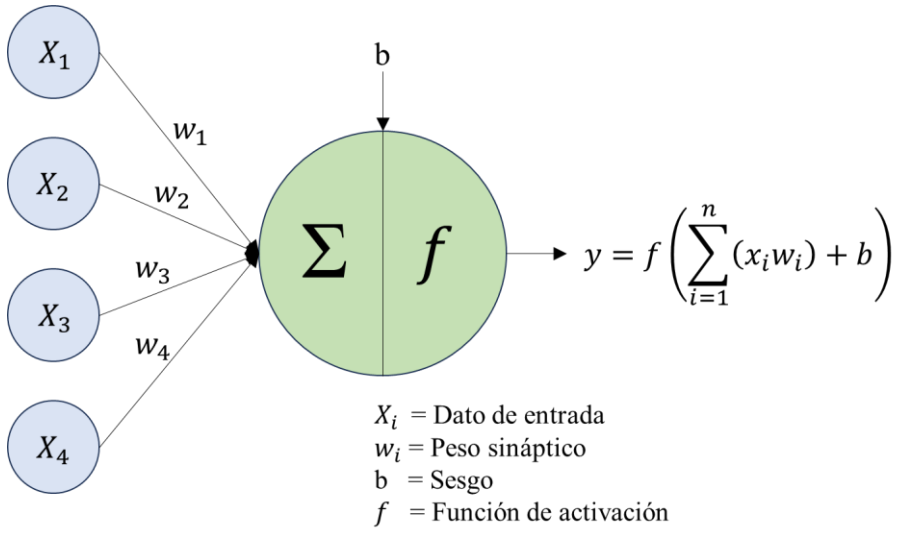

Figura 3.2 Esquema de Funcionamiento de una neurona artificial

El valor de salida de cada neurona se obtiene de aplicar la función de activación al valor obtenido de la función de propagación y el sesgo, entregando como resultado un valor perteneciente un dominio acotado definido según la función de activación utilizada.

$$Y_j^L = f\left(\sum_k w_{j,k}^L X^{L-1}{}_k + b_j^L\right) \tag{3.1}$$

Donde Y_j^L = es la salida de una neurona j de la capa L

$w_{j,k}^L$ = es el peso que conecta la neurona k con la neurona j

$X^{L-1}{}_k$ = es la salida de la neurona k de la capa L-1

b_j^L = es el sesgo asociado a la neurona j de la capa L

f = es la función de activación

Las principales funciones de activación son la función sigmoidea (logsig) que acota la salida en el intervalo]0,1[(ec 3.2), la tangente hiperbólica (tansig) que acota la salida en el intervalo]-1,1[(ec.

3.3) y la función lineal (Purelin) con salidas que pertenecen al conjunto de los números reales R (ec 3.4) (Zarringol, 2021) (ver Figura 3.3).

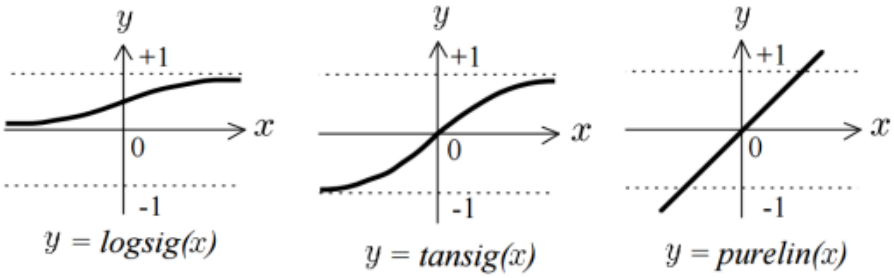

Figura 3.3 Principales funciones de activación (Navarro, 2016)

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{3.3}$$

$$f(x) = x \tag{3.4}$$

Previo al desarrollo del entrenamiento de la RNA, es necesario normalizar la base pues generalmente los valores de los datos de entrada y salida no poseen ordenes de magnitud similares (Abellán, 2021), este preproceso es fundamental para obtener resultados satisfactorios en la etapa de entrenamiento.

$$x_{norm_{logsig}} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.5}$$

$$x_{norm_{tansig}} = 2\frac{x - x_{min}}{x_{max} - x_{min}} - 1 \tag{3.6}$$

La estructura de las RNA se conoce como “Feed-Forward Perceptrón Multicapa”, pues la información se trasmite hacia adelante (Feed-Forward) a través de capas compuestas por neuronas denominadas Perceptrones (Bojórquez et al., 2016).

Considerar una única capa oculta en la RNA es suficiente para la gran mayoría de los problemas de ingeniería (Bojórquez et al., 2016 y Navarro, 2016), por otra parte, la cantidad de neuronas en la capa

oculta no posee un mínimo o un máximo. Determinar la arquitectura que mejor ajusta al modelo corresponde a un proceso iterativo basado en el aumento de la cantidad de neuronas en la capa oculta.

3.4. Modelo matemático de red neuronal artificial

El modelo de RNA se encuentra compuesto por matrices y vectores. El vector de valores de entrada (X_1) posee dimensiones 1xM donde M es la cantidad de entradas, la matriz de pesos que conecta la capa de entrada con la capa oculta (IW) posee dimensiones MxN, con N la cantidad de neuronas en la capa oculta y la matriz de pesos que conecta la capa oculta con la capa de salida (LW) posee dimensiones NxO, donde O es la cantidad de salidas. Por otra parte, se obtienen los vectores de sesgos B_1 y B_2 de dimensiones 1xN y 1xO respectivamente.

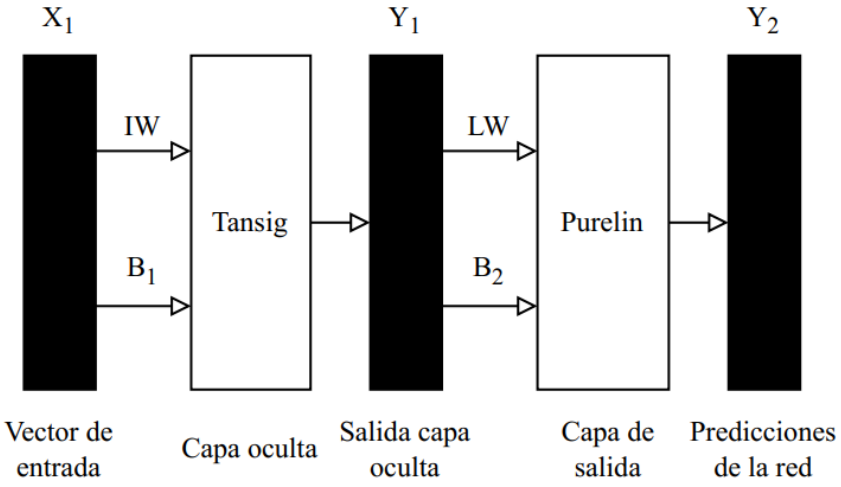

Figura 3.4 Modelo matemático de una RNA

De esta manera la (ec. 3.7) generaliza el vector de salidas (Y_2) en función de las entradas, pesos y sesgos para una red con una capa oculta.

$$Y_2(X_1, IW, LW, B_1, B_2) = Purelin(Tansig(X_1 \cdot IW + B_1) \cdot LW + B_2) \quad (3.7)$$

3.5. Aprendizaje de una red neuronal artificial

El aprendizaje de una RNA se basa en mejorar su desempeño a la hora de reproducir los valores provenientes de la base de datos, es decir, reducir el error de regresión.

El error de la red es obtenido al comparar los valores predichos por regresión y los valores de salida provenientes de la base de datos para calcular el error (costo o perdida) asociado a cada dato predicho (Navarro, 2016). La función de error utilizada en las RNA aplicadas a problemas de regresión se conoce como Error Cuadrático Medio, o de sus siglas en ingles MSE (Mean Squared Error), este corresponde a la ponderación de los errores medidos para cada dato de salida.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - y_i)^2 \tag{3.8}$$

Donde MSE = función error cuadrático medio

N = cantidad de salidas

t_i = valor objetivo de la muestra i

y_i = valor predicho de la muestra i

El aprendizaje de la RNA se basa en minimizar el valor de la función error, esto se logra estimando la combinación de valores de pesos (IW y LW) y sesgos (B_1 y B_2) hasta alcanzar un mínimo relativo o absoluto de la función MSE.

El mínimo de una función se determina cuando la derivada de esta alcanza el cero, en este caso $\nabla(MSE) \approx 0$ pues la función MSE se encuentra expresada en función de las predicciones de la RNA (ec 3.8), las cuales, a su vez, dependen de los pesos y sesgos (ec 3.7).

$$\nabla(\text{MSE}) = \left(\frac{\partial MSE}{\partial w_{i,j}^{L}}, \ldots, \frac{\partial MSE}{\partial b_i^{L}}\right) \approx 0 \tag{3.9}$$

Donde $\nabla(MSE)$ = corresponde al gradiente de la función error.

$w_{i,j}^{L}$ = corresponde al peso que conecta la neurona i a la neurona j de la capa L.

b_i^L = corresponde al sesgo de la neurona i de la capa L.

En una primera iteración la asignación de pesos y sesgos es de forma aleatoria, obteniendo así un valor de MSE para esa combinación, seguidamente entra en acción el algoritmo de Descenso de Gradiente (ec 3.10 y 3.11) el cual se encarga de actualizar todos los pesos y sesgos. (Goodfellow, 2016).

$$w_{i,j_{nuevo}}^{L} = w_{i,j_{actual}}^{L} - \eta \nabla \mathrm{MSE} \tag{3.10}$$

$$b_{j\ nuevo}^{L} = b_{j\ actual}^{L} - \eta \nabla \mathrm{MSE} \tag{3.11}$$

Donde η corresponde a la tasa de aprendizaje, parámetro que define la velocidad y estabilidad del aprendizaje, no existiendo una regla para asignar su magnitud. Cuando los valores de η son cercanos a 0.1 el aprendizaje es inestable y cuando los valores son muy pequeños (por ejemplo: 0.00001) el algoritmo de aprendizaje toma demasiado tiempo converger a una combinación óptima de parámetros, dejando la definición de η a criterio según ensayo y error (Goodfellow, 2016).

El descenso de gradiente es complementado con el algoritmo de Backpropagation, o retro-propagación del error, el cual se encarga del proceso iterativo de calcular el MSE nuevo y actualizar la red hasta converger a un MSE mínimo controlando el aprendizaje de la red.

La retropropagación del error posee este nombre debido a que es un proceso que inicia desde la capa de salida, en donde se calcula el valor de MSE de las predicciones, y luego el error y las correcciones efectuadas a los pesos se trasladan desde la capa de salida hasta la capa de entrada (Navarro, 2016).

En complemento al MSE se calcula el Coeficiente de Correlación (R) y Determinación (R^2), este coeficiente no influye en el proceso de aprendizaje de la red, sin embargo, es de gran importancia para evaluar su desempeño estadístico (ec. 3.12 y 3.13). Los valores que toman los coeficientes R y R^2 se encuentran acotados entre el intervalo [0,1], en donde valores cercanos a 0 reflejan una mala calidad de ajuste y los cercanos a 1 reflejan una buena calidad de ajuste de la RNA (Abellán, 2021).

$$R = \sqrt{1 - \frac{\sum_{i=1}^{N}(t_i - y_i)^2}{\sum_{i=1}^{N}(y_i)^2}} \tag{3.12}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(t_i - y_i)^2}{\sum_{i=1}^{N}(y_i)^2} \tag{3.13}$$

3.6. Entrenamiento de una red neuronal artificial

El entrenamiento de la red corresponde al proceso de actualizar los valores de pesos y sesgo hasta determinar una combinación de estos que minimice la función de error. En otras palabras, es el mecanismo por el cual se concreta el aprendizaje, controlando a su vez los fenómenos de sub-ajuste (underfitting) y sobre-aprendizaje (overfitting).

El bucle de entrenamiento es un ciclo iterativo que realiza N actualizaciones hasta lograr una combinación de parámetros satisfactoria. El término "época" se refiere a cuántas veces la red procesa todo el conjunto de datos. El algoritmo opera en mini-lotes, por ejemplo, si hay 600 muestras de entrenamiento y cada mini-lote contiene 60 muestras, se necesitan 10 iteraciones para completar una época (Goodfellow, 2016).

El sobre-aprendizaje ocurre cuando la RNA se ajusta demasiado a los datos de entrenamiento entregando valores de MSE en el rango $[0,10^{-4}[$, capturando no solo los patrones generales, sino también incluyendo el ruido y obteniendo resultados deficientes cuando la RNA se enfrenta a datos nuevos (Navarro, 2016). Por otra parte, el sub-ajuste ocurren cuando la RNA no logra aprender y ajustarse a los datos, obteniendo un modelo con un elevado valor de MSE e incapaz de generalizar las relaciones entre las entradas y salidas (ver Figura 3.5). Para evitar estos fenómenos se fija un MSE objetivo en el rango de valores $[10^{-4}, 10^{-1}[$, el cual nace de los resultados obtenidos por diversos investigadores (Navarro, 2016; Zarrigol, 2021; Vasquez, 2022).

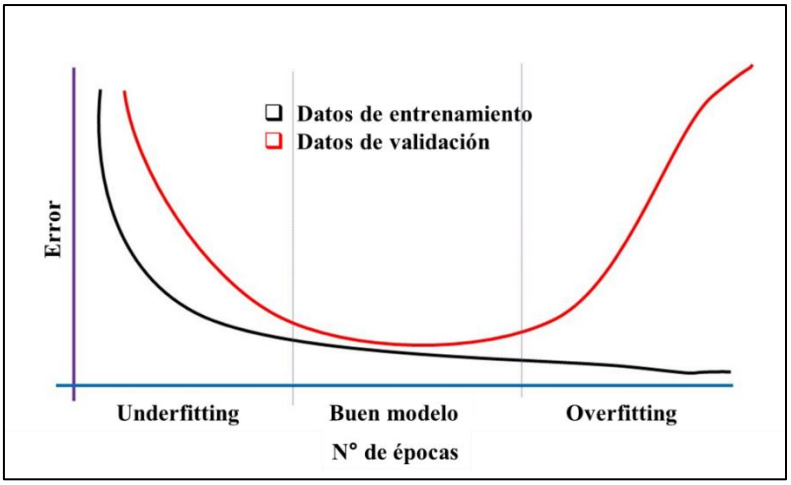

Figura 3.5 Evolución de entrenamiento de una red neuronal.

Con el propósito de reducir el sobre-aprendizaje y mejorar la calidad del ajuste, al iniciar el ciclo de entrenamiento se divide la base de datos en tres conjuntos: Entrenamiento (Train), Validación (Validation) y Prueba (Test).

La distribución de estos conjuntos no posee una regla en general, sin embargo, diversos investigadores optan por la distribución de 70% de los datos para entrenamiento, 15% de los datos para validación y 15% de los datos para test.

Después de entrenar la RNA es necesario evaluar su rendimiento con datos de validación no utilizados durante el entrenamiento, controlando el sobreajuste mediante la actualización de pesos y sesgos. Finalmente, los datos de prueba se utilizan al concluir el entrenamiento de la RNA para medir su precisión en datos no vistos (Navarro, 2016).

Para prevenir el sobre-aprendizaje, se implementa la estrategia de detención temprana (Early-stopping). Esta técnica se basa en analizar el comportamiento del MSE en los datos de validación, identificando la época en la cual el MSE comienza a aumentar, marcando así el punto de detención temprana. Aunque en las siguientes épocas el MSE de los datos de entrenamiento pueda disminuir, la precisión de la red tiende a saturarse, debido a que aumenta el error en el conjunto de datos de validación, como se evidencia en la Figura 3.5 (Goodfellow, 2016).

La construcción de una RNA se puede dividir en tres etapas: pre-proceso el cual se encuentra caracterizado con la definición de la arquitectura de red y normalización de datos de entrada y salida; Proceso, en donde se ejecuta el algoritmo de entrenamiento obteniendo resultados de desempeño MSE y R^2, y los valores de pesos y sesgos que conforman la red; y post-proceso donde se desnormalizan las salidas de la red para poder realizar predicciones a partir de nuevas entradas (ver Figura 3.6).

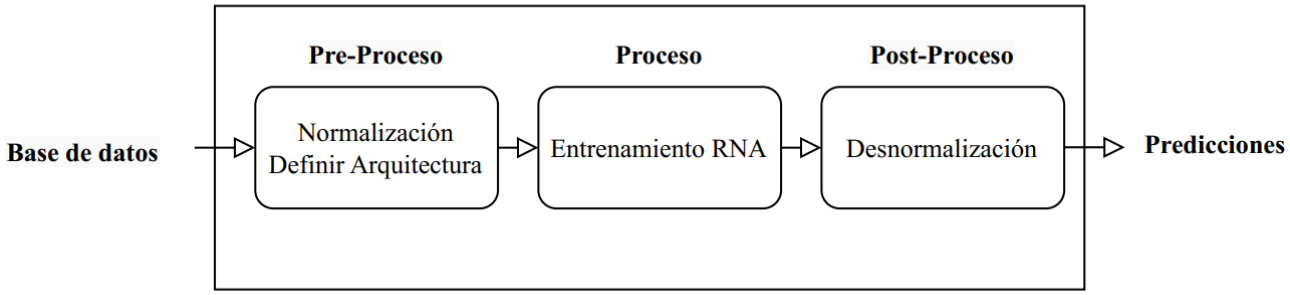

Figura 3.6 Esquema de proceso de entrenamiento RNA

En el presente trabajo las RNA serán construidas y entrenadas a través del Toolbox Neural Network Fitting de la versión R2020B de MATLAB® (ver Anexo 3.1). Dentro de este Toolbox se encuentra el algoritmo de entrenamiento Levenberg-Mardquardt (Demuth, 2002), el cual corresponde a una optimización del algoritmo de Descenso de Gradiente logrando un aumento de la velocidad de convergencia en la minimización de la función de costo (Navarro, 2016).

3.7. Conclusiones

En este capítulo se abordó el concepto y funcionamiento de las Redes Neuronales Artificiales (RNA), destacando su arquitectura multicapa y el proceso de aprendizaje basado en la minimización del Error Cuadrático Medio (MSE), junto con detallar la retropropagación del error como mecanismo clave para ajustar los pesos y sesgos de la red, finalizando con una formulación matemática general de la RNA.

En términos generales, este algoritmo de inteligencia artificial (IA) se encuentra conformado por un conjunto de matrices y vectores que poseen valores calibrados capaces de obtener predicciones a partir de una base de datos dada. Esta descripción revela la naturaleza algebraica y aritmética de la IA,

transformándola esencialmente en "cajas negras", esto se traduce en una interpretación poco intuitiva de cómo los valores de entrada se reflejan en los valores de salida.

Por otra parte, se destaca la importancia de la normalización de los datos en la etapa de pre-proceso para optimizar la etapa de entrenamiento de datos. Se introduce la técnica de "early-stopping" durante la etapa de entrenamiento para prevenir el sobreajuste de la red. Además, se menciona el algoritmo Levenberg-Mardquardt para optimizar el Descenso de Gradiente en el Toolbox Neural Network Fitting de MATLAB® para implementar computacionalmente el algoritmo de redes neuronales artificiales.

CAPÍTULO 4: DESARROLLO DE UNA BASE DE DATOS TEÓRICA PARA EL ENTRENAMIENTO DE UNA RED NEURONAL DE DISEÑO ÓPTIMO A FLEXIÓN DE VIGAS.

4.1. Introducción

Con el propósito de estudiar las dimensiones transversales y cuantías de armaduras longitudinales de vigas de hormigón armado, se desarrolla una base de datos de 1302 vigas que considera como función objetivo la optimización del costo del elemento. Para ello se consideran como variables de entrada las condiciones de apoyo, longitud y magnitud de carga uniformemente distribuida sobre el elemento. Esta formulación corresponde a una adaptación a la presentada por Chakrabarty (1992) para vigas simplemente apoyadas, extendiéndola en este caso a vigas bi-empotradas y cantiléver.

4.2. Datos preliminares

Previo al diseño de las armaduras a flexión es relevante establecer los criterios de diseño utilizados y enlistar los valores de variables que se mantendrán constantes. Adicionalmente se debe indicar que la RNA se desarrolla solo para la determinación de las dimensiones geométricas de la sección rectangular (B y H) y las cuantías de armaduras longitudinales A_s y $A_{S'}$. Las armaduras de corte (estribos) quedan fuera de este análisis de optimización, en el caso de querer estimarlas se deben seguir lo procedimientos estándar para este tipo de armaduras una vez obtenidas las dimensiones geométricas y armaduras longitudinales.

- Materiales y Costos

Dado que el diseño se enfoca en optimizar el costo de las vigas, la selección de materiales de construcción se restringe a opciones comerciales. Los costos de estos materiales se extrajeron del Manual de Costos ONDAC (2017) y fueron actualizados a octubre del año 2023 utilizando la calculadora del Índice de Precios al Consumidor (IPC) del Instituto Nacional de Estadísticas.

Tabla 4.1 Listado de precios unitarios de materiales de construcción actualizado al año 2023

Material	Costo (CLP)	Unidad
Hormigón G20	$ 142,043	m3
Hormigón G25	$ 146,395	m3
Hormigón G30	$ 157,381	m3
Acero A630-420H	$ 1,612	kg
Moldaje Vigas	$ 15,617	m2

- Propiedades mecánicas de los materiales

Tabla 4.2. Magnitudes de los parámetros fijos usados

Nombre	Descripción	Magnitud	Unidad
f'_c	Resistencia específica a la compresión del hormigón	20-25-30	MPa
f_y	Resistencia a la fluencia del acero	420	MPa
w_c	Peso específico del hormigón	2.325	T/m3
w_s	Peso específico del acero	7.85	T/m3
E_s	Módulo de elasticidad del acero	210000	MPa
e_{cu}	Deformación última del hormigón	0.003	-
e_y	Límite de deformación de fluencia del acero	0.002	-
rec	Recubrimiento libre	4	cm

4.3. Base de datos analítica

La metodología empleada en la elaboración de las bases de datos considera tres condiciones de apoyo: simplemente apoyada, bi-empotrada y en voladizo (ver Figura 4.1), y tres grados diferentes de hormigón G20, G25 y G30 según la clasificación de la NCh430:2008 (INN, 2008), generándose un total de nueve bases de datos.

En el análisis de cada base de datos, se destacan como variables predominantes el largo de la viga (L), la carga muerta distribuida (DL) y la carga viva distribuida (LL). Para cada condición de apoyo se establece el intervalo de valores de longitud y carga que estas variables pueden abarcar. La elección de estos se basa en los órdenes de magnitud que razonablemente se encuentran en la práctica del diseño (ver Tabla 4.3).

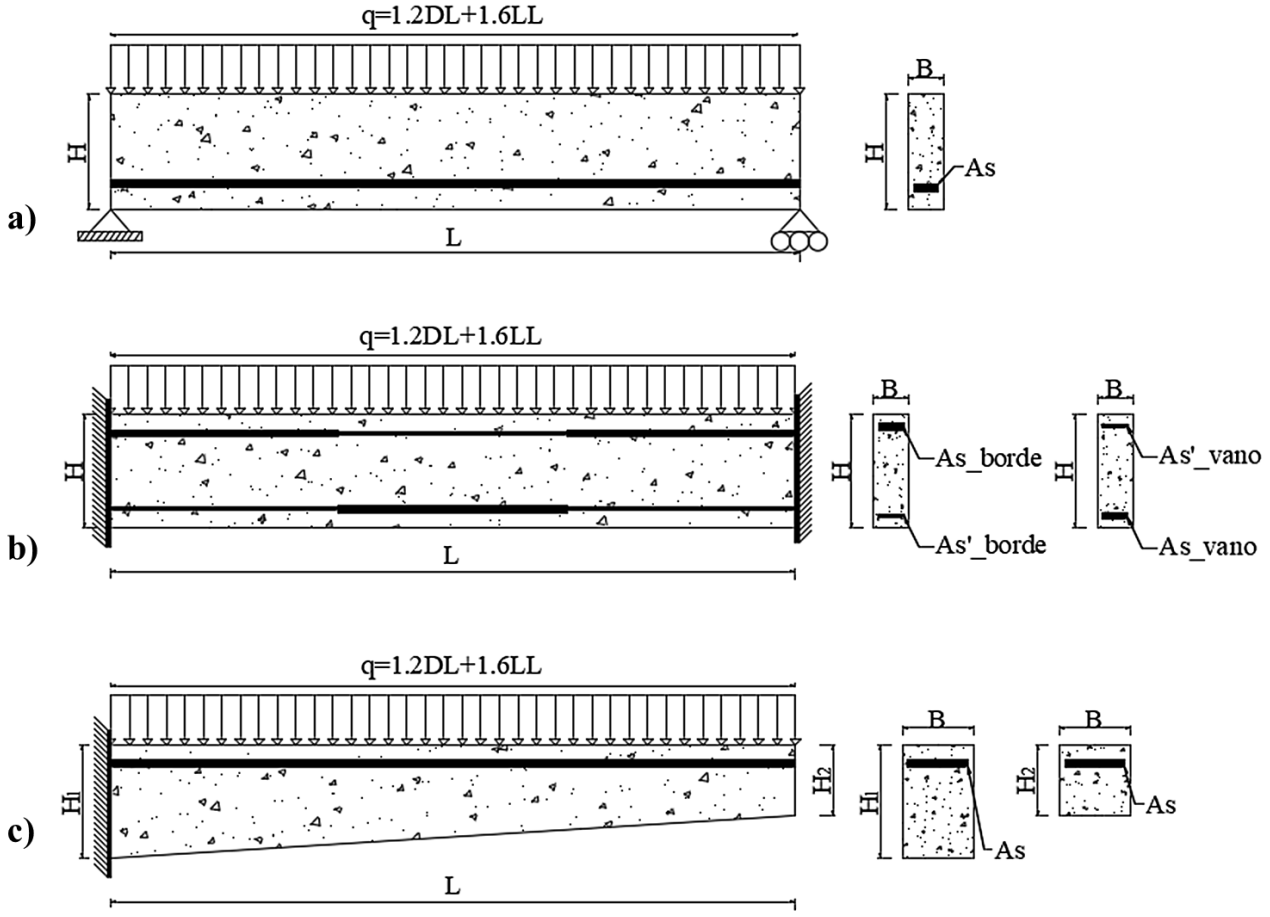

Figura 4.1 Condición de apoyo a) Simplemente apoyado; b) Bi-empotrado y c) Voladizo

Tabla 4.3 Rango de valores de entrada para vigas

Condición apoyo	Largo de la viga (m)	Carga Muerta (kN/m)	Carga Viva (kN/m)
Simplemente apoyado	$3 \leq L \leq 7$	$10 \leq DL \leq 60$	$10 \leq LL \leq 60$
Bi-empotrado	$3 \leq L \leq 7$	$10 \leq DL \leq 60$	$10 \leq LL \leq 60$
Voladizo	$1 \leq L \leq 3$	$10 \leq DL \leq 60$	$10 \leq LL \leq 60$

La base de datos considera un aumento incremental del largo de viga unitario (1 metro) exceptuando el caso “Voladizo” el cual considera un incremento de la longitud de viga de 0.5 m en cada iteración.

Las cargas se incrementan en 5 kN/m simultaneo y alternado para DL y LL, de tal manera que estas tomen el mismo valor o difieran en 5 kN (ver Figuras 4.2, 4.3 y 4.4). Las solicitaciones de momento

flector último (Mu) varían dependiendo la magnitud de la carga de diseño (ec 4.1), largo de la viga y la condición de apoyos (ec. 4.2). Se considera la combinación 1.2DL+1.6LL para obtener la carga uniformemente distribuida (q), en donde DL considera la contribución del peso propio de la viga y la carga muerta externa uniformemente distribuida.

$$q = 1.2DL + 1.6LL \tag{4.1}$$

$$Mu = q \cdot L^2 \cdot k \tag{4.2}$$

Donde: q = carga uniformemente distribuida de diseño

L = largo de la viga

k = Coeficiente de momento máximo o mínimo.

Tabla 4.4 Coeficiente de momento máximo y mínimo en vigas

Condición Apoyo	Simplemente apoyado	Bi-empotrado	Voladizo
Momento máximo	1/8	1/12	-
Momento mínimo	-	1/24	1/2

4.3.1 Viga Simplemente apoyada

La formulación presentada por Chakrabarty (1992) corresponde a un modelo de optimización del costo de una viga de H-A rectangular a flexión. La función objetivo a minimizar (ec. 4.3) depende de las dimensiones (ancho, alto y cuantía de acero longitudinal) y de los datos preliminares expuestos en el subcapítulo 4.2, respetando la compatibilidad de unidades de medida para obtener un valor de costo en pesos chilenos (CLP) por unidad de longitud.

$$y = C_1X_1 + C_2X_2X_3 + C_3X_2 + C_4X_3 \tag{4.3}$$

y = Costo de la viga por unidad de longitud (CLP/cm)

C_1 = Costo asociado al volumen del acero (CLP/cm^3)

C_2 = Costo asociado al volumen de hormigón (CLP/cm^3)

C_3 = Costo asociado al valor de moldaje vertical por área (CLP/cm^2)

C_4 = Costo asociado al valor de moldaje horizontal por área (CLP/cm^2)

X_1 = Variable asociada al área de acero longitudinal en tensión (A_s) de la viga (cm^2)

X_2 = Variable asociada a la altura (H) de la viga (cm)

X_3 = Variable asociada al ancho (B) de la viga (cm)

Los valores de las constantes C_1, C_2, C_3 y C_4 se obtienen a partir los datos de las tablas 4.1 y 4.2, los valores de C_3 y C_4 difieren pues para el moldaje vertical se consideran ambos costados y para el moldaje horizontal se considera solo la cara inferior de la viga.

Este modelo está sujeto a las restricciones detalladas en la Tabla 4.5 las cuales determinan las dimensiones de la viga y garantizan el cumplimiento del código ACI318-19. Las restricciones R1, R2, R3 y R4 presentan modificaciones a las propuestas por Chakrabarty en 1992 para cumplir con las exigencias del código actual. Sumado a esto, las restricciones R5, R6 y R7 fueron añadidas con la misma finalidad, complementando el modelo de optimización.

Tabla 4.5. Restricciones que gobiernan el caso viga simplemente apoyada

ID	Restricción	Expresión	Ref. ACI318-19
R1	Condición de equilibrio	$T = C$	Cap. 22.2
R2	Demanda/Capacidad	$\frac{M_u}{\phi M_n} = 0.9$	9.5.1.1 (a)
R3	Capacidad a flexión de diseño	$\phi A_s \cdot f_y \cdot \left(d - \frac{c \cdot \beta_1}{2}\right) = 0.9 \cdot \phi M_n$	Cap. 22.2
R4	Razón de aspecto (Ancho/Altura útil)	$\frac{b}{d} = \frac{1}{3}$	
R5	Altura mínima	$h_{min} \leq h_{viga}$	Tabla 9.3.1.1
R6	Falla controlada por tensión	$\varepsilon_s \geq 0.005$	Tabla 21.2
R7	Área de acero longitudinal mínima	$A_{s_{min}} \leq A_{s_{viga}}$	9.6.1.2 (a), (b)

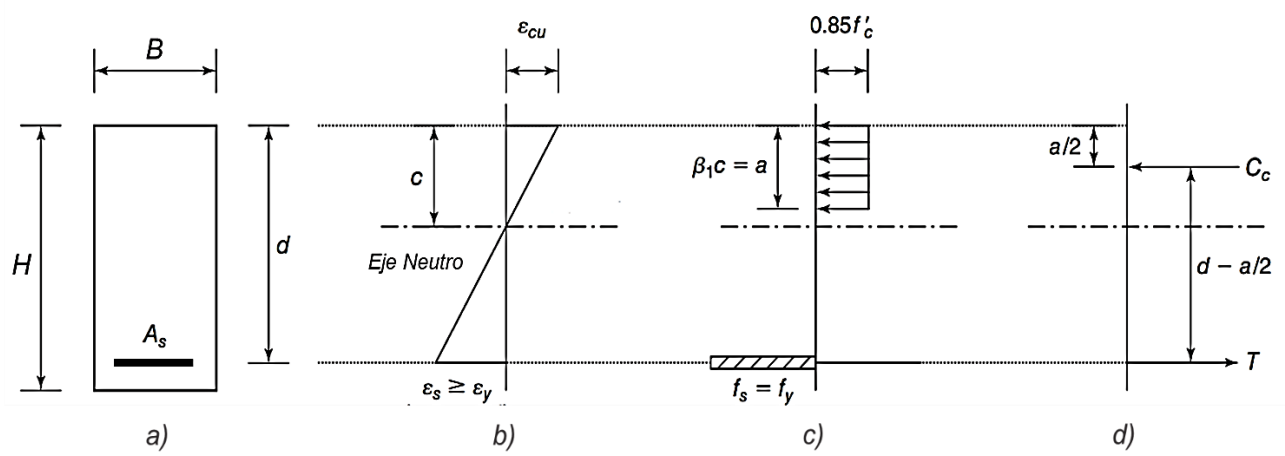

Figura 4.2 Sección simplemente reforzada cuando la capacidad a flexión es alcanzada a) Sección simplemente reforzada; b) Diagrama de deformaciones; c) Distribución de tensiones; d) Fuerzas Internas. (Wight, 2016. Modificado)

Se impone que para el diseño de todas las vigas se satisfaga un factor de utilización, definido como la razón $\frac{M_u}{\phi M_n}$, del 90% en la restricción R2. Esto se incluye para mantener holgura en el diseño a flexión de las vigas sin comprometer la optimización de estas. La restricción R4 impone una razón de aspecto de la viga de tal manera que los diseños posean homogeneidad en sus dimensiones para cada largo, estado de carga, condición de apoyo y material base. La restricción R6 requiere que la deformación unitaria del refuerzo a tracción mayor o igual al 0.005 que garantice un comportamiento dúctil del elemento (falla controlada por tracción). Esta restricción impone de manera implícita un valor máximo del área de acero longitudinal. Adicionalmente, si R6 se satisface, el factor de reducción de resistencia es ϕ=0.90, constante para todas las vigas de este estudio. Por último, las demás restricciones otorgan el cumplimiento normativo en el diseño a flexión de cada viga.

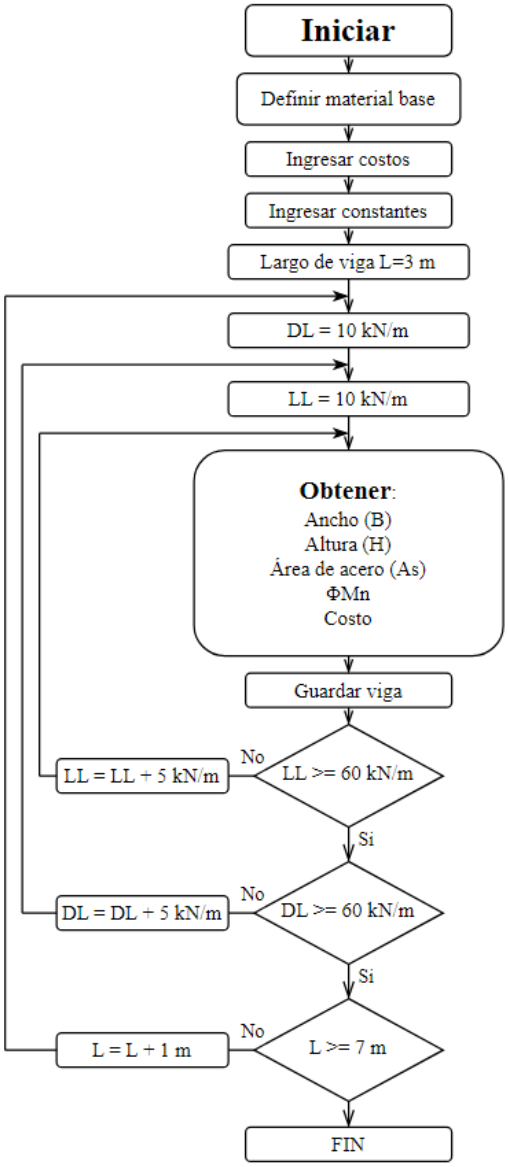

Figura 4.3 Diagrama de flujo de la base de datos para vigas simplemente apoyadas

Se desarrolla una planilla de cálculo en MS Excel utilizando el algoritmo de solver GRG Nonlinear (Microsoft Corporation, 2023) para construir la base de datos y dar cumplimiento a todas las restricciones impuestas. Se diseñan 155 vigas rectangulares de hormigón armado a flexión optimizadas para cada material base (G20, G25 y G30). Obteniéndose, un total de 465 vigas a flexión simplemente apoyadas.

Tabla 4.6 Rango de valores para vigas simplemente apoyadas

Variable	Parámetros	G20	G25	G30
Variables de entrada	L (m)	3 - 7	3 - 7	3 - 7
	q (kN/m)	28 - 168	28 - 168	28 - 168
Variables de salida	A_s (cm^2)	3.8 - 39.5	4.1 - 42.4	4.3 - 44.6
	B (cm)	9.9 - 31.9	9.2 - 29.6	8.7 - 28.0
	H (cm)	33.8 - 99.8	31.6 - 92.8	30.1 - 87.9
	ϕMn (kN-m)	36 - 1203	36 - 1194	36 - 1189
	Costo (CLP/cm)	217 - 1315	208 - 1274	204 - 1270

4.3.2 Viga Bi-Empotrada

Extendiendo la formulación de Chakrabarty (1992) se modifica la función objetivo agregando una variable asociada al área de acero longitudinal en compresión X_4 en el empotramiento (ec. 4.3).

$$y = C_1X_1 + C_2X_2X_3 + C_3X_2 + C_4X_3 + C_1X_4 \quad (4.3)$$

y = Costo de la viga por unidad de longitud (CLP/cm)

C_1 = Costo asociado al volumen del acero (CLP/cm^3)

C_2 = Costo asociado al volumen de hormigón (CLP/cm^3)

C_3 = Costo asociado al valor de moldaje vertical por área (CLP/cm^2)

C_4 = Costo asociado al valor de moldaje horizontal por área (CLP/cm^2)

X_1 = Variable asociada al área de acero longitudinal en tensión (A_s) de la viga (cm^2)

X_2 = Variable asociada a la altura (H) de la viga (cm)

X_3 = Variable asociada al ancho (B) de la viga (cm)

X_4 = Variable asociada al área de acero longitudinal en compresión (A'_s) de la viga (cm^2)

Tabla 4.7 Restricciones modificadas para vigas Bi-empotradas

ID	Restricción	Expresión	Ref. ACI318-19
R1	Condición de equilibrio	$T = C_c + C_s$	Cap. 22.2
R2	Demanda/Capacidad	$\frac{M_u}{\phi M_n} = 0.9$	9.5.1.1 (a)

R3	Capacidad a flexión de diseño	$\phi\left(C_c\left(d-\frac{a}{2}\right)+C_s(d-d')\right)=0.9\cdot\phi M_n$	Cap. 22.2
R4	Razón de aspecto (Ancho/Altura útil)	$\frac{b}{d}=\frac{1}{3}$	
R5	Altura mínima	$h_{min}\leq h_{viga}$	Tabla 9.3.1.1
R6	Falla controlada por tensión	$\varepsilon_s\geq 0.005$	Tabla 21.2
R7	Área de acero longitudinal mínima	$A_{s_{min}}\leq A_{s_{viga}}$	9.6.1.2 (a), (b)

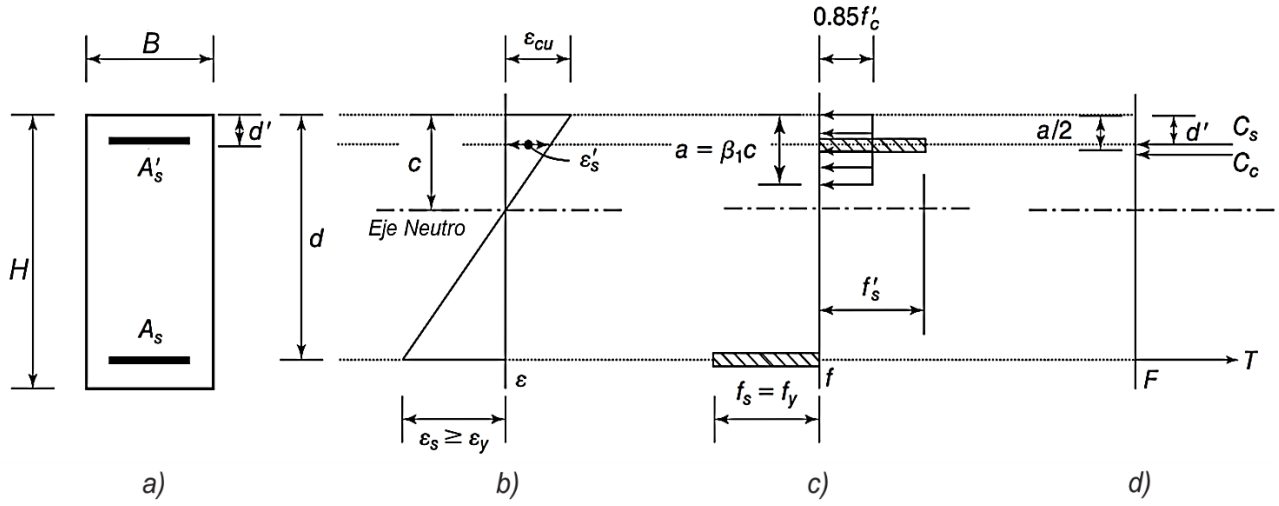

Figura 4.4 Sección doblemente reforzada cuando la capacidad a flexión es alcanzada a) Sección doblemente reforzada; b) Diagrama de deformaciones; c) Distribución de tensiones; d) Fuerzas Internas. (Wight, 2016. Modificado)

La metodología seguida para la construcción de la base de datos se basa en determinar las dimensiones de la sección transversal de la viga (ancho y alto) y las cuantías de armaduras en el empotramiento. En la tabla 4.7 presenta las modificaciones a las restricciones R1 y R3 expuestas en la tabla 4.5, pues se incorpora el efecto del acero en compresión en el empotramiento.

Se determinan los valores de ancho (B), alto (H), área de acero longitudinal en tensión (As_{borde} (X_1)), área de acero longitudinal en compresión ($A's_{borde}$ (X_4)) y capacidad a flexión de diseño (ϕMn_{borde})

considerando una razón $\frac{M_u}{\phi M_n}$ del 90% cumpliendo con el modelo de optimización expuesto y las restricciones de las tablas 4.5 y 4.7.

Posterior a esto, se ingresan los valores de ancho (B) y alto (H) para estimar la armadura en tensión (As_{vano}), compresión (As'_{vano}) y la capacidad a flexión de diseño (ϕMn_{vano}) considerando una razón $\frac{M_u}{\phi M_n}$ del 80% con el propósito de compatibilizar las magnitudes de área de acero en comparación a las obtenidas en el empotramiento, dado que las solicitaciones disminuyen a la mitad.

Debido que el área de acero en el vano es menor al área de acero en el en el empotramiento, el costo (CLP/cm) es calculado de manera conservadora como el costo obtenido en el tramo empotrado.

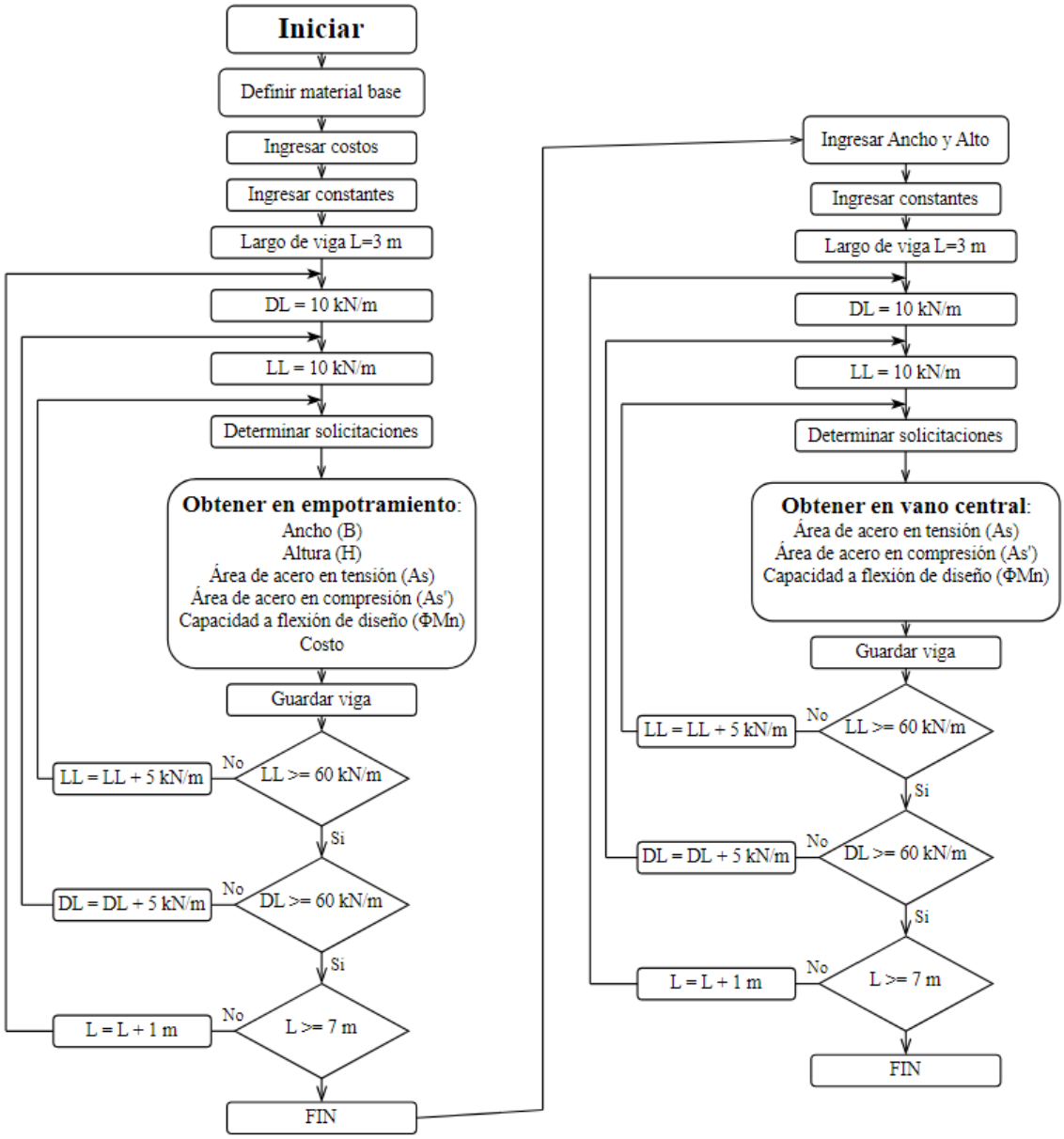

Figura 4.5 Diagrama de flujo de la base de datos para vigas bi-empotradas

Obteniéndose 155 vigas bi-empotradas para cada material base, sin embargo, debido a que el modelo Solver de MS Excel no alcanza una convergencia adecuada en algunos casos de carga se procede a depurar las vigas que incumplen las restricciones impuestas. Quedando un total de 135 vigas para el material G20, 134 para G25 y 134 para G30. Conformando una base de datos de 403 vigas rectangulares de H-A bi-empotradas.

Tabla 4.8 Rango de valores para vigas Bi-empotradas

Variable	Parámetro	G20	G25	G30
Variables de entrada	L (m)	3 - 7	3 - 7	3 - 7
	q (kN/m)	28 - 168	28 - 168	28 - 168
Variables de salida	B (cm)	8.1 - 25.6	7.7 - 25.1	7.3 - 25.6
	H (cm)	28.4 - 80.7	27.0 - 79.4	25.9 - 80.7
	As_{borde} (cm^2)	3.1 - 31.5	3.3 - 33.5	3.4 - 34.2
	As'_{borde} (cm^2)	0.7 - 6.5	0.6 - 6.3	0.5 - 6.5
	ϕMn_{borde} (kN-m)	24 - 788	24 - 785	24 - 788
	As_{vano} (cm^2)	1.6 - 16.3	1.8 - 19.0	1.9 - 22.9
	As'_{vano} (cm^2)	0.7 - 6.5	0.6 - 6.3	0.5 - 6.5
	ϕMn_{vano} (kN-m)	14 - 445	14 - 507	14 - 619
	Costo (CLP/cm)	327 - 2251	311 - 2217	306 - 2388

4.3.3 Viga Cantilever

De manera análoga al caso de vigas bi-empotradas se modifica la función objetivo a minimizar dado que las vigas en voladizo solo poseen solicitaciones de momento negativo.

Las dimensiones de la viga se diseñan a partir de las solicitaciones del empotramiento, determinando los valores del ancho (B), altura inicial (H_1), área de acero longitudinal en tensión (As) (ver Figura 4.1 (c)), capacidad a flexión de diseño (ϕMn) y el costo asociado a las dimensiones del empotramiento a partir de la función objetivo de la ecuación (4.4).

$$y = C_1X_1 + C_2X_2X_3 + C_3X_2 + C_4X_3 \quad (4.4)$$

y = Costo de la viga por unidad de longitud (CLP/cm)

C_1 = Costo asociado al volumen del acero (CLP/cm^3)

C_2 = Costo asociado al volumen de hormigón (CLP/cm^3)

C_3 = Costo asociado al valor de moldaje vertical por área (CLP/cm^2)

C_4 = Costo asociado al valor de moldaje horizontal por área (CLP/cm^2)

X_1 = Variable asociada al área de acero longitudinal en tensión (A_s) de la viga (cm^2)

X_2 = Variable asociada a la altura (H) de la viga (cm)

X_3 = Variable asociada al ancho (B) de la viga (cm)

Tabla 4.9 Restricciones modificadas para vigas Cantilever

ID	Restricción	Expresión	Ref. ACI318-19
R1	Condición de equilibrio	$T = C$	Cap. 22.2
R2	Demanda/Capacidad	$\frac{M_u}{\phi M_n} = 0.8$	9.5.1.1 (a)
R3	Capacidad a flexión de diseño	$\phi A_s \cdot f_y \cdot \left(d - \frac{c \cdot \beta_1}{2}\right) = 0.8 \cdot \phi M_n$	Cap. 22.2
R4	Razón de aspecto (Ancho/Altura útil)	$\frac{b}{d} = \frac{2}{3}$	
R5	Altura mínima	$h_{min} \leq h_{viga}$	Tabla 9.3.1.1
R6	Falla controlada por tensión	$\varepsilon_s \geq 0.005$	Tabla 21.2
R7	Área de acero longitudinal mínima	$A_{s_{min}} \leq A_{s_{viga}}$	9.6.1.2 (a), (b)

La tabla 4.9 presenta las restricciones de diseño a flexión impuestas para la construcción de la base de datos, en este caso las vigas son diseñadas para una razón $\frac{M_u}{\phi M_n}$ del 80% en el empotramiento y una razón de aspecto de 2/3, esta holgura se debe a que la resistencia de diseño se ve reducida a medida que aumenta el largo de viga pues su altura disminuye hasta igualar al ancho.

Se impone que la altura final de la viga cantilever (H_2) (ver Figura 4.1 (c)) sea igual al ancho de la viga, de tal manera que la sección transversal del extremo libre de la viga sea cuadrada. Posterior a esto, se verifica que la capacidad a flexión de diseño de la sección ubicada a la mitad de la viga sea mayor a la solicitación. Finalmente se estima el costo/largo considerando la sección transversal variable (ec. 4.5), pues el volumen de hormigón y el área de moldaje vertical disminuye.

$$y = C_1 X_1 + C_2 \left(\frac{H_1 + H_2}{2}\right) X_3 + C_3 \left(\frac{H_1 + H_2}{2}\right) + C_4 X_3 \quad (4.5)$$

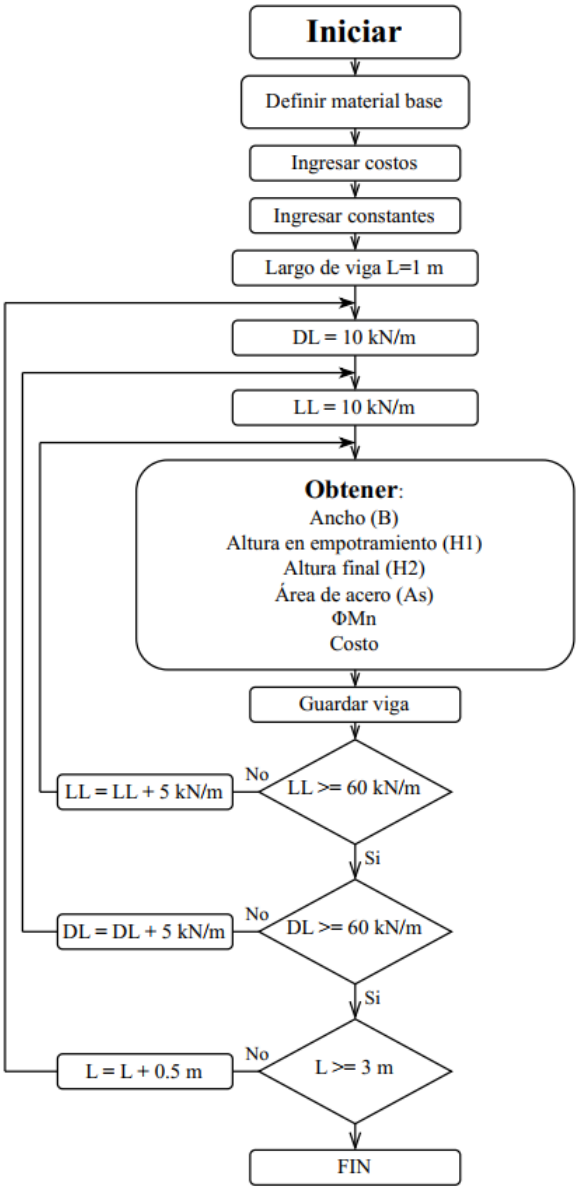

Figura 4.6 Diagrama de flujo de la base de datos para vigas Cantilever

En este caso se obtienen 155 vigas para el material G20, 155 para G25 y 124 para G30 obteniendo un total de 434 vigas cantilever de H-A con sección variable.

Tabla 4.10 Rango de valores para vigas cantilver.

Variable	Parámetro	G20	G25	G30
Variables de entrada	L (m)	1 - 3	1 - 3	1 - 3
	q (kN/m)	28 - 168	28 - 168	28 - 168
Variables de salida	B (cm)	12.5 - 47.6	11.6 - 44.0	10.9 - 43.5
	H_1 (cm)	22.7 - 75.3	21.4 - 70.0	21.4 - 70.0
	H_2 (cm)	12.5 - 47.6	11.6 - 44.0	10.9 - 43.5
	As (cm^2)	3.0 - 43.8	3.2 - 46.9	3.4 - 48.4
	Costo (CLP/cm)	295 - 3273	273 - 3010	265 - 2935
	ϕMn (kN-m)	18 - 992	18 - 985	18 - 980

4.4. Conclusiones

En este capítulo se describió la generación de las bases de datos para el posterior entrenamiento de las redes neuronales, siendo los conjuntos de datos de entrada y salida la columna vertebral en el proceso de confección de modelos predictivos de regresión.

La base de datos generada contiene un total de 1302 vigas diseñadas a flexión, agrupadas según su condición de apoyo. Los diseños obtenidos se ajustan a lo esperado para este tipo de elementos, en el sentido que en las secciones en donde se presentan las máximas solicitaciones se obtienen las mayores dimensiones y cuantías de armaduras de acero. Cabe señalar que los valores de ancho, alto y área de acero longitudinal se encuentran con más decimales que los ingenierilmente aceptados en la práctica profesional. Esto con el objetivo de proporcionar una base de datos que requiera la mínima intervención posible para no condicionar la etapa de entrenamiento de las RNA.

CAPÍTULO 5: APLICACIÓN DE RNA PARA EL DISEÑO A FLEXIÓN DE VIGAS.

5.1. Introducción

Con el objetivo de construir un modelo de Machine Learning basado en el algoritmo de Redes Neuronales Artificiales capaz de aprender de los resultados obtenidos en el capítulo anterior, se procede a entrenar diversas RNA para determinar las arquitecturas que poseen mejor desempeño en la estimación de las dimensiones y armaduras longitudinales optimas en vigas de hormigón armado.

Se verifica la robustez de los modelos de RNA obtenidos a través de un análisis de sensibilidad de las variables de salida en función de las variables de entrada para obtener conclusiones acerca de la confiabilidad del modelo. Para facilitar la estimación de las dimensiones óptimas se construye una interfase grafica de IA en entornos MATLAB® que aplica RNA entrenadas para realizar diseños a flexión de vigas de H-A a partir de datos iniciales ingresados por un usuario.

5.2. Metodología de entrenamiento de Redes Neuronales Artificiales

El procedimiento para abordar cada caso de estudio inicia con la definición de las variables que generalizan el diseño a flexión, identificando cuales de estas corresponden a variables de entrada y salida. Posterior a esto hay un proceso de normalización de los datos en el intervalo [-1,1] para compatibilizar el orden de magnitud de las variables de entrada y salida.

La arquitectura de cada red se define utilizando el Toolbox de Deep Learning de MATLAB®, en su versión R2020a. Para las neuronas de la capa oculta, se ha optado por la función de activación *Tansig*, mientras que las neuronas de la capa de salida utilizan la función *Purelin*.

Se establece un mínimo de una neurona en la capa oculta, incrementando linealmente el número de estas hasta determinar la configuración que demuestre el mejor desempeño. Esta práctica sigue una metodología común entre diversos investigadores. Por ejemplo, se ha alcanzado un número máximo

de 60 neuronas en la capa oculta (Navarro, 2016), 55 neuronas en la capa oculta (Zarringol, 2021), y 30 neuronas en la capa oculta (Abellán, 2021), entre otros estudios.

Siguiendo el enfoque adoptado en otras investigaciones (Aguilar, 2013) (Chopra, 2018), se ha optado por una partición de datos que asigna el 70% para el entrenamiento, el 15% para la validación y 15% para el conjunto de prueba.

Después de completar el proceso de entrenamiento de las redes neuronales, se procede a identificar aquellas que exhiben los valores más bajos en los coeficientes de Error Cuadrático Medio (MSE) dentro del intervalo objetivo [10^{-4}, 10^{-1}[y Coeficiente de Determinación (R) cercanos a 1, con el fin de evaluar la calidad de cada modelo.

5.3. CASO I: Viga simplemente apoyada

5.3.1 Arquitectura de la RNA

Las variables de salida que generalizan el diseño a flexión corresponden al área de acero longitudinal en tensión (As) ancho (B) y alto (H) de viga, capacidad a flexión de diseño (ϕMn) y Costo por metro lineal (Costo), es decir, se identifican dos variables de entrada y cinco variables de salida (ver Figura 5.1).

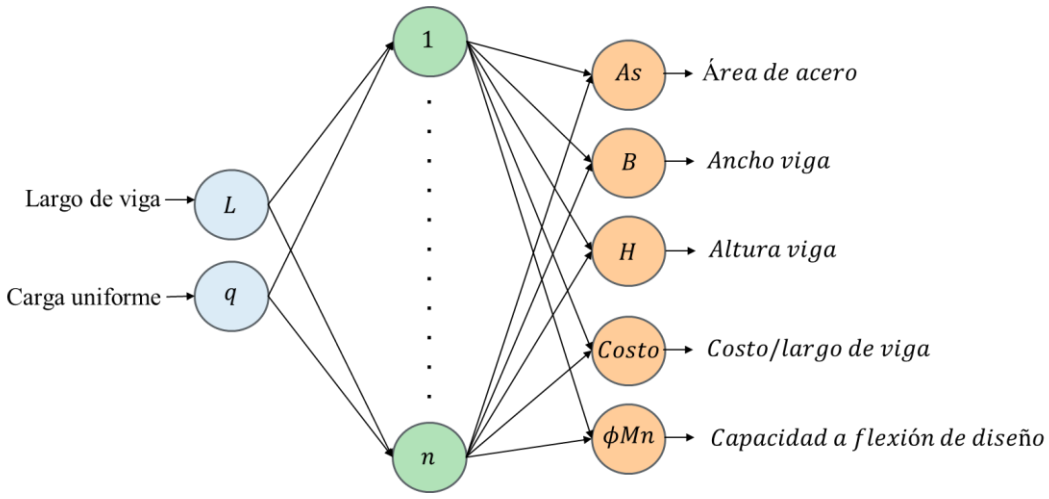

Figura 5.1 Arquitectura de la RNA caso simplemente apoyada

Se desarrollan siete RNA iniciando con una neurona en la capa oculta hasta alcanzar un total de siete. Estas redes de identifican mediante la nomenclatura:

SIMP_fcA_B

SIMP = Red para vigas simplemente apoyadas.

A = Material base hormigón G20, G25 o G30.

B = Etiqueta de la red (1, 2, 3, …, 7).

La base de datos para cada material base posee 155 muestras con cinco salidas. En otras palabras, la base de datos alberga al menos 755 valores de salida en total. El conjunto de datos de entrenamiento consta de 543 salidas (app. el 70% de 755), mientras que los conjuntos de validación y prueba contienen 116 salidas (app. el 15% de 755) cada uno.

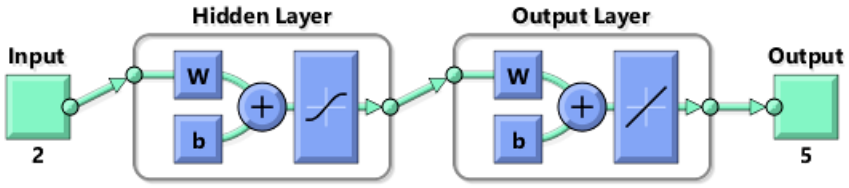

Figura 5.2 Esquema de red neuronal artificial para el diseño a flexión de vigas simplemente apoyadas

5.3.2 Entrenamiento de la RNA

Los resultados de entrenamiento se obtienen al calcular el factor R para el conjunto de entrenamiento, prueba y el conjunto completo de datos, junto con el MSE de la totalidad de las salidas (ver Tablas 5.1, 5.2 y 5.3).

Tabla 5.1 Resultados de entrenamiento de las RNA SIMP_fc20.

N° neuronas en la capa oculta	R			MSE
	Entrenamiento	Test	Total	
1	0.989	0.986	0.988	5.95E-03
2	0.993	0.993	0.992	3.86E-03
3	**0.999**	**0.999**	**0.999**	**3.44E-04**
4	0.999	0.999	0.999	4.03E-05
5	1	1	1	5.20E-06
6	1	1	1	1.73E-06
7	1	1	1	4.29E-07

Tabla 5.2 Resultados de entrenamiento de las RNA SIMP_fc25.

N° neuronas en la capa oculta	R			MSE
	Entrenamiento	Test	Total	
1	0.989	0.993	0.988	5.95E-03
2	0.996	0.995	0.995	2.08E-03
3	**0.999**	**0.999**	**0.999**	**2.13E-04**
4	0.999	0.999	0.999	3.00E-05
5	0.999	0.999	0.999	7.01E-06
6	1	1	1	2.39E-06
7	1	1	1	1.04E-06

Tabla 5.3 Resultados de entrenamiento de las RNA SIMP_fc30.

N° neuronas en la capa oculta	R			MSE
	Entrenamiento	Test	Total	
1	0.988	0.991	0.988	5.95E-03
2	0.996	0.995	0.995	3.89E-03
3	**0.999**	**0.999**	**0.999**	**2.11E-04**
4	0.999	0.999	0.999	3.73E-05
5	0.999	0.999	0.999	9.25E-06
6	1	1	1	1.62E-06
7	1	1	1	1.22E-06

Para determinar la arquitectura que mejor se ajusta a la base de datos entregada se identifica el valor del coeficiente de correlación R sea más cercano a 1 (Zarringol, 2021), en este caso, todas las arquitecturas poseen valores aceptables mayores a 0.9.

Con respecto al MSE observamos que, a partir de una neurona en la capa oculta, el error de predicción alcanza niveles aceptables, alcanzando el error objetivo con tres neuronas. Por otra parte, en las redes con seis y siete neuronas en la capa oculta, se evidencia que el MSE converge hacia un valor asintótico

menor, en donde esto sugiere que las redes más complejas reflejan indicios de sobre-aprendizaje. Luego las redes SIMP_fc20_3, SIMP_fc25_3 y SIMP_fc30_3 son las que logran generalizar el caso de estudio con mayor calidad.

Adicionalmente, se presentan los resultados de regresión de estas redes en la Figuras 5.3 y Anexo 5.1.1, graficados para tres conjuntos de particiones y para el conjunto total de datos. Los rangos de valores que toman los ejes son [-1,1] debido a la normalización elegida. En el eje horizontal se muestra el valor objetivo (Target), un valor fijo proveniente de la base de datos, mientras que en el eje vertical se representa el valor de salida (Output) calculado por la red. Al comparar estos valores, se observa que la red establece correctamente la relación entre el valor esperado y el valor calculado para cada conjunto de datos. Esto respalda la afirmación de que estos modelos de RNA entrenados no sufren de sobreajuste (overfitting), ya que la calidad de la predicción es coherente en cada conjunto de datos.

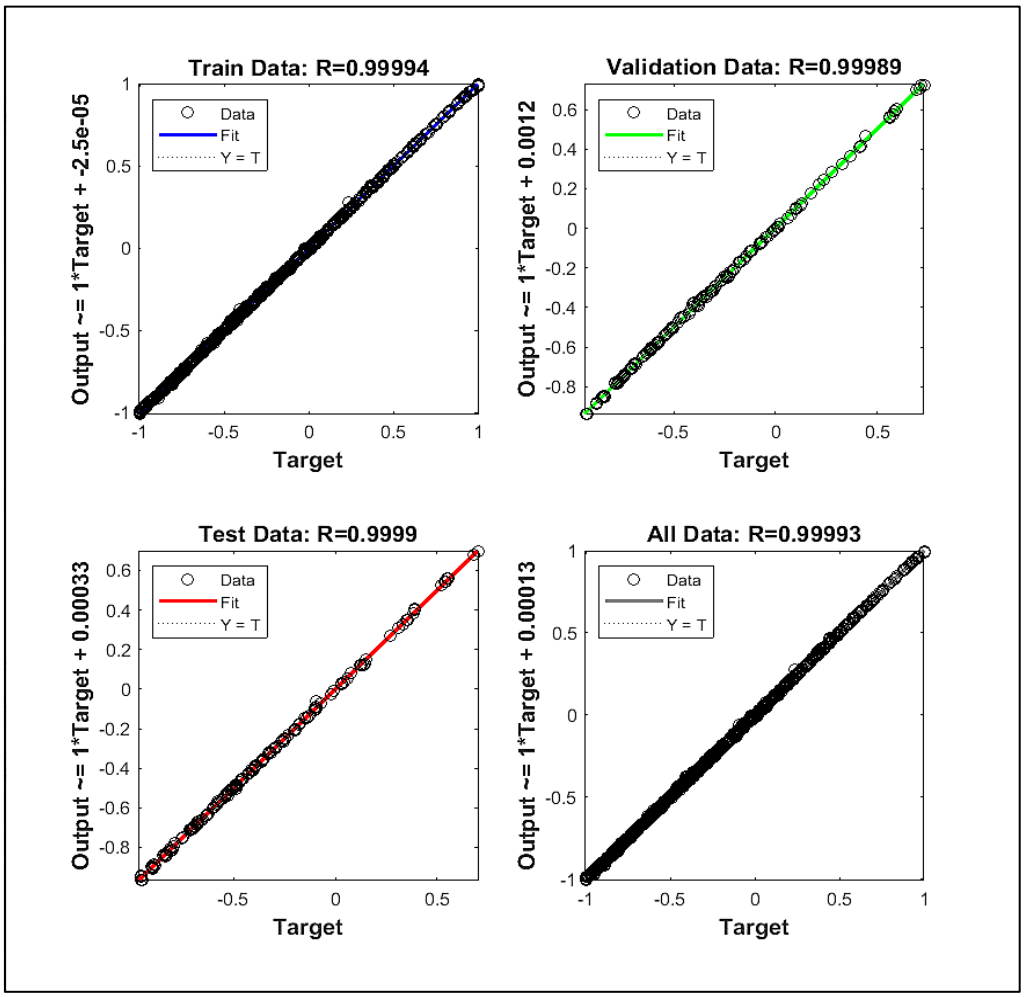

Figura 5.3 Regresión red SIMP_fc20_3

5.4. CASO II: Viga Bi-Empotrada

5.4.1 Arquitectura de la RNA

Las variables de salida que generalizan el diseño corresponden al Ancho (B) y Alto (H) de viga, Área de acero longitudinal en tensión en el borde (As_{borde}), Área de acero longitudinal en compresión en el borde (As'_{borde}), capacidad a flexión de diseño en el borde (ϕMn_{borde}), Área de acero longitudinal en tensión en el vano (As_{vano}), Área de acero longitudinal en compresión en el vano (As'_{vano}), capacidad a flexión de diseño en el vano (ϕMn_{vano}) y costo por unidad de largo (Costo). Es decir, se identifican dos variables de entrada y nueve variables de salida (ver Figura 5.4).

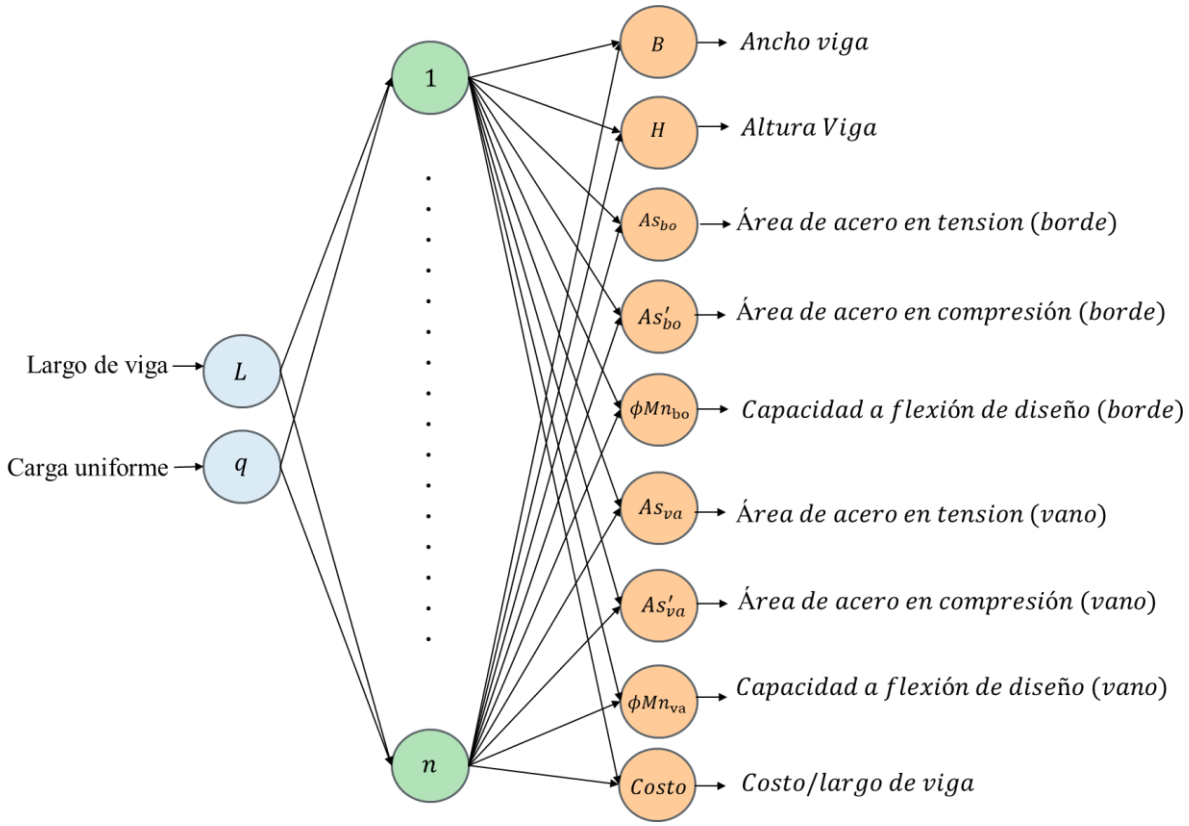

Figura 5.4 Arquitectura de la RNA caso viga Bi-Empotrada

De manera análoga al caso anterior, se desarrollan siete RNA iniciando con una neurona en la capa oculta hasta alcanzar siete neuronas. Estas redes de identifican mediante la nomenclatura:

EMP_fcA_B

EMP = Red para vigas bi-empotradas.

A = Material base hormigón G20, G25 o G30.

B = Etiqueta de la red (1, 2, 3, …, 7).

La base de datos para cada material base posee a lo menos 134 muestras con nueve salidas. En otras palabras, la base de datos alberga al menos 1206 valores de salida en total. El conjunto de datos de entrenamiento consta de 846 salidas (app. el 70% de 1206), mientras que los conjuntos de validación y prueba contienen 180 (app. el 15% de 1206) salidas cada uno.

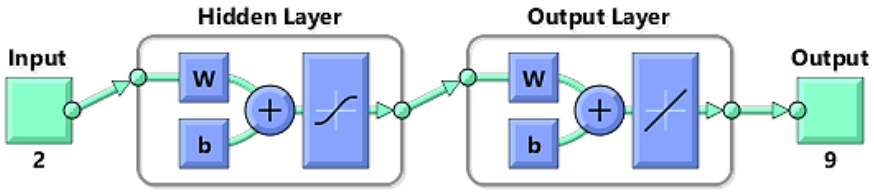

Figura 5.5 Esquema de red neuronal artificial para el diseño a flexión de vigas bi-empotradas

5.4.2 Entrenamiento de la RNA

Los resultados de entrenamiento se obtienen al calcular el factor R para el conjunto de entrenamiento, prueba y el conjunto completo de datos, junto con el MSE de la totalidad de las salidas (ver Tablas 5.4, 5.5 y 5.6).

Tabla 5.4 Resultados de entrenamiento de las RNA EMP_fc20

N° neuronas en la capa oculta	**R**			**MSE**
	Entrenamiento	Test	Total	
1	0.986	0.99	0.988	5.86E-03
2	0.991	0.99	0.99	4.37E-03
3	**0.999**	**0.999**	**0.999**	**3.72E-04**
4	0.999	0.999	0.999	3.46E-05
5	0.999	0.999	0.999	1.90E-05
6	1	1	1	9.64E-06
7	1	1	1	3.92E-06

Tabla 5.5 Resultados de entrenamiento de las RNA EMP_fc25

N° neuronas en la capa oculta	R			MSE
	Entrenamiento	Test	Total	
1	0.987	0.989	0.987	5.67E-03
2	0.995	0.99	0.995	2.09E-03
3	**0.999**	**0.999**	**0.999**	**2.20E-04**
4	0.999	0.999	0.999	5.19E-05
5	1	1	1	2.67E-05
6	1	1	1	1.12E-05
7	1	1	1	1.54E-05

Tabla 5.6 Resultados de entrenamiento de las RNA EMP_fc30

N° neuronas en la capa oculta	R			MSE
	Entrenamiento	Test	Total	
1	0.990	0.991	0.990	4.66E-03
2	0.996	0.997	0.996	1.92E-03
3	**0.999**	**0.999**	**0.999**	**2.42E-04**
4	0.999	0.999	0.999	4.91E-05
5	0.999	0.999	0.999	4.90E-05
6	0.999	0.999	0.999	1.48E-05
7	0.999	0.999	0.999	1.33E-05

Las redes EMP_fc20_3, EMP_fc25_3 y EMP_fc30_3 ajustan con una alta calidad la base de datos, destacando un valor de MSE dentro del intervalo objetivo y buen factor de correlación R, generalizando el caso de estudio (ver Figura 5.6 y Anexo 5.1.1).

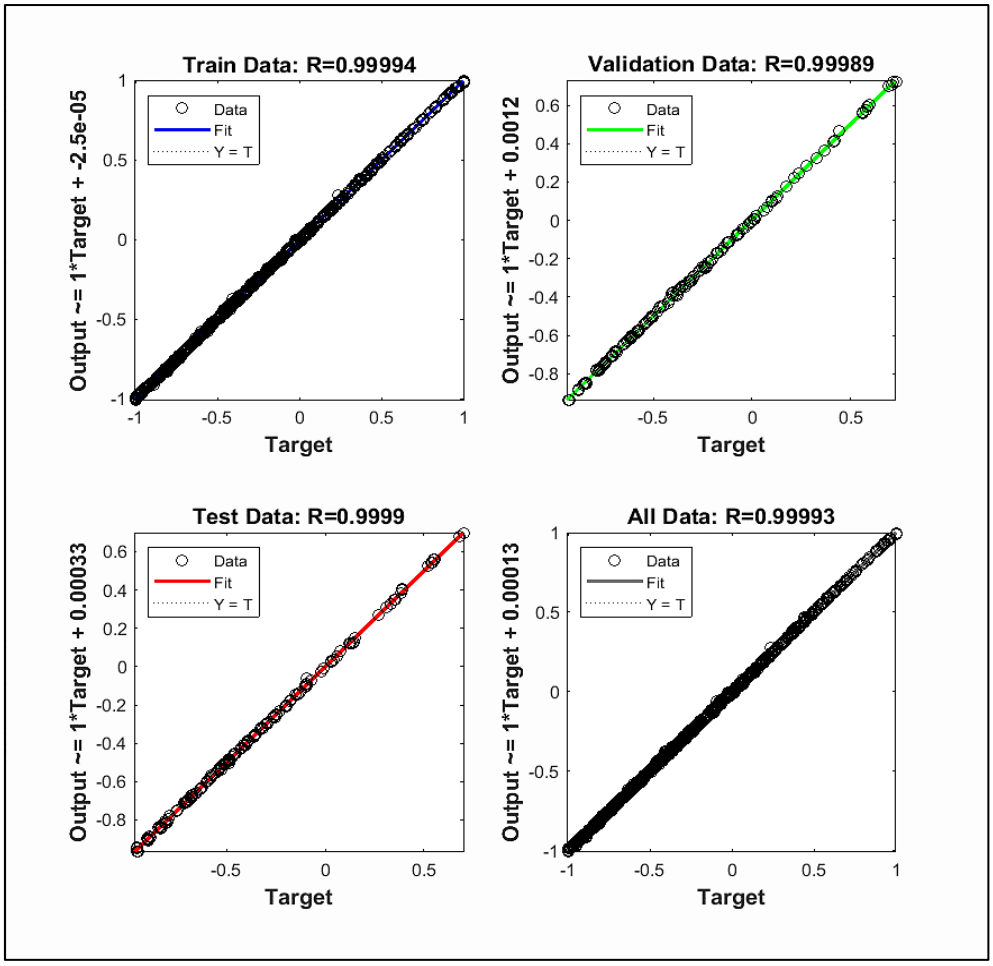

Figura 5.6 Regresión red EMP_fc20_3

5.5. CASO III: Viga Cantilever

5.5.1 Arquitectura de la RNA

Las variables de salida que generalizan el diseño a flexión de vigas corresponden al Ancho (B) y Altura inicial en empotramiento (H_1), Altura final en voladizo (H_2), Área de acero longitudinal en tensión (As), capacidad a flexión de diseño (ϕMn), y costo por unidad de largo (Costo). Es decir, se identifican dos variables de entrada y seis variables de salida (ver Figura 5.7).

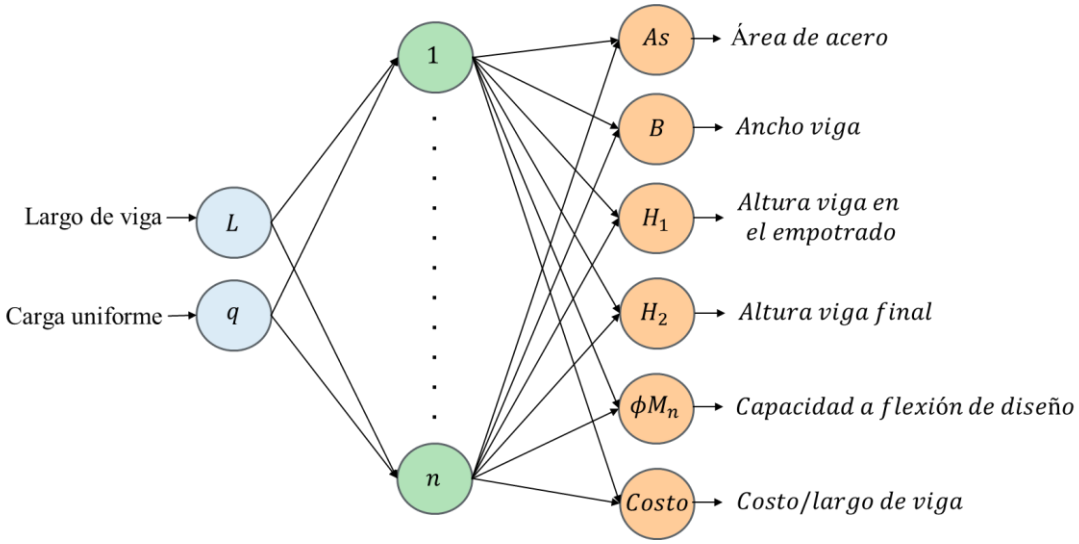

Figura 5.7 Arquitectura de la RNA caso viga cantilever

De manera análoga al Caso I y II, se desarrollan siete arquitecturas de RNA variando la cantidad de neuronas en la capa oculta. Estas redes de identifican con la nomenclatura:

CANT_fcA_B

CANT = Red para vigas cantilever.

A = Material base hormigón G20, G25 o G30.

B = Etiqueta de la red (1, 2, 3, …, 7).

La base de datos posee 155 muestras con seis salidas para las vigas CANT_fc20 y CANT_fc25. Para las vigas CANT_fc30 la base de datos posee 124 muestras con seis salidas.

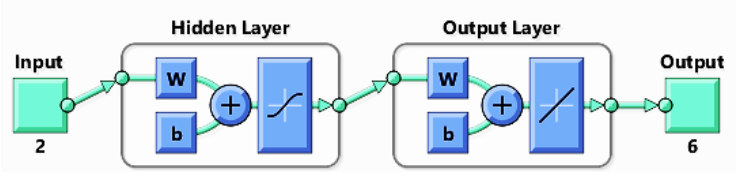

Figura 5.8 Esquema de red neuronal artificial para el diseño a flexión de vigas cantilever

5.5.2 Entrenamiento de la RNA

Los resultados de entrenamiento se obtienen al calcular el factor R para el conjunto de entrenamiento, prueba y el conjunto completo de datos, junto con el MSE de la totalidad de las salidas (ver Tablas 5.7, 5.8 y 5.9).

Tabla 5.7 Resultados de entrenamiento de las RNA CANT_fc20

N° neuronas en la capa oculta	R			MSE
	Entrenamiento	Test	Total	
1	0.988	0.989	0.987	6.85E-03
2	0.995	0.993	0.995	2.70E-03
3	**0.999**	**0.999**	**0.999**	**2.82E-04**
4	0.999	0.999	0.999	8.25E-05
5	0.999	0.999	0.999	1.23E-05
6	1	1	1	5.73E-06
7	1	1	1	1.04E-06

Tabla 5.8 Resultados de entrenamiento de las RNA CANT_fc25

N° neuronas en la capa oculta	R			MSE
	Entrenamiento	Test	Total	
1	0.988	0.986	0.987	6.76E-03
2	0.993	0.994	0.993	3.41E-03
3	0.999	0.999	0.999	4.23E-04
4	0.999	0.999	0.999	2.78E-04
5	0.999	0.999	0.999	2.73E-04
6	0.999	0.999	0.999	2.31E-04
7	**0.999**	**0.999**	**0.999**	**2.14E-04**

Tabla 5.9 Resultados de entrenamiento de las RNA CANT_fc30

N° neuronas en la capa oculta	R			MSE
	Entrenamiento	Test	Total	
1	0.986	0.99	0.988	6.50E-03
2	0.995	0.993	0.995	2.95E-03
3	0.999	0.999	0.999	4.56E-04
4	0.999	0.999	0.999	1.80E-04
5	0.999	0.999	0.999	1.29E-04
6	**0.999**	**0.999**	**0.999**	**1.13E-04**
7	0.999	0.999	0.999	1.19E-04

Las RNA CANT_fc20_3, CANT_fc25_7 y CANT_fc30_6 son las redes que ajustan con mayor calidad y precisión la base de datos (ver Figura 5.9 y Anexo 5.1.1).

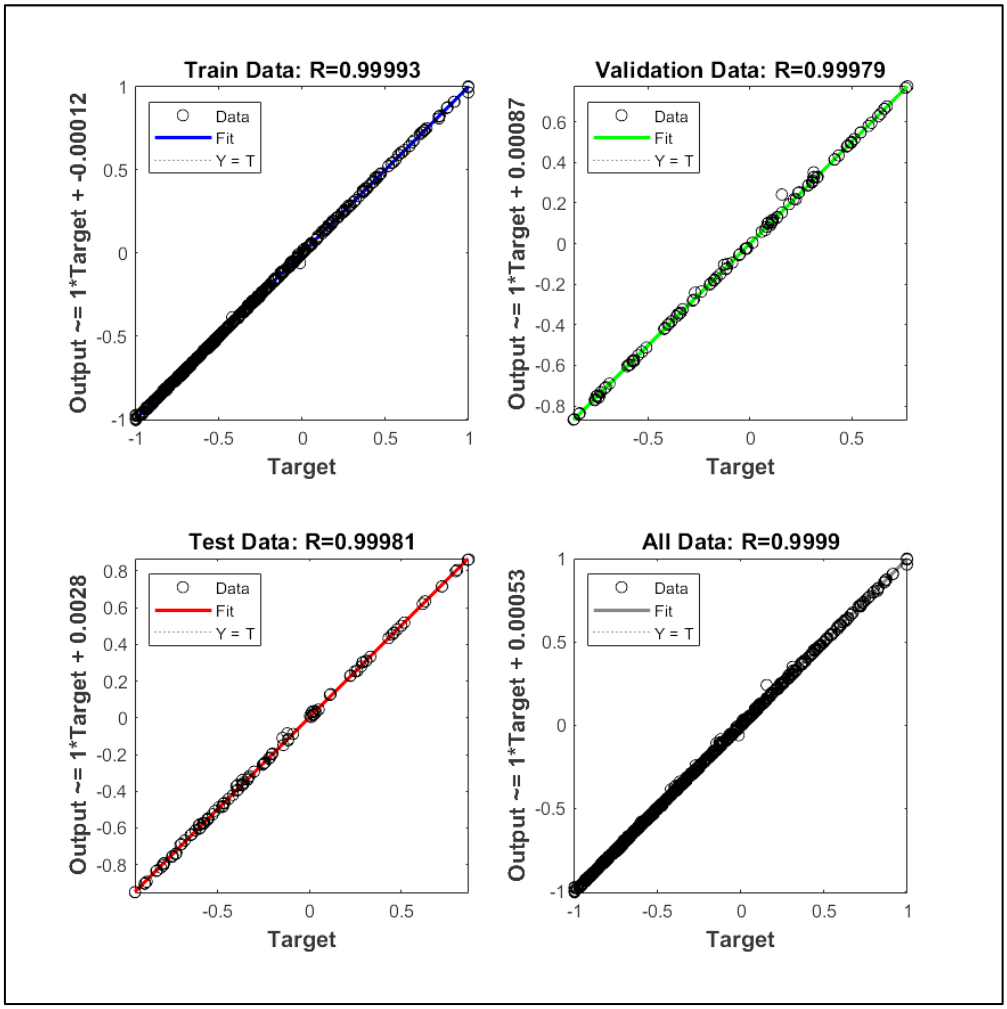

Figura 5.9 Regresión red CANT_fc20_3

5.6. Análisis de sensibilidad

Para analizar cómo las variables de salida más relevantes responden ante variaciones en las variables de entrada en el Caso I, se realiza un análisis de sensibilidad de las redes neuronales SIMP_fc20_3, SIMP_fc25_3 y SIMP_fc30_3. En este caso la longitud varía entre 3 y 7 metros y la carga entre 28 y 168 kN/m.

Los resultados de las salidas se encuentran normalizados con respecto al valor medio que estas toman, esto es para ilustrar de mejor manera las variaciones y tener un orden de magnitud de la proporción de estas variaciones.

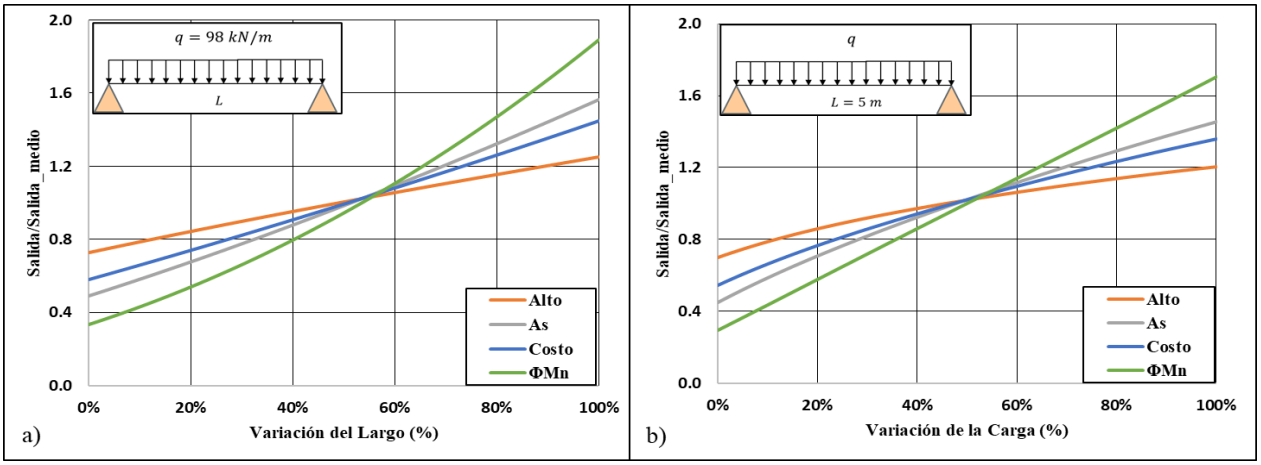

Figura 5.10 Sensibilidad Red SIMP_fc20_3: a) Fijando la carga "q". b) Fijando el largo "L"

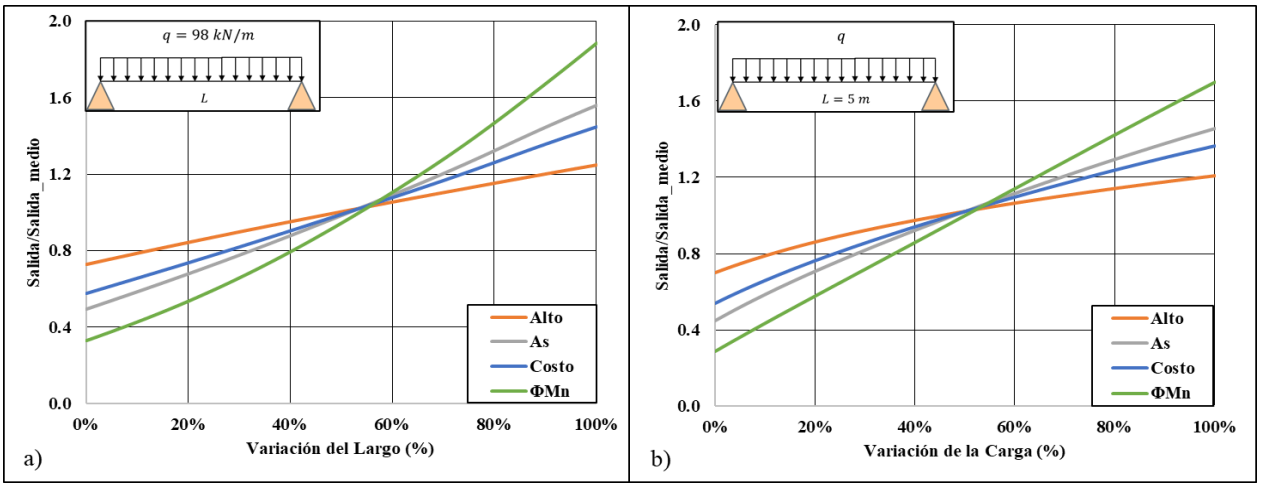

Figura 5.11 Sensibilidad Red SIMP_fc25_3: a) Fijando la carga "q". b) Fijando el largo "L"

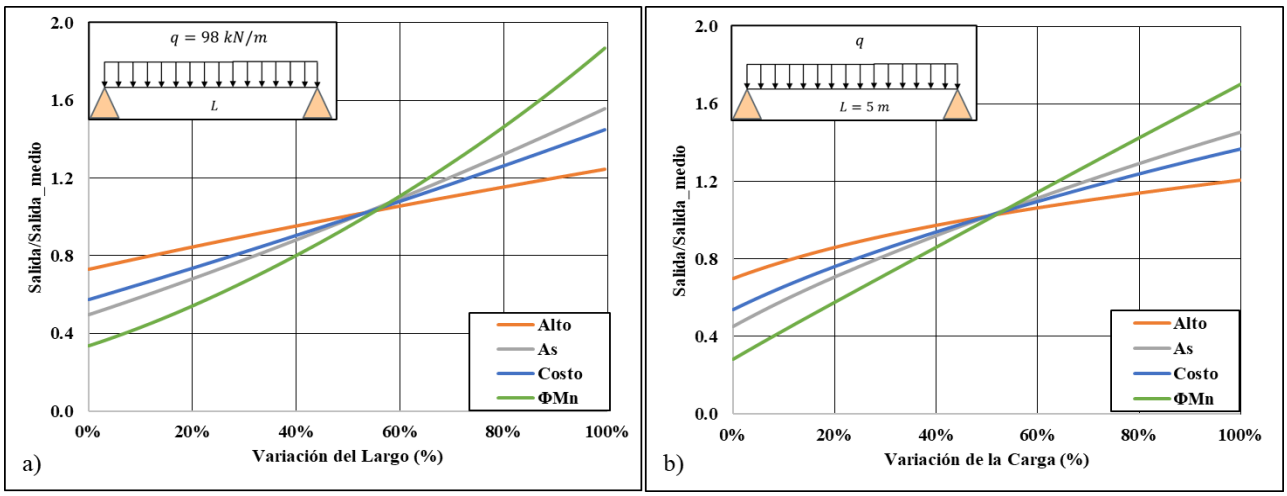

Figura 5.12 Sensibilidad Red SIMP_fc30_3: a) Fijando la carga "q". b) Fijando el largo "L"

Para el Caso II, correspondiente a la viga bi-empotrada se analizan las redes EMP_fc20_3, EMP_fc25_3 y EMP_fc30_3 en un rango completo de variación para cada variable de entrada, abarcando el 100% de su rango, es decir, el largo varía entre 3 y 7 metros y la carga entre 28 y 168 kN/m.

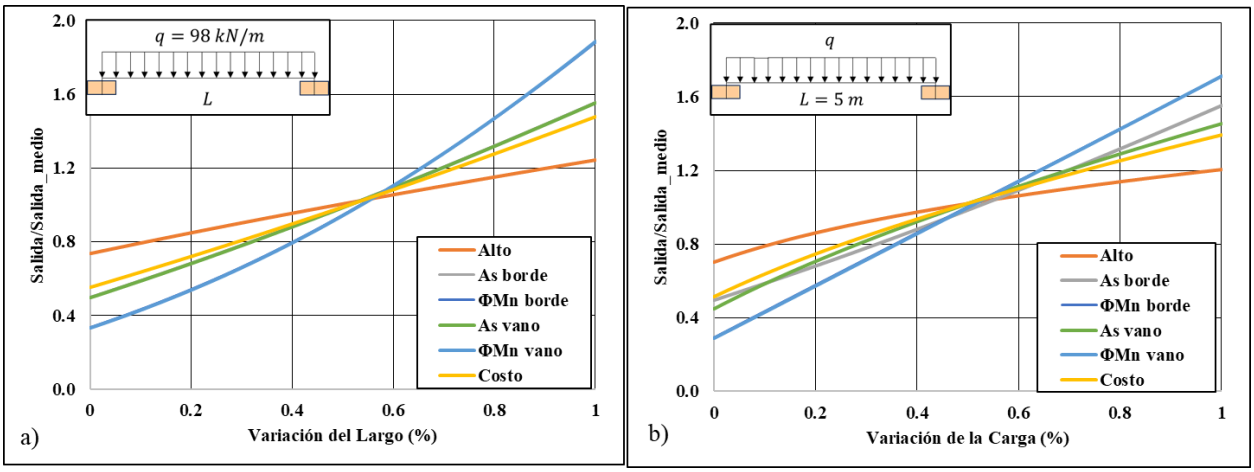

Figura 5.13 Sensibilidad Red EMP_fc20_3: a) Fijando la carga "q". b) Fijando el largo "L"

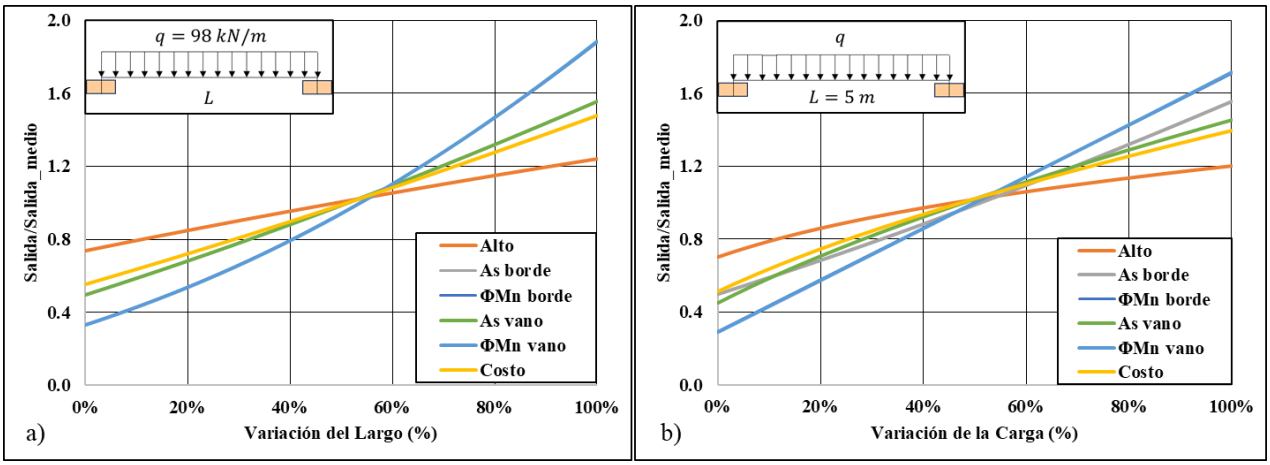

Figura 5.14 Sensibilidad Red EMP_fc25_3: a) Fijando la carga "q". b) Fijando el largo "L"

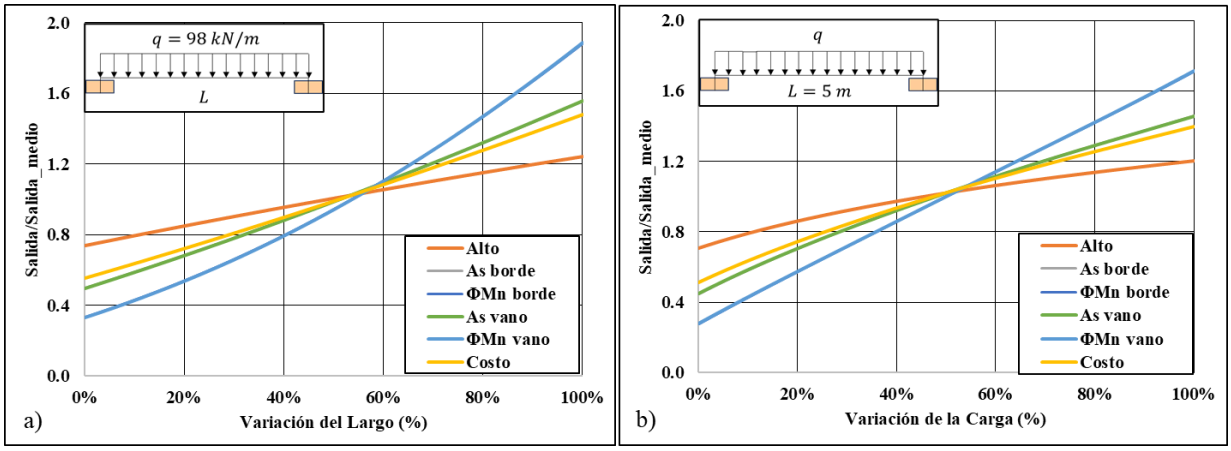

Figura 5.15 Sensibilidad Red EMP_fc30_3: a) Fijando la carga "q". b) Fijando el largo "L"

Finalmente, para el Caso III, se seleccionan las redes CANT_fc20_7, CANT_fc25_7 y CANT_fc30_6, analizando sus variables de salida en el 100% del rango de variación de sus entradas, es decir, el largo varía entre 1 y 3 metros y la carga entre 28 y 168 kN/m.

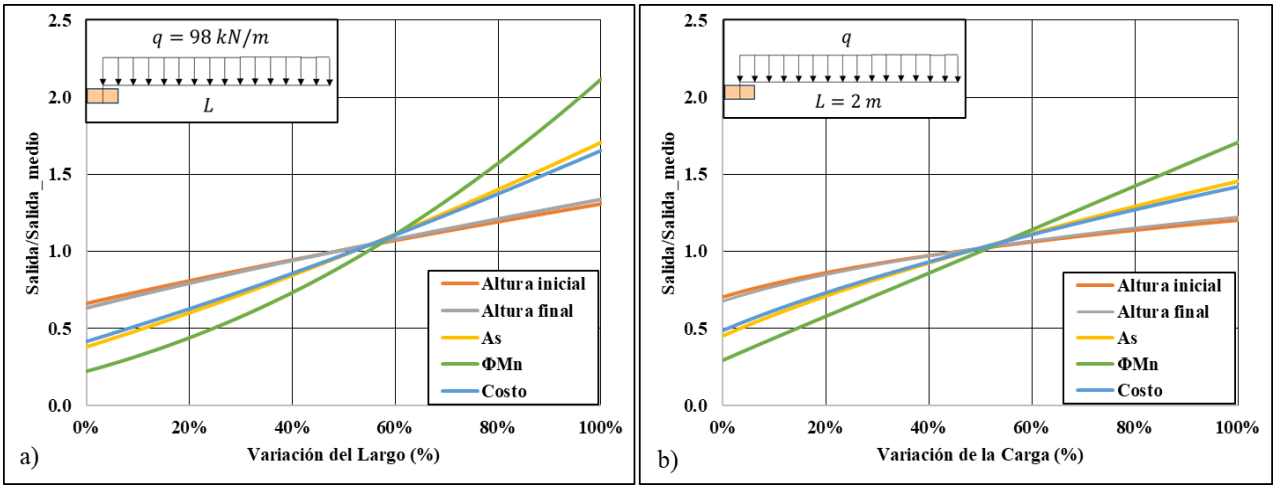

Figura 5.16 Sensibilidad Red CANT_fc20_3: a) Fijando la carga "q". b) Fijando el largo "L"

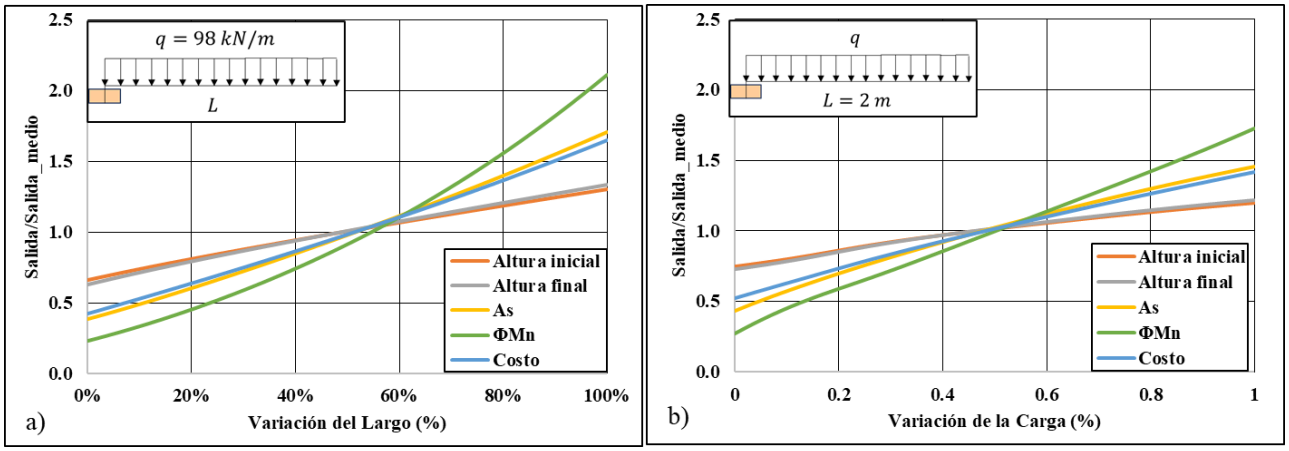

Figura 5.17 Sensibilidad Red CANT_fc25_7: a) Fijando la carga "q". b) Fijando el largo "L"

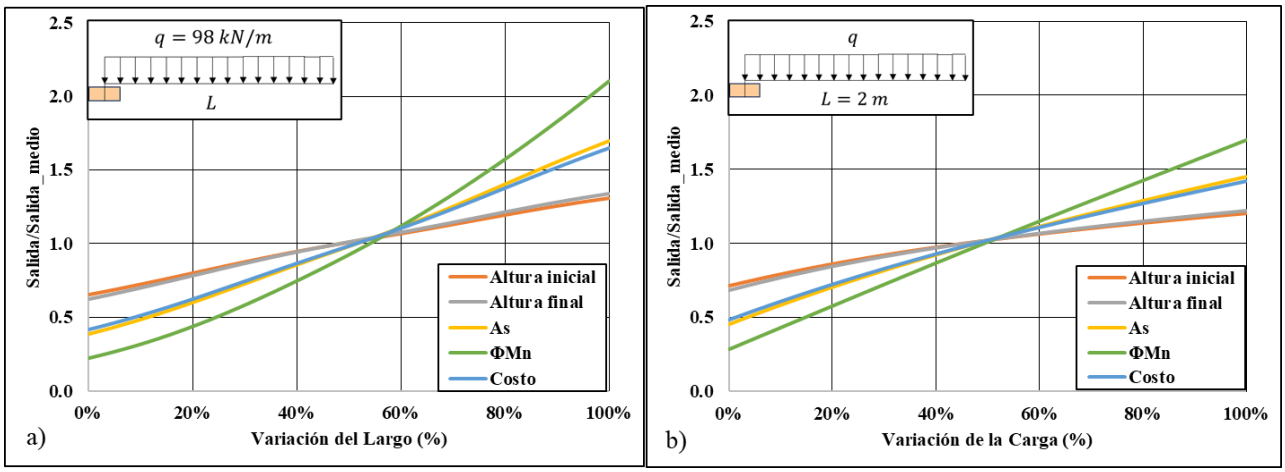

Figura 5.18 Sensibilidad Red CANT_fc30_6: a) Fijando la carga "q". b) Fijando el largo "L"

Todas las variables de salida exhiben un comportamiento creciente al aumentar la carga (q) o la longitud (L), reflejando un incremento en las dimensiones de la viga, armaduras longitudinales, capacidad y costo.

Respecto al impacto percibido por las variables de salida con respecto a la variación de las variables de entrada, se observa que las variaciones en la longitud (L) tienen un impacto más significativo en comparación con las variaciones en la carga distribuida (q). Este fenómeno se atribuye a que la demanda a flexión es proporcional al cuadrado del largo y linealmente a la carga, reflejando este impacto en las curvas de concavidad positiva para la variación del largo y negativa para la variación de la carga.

Para todos los casos de estudio la variable de salida más relevante corresponde a ϕMn puesto que este depende en forma cuadrática al largo del elemento, alcanzando variaciones en el rango de 20% y 200% con respecto al valor medio.

En menor medida de sensibilidad se encuentran el área de acero longitudinal y el costo. Por último, la variable de salida menos sensible es la altura de la viga. Esta característica se atribuye a la formulación de la base de datos, que se centra en la optimización del costo. Aumentar la altura implicaría un rápido aumento en el volumen de hormigón y en el área de moldaje, generando un incremento significativo en el costo de la viga.

En términos generales, las respuestas de las variables de salida concuerdan con las expectativas provenientes de los principios físicos que gobiernan el diseño a flexión, otorgando robustez a estos modelos. Esto se debe a que las redes han sido entrenadas con bases de datos analíticas, lo que evita la aparición de singularidades y puntos de inflexión en las curvas sensibilidad.

5.7. Interfase grafica de usuario

Debido a la extensa cantidad de casos de estudio y la complejidad asociada con la implementación de redes entrenadas, así como la necesidad de conocimientos en programación, se ha desarrollado una interfaz de usuario mediante la herramienta APP Designer de MATLAB®. Esta interfaz simplifica el proceso para el usuario al solicitar la introducción manual de la condición de apoyo, el material base, los estados de carga viva y muerta y largo de viga. La interfaz se encarga del preprocesamiento de los datos ingresados por el usuario, llamada de redes neuronales entrenadas, ejecución y el postproceso de las salidas de la red.

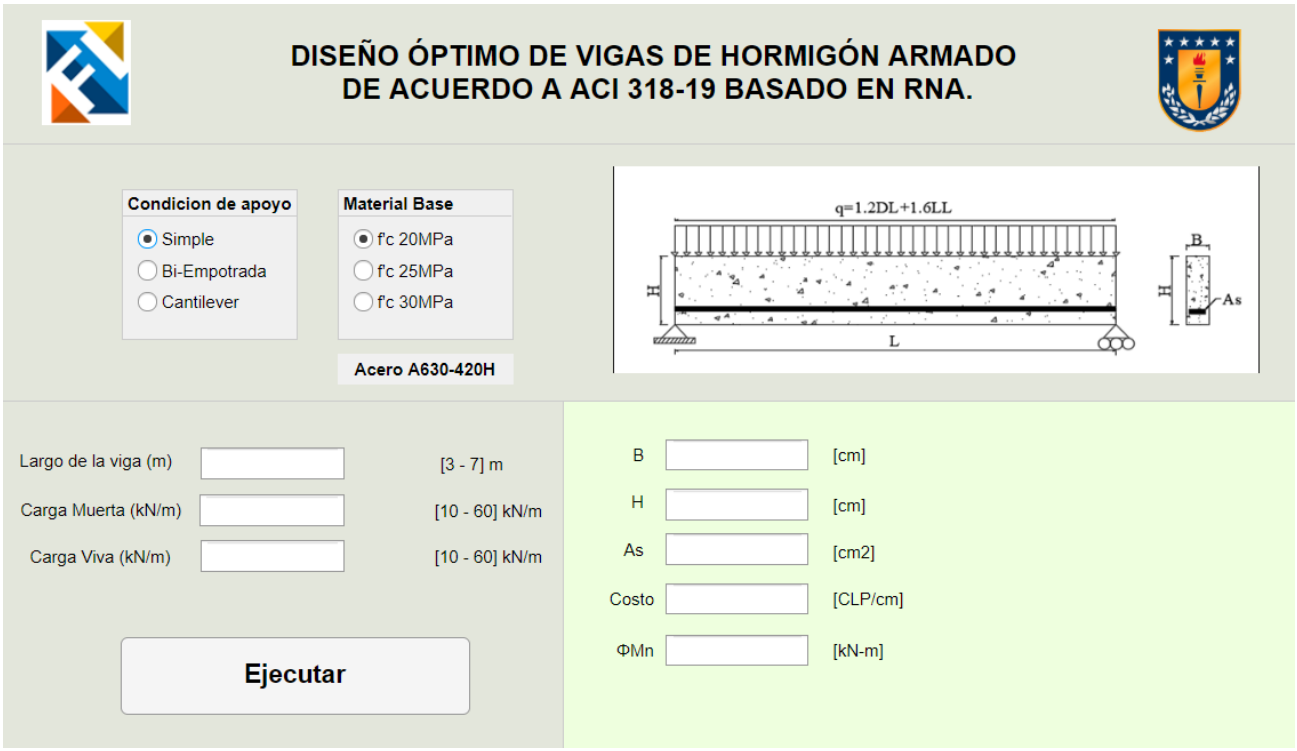

Figura 5.19 Interfaz gráfica de usuario para el caso de viga simplemente apoyada

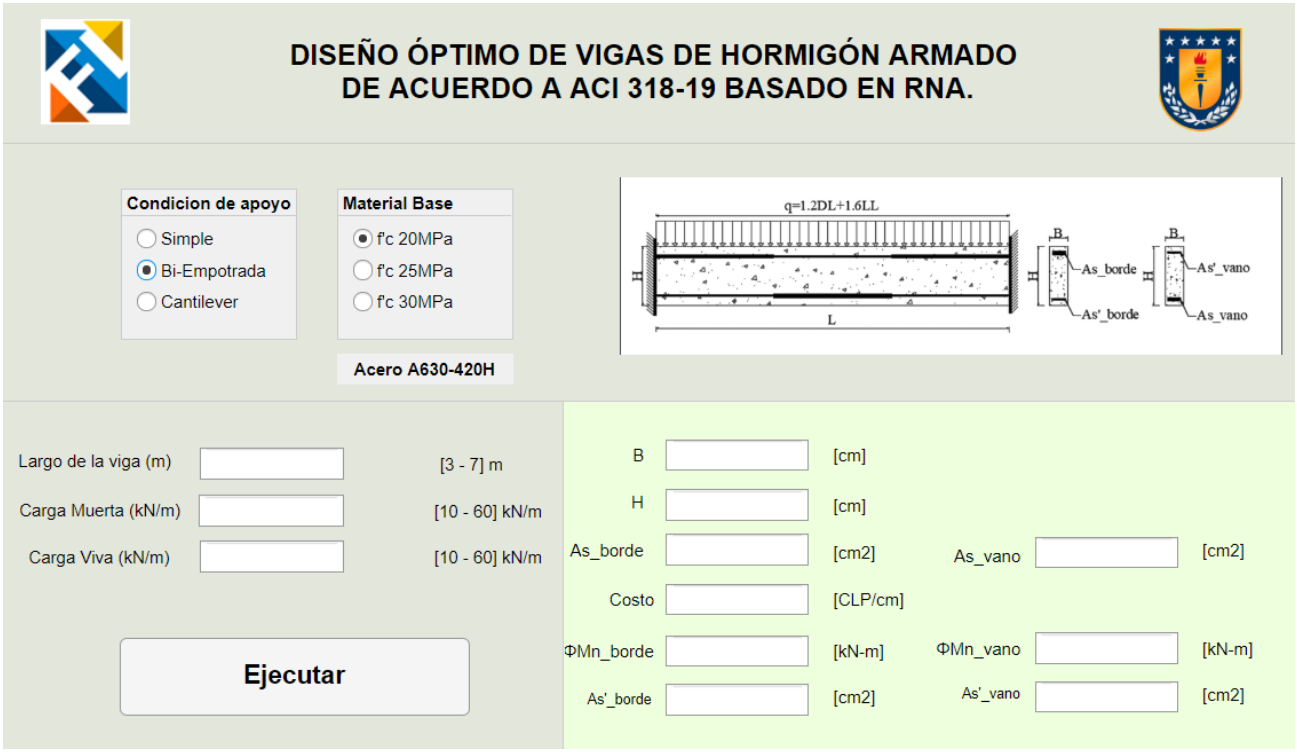

Figura 5.20 Interfaz gráfica de usuario para el caso de viga Bi-empotrada

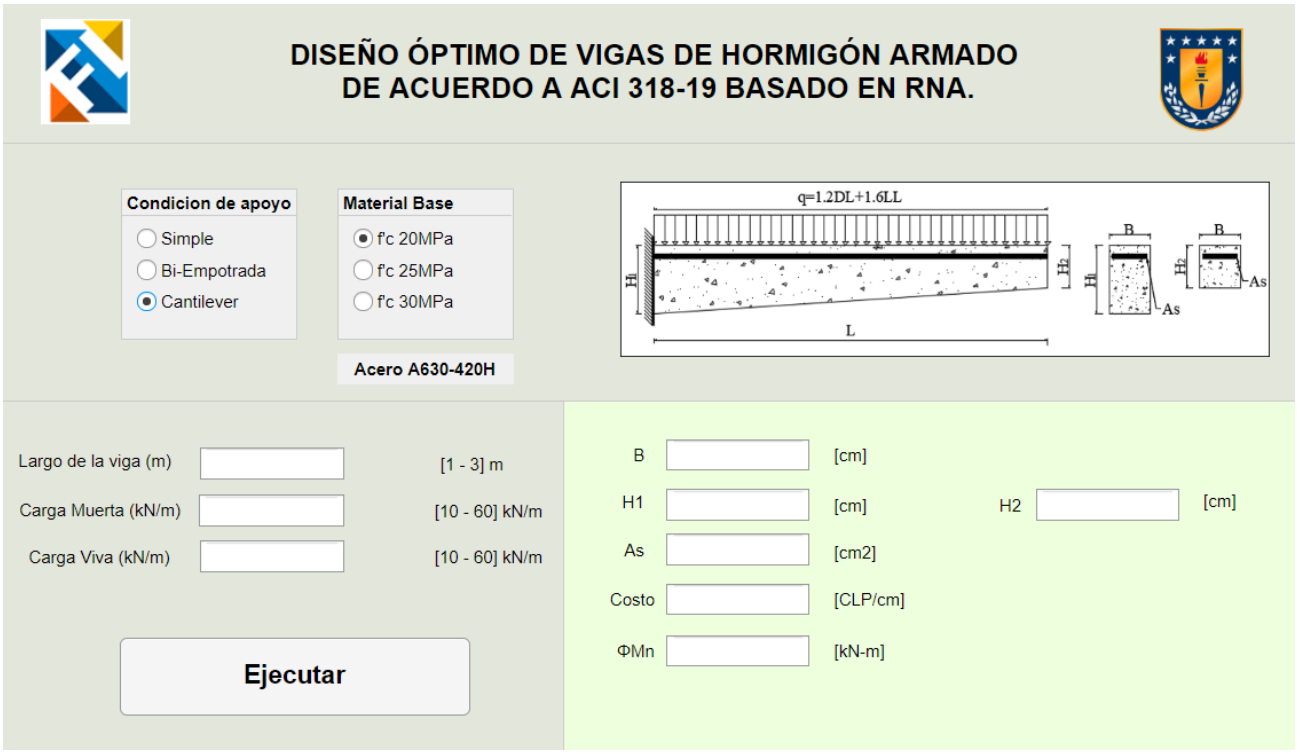

Figura 5.21 Interfaz gráfica de usuario para el caso de viga Cantilever

5.8. Conclusiones

Los resultados obtenidos tras la implementación del algoritmo de redes neuronales artificiales han demostrado un rendimiento altamente satisfactorio en los casos analizados. Para cada condición de apoyo (simplemente apoyado, bi-empotrado y en voladizo) se consideraron tres subcasos asociados a diferentes grados de hormigón utilizados como materiales base. Dentro de cada subcaso, se llevó a cabo un análisis del rendimiento de siete arquitecturas de red distintas. Como resultado, la aplicación de redes neuronales a las nueve bases de datos conllevó el entrenamiento de un total de 63 redes neuronales. Este proceso culminó en la obtención de nueve redes neuronales que demostraron un mejor desempeño en la tarea de diseñar a flexión vigas de hormigón armado.

Para cada sub-caso, la selección de la arquitectura de cada RNA se basó en los resultados de los obtenidos de los factores R y MSE posterior a la etapa de entrenamiento. Se destaca un factor R de 0.999 y un MSE del orden de 10^{-4} para arquitecturas de baja complejidad. Arquitecturas que con solo tres neuronas en la capa oculta demostraron capacidad para generalizar los casos estudiados. Este resultado se atribuye principalmente a la homogeneidad de las bases de datos utilizadas.

El análisis de sensibilidad de las RNA en el diseño a flexión de vigas de hormigón armado muestra que la longitud tiene un impacto significativo en las variables de salida, especialmente en la capacidad a flexión de diseño. La carga también influye, pero en menor medida, por otra parte, la altura es la variable de salida menos sensible a las variables de entrada debido a la optimización del costo en la base de datos.

CAPÍTULO 6: CONCLUSIONES

En este trabajo se aplicaron algoritmos de Redes Neuronales Artificiales (RNA) para el diseño óptimo a flexión de vigas de hormigón armado de acuerdo con los requerimientos del código ACI 318-19. Se desarrolló una plataforma de Inteligencia Artificial (IA) basada en RNA que se despliega gráficamente y permite al usuario realizar el diseño óptimo a flexión de vigas de hormigón armado a partir de las RNA entrenadas de mejor desempeño.

Se concluye la factibilidad de llevar a cabo el diseño de vigas rectas a flexión de hormigón armado mediante RNA, obteniendo secciones transversales, armaduras longitudinales y resistencia de diseño. Se debe tener en cuenta que la interpolación de valores de entrada genera resultados con decimales ajenos a las dimensiones (H, B y As) encontrados comúnmente en la práctica. Al respecto, es imprescindible la aproximación de estos valores por parte del usuario con los decimales adecuados al contexto de cada dimensión.

La revisión de los alcances de RNA en aplicaciones en Ingeniería Estructural indica que este algoritmo de ML se destaca por su fácil implementación computacional y versatilidad a la hora de abordar problemas de esta disciplina. En este caso, las arquitecturas de redes solo requerían una capa oculta y las variables de entrada “carga distribuida” y “largo de viga” para obtener las variables de salida que generalizan el diseño a flexión de vigas.

En base a los resultados de construcción y entrenamiento de RNA, se obtuvo que existe una relación directa entre la cantidad de neuronas en la capa oculta y la eficacia de aprendizaje de la red. Las redes con menos neuronas demostraron un mayor error de predicción en comparación con las redes con más neuronas, las cuales lograron valores de error oscilando entre 10^{-6} y 10^{-4}. No obstante, al mejorar la precisión en el aprendizaje de la base de datos, surge el riesgo de sobreajuste de la red, comprometiendo la capacidad de generalización del problema. Por esta razón, se optó por seleccionar arquitecturas que poseen un MSE mínimo de 10^{-4}.

A partir de los resultados de análisis de sensibilidad, se evidencia que las variables de salida de las RNA entrenadas responden a los principios físicos provenientes del diseño a flexión de vigas. Demostrando que el largo de la viga influye en mayor medida que la carga, pues la demanda de flexión aumenta con el cuadrado del largo. La robustez de estos modelos de RNA, se debe principalmente a la homogeneidad de las bases de datos utilizadas, pues existió una metodología llevada a cabo para dar cumplimiento de las restricciones de diseño referentes a la normativa vigente y la depuración de datos anómalos.

A pesar de la calidad de los resultados de entrenamiento, se destaca que la efectividad de las redes neuronales está vinculada a la representatividad de las bases de datos utilizadas. La precisión de las predicciones de la red podría verse afectada en el caso que se considere una base de datos con menor cantidad de estados de carga o largos de viga, o si presenta una distribución irregular de los datos para el mismo rango de valores de estudio.

Se recomienda utilizar estas herramientas de manera consciente, comprendiendo los fundamentos matemáticos propios de la física del problema y la formulación del algoritmo, pues los resultados proporcionados están limitados a dominios de valores de entrada específicos según los datos provistos en su etapa de entrenamiento.

REFERENCIAS

ACI (American Concrete Institute) Committee 318. (2019). Building Code Requirements or Structural Concrete (ACI 318-19) and Commentary (ACI 318R-19). Farmington Hills, MI.

Aguilar, V. (2013). Estimación de la resistencia al corte de muros de albañilería armada. Estudio comparativo y uso de redes neuronales artificiales [Estimation of shear strength of reinforced masonry walls. Comparative study and use of artificial neural networks]. Memoria de Ingeniero Civil. Valdivia: Universidad Austral de Chile.

Aguilar, V., Sandoval, C., Adam, J. M., Garzón-Roca, J., & Valdebenito, G. (2016). Prediction of the shear strength of reinforced masonry walls using a large experimental database and artificial neural networks. *Structure and Infrastructure Engineering*, 12(12), 1661-1674. https://doi.org/10.1080/15732479.2016.1157824

Bojórquez, J., Tolentino, D., Ruiz, S., Bojórquez, E. (2016). DISEÑO SÍSMICO PRELIMINAR DE EDIFICIOS DE CONCRETO REFORZADO USANDO REDES NEURONALES ARTIFICIALES. *Concreto y Cemento. Investigacion y Desarrollo*, 7(2), 60-78.

Chakrabarty, B. K. (1992). Model for Optimal Design of Reinforced Concrete Beam. *Journal of Structural Engineering*, *118*(11), 3238-3242. https://doi.org/10.1061/(ASCE)0733-9445(1992)118:11(3238)

Chopra, P., Sharma, R. K., Kumar, M., & Chopra, T. (2018). Comparison of Machine Learning Techniques for the Prediction of Compressive Strength of Concrete. *Advances in Civil Engineering*, *2018*, 1-9. https://doi.org/10.1155/2018/5481705

Demuth, H., Beale, M. y Hagan, M. *Neural Networks Toolbox Users Guide, Version 4* (The MathWorks Inc, Natick, USA, 2002)

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. [Disponible en: http://www.deeplearningbook.org

Instituto Nacional de Normalización. (2008). *Hormigón armado - Requisitos de diseño y cálculo.* (NCh 430.Of2008). Santiago, Chile: Autor.

Kotsovou, G. M., Ahmad, A., Cotsovos, D. M., & Lagaros, N. D. (2020). Reappraisal of methods for calculating flexural capacity of reinforced concrete members. *Proceedings of the Institution of Civil Engineers - Structures and Buildings*, *173*(4), 279-290. https://doi.org/10.1680/jstbu.18.00110

Mehdi Ghasri (2023). ANN MATLAB code. (https://www.mathworks.com/matlabcentral/fileexchange/122912-ann-matlab-code), MATLAB Central File Exchange. Recuperado September 9, 2023.

Microsoft Corporation. *(2023). Microsoft 365 [Software]. Redmond, WA: Microsoft.*

Mukherjee, A., & Deshpande, J. M. (1995). Modeling Initial Design Process using Artificial Neural Networks. *Journal of Computing in Civil Engineering*, *9*(3), 194-200. https://doi.org/10.1061/(ASCE)0887-3801(1995)9:3(194)

Navarro Ferrer, F. (2016). *Modelos predictivos de las características prestacionales de hormigones fabricados en condiciones industriales* [Universitat Politècnica de València]. https://doi.org/10.4995/Thesis/10251/67272

Senouci, A. (2000). *PRELIMINARY DESIGN OF REINFORCED CONCRETE BEAMS USING NEURAL NETWORKS*. Engineering Journal of the University of Qatar, Vol. 13.

Senouci, A. B. and Abdul-Salam M.A. (1998). *Prediction of Reinforced Concrete Beam Depth Using Neural Networks*. Engineering Journal of the University of Qatar, Vol. 11, pp. 117-132.

Thai, H.-T. (2022). Machine Learning for Structural Engineering: A state-of-the-art review. Structures, 38, 448–491. https://doi.org/10.1016/j.istruc.2022.02.003

Vasquez, J. (2022). Optimización de dos tipos de losas no preesforzadas de grandes luces en edificacioens de concreto armado. Tesis de Ingeniero Civil Ambiental. Chiclayo: Universidad Católica Santo Toribio de Mogrovejo. https://orcid.org/0000-0003-0076-3211/print

Wight, J. K. (2016). *Reinforced concrete: Mechanics and design, global edition*.

Zarringol, M., Thai, H.-T., & Naser, M. Z. (2021). Application of machine learning models for designing CFCFST columns. *Journal of Constructional Steel Research*, *185*, 106856. https://doi.org/10.1016/j.jcsr.2021.106856

ANEXOS 3.1 CÓDIGO PARA CONSTRUIR UNA RNA

A continuación, se presenta la rutina empleada para la construcción y entrenamiento de redes neuronales artificiales, esta rutina corresponde a una modificación de la rutina publicada por Mehdi Ghasri. A través de esta rutina se modifica la cantidad de neuronas en la capa oculta, el algoritmo de entrenamiento, las funciones de activación, etc.

```
% Copyright (c) 2021, Mehdi Ghasri
% Email: Eng.mehdighasri@gmail.com
% All rights reserved.
%
% Redistribution and use in source and binary forms, with
or without
% modification, are permitted provided that the following
conditions are met:
%
% * Redistributions of source code must retain the above
copyright notice, this
%   list of conditions and the following disclaimer.
%
% * Redistributions in binary form must reproduce the above
copyright notice,
%   this list of conditions and the following disclaimer in
the documentation
%   and/or other materials provided with the distribution
% * Neither the name of IUPUI nor the names of its
%   contributors may be used to endorse or promote products
derived from this
%   software without specific prior written permission.
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE
% DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE
% FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,
OR CONSEQUENTIAL
% DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR
% SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER
% CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY,
```

```
% OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE
% OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

clc
clear all

%%
%%base de datos

load('entrada_norm.mat')
load('salida_norm.mat')

inputs=[entrada_norm'];
targets=[salida_norm'];

%%
%%creacion de la red

neuronas_ocultas = 7;
TF={'tansig','purelin'};
red=newff(inputs,targets,neuronas_ocultas,TF);


%% division de datos

red.divideParam.trainRatio=70/100;
red.divideParam.valRatio=15/100;
red.divideParam.testRatio=15/100;

%% Caracterizacion de la red; aprendizaje, desempeño,
epocas

red.trainFcn='trainlm';
red.performFcn = 'mse';

red.plotFcns =
{'plotperform','ploterrhist','plotregression','plotfit'};
red.trainParam.showWindow=true;
red.trainParam.showCommandLine=false;
red.trainParam.show=1;
red.trainParam.epochs=500;
red.trainParam.goal=1e-9;
red.trainParam.max_fail=20;


%% entrenamiento

[red,tr]=train(red,inputs,targets);
```

```
%test
outputs = red(inputs);
errors = gsubtract(targets,outputs);
performance = perform(red,targets,outputs);


%% Recalculate Training Performance

trainInd=tr.trainInd;
trainInputs = inputs(:,trainInd);
trainTargets = targets(:,trainInd);
trainOutputs = outputs(:,trainInd);
trainErrors = trainTargets-trainOutputs;
trainPerformance = perform(red,trainTargets,trainOutputs);

%% Recalculate Validation Performance

valInd=tr.valInd;
valInputs = inputs(:,valInd);
valTargets = targets(:,valInd);
valOutputs = outputs(:,valInd);
valErrors = valTargets-valOutputs;
valPerformance = perform(red,valTargets,valOutputs);

%% Recalculate Test Performance

testInd=tr.testInd;
testInputs = inputs(:,testInd);
testTargets = targets(:,testInd);
testOutputs = outputs(:,testInd);
testError = testTargets-testOutputs;
testPerformance = perform(red,testTargets,testOutputs);

%% View the Network

view(red);

%% Plots
% Uncomment these lines to enable various plots.

figure;
plotperform(tr);
figure;
plottrainstate(tr);
figure;
plotregression(trainTargets,trainOutputs,'Train Data',...
    valTargets,valOutputs,'Validation Data',...
    testTargets,testOutputs,'Test Data',...
    targets,outputs,'All Data');
figure;
ploterrhist(errors);
```

```
pesos_inputs=[red.IW];
pesos_capa_oculta=[red.lw];
sesgos=[red.b];
```

ANEXO 5.1: RESULTADOS DE ENTRENAMIENTO DE RNA

A continuación, se presentan los resultados del entrenamiento de las redes mejor calificadas para cada caso de estudio. Se obtienen los gráficos de regresión, las matrices de pesos (IW y LW) y sesgos (B_1 y B_2) que conforman las redes.

Anexo 5.1.1: Resultados de regresión de RNA

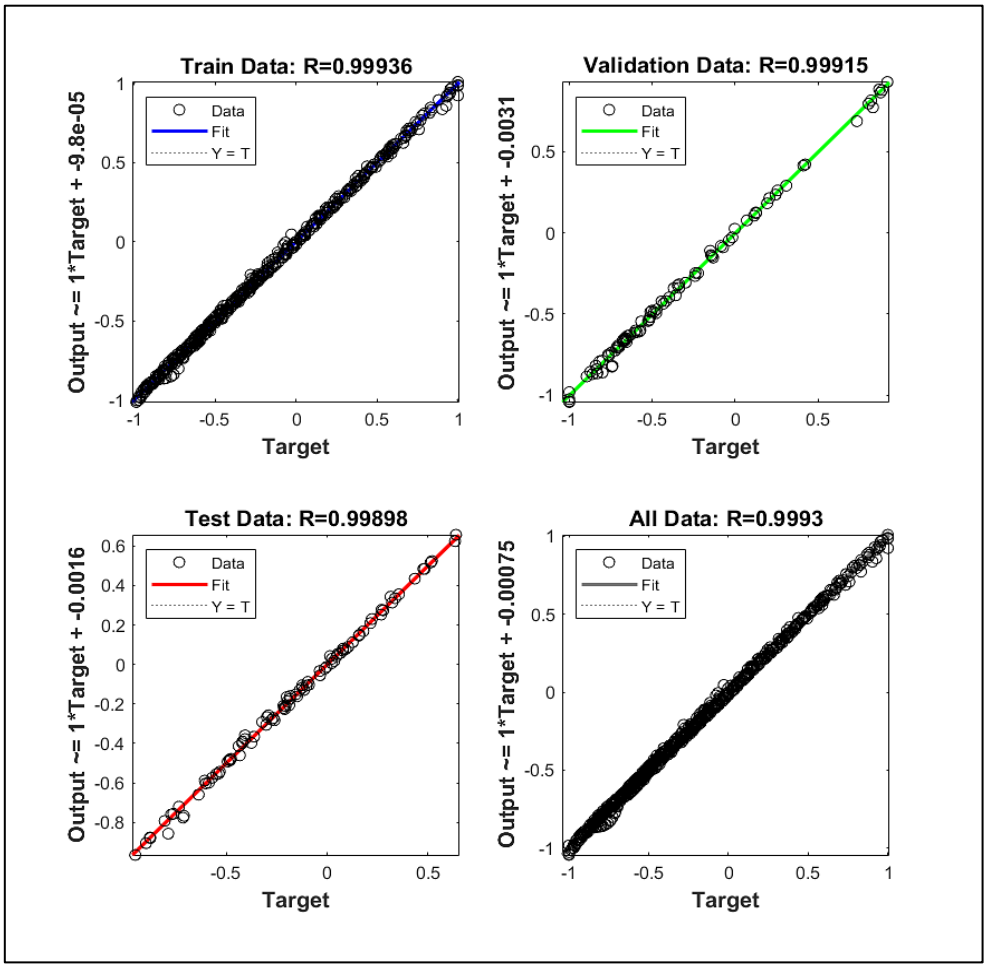

Figura A.5.1.1 Regresión red SIMP_fc25_3

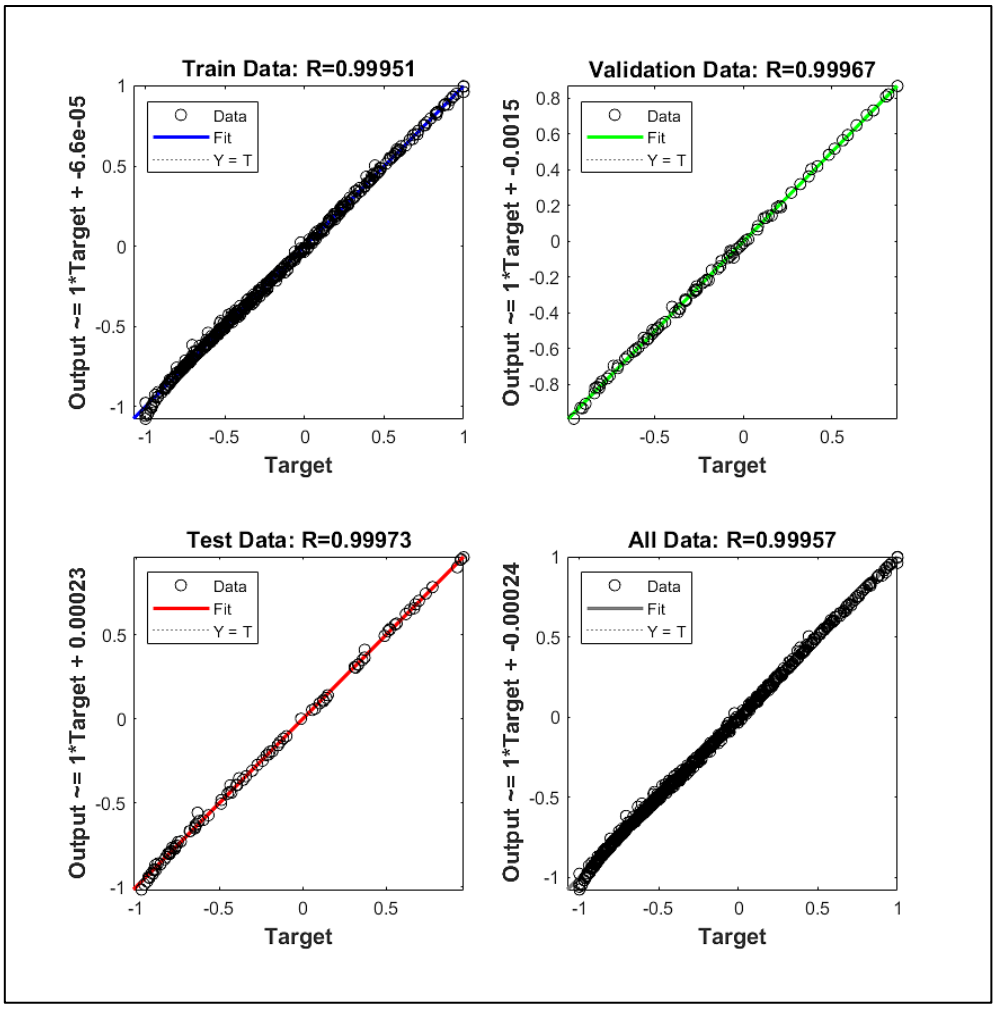

Figura A.5.1.2 Regresión red SIMP_fc30_3

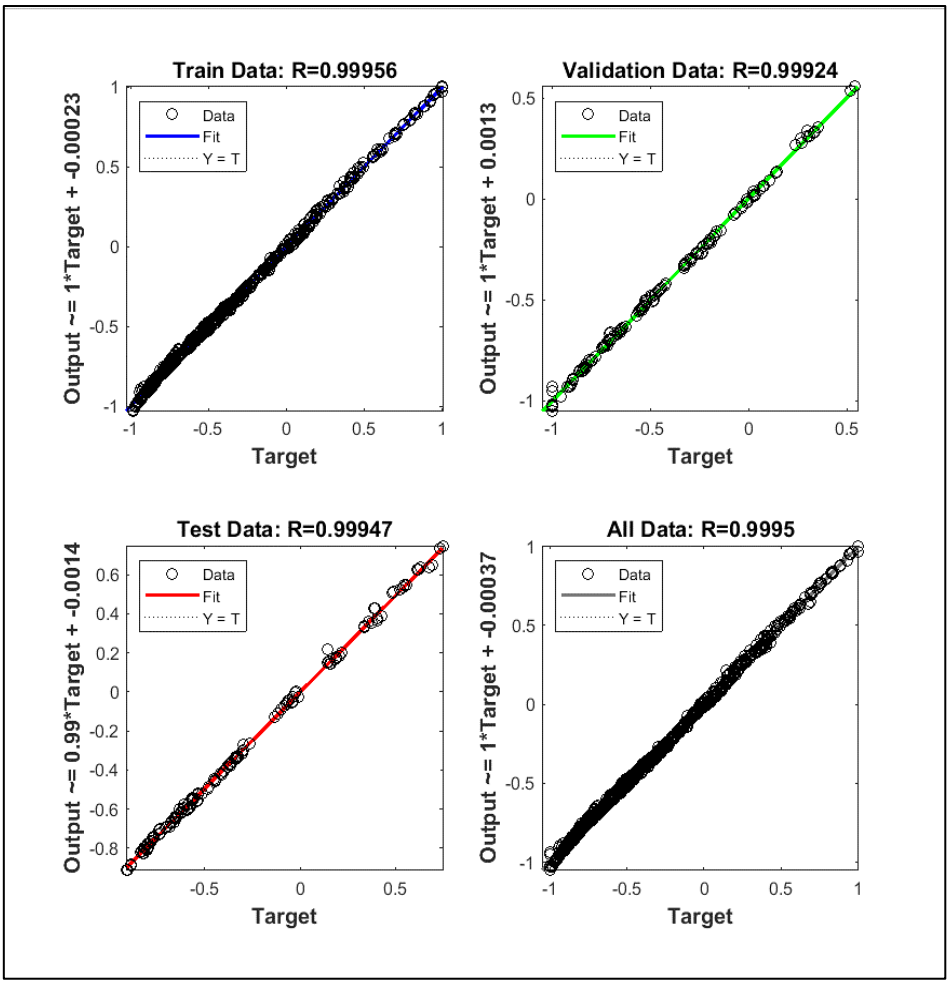

Figura A.5.1.3 Regresión red EMP_fc25_3

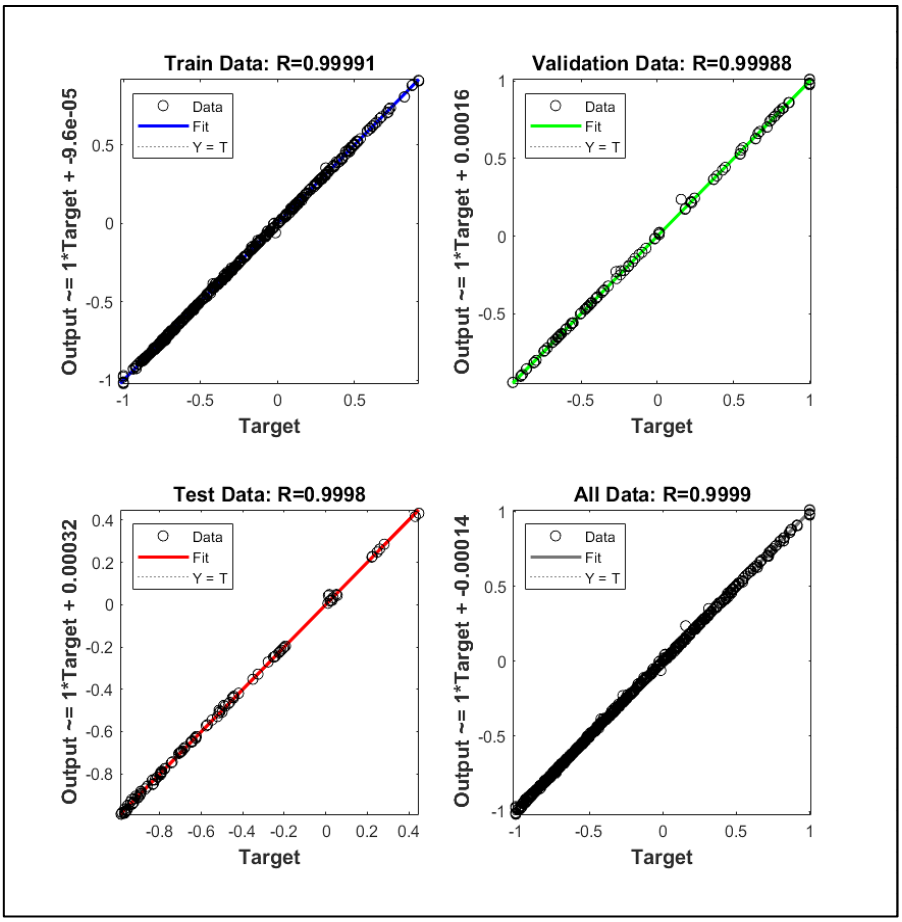

Figura A.5.1.4 Regresión red EMP_fc30_3

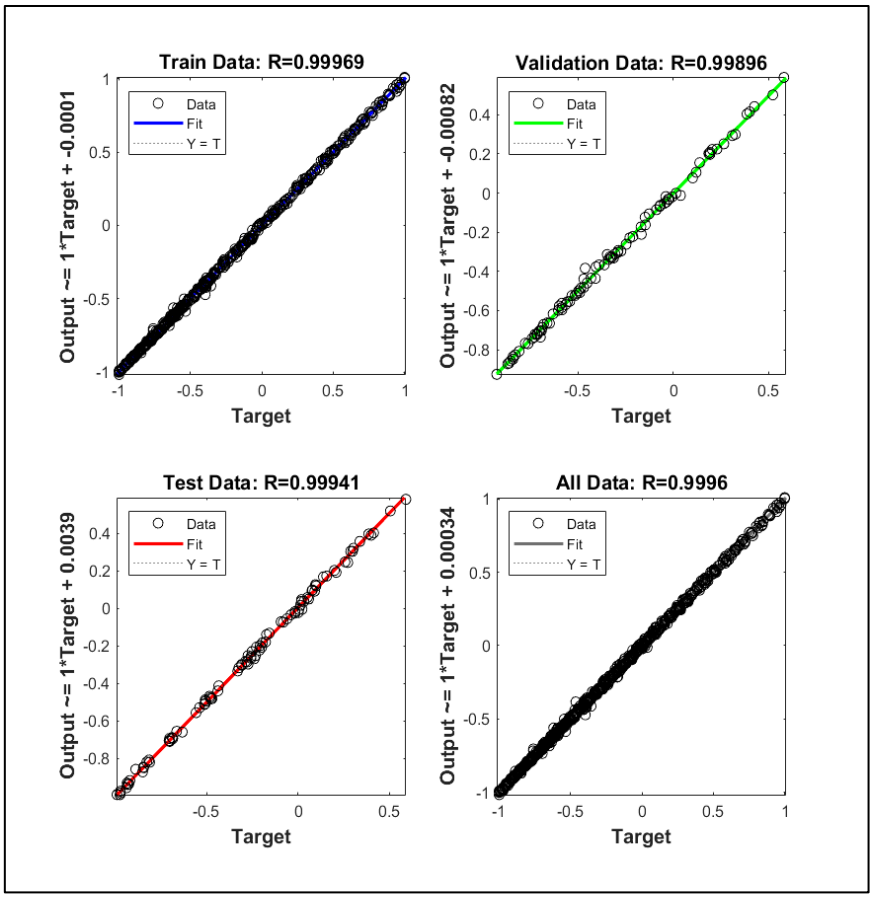

Figura A.5.1.5 Regresión red CANT_fc25_7

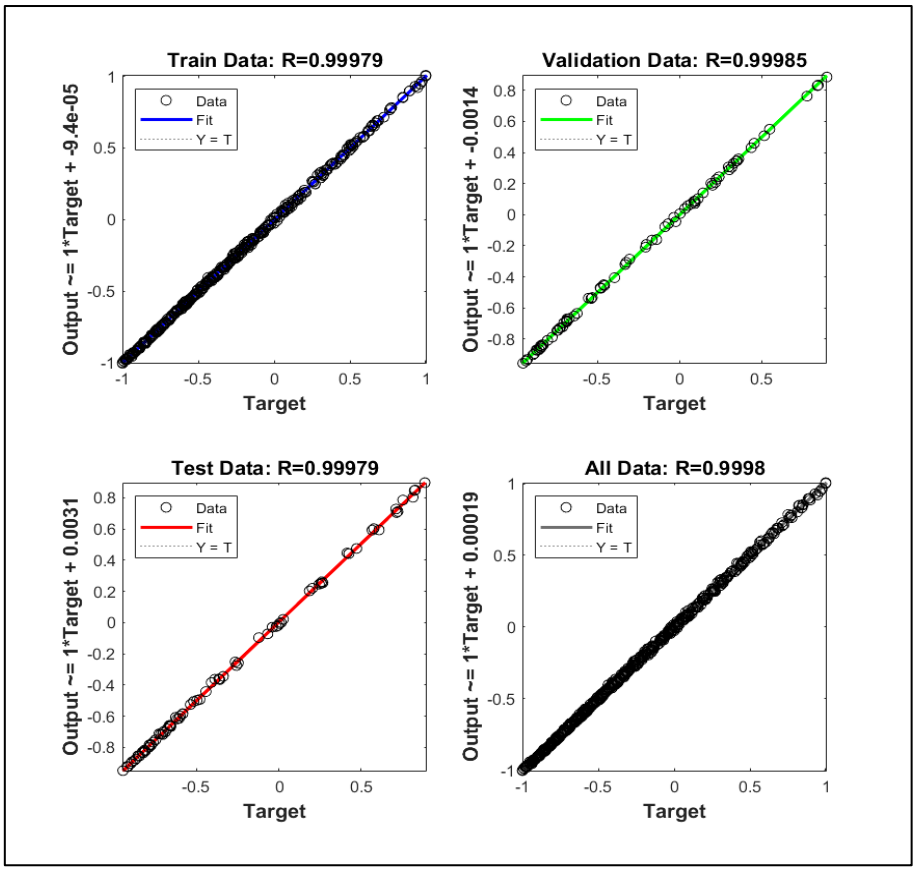

Figura A.5.1.6 Regresión red CANT_fc30_6

Anexo 5.1.2: Matrices y Vectores RNA Vigas Simplemente apoyadas

Red SIMP_fc20_3

IW =

-0.13519875	1.29588353
-0.41901396	-0.24995192
-0.28732208	0.33059247

LW =

1.4786	-1.0092	0.8373
1.4786	-1.0092	0.8373
0.6732	-2.1230	1.6855
1.4786	-1.0092	0.8373
0.8589	-1.8661	1.4899

B_1 =

2.5608969
1.23948967

B_2 =

-2.45180028
-2.45180028
-0.9714316

1.36559176

-2.45180028
-1.31279549

Red SIMP_fc25_3

IW =

0.12116066	-1.06239053
3.65005765	-0.38640096
0.91809029	-0.19899795

LW =

-3.4972	0.0045	0.0621
-3.4972	0.0045	0.0621
-2.0142	0.0261	0.2922
-3.4972	0.0045	0.0621
-2.3406	0.0214	0.2415

B_1 =

-2.60243397
-2.3708921
0.02938307

B_2 =

-4.12211982
-4.12211982
-2.38518392
-4.12211982
-2.76751221

Red SIMP_fc30_3

IW =

-0.87785281	-0.81371628
0.25591546	-1.23829895
-0.74968803	0.95120272

1.47857175	-1.00920575	0.83727165	-0.75083387	-0.69080973	-0.04532339	1.02117365
1.47857175	-1.00920575	0.83727165	-0.75083387	-0.69080973	-0.04532339	1.02117365

0.67321989	-2.12297111	1.68550258	-1.91448054	1.6967047	0.32787965	0.88944417
1.47857175	-1.00920575	0.83727165	-0.75083387	-0.69080973	-0.04532339	1.02117365
0.85892906	-1.86614369	1.48990575	-1.64615078	1.14615858	0.24182133	0.91982017
-0.17444279	-3.77794904	2.92268413	-0.70631801	1.04565095	0.44859632	0.41413543

B_1 =

1.8537493
-2.375003
0.77090294

B_2 =

-1.62447361
-1.62447361
0.12888182
-1.62447361
-0.23718801

Anexo 5.1.3 Matrices y Vectores RNA Vigas Bi-empotradas

Red EMP_fc20_3

IW =

-0.03646	1.182408
0.579059	-0.27013
0.466425	0.177744

LW =

0.067326	-0.52065	1.22495
0.067326	-0.52065	1.22495
-0.92183	-1.1614	2.515615
-1.13005	-1.15199	2.500308
0.067326	-0.52065	1.22495
0.540017	-1.94612	4.100285
-1.13005	-1.15199	2.500308
-1.13005	-1.15199	2.500308
0.818183	-0.48514	1.153914

B_1 =

2.425737
-1.33551
-1.11476

B_2 =

-2.24489
-2.24489
-0.72977
-0.74909
-2.24489
1.098185
-0.74909
-0.74909
-2.36892

Red EMP_fc25_3

IW =

-0.36123	0.158116
-0.12442	1.08585
-0.32852	-0.08043

LW =

1.403335	7.850575	-1.99199
1.403335	7.850575	-1.99199
4.157199	3.816606	-5.43406
4.136834	3.879274	-5.41759
1.403335	7.850575	-1.99199
7.180512	-1.03099	-9.17914
4.136834	3.879274	-5.41759
4.136834	3.879274	-5.41759
1.213144	8.133864	-1.7786

B_1 =

1.291688
3.114667
0.991063

B_2 =

-8.68475
-8.68475
-3.93869
-3.97717
-8.68475
1.468176
-3.97717
-3.97717
-9.03849

Red EMP_fc30_3

IW =

0.961841	-0.4341
0.140775	-1.14861
-0.64862	-0.30918

LW =

-0.20588	-0.88117	-0.74416
-0.20588	-0.88117	-0.74416
-0.42878	-0.45385	-1.35544
-0.42815	-0.44691	-1.34828
-0.20588	-0.88117	-0.74416
-0.69048	0.001986	-2.11141
-0.42815	-0.44691	-1.34828
-0.42815	-0.44691	-1.34828
-0.25859	-0.80496	-0.80531

B_1 =

-1.52775
-1.98957
1.202197

B_2 =

-0.88758
-0.88758
-0.11547
-0.10048
-0.88758
0.562089
-0.10048
-0.10048
-0.98447

Anexo 5.1.4 Matrices y Vectores RNA Vigas Cantilever

Red CANT_fc20_3

IW =

0.636322	0.376686
-0.27269	-0.29261
0.130444	-1.26925

LW =

0.464797	-2.36208	-2.31029
0.464797	-2.36208	-2.31029
0.464797	-2.36208	-2.31029

1.002818	-3.23803	-1.01789
0.963249	-3.17361	-1.11294
1.94068	-1.95244	0.142687

B_1 =

-1.31366
-0.32993
-2.81703

B_2 =

-4.17787
-4.17787
-4.17787
-3.78317
-3.8122
-2.93972

Red CANT_fc25_7

IW =

-1.06821	0.31012
-0.6799	-0.23095
-3.95892	-0.0187
0.648519	0.185196
0.657621	0.172032
0.111367	-1.33239
0.354279	-1.65625

LW =

0.677662	-0.35021	0.115015	0.35851	1.381431	-0.14118	-0.04726
0.677662	-0.35021	0.115015	0.35851	1.381431	-0.14118	-0.04726
0.677662	-0.35021	0.115015	0.35851	1.381431	-0.14118	-0.04726
1.103014	-1.15463	0.146276	-0.5816	1.737943	-0.31225	0.112439
0.976614	-1.24827	0.122924	-0.2977	1.524198	-0.07476	-0.03915
1.166859	-2.83333	0.087731	-0.51924	1.104637	0.004386	-0.03964

B_1 =

1.564578
1.463811
0.054451
1.18566
-0.2264
-1.1458
-0.86638

B_2 =

-0.43401
-0.43401
-0.43401
0.426361
0.461296
1.612485

Red CANT_fc30_6

IW =

0.28317	0.78926
-0.748	-0.72372
-0.47371	-0.09039
-0.04588	0.87608
-0.25804	-1.09483
-2.74538	0.072111

LW =

0.095461	-0.24448	-1.28287	1.490345	0.634797	-0.07305
0.095461	-0.24448	-1.28287	1.490345	0.634797	-0.07305
0.095461	-0.24448	-1.28287	1.490345	0.634797	-0.07305
-0.5102	-0.86053	-1.2264	1.751675	1.043301	-0.01522
-0.46681	-0.8164	-1.23044	1.732953	1.014035	-0.01936
-1.1523	-1.59315	-0.90178	1.512085	0.979995	0.00652

B_1 =

-1.76465
1.518663
-0.11474
1.215704
-1.46423
-2.24657

B_2 =

-0.56625
-0.56625
-0.56625
-0.65643
-0.64997
-0.6285

ANEXO 5.2: CÓDIGO INTERFAZ GRÁFICA

A continuación, se presenta la rutina asociada a la interfaz gráfica. A través de ella se selecciona la condición de apoyo, material base y se realiza el ingreso de datos iniciales, la interfaz realiza el proceso de cargar las redes, preproceso de los datos de entrada, postproceso de los resultados y visualización de estos.

```
classdef app1 < matlab.apps.AppBase


    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        CondiciondeapoyoButtonGroup     matlab.ui.container.ButtonGroup
        SimpleButton                    matlab.ui.control.RadioButton
        BiEmpotradaButton               matlab.ui.control.RadioButton
        CantileverButton                matlab.ui.control.RadioButton
        MaterialBaseButtonGroup         matlab.ui.container.ButtonGroup
        fc20MPaButton                   matlab.ui.control.RadioButton
        fc25MPaButton                   matlab.ui.control.RadioButton
        fc30MPaButton                   matlab.ui.control.RadioButton
        Image4                          matlab.ui.control.Image
        Image5                          matlab.ui.control.Image
        Image6                          matlab.ui.control.Image
        AceroA630420HLabel              matlab.ui.control.Label
        Panel                           matlab.ui.container.Panel
        Image2                          matlab.ui.control.Image
        Image                           matlab.ui.control.Image
        Label                           matlab.ui.control.Label
        Label_2                         matlab.ui.control.Label
        Panel_2                         matlab.ui.container.Panel
        LargodelavigamLabel             matlab.ui.control.Label
        LargodelavigamEditField         matlab.ui.control.EditField
        CargaMuertakNmLabel             matlab.ui.control.Label
        CargaMuertakNmEditField         matlab.ui.control.EditField
        CargaVivakNmLabel               matlab.ui.control.Label
        CargaVivakNmEditField           matlab.ui.control.EditField
        mLabel                          matlab.ui.control.Label
        kNmLabel_2                      matlab.ui.control.Label
        kNmLabel                        matlab.ui.control.Label
        EjecutarButton                  matlab.ui.control.Button
        Panel_3                         matlab.ui.container.Panel
        BEditFieldLabel                 matlab.ui.control.Label
        HLabel                          matlab.ui.control.Label
        BEditField                      matlab.ui.control.EditField
        HEditField                      matlab.ui.control.EditField
        AsEditField                     matlab.ui.control.EditField
        cmLabel                         matlab.ui.control.Label
        cmLabel_2                       matlab.ui.control.Label
        cm2Label                        matlab.ui.control.Label
```

```
        CostoEditFieldLabel          matlab.ui.control.Label
        CostoEditField               matlab.ui.control.EditField
        CLPcmLabel                   matlab.ui.control.Label
        As_vanoLabel                 matlab.ui.control.Label
        H2Label                      matlab.ui.control.Label
        H2EditField                  matlab.ui.control.EditField
        As_vanoEditField             matlab.ui.control.EditField
        F_mnEditField                matlab.ui.control.EditField
        kNmLabel_3                   matlab.ui.control.Label
        Mn_vanoLabel                 matlab.ui.control.Label
        Fmn_vanoEditField            matlab.ui.control.EditField
        As_comp_bordeEditField       matlab.ui.control.EditField
        cm2Label_2                   matlab.ui.control.Label
        As_vanoLabel_2               matlab.ui.control.Label
        As_comp_vanoEditField        matlab.ui.control.EditField
        cmLabel_3                    matlab.ui.control.Label
        cm2Label_3                   matlab.ui.control.Label
        cm2Label_4                   matlab.ui.control.Label
        kNmLabel_4                   matlab.ui.control.Label
        As_bordeLabel                matlab.ui.control.Label
        MnLabel                      matlab.ui.control.Label
        AsLabel                      matlab.ui.control.Label
    end


properties (Access = private)
% Description
end
methods (Access = private)
% function startupFcn(app)
% %function SeriesNetwork
% %app.net = load("SIMP_fc20_7.mat");
% %app.net = app.net.net;
% % ... %
% end
end


% Callbacks that handle component events
methods (Access = private)


% Button pushed function: EjecutarButton
function EjecutarButtonPushed(app, event)
if (app.MaterialBaseButtonGroup.SelectedObject == app.fc20MPaButton &&
app.CondiciondeapoyoButtonGroup.SelectedObject == app.SimpleButton)
L = str2double(app.LargodelavigamEditField.Value);
DL = str2double(app.CargaMuertakNmEditField.Value);
LL = str2double(app.CargaVivakNmEditField.Value);
L_norm=2*(L*100-300)/(700-300)-1; %aqui se pasa a cm
q=1.2*DL*10+1.6*LL*10; %aqui se pasa a N/cm
q_norm=2*(q-280)/(1680-280)-1;
% app.BEditField.Value = num2str(L);
```

```
% app.HEditField.Value = num2str(DL);
% app.AsEditField.Value = num2str(LL);
% app.BEditField.Value = num2str(L_norm);
% app.AsEditField.Value = num2str(q);
% app.HEditField.Value = num2str(q_norm);
%cargar red neuronal
x=([L_norm q_norm]');
%startupFcn(app)
load("SIMP_fc20_7.mat");
y = red(x);
%resultados red neuronal
B_norm=y(1);
H_norm=y(2);
As_norm=y(3);
C_norm=y(4);
Costo_norm=y(5);
f_Mn_norm=y(6);
%%post-proceso de des-normalización
B=(B_norm+1)*(31.944-9.932)/2+9.932; %sale en cm
H=(H_norm+1)*(99.833-33.796)/2+33.796; %sale en cm
As=(As_norm+1)*(39.496-3.818)/2+3.818; %sale en cm2
C=(C_norm+1)*(35.937-11.173)/2+11.173; %sale en cm
Costo=(Costo_norm+1)*(1314.886-217.098)/2+217.098; %sale en CLP/CM
f_Mn=((f_Mn_norm+1)*(120271383.799-
3614794.209)/2+3614794.209)/10^5; %sale en kN-m
app.BEditField.Value = num2str(B);
app.HEditField.Value = num2str(H);
app.AsEditField.Value = num2str(As);
%app.CEditField.Value = num2str(C);
app.CostoEditField.Value = num2str(Costo);
app.F_mnEditField.Value = num2str(f_Mn);
end
if (app.MaterialBaseButtonGroup.SelectedObject == app.fc25MPaButton &&
app.CondiciondeapoyoButtonGroup.SelectedObject == app.SimpleButton)
%%ESTAS ENTRARIAN DE LA GUI!!!!
L = str2double(app.LargodelavigamEditField.Value);
DL = str2double(app.CargaMuertakNmEditField.Value);
LL = str2double(app.CargaVivakNmEditField.Value);
%%normalizar variables
L_norm=2*(L*100-300)/(700-300)-1; %aqui se pasa a cm
q=1.2*DL*10+1.6*LL*10; %aqui se pasa a N/cm
q_norm=2*(q-280)/(1680-280)-1;
%%cargar red neuronal
load('SIMP_fc25_7');
x=([L_norm q_norm]');
y=red(x);
%resultados red neuronal
B_norm=y(1);
H_norm=y(2);
As_norm=y(3);
C_norm=y(4);
Costo_norm=y(5);
f_Mn_norm=y(6);
%%post-proceso de des-normalización
```

```
B=(B_norm+1)*(29.586-9.207)/2+9.207; %sale en cm
H=(H_norm+1)*(92.758-31.621)/2+31.621; %sale en cm
As=(As_norm+1)*(42.350-4.101)/2+4.101; %sale en cm2
C=(C_norm+1)*(33.284-10.358)/2+10.358; %sale en cm
Costo=(Costo_norm+1)*(1274.033-207.704)/2+207.704; %sale en CLP/CM
f_Mn=((f_Mn_norm+1)*(119443341.751-
3599567.945)/2+3599567.945)/10^5; %sale en kN-m
app.BEditField.Value = num2str(B);
app.HEditField.Value = num2str(H);
app.AsEditField.Value = num2str(As);
%app.CEditField.Value = num2str(C);
app.CostoEditField.Value = num2str(Costo);
app.F_mnEditField.Value = num2str(f_Mn);
end
if (app.MaterialBaseButtonGroup.SelectedObject == app.fc30MPaButton &&
app.CondiciondeapoyoButtonGroup.SelectedObject == app.SimpleButton)
%%ESTAS ENTRARIAN DE LA GUI!!!!
L = str2double(app.LargodelavigamEditField.Value);
DL = str2double(app.CargaMuertakNmEditField.Value);
LL = str2double(app.CargaVivakNmEditField.Value);
%%normalizar variables
L_norm=2*(L*100-300)/(700-300)-1; %aqui se pasa a cm
q=1.2*DL*10+1.6*LL*10; %aqui se pasa a N/cm
q_norm=2*(q-280)/(1680-280)-1;
%%cargar red neuronal
load('SIMP_fc30_7');
x=([L_norm q_norm]');
y=red(x);
%resultados red neuronal
B_norm=y(1);
H_norm=y(2);
As_norm=y(3);
C_norm=y(4);
Costo_norm=y(5);
f_Mn_norm=y(6);
%%post-proceso de des-normalización
B=(B_norm+1)*(27.957-8.705)/2+8.705; %sale en cm
H=(H_norm+1)*(87.872-30.116)/2+30.116; %sale en cm
As=(As_norm+1)*(44.617-4.326)/2+4.326; %sale en cm2
C=(C_norm+1)*(31.452-9.793)/2+9.793; %sale en cm
Costo=(Costo_norm+1)*(1269.803-203.700)/2+203.700; %sale en CLP/CM
f_Mn=((f_Mn_norm+1)*(118907687.815-
3589659.567)/2+3589659.567)/10^5; %sale en kN-m
app.BEditField.Value = num2str(B);
app.HEditField.Value = num2str(H);
app.AsEditField.Value = num2str(As);
%app.CEditField.Value = num2str(C);
app.CostoEditField.Value = num2str(Costo);
app.F_mnEditField.Value = num2str(f_Mn);
end
if(app.MaterialBaseButtonGroup.SelectedObject == app.fc20MPaButton &&
app.CondiciondeapoyoButtonGroup.SelectedObject == app.BiEmpotradaButton)
L = str2double(app.LargodelavigamEditField.Value);
DL = str2double(app.CargaMuertakNmEditField.Value);
```

```
LL = str2double(app.CargaVivakNmEditField.Value);
L_norm=2*(L*100-300)/(700-300)-1; %aqui se pasa a cm
q=1.2*DL*10+1.6*LL*10; %aqui se pasa a N/cm
q_norm=2*(q-280)/(1680-280)-1;
%%cargar red neuronal
load('EMP_fc20_7');
x=([L_norm q_norm]');
y=red(x);
%resultados red neuronal
B_norm=y(1);
H_norm=y(2);
As_borde_norm=y(3);
As_comp_borde_norm=y(4);
c_borde_norm=y(5);
fMn_borde_norm=y(6);
As_vano_norm=y(7);
As_comp_vano_norm=y(8);
c_vano_norm=y(9);
fMn_vano_norm=y(10);
Costo_norm=y(11);
%%post-proceso de des-normalización
B=(B_norm+1)*(25.5515-8.1234)/2+8.1234; %sale en cm sin decimal
H=(H_norm+1)*(80.6545-28.3702)/2+28.3702; %sale en cm sin decimal
As_borde=(As_borde_norm+1)*(31.5344-3.0840)/2+3.0840; %sale en cm2 1
decimal
As_comp_borde=(As_comp_borde_norm+1)*(6.5288-0.6599)/2+0.6599; %sale en
cm2 1 decimal
%c_borde=(c_borde_norm+1)*(28.7454-9.1388)/2+9.1388; %sale en CLP/CM sin
decimal
fMn_borde=((fMn_borde_norm+1)*(78755472.6930-
2385878.6647)/2+2385878.6647)/10^5; %sale en kN-m sin decimal
As_vano=(As_vano_norm+1)*(16.3220-1.6497)/2+1.6497; %1 decimal
As_comp_vano=(As_comp_vano_norm+1)*(6.5288-0.6599)/2+0.6599; %1 decimal
%c_vano=(c_vano_norm+1)*(11.5765-5.1876)/2+5.1876; %sin decimal
fMn_vano=((fMn_vano_norm+1)*(44471707.2281-
1368672.5528)/2+1368672.5528)/10^5; %sin decimal
Costo=(Costo_norm+1)*(2250.6519-326.9338)/2+326.9338; %sin decimal
app.AsLabel.Text = 'As_borde';
%app.CLabel.Text = 'c_borde';
app.MnLabel.Text = 'ΦMn_borde';
app.BEditField.Value = num2str(B);
app.HEditField.Value = num2str(H);
app.AsEditField.Value = num2str(As_borde);
app.As_vanoEditField.Value = num2str(As_vano);
app.As_comp_bordeEditField.Value = num2str(As_comp_borde);
app.As_comp_vanoEditField.Value = num2str(As_comp_vano);
%app.CEditField.Value = num2str(c_borde);
%app.c_vanoEditField.Value = num2str(c_vano);
app.CostoEditField.Value = num2str(Costo);
app.F_mnEditField.Value = num2str(fMn_borde);
app.Fmn_vanoEditField.Value = num2str(fMn_vano);
end
if(app.MaterialBaseButtonGroup.SelectedObject == app.fc25MPaButton &&
app.CondiciondeapoyoButtonGroup.SelectedObject == app.BiEmpotradaButton)
```

```
L = str2double(app.LargodelavigamEditField.Value);
DL = str2double(app.CargaMuertakNmEditField.Value);
LL = str2double(app.CargaVivakNmEditField.Value);
L_norm=2*(L*100-300)/(700-300)-1; %aqui se pasa a cm
q=1.2*DL*10+1.6*LL*10; %aqui se pasa a N/cm
q_norm=2*(q-280)/(1680-280)-1;
%%cargar red neuronal
load('EMP_fc25_6');
x=([L_norm q_norm]');
y=red(x);
%resultados red neuronal
B_norm=y(1);
H_norm=y(2);
As_borde_norm=y(3);
As_comp_borde_norm=y(4);
c_borde_norm=y(5);
fMn_borde_norm=y(6);
As_vano_norm=y(7);
As_comp_vano_norm=y(8);
c_vano_norm=y(9);
fMn_vano_norm=y(10);
Costo_norm=y(11);
%%post-proceso de des-normalización
B=(B_norm+1)*(24.0668-7.6502)/2+7.6502; %sale en cm sin decimal
H=(H_norm+1)*(76.2003-26.9506)/2+26.9506; %sale en cm sin decimal
As_borde=(As_borde_norm+1)*(33.5221-3.2718)/2+3.2718; %sale en cm2 1
decimal
As_comp_borde=(As_comp_borde_norm+1)*(5.7921-0.5853)/2+0.5853; %sale en
cm2 1 decimal
c_borde=(c_borde_norm+1)*(27.0751-8.6065)/2+8.6065; %sale en CLP/CM sin
decimal
fMn_borde=((fMn_borde_norm+1)*(78467190.6554-
2380341.7210)/2+2380341.7210)/10^5; %sale en kN-m sin decimal
As_vano=(As_vano_norm+1)*(17.3763-1.7558)/2+1.7558; %1 decimal
As_comp_vano=(As_comp_vano_norm+1)*(5.7921-0.5853)/2+0.5853; %1 decimal
c_vano=(c_vano_norm+1)*(11.5775-4.9253)/2+4.9253; %sin decimal
fMn_vano=((fMn_vano_norm+1)*(44378563.3442-
1374502.7468)/2+1374502.7468)/10^5; %sin decimal
Costo=(Costo_norm+1)*(2129.0323-310.5601)/2+310.5601; %sin decimal
app.AsLabel.Text = 'As_borde';
%app.CLabel.Text = 'c_borde';
app.MnLabel.Text = 'ΦMn_borde';
app.BEditField.Value = num2str(B);
app.HEditField.Value = num2str(H);
app.AsEditField.Value = num2str(As_borde);
app.As_vanoEditField.Value = num2str(As_vano);
app.As_comp_bordeEditField.Value = num2str(As_comp_borde);
app.As_comp_vanoEditField.Value = num2str(As_comp_vano);
%app.CEditField.Value = num2str(c_borde);
%app.c_vanoEditField.Value = num2str(c_vano);
app.CostoEditField.Value = num2str(Costo);
app.F_mnEditField.Value = num2str(fMn_borde);
app.Fmn_vanoEditField.Value = num2str(fMn_vano);
end
```

```
if(app.MaterialBaseButtonGroup.SelectedObject == app.fc30MPaButton &&
app.CondiciondeapoyoButtonGroup.SelectedObject == app.BiEmpotradaButton)
L = str2double(app.LargodelavigamEditField.Value);
DL = str2double(app.CargaMuertakNmEditField.Value);
LL = str2double(app.CargaVivakNmEditField.Value);
L_norm=2*(L*100-300)/(700-300)-1; %aqui se pasa a cm
q=1.2*DL*10+1.6*LL*10; %aqui se pasa a N/cm
q_norm=2*(q-280)/(1680-280)-1;
%%cargar red neuronal
load('EMP_fc30_7');
x=([L_norm q_norm]');
y=red(x);
%resultados red neuronal
B_norm=y(1);
H_norm=y(2);
As_borde_norm=y(3);
As_comp_borde_norm=y(4);
c_borde_norm=y(5);
fMn_borde_norm=y(6);
As_vano_norm=y(7);
As_comp_vano_norm=y(8);
c_vano_norm=y(9);
fMn_vano_norm=y(10);
Costo_norm=y(11);
B=(B_norm+1)*(22.6946-7.2974)/2+7.2974; %sale en cm sin decimal
H=(H_norm+1)*(72.0839-25.8921)/2+25.8921; %sale en cm sin decimal
As_borde=(As_borde_norm+1)*(34.2378-3.4170)/2+3.4170; %sale en cm2 1
decimal
As_comp_borde=(As_comp_borde_norm+1)*(5.1505-0.5325)/2+0.5325; %sale en
cm2 1 decimal
c_borde=(c_borde_norm+1)*(25.5315-8.2095)/2+8.2095; %sale en CLP/CM sin
decimal
fMn_borde=((fMn_borde_norm+1)*(75525922.5780-
2376412.5670)/2+2376412.5670)/10^5; %sale en kN-m sin decimal
As_vano=(As_vano_norm+1)*(18.0266-1.8638)/2+1.8638; %1 decimal
As_comp_vano=(As_comp_vano_norm+1)*(5.1505-0.5325)/2+0.5325; %1 decimal
c_vano=(c_vano_norm+1)*(11.5422-4.7720)/2+4.7720; %sin decimal
fMn_vano=((fMn_vano_norm+1)*(43253812.4415-
1394507.6724)/2+1394507.6724)/10^5; %sin decimal
Costo=(Costo_norm+1)*(2059.9638-305.7433)/2+305.7433; %sin decimal
app.AsLabel.Text = 'As_borde';
%app.CLabel.Text = 'c_borde';
app.MnLabel.Text = 'ΦMn_borde';
app.BEditField.Value = num2str(B);
app.HEditField.Value = num2str(H);
app.AsEditField.Value = num2str(As_borde);
app.As_vanoEditField.Value = num2str(As_vano);
app.As_comp_bordeEditField.Value = num2str(As_comp_borde);
app.As_comp_vanoEditField.Value = num2str(As_comp_vano);
%app.CEditField.Value = num2str(c_borde);
%app.c_vanoEditField.Value = num2str(c_vano);
app.CostoEditField.Value = num2str(Costo);
app.F_mnEditField.Value = num2str(fMn_borde);
app.Fmn_vanoEditField.Value = num2str(fMn_vano);
```

```
end
if(app.MaterialBaseButtonGroup.SelectedObject == app.fc20MPaButton &&
app.CondiciondeapoyoButtonGroup.SelectedObject == app.CantileverButton)
L = str2double(app.LargodelavigamEditField.Value);
DL = str2double(app.CargaMuertakNmEditField.Value);
LL = str2double(app.CargaVivakNmEditField.Value);
L_norm=2*(L*100-100)/(300-100)-1; %aqui se pasa a cm
q=1.2*DL*10+1.6*LL*10; %aqui se pasa a N/cm
q_norm=2*(q-280)/(1680-280)-1;
%%cargar red neuronal
load('CANT_fc20_7');
x=([L_norm q_norm]');
y=red(x);
%resultados red neuronal
B_norm=y(1);
H1_norm=y(2);
H2_norm=y(3);
As_norm=y(4);
Costo_norm=y(5);
f_Mn_norm=y(6);
%%post-proceso de des-normalización
B=(B_norm+1)*(47.558-12.493)/2+12.493; %sale en cm
H1=(H1_norm+1)*(75.336-22.740)/2+22.740; %sale en cm
H2=(H2_norm+1)*(47.558-12.493)/2+12.493; %sale en cm2
As=(As_norm+1)*(43.770-3.020)/2+3.020; %sale en cm
Costo=(Costo_norm+1)*(3273.359-295.058)/2+295.058; %sale en CLP/CM
f_Mn=((f_Mn_norm+1)*(99216679.012-1798578.650)/2+1798578.650)/10^5; %sale
en kN-m
app.HLabel.Text = 'H1';
app.BEditField.Value = num2str(B);
app.HEditField.Value = num2str(H1);
app.H2EditField.Value = num2str(H2);
app.AsEditField.Value = num2str(As);
%app.CEditField.Value = num2str(C);
app.CostoEditField.Value = num2str(Costo);
app.F_mnEditField.Value = num2str(f_Mn);
end
if(app.MaterialBaseButtonGroup.SelectedObject == app.fc25MPaButton &&
app.CondiciondeapoyoButtonGroup.SelectedObject == app.CantileverButton)
L = str2double(app.LargodelavigamEditField.Value);
DL = str2double(app.CargaMuertakNmEditField.Value);
LL = str2double(app.CargaVivakNmEditField.Value);
L_norm=2*(L*100-100)/(300-100)-1; %aqui se pasa a cm
q=1.2*DL*10+1.6*LL*10; %aqui se pasa a N/cm
q_norm=2*(q-280)/(1680-280)-1;
%%cargar red neuronal
load('CANT_fc25_7');
x=([L_norm q_norm]');
y=red(x);
%resultados red neuronal
B_norm=y(1);
H1_norm=y(2);
H2_norm=y(3);
As_norm=y(4);
```

```
Costo_norm=y(5);
f_Mn_norm=y(6);
%%post-proceso de des-normalización
B=(B_norm+1)*(44.037-11.584)/2+11.584; %sale en cm
H1=(H1_norm+1)*(70.056-21.376)/2+21.376; %sale en cm
H2=(H2_norm+1)*(44.037-11.584)/2+11.584; %sale en cm2
As=(As_norm+1)*(46.913-3.246)/2+3.246; %sale en cm
Costo=(Costo_norm+1)*(3010.966-273.305)/2+273.305; %sale en CLP/CM
f_Mn=((f_Mn_norm+1)*(98468372.506-1792343.487)/2+1792343.487)/10^5; %sale
en kN-m
app.HLabel.Text = 'H1';
app.BEditField.Value = num2str(B);
app.HEditField.Value = num2str(H1);
app.H2EditField.Value = num2str(H2);
app.AsEditField.Value = num2str(As);
%app.CEditField.Value = num2str(C);
app.CostoEditField.Value = num2str(Costo);
app.F_mnEditField.Value = num2str(f_Mn);
end
if(app.MaterialBaseButtonGroup.SelectedObject == app.fc30MPaButton &&
app.CondiciondeapoyoButtonGroup.SelectedObject == app.CantileverButton)
L = str2double(app.LargodelavigamEditField.Value);
DL = str2double(app.CargaMuertakNmEditField.Value);
LL = str2double(app.CargaVivakNmEditField.Value);
L_norm=2*(L*100-100)/(300-100)-1; %aqui se pasa a cm
q=1.2*DL*10+1.6*LL*10; %aqui se pasa a N/cm
q_norm=2*(q-280)/(1680-280)-1;
%%cargar red neuronal
load('CANT_fc30_6');
x=([L_norm q_norm]');
y=red(x);
%resultados red neuronal
B_norm=y(1);
H1_norm=y(2);
H2_norm=y(3);
As_norm=y(4);
Costo_norm=y(5);
f_Mn_norm=y(6);
%%post-proceso de des-normalización
B=(B_norm+1)*(41.559-10.943)/2+10.943; %sale en cm
H1=(H1_norm+1)*(66.339-20.414)/2+20.414; %sale en cm
H2=(H2_norm+1)*(41.559-10.943)/2+10.943; %sale en cm
As=(As_norm+1)*(49.296-3.418)/2+3.418; %sale en cm2
Costo=(Costo_norm+1)*(2935.095-265.473)/2+265.473; %sale en CLP/CM
f_Mn=((f_Mn_norm+1)*(97958654.238-1788199.024)/2+1788199.024)/10^5; %sale
en kN-m
app.HLabel.Text = 'H1';
app.BEditField.Value = num2str(B);
app.HEditField.Value = num2str(H1);
app.H2EditField.Value = num2str(H2);
app.AsEditField.Value = num2str(As);
%app.CEditField.Value = num2str(C);
app.CostoEditField.Value = num2str(Costo);
app.F_mnEditField.Value = num2str(f_Mn);
```

```
end
end


% Selection changed function: CondiciondeapoyoButtonGroup
function CondiciondeapoyoButtonGroupSelectionChanged(app, event)
selectedButton = app.CondiciondeapoyoButtonGroup.SelectedObject;
% if selectedButton == app.SimpleButton
% app.Image3.Visible = 1;
% else
% end
% if selectedButton == app.BiEmpotradaButton
% app.Image4.Visible = 1;
% else
% app.Image4.Visible = 0;
% end
% if selectedButton == app.CantileverButton
% app.Image5.Visible = 1;
% else
% app.Image5.Visible = 0;
% end
if app.CondiciondeapoyoButtonGroup.SelectedObject == app.SimpleButton
app.HLabel.Text = 'H';
app.AsLabel.Text = 'As';
app.MnLabel.Text = 'ΦMn';
app.mLabel.Text = '[3 - 7] m';
app.As_comp_bordeEditField.Visible = 0;
app.As_comp_vanoEditField.Visible = 0;
app.H2EditField.Visible = 0;
app.As_vanoEditField.Visible = 0;
app.Fmn_vanoEditField.Visible = 0;
app.Image5.Visible = 0;
app.Image4.Visible = 0;
app.Image6.Visible = 1;
app.H2Label.Visible = 0;
app.As_vanoLabel.Visible = 0;
app.Mn_vanoLabel.Visible = 0;
app.As_vanoLabel_2.Visible = 0;
app.As_bordeLabel.Visible = 0;
app.cm2Label_2.Visible = 0;
app.cm2Label_3.Visible = 0;
app.cm2Label_4.Visible = 0;
app.cmLabel_3.Visible = 0;
app.kNmLabel_4.Visible = 0;
elseif selectedButton == app.BiEmpotradaButton
app.HLabel.Text = 'H';
app.AsLabel.Text = 'As_borde';
app.MnLabel.Text = 'ΦMn_borde';
app.mLabel.Text = '[3 - 7] m';
app.As_comp_bordeEditField.Visible = 1;
app.As_comp_vanoEditField.Visible = 1;
app.H2EditField.Visible = 0;
app.As_vanoEditField.Visible = 1;
app.Fmn_vanoEditField.Visible = 1;
```

```
app.Image5.Visible = 0;
app.Image4.Visible = 1;
app.Image6.Visible = 0;
app.H2Label.Visible = 1;
app.As_vanoLabel.Visible = 1;
app.Mn_vanoLabel.Visible = 1;
app.As_vanoLabel_2.Visible = 1;
app.As_bordeLabel.Visible = 1;
app.cm2Label_2.Visible = 1;
app.cm2Label_3.Visible = 1;
app.cm2Label_4.Visible = 1;
app.cmLabel_3.Visible = 0;
app.kNmLabel_4.Visible = 1;
app.H2Label.Visible = 0;
elseif selectedButton == app.CantileverButton
app.AsLabel.Text = 'As';
app.MnLabel.Text = 'ΦMn';
app.mLabel.Text = '[1 - 3] m';
app.Image5.Visible = 1;
app.Image4.Visible = 0;
app.Image6.Visible = 0;
app.As_comp_bordeEditField.Visible = 0;
app.As_comp_vanoEditField.Visible = 0;
app.H2EditField.Visible = 1;
app.As_vanoEditField.Visible = 0;
app.Fmn_vanoEditField.Visible = 0;
app.H2Label.Visible = 1;
app.As_vanoLabel.Visible = 0;
app.Mn_vanoLabel.Visible = 0;
app.As_vanoLabel_2.Visible = 0;
app.As_bordeLabel.Visible = 0;
app.cm2Label_2.Visible = 0;
app.cm2Label_3.Visible = 0;
app.cm2Label_4.Visible = 0;
app.cmLabel_3.Visible = 1;
app.kNmLabel_4.Visible = 0;
app.HLabel.Text = 'H1';
elseif selectedButton == (app.SimpleButton & app.BiEmpotradaButton &
app.CantileverButton)
app.Image6.Visible = 1;
app.Image4.Visible = 0;
app.Image5.Visible = 0;
end
end


% Image clicked function: Image4
function Image4Clicked(app, event)
end


% Selection changed function: MaterialBaseButtonGroup
function MaterialBaseButtonGroupSelectionChanged(app, event)
selectedButton = app.MaterialBaseButtonGroup.SelectedObject;
```

```
end


% Callback function
function ResetButtonPushed(app, event)
clear
end
end


% Component initialization
methods (Access = private)


% Create UIFigure and components
function createComponents(app)


% Create UIFigure and hide until all components are created
app.UIFigure = uifigure('Visible', 'off');
app.UIFigure.Color = [0.8902 0.902 0.8588];
app.UIFigure.Position = [100 100 906 524];
app.UIFigure.Name = 'MATLAB App';


% Create CondiciondeapoyoButtonGroup
app.CondiciondeapoyoButtonGroup = uibuttongroup(app.UIFigure);
app.CondiciondeapoyoButtonGroup.SelectionChangedFcn =
createCallbackFcn(app, @CondiciondeapoyoButtonGroupSelectionChanged,
true);
app.CondiciondeapoyoButtonGroup.Title = 'Condicion de apoyo';
app.CondiciondeapoyoButtonGroup.FontWeight = 'bold';
app.CondiciondeapoyoButtonGroup.Position = [85 288 123 106];


% Create SimpleButton
app.SimpleButton = uiradiobutton(app.CondiciondeapoyoButtonGroup);
app.SimpleButton.Text = 'Simple';
app.SimpleButton.Position = [11 60 59 22];
app.SimpleButton.Value = true;


% Create BiEmpotradaButton
app.BiEmpotradaButton = uiradiobutton(app.CondiciondeapoyoButtonGroup);
app.BiEmpotradaButton.Text = 'Bi-Empotrada';
app.BiEmpotradaButton.Position = [11 38 95 22];


% Create CantileverButton
app.CantileverButton = uiradiobutton(app.CondiciondeapoyoButtonGroup);
app.CantileverButton.Text = 'Cantilever';
app.CantileverButton.Position = [11 16 76 22];
```

```
% Create MaterialBaseButtonGroup
app.MaterialBaseButtonGroup = uibuttongroup(app.UIFigure);
app.MaterialBaseButtonGroup.SelectionChangedFcn = createCallbackFcn(app,
@MaterialBaseButtonGroupSelectionChanged, true);
app.MaterialBaseButtonGroup.Title = 'Material Base';
app.MaterialBaseButtonGroup.FontWeight = 'bold';
app.MaterialBaseButtonGroup.Position = [236 288 123 106];


% Create fc20MPaButton
app.fc20MPaButton = uiradiobutton(app.MaterialBaseButtonGroup);
app.fc20MPaButton.Text = 'f''c 20MPa';
app.fc20MPaButton.Position = [11 60 75 22];
app.fc20MPaButton.Value = true;


% Create fc25MPaButton
app.fc25MPaButton = uiradiobutton(app.MaterialBaseButtonGroup);
app.fc25MPaButton.Text = 'f''c 25MPa';
app.fc25MPaButton.Position = [11 38 75 22];


% Create fc30MPaButton
app.fc30MPaButton = uiradiobutton(app.MaterialBaseButtonGroup);
app.fc30MPaButton.Text = 'f''c 30MPa';
app.fc30MPaButton.Position = [11 16 75 22];


% Create Image4
app.Image4 = uiimage(app.UIFigure);
app.Image4.ScaleMethod = 'fill';
app.Image4.ImageClickedFcn = createCallbackFcn(app, @Image4Clicked,
true);
app.Image4.Position = [434 278 463 124];
app.Image4.ImageSource = 'viga_emp1.png';


% Create Image5
app.Image5 = uiimage(app.UIFigure);
app.Image5.Position = [434 257 453 161];
app.Image5.ImageSource = 'viga_cant1.png';


% Create Image6
app.Image6 = uiimage(app.UIFigure);
app.Image6.ScaleMethod = 'fill';
app.Image6.Position = [429 265 453 145];
app.Image6.ImageSource = 'viga_simp1.png';


% Create AceroA630420HLabel
app.AceroA630420HLabel = uilabel(app.UIFigure);
app.AceroA630420HLabel.BackgroundColor = [0.9412 0.9412 0.9412];
app.AceroA630420HLabel.HorizontalAlignment = 'center';
```

```
app.AceroA630420HLabel.FontWeight = 'bold';
app.AceroA630420HLabel.Position = [236 257 123 22];
app.AceroA630420HLabel.Text = 'Acero A630-420H';


% Create Panel
app.Panel = uipanel(app.UIFigure);
app.Panel.BackgroundColor = [0.8118 0.9255 1];
app.Panel.Position = [1 425 906 100];


% Create Image2
app.Image2 = uiimage(app.Panel);
app.Image2.Position = [21 14 76 73];
app.Image2.ImageSource = 'logo_FI.png';


% Create Image
app.Image = uiimage(app.Panel);
app.Image.Position = [801 8 76 73];
app.Image.ImageSource = 'logo_udec.png';


% Create Label
app.Label = uilabel(app.Panel);
app.Label.HorizontalAlignment = 'center';
app.Label.FontSize = 20;
app.Label.FontWeight = 'bold';
app.Label.Position = [189 30 535 61];
app.Label.Text = {'DISEÑO ÓPTIMO DE VIGAS DE HORMIGÓN ARMADO'; ' DE
ACUERDO A ACI 318-19 BASADO EN RNA.'};


% Create Label_2
app.Label_2 = uilabel(app.Panel);
app.Label_2.HorizontalAlignment = 'center';
app.Label_2.FontSize = 14;
app.Label_2.Position = [189 1 535 30];
app.Label_2.Text = 'Desarrollado por: Sebastián Parra Vergara';


% Create Panel_2
app.Panel_2 = uipanel(app.UIFigure);
app.Panel_2.BackgroundColor = [0.9882 0.9882 0.851];
app.Panel_2.Position = [1 1 392 245];


% Create LargodelavigamLabel
app.LargodelavigamLabel = uilabel(app.Panel_2);
app.LargodelavigamLabel.HorizontalAlignment = 'right';
app.LargodelavigamLabel.Position = [7 192 112 22];
app.LargodelavigamLabel.Text = 'Largo de la viga (m)';
```

```
% Create LargodelavigamEditField
app.LargodelavigamEditField = uieditfield(app.Panel_2, 'text');
app.LargodelavigamEditField.Position = [139 190 101 22];


% Create CargaMuertakNmLabel
app.CargaMuertakNmLabel = uilabel(app.Panel_2);
app.CargaMuertakNmLabel.HorizontalAlignment = 'right';
app.CargaMuertakNmLabel.Position = [8 158 118 22];
app.CargaMuertakNmLabel.Text = 'Carga Muerta (kN/m)';


% Create CargaMuertakNmEditField
app.CargaMuertakNmEditField = uieditfield(app.Panel_2, 'text');
app.CargaMuertakNmEditField.Position = [138 157 102 22];


% Create CargaVivakNmLabel
app.CargaVivakNmLabel = uilabel(app.Panel_2);
app.CargaVivakNmLabel.HorizontalAlignment = 'right';
app.CargaVivakNmLabel.Position = [14 125 104 22];
app.CargaVivakNmLabel.Text = 'Carga Viva (kN/m)';


% Create CargaVivakNmEditField
app.CargaVivakNmEditField = uieditfield(app.Panel_2, 'text');
app.CargaVivakNmEditField.Position = [139 125 101 22];


% Create mLabel
app.mLabel = uilabel(app.Panel_2);
app.mLabel.Position = [307 190 50 22];
app.mLabel.Text = '[3 - 7] m';


% Create kNmLabel_2
app.kNmLabel_2 = uilabel(app.Panel_2);
app.kNmLabel_2.Position = [303 157 81 22];
app.kNmLabel_2.Text = '[10 - 60] kN/m';


% Create kNmLabel
app.kNmLabel = uilabel(app.Panel_2);
app.kNmLabel.Position = [303 125 81 22];
app.kNmLabel.Text = '[10 - 60] kN/m';


% Create EjecutarButton
app.EjecutarButton = uibutton(app.Panel_2, 'push');
app.EjecutarButton.ButtonPushedFcn = createCallbackFcn(app,
@EjecutarButtonPushed, true);
app.EjecutarButton.FontName = 'Arial';
app.EjecutarButton.FontSize = 18;
app.EjecutarButton.FontWeight = 'bold';
```

```
app.EjecutarButton.Position = [83 25 245 54];
app.EjecutarButton.Text = 'Ejecutar';


% Create Panel_3
app.Panel_3 = uipanel(app.UIFigure);
app.Panel_3.BackgroundColor = [0.8314 1 0.6588];
app.Panel_3.Position = [394 1 513 245];


% Create BEditFieldLabel
app.BEditFieldLabel = uilabel(app.Panel_3);
app.BEditFieldLabel.HorizontalAlignment = 'right';
app.BEditFieldLabel.Position = [32 196 25 22];
app.BEditFieldLabel.Text = 'B';


% Create HLabel
app.HLabel = uilabel(app.Panel_3);
app.HLabel.HorizontalAlignment = 'right';
app.HLabel.Position = [6 163 51 22];
app.HLabel.Text = 'H';


% Create BEditField
app.BEditField = uieditfield(app.Panel_3, 'text');
app.BEditField.Position = [72 196 100 22];


% Create HEditField
app.HEditField = uieditfield(app.Panel_3, 'text');
app.HEditField.Position = [72 161 100 22];


% Create AsEditField
app.AsEditField = uieditfield(app.Panel_3, 'text');
app.AsEditField.Position = [72 130 100 22];


% Create cmLabel
app.cmLabel = uilabel(app.Panel_3);
app.cmLabel.Position = [189 196 28 22];
app.cmLabel.Text = '[cm]';


% Create cmLabel_2
app.cmLabel_2 = uilabel(app.Panel_3);
app.cmLabel_2.Position = [189 161 28 22];
app.cmLabel_2.Text = '[cm]';


% Create cm2Label
app.cm2Label = uilabel(app.Panel_3);
app.cm2Label.Position = [189 128 35 22];
```

```
app.cm2Label.Text = '[cm2]';


% Create CostoEditFieldLabel
app.CostoEditFieldLabel = uilabel(app.Panel_3);
app.CostoEditFieldLabel.HorizontalAlignment = 'right';
app.CostoEditFieldLabel.Position = [27 95 37 22];
app.CostoEditFieldLabel.Text = 'Costo';


% Create CostoEditField
app.CostoEditField = uieditfield(app.Panel_3, 'text');
app.CostoEditField.Position = [72 95 100 22];


% Create CLPcmLabel
app.CLPcmLabel = uilabel(app.Panel_3);
app.CLPcmLabel.Position = [189 95 55 22];
app.CLPcmLabel.Text = '[CLP/cm]';


% Create As_vanoLabel
app.As_vanoLabel = uilabel(app.Panel_3);
app.As_vanoLabel.HorizontalAlignment = 'right';
app.As_vanoLabel.Position = [268 126 52 22];
app.As_vanoLabel.Text = 'As_vano';


% Create H2Label
app.H2Label = uilabel(app.Panel_3);
app.H2Label.HorizontalAlignment = 'right';
app.H2Label.Position = [295 161 25 22];
app.H2Label.Text = 'H2';


% Create H2EditField
app.H2EditField = uieditfield(app.Panel_3, 'text');
app.H2EditField.Position = [330 161 100 22];


% Create As_vanoEditField
app.As_vanoEditField = uieditfield(app.Panel_3, 'text');
app.As_vanoEditField.Position = [330 128 100 22];


% Create F_mnEditField
app.F_mnEditField = uieditfield(app.Panel_3, 'text');
app.F_mnEditField.Position = [72 59 100 22];


% Create kNmLabel_3
app.kNmLabel_3 = uilabel(app.Panel_3);
app.kNmLabel_3.Position = [189 59 41 22];
app.kNmLabel_3.Text = '[kN-m]';
```

```
% Create Mn_vanoLabel
app.Mn_vanoLabel = uilabel(app.Panel_3);
app.Mn_vanoLabel.HorizontalAlignment = 'right';
app.Mn_vanoLabel.Position = [256 61 64 22];
app.Mn_vanoLabel.Text = 'Φ Mn_vano';


% Create Fmn_vanoEditField
app.Fmn_vanoEditField = uieditfield(app.Panel_3, 'text');
app.Fmn_vanoEditField.Position = [330 61 100 22];


% Create As_comp_bordeEditField
app.As_comp_bordeEditField = uieditfield(app.Panel_3, 'text');
app.As_comp_bordeEditField.Position = [72 25 100 22];


% Create cm2Label_2
app.cm2Label_2 = uilabel(app.Panel_3);
app.cm2Label_2.Position = [189 27 35 22];
app.cm2Label_2.Text = '[cm2]';


% Create As_vanoLabel_2
app.As_vanoLabel_2 = uilabel(app.Panel_3);
app.As_vanoLabel_2.HorizontalAlignment = 'right';
app.As_vanoLabel_2.FontSize = 10;
app.As_vanoLabel_2.Position = [250 29 70 22];
app.As_vanoLabel_2.Text = 'As''_vano';


% Create As_comp_vanoEditField
app.As_comp_vanoEditField = uieditfield(app.Panel_3, 'text');
app.As_comp_vanoEditField.Position = [330 27 100 22];


% Create cmLabel_3
app.cmLabel_3 = uilabel(app.Panel_3);
app.cmLabel_3.Position = [447 164 28 22];
app.cmLabel_3.Text = '[cm]';


% Create cm2Label_3
app.cm2Label_3 = uilabel(app.Panel_3);
app.cm2Label_3.Position = [447 130 35 22];
app.cm2Label_3.Text = '[cm2]';


% Create cm2Label_4
app.cm2Label_4 = uilabel(app.Panel_3);
app.cm2Label_4.Position = [447 29 35 22];
app.cm2Label_4.Text = '[cm2]';
```

```
% Create kNmLabel_4
app.kNmLabel_4 = uilabel(app.Panel_3);
app.kNmLabel_4.Position = [447 61 41 22];
app.kNmLabel_4.Text = '[kN-m]';


% Create As_bordeLabel
app.As_bordeLabel = uilabel(app.Panel_3);
app.As_bordeLabel.HorizontalAlignment = 'right';
app.As_bordeLabel.FontSize = 10;
app.As_bordeLabel.Position = [-15 25 77 22];
app.As_bordeLabel.Text = 'As''_borde';


% Create MnLabel
app.MnLabel = uilabel(app.Panel_3);
app.MnLabel.HorizontalAlignment = 'right';
app.MnLabel.Position = [-30 59 94 22];
app.MnLabel.Text = 'ΦMn';


% Create AsLabel
app.AsLabel = uilabel(app.Panel_3);
app.AsLabel.HorizontalAlignment = 'right';
app.AsLabel.Position = [-10 130 67 22];
app.AsLabel.Text = 'As';


% Show the figure after all components are created
app.UIFigure.Visible = 'on';
end
end


% App creation and deletion
methods (Access = public)


% Construct app
function app = app1


% Create UIFigure and components
createComponents(app)


% Register the app with App Designer
registerApp(app, app.UIFigure)


if nargout == 0
clear app
```

```
end
end


% Code that executes before app deletion
function delete(app)


% Delete UIFigure when app is deleted
delete(app.UIFigure)
end
end
end
```

ANEXO 5.3: DESARROLLO DE UNA RNA A PARTIR DE UNA BASE DE DATOS EXPERIMENTAL

Introducción

Una de las aplicaciones de RNA se encuentra en la predicción de valores numéricos mediante la regresión, como se detalló en el tercer capítulo. En este capítulo, se ilustrará una aplicación específica del algoritmo de Redes Neuronales Artificiales en la estimación de la resistencia a la compresión axial de probetas de hormigón. Este análisis se basa en una base de datos experimental recopilada de los trabajos de Chopra y colaboradores (2018).

Resistencia a la compresión axial de probetas de hormigón

El hormigón es un material ampliamente utilizado en la construcción debido a sus diversas propiedades esenciales, como su facilidad de transporte y manipulación, la capacidad de proporcionar resistencia a la compresión a los elementos estructurales, contribuir a la rigidez de la estructura y ofrecer aislamiento térmico y acústico en edificaciones, entre otras. En el ámbito de la Ingeniería Estructural, la propiedad que cobra mayor relevancia durante la fase de diseño es la resistencia a la compresión, ya que está directamente vinculada a la fortaleza de un elemento estructural.

Poder predecir la resistencia a la compresión del hormigón es de vital importancia para poder construir estructuras seguras, disminuyendo así la variabilidad y el riesgo asociados a las discrepancias entre los valores usados en el diseño y en la construcción (Chopra, 2018).

La composición típica de la mezcla de hormigón es cemento (c), agua (w), agregado fino (s) y agregado grueso (ca), la distribución de estos materiales determina la resistencia a la compresión axial del hormigón, también denominada como f’c para una edad de N días.

El ensayo para obtener la resistencia a la compresión de una probeta es de carácter destructivo, esto significa que la probeta ensayada no es reutilizable y solo contribuye en un aumento de contaminación por la generación de residuos no degradables, debido a esto, es importante definir un modelo capaz de predecir la resistencia del hormigón para una edad definida.

Recopilación de datos

La base de datos utilizada en el caso de estudio (Chopra et al., 2018) es de naturaleza experimental y abarca la resistencia a la compresión axial de probetas de hormigón a tres edades distintas (28, 56 y 91 días) y para diversas dosificaciones. Las variables que influyen en la determinación de esta resistencia son las proporciones de dosificación utilizadas (cemento:arena:grava), la relación agua/cemento y la densidad del cemento (kg/m3).

La dosificación se encuentra normalizada con respecto al contenido de cemento, es decir (cemento: arena: grava) corresponde a (1: arena: grava). De esta forma las variables de entrada del problema pueden ser caracterizadas como: Razón agua/cemento (w/c), dosificación arena/cemento (A/c), dosificación grava/cemento (G/c) y la densidad del cemento (C) y las variables de salida corresponden a las resistencias para la edad de 28, 56 y 91 días (f'c28) (f'c56) y (f'c91).

Tabla A.5.3.1 Base de datos

ID	w/c	A/c	G/c	C (kg/m3)	f'c 28días (MPa)	f'c 56días (MPa)	f'c 91días (MPa)
1	0.53	1.58	3.048	375	36.84	40.92	44.52
2	0.5	1.43	2.824	400	43.13	50.22	51.97
3	0.53	1.54	2.993	400	38.58	45.51	47.49
4	0.47	1.27	2.575	425	47.16	51.25	54.27
5	0.49	1.39	2.773	425	45.05	50.72	52.85
6	0.44	1.14	2.352	450	49.63	54.48	58.04
7	0.47	1.25	2.541	450	47.42	51.34	55.3
8	0.42	1.05	2.194	475	54.01	57.91	60.15
9	0.44	1.19	2.463	475	50.05	55.72	58.31
10	0.53	1.58	3.048	375	37.81	43.5	47.55
11	0.5	1.43	2.824	400	44.11	50.98	52.56
12	0.53	1.54	2.993	400	40.9	46.56	51.07

13	0.47	1.27	2.575	425	47.51	52.92	54.47
14	0.49	1.397	2.77	425	45.3	51.47	53.09
15	0.51	1.51	2.946	425	42.54	49.05	51.19
16	0.44	1.14	2.352	450	52.03	56.26	59.19
17	0.47	1.25	2.541	450	48.74	53.42	55.03
18	0.49	1.37	2.73	450	46.59	53.21	53.67
19	0.42	1.05	2.193	475	54.49	58.65	63.07
20	0.44	1.19	2.462	475	53.06	56.67	62.57
21	0.46	1.23	2.51	475	49.18	54.04	57.1
22	0.52	1.426	2.02	425	40.02	46.92	48.48
23	0.49	1.288	1.86	450	45.25	50.43	53.09
24	0.51	0.393	1.98	450	42.68	48.54	49.63
25	0.46	1.167	1.71	475	48.67	53.48	56.5
26	0.48	1.264	1.83	475	45.52	50.97	53.63
27	0.51	1.392	3.25	350	39.52	43.31	46.13
28	0.54	1.497	3.42	350	31.66	37.18	43.92
29	0.48	1.245	2.98	375	42.73	48.23	52.23
30	0.51	1.354	3.18	375	40.69	44.46	46.42
31	0.45	1.1	2.703	400	47.99	52.95	55.51
32	0.48	1.21	2.915	400	44.89	51.2	53.85
33	0.42	0.98	2.468	425	51.25	57.55	59.5
34	0.45	1.08	2.677	425	49.05	54.14	57.35
35	0.42	0.97	2.454	450	53.69	57.77	59.89
36	0.54	1.49	3.424	350	36.64	43.46	46.55
37	0.51	1.35	3.185	375	41.57	46.81	50.04
38	0.48	1.21	2.915	400	46.22	52.58	53.07
39	0.45	1.09	2.677	425	50.35	56.02	58.32
40	0.42	0.98	2.454	450	54.11	58.52	62.28
41	0.53	1.43	2.412	375	37.3	43.51	46.63
42	0.5	1.32	2.211	400	44.04	50.53	52.55
43	0.53	1.44	2.364	400	39.61	46.09	48.17
44	0.47	1.19	2.032	425	47.37	51.31	54.77
45	0.49	1.29	2.175	425	44.69	50.69	52.75
46	0.44	1.07	1.856	450	50.93	55.71	59.05
47	0.47	1.17	2.003	450	48.08	52.63	55.61
48	0.42	0.95	1.681	475	54.14	58.21	61.11
49	0.44	1.06	1.841	475	51.31	56.37	59.51

Tabla A.5.3.2 Rango de valores de los parámetros.

w/c	A/c	G/c	C (kg/m3)
0.42 - 0.54	0.39 - 1.58	1.68 - 3.24	350 - 475

Arquitectura de la Red Neuronal Artificial

Previo a la construcción de la R.N.A. es fundamental definir el preproceso de la base de datos. Se identifican cuatro variables de entrada y tres variables de salida que conformarán la red (ver Figura A.5.3.1). Además, podemos notar que estas variables no poseen ordenes de magnitud similares en sus rangos de valores, debido a esto, se propone una normalización de datos en el rango de [-1,1].

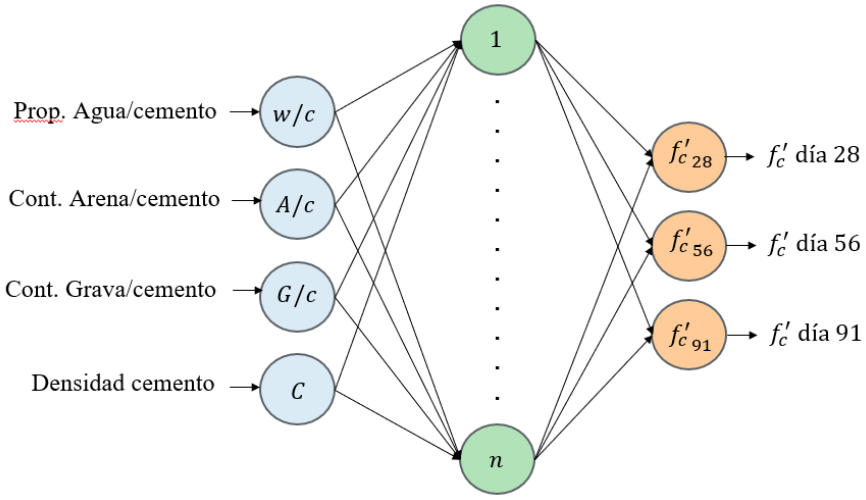

Figura A.5.3.1 Arquitectura de la RNA con variables de entrada y salida, y capas de neuronas.

Se desarrollan siete Redes Neuronales Artificiales para determinar la arquitectura modificando la cantidad de neuronas en la capa oculta hasta determinar la arquitectura que mejor se ajusta al problema. Estas redes de identifican mediante la nomenclatura:

PROB_fc_A

PROB_fc: Red encargada de determinar el f'c de una probeta.
A: Cantidad de neuronas en la capa oculta

La partición de datos elegida corresponde a 70% de los datos para entrenamiento, 15% para validación y 15% para prueba. La base de datos posee 49 muestras con tres resultados, es decir, la base de datos posee 147 valores de resistencia a la compresión medidos, así el conjunto de datos de entramiento corresponde a 105 datos de resistencia a la compresión del hormigón, 21 datos para validación y 21 datos para prueba.

La red neuronal artificial se construye, entrena y valida en MATLAB® R2020a como una red hacia adelante (feedforward) perceptrón-multicapa y el algoritmo de entrenamiento Levenberg-Marquardt backpropagation. La función de activación asignada para las neuronas de la capa oculta corresponde a la tangente hiperbólica y para el caso de las neuronas de la capa de salida se designa la función lineal Purelin. El vector de entrada posee 4 valores y el vector de salida posee tres valores (ver Figura A.5.3.2)

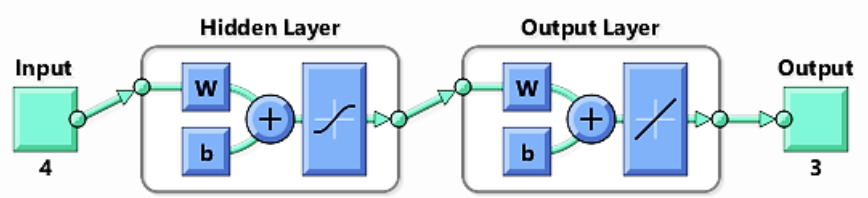

Figura A.5.3.2 Esquema de red neuronal para la predicción de la resistencia axial en probetas

Resultados de entrenamiento

Se obtienen los resultados del factor de correlación R tanto para el conjunto completo de datos como específicamente para el conjunto de datos de entrenamiento (que es el conjunto de datos más extenso), y también para el conjunto de datos de prueba, con el fin de evaluar el rendimiento de la red neuronal en la predicción de datos nuevos. Por otra parte, se determina el valor del Error Cuadrático Medio (MSE) del total de las salidas.

Tabla A.5.3.3 Resultados de las predicciones de las RNA

N° neuronas en la capa oculta	R			MSE
	Entrenamiento	Test	Total	
1	0.966	0.973	0.969	1.50E-02
2	0.974	0.971	0.973	1.32E-02
3	0.979	0.972	0.977	1.18E-02
4	0.983	0.972	0.979	1.02E-02
5	**0.986**	**0.976**	**0.981**	**9.28E-03**
6	0.979	0.978	0.979	1.03E-02
7	0.986	0.971	0.977	1.18E-02

Para determinar la arquitectura que mejor se ajusta a la base de datos entregada se identifica que el valor del coeficiente de correlación R sea más cercano a 1, esto indica que hay una mejor correlación entre los resultados provenientes de la base de datos y las

predicciones hechas por la R.N.A. Además, se identifica la arquitectura que posee un menor valor de MSE, pues esto indica que la R.N.A. posee una mayor precisión que otras (Zarringol, 2021).

De esta manera podemos justificar que la red neuronal **PROB_fc_5**, la cual posee cinco neuronas en la capa oculta, es la RNA que se ajusta con mayor calidad y precisión en la estimación de la resistencia de probetas axial a la compresión de probetas de hormigón.

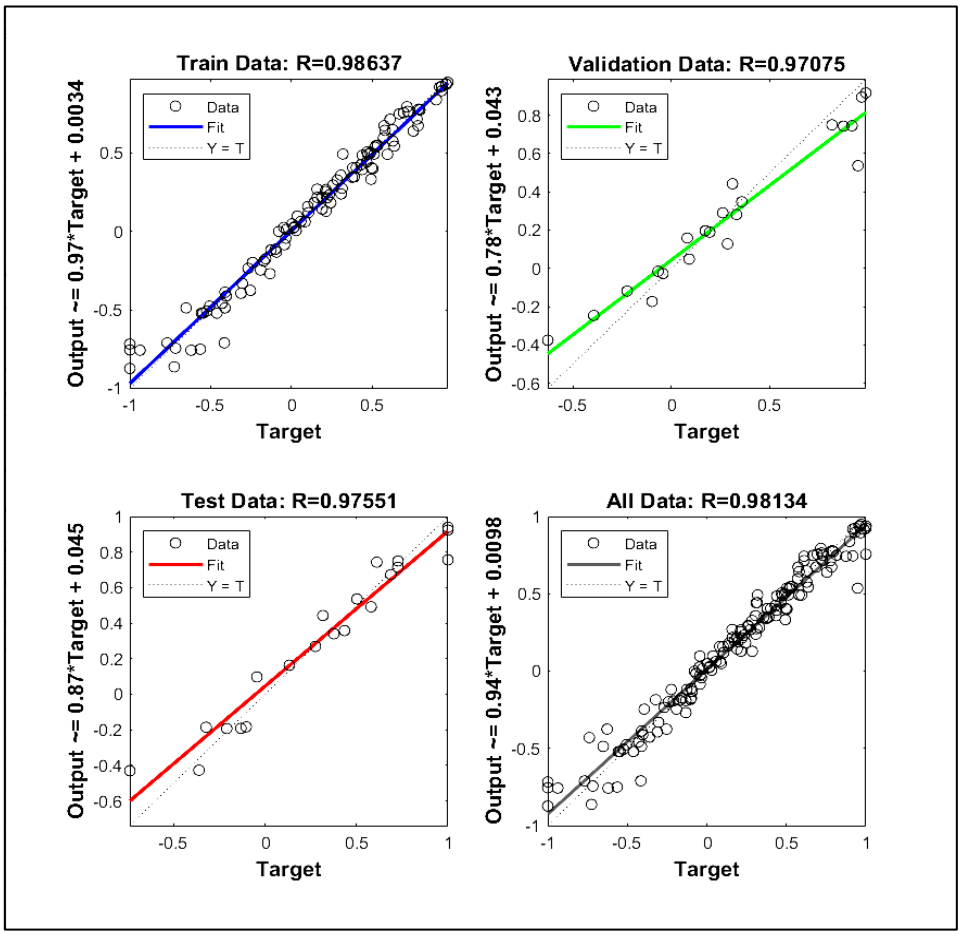

Figura A.5.3.3 Regresión de los datos predichos por la red PROB_fc_5.

La Figura 5.4 corresponde a la gráfica de regresión de la RNA para el total de los datos y la partición en los conjuntos de entrenamiento, validación y test. En el eje X se encuentran los datos “Target” los cuales corresponden a las salidas de la base de datos experimental y en el Eje Y se encuentran los datos “Output” correspondientes a las salidas predichas por la RNA, estos datos se encuentran normalizados en el rango de [-1,1].

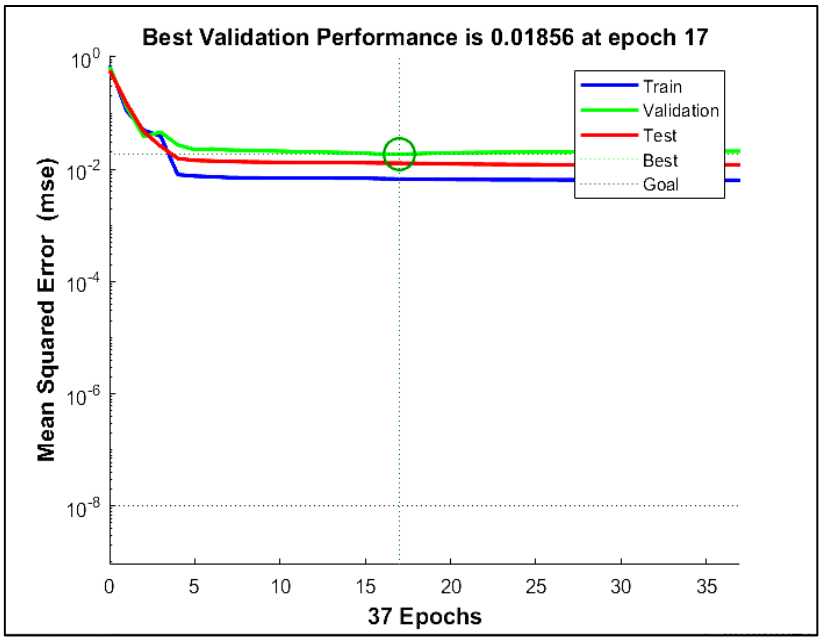

Figura A.5.3.4 Desempeño del entrenamiento de la red PROB_fc_5.

El entrenamiento de la red es el proceso de actualizar pesos y sesgos hasta que la red sea capaz de reducir el error en sus predicciones. En este caso, la figura 5.5 detalla entrenamiento de la red PROB_fc_5 la cual tomó 37 épocas de entrenamiento, es decir, se actualizaron 37 veces cada peso y sesgo. Sin embargo, este entrenamiento se ve detenido tempranamente en la época 17, pues el error del conjunto de datos de validación aumenta a medida que aumenta la cantidad de épocas, en vez de disminuir como en el conjunto de datos de entrenamiento. Esta técnica se conoce como detención temprana o "Early Stopping", y es una forma de controlar el sobreaprendizaje u "Overfitting" de la red.

Análisis de sensibilidad

Se utiliza la red neuronal PROB_fc_5 para analizar el comportamiento de las variables de salida ante la variación de las variables de entrada. El rango de variación de valores de cada variable de entrada abarca el 100% de su rango, obteniendo predicciones de resistencia de las probetas de hormigón para cada caso a través de la red neuronal entrenada (ver Figura A.5.3.5).

Los resultados de resistencia f'c graficados se encuentran normalizados con respecto al valor de f'c promedio para las edades 28, 56 y 91 días para ilustrar de mejor manera las

variaciones entre extremos de los valores de resistencia obtenidos para cada cambio en la dosificación.

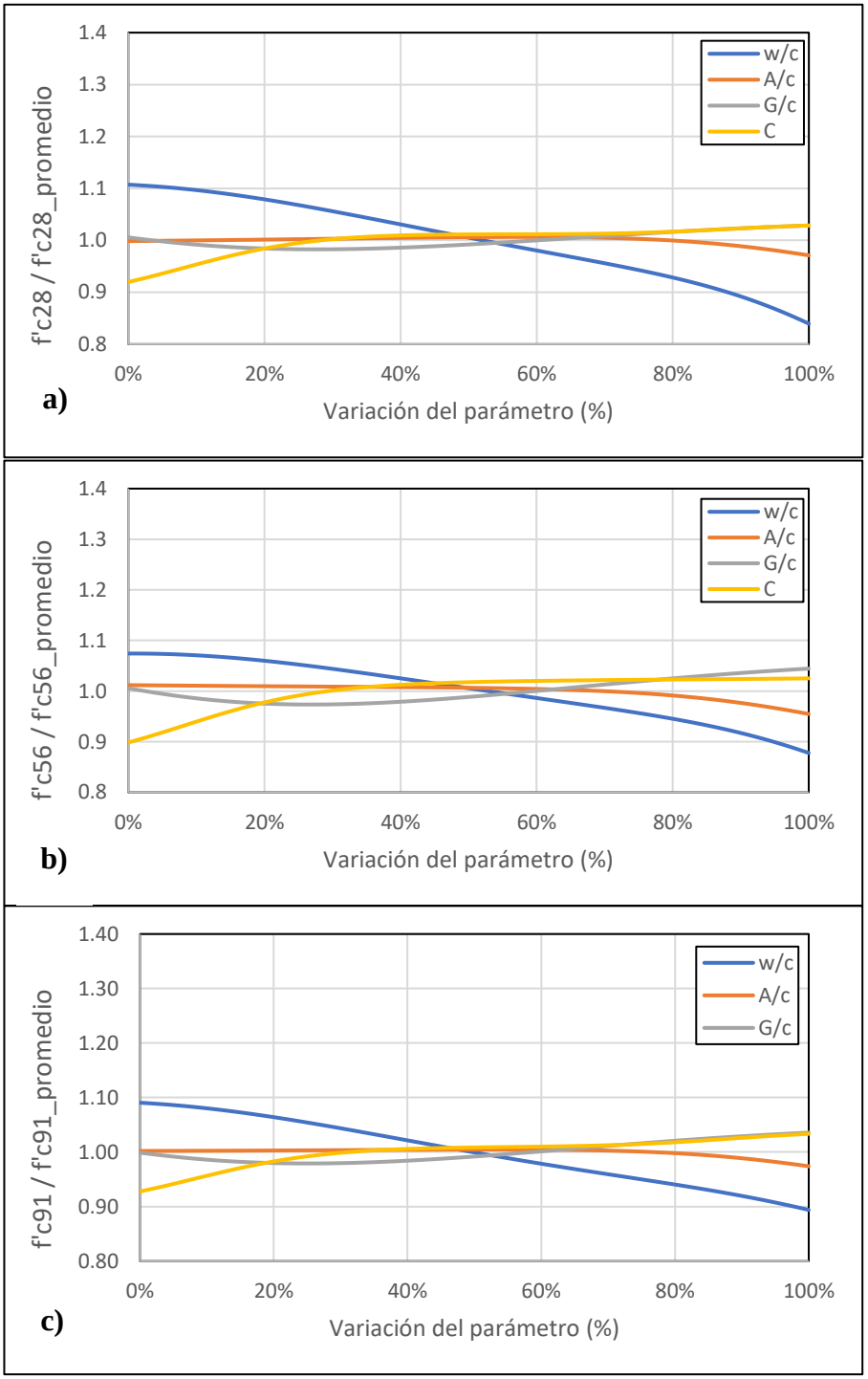

Figura A.5.3.5 Análisis de sensibilidad (a) f'c día 28 (b) f'c día 56 (c) f'c día 91.

En términos generales la variable asociada a la proporción agua/cemento de la mezcla (w/c) posee mayor influencia en la estimación de la resistencia a la compresión axial de probetas de hormigón en comparación a las otras tres variables de entrada, obteniendo

una variación máxima del 16% de la resistencia promedio alcanzada por la probeta para su edad más temprana. En menor grado de influencia se encuentra la variable asociada a la densidad del cemento (C), y en menor aun grado se encuentran a la par las variables asociadas a la dosificación; la proporción arena y grava con respecto al cemento.

Otro resultado del análisis de sensibilidad es que las variables de entrada poseen una atenuación de su influencia a medida que aumenta la edad de la probeta, donde las probetas de 28 días de edad poseen mayores cambios en su resistencia al variar los parámetros de entrada en comparación a las probetas de 91 días de edad.

Conclusiones

Las Redes Neuronales Artificiales exhiben una capacidad notable para llevar a cabo regresiones en diversas arquitecturas entrenadas. Específicamente, se observa un factor de correlación superior a 0.98 cuando se utiliza una red con cinco neuronas en la capa oculta, lo que resulta en una precisión significativa al predecir la resistencia a la compresión axial de probetas de hormigón.

A través del análisis de sensibilidad de la red, se resalta que las variables que rigen el problema muestran un comportamiento esperado. Esto sugiere que la base de datos ha sido depurada de datos anómalos que no contribuyen al proceso de generalización del problema.

Es importante señalar que los alcances de la red neuronal están condicionados por el rango de valores presente en la base de datos, constituyendo esta limitación la principal del algoritmo. La solución a este desafío implica manejar un volumen de datos más extenso, manteniendo al mismo tiempo la compatibilidad de los materiales y la metodología de ensayo. Esto permitirá ampliar aún más la capacidad de la red para estimar la resistencia a la compresión axial de las probetas de hormigón y así ser un aporte al medio ambiente reemplazando los métodos de ensayo de tipo destructivos clásicos por modelos predictivos de alta precisión.

UNIVERSIDAD DE CONCEPCIÓN – FACULTAD DE INGENIERÍA
RESUMEN DE MEMORIA DE TÍTULO

Departamento	**:** Departamento de Ingeniería Civil
Carrera	**:** Ingeniería Civil
Nombre del memorista	**:** Sebastián Nicolás Parra Vergara
Título de la memoria	**:** Aplicaciones de redes neuronales artificiales para el diseño óptimo a flexión de vigas de hormigón armado.
Fecha de la presentación oral	**:**
Profesor(es) Guía	**:** Dr. Patricio Cendoya H.
Profesor(es) Revisor(es)	**:** Dr. Víctor Aguilar V.
Concepto	**:**
Calificación	**:**

Resumen

En los últimos años, hemos sido testigos de un crecimiento exponencial en la aplicación de algoritmos de inteligencia artificial en diversos ámbitos de estudio dentro de la Ingeniería Estructural. En este contexto, se vuelve necesario emplear herramientas computacionales para identificar las potenciales amenazas, oportunidades y limitaciones que estas tecnologías pueden ofrecer al ejercicio de la profesión.

Tras describir los algoritmos de Machine Learning y sus clasificaciones, se profundizó en el análisis del algoritmo de Redes Neuronales Artificiales (RNA), detallando sus fundamentos teóricos, describiendo su estructura y método de aprendizaje, así como sus diversas aplicaciones en la Ingeniería Estructural. Paralelamente, se implementó una rutina en entornos MATLAB®.

Posteriormente, se abordó la aplicación de las RNA para el diseño de vigas rectangulares de hormigón armado a flexión bajo condiciones de apoyo simple, empotrado y voladizo con carga distribuida. Para ello, se proporcionó una base de datos de 1302 vigas diseñadas a flexión analíticamente, cumpliendo con los requisitos del código ACI 318-19, junto con una formulación basada en la optimización del costo de la viga.

Finalmente, se demostró la alta capacidad de aprendizaje de las RNA entrenadas, junto con las limitaciones de este modelo, donde se destaca la necesidad de bases de datos robustas y la naturaleza de "caja negra" de las RNA. Además, y cumpliendo con otro objetivo del presente trabajo, se generó una interfaz gráfica para permitir el uso de las RNA entrenada a todo tipo de usuarios.