DISEÑO DE UN ESTIMADOR DE ANGULO DE BALANCEO DE UN VEHÍCULO INDUSTRIAL MEDIANTE REDES NEURONALES

TRABAJO FIN DE GRADO

Alberto Morales Pizarro

Dirigido por

Leandro José Vargas

Universidad Carlos III de Madrid

Grado de Ingeniería Mecánica

Departamento de Ingeniería Mecánica

Leganés 2016

A mis padres Juan y Elena,

gracias a su apoyo incondicional.

A mi novia Jhannete,

por sus ánimos y su cariño.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

Figuras del anexo A

Figuras del anexo B

ÍNDICE DE TABLAS

NOMENCLATURA

θ	Angulo de balanceo
δ	Angulo de dirección
η	Parámetro de la tasa de aprendizaje
$\dot{\theta}$	Variación del ángulo de balanceo
$\dot{\psi}$	Variación del ángulo de guiñada
$\varphi_j(\cdot)$	Función de activación asociada a la neurona j
a_y	Aceleración lateral
a_x	Aceleración longitudinal
b_j	Sesgo aplicado a la neurona j
$e_j(n)$	Error de la señal en la salida de la neurona j en la iteración n
$E(n)$	Suma instantánea del error cuadrático en la iteración n
E_{av}	Media del error cuadrático
$d_j(n)$	Respuesta deseada para la neurona j
m_l	Número de nodos en la capa l del Perceptron Multicapa
$o_k(n)$	Elemento kth del vector de salida global
$v_j(n)$	Suma de todos los pesos y sesgos conectados a la neurona j
V_x	Velocidad longitudinal
$w_{ji}(n)$	Peso sináptico existente entre la salida de la neurona i y la entrada de la neurona j
$\Delta w_{ji}(n)$	Corrección del peso sináptico
$x_i(n)$	Elemento ith del vector de entrada
y_{ij}	Deformación de la suspensión izquierda/derecha del eje delantero/trasero
$y_j(n)$	Función que aparece a la salida de la neurona j en la iteración n

Capítulo 1

Introducción y objetivos

1.1. Introducción

Los recientes avances que ha tenido el sector de los vehículos industriales, junto con la necesidad de incrementar los niveles de seguridad en la conducción de los mismos, requieren del diseño de nuevos y mejores sistemas de seguridad. Desde el 2014, todos los vehículos industriales comercializados en Europa deben disponer del sistema ESC (Control Electrónico de Estabilidad). Este sistema se alimenta de señales provenientes de sensores instalados en el vehículo, permitiendo una actuación sobre el sistema de frenos en caso de pérdida de control del vehículo.

Con objeto de reducir los accidentes de tráfico en las carreteras, en los últimos años se han aumentado los requisitos de seguridad para los pasajeros. En España, la Dirección General de Tráfico (DGT) proporciona en sus publicaciones los anuarios estadísticos de accidentes. Como se aprecia en la Figura 1, el número de fallecidos por accidentes de tráfico tuvo su pico en el año 1989 en el que se produjeron 9344 víctimas mortales. A partir de ese año se observa una tendencia descendente hasta reducir esa cifra a las 1688 víctimas mortales que se produjeron en el año 2014.

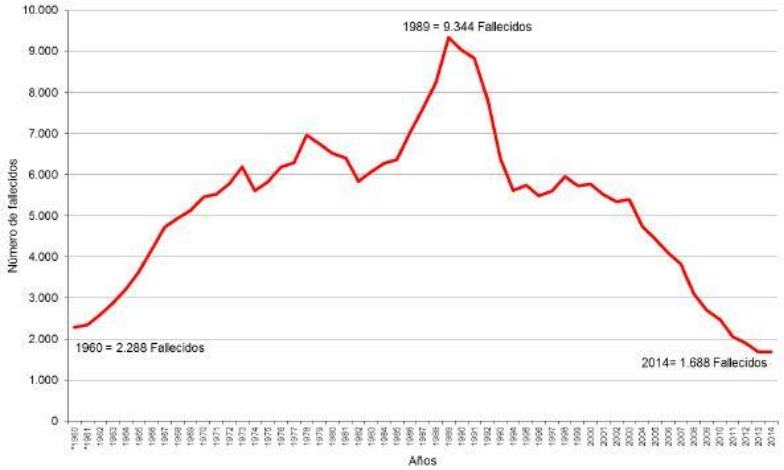


Figura 1. Víctimas mortales por accidente de tráfico. Serie 1965 a 2014 [1].

Este descenso en el número de víctimas mortales se debe en gran medida al desarrollo de sistemas de seguridad activa de los vehículos, tales como el sistema de control de estabilidad de guiñada o el sistema de control antivuelco. Por lo general estos sistemas dependen del conocimiento de variables dinámicas que deben ser obtenidas de los vehículos y del diseño de

programas capaces de mejorar el comportamiento del vehículo en situaciones críticas [2]–[4]. En general, la tasa de guiñada es una magnitud fácilmente medible en el vehículo pero el ángulo de balanceo, el cual tiene una gran importancia para la seguridad activa de los vehículos, presenta grandes inconvenientes para ser medido directamente en los coches modernos, debido al coste que supone hacerlo y a los problemas de fiabilidad. Como consecuencia de ello, esta variable puede ser estimada.

1.2. Objetivos del proyecto

El objetivo de este trabajo es diseñar un estimador del ángulo de balanceo, mediante redes neuronales, que sea capaz de predecir dicho valor para cualquier maniobra del vehículo. La motivación de este objetivo proviene de la importancia e influencia del ángulo de balanceo en los modelos de estabilidad lateral, ya que el desplazamiento del centro de gravedad genera un momento de vuelco adicional.

En resumen, se puede establecer como objetivo general lo siguiente:

- Diseñar un estimador del ángulo de balanceo de un vehículo industrial mediante el uso de redes neuronales.

Para cumplir con este propósito es necesario cumplir objetivos específicos tales como:

- Instrumentar un vehículo industrial para validar el modelo de simulación mediante ensayos dinámicos.

- Crear un modelo de simulación de un vehículo industrial tipo furgón, mediante el programa TruckSim.

- Diseñar una red neuronal para la estimación del ángulo de balanceo del vehículo.

- Validar el estimador mediante ensayos dinámicos con un vehículo instrumentado.

Capítulo 2

Estado del arte

2.1. Antecedentes

En los últimos años, se ha dedicado una gran cantidad de esfuerzo a la estimación del ángulo de balanceo del vehículo. En [5], basándose en las dinámicas no lineales del vehículo y en el modelo Dugoff de neumático proponen la estimación del ángulo de balanceo mediante la utilización de las mediciones obtenidas (velocidades angulares de las ruedas, aceleración lateral y longitudinal, variación del ángulo de guiñada, variación del ángulo de balanceo y ángulo de giro del volante) en los sensores que disponen en las cuatro ruedas de un vehículo eléctrico. Dicho modelo se evalúa experimentalmente bajo una variedad de maniobras y condiciones de la carretera. En [6], se realiza la estimación del ángulo de balanceo y del ángulo de cabeceo mediante la medición de la aceleración y la tasa de rotación de un vehículo en las tres dimensiones a una alta frecuencia de muestreo junto con el uso de unidades de medición inercial (IMU : Inertial Measurement Units) y de un modelo cinemático, obteniéndose estimaciones precisas de ambos ángulos. En [7], mediante la utilización del algoritmo RLS (*Recursive Least Squeares*) y un filtro de Kalman se propone un método para estimar con precisión el ángulo de deslizamiento lateral del vehículo y el ángulo de balanceo usando sensores de fuerza en los neumáticos. Los resultados experimentales demuestran que el diseño del filtro de Kalman proporciona una estimación fiable y sin retraso de fase notable de ambos ángulos cuando se usan las fuerzas laterales de los neumáticos. En [8], se utilizan sensores de fuerza lateral para obtener la fuerza lateral del neumático y con esta variable mediante un observador lineal implementado en un vehículo experimental se estima el ángulo de balanceo. Con dicho ángulo estimado se diseña un controlador de momento de balanceo basado en la metodología de control general de dos grados de libertad (2-GDL).

La utilización de redes neuronales artificiales (RNA) supone un método alternativo para la estimación de variables dinámicas que afectan a la seguridad activa de los vehículos. Existen un gran número de referencias exitosas en lo que se refiere al uso de las RNA para estimar dichas variables [9]–[11]. En estas referencias se hace uso de las mismas para desarrollar sistemas de alerta que permitan al conductor evitar una situación inminente de peligro. De igual manera, en [12] mediante una red neuronal que estima la estabilidad de giro instantánea a partir de entradas que son fácilmente medibles en vehículos equipados con sistemas ESC (variación del ángulo de guiñada, aceleración lateral…) se desarrolla un software de alerta de vuelco con el objetivo de predecir todas las situaciones de vuelco inminente y dar aviso inmediato al conductor.

Desde que se iniciaron los fundamentos de la computación neuronal hasta la actualidad, se han desarrollado distintos tipos de arquitecturas de entrenamiento, siendo probablemente las más populares y por tanto las de más aplicación, las arquitecturas con algoritmos de

propagación hacia atrás (BP) [13], [14]. En [15], se comparan dos arquitecturas de redes neuronales con el fin de determinar cuál de ellas ofrece mejores resultados. La primera es la llamada Perceptron Multicapa (MLP: Multiplayer Perceptron), la cual es una de las redes neuronales más comunes, mientras que la segunda son los llamados Mapas Autoorganizados (SOM: Self-Organizing Map). Los resultados muestran que ambas arquitecturas proporcionan pequeños errores en las estimaciones teniendo el Perceptron Multicapa la ventaja de obtener una convergencia más rápida.

Para problemas de estimaciones, normalmente se utiliza un algoritmo BP para entrenar al Perceptron Multicapa. Existen múltiples variaciones de este tipo de algoritmo, pero el Levenberg-Marquardt (LM) es el más utilizado por su rapidez de cálculo y eficiencia, dado que elimina el cálculo de la matriz Hessiana que realizan los otros algoritmos.

Por todo ello, en este trabajo se propone la estimación del ángulo de balanceo de un vehículo industrial mediante el diseño de una red neuronal artificial. La arquitectura de esta red será el Perceptron Multicapa, la cual será entrenada con un algoritmo BP.

2.2. Historia de las redes neuronales

En este apartado se destacan los aspectos más relevantes de las RNA desde sus inicios en 1936 hasta la década de los 90:

- 1936: Alan Turing fue el primero en estudiar el cerebro como una forma de ver el mundo de la computación. Sin embargo, los primeros teóricos que concibieron los fundamentos de la computación neuronal fueron Warren McCulloch, un neurofisiólogo, y Walter Pitts, un matemático, quienes en 1943, lanzaron una teoría acerca de la forma de trabajar de las neuronas (Un Cálculo Lógico de la Inminente Idea de la Actividad Nerviosa – Boletín de Matemática Biofísica 5: 115-133). Ellos modelaron una red neuronal simple mediante circuitos eléctricos. El entusiasmo que despertó el modelo neuronal impulsó la investigación en esta línea durante los años 50 y 60.
- 1949: Donald Hebb fue el primero en explicar los procesos del aprendizaje (el cual es el elemento básico de la inteligencia humana) desde un punto de vista psicológico, desarrollando una regla de cómo ocurre el aprendizaje. Aun hoy, éste es el fundamento de la mayoría de las funciones de aprendizaje que pueden hallarse en una red neuronal. Su idea fue que el aprendizaje ocurría cuando ciertos cambios en una neurona eran activados.
- 1950: Karl Lashley encontró en una serie de ensayos que la información no era almacenada en forma centralizada en el cerebro sino que era distribuida encima de él.
- 1956: Frecuentemente se menciona el Congreso de Dartmouth para indicar el nacimiento de la inteligencia artificial.
- 1957: Frank Rosenblatt comenzó el desarrollo del Perceptron. Esta es la red neuronal más antigua; utilizándose hoy en día para aplicación como identificador de patrones. Este modelo era capaz de generalizar, es decir, después de haber aprendido una serie

de patrones podía reconocer otros similares, aunque no se le hubiesen presentado en el entrenamiento. Sin embargo, tenía una serie de limitaciones, por ejemplo, su incapacidad para resolver el problema de la función OR-exclusiva y, en general, era incapaz de clasificar clases no separables linealmente.

- 1959: Frank Rosenblatt publicó el libro Principios de Neurodinámica en el cual confirmó que, bajo ciertas condiciones, el aprendizaje del Perceptron convergía hacia un estado finito (Teorema de Convergencia del Perceptron).
- 1960: Bernand Widroff y Marcian Hoff desarrollaron una importante variación del algoritmo de aprendizaje del Perceptron, la denominada ley de Widrow-Hoff, que dio lugar al modelo ADALINE (ADAptative LINear Elements). Esta fue la primera red neuronal aplicada a un problema real (filtros adaptativos para eliminar ecos en las líneas telefónicas) que ha sido utilizado comercialmente durante varias décadas.
- 1961: Karl Steinbeck desarrolló el primer modelo matemático de memoria asociativa, llamado Lernmatrix.
- 1969: En éste año casi se produjo la muerte abrupta de las RNA, ya que Marvin Minsky y Seymour Papert probaron (matemáticamente) que el Perceptron no era capaz de resolver problemas relativamente fáciles, tales como el aprendizaje de una función no-lineal. Esto demostró que el Perceptron era muy débil, dado que las funciones no lineales son extensamente empleadas en computación y en los problemas del mundo real.
- 1974: Paul Werbos desarrolló la idea básica del algoritmo de aprendizaje BP, cuyo significado quedó definitivamente aclarado en 1985.
- 1977: Stephen Grossberg publicó la Teoría de Resonancia Adaptada (TRA), la cual es una arquitectura de red que se diferencia de todas las demás previamente inventadas. Esta red simula otras habilidades del cerebro: memoria a largo y corto plazo.

  En ese mismo año James Anderson desarrolló el Asociador Lineal y su extensión conocida como red Brain-State-in-a-Box, que permitieron modelizar funciones arbitrariamente complejas y Kohonen desarrolló la red auto asociativa LVQ, especializada en la detección de clusters de ejemplos.
- 1985: John Hopfield provocó el renacimiento de las redes neuronales con su libro: Computación neuronal de decisiones en problemas de optimización. Ese mismo año el instituto americano de Física establece la reunión anual Neuronal Networks for Computing.
- 1986: David Rumelhart y Geoffrey Hinton redescubrieron el algoritmo de aprendizaje BP, el cual evitaba los problemas observados en el aprendizaje del Perceptron Simple. Este algoritmo constituye desde entonces una de las reglas de aprendizaje de mayor utilización en el ámbito de la neurocomputacion para el entrenamiento de la red conocida como Perceptron Multicapa.
- 1987: Se celebró la primera conferencia internacional sobre redes neuronales artificiales (RNA). Ese mismo año se creó la Sociedad Internaciones de Redes Neuronales (INNS).
- Década de los 90: Las RNA comienzan a ser aplicadas en distintos campos del conocimiento: clasificación de patrones, robótica, visión artificial, procesamiento de

señales, reconocimiento de escritura y habla, etc. En la actualidad son numerosos los trabajos que se realizan cada año, las aplicaciones nuevas que surgen y las empresas que lanzan nuevos productos, tanto en hardware como en software, especialmente para simulación.

2.3. Definición de las Redes Neuronales Artificiales (RNA)

Existen múltiples definiciones de las RNA entre las que destacan las siguientes:

- En [16] se definen como una nueva forma de computación, inspirada en modelos biológicos. Un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles.
- En [17] como redes interconectadas masivamente en paralelo de elementos simples y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico.
- En [18] como un sistema de computación constituido por un gran número de elementos simples de procesamiento muy interconectados, que procesan información por medio de su estado dinámico como respuesta a entradas externas.

En cuanto a las definiciones, cabe señalar la propuesta por el grupo PDP (Parallel Distributed Processing Research Group) de la Universidad de California-San Diego, pionero en la investigación de las RNA, que considera que un sistema neuronal está compuesto por los siguientes elementos básicos:

- Un conjunto de procesadores elementales o neuronas artificiales.
- Un patrón de conectividad o arquitectura.
- Una dinámica de actividades.
- Una regla o dinámica de aprendizaje.
- El entorno donde opera.

Desde un punto de vista formal, una RNA puede definirse haciendo uso del concepto de grafo. Este es un objeto integrado por un conjunto de nodos (vértices) y de conexiones (links) entre los mismos, denominándose dirigido cuando todas las conexiones tienen asignado un sentido y no dirigido cuando tales conexiones son bidireccionales. Asimismo, puede hablarse de grafos densos (cuando casi todos los nodos están conectados entre sí) y de grafos dispersos (cuando las conexiones entre nodos son escasas).

La forma más habitual de representación de grafos caracteriza a los nodos a través de círculos y a las conexiones como líneas o flechas, lo que guarda una estrecha relación con la representación tradicional de una RNA, como se muestra en la Figura 2.

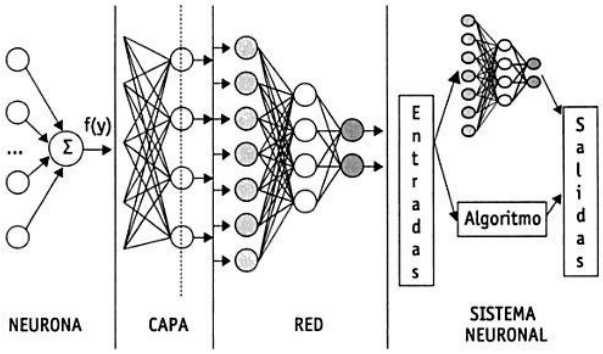


Figura 2. Estructura jerárquica de un sistema basado en RNA [16]

Otra manera de representar un grafo es como una matriz de conexiones, que será simétrica si el grafo es no dirigido, o mediante el empleo de una lista de conexiones, indicativa de la forma de conexión entre los nodos.

En [16] definen a la red neuronal como un grafo dirigido, con las siguientes propiedades:

- A cada nodo j se le asocia una variable de estado x_j .
- A cada conexión (i,j) de los nodos i y j se le asocia un peso $w_{ij} \in \mathbb{R}$.
- En muchas ocasiones a cada nodo se le asocia un umbral de disparo θ_j .
- Para cada nodo j se define una función $f_j(x_j, w_{ij}, \theta_j)$, que depende de los pesos de sus conexiones, del umbral y de los estados de los nodos i a él conectados. Esta función proporciona el nuevo estado del nodo.

Considerando el lenguaje habitual de los grafos, pueden establecerse las siguientes equivalencias:

- Un nodo se representa mediante una neurona.
- Una conexión se representa mediante una sinapsis.
- Una neurona de entrada es aquella sin conexiones entrantes.
- Una neurona de salida es aquella sin conexiones salientes.
- Las neuronas que no son de entrada ni de salida se denominan neuronas ocultas.
- Una red es unidireccional si no presenta bucles cerrados de conexiones.
- Una red es recurrente si el flujo de información puede encontrar un bucle de atrás hacia adelante, esto es, una retroalimentación.

Finalmente, con objeto de establecer una analogía directa entre la actividad sináptica biológica y las RNA, en la Tabla 1 se presentan una serie de relaciones fundamentales.

Tabla 1. Analogía entre la neurona biológica y la neurona real [16]

Neurona biológica	Neurona artificial
- Señales que llegan a la sinapsis	- Entradas a la neurona
- Carácter excitador o inhibidor de las sinapsis de entrada	- Pesos de entrada
- Estimulo total de la neurona	- $\sum (Entrada_i \, . \, Peso_i)$
- Activación o no de la neurona	- Función de activación
- Respuesta de la neurona	- Función de salida

2.4. Principios de funcionamiento de las RNA

Debido a su constitución y a sus fundamentos, las RNA presentan un gran número de características semejantes a las del cerebro. Por ejemplo, son capaces de aprender de la experiencia, generalizar de casos anteriores a casos nuevos, abstraer características esenciales a partir de entradas que representan información irrelevante, etc. Esto hace que ofrezcan numerosas ventajas y que este tipo de tecnología se esté aplicando en múltiples áreas. Para [16] las cinco ventajas más importantes son las citadas a continuación:

- ***Aprendizaje adaptativo:*** esta es quizás la característica más importante de las RNA, ya que pueden comportarse en función de un entrenamiento con una serie de ejemplos ilustrativos. De esta manera, no es necesario elaborar un modelo a priori, ni establecer funciones probabilísticas. Las redes neuronales son dinámicas, pues son capaces de estar constantemente cambiando para adaptarse a las nuevas condiciones.
  En el proceso de aprendizaje, los enlaces ponderados de las neuronas se ajustan de manera que se obtengan ciertos resultados específicos. Una red neuronal no necesita un algoritmo para resolver un problema, ya que ella puede generar su propia distribución de pesos en los enlaces mediante el aprendizaje. También existen redes que continúan aprendiendo a lo largo de su vida, después de completado su periodo de entrenamiento.
  La función del diseñador es únicamente la obtención de la arquitectura apropiada. No es función del diseñador cómo la red aprenderá a discriminar. Sin embargo, sí es necesario que desarrolle un buen algoritmo de aprendizaje que le proporcione a la red la capacidad de discriminar, mediante un entrenamiento con patrones.

- ***Auto-organización***: las redes neuronales emplean su capacidad de aprendizaje adaptativo para auto organizar la información que reciben durante el aprendizaje y/o la operación. Mientras que el aprendizaje es la modificación de cada elemento procesal, la auto organización consiste en la modificación de la red neuronal completa para llevar a cabo un objetivo específico.
  Cuando las redes neuronales se usan para reconocer ciertas clases de patrones, ellas auto organizan la información usada. Por ejemplo, la red neuronal de propagación hacia atrás (BPNN), creará su propia representación característica, mediante la cual puede reconocer ciertos patrones.

Esta auto organización provoca la generalización (facultad de las redes neuronales de responder apropiadamente cuando se les presentan datos o situaciones a las que no había sido expuesta anteriormente) de la entrada para obtener una respuesta. Esta característica es muy importante cuando se tiene que solucionar problemas en los cuales la información de entrada no es muy clara; además permite que el sistema proporcione una solución, incluso cuando la información de entrada está especificada de forma incompleta.

- ***Tolerancia a fallos***: las redes neuronales fueron los primeros métodos computacionales con la capacidad inherente de tolerancia a fallos, comparados con los sistemas tradicionales, los cuales pierden su funcionalidad cuando sufren un pequeño error de memoria. En las redes neuronales, si se produce un fallo en un número no muy grande de neuronas, aunque el comportamiento del sistema se ve influenciado, no sufre una caída repentina.
  Existen dos aspectos diferentes de las RNA en cuanto a la tolerancia a fallos:
  a) Pueden aprender a reconocer patrones con ruido, distorsionados o incompletos.
  b) Pueden seguir realizando su función (con cierta degradación) aunque se destruya parte de la red.

  La razón por la que las redes neuronales son tolerantes a los fallos, es que tienen su información distribuida en las conexiones entre neuronas, existiendo cierto grado de redundancia en este tipo de almacenamiento. La mayoría de los ordenadores algorítmicos y sistemas de recuperación de datos almacenan cada pieza de información es un espacio único, localizado y direccionable. En cambio, las redes neuronales almacenan información no localizada. Por lo tanto, la mayoría de las interconexiones entre los nodos de la red tendrán sus valores en función de los estímulos recibidos, y se generará un patrón de salida que represente información almacenada.

- ***Operación en tiempo real***: de todos los métodos existentes, las RNA son las más indicadas para el reconocimiento de patrones en tiempo real, debido a que trabajan en paralelo actualizando todas sus instancias simultáneamente. Para que la mayoría de las redes puedan operar en un entorno de tiempo real es necesario un cambio mínimo en los pesos de las conexiones o entrenamiento.

- ***Fácil inserción dentro de la tecnología existente***: una red individual puede ser entrenada para desarrollar una única y bien definida tarea (tareas complejas, que hagan múltiples selecciones de patrones, requerirán sistemas de redes interconectadas). Con las herramientas de computación existentes una red puede ser rápidamente entrenada, comprobada, verificada y trasladada a una implementación de hardware de bajo coste. Por lo tanto, no se presentan dificultades para la inserción de redes neuronales en aplicaciones específicas, por ejemplo de control, dentro de los sistemas existentes. De esta manera, las redes neuronales se pueden utilizar para

mejorar sistemas en forma incremental y cada paso puede ser evaluado antes de acometer un desarrollo más amplio.

2.5. Elementos básicos de una RNA

La Figura 3 muestra un esquema de una red neuronal:

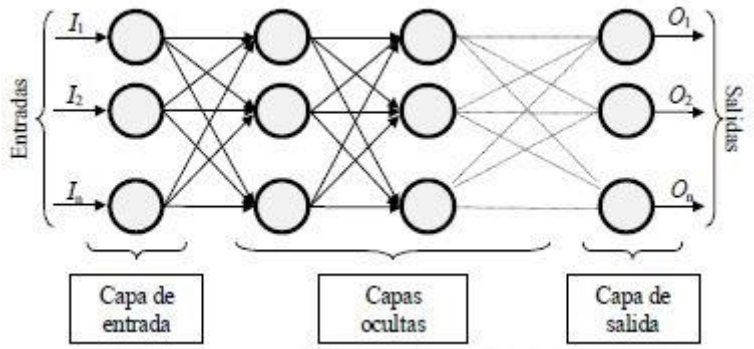


Figura 3. Estructura de una RNA multicapa [16]

Esta red está constituida por neuronas interconectadas y ordenadas en tres capas (esto último puede variar). Los datos ingresan por medio de la capa de entrada, pasan a través de la capa oculta y salen por la capa de salida.

En la Figura 4 se observan las similitudes entre una neurona real y una artificial (tienen entradas, utilizan pesos y generan salidas).

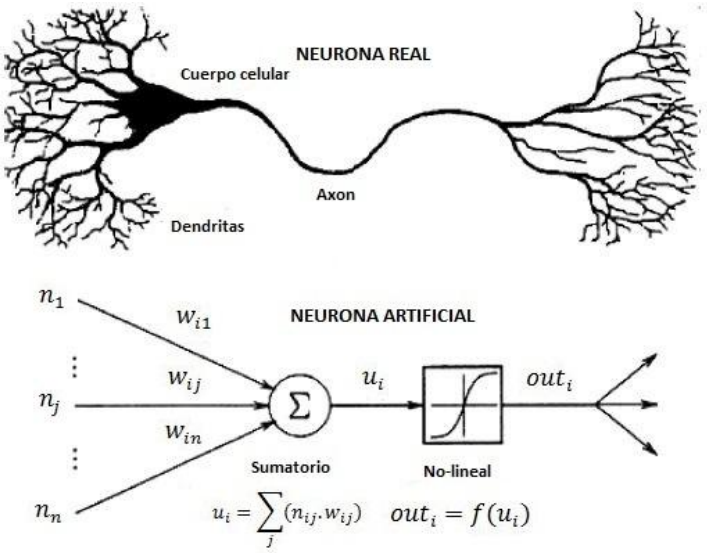


Figura 4. Comparación entre una neurona biológica y una artificial [16]

2.5.1. Función de entrada (*input function*)

La neurona trata a muchos valores de entrada como si fueran uno sólo; esto recibe el nombre de entrada global. Por tanto, ahora hay que afrontar el problema de cómo se pueden combinar estas simples entradas (n_{i1} ,n_{i2} ,…) dentro de la entrada global, gn_i. Esto se logra a través de la función de entrada, la cual se calcula a partir del vector entrada. La función de entrada puede describirse como sigue:

$$input_i = (n_{i1}.w_{i1}).(n_{i2}.w_{i2}) \dots (n_{in}.w_{in}) \tag{2.1}$$

donde n representa al número de entradas a la neurona N_i y w_i al peso.

Los valores de entrada se multiplican por los pesos anteriormente ingresados a la neurona. Por consiguiente, los pesos que generalmente no están restringidos cambian la medida de influencia que tienen los valores de entrada. Es decir, que permiten que un gran valor de entrada tenga solamente una pequeña influencia, si éstos son lo suficientemente pequeños.

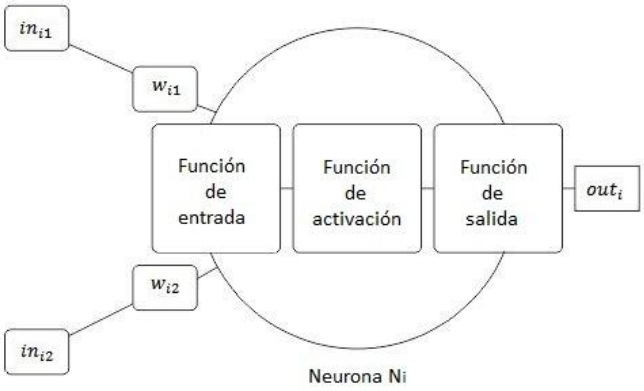


Figura 5. Ejemplo de una neurona con dos entradas y una salida [19]

El conjunto de todas las n entradas $n_i = (n_{i1}, n_{i2}, \dots, n_{in})$ es comúnmente llamado vector de entrada. Algunas de las funciones de entrada más comúnmente utilizadas y conocidas son:

1) Sumatoria de las entradas pesadas: es la suma de todos los valores de entrada a la neurona, multiplicados por sus correspondientes pesos (Figura 4).

$$u_i = \sum_j (n_{ij}.w_{ij}), \quad con\ j = 1,2,\dots,n \tag{2.2}$$

2) Productoria de las entradas pesadas: es el producto de todos los valores de entrada a la neurona, multiplicados por sus correspondientes pesos.

$$u_i = \prod_j (n_{ij}.w_{ij}), \quad con\ j = 1,2,\dots,n \tag{2.3}$$

3) Máximo de las entradas pesadas: solamente toma en consideración el valor de entrada más fuerte, previamente multiplicado por su peso correspondiente.

$$u_i = Max(n_{ij}.w_{ij}), \quad con\ j = 1,2,\dots,n \tag{2.4}$$

2.5.2. Función de activación (*activation function*)

Una neurona biológica puede estar activa (excitada) o inactiva (no excitada); es decir que tiene un estado de activación. Las neuronas artificiales también tienen diferentes estados de activación; algunas de ellas, solamente dos al igual que las biológicas, pero otras pueden tomar cualquier valor dentro de un conjunto determinado.

La función de activación calcula el estado de actividad de una neurona; transformando la entrada global (menos el umbral, θ_i) en un valor (estado) de activación, cuyo rango normalmente va de (0 a 1) o de (-1 a 1). Esto es así, porque una neurona puede estar totalmente inactiva (0 o -1) o activa (1).

La función de activación, es una función de la entrada global (u_i) menos el umbral (θ_i). Las funciones de activación más comúnmente utilizadas son las siguientes:

1) Función lineal

$$f(x) = \begin{cases} -1 & x \leq -1/a \\ a.x & -1/a < x < 1/a \\ 1 & x \geq 1/a \end{cases} \tag{2.5}$$

Con x = u_i - θ_i , y a > 0

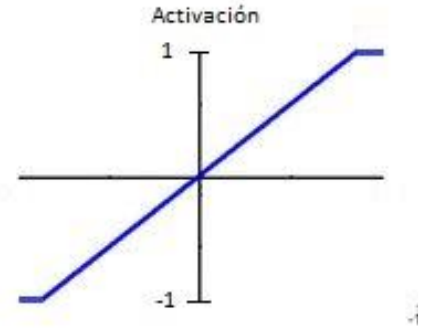


Figura 6. Función de activación lineal

Los valores de salida obtenidos por medio de esta función de activación serán a.(u_i - θ_i), cuando el argumento de (u_i - θ_i) esté comprendido dentro del rango (-1/a, 1/a). Por encima o por debajo de esta zona se fija la salida en 1 o -1, respectivamente. Cuando a=1 (dado que la misma afecta a la pendiente de la gráfica), la salida es igual a la entrada. Esta función está saturada en sus extremos, es de gran sencillez computacional y resulta más plausible desde el punto de vista biológico.

2) Función sigmoidea

$$f\,(x) = \frac{1}{1 + \ e^{-ax}}\ ;\ \ con\ x = u_i - \theta_i \qquad (2.6)$$

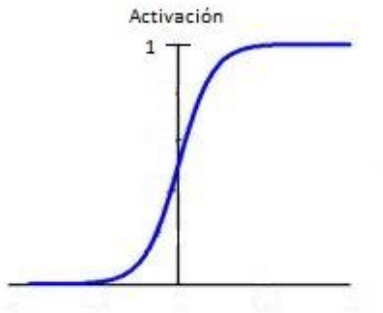


Figura 7. Función de activación sigmoidea

Los valores de salida que proporciona esta función están comprendidos dentro de un rango que va de 0 a 1. Al modificar el valor de a se ve afectada la pendiente de la función de activación. Estas funciones admiten la aplicación de las reglas de aprendizaje típicas de la función escalón, con la ventaja adicional que la derivada se encuentra definida en todo el intervalo, lo que permite emplear algoritmos de entrenamiento más avanzados.

3) Función tangente hiperbólica

$$f(x) = \frac{e^{ax} - e^{-ax}}{e^{ax} + \ e^{-ax}}\ ;\ \ \ con\ x = u_i - \theta_i \qquad (2.7)$$

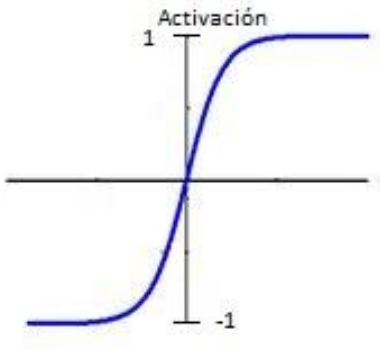


Figura 8. Función de activación tangente hiperbólica

Los valores de salida de esta función están comprendidos dentro del rango que va de -1 a 1.

2.5.3. Función de salida (output function)

El último componente que necesita una neurona es la función de salida. El valor resultante de esta función es la salida de la neurona i (out_i); por tanto, la función de salida determina que valor se transfiere a las neuronas vinculadas. Si la función de activación está por debajo de un umbral determinado, ninguna salida se pasa a la neurona subsiguiente. Normalmente, no se permite cualquier valor como una entrada para una neurona, por lo tanto, los valores de salida están comprendidos en el rango [0,1] o [-1,1].

Dos de las funciones de salida más comunes son:

- Ninguna: este es el tipo de función más sencillo, tal que la salida es la misma que la entrada. Es también llamada función identidad.
- Binaria: $\begin{cases} 1 & si \quad act_i \geq \xi_i \\ 0 & de\ lo\ contrario \end{cases}$, $donde\ \xi_i\ es\ el\ umbral$

2.6. Aprendizaje, Validación y Codificación

2.6.1. Niveles o capas de una red neuronal

La distribución de neuronas dentro de la red se realiza formando niveles o capas, con un número determinado de dichas neuronas en cada una de ellas. A partir de su situación dentro de la red, se pueden distinguir tres tipos de capas:

- De entrada: es la capa que recibe directamente la información proveniente de las fuentes externas de la red.
- Ocultas: son internas a la red y no tienen contacto directo con el entorno exterior. El número de niveles ocultos puede estar entre cero y un número ¨n¨. Las neuronas de las capas ocultas pueden estar interconectadas de distintas maneras, lo que determina, junto con su número, las distintas topologías de redes neuronales.
- De salidas: transfieren información de la red hacia el exterior.

2.6.2. Mecanismos de aprendizaje

En una red neuronal los datos de entrada se procesan a través de ella con el propósito de lograr una salida. Una red neuronal debe aprender a calcular la salida correcta para cada constelación (arreglo o vector) de entrada en el conjunto de ejemplos. Este proceso de aprendizaje se denomina: *proceso de entrenamiento o acondicionamiento*. El conjunto de datos (o conjunto de ejemplos) sobre el cual se basa este proceso es llamado: *conjunto de datos de entrenamiento.*

Existen múltiples definiciones del concepto de aprendizaje, de las cuales se destaca:

El aprendizaje consiste en la modificación del comportamiento inducido por la interacción con el entorno y como resultado de experiencias conducente al establecimiento de nuevos modelos de respuesta a estímulos externos [16] .

Dado que ni la topología de la red ni las diferentes funciones de cada neurona (entrada, activación y salida) pueden cambiar durante el aprendizaje, mientras que los pesos sobre cada una de las conexiones si pueden hacerlo, el aprendizaje de una red neuronal significa adaptación de pesos.

Por tanto, el aprendizaje es el proceso por el cual una red neuronal modifica sus pesos en respuesta a una información de entrada. Los cambios que se producen durante el mismo se reducen a la destrucción, modificación y creación de conexiones entre las neuronas. En los sistemas biológicos existe una continua destrucción y creación de conexiones entre las neuronas. En los modelos de redes neuronales artificiales, la creación de una nueva conexión implica que el peso de la misma pasa a tener un valor distinto de cero. De la misma manera, una conexión se destruye cuando su peso pasa a ser cero.

Durante el proceso de aprendizaje, los pesos de las conexiones de la red sufren modificaciones, por lo tanto, se puede afirmar que este proceso ha terminado (la red ha aprendido) cuando los valores de los pesos permanecen constantes ($dw_{ij}/dt = 0$).

Un aspecto importante respecto al aprendizaje de las redes neuronales es conocer cómo se modifican los valores de los pesos, es decir, cuáles son los criterios que se siguen para cambiar el valor asignado a las conexiones cuando se pretende que la red aprenda una nueva información.

Se puede distinguir dos métodos de aprendizaje importantes, cuya distinción procede del campo de reconocimiento de patrones:

1) Aprendizaje supervisado
2) Aprendizaje no supervisado

La diferencia fundamental entre ambos tipos radica en la existencia o inexistencia de un agente externo o supervisor, que controla el proceso de aprendizaje de la red.

Otro criterio que se puede utilizar para diferenciar las reglas de aprendizaje se basa en considerar si la red puede aprender durante su funcionamiento habitual o si el aprendizaje supone la desconexión de la red, es decir, su inhabilitación hasta que el proceso termine. En el primer caso, se trataría de un aprendizaje on line, mientras que el segundo se conoce como off line.

Cuando el aprendizaje es off line, se distingue entre una fase de aprendizaje o entrenamiento y una fase de operación o funcionamiento, existiendo un conjunto de datos de datos de entrenamiento y un conjunto de datos de test o prueba, que serán utilizados en la

correspondiente fase. Además, los pesos de las conexiones permanecen fijos después que termina la etapa de entrenamiento de la red. Debido precisamente a su carácter estático, estos sistemas no presentan problemas de estabilidad en su funcionamiento.

Una generalización de la fórmula o regla para enumerar los cambios en los pesos es la siguiente:

$$Peso\ nuevo = Peso\ Viejo + Cambio\ de\ Peso$$

Matemáticamente esto es:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \tag{2.8}$$

donde t hace referencia a la etapa de aprendizaje, w_{ij}(t + 1) al peso nuevo y w_{ij}(t) al peso viejo.

2.6.2.1. Aprendizaje supervisado

El aprendizaje supervisado se caracteriza porque el proceso de aprendizaje se realiza mediante un entrenamiento controlado por un agente externo (supervisor, maestro) que determina la respuesta que debería generar la red a partir de una entrada determinada. El supervisor controla la salida de la red y en caso de que esta no coincida con la deseada, se procederá a modificar los pesos de las conexiones, con el fin de conseguir que la salida obtenida se aproxime a la deseada. En la Figura 9 se muestra el proceso que se sigue en el aprendizaje supervisado.

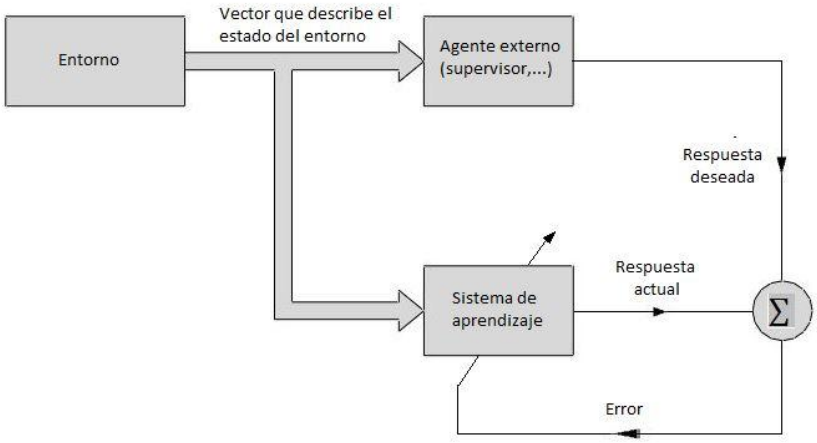


Figura 9. Diagrama de bloques del aprendizaje supervisado

En este tipo de aprendizaje se suelen considerar, a su vez, tres formas de llevarlo a cabo que dan lugar a los siguientes aprendizajes supervisados:

1) Aprendizaje por corrección de error

Consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos a la salida de la red, es decir, en función del error cometido en la salida.

Un ejemplo de este tipo de entrenamiento de algoritmos lo constituye la regla de aprendizaje del Perceptron, utilizada en el entrenamiento de la red de mismo nombre que desarrolló Rosenblatt en 1958, la cual presenta algunas limitaciones, como la no consideración de la magnitud del error global cometido durante el proceso completo de aprendizaje. Esta es una regla muy simple, para cada neurona en la capa de salida se le calcula la desviación a la salida objetivo como el error, δ. El cual luego se utiliza para cambiar los pesos sobre la conexión de la neurona precedente. El cambio de los pesos por medio de la regla de aprendizaje del Perceptron se realiza según la siguiente regla:

$$\Delta w_{ij} = \sigma * out_j * (a_{qi} - out_i) \tag{2.9}$$

$$\delta = (a_{qi} - out_i) \tag{2.10}$$

donde a_{qi} es la salida deseada/objetivo de la neurona de salida N_i y σ el aprendizaje.

La salida de la neurona N_j (out_j) se utiliza, porque este valor influye en la entrada global y, por ende, en la activación y luego en la salida de la neurona N_i . Esto es semejante a un efecto en cadena (Figura 10).

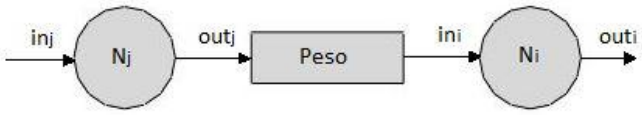


Figura 10. Influencia de la salida de la neurona N_j en la entrada de la neurona N_i [19]

Con objeto de superar las limitaciones del Perceptron Simple, Widrow y Hoff desarrollaron un nuevo algoritmo de aprendizaje más rápido y con mayor campo de aplicación, conocido como regla del error mínimo cuadrado (Least-Mean-Squared-Error: LMS Error) o regla Widrow-Hoff(en el caso de funciones de activación de tipo lineal), redenominada como regla delta para las funciones de activación de tipo sigmoidea. La LMS Error también utiliza la desviación a la salida objetivo, pero toma en consideración a todas las neuronas predecesoras que tiene la neurona de salida. Esto permite cuantificar el error global cometido en cualquier momento durante el proceso de entrenamiento de la red, lo cual es importante, ya que cuanta más información se tenga sobre el error cometido, más rápido se puede aprender. Luego el error calculado (δ) es igualmente repartido entre las conexiones de las neuronas predecesoras.

Por último, cabe destacar la regla de aprendizaje BP, también conocida como regla LMS multicapa, la cual es una generalización de la regla de aprendizaje Delta. Esta es la primera regla de aprendizaje que permitió realizar cambios sobre los pesos en las conexiones de la capa oculta.

2) Aprendizaje por refuerzo

Se trata de un aprendizaje supervisado, más lento que el anterior, que se basa en la idea de no disponer de un ejemplo completo del comportamiento deseado, es decir, de no indicar durante el entrenamiento exactamente que se desea que proporcione la red ante una determinada entrada.

En el aprendizaje por refuerzo la función del supervisor se reduce a indicar mediante una señal de refuerzo si la salida obtenida en la red se ajusta a la deseada (éxito = +1 o fracaso = -1), y en función de ello se ajustan los pesos basándose en un mecanismo de probabilidades. Se puede decir que en este tipo de aprendizaje la función del supervisor se asemeja más a la de un crítico (que opina sobre la respuesta de la red) que a la de un maestro (que indica a la red la respuesta concreta que debe generar), como ocurría en el caso de supervisión por corrección de error.

3) Aprendizaje estocástico

Este tipo de aprendizaje consiste básicamente en realizar cambios aleatorios en los valores de los pesos de las conexiones de la red y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidad.

En el aprendizaje estocástico se suele hacer una analogía en términos termodinámicos, asociando a la red neuronal con un sólido físico que tiene cierto estado energético. En el caso de la red, la energía de la misma representaría el grado de estabilidad de la red, de tal forma que el estado de mínima energía correspondería a una situación en la que los pesos de las conexiones consiguen que su funcionamiento sea el que más se ajusta al objetivo deseado.

Según esto, el aprendizaje consistiría en realizar un cambio aleatorio de los valores de los pesos y determinar la energía de la red (habitualmente la función energía es una función de Liapunov). Si la energía es menor después del cambio, es decir, si el comportamiento de la red se acerca al deseado, se acepta el cambio; si, por el contrario, la energía no es menor, se aceptaría el cambio en función de una determinada y preestablecida distribución de probabilidades.

En la Tabla 2 se resumen los principales modelos de RNA en función del tipo de aprendizaje supervisado que utilizan:

Tabla 2. Principales modelos de RNAs con aprendizaje supervisado [16]

Tipo de aprendizaje supervisado		Modelo de red
Aprendizaje por corrección de error	Off-line	Perceptron Adaline/Madaline BP Brain-State-in-a-Box
Aprendizaje por refuerzo	On-line	Linear Reward Penalty Adaptive Reward Penalty Adaptive Heuristic Critic
Aprendizaje estocástico	Off-line	Boltzmann Machine Cauchy Machine

2.6.2.2. Aprendizaje no supervisado

Las redes con aprendizaje no supervisado (también conocido como autosupervisado) no requieren influencia externa para ajustar los pesos de las conexiones entre las neuronas. La red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta. En la Figura 11 se muestra el proceso que se sigue en el aprendizaje no supervisado.

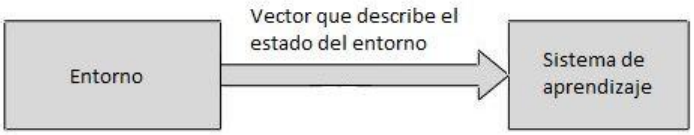


Figura 11. Diagrama de bloques del aprendizaje no supervisado

Estas redes deben encontrar las características, regularidades, correlaciones o categorías que se puedan establecer entre los datos que se presenten en su entrada. Existen varias posibilidades en cuanto a la interpretación de la salida de estas redes, que dependen de su estructura y del algoritmo de aprendizaje empleado.

En algunos casos, la salida representa el grado de familiaridad o similitud entre la información que se le está presentando en la entrada y las informaciones que se le han mostrado hasta entonces (en el pasado). En otro caso, podría realizar una clusterizacion (clustering) o establecimiento de categorías, indicando la red, a la salida, a qué categoría pertenece la información presentada a la entrada, siendo la propia red quién debe encontrar las categorías apropiadas a partir de las correlaciones entre las informaciones presentadas.

En cuanto a los algoritmos de aprendizaje no supervisado, en general se suelen considerar dos tipos, que dan lugar a los siguientes aprendizajes:

1) Aprendizaje hebbiano

Esta regla de entrada es la base de muchas otras, la cual pretende medir la familiaridad o extraer características de los datos de entrada. El fundamento es una suposición bastante simple: si dos neuronas N_i y N_j toman el mismo estado simultáneamente (ambas activas o ambas inactivas), el peso de la conexión entre ambas se incrementa.

Las entradas y salidas permitidas a la neurona son: {-1,1} o {0,1} (neuronas binarias). Esto puede explicarse porque la regla de aprendizaje de Hebb[1] se originó a partir de la neurona biológica clásica, que solamente puede tener dos estados: activa o inactiva.

2) Aprendizaje competitivo y comparativo

Se orienta a la clusterizacion o clasificación de los datos de entrada. Como característica principal del aprendizaje competitivo se puede decir que, si un patrón nuevo se determina que pertenece a una clase reconocida previamente, entonces la inclusión de este nuevo patrón a esta clase matizará la representación de la misma. Si el patrón de entrada se determinó que no pertenece a ninguna de las clases reconocida anteriormente, entonces la estructura y los pesos de la red neuronal serán ajustados para reconocer la nueva clase.

En la Tabla 3 se resumen los principales modelos de RNA en función del tipo de aprendizaje no supervisado que utilizan:

Tabla 3. Principales modelos de RNAs con aprendizaje no supervisado [16]

Tipo de aprendizaje no supervisado	Modelo de red
Aprendizaje Hebbiano	Red Hopfield (1982) Additive Grossberg (1973) Learning Matrix (1961) Bidirectional Associative Memory (1988) Temporal Associative Memory (1972)
Aprendizaje competitivo y comparativo	Learning Vector Quantization (LVQ)

2.6.3. Elección del conjunto inicial de pesos

Antes de empezar el proceso de entrenamiento de la red se debe determinar un estado inicial, lo que significa inicializar un conjunto de pesos para las diversas conexiones entre las neuronas de la red neuronal. Estas conexiones pueden inicializarse otorgando un peso aleatorio a cada una, encontrándose los mismos dentro de un cierto intervalo (del tipo [-n, n], donde n es un número natural positivo).

Durante el transcurso del entrenamiento los pesos no se encuentran restringidos a dicho intervalo.

[1] Regla de Hebb: Los pesos que unen a dos neuronas aumentan si se activan al mismo tiempo y disminuyen si se activan en distinto momento.

2.6.4. Detención del proceso de aprendizaje

Para determinar cuándo se detendrá el proceso de aprendizaje, es necesario establecer una condición de detención.

Una condición para detener el entrenamiento puede ser cuándo el cálculo del error cuadrático sobre todos los ejemplos del entrenamiento haya alcanzado un mínimo o cuándo para cada uno de los ejemplos dados, el error observado esté por debajo de un determinado umbral. Otra condición de detención del aprendizaje puede ser cuando se hayan completado un cierto número de iteraciones de entrenamiento.

Luego de alcanzarse la condición de detención, los pesos no se volverán a cambiar. Entonces se puede decir que la transformación de los datos de entrada a los de salida está resuelta. Esto se puede interpretar como una función ¨f¨ oculta en el conjunto de la red neuronal. Esta función es exactamente la ¨instrucción¨ de cómo la salida será calculada a partir de una constelación (vector) de entrada.

2.6.5. Codificación de los datos de entrada

Los datos tienen que ser codificados, es decir, deben hallarse valores apropiados para representar las características simbólicas. Se distinguen dos tipos de variables a ser codificadas:

- Variables o atributos numéricos (frecuentemente llamadas continuas)
- Variables o atributos simbólicos (frecuentemente llamados discretos)

Un atributo numérico es aquel que puede tomar cualquier valor dentro de un cierto intervalo [a,b]; donde ¨a¨ puede ser - ∞ (menos infinito) y ¨b¨, ∞ (infinito). Pero si los pesos son dados por un cierto número de términos, semejantes a alto o bajo; entonces el atributo se denomina simbólico. Por lo tanto, dividiendo el intervalo [a,b] de una variable numérica dentro de subintervalos, podemos confeccionar un atributo continuo pseudodiscreto.

En la Tabla 4 se presentan los procedimientos para la codificación de los datos de entrada:

Tabla 4. Resumen de los procedimientos de codificación [16]

Nombre	Valores	Procedimiento de codificación
Variables numéricas	Numéricos	Función de transformación lineal
Variables simbólicas		
• Sin orden	n valores simbólicos	Cada valor simbólico se corresponde con una neurona de entrada binaria
• Con orden		Cada valor simbólico se codifica como un segmento del intervalo de codificación
Variable pseudodiscretas	Numéricos, pero dividido dentro de T subintervalos	Cada subintervalo corresponde a una neurona binaria.

2.6.6. Validación de la red neuronal

Después del proceso de entrenamiento los pesos de las conexiones en la red neuronal quedan fijos. El siguiente paso debe ser comprobar si la red neuronal puede resolver nuevos problemas, del tipo general, para los que ha sido entrenado. Por lo tanto con el propósito de validar la red neuronal se requiere de otro conjunto de datos, denominado conjunto de validación o testeo.

Cada ejemplo del conjunto de evaluación contiene los valores de las variables de entrada, con su correspondiente solución tomada; pero ahora esta solución no se le es otorgada a la red neuronal. Luego se compara la solución calculada para cada ejemplo de validación con la solución conocida.

2.6.6.1. Procesos para evitar el *sobreaprendizaje*

En el proceso de entrenamiento de la red se debe considerar, junto al error cometido respecto al conjunto de patrones de aprendizaje (¨error de aprendizaje¨), el denominado error de generalización, que se puede medir utilizando un conjunto de test distinto a la muestra de entrenamiento.

De esta forma, interesa más una buena generalización de la red (bajo error en test fuera de la submuestra de aprendizaje), que la consecución de un error muy reducido en la submuestra de entrenamiento, ya que ello será indicativo de que el sistema ha capturado correctamente las relaciones subyacentes en los datos.

Experimentalmente se sabe que, si se entrena la red para alcanzar un error de aprendizaje muy reducido (por ejemplo, inferior al 1%), el error en testeo se degrada, obteniéndose una gráfica como la que se muestra en la figura 12.

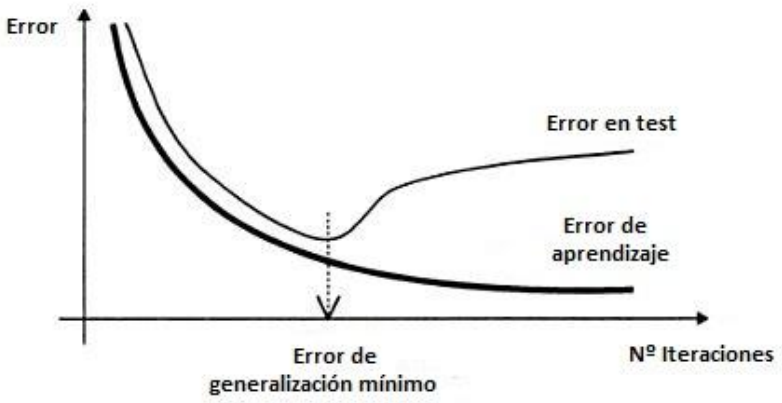


Figura 12. Análisis del fenómeno de sobreaprendizaje de la red [16]

Tal y como se observa, tras una etapa inicial, en la que la tasa de error puede oscilar, el error de aprendizaje disminuye monótonamente, mientras que el error en test disminuye únicamente hasta cierto punto, para incrementarse después, como consecuencia del excesivo ajuste de la red a las particularidades de los patrones empleados en el entrenamiento, que comienzan a ser memorizados en perjuicio de la capacidad de generalización del sistema.

Este fenómeno, conocido como sobreaprendizaje (overtraining), puede evitarse utilizando procesos de validación cruzada (cross validation), esto es, entrenando y validando la red a la vez, a fin de detectar el punto óptimo de aprendizaje. Estos procesos de validación son ampliamente aplicados en la fase de desarrollo de redes supervisadas (como el Perceptron Multicapa).

De esta forma, una vez entrenada la RNA resulta necesario evaluar los resultados obtenidos para determinar su validez práctica y su capacidad de generalización.

Existen dos grandes grupos de criterios para la evaluación de las RNA:

1) Criterios ¨dentro de la muestra¨

Los criterios ¨dentro de la muestra¨ tratan de analizar la capacidad de la RNA para caracterizar correctamente a la muestra de datos utilizada para su entrenamiento. Si bien se han propuesto distintas medidas alternativas de desempeño, cabe destacar las siguientes:

- Los coeficiente de correlación múltiple cuadrática R^2 y R^2 ajustada[2], indicativos de la proximidad existente entre las salidas generadas por la red (y_i) y las salidas deseadas (t_i):

$$R^2 = \frac{\sum_{i=1}^{p}(y_i - \bar{t})^2}{\sum_{i=1}^{p}(t_i - \bar{t})^2} = 1 - \frac{\sum_{i=1}^{p}(t_i - y_i)^2}{\sum_{i=1}^{p}(t_i - \bar{t})^2} \quad (2.10)$$

$$R^2_{ajustada} = \frac{P.R^2 - N}{P - N} \quad (2.11)$$

siendo P el total de individuos empleados para el aprendizaje del sistema (i = 1,…,P), $\bar{t}$ la salida media esperada para el conjunto de individuos analizados y N el número de variables explicativas incluidas en el modelo. Estos coeficientes toman valores en el intervalo [0,1], donde 0 indica la inexistencia de correlación entre la variable dependiente y el modelo desarrollado, mientras que 1 informa de la existencia de correlación perfecta.

Por otra parte, el criterio de información ¨Hannan-Quinn¨ resulta muy útil para la evaluación de modelos autorregresivos. Este criterio incluye un término de penalización

[2] Este ajuste trata de evitar la mejora artificial que se producen el coeficiente R^2 cuando se incremente el número de patrones del modelo.

que considera el número de parámetros del modelo (k), siendo preferible la RNA que minimice la expresión:

$$H\&Q = \left[\ln\left(\sum_{i=1}^{p}\frac{(t_i - y_i)^2}{P}\right)\right] + \frac{k\,\{\ln[\ln(P)]\}}{P} \qquad (2.12)$$

Otras dos medidas de validación ¨dentro de la muestra¨ que incluyen términos de penalización son el criterio de Akaike y el criterio de Schwartz:

$$Akaike = \left[\ln\left(\sum_{i=1}^{p}\frac{(t_i - y_i)^2}{P}\right)\right] + \frac{2k}{P} \qquad (2.13)$$

$$Shwartz = \left[\ln\left(\sum_{i=1}^{p}\frac{(t_i - y_i)^2}{P}\right)\right] + \frac{k.\ln(P)}{P} \qquad (2.14)$$

Por último, el análisis de los residuos o diferencia entre la salida deseada y la salida obtenida por la RNA para cada patrón, puede proporcionar información muy valiosa sobre la presencia de sesgo en el modelo (distribución sistemática de residuos), simetría (análisis de aleatoriedad de los residuos) y normalidad (presencia o no de ruido blanco en la distribución de los residuos).

2) Criterios ¨fuera de muestra¨

Existen distintos criterios que analizan la capacidad de generalización de las RNA, o capacidad para responder correctamente ante la presencia de patrones nuevos. Para ellos, resulta necesario definir la función de pérdida L a utilizar para estimar el error de predicción cometido por el modelo, siendo las más habituales el error absoluto o cuadrático (problemas de aproximación de funciones) y el error total de clasificación procedente de tablas de contingencia, véase Tabla 5.

Tabla 5. Funciones de error de predicción [16]

Función de error	Definición
Error medio absoluto (¨mean absolute error¨)	$MAE = \frac{\sum_{i=1}^{P}\sum_{j=1}^{M}\lvert y_{ij} - t_{ij}\rvert}{P}$
Error cuadrático medio (¨mean squared error¨)	$MSE = \frac{\sum_{i=1}^{P}\sum_{j=1}^{M}(y_{ij} - t_{ij})^2}{P}$
Raíz del error cuadrático medio (¨root mean squared error¨)	$RMSE = \sqrt{\frac{\sum_{i=1}^{P}\sum_{j=1}^{M}(y_{ij} - t_{ij})^2}{P}}$

Asimismo, se han definido distintas variantes del error cuadrático medio para modelos lineales, tales como el criterio de validación cruzada generalizada o el error de predicción al cuadrado. En el caso de modelos no lineales, la variante más destacada es la medida llamada error de predicción generalizado.

Una vez definida la función de error a utilizar pueden distinguirse distintos criterios de validación fuera de muestra, entre los que destacan:

- Error aparente o de resustitución (apparent error o resubstitutuion error).
- División de los datos o técnicas de ¨entrenamiento y test¨ (test-and-train).
- Modelos de resampling.

2.7. Principales topologías de las RNA

La topología o arquitectura de una red neuronal consiste en la organización y disposición de las neuronas en la misma, formando capas o agrupaciones de neuronas más o menos alejadas de la entrada y salida de dicha red. En este sentido, los parámetros fundamentales de la red son: el número de capas, el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre neuronas.

2.7.1. Redes monocapa

En las redes monocapa, se establecen conexiones entre las neuronas que pertenecen a la única capa que constituye la red. Las redes monocapas se utilizan generalmente en tareas relacionadas con lo que se conoce como autoasociación (regenerar información de entrada que se presenta a la red de forma incompleta o distorsionada).

2.7.2. Redes multicapa

Las redes multicapa son aquellas que disponen de un conjunto de neuronas agrupadas en varios (2, 3, etc.) niveles o capas. En estos casos, una forma para distinguir la capa a la que pertenece una neurona, consistiría en fijarse en el origen de las señales que recibe a la entrada y el destino de la señal de salida. Normalmente, todas las neuronas de una capa reciben señales de entrada desde otra capa anterior (la cual está más cerca a la entrada de la red), y envían señales de salida a una capa posterior (que está más cerca a la salida de la red). A estas conexiones se las denomina conexiones hacia adelante o feedforward.

No obstante, en un gran número de estas redes también existe la posibilidad de conectar la salida de las neuronas de las capas posteriores a la entrada de capas anteriores; a estas conexiones se las denomina conexiones hacia atrás o feedback.

Estas dos posibilidades permiten distinguir entre dos tipos de redes con múltiples capas: las redes con conexiones hacia adelante o redes feedforward, y las redes que disponen de conexiones tanto hacia adelante como hacia atrás o redes feedforward/feedback.

2.7.3. Conexión entre neuronas

La conectividad entre los nodos de una red neuronal está relacionada con la forma en que las salidas de las neuronas están canalizadas para convertirse en entradas de otras neuronas. La señal de salida de un nodo puede ser una entrada de otro elemento de proceso, o incluso ser una entrada de sí mismo (conexión autorecurrente).

Cuando ninguna salida de las neuronas es entrada de neuronas del mismo nivel o de niveles precedentes, la red se describe como de conexión hacia delante (ver Figura 2). Cuando las salidas pueden ser conectadas como entradas de neuronas de niveles previos o del mismo nivel, incluyéndose ellas mismas, la red es de conexión hacia atrás.

Las redes de propagación hacia atrás que tienen lazos cerrados son llamadas: sistemas recurrentes.

2.7.4. Redes de propagación hacia atrás (backpropagation)

El nombre de backpropagation resulta de la forma en que el error es propagado hacia atrás a través de la red neuronal, en otras palabras el error se propaga hacia atrás desde la capa de salida. Esto permite que los pesos sobre las conexiones de las neuronas ubicadas en las capas ocultas cambien durante el entrenamiento.

El cambio de los pesos en las conexiones de las neuronas además de influir sobre la entrada global, influye en la activación y por tanto en la salida de una neurona. Por lo tanto, es de gran utilidad considerar la variación de la función activación al modificarse el valor de los pesos. Esto se llama sensibilidad de la función activación, de acuerdo al cambio en los pesos.

2.8. Principales modelos de RNA

Los principales modelos de redes neuronales existentes en la actualidad son los siguientes: el Perceptron Simple, el Asociador Lineal, las Redes Adaline y Madaline, la Red Hopfield, las Funciones de base radial, los Mapas Autoorganizados de Características de Kohonen y el Perceptron Multicapa. A continuación se describen las características de cada uno de los modelos, prestando especial atención al Perceptron Multicapa el cual es el modelo utilizado para alcanzar el objetivo propuesto en este trabajo.

- El Perceptron Simple es el primer modelo de red neuronal que existe (apartado 2.2), el cual permite discriminar entre dos clases linealmente separables (el denominado problema OR), pero no entre clases linealmente no separables (problema OR exclusivo o XOR). Puede utilizarse tanto para tareas de clasificación como para la representación de funciones booleanas.
- El Asociador Lineal constituye uno de los principales modelos de redes neuronales unidireccionales y está basado en el aprendizaje Hebbiano. Este modelo utiliza como función de activación la función lineal.

- La red ADALINE (ADAptive LINear Elemente) y la red MADALINE (Multiple ADALINE) fueron desarrolladas por Widrow y Hoff. ADALINE se desarrollo para el reconocimiento de patrones y es un dispositivo que consta de un solo elemento de procesamiento por lo que técnicamente no es una red, sin embargo es un elemento muy importante ya que de él se derivan redes más complejas. MADALINE consta de una capa de ADALINE y una función de mayoría cuya respuesta binaria depende de las respuestas de las ADALINE. Su principal inconveniente es que el proceso de entrenamiento es muy lento.
- La red Hopfield es una red recurrente y completamente interconectada. Funciona como una memoria asociativa no lineal. De esta forma puede ser usada como una herramienta de optimización o para el reconocimiento de patrones. En esta red los pesos se pueden calcular y se mantienen fijos durante el aprendizaje de los patrones. Solamente cambia el estado de las neuronas.
- Las funciones de base radial constituyen un modelo cada vez más utilizado debido a su simplicidad, generalidad y rapidez de aprendizaje. Permiten modelar sistemas no lineales. El aprendizaje de este tipo de redes requiere establecer a priori el número de nodos ocultos, para lo que se puede hacer uso de métodos de prueba y error, si bien resulta muy útil la aplicación de algún modelo constructivo, como el algoritmo auto organizado jerárquico (¨Hierarchically Self-Organizing Learning Algorithm¨).
- Los Mapas Autoorganizados de Características de Kohonen incorporan a su estructura interna de conexiones rasgos comunes, regularidades, correlaciones o categorías descubiertas en los datos de entrada. Este tipo de red posee un aprendizaje no supervisado competitivo (ver apartado 2.6.2.2). El objetivo de este aprendizaje es categorizar los datos que se introducen en la red. En este modelo las conexiones entre las dos capas que forman la red son siempre hacia adelante.

Puede encontrarse una descripción detallada de estos modelos en [16].

2.8.1. El Perceptron Multicapa

2.8.1.1. Introducción

El Perceptron Multicapa es una red neuronal artificial formada por múltiples capas, esto le permite resolver problemas que no son linealmente separables, lo cual es la principal limitación del Perceptron, como ya se explicó anteriormente. El Perceptron Multicapa puede ser total o localmente conectado. En el primer caso cada salida de una neurona de la capa i es entrada de todas las neuronas de la capa i +1, mientras que en segundo caso cada neurona de la capa i es entrada de una serie de neuronas (región) de la capa i +1.

La propagación hacia atrás, es un algoritmo utilizado en el entrenamiento de estas redes, por ello, el Perceptron Multicapa también es conocido como red de retropropagación.

En esta red las funciones de transferencia de los elementos de procesado (neuronas) tienen que ser derivables.

En cuanto a sus limitaciones cabe destacar:

- El Perceptron Multicapa no extrapola bien, es decir, si la red se entrena mal o de manera insuficiente, las salidas pueden ser imprecisas.
- La existencia de mínimos locales en la función de error dificulta considerablemente el entrenamiento, pues una vez alcanzado un mínimo, el entrenamiento se detiene aunque no se haya alcanzado la tasa de convergencia fijada.

Cuando se cae en un mínimo local sin satisfacer el porcentaje de error permitido se puede considerar: cambiar la topología de la red (número de capas y número de neuronas), comenzar el entrenamiento con unos pesos iniciales diferentes, modificar los parámetros de aprendizaje, modificar el conjunto de entrenamiento o presentar los patrones en otro orden.

El Perceptron Multicapa (MLP: Multiplayer Perceptron) se utiliza para resolver problemas de estimaciones, de asociación de patrones, segmentación de imágenes, compresión de datos, etc. En la Figura 13 se presenta un ejemplo de la estructura del Perceptron Multicapa:

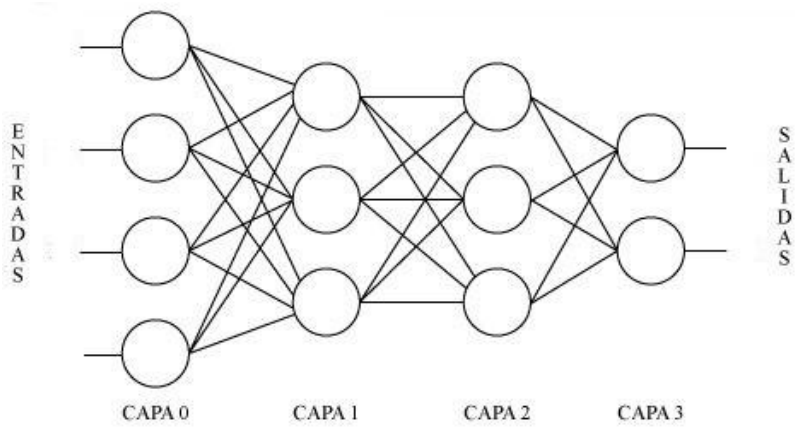


Figura 13. Perceptron Multicapa con dos capas ocultas [16]

2.8.1.2. Algoritmo BP de entrenamiento del Perceptron Multicapa

Tal y como se menciona en el apartado 2.1 el algoritmo básico de aprendizaje que usa el Perceptron multicapa es el algoritmo BP, mediante el cual se van adaptando todos los parámetros de la red**.** Este algoritmo realiza dos pasadas a través de las diferentes capas de la red: una fase de avance y otra fase de retroceso (ver Figura 14). Inicialmente, se aplica un vector de entrada a los nodos, y su efecto se propaga a través de las capas de la red, lo que finalmente produce un conjunto un de salidas como respuesta de la red. Durante la fase de avance los pesos de la red son fijos, mientras que durante la fase de retroceso estos pesos se ajustan de acuerdo con una regla de corrección de errores. En concreto, se genera una señal de error cuyo valor será la diferencia entre la salida de la red y el objetivo a estimar. Este error se propaga hacia atrás a través de la red en sentido contrario al de las conexiones sinápticas,

de ahí el nombre de error BP. Los pesos de todas las conexiones se ajustan para que la respuesta de la red se acerque a la respuesta deseada.

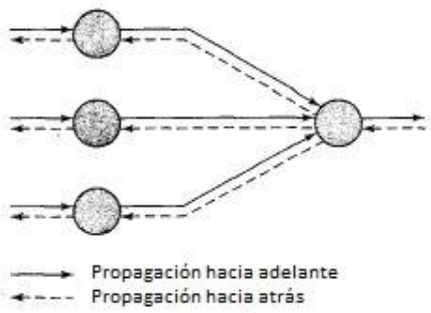


Figura 14. Fases de aprendizaje del algoritmo BP [20]

El tipo de entrenamiento que sigue este tipo de algoritmo es el entrenamiento supervisado (ver apartado 2.6.2.1). El aprendizaje de la red se plantea como un problema de minimización de una determinada función de error. Para facilitar la carga matemática que participa en este problema, a continuación se presenta la notación utilizada:

- Los índices *i, j* y *k* se refieren a las diferentes neuronas en la red; si la señal se propaga a través de la red de izquierda a derecha, la neurona j se sitúa en una capa a la derecha de la neurona i, y la neurona k se sitúa en una capa a la derecha de la neurona j, estando la neurona j en una capa oculta.
- En la iteración n, se introduce a la red el patrón de entrenamiento nth.
- El símbolo $E(n)$ se refiere a la suma instantánea del error cuadrático en la iteración n. La media de $E(n)$ sobre todos los valores de n produce el error promedio E_{av}.
- El símbolo $e_j(n)$ se refiere al error de la señal en la salida de la neurona j en la iteración n.
- El símbolo $d_j(n)$ se refiere a la respuesta deseada para la neurona j y se usa para calcular $e_j(n)$.
- El símbolo $y_j(n)$ se refiere a la señal de la función que aparece a la salida de la neurona j en la iteración n.
- El símbolo $w_{ji}(n)$ denota el peso sináptico existente entre la salida de la neurona i y la entrada de la neurona j en la iteración n. La corrección que se aplica a este peso en la iteración se denota como $\Delta w_{ji}(n)$.
- La suma de todos los pesos más los sesgos conectados a la neurona j en la iteración n se denota como $v_j(n)$; esta suma constituye la señal que se aplica a la función de activación asociada a la neurona j.
- La función de activación asociada con la neurona j se denota cómo $\varphi_j(\cdot)$.
- El sesgo aplicado a la neurona j se denota como b_j; su efecto se representa por una sinapsis de peso $w_{j0} = b_j$ conectado a una entrada fija igual a +1.

- El elemento *i*th del vector de entrada se denota como $x_i(n)$.
- El elemento *k*th del vector de salida global se denota como $o_k(n)$.
- El parámetro de la tasa de aprendizaje se denota como η.
- El símbolo m_l denota el tamaño (número de nodos) en la capa l del Perceptron Multicapa (l=0,1,…,L); por tanto, m_0 denota el tamaño de la primera capa oculta y m_L el tamaño de la capa de salida. Tambien se usa la notación $m_L = M$.

Una vez introducida la notación, se presentan las ecuaciones que rigen el algoritmo BP. El error a la salida de la neurona j en la iteración n se define como:

$$e_j(n) = d_j(n) - y_i(n) \tag{2.15}$$

Se define el valor instantáneo de la energía de error para la neurona j como $\frac{1}{2}e_j^2(n)$. Correspondientemente, el valor instantáneo $E(n)$ de la energía del error total se obtiene el valor $\frac{1}{2}e_j^2(n)$ de todas las neuronas en la capa de salida; estas son las únicas neuronas visibles cuyos errores de señal pueden ser calculados directamente. Por tanto, se puede escribir:

$$E(n) = \frac{1}{2}\sum_{j\epsilon C} e_j^2(n) \tag{2.16}$$

donde el conjunto C incluye todas las neuronas en la capa de salida de la red. Se establece que N denote el número total de patrones (ejemplos) contenidos en el conjunto de entrenamiento. La energía del error cuadrático medio se obtiene sumando $E(n)$ sobre todo n y después normalizando con respecto al tamaño del conjunto N, como se muestra en la Ec. (2.17):

$$E_{av} = \frac{1}{N}\sum_{n=1}^{N} E(n) \tag{2.17}$$

La energía del error instantáneo $E(n)$, y por tanto la energía del error medio E_{av}, es una función de todos los parámetros libres (pesos sinápticos y sesgos) de la red. Para un conjunto de entrenamiento dado, E_{av} representa la función de coste como una medida del rendimiento de aprendizaje. El objetivo del proceso de aprendizaje es ajustar estos parámetros libres de la red para minimizar E_{av}.

Por tanto, la media aritmética de estos cambios individuales de los pesos a lo largo del conjunto de entrenamiento es una estimación del verdadero cambio que puede resultar de la modificación de los pesos basándose en la minimización de la función de coste E_{av} sobre todo el conjunto de entrenamiento.

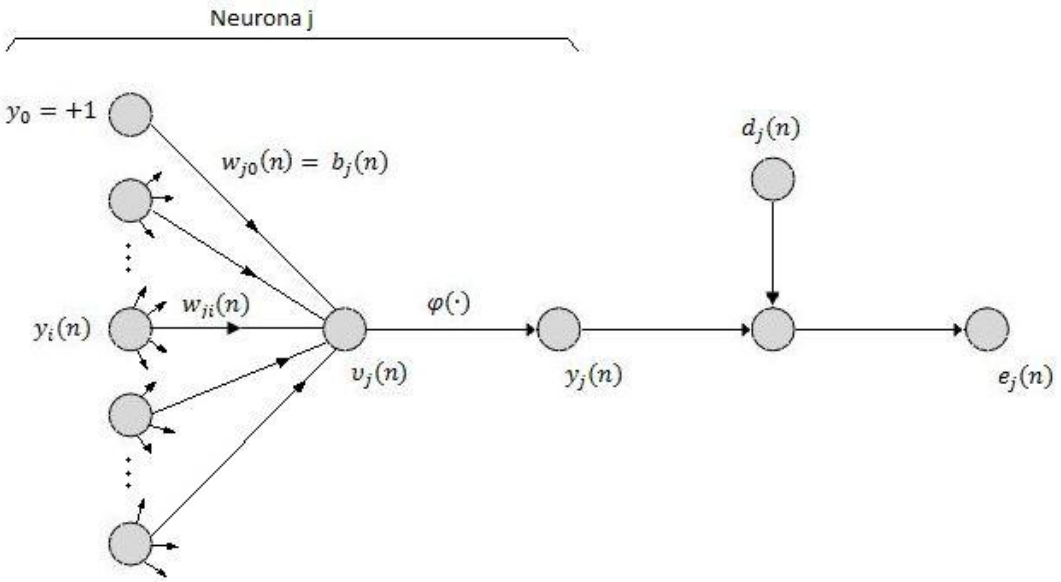


Figura 15. Gráfico del flujo de señal de la neurona j [20]

En la Figura 15 se representa una neurona j a la que entran un conjunto de señales procedentes de una capa de neuronas situadas a su izquierda. Por lo tanto, el sumatorio $v_j(n)$ que se produce antes de la función de activación asociada a la neurona j es:

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \tag{2.18}$$

donde m es el número total de entradas (excluyendo los sesgos) a la neurona j. El peso sináptico w_{jo} (correspondiente a la entrada fija $y_0 = +1$) es igual al sesgo b_j aplicado a la neurona j. Por tanto la la señal de la función $y_j(n)$ que aparece a la salida de la neurona j en la iteración n es:

$$y_j(n) = \varphi_j\left(v_j(n)\right) \tag{2.19}$$

El algoritmo BP aplica la corrección $\Delta w_{ji}(n)$ al peso $w_{ji}(n)$, el cual es proporcional a la derivada parcial $\partial E(n)/\partial w_{ji}(n)$. De acuerdo con la regla de la cadena, se puede expresar este gradiente como:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{2.20}$$

La derivada parcial $\partial E(n)/\partial w_{ji}(n)$ representa un factor de sensibilidad. Derivando ambos lados de la Ec. (2.16) con respecto a $e_j(n)$, se tiene:

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \tag{2.21}$$

Derivando ambos lados de la Ec. (2.15) con respecto a $y_j(n)$, se tiene:

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{2.22}$$

A continuación, derivando la Ec. (2.19) con respecto a $v_j(n)$, se tiene:

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'\left(v_j(n)\right) \tag{2.23}$$

Finalmente, derivando la Ec. (2.18) con respecto a $w_{ji}(n)$ se obtiene:

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{2.24}$$

Introduciendo las Ecs. (2.11) a (2.14) en (2.20) se obtiene:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'\left(v_j(n)\right) y_i(n) \tag{2.25}$$

La corrección $\Delta w_{ji}(n)$ aplicada a $w_{ji}(n)$ se define por la regla delta:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)} \tag{2.26}$$

donde η es el parámetro de la tasa de aprendizaje del algoritmo BP. El uso del signo menos en la Ec. (2.26) representa el descenso del gradiente en el espacio de los pesos (búsqueda de una dirección para el cambio de pesos que reduzca el valor de $E(n)$. Por tanto, el uso de la Ec. (2.25) en (2.26) establece:

$$\Delta w_{ij}(n) = \eta \delta_j(n) y_i(n) \tag{2.27}$$

donde el gradiente local $\delta_j(n)$ se define como:

$$\begin{aligned} \delta_j(n) &= -\frac{\partial E(n)}{\partial v_j(n)} \\ &= -\frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \\ &= e_j(n)\varphi_j'\left(v_j(n)\right) \end{aligned} \tag{2.28}$$

El gradiente local requiere cambio en los pesos sinápticos. De acuerdo con la Ec. (2.28), el gradiente local $\delta_j(n)$ para la neurona j es igual al producto del correspondiente error de señal $e_j(n)$ para esa neurona con la derivada $\varphi_j'(v_j(n))$ de la función de activación asociada.

A partir de las Ecs. (2.27) y (2.28) se observa que un factor clave que interviene en el cálculo del ajuste del peso $\Delta w_{ij}(n)$ es la señal de error $e_j(n)$ a la salida de la neurona j. En este contexto, se pueden identificar dos casos distintos, dependiendo de donde se localice la neurona j en la red. Caso 1: la neurona j es un nodo de salida. Este caso es sencillo de manejar porque a cada nodo de salida se le ha proporcionado una salida deseada, pudiéndose calcular el error asociado de manera explicada anteriormente. Caso 2: la neurona j es un nodo oculto. Aunque las neuronas ocultas no son directamente accesibles, comparten responsabilidad con cualquier error que se comete a la salida de la red. La cuestión, por tanto, es saber cómo penalizar o premiar a dichas neuronas por su parte de responsabilidad.

Caso 1: La neurona j es un nodo de salida

Cuando la neurona j se sitúa en la capa de salida de la red, dispone de información de la salida deseada. Se usa la Ec. (2.15) para calcular el error de señal $e_j(n)$ asociado a esa neurona (ver Figura 15). Una vez determinado $e_j(n)$, se utiliza la Ec. (2.28) para calcular el gradiente local $\delta_{\mathrm{j}}(\mathrm{n})$.

Caso 2: La neurona j es un nodo oculto

Cuando la neurona j se sitúa en una capa oculta de la red no tiene especificada una salida deseada para esa neurona. Por tanto, el error de señal para una neurona oculta tiene que ser determinado de forma recursiva en función de las señales de error de todas las neuronas a las que esa neurona oculta está conectada directamente; en este punto es donde el desarrollo del algoritmo BP se complica. Se considera la situación descrita en la Figura 16, que representa a la neurona j como un nodo oculto de la red. De acuerdo con la Ec. (2.28), se puede redefinir el gradiente local $\delta_{\mathrm{j}}(\mathrm{n})$ para la neurona oculta j como:

$$\delta_{\mathrm{j}}(\mathrm{n}) = -\frac{\partial E(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} \tag{2.29}$$

$$= -\frac{\partial E(n)}{\partial y_j(n)}\,\varphi_j'(v_j(n))$$

donde en la segunda línea se ha usado la Ec. (2.23). Para calcular la derivada parcial $\partial E(n)/\partial y(n)$, se procede como sigue. De la Figura 16 se observa que:

$$E(n) = \frac{1}{2}\sum_{k\epsilon C} e_k^2(n) \tag{2.30}$$

la neurona k es un nodo de salida. Esta ecuación es la Ec. (2.16) sustituyendo el subíndice k en lugar del j. Se hace esto para evitar confusión con el uso del subíndice j que en el caso 2 se refiere a una neurona oculta. Derivando la Ec. (2.30) con respecto a la señal de la función $y_j(n)$ se tiene:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} \tag{2.31}$$

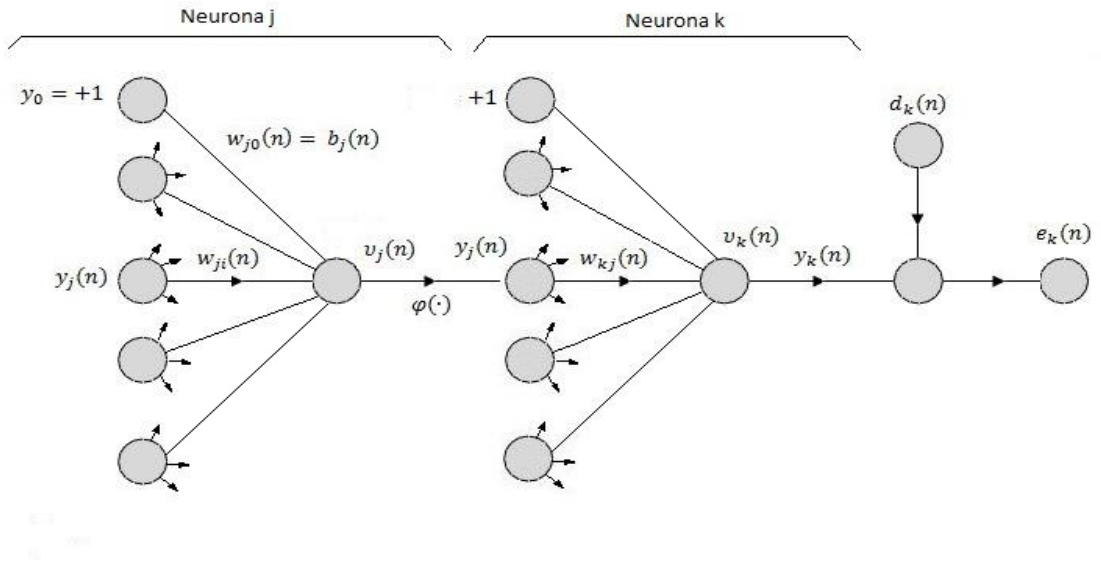


Figura 16. Gráfico del flujo de señal de la neurona de salida k conectada con la neurona oculta j [20]

A continuación se utiliza la regla de la cadena para la derivada parcial $\partial e_k(n)/\partial y_j(n)$, y se reescribe la Ec. (2.31) como:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)} \tag{2.32}$$

Sin embargo, de la Fig. XX, se tiene que:

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k\big(v_k(n)\big) \tag{2.33}$$

Por lo tanto:

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'\big(v_k(n)\big) \tag{2.34}$$

Tambien se observa de la Figura 16 que el sumatorio para la neurona k es:

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n) \tag{2.35}$$

Donde m es el número total de entradas (excluyendo los sesgos) aplicados a la neurona k. Igual que en el caso anterior, el peso sináptico $w_{k0}(n)$ es igual al sesgo $b_k(n)$ aplicado a la neurona k, y la correspondiente entrada se fija con el valor +1. Derivando la Ec. (2.35) con respecto a $y_j(n)$ se tiene:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n) \tag{2.36}$$

Utilizando las Ecs. (2.34) y (2.36) en (2.32) se obtiene la derivada parcial deseada:

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n)\,\varphi_k'\big(v_k(n)\big)w_{kj}(n) \tag{2.37}$$

$$= -\sum_k \delta_k(n)\,w_{kj}(n)$$

Donde en la segunda línea se utiliza la definición de gradiente local $\delta_k(n)$ dada en la Ec. (2.28) sustituyendo el índice j por el k.

Finalmente, utilizando la Ec. (2.37) en (2.29), se tiene la formula BP para el gradiente local $\delta_j(n)$:

$$\delta_j(n) = \varphi_j'\left(v_j(n)\right)\sum_k \delta_k(n)\,w_{kj}(n) \tag{2.38}$$

La Figura 17 muestra la representación del flujo de señal de la Ec. (2.38), asumiendo que la capa de salida consiste en m_L neuronas.

El factor $\varphi_j'(v_j(n))$ involucrado en el cálculo del gradiente local $\delta_j(n)$ en la Ec. (2.38) depende solamente de la función de activación asociada a la neurona oculta j. El factor restante en este cálculo, incluido en el sumatorio sobre k, depende de dos términos. El primer término, $\delta_k(n)$, requiere conocimiento de los errores de señal $e_k(n)$, para todas las neuronas que se encuentran en la capa inmediatamente a la derecha de la neurona oculta j, y para aquellas que están directamente conectadas a la neurona j (ver Figura 16). El segundo término, $w_{kj}(n)$, consiste en los pesos asociados a dichas conexiones.

A continuación se resumen las relaciones que se han establecido para el algoritmo BP. Primero, la corrección $\Delta w_{ji}(n)$ aplicada al peso sináptico que conecta la neurona i con la j se define mediante la regla delta:

$$\begin{pmatrix} Peso\ de\ la \\ conexión \\ \Delta w_{ji}(n) \end{pmatrix} = \begin{pmatrix} Parámetro\ de\ la \\ tasa\ de\ aprendizaje \\ \eta \end{pmatrix}.\begin{pmatrix} Gradiente \\ local \\ \delta_i(n) \end{pmatrix}.\begin{pmatrix} Señal\ de\ entrada \\ a\ la\ neurona\ j \\ y_i(n) \end{pmatrix} \tag{2.39}$$

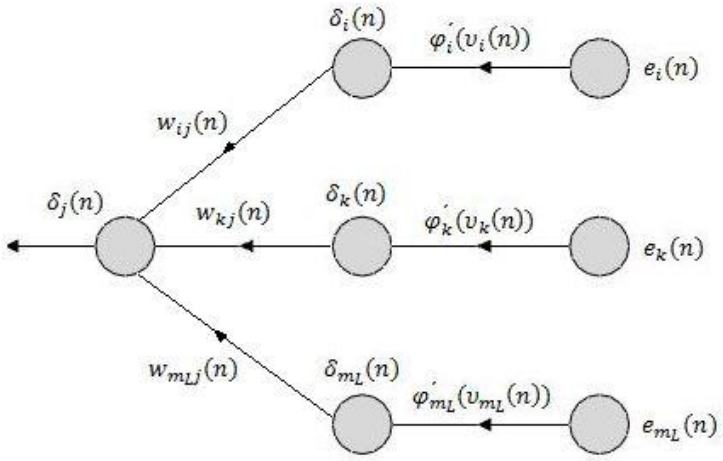


Figura 17. Gráfico del flujo de señal de una parte del sistema en la fase de retroceso [20]

Segundo, el gradiente local $\delta_j(n)$ depende de si la neurona j es un nodo de salida o un nodo oculto:

1. Si la neurona j es un nodo de salida, $\delta_j(n)$ es igual al producto de la derivada $\varphi_j'(v_j(n))$ por el error de señal $e_j(n)$, ambos asociados a la neurona j (ver Ec. 2.28).
2. Si la neurona j es un nodo oculto, $\delta_j(n)$ es igual al producto $\varphi_j'(v_j(n))$ por la suma de los pesos de los δ calculados para las neuronas de la siguiente capa oculta o de salida que estén conectadas a la neurona j (ver Ec. 2.38).

Los dos pasos en la computación

En la aplicación del algoritmo BP se distinguen dos pasos de computación. El primer paso se refiere a la fase de avance, y el segundo se refiere a la fase de retroceso.

En la fase de avance a través de la red, los pesos sinápticos se mantienen inalterados, y las señales de función de la red se calculan neurona por neurona. La señal de función que aparece a la salida de la neurona j se calcula como:

$$y_j(n) = \varphi\left(v_j(n)\right) \tag{2.40}$$

donde v_j(n) es el sumatorio de la entrada j, definido como:

$$v_j(\mathrm{n}) = \sum_{\mathrm{i}=0}^{\mathrm{m}} \mathrm{w_{ji}(n)} y_i(\mathrm{n}) \tag{2.41}$$

donde m es el número total de entradas (excluyendo el sesgo) aplicadas a la neurona j, $\mathrm{w_{ji}(n)}$ es el peso entre la conexión de la neurona i y la neurona j, y $y_i(\mathrm{n})$ es la señal de entrada a la neurona j, o equivalentemente la señal de la función que aparece a la salida de la

neurona j. Si la neurona j está en la primera capa oculta de la red, $m = m_0$ y el índice i se refiere al terminal de entrada *i*th de la red, para el que se tiene:

$$y_i(n) = x_i(n) \tag{2.42}$$

donde $x_i(n)$ es el elemento *i*th del vector de entrada. Si, por otra parte, la neurona j está en la capa de salida de la red, $m = m_L$ y el índice j se refiere al terminal de salida *j*th de la red, para que se tiene:

$$y_j(n) = o_j(n) \tag{2.43}$$

donde $o_j(n)$ es el elemento *j*th del vector de salida. Esta salida se compara con la respuesta deseada $d_j(n)$, obteniendo el error de señal $e_j(n)$ para la neurona de salida *j*th. Por lo tanto la fase de avance comienza presentando el vector de entrada a la primera capa oculta, y termina en la capa de salida mediante el cálculo de la señal de error para cada neurona.

Por otra parte, la fase de retroceso comienza en la capa de salida haciendo pasar las señales de error a través de la red de derecha a izquierda, capa a capa y calculando el gradiente local (δ) para cada neurona. Este proceso recursivo permite que los pesos de la red se sometan a cambios de acuerdo la regla delta de la Ec. (2.39). Para una neurona localizada en la capa de salida, δ es simplemente igual al error de esa neurona multiplicado por la primera derivada de su no-linealidad. Por lo tanto, se usa la Ec. (2.39) para calcular el cambio de los pesos de todas las conexiones que llegan a la capa de salida. Dados los gradientes locales para las neuronas de la capa de salida, se utiliza la Ec. (2.38) para calcular los gradientes de todas las neuronas en la penúltima capa y por tanto los cambios en todos los pesos de las conexiones que llegan a dicha capa. Este cálculo recursivo se continúa, capa a capa, propagando los cambios a todos los pesos de la red.

Función de activación

El cálculo del δ de cada neurona del Perceptron Multicapa requiere conocimientos sobre la derivada de la función de activación asociada a cada neurona. Es necesario que la función $\varphi(\cdot)$ sea continua para que exista su derivada. Básicamente, el único requisito que debe cumplir una función de activación es que sea derivable. Una función diferenciable no-lineal que se utilizan comúnmente en el Perceptron Multicapa es la función sigmoidal. A continuación se describen dos formas de la misma:

1. Función logística. Esta forma se define como:

$$\varphi_j(v_j(n)) = \frac{1}{1 + \exp\left(-av_j(n)\right)} \tag{2.44}$$

$$a > 0 \; y \; -\infty < v_j(n) < \infty$$

donde $v_j(n)$ es el sumatorio a la entrada de la neurona j. De acuerdo con esta no-linealidad, la amplitud de la salida se encuentra en el rango $0 \leq y_j \leq 1$. Derivando la Ec. (2.44) con respecto a $v_j(n)$, se tiene:

$$\varphi_j^{'}\left(v_j(n)\right) = \frac{a.\exp\left(-av_j(n)\right)}{[1+\exp(-av_j(n))]^2} \tag{2.45}$$

Con $y_j(n) = \varphi_j(v_j(n))$, se puede eliminar el término $\exp\left(-av_j(n)\right)$ de la Ec. (2.45), y así expresar la derivada $\varphi_j^{'}(v_j(n))$ como:

$$\varphi_j^{'}(v_j(n)) = ay_j(n)\left[1-y_j(n)\right] \tag{2.46}$$

Para una neurona localizada en la capa de salida, $y_j(n) = o_j(n)$. Por lo tanto, se puede expresar el gradiente local de la neurona j como:

$$\delta_j(n) = e_j(n)\varphi_j^{'}(v_j(n)) = a\left[d_j(n)-o_j(n)\right]o_j(n)\left[1-o_j(n)\right] \tag{2.47}$$

Por otra parte, para una neurona j cualquiera de la capa oculta, se puede expresar el gradiente como:

$$\delta_j(n) = \varphi_j^{'}(v_j(n))\sum_k \delta_k(n)w_{kj}\,(n) = ay_j(n)\left[1-y_j(n)\right]\sum_k \delta_k(n)w_{kj}\,(n) \tag{2.48}$$

En la Ec. (2.46) la derivada $\varphi_j^{'}(v_j(n))$ alcanza su máximo valor en $y_j(n) = 0{,}5$, y su mínimo valor (cero) en $y_j(n) = 0$. Dado que la cantidad del cambio en los pesos de la red es proporcional a la derivada $\varphi_j^{'}(v_j(n))$, se deduce que para una función de activación sigmoidal los pesos cambian más en las neuronas de la red en las que las señales de la función se encuentran en valores medios. Para Rumelhart, esta característica del aprendizaje BP contribuye a su estabilización como algoritmo de entrenamiento.

2. Función tangente hiperbólica. Esta forma se define como:

$$\varphi_j(v_j(n)) = a\tanh(bv_j(n)), \qquad (a,b) > 0 \tag{2.49}$$

donde a y b son constantes. Su derivada con respecto a $v_j(n)$ es dada por:

$$\varphi_j^{'}(v_j(n)) = ab\,\mathrm{sech}^2(bv_j(n)) = ab(1-\tanh^2(bv_j(n))) = \tag{2.50}$$

$$= \frac{b}{a}\left[a-y_j(n)\right]\left[a+(y_j(n)\right]$$

Para una neurona j localizada en la capa de salida, el gradiente local es:

$$\delta_j(n) = \; e_j(n)\varphi_j^{'}(v_j(n)) = \frac{b}{a}[d_j(n)-o_j(n)][a-o_j(n)][a+o_j(n)] \tag{2.51}$$

Para una neurona j de la capa oculta se tiene:

$$\delta_j(n) = \frac{b}{a}[a - y_j(n)][a + y_j(n)] \sum_k \delta_k(n) w_{kj}(n) \tag{2.52}$$

Usando las Ecs. (2.47) y (2.48) para la función logística y las Ecs. (2.51) y (2.52) para la función tangente hiperbólica, se puede calcular el gradiente local δ_j sin tener conocimiento explicito de la función de activación.

Ritmo de aprendizaje

El algoritmo BP proporciona una aproximación a la trayectoria en el espacio de los pesos por el método de descenso más rápido. Cuanto más pequeño es el parámetro de la tasa (ritmo) de aprendizaje η, más pequeños son los cambios en los pesos de la red en cada iteración y por tanto más suave será la trayectoria en el espacio de los pesos. Esta mejora, se consigue a cambio de ritmo de aprendizaje. Si, por otra parte, se establece un parámetro de la tasa de aprendizaje muy grande para aumentar el ritmo de aprendizaje, los cambios resultantes en los pesos son tan grandes que la red puede volverse inestable. Un método sencillo de incrementar el ritmo de aprendizaje evitando el peligro de inestabilidad consiste en modificar la regla delta de la Ec. (2.27) incluyendo el siguiente término $\alpha \Delta w_{ji}(n-1)$:

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \tag{2.53}$$

donde α es normalmente un número positivo llamado cte. momento. Esta controla el bucle de retroalimentación que actúa alrededor de $\Delta w_{ji}(n)$, tal y como se muestra en la Figura X, donde z^{-1} es el operador de la unidad de retardo. A la Ec. (2.53) se la conoce como regla delta generalizada.

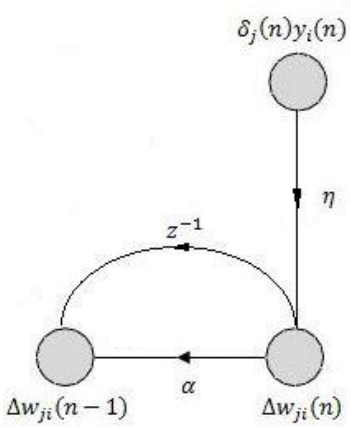


Figura 18. Gráfico del flujo de señal ilustrando el efecto del momento constante $\boldsymbol{\alpha}$

Se puede reescribir la Ec. (2.53) como una serie de tiempo con el índice t. Este índice abarca desde el tiempo inicial 0 hasta el tiempo presente n. La Ec. (2.53) puede ser vista como una

ecuación diferencial de primer orden en la corrección del peso $\Delta w_{ji}(n)$. Resolviendo esta ecuación para $\Delta w_{ji}(n)$ se tiene:

$$\Delta w_{ji}(n) = \eta \sum_{t=0}^{n} \alpha^{n-t} \delta_j(t) y_i(t) \tag{2.54}$$

Que representa una serie de tiempo de duración n+1. De las Ecs. (2.25) y (2.26) se tiene que el producto $\delta_j(n) y_i(n)$ es igual a - $\partial E(n)/\partial w_{ji}(n)$. Por tanto, se puede reescribir la Ec. (2.54) de la siguiente manera:

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial E(t)}{\partial w_{ji}(t)} \tag{2.55}$$

De acuerdo con esta relación, se pueden hacer las siguientes observaciones:

1. El ajuste actual de $\Delta w_{ji}(n)$ representa la suma de una serie de tiempo ponderado exponencialmente. Para que las serie de tiempo sean convergentes, la constante de momento debe estar restringida al rango $0 \leq |\alpha| \leq 1$. Cuando α es igual a 0, el algoritmo BP opera sin momento. Tambien la cte. α puede ser positiva o negativa, aunque es improbable que en la práctica se use un valor negativo.
2. Cuando la derivada parcial $\partial E(t)/\partial w_{ji}(t)$ tiene el mismo signo algebraico en iteraciones consecutivas, la suma $\Delta w_{ji}(n)$ crece en magnitud, por lo que el peso $w_{ji}(n)$ se ajusta con un gran cambio.
3. Cuando la derivada parcial $\partial E(t)/\partial w_{ji}(t)$ tiene signos algebraicos contrarios en iteraciones consecutivas, la suma $\Delta w_{ji}(n)$ decrece en magnitud, por lo que el peso $w_{ji}(n)$ se ajusta con un pequeño cambio.

La incorporación del momento al algoritmo BP representa una modificación mínima de la actualización de los pesos, sin embargo tiene efectos beneficiosos en el aprendizaje del algoritmo. Además este momento previene que el proceso de aprendizaje acabe en algún mínimo local de la superficie de error.

Criterio de parada

En general, en el algoritmo BP no hay criterios bien definidos para detener su operación. Más bien, hay algunos criterios, cada uno con su propio merito práctico, que pueden ser utilizados para terminar el ajuste de los pesos. Para formular un criterio de este tipo, es lógico pensar en términos de las propiedades de un mínimo local o global de la superficie de error (Figura 19).

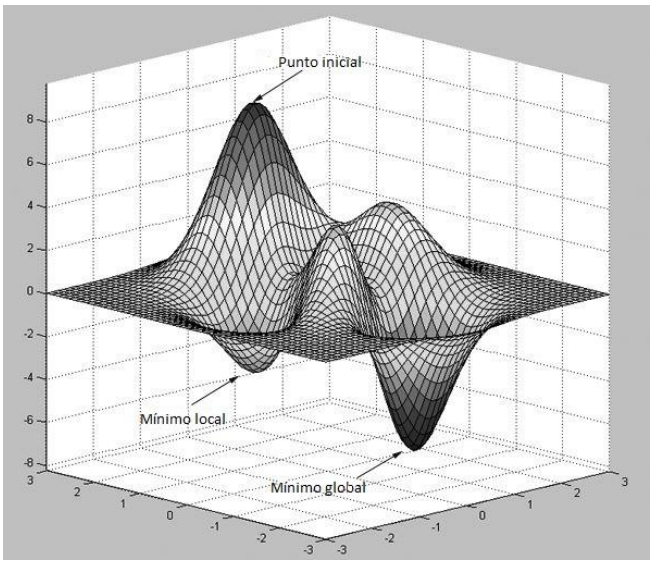


Figura 19. Ejemplo de superficie de error [20]

Se establece que el vector de pesos w* denote un mínimo, que puede ser local o global. Una condición necesaria para que w* sea un mínimo es que el vector gradiente g(w) de la superficie de error con respecto al vector de pesos w sea cero en w = w*. Por lo tanto, Kramer y Sangiovanni-Vincentelli definen un criterio de convergencia razonable para el aprendizaje BP de la siguiente manera:

Se considera que el algoritmo BP ha convergido cuando la norma Euclidiana del vector gradiente alcanza un límite suficientemente pequeño.

El inconveniente de este criterio de convergencia es que, para ensayos con éxito, el tiempo de aprendizaje puede ser grande. Además, requiere el cálculo del vector gradiente g(w).

Otra propiedad única de un mínimo que se puede usar es el hecho de que la función de coste o el error de medida $E_{av}(w)$ es estacionario en el punto w = w*. Por lo tanto, se puede sugerir un criterio de convergencia distinto:

Se considera que el algoritmo BP ha convergido cuando la tasa absoluta de cambio en el error cuadrático medio para cada iteración es suficientemente pequeña.

Se considera que la tasa de cambio en el error cuadrático medio es suficientemente pequeña si se encuentra en un rango de 0,1 a 1% por iteración.

Capítulo 3

Metodología para el diseño del estimador

La metodología propuesta para el diseño del estimador se divide en dos etapas: en la primera se genera un modelo de simulación del vehículo mediante TruckSim y en la segunda se diseña y se valida la red neuronal que estimará el ángulo de balanceo.

En la Figura 20 se presenta el flujo de trabajo seguido para el desarrollo del estimador.

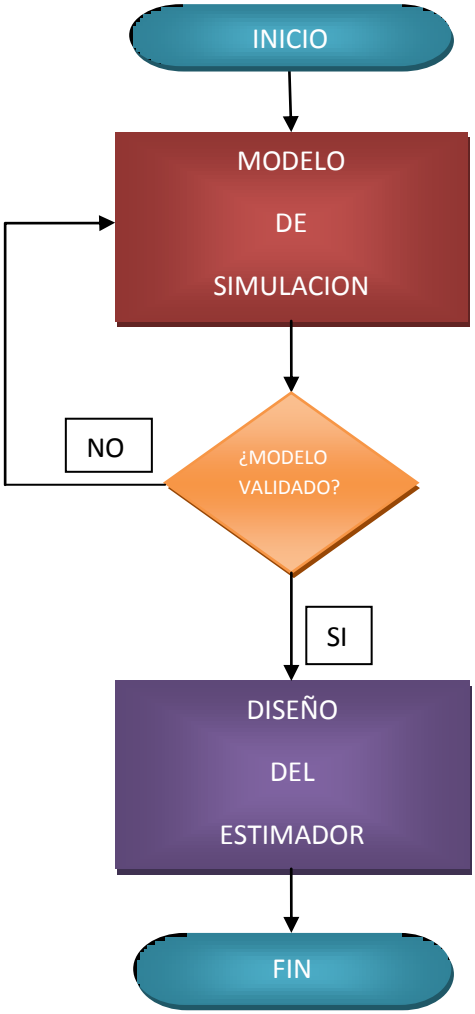


Figura 20. Flujo de trabajo para el diseño del estimador

En la etapa correspondiente al modelo de simulación se desarrollan los siguientes apartados:

- Instrumentación del vehículo real: se describen los sensores utilizados para la obtención de las variables dinámicas, así como su localización en el vehículo real.
- Realización de ensayos dinámicos: una vez instrumentado el vehículo se realizan maniobras de doble y simple cambio de carril para registrar los valores de las variables durante las mismas y de esta manera disponer de datos experimentales que se usarán para la validación del modelo.
- Diseño y validación del modelo de simulación: mediante el software TruckSim se diseña un modelo de furgoneta que se asemeje al vehículo real. Esto se realiza variando los diferentes componentes del modelo de simulación (componentes de la suspensión, de la dirección…) hasta que se consigue el menor error entre los datos experimentales y los del modelo, mediante un proceso de prueba y error.

Una vez validado el modelo de simulación se desarrolla la segunda etapa. Esta se compone de los siguientes apartados:

- Simulación de maniobras: previo al diseño del estimador se realizan simulaciones de una serie de maniobras con el objetivo de disponer de una amplia variedad de datos para el entrenamiento de la red neuronal.
- Programación y análisis de sensibilidad de la red neuronal: a través de Matlab se programa la red neuronal. Se realiza un análisis de sensibilidad para determinar con cuántas entradas a la red y con cuántas neuronas en la capa oculta se consigue la mejor estimación de la variable objetivo.
- Validación de la red neuronal: para validar la red propuesta en el apartado anterior se realiza una simulación de la misma con maniobras experimentales.

Capítulo 4

Modelo de Simulación

4.1. Instrumentación del vehículo real

El vehículo utilizado es una furgoneta Mercedes Sprinter. Este vehículo ha sido equipado con un sensor de ángulo de dirección MSW 250 Nm de la marca Kistler, una antena registradora de datos Vbox 3i, un sensor IMU y dos antenas GPS de la marca Racelogic. El IMU se instala en el suelo de la furgoneta cerca de su centro de gravedad, mientras que las dos antenas se montan en el techo formando 90 grados con el eje longitudinal del vehículo con el fin de medir el ángulo de balanceo con la mayor precisión posible.

4.1.1. Sensor del ángulo de dirección

El sensor utilizado para medir el ángulo de dirección es un volante marca Kistler, modelo MSW 250. Se caracteriza por su tamaño y peso reducido y por ofrecer una gran precisión en la medición sin perjudicar a otros elementos de control. Conviene señalar que este sensor es capaz de medir el par de dirección, la velocidad de la dirección y el ángulo de dirección.

Este modelo incorpora un volante y unos adaptadores ajustables, de manera que este conjunto pueda montarse al volante del vehículo a utilizar, tal y como se observa en la Figura 21. En la Figura 22 se muestra el conjunto anterior instalado al volante de la furgoneta con el cableado correspondiente para registrar la medición del ángulo de dirección.

Figura 21. Sensor del ángulo de balanceo

Figura 22. Sensor del ángulo de dirección instalado en la furgoneta

En la Tabla 6 se incluyen los datos técnicos del sensor utilizado:

Tabla 6. Datos técnicos del sensor del ángulo de dirección

	Unidades	Valor
Fuente de alimentación	V	10…28
Consumo de energía a 12 V	W	<20
Tiempo de filtrado	ms	2…512 (o sin filtrar)
Frecuencia de medición	Hz	1000
Rango de temperaturas		
- Rango de temperaturas nominal	ºC	0…70
- Rango de temperaturas de operación	ºC	-20…80
Grado de protección		IP40
Momento de inercia	Kg.cm^2	80
Peso	Kg	2,8
Par de dirección		
Rango de medición	N.m	±250
Sobrecarga		
-Esfuerzo de torsión	N.m	±500
-Momento de flexión	N.m	±750
Precisión	%FSO	±0,15
Desviación lineal	%FSO	±0,15
Angulo de dirección		
Rango de medición	∘	≥ ±1250
Velocidad de dirección	$\circ/s$	≤ 2000
Resolución	∘	0,015
Precisión	∘	±0,1
Adaptación del volante		
Diámetro del eje	mm	75
Altura máxima del cuerpo de medición	mm	33
Paso de la rosca del tornillo (M4/16)	mm	90

4.1.2. Sensor de velocidades y aceleraciones

Para medir la aceleración lateral y longitudinal, así como la variación en el ángulo de guiñada y de balanceo se utiliza un sensor RLVB IMU 03 de la marca Racelogic (Figura 23). Este sensor proporciona mediciones muy precisas de dichas variables usando tres sensores que miden la variación de los ángulos de guiñada, balanceo y cabeceo, y tres acelerómetros. Está construido con una carcasa a prueba de salpicaduras por lo que es ideal para su uso en entornos hostiles, así como en pruebas de automoción.

Figura 23. Sensor RLVB IMU 03

En la Tabla 7 se incluyen los datos técnicos del sensor:

Tabla 7. Datos técnicos del sensor RLVB IMU 03

	Unidades	Valor
Resolución	$\circ/s$	0,01
Ancho de banda	Hz	40
Aceleración		
Rango	G	±1,7
Resolución	mg	1
Ancho de banda	Hz	50
Sensor de temperatura		
Rango de calibración de temperatura	ºC	-10 a 50
Resolución	ºC	0,1
Corriente	mA	150
Voltaje	V DC	8 a 30
Temperatura de operación	ºC	-30 a 70

Este sensor está diseñado para su uso con un registrador de datos GPS (Figura 24). Cuando se utiliza justo con este registrador (VBOX 3i) los datos del sensor pueden ser integrados sin problemas lo que asegura datos de primera calidad, incluso cuando se interrumpe la recepción del satélite.

Figura 24. Registrador de datos VBOX 3i

El sensor RLVB IMU se sitúa en el suelo de la furgoneta mientras que el registrador de datos con todas las conexiones necesarias para la medición de las anteriores variables se sitúa en el salpicadero de la furgoneta, tal y como se observa en la Figura 25.

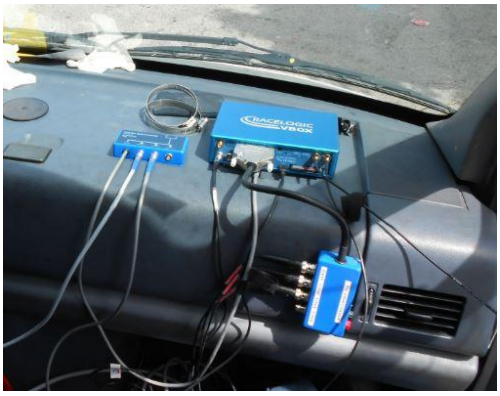


Figura 25. Registrador de datos VBOX 3i

En la Tabla 8 se incluyen los datos técnicos del registrador de datos VBOX 3i:

Tabla 8. Datos técnicos del registrador de datos VBOX 3i

	Unidades	Valor
Velocidad		
Precisión	Km/h	0,1
Frecuencia de actualización	Hz	100
Velocidad máxima	Km/h	1600
Velocidad mínima	Km/h	0,1
Resolución	Km/h	0,01
Aceleración		
Precisión		0,50%
Frecuencia de actualización	Hz	100
Aceleración máxima	G	20
Resolución	G	0,01

4.1.3. Sensor del ángulo de balanceo

Para medir el ángulo de balanceo en el vehículo experimental se utilizan dos antenas de la marca Racelogic conectadas al registrador de datos VBOX 3i mostrado en el apartado anterior. Se utiliza el modelo de antena RLVB3iSL. Además del ángulo de balanceo estas antenas tienen la capacidad de medir con un alto nivel de precisión los ángulos de deslizamiento e inclinación, lo que las convierte en unos sensores ideales para pruebas dinámicas de vehículos.

En la Tabla 9 se presentan las precisiones que se consiguen en las mediciones en función de la separación entre las antenas:

Tabla 9. Precisión en la medida del ángulo de balanceo

Separación entre las antenas (m)	Angulo de deslizamiento (°)	Angulo de balanceo/inclinación (°)
0,5	<0,2	<0,14
1	<0,1	<0,07
1,5	<0,067	<0,047
2	<0,05	<0,035
2,5	<0,04	<0,028

El lugar recomendado para la localización de estas antenas en vehículos industriales es el lugar más alto posible y que se encuentre libre de obstrucciones metálicas. Además para el montaje de estas antenas se dispone de un kit para su correcto anclaje a la superficie elegida. El lugar elegido para la instalación de dicho dispositivo en la furgoneta es el techo de la misma con una separación entre ambas antenas de 1,5 metros con lo que se obtiene una precisión inferior a 0,047° en la medición del ángulo de balanceo.

Figura 26. Localización de las antenas para medir el ángulo de balanceo

Una vez instalados estos sensores en la furgoneta se realizan una serie de maniobras de doble cambio de carril y de simple cambio de carril a distintas velocidades y se guardan los datos obtenidos para las distintas variables para poder usarlos en el apartado siguiente para la validación del modelo de TruckSim.

4.2. Diseño y validación del modelo de simulación

4.2.1. Diseño del modelo de simulación

En este apartado se desarrolla el modelo de simulación utilizando el software TruckSim, el cual es un software usado comúnmente para el modelado de la dinámica de vehículos industriales tales como camiones con remolque, autobuses y furgonetas. Tal y como se indica en la metodología, el diseño se realiza modificando los diferentes parámetros del vehículo hasta conseguir el menor error entre ambos modelos para las variables obtenidas.

Este software cuenta con una librería de vehículos industriales en la que el modelo Large European Van (Figura 28) es el que mejor se adapta a la geometría del vehículo real (Figura 27). A partir de este modelo se modifican sus componentes hasta obtener un comportamiento dinámico lo más cercano al vehículo real.

Figura 27. Mercedes Sprinter

Figura 28. Modelo de simulación

Para realizar el proceso de ajuste del modelo de TruckSim se modifican los sistemas que influyen de manera determinante en la dinámica del vehículo, dejando los valores que establece por defecto el vehículo seleccionado para los demás componentes.

Los sistemas que se modifican en el modelo son los siguientes:

- Geometrías y pesos del vehículo
- Sistema de dirección
- Sistema de suspensión

A continuación se presentan los valores de los sistemas que se modifican en el modelo de simulación. El procedimiento para realizar este ajuste, así como la descripción de los sistemas que se pueden modelar en TruckSim se incluyen en el Anexo A.

- Geometrías y pesos del vehículo

La tabla 10 muestra la localización del centro de gravedad, de la masa suspendida y los detalles de inercia del modelo. Estos valores son extraídos de la ficha técnica del vehículo real.

Tabla 10. Masa suspendida, centro de gravedad y datos de inercia del modelo de simulación

Parámetro del vehículo	Valor	Unidades
Masa Suspendida	1700	kg
Eje de Balanceo (I_{xx})	500	kg.m^2
Eje de Cabeceo (I_{yy})	2975	kg.m^2
Eje de Guiñada (I_{zz})	2975	kg.m^2
Producto (I_{xy})	0	kg.m^2
Producto (I_{xz})	0	kg.m^2
Producto (I_{yz})	0	kg.m^2
Distancia horizontal del cg desde el eje delantero	1509,6	mm
Distancia vertical del cg desde el suelo	980	mm
Distancia lateral del cg	-6,33	mm
Altura de la furgoneta	1874	mm
Ancho de la furgoneta	1983	mm

- Sistema de dirección

En la Tabla 11 se muestran los valores de los parámetros del sistema de dirección.

Tabla 11. Parámetros del sistema de dirección

	Valor		Unidad
Nominal steering gear ratio	19,8		deg/deg
Tie rod compliance	0,001		deg/N.m
Steering column compliance	0,002		deg/N.m
Axle wrap compliance	0,001		deg/N.m
Ratio: Steer / Wrap	0,09		deg/deg
Wheel Steer vs. axle jounce	0,0001		deg/mm
	Left	Right	
Lateral offset	0	0	mm
Lateral inclination	7,2	7,2	deg
X coordinate	0	0	mm
Caster angle	5,2	5,2	deg

- Sistema de suspensión

En las siguientes tablas se muestran todos los parámetros que se modifican en el modelo de TruckSim relativos a la suspensión.

Tabla 12. Parámetros de la suspensión

	Eje conducido	Eje tractor	Unidad
Ancho de vía	1638	1630	mm
Altura del cg desde el suelo	348	348	mm
Altura del centro de la rueda desde el suelo	348	348	mm
Coordenada lateral del centro del eje	0	0	mm
Inercia de giro – lado izquierdo	5	5	kg.m^2
Inercia de giro – lado derecho	5	5	kg.m^2
Masa no suspendida	200	250	kg
Eje de balanceo e inercia de guiñada	58,5	142	kg.m^2
Coeficiente lineal de dirección de balanceo	0,6	0,6	deg/deg
Convergencia rueda izquierda	-0,11	0,07	deg
Convergencia rueda derecha	0,45	0,39	deg
Inclinación rueda izquierda	1,18	-1,12	deg
Inclinación rueda derecha	0,56	0,51	deg

En la Tabla 13 se muestran los valores que se establecen para los componentes de la suspensión, mientras que en la Tabla 14 se indican los valores de los coeficientes para dichos componentes. Estos coeficientes influyen de manera muy importante en la dinámica del vehículo, y como se observa en la tabla, se realiza un ajuste hasta la milésima en alguno de ellos. En el Anexo A se incluye una explicación de cada uno de estos parámetros

Tabla 13. Valores de los componentes de la suspensión para ambos ejes

Parámetro		Eje conducido	Eje tractor	Unidad
Rigidez de las ballestas izquierda y derecha		80	200	N/mm
Holgura entre anti-rebotes (¨Jounce Stops Clearance¨)		100	60	mm
Holgura entre anti-rebotes (¨Rebound Stops Clearance¨)		60	60	mm
Separación entre Izquierdo y derecho	Muelles	900	900	mm
	Amortiguadores	880	850	mm
	Anti-Rebotes (¨Jounce Stops¨)	890	100	mm
	Anti-Rebotes (¨Rebound Stops¨)	890	100	mm
Momento auxiliar de balanceo		100	100	N-m/deg
Amortiguados auxiliar de balanceo		150	150	N-m.sec/deg

Tabla 14. Coeficientes para ambos ejes

Parámetro	Eje conducido		Eje tractor		Unidad
	Izquierdo	Derecho	Izquierdo	Derecho	
Toe Vs Fx	0	0	0	0	deg/N
Steer vs Fy	0,00001	0,00001	0,0001	0,0001	deg/N
Steer vs Mz	0,001	0,001	0	0	deg/(N-m)
Camber vs Fx	0	0	0	0	deg/N
Inclination vs Fy	0,001	0	0	0	deg/N
Inclination vs Mz	0	0	0	0	deg/(N-m)
Axle longitudinal vs Fx	0		0		mm/N
Axle lateral vs Fy	0,001		0		mm/N

Por último, en la Tabla 15 se presentan los valores que se adoptan para las relaciones mecánicas para los componentes de las suspensiones.

Tabla 15. Relaciones mecánicas para los componentes de las suspensiones

Parámetro	Eje conducido		Eje tractor	
	Izquierdo	Derecho	Izquierdo	Derecho
Muelles	0,75	0,75	1,2	1,2
Amortiguadores	1	1	1	1
Anti-rebotes (¨Jounce stops¨)	1	1	1	1
Anti-rebotes (¨Rebound stops¨)	1	1	1	1

4.2.2. Validación del modelo de simulación

Este apartado está dedicado a la validación del modelo de simulación anterior. La validación se realiza utilizando maniobras como son el doble y simple cambio de carril. Los resultados de la simulación de estas maniobras se comparan con los datos obtenidos experimentalmente con los sensores descritos en el apartado 4.1.

Las variables dinámicas que se utilizan para validar el modelo de furgoneta son la aceleración lateral (a_y), aceleración longitudinal (a_x), ángulo de balanceo (θ), variación del ángulo de balanceo ($\dot{\theta}$) y variación del ángulo de guiñada ($\dot{\psi}$).

Tras realizar el proceso de ajuste de los valores de los distintos componentes mecánicos de la furgoneta se comparan los datos obtenidos para las anteriores variables tanto en las maniobras experimentales como en las maniobras simuladas en TruckSim.

Para este proceso de validación se utilizan dos maniobras de doble cambio de carril a 50 y 60 Km/h y dos maniobras de simple cambio de carril a 50 y 60 Km/h.

La Figura 29 muestra un esquema del proceso de obtención de los datos:

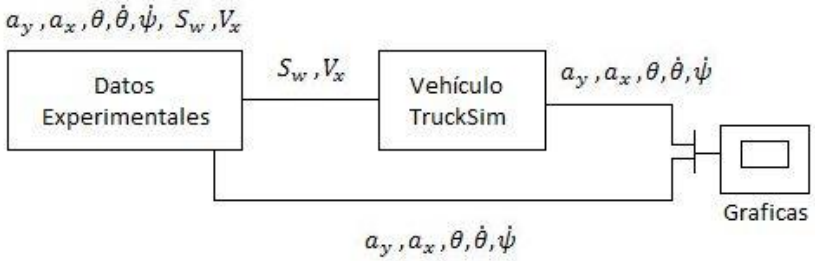


Figura 29. Esquema del proceso de obtención de las gráficas

4.2.3. Doble cambio de carril a 50 Km/h (DCL50)

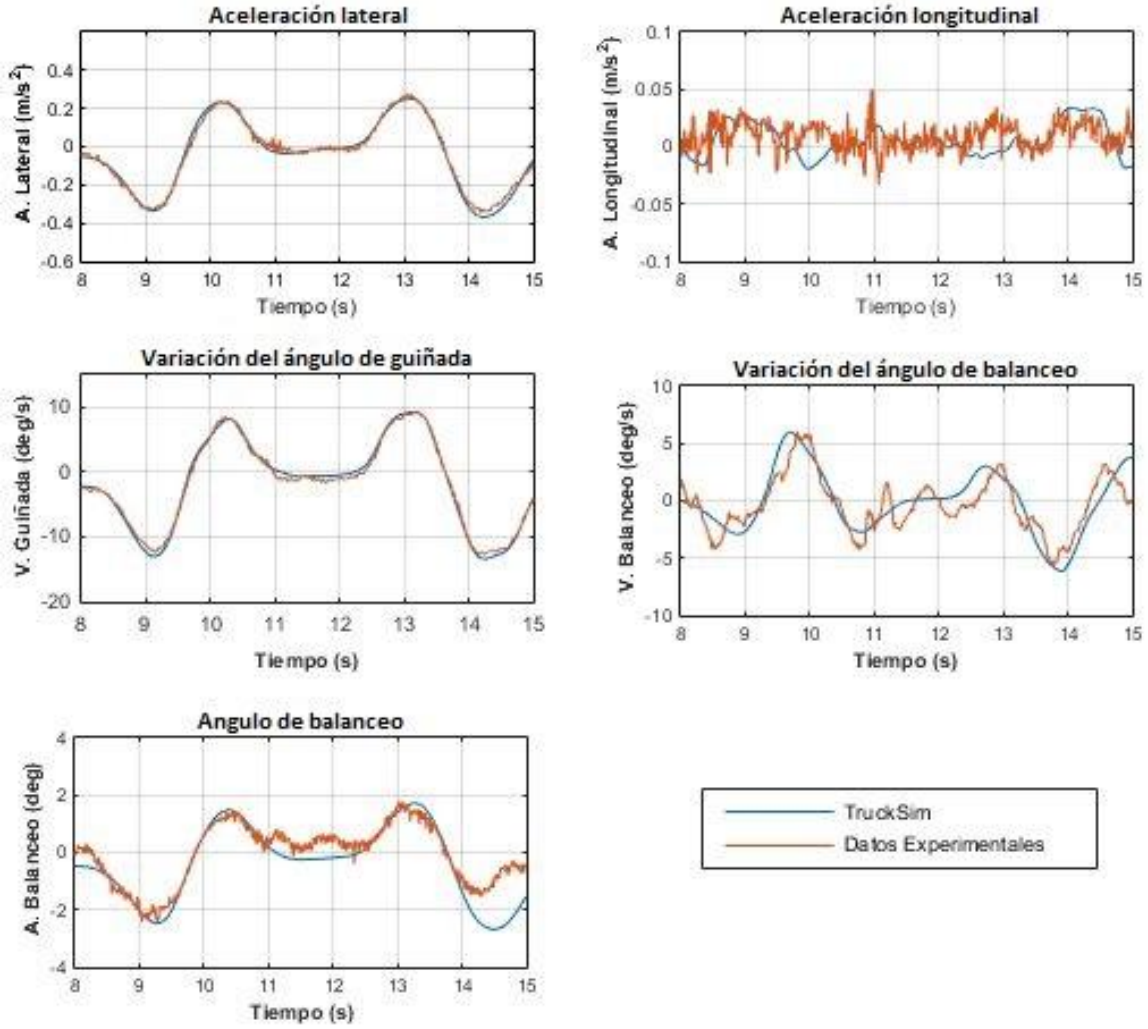


Figura 30. Validación del modelo de TruckSim - Doble cambio de carril a 50 Km/h

4.2.4. Doble cambio de carril a 60 Km/h (DCL60)

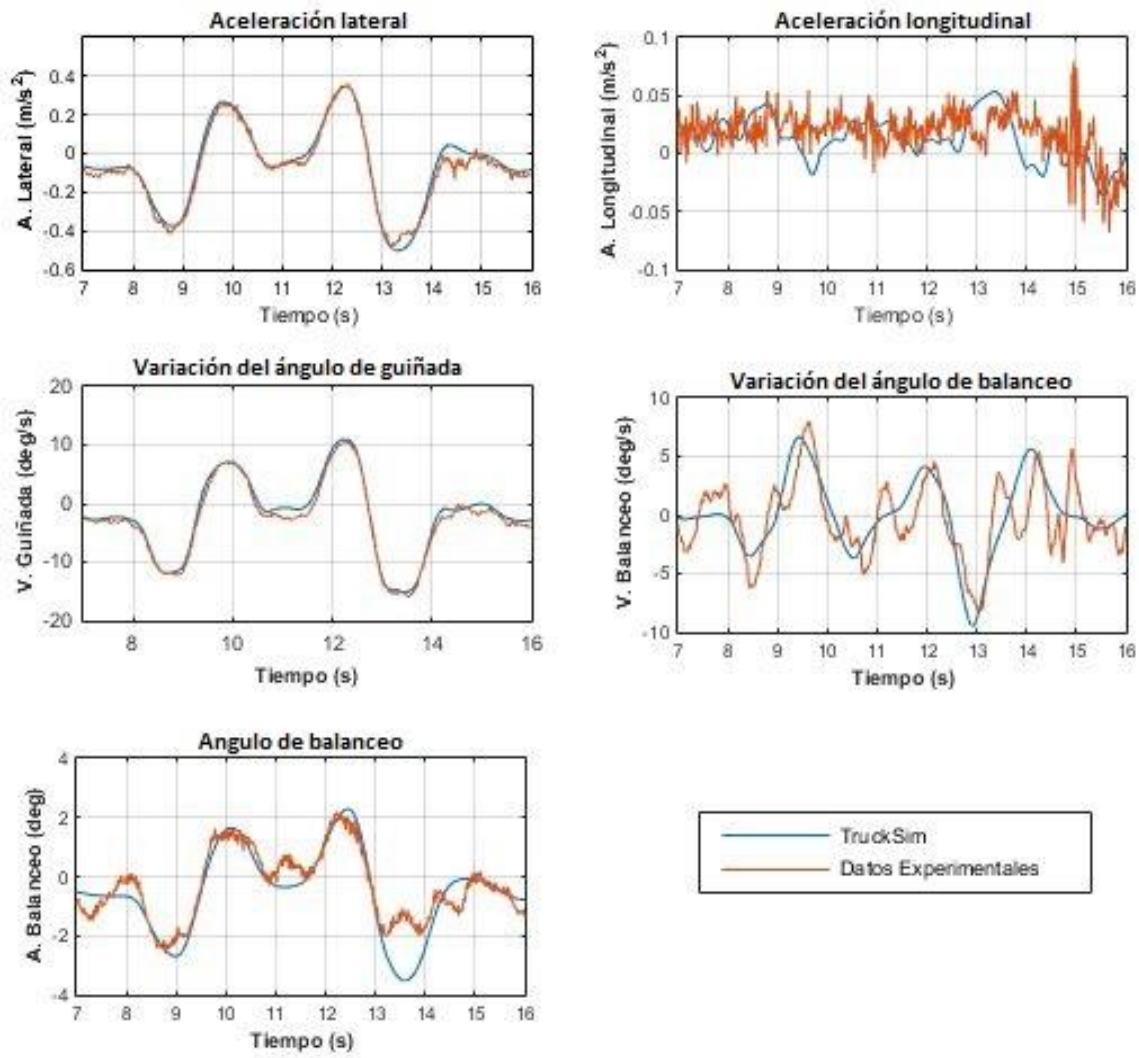


Figura 31. Validación del modelo de TruckSim - Doble cambio de carril a 60 Km/h

En las dos maniobras de doble cambio de carril se observa como el modelo de TruckSim se ajusta bien para la aceleración lateral, la variación del ángulo de guiñada y para el ángulo de balanceo. Las irregularidades presentes en la calzada causan que el modelo tenga dificultades para ajustarse en la variación del ángulo de balanceo.

4.2.5. Simple cambio de carril a 50 Km/h (CL50)

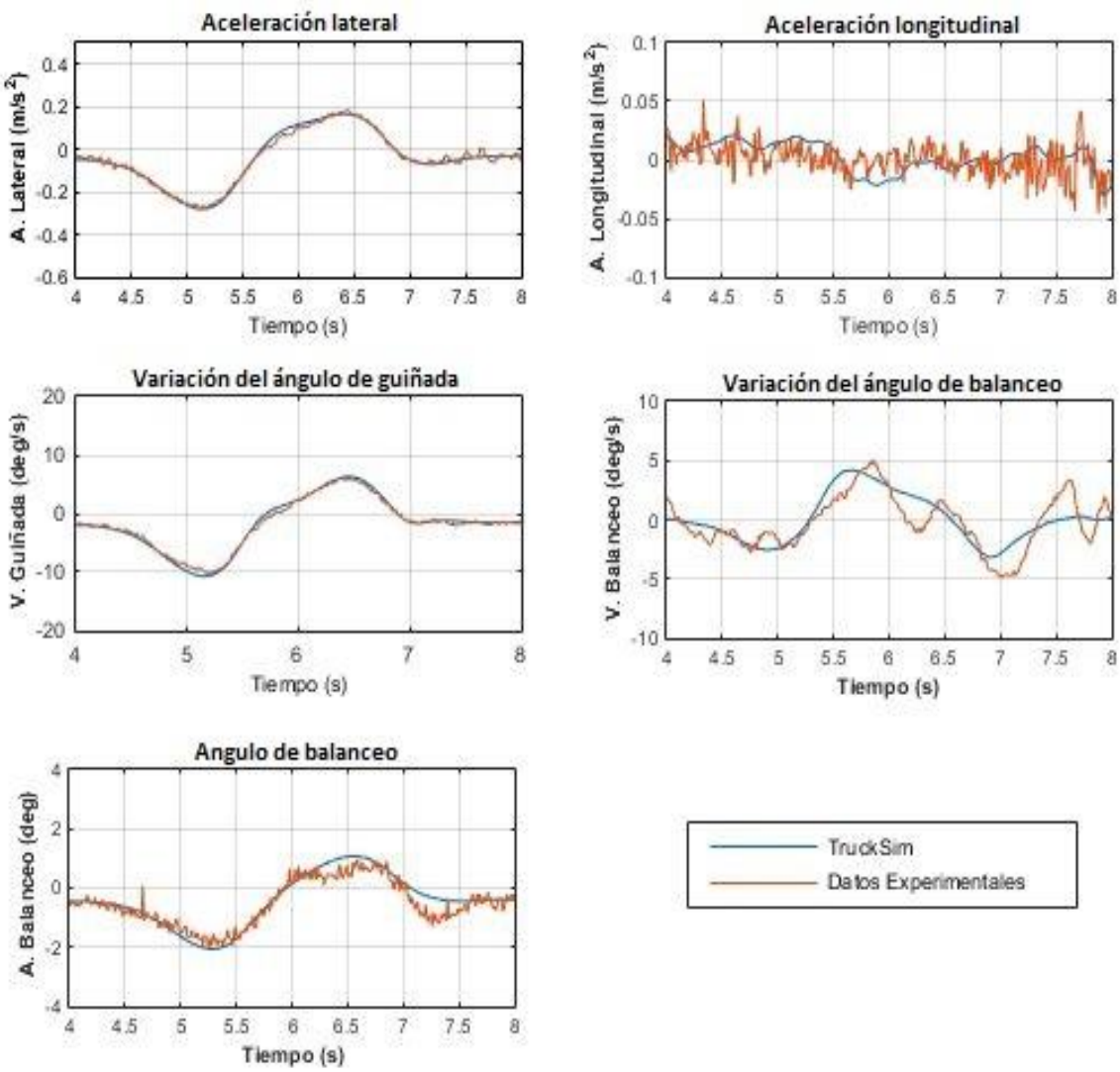


Figura 32. Validación del modelo de TruckSim - Cambio de carril a 50 Km/h

4.2.6. Simple cambio de carril a 60 Km/h (CL60)

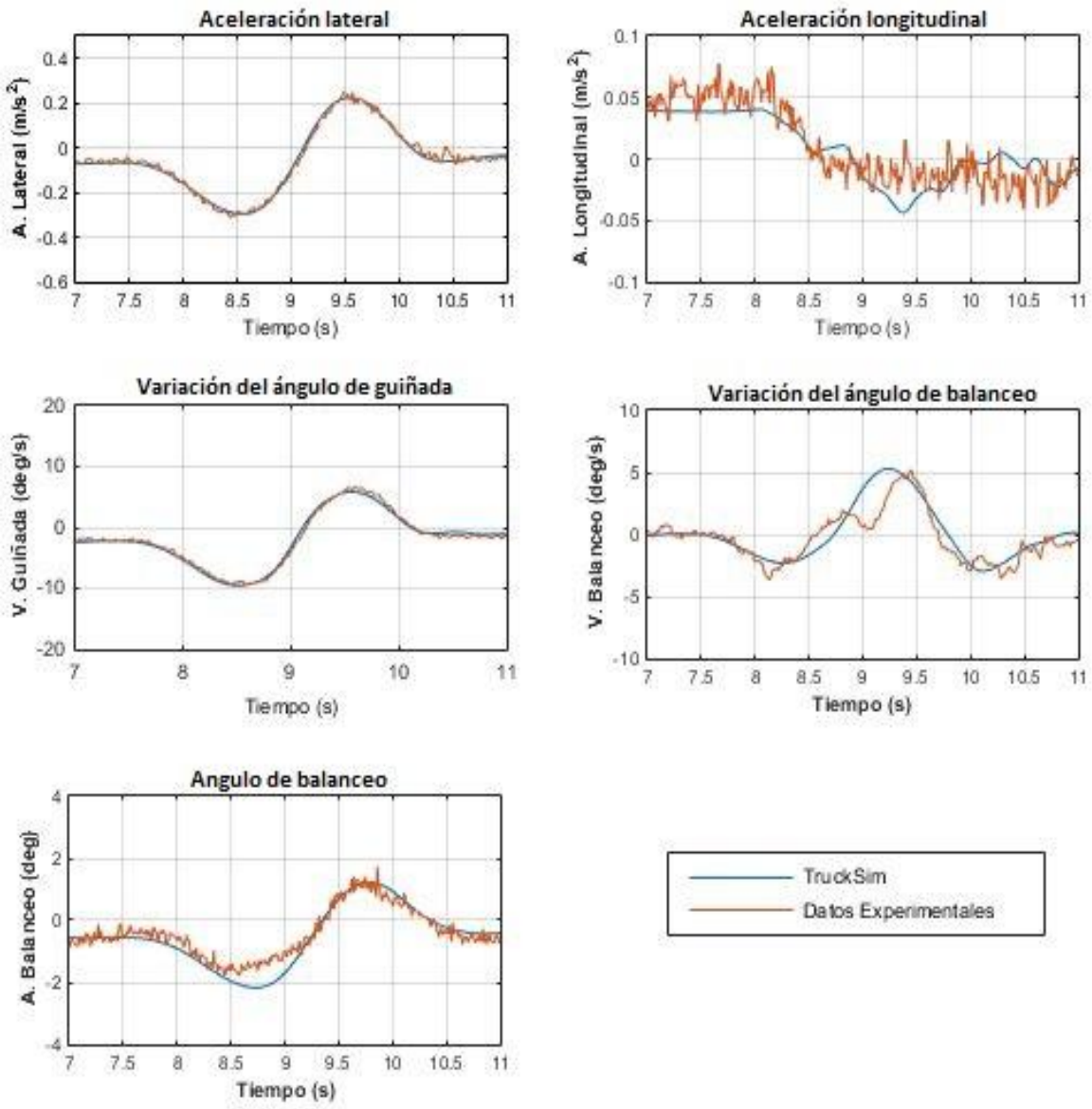


Figura 33. Validación del modelo de TruckSim - Cambio de carril a 60 Km/h

Las maniobras de simple cambio de carril muestran la misma tendencia que las de doble cambio en lo que se refiere al ajuste a las gráficas reales.

4.2.7. Cálculo de errores

Para calcular el error existente entre los valores experimentales y los simulados en TruckSim se utiliza el coeficiente de determinación (R^2), el cual es una medida estadística de la bondad del ajuste o fiabilidad de los valores simulados a los experimentales. Este coeficiente es una medida adimensional, de fácil cálculo e interpretación, dado que sus posibles valores se encuentran acotados entre 0 y 1.

Tabla 16. Errores del modelo de TruckSim

	Error (R^2)			
Variable/Maniobra	DCL50	DCL60	CL50	CL60
a_y	0,9784	0,9486	0,9674	0,9628
a_x	0,5415	0,6492	0,5432	0,6689
$\dot{\psi}$	0,9903	0,9627	0,9805	0,9693
$\dot{\theta}$	0,5239	0,3955	0,4467	0,3528
θ	0,5663	0,6083	0,6642	0,5301

Para las maniobras realizadas se puede comprobar que para la aceleración lateral y la variación del ángulo de guiñada se obtienen coeficientes de determinación de valores cercanos a 1, lo que indica que existe una relación muy estrecha entre los datos experimentales y los simulados.

Los valores medios obtenidos para la aceleración longitudinal, la variación del ángulo de balanceo y el ángulo de balanceo se deben a los datos obtenidos experimentalmente para esta variable. Tal y como se observa en las figuras de los apartados anteriores las curvas experimentales presentan irregularidades a las que el modelo de TruckSim no puede ajustarse. La Figura 34 muestra un ejemplo de estas irregularidades, las cuales pueden ser debidas a un terreno irregular (presencia de baches, alcantarillas…).

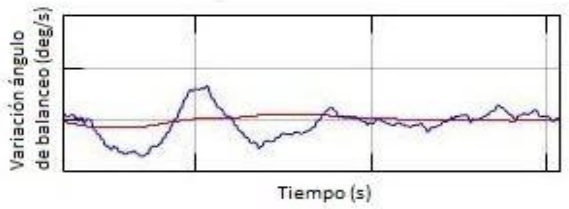


Figura 34. Irregularidades de las gráficas experimentales

Estas irregularidades son las causantes de que el coeficiente de determinación para estas tres variables no se acerque a la unidad. Sin embargo, en las Figuras 30, 31, 32 y 33 se observa que en el momento en el que se realiza la maniobra de doble cambio de carril o de cambio de carril si existe similitud entre ambas curvas.

Por tanto, a la vista de los resultados obtenidos especialmente para las variables de aceleración lateral y variación del ángulo de guiñada se valida el modelo realizado en TruckSim.

Capítulo 5

Diseño del estimador

5.1. Obtención de datos de entrenamiento

Una vez validado el modelo de furgoneta de TruckSim se realizan una serie de maniobras en dicho software con objeto de obtener los valores de las variables dinámicas que se van a utilizar en el entrenamiento de la red neuronal. Se realizan maniobras de doble cambio de carril, simple cambio de carril y slalom a distintas velocidades y diferentes coeficientes de fricción (1, 0.85, 0.5, 0.3), de tal forma que la red disponga de datos de entrenamiento que abarquen una gran cantidad de posibles situaciones en la carretera. Las velocidades que se utilizan para las distintas maniobras varían desde 30 hasta 150 km/h.

Conviene señalar que cuando la velocidad es alta y el coeficiente de fricción es bajo se produce una situación de inestabilidad dinámica en el vehículo simulado y por tanto los datos de esas maniobras no son válidos para formar la matriz de entrenamiento de la red neuronal.

En la Tabla 17 se muestran las maniobras válidas para formar la matriz de entrenamiento. En total se obtienen 122 maniobras válidas.

Tabla 17. Maniobras válidas para la matriz de entrenamiento

<table>
<tr><th rowspan="2">Coeficiente de fricción</th><th rowspan="2">Maniobra</th><th colspan="13">Velocidad (Km/h)</th></tr>
<tr><th>30</th><th>40</th><th>50</th><th>60</th><th>70</th><th>80</th><th>90</th><th>100</th><th>110</th><th>120</th><th>130</th><th>140</th><th>150</th></tr>
<tr><td rowspan="3">1/0,85</td><td>DCL</td><td colspan="13" rowspan="3">Validas</td></tr>
<tr><td>CL</td></tr>
<tr><td>SL</td></tr>
<tr><td rowspan="3">0,5</td><td>DCL</td><td colspan="9">Validas</td><td colspan="4">No validas</td></tr>
<tr><td>CL</td><td colspan="13">Validas</td></tr>
<tr><td>SL</td><td colspan="9">Validas</td><td colspan="4">No validas</td></tr>
<tr><td rowspan="3">0,3</td><td>DCL</td><td colspan="5" rowspan="3">Validas</td><td colspan="8" rowspan="3">No validas</td></tr>
<tr><td>CL</td></tr>
<tr><td>SL</td></tr>
</table>

De cada maniobra se extraen los valores de las siguientes variables: aceleración lateral (a_y), aceleración longitudinal (a_x), ángulo de dirección (δ), variación del ángulo de balanceo ($\dot{\theta}$), ángulo de balanceo (θ), variación del ángulo de guiñada ($\dot{\psi}$) y las deformaciones de las suspensiones izquierda y derecha tanto del eje delantero como del trasero (y_{ij}). Estas variables se usarán para entrenar a la red neuronal excepto el ángulo de balanceo que será el valor deseado.

5.2. Análisis de sensibilidad de la red neuronal

Dado que la red neuronal propuesta utiliza un algoritmo de entrenamiento BP es necesario determinar con cuantas entradas se consiguen los mejores resultados. De igual manera se debe determinar el número de neuronas en la capa oculta con el cual se consigue el mejor entrenamiento.

Para ello se realizan una serie de entrenamientos combinando diferentes entradas a la red y diferentes números de neuronas en la capa oculta. La nomenclatura que se va a utilizar para caracterizar la red neuronal es la siguiente: X-J-K, donde X es el número de entradas a la red, J es el número de neuronas en la capa oculta y K el número de neuronas en la capa de salida.

Puesto que no se disponen de sensores para obtener los datos de las deformaciones de las suspensiones, el análisis de sensibilidad se realiza para las variables de las que se disponen datos experimentales. Del modelo de TruckSim se extraen las deformaciones de las suspensiones con el objetivo de disponer más entradas y analizar si la adición de esas variables mejora el poder de predicción de la red neuronal.

Por tanto, para determinar que variable de las mencionadas anteriormente tiene una contribución más importante para una predicción precisa del ángulo de balanceo se realiza un primer análisis de sensibilidad, en el cuál se utiliza una red neuronal 1-2-1. Los resultados obtenidos se muestran en la tabla 18.

Tabla 18. Análisis de sensibilidad inicial

VARIABLE DE ENTRADA	ANGULO DE BALANCEO(MSE x 10^{-2})
	1-2-1
a_y	11,06
a_x	70,51
δ	29,15
$\dot{\theta}$	71,14
$\dot{\psi}$	26,54

Los resultados muestran que individualmente la variable que más contribuye a una predicción precisa del ángulo de balanceo es la aceleración lateral. Por lo tanto, se combinará esta variable con el resto, con el objetivo de determinar con cuántas entradas se consigue el menor error en la estimación de la variable objetivo.

En cuanto al número de neuronas en la capa oculta, se van a probar 6 arquitecturas diferentes (X-2-1, X-3-1, X-5-1, X-10-1, X-20-1, X-50-1). El objetivo es determinar si un aumento en el número de neuronas de la capa oculta proporciona mejores resultados. En la Tabla 19 se observan los resultados obtenidos del entrenamiento de las redes neuronales considerando un total de 122 maniobras (219.523 datos).

Tabla 19. Análisis de sensibilidad para 122 maniobras

VARIABLES DE ENTRADA					ANGULO DE BALANCEO(MSE x $\mathbf{10^{-2}}$)					
a_y	a_x	δ	$\dot{\theta}$	$\dot{\psi}$	X-2-1	X-3-1	X-5-1	X-10-1	X-20-1	X-50-1
2 ENTRADAS										
●	●				9,54	8,97	9,02	8,89	8,64	9,16
●		●			11,64	11,62	12,21	11,44	11,50	11,68
●			●		3,40	3,25	3,23	3,04	2,90	2,94
●				●	10,34	10,76	10,70	10,24	10,13	9,94
3 ENTRADAS										
●	●	●			11,62	11,50	11,87	11,45	11,64	11,91
●	●		●		3,40	2,67	3,21	2,86	2,19	2,60
●	●			●	9,42	10,73	10,70	10,05	10,00	10,35
●		●	●		1,96	1,9	3,23	3,33	3,34	3,15
●		●		●	12,63	14,53	14,18	10,36	10,46	9,57
●			●	●	3,15	2,96	3,38	3,60	3,82	2,88
4 ENTRADAS										
●	●	●	●		1,96	1,84	3,17	3,28	3,40	3,32
●	●	●		●	8,91	14,49	14,27	10,17	10,54	10,01
●	●		●	●	3,15	3,28	3,41	3,58	3,67	2,57
●		●	●	●	1,40	2,29	2,61	2,13	2,48	2,86
5 ENTRADAS										
●	●	●	●	●	1,40	2,14	2,73	2,97	3,01	-

Una tendencia general que se puede observar en los resultados de la anterior tabla es que usando más entradas para la red se mejora el poder de predicción del ángulo de balanceo. En cuanto al número óptimo de neuronas en la capa oculta, se observa que para redes con 2 entradas los mejores resultados se obtienen con un número alto de neuronas, pero cuando el número de entradas aumenta es más eficaz usar un número pequeño de neuronas (2 ó 3).

En cuanto a la importancia de cada variable para la estimación del ángulo de balanceo queda reflejado que la variación del ángulo de balanceo contribuye notablemente a la mejora de los resultados y que la adición de la aceleración longitudinal a las otras cuatro variables no mejora los resultados. Así, una red que usa como entradas la aceleración lateral, el ángulo de dirección, la variación del ángulo de balanceo y del ángulo de guiñada y 2 neuronas en la capa oculta estima el mejor resultado para el ángulo de balanceo tal y como se muestra en la Figura 35 (MSE = $1{,}4.10^{-2}$). Por otra parte, una de las redes más sencillas, usando el ángulo de dirección como entrada, y 2 neuronas en la capa oculta estima el ángulo de balanceo con un error notablemente mayor, tal y como se refleja en la Figura 36 (MSE = $29{,}15.10^{-2}$).

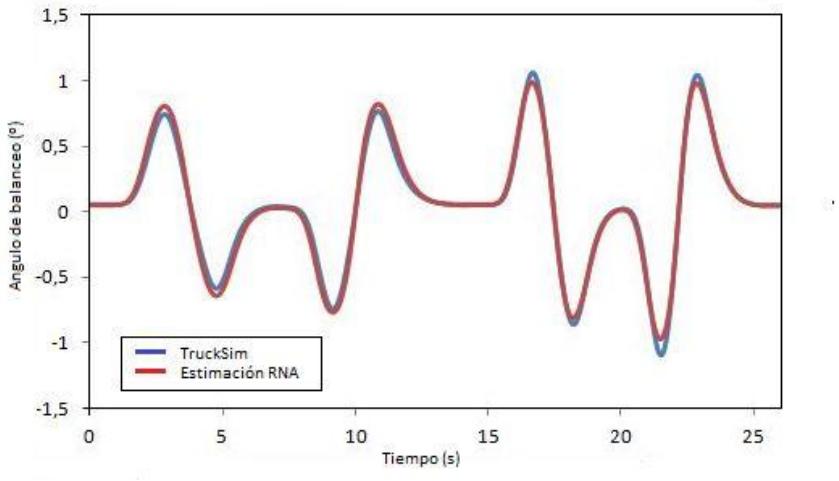


Figura 35. Predicción del ángulo de balanceo usando una red neuronal 4-2-1

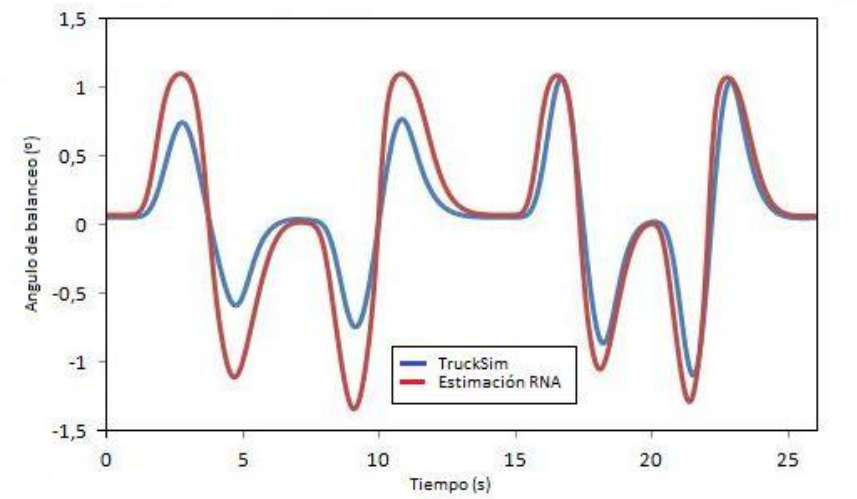


Figura 36. Predicción del ángulo de balanceo usando una red neuronal 1-2-1

Con objeto de comprobar el efecto que tiene la cantidad de datos de entrenamiento sobre los resultados que proporciona la red, se utiliza una red 4-2-1 ($a_y, \delta, \dot{\theta}, \dot{\psi}$) cuyo resultado del entrenamiento con 122 maniobras se observa en la tabla 19. Esta se somete a un nuevo entrenamiento pero esta vez con 39 maniobras (66.441 datos), 13 correspondientes a doble cambio de carril, 13 de simple cambio de carril y 13 de slalom.

Tabla 20. Análisis de sensibilidad para 39 maniobras

VARIABLES DE ENTRADA					ANGULO DE BALANCEO(MSE x $\mathbf{10^{-2}}$)					
a_y	a_x	δ	$\dot{\theta}$	$\dot{\psi}$	4-2-1	4-3-1	4-5-1	4-10-1	4-20-1	4-50-1
4 ENTRADAS										
•		•	•	•	1,42	1,02	1,97	1,62	1,15	2,64

Por último, se realiza el entrenamiento de una red neuronal introduciendo todas las variables de entrada que se obtuvieron del modelo de furgoneta de TruckSim para un total de 39 maniobras. Como se indicó anteriormente, experimentalmente no se dispone de sensores de deformación de las suspensiones y el objetivo de estos entrenamientos adicionales es determinar su posible utilidad.

Tabla 21. Análisis de sensibilidad con todas las variables para 39 maniobras

VARIABLES DE ENTRADA									ANGULO DE BALANCEO(MSE x 10^{-4})					
a_y	a_x	δ	$\dot{\theta}$	$\dot{\psi}$	y_{L1}	y_{R1}	y_{L2}	y_{R2}	9-2-1	9-3-1	9-5-1	9-10-1	9-20-1	9-50-1
9 ENTRADAS														
•	•	•	•	•	•	•	•	•	0,155	0,119	0,113	0,037	0,02	-

Tal y como se observa en la Tabla 21 el error en la predicción del ángulo de balanceo disminuye considerablemente cuando se incorporan como entradas las deformaciones de las suspensiones delanteras y traseras.

Por tanto, a partir de los datos mostrados anteriormente, se pueden hacer las siguientes conclusiones generales:

- En general, los resultados mejoran cuando la red dispone de más variables de entrada. Sin embargo, en la Tabla 19 se observa que la adición de la aceleración longitudinal al resto de variables no mejora los resultados, por tanto esta variable no se incluye en la red que se validará en el apartado siguiente. En cuanto al número de neuronas no puede concluirse que un aumento del número de las mismas en la capa oculta proporcione siempre mejores resultados, por tanto para cada configuración deberá determinarse el número idóneo.
- El segundo análisis de sensibilidad realizado muestra que los resultados incluso mejoran cuando se utiliza una cantidad menor de datos de entrada que la usada inicialmente para cada una de variables. Además de mejorar los resultados esta disminución de los datos de entrada presenta otras dos ventajas: la primera es que el tiempo de entrenamiento de la red disminuye notablemente, y la segunda es que de esta manera se asegura evitar el error de aprendizaje (sobreaprendizaje), explicado en el apartado 2.6.6.1.
- La incorporación de los datos de las deformaciones de las suspensiones como entradas para el entrenamiento de la red neuronal mejora notablemente la estimación del ángulo de balanceo que proporciona la misma. Por tanto, será necesaria la incorporación de sensores de deformación en el vehículo experimental para obtener dichas variables y de esta manera obtener estimaciones más precisas del ángulo de balanceo.

Teniendo en cuenta estas conclusiones, la red neuronal a validar en el siguiente apartado será una red 4-3-1 que tendrá como entradas la aceleración lateral, el ángulo de dirección, la variación del ángulo de balanceo y la variación del ángulo de guiñada, y usará los datos correspondientes a 39 maniobras para cada variable.

La Tabla 22 muestra las principales características de la red neuronal que se utilizará para la estimación del ángulo de balanceo:

Tabla 22. Principales características de la red neuronal

Características	
Nº Capas	2
Nº Neuronas capa oculta	3
Nº Entradas	4
Función de activación capa oculta	Función sigmoidea
Función de activación capa de salida	Función lineal
Función de entrenamiento	Función ¨*trainlm*¨
Función de rendimiento	Función MSE
Nº Iteraciones	100
Función división de datos	Función ¨*dividerand*¨

El código desarrollado y la justificación de la utilización de las anteriores características se adjuntan en el Anexo B.

5.3. Validación de la red neuronal sin influencia de la suspensión

Para realizar la validación de la red neuronal anteriormente propuesta se presentan a la red unas maniobras diferentes a las utilizadas para el entrenamiento de la misma y se estudia la estimación que proporciona la misma del ángulo de balanceo. Se utilizan dos tipos de maniobras: unas que realizó la furgoneta experimentalmente y otras obtenidas en el software TruckSim.

- Validación de la red neuronal mediante maniobras experimentales

Para poder obtener los datos que proporciona la red neuronal al simular dicha maniobra se utiliza el entorno de Simulink. Previo a ello, se carga en archivo .net que genera la red al ser entrenada (Figura 37), cuya obtención se explica en el Anexo B.

En esta figura se observan las cuatro entradas a la red y el bloque que representa la red neuronal. A continuación, se introduce la maniobra experimental a simular y asegurando que las entradas a la red neuronal sean las correctas y en el orden correcto (Figura 38) se obtendrá la gráfica que representa el ángulo de balanceo de la maniobra experimental y el estimado por la red neuronal.

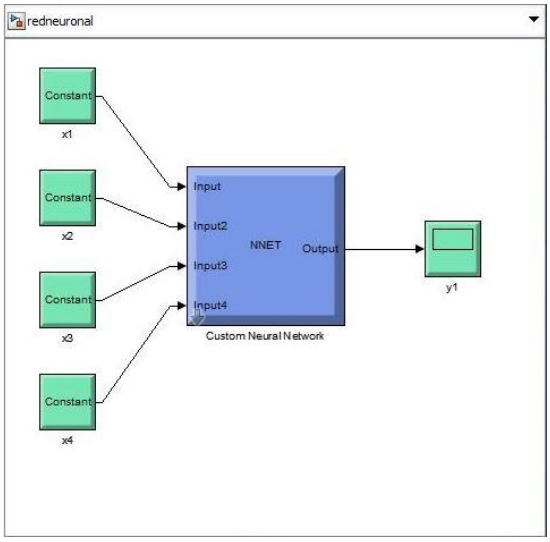


Figura 37. Red neuronal en Simulink

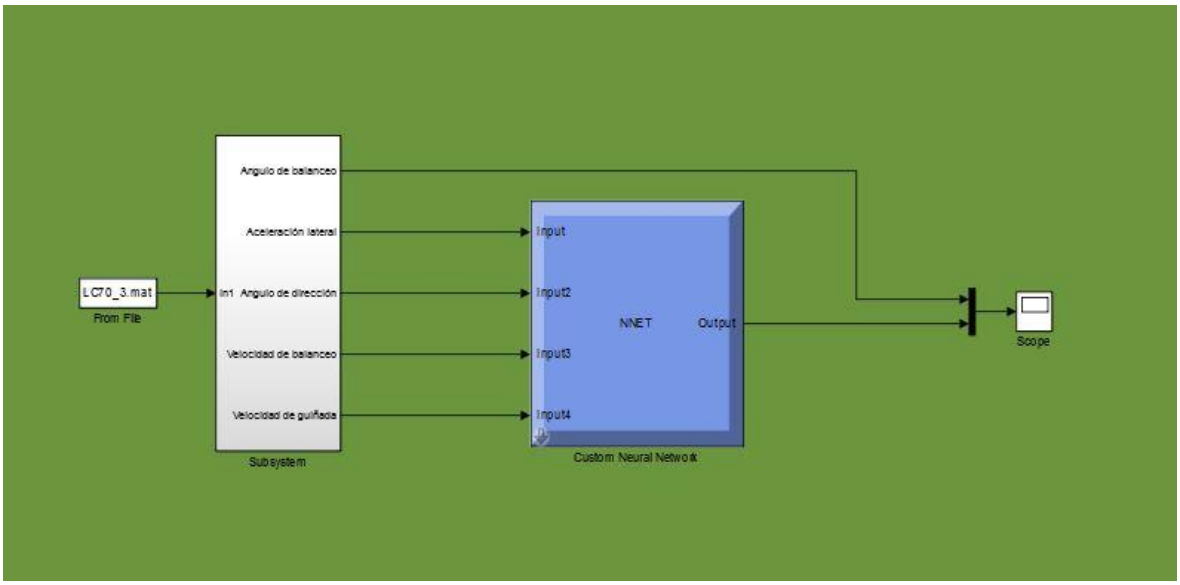


Figura 38. Procedimiento de validación de la red neuronal para maniobras experimentales

Se utilizan cuatro maniobras experimentales para simular la red neuronal propuesta. Estas maniobras son:

- Doble cambio de carril a 40 km/h (Figura 39).
- Doble cambio de carril a 60 km/h (Figura 40).
- Simple cambio de carril a 50 km/h (Figura 41).
- Simple cambio de carril a 70 km/h (Figura 42).

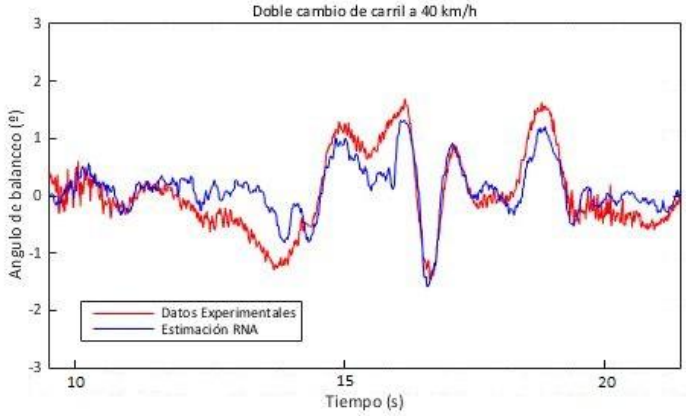


Figura 39. Simulación con maniobra de doble cambio de carril a 40 km/h

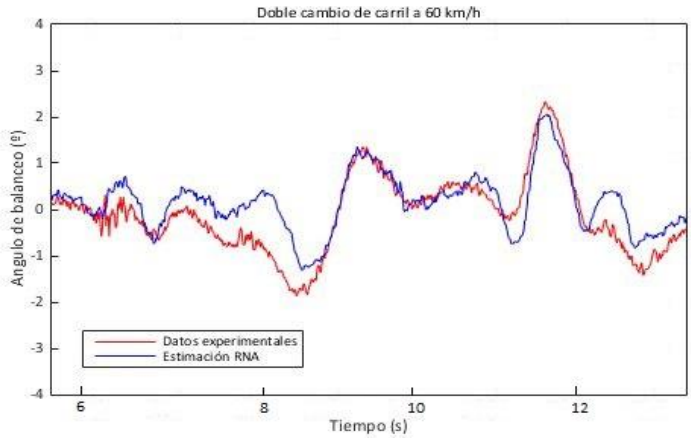


Figura 40. Simulación con maniobra de doble cambio de carril a 60 km/h

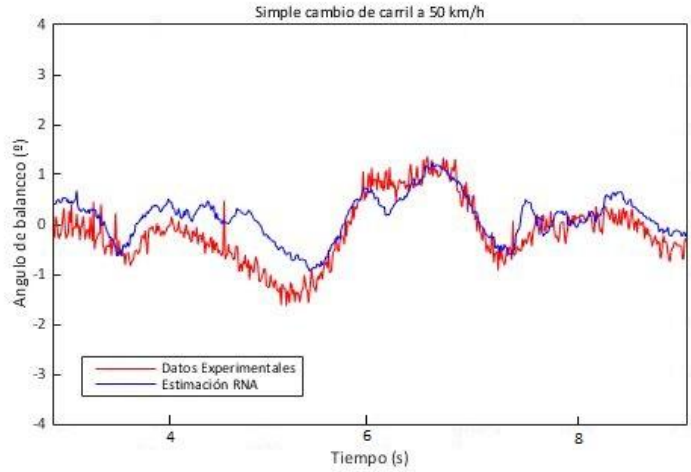


Figura 41. Simulación con maniobra de simple cambio de carril a 50 km/h

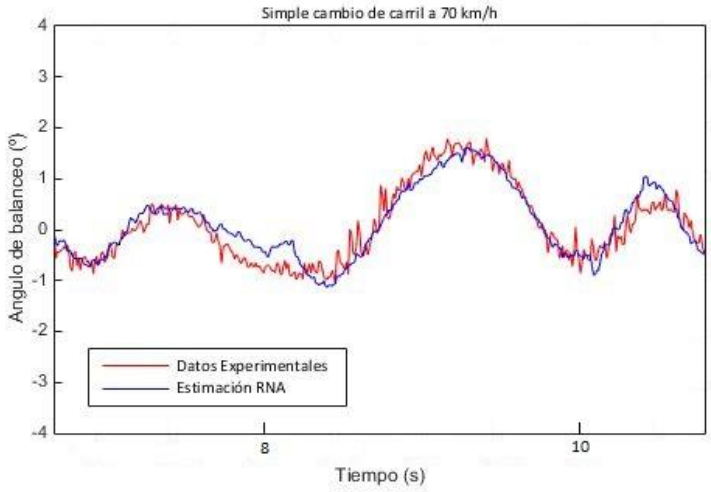


Figura 42. Simulación con maniobra de simple cambio de carril a 70 km/h

Tal y como se observa en las anteriores figuras la red neuronal propuesta es capaz de generalizar y estimar un ángulo de balanceo que se aproxima al objetivo cuando recibe datos de entrada distintos a los que se usaron para entrenar a la misma. Esto quiere decir que se ha evitado el sobreaprendizaje.

- Validación mediante maniobras simuladas en TruckSim

Al igual que en apartado anterior asegurando que las entradas a la red neuronal sean las correctas y en el orden correcto (Figura 43) se obtendrá la gráfica que representa el ángulo de balanceo de la maniobra simulada en TruckSim y el estimado por la red neuronal.

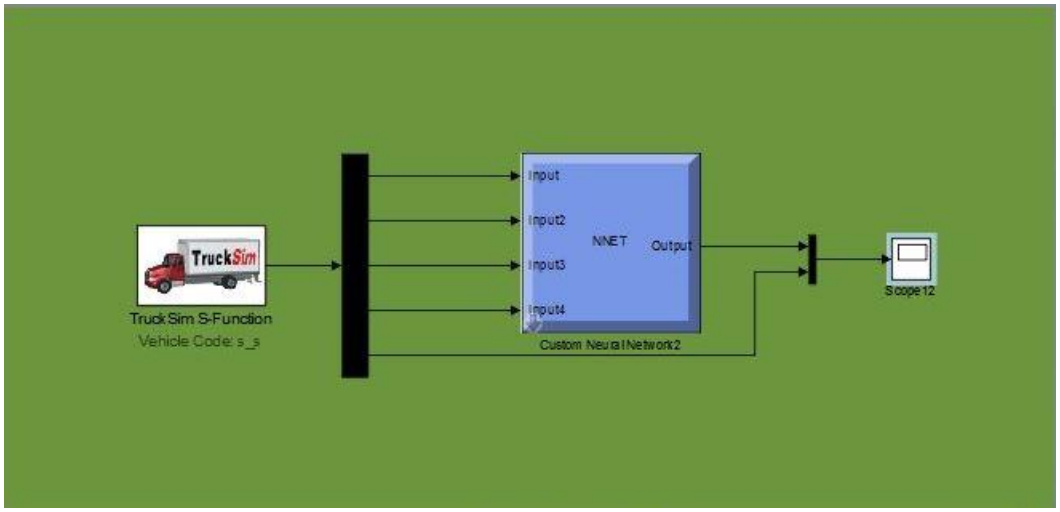


Figura 43. Procedimiento de validación de la red neuronal para maniobras simuladas en TruckSim

Se utilizan tres maniobras realizadas en TruckSim, diferentes a las utilizadas para el entrenamiento de la red neuronal, para simular la red propuesta. Estas maniobras son:

- Doble cambio de carril a 65 Km/h y μ = 0,75 (Figura 44)
- Simple cambio de carril a 55 Km/h y μ = 0,75 (Figura 45)
- Slalom a 55 Km/h y μ = 0,75 (Figura 46)

Figura 44. Doble cambio de carril a 65 Km/h

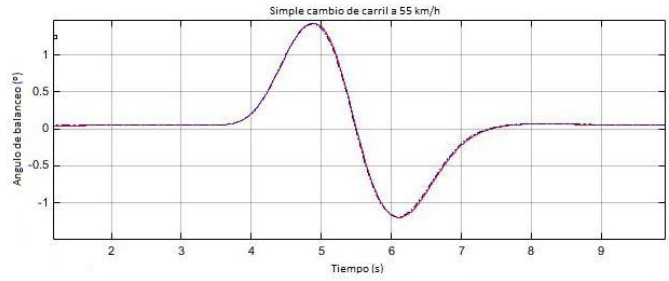


Figura 45. Simple cambio de carril a 55 Km/h

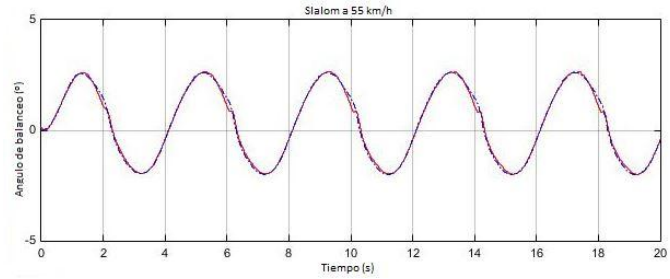


Figura 46. Slalom a 55 Km/h

Como se observa en las anteriores figuras, para maniobras simuladas en el software TruckSim, la red neuronal estima el ángulo de balanceo con gran precisión aunque estas maniobras tengan velocidades y coeficientes de fricción distintos a las maniobras que se utilizaron para el entrenamiento de la misma.

Por tanto, a la vista de los resultados obtenidos en estos dos apartados, se puede validar la red neuronal propuesta.

5.4. Validación de la red neuronal con influencia de la suspensión

Como se indicó anteriormente la adición de las deformaciones de las suspensiones delanteras y traseras mejora notablemente la estimación del ángulo de balanceo que proporciona la red neuronal.

Dado que no se disponen de los datos experimentales de las deformaciones de las suspensiones, en este trabajo se simula una red que tiene como entradas todas las variables que se obtuvieron de TruckSim (Tabla 21) mediante una serie de maniobras realizadas en dicho software, proponiendo para futuros trabajos la simulación de la misma con maniobras experimentales, tal y como se realiza en el apartado anterior.

Se utilizan seis maniobras con distintas velocidades y distintos coeficientes de fricción respecto a los que se usaron para entrenar a la red. Estas seis maniobras son:

- Doble cambio de carril a 65 km/h y μ = 0,6 (Figura 47).
- Doble cambio de carril a 85 km/h y μ = 0,75 (Figura 48).
- Simple cambio de carril a 55 km/h y μ = 0,6 (Figura 49).
- Simple cambio de carril a 75 km/h y μ = 0,75 (Figura 50).
- Slalom a 45 km/h y μ = 0,6 (Figura 51).
- Slalom a 55 Km/h y μ = 0,75 (Figura 52).

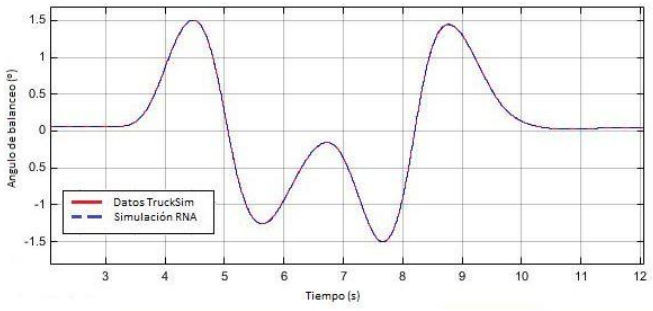


Figura 47. Doble cambio de carril a 65 km/h

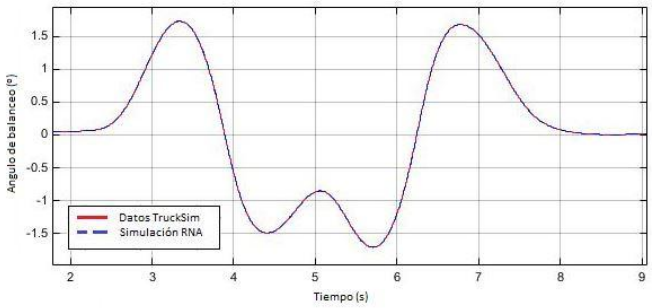


Figura 48. Doble cambio de carril a 85 km/h

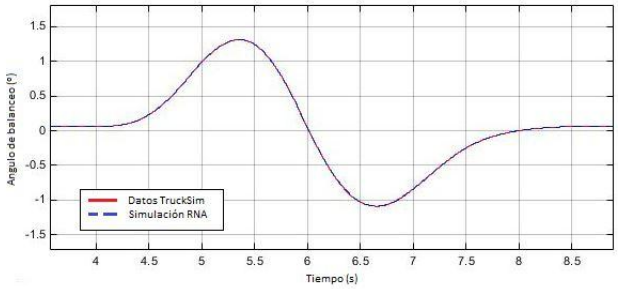


Figura 49. Simple cambio de carril a 55 km/h

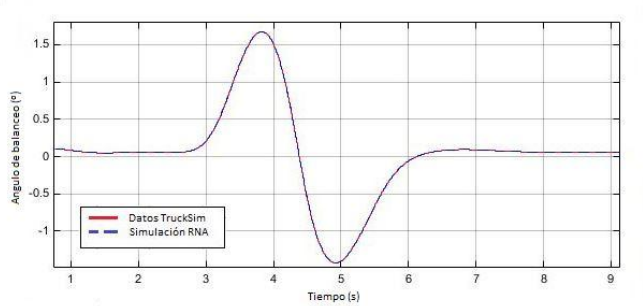


Figura 50. Simple cambio de carril a 75 km/h

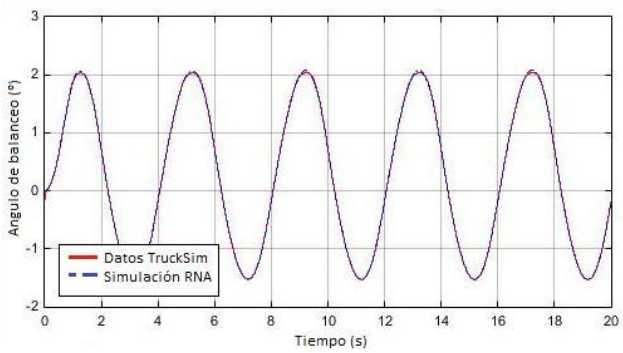


Figura 51. Slalom a 45 km/h

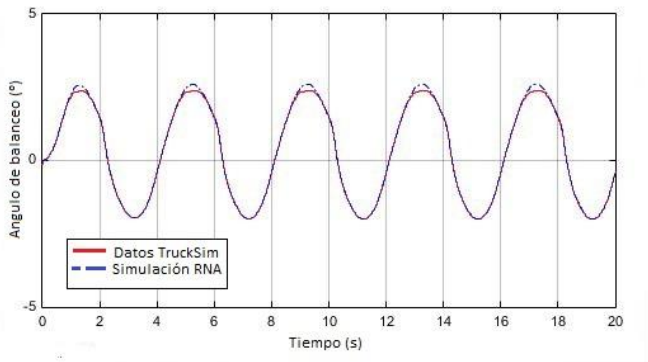


Figura 52. Slalom a 55 km/h

Las figuras anteriores muestran que la estimación del ángulo de balanceo que proporciona esta red es idéntica a los datos que proporciona TruckSim para dicha variable. El ajuste que proporciona esta red para maniobras simuladas en TruckSim es superior al que proporciona la red propuesta en el apartado anterior, y esto unido a que el error en el entrenamiento era notablemente inferior, indica que esta red dará una estimación del ángulo de balanceo para maniobras experimentales mucho más precisa. Por tanto es recomendable realizar un trabajo de mejora de la red validada en el apartado anterior cuando se dispongan de sensores de deformaciones de las suspensiones.

Capítulo 6

Conclusiones y desarrollos futuros

6.1. Conclusiones

En este trabajo se ha desarrollado un estimador del ángulo de balanceo de vehículos industriales mediante la utilización de redes neuronales y se comprobado su validez utilizando ensayos experimentales. En base al trabajo realizado se puede concluir lo siguiente:

- Es esencial la disponibilidad de datos experimentales fiables para poder desarrollar un modelo de vehículo en TruckSim que se asemeje al vehículo real, y de esta manera la red neuronal será entrenada con datos similares a los que el vehículo estará sometido cuando circule en carretera.
- Se ha desarrollado con éxito un modelo de vehículo en TruckSim similar al vehículo experimental y se ha comprobado en dicho software como los parámetros relativos a la dirección y a la suspensión afectan de manera significativa a la dinámica del vehículo.
- Se ha desarrollado la programación de una red neuronal entrenada con un algoritmo BP del tipo Levenberg-Marquardt en Matlab y se ha probado su validez. Así mismo, se han analizado las distintas posibilidades que ofrecía dicha programación, tales como las distintas funciones de activación o de entrenamiento, y finalmente se ha utilizado la opción que proporcionaba menor error en la estimación del ángulo de balanceo.
- Mediante un análisis de sensibilidad, en el cual se han entrenado 90 redes neuronales diferentes, se ha determinado que una red neuronal con 4 entradas a la red (aceleración lateral, ángulo de dirección, variación del ángulo de balanceo y variación del ángulo de guiñada), 3 neuronas en la capa oculta y datos correspondientes a 39 maniobras para cada variable proporciona la mejor estimación del ángulo de balanceo y a su vez es capaz de generalizar y estimar este ángulo para maniobras diferentes a las usadas para el entrenamiento de la misma. Se ha validado esta red tanto con maniobras experimentales como con maniobras simuladas en TruckSim.
- Se ha comprobado que la incorporación de los datos de las deformaciones de las suspensiones como entradas a la red mejora la capacidad de predicción de la red neuronal y se ha validado esta red con 9 entradas usando maniobras experimentales.

Por tanto, se puede concluir que se han alcanzado los objetivos propuestos durante el desarrollo de este proyecto.

6.2. Desarrollos futuros

Debido al número tan elevado de víctimas mortales en accidentes de tráfico que se producen en las carreteras los trabajos e investigaciones que se realicen en material de seguridad en los vehículos no deben detenerse. Por tanto, desarrollado este estimador, se puede contemplar como desarrollo futuro lo siguiente:

- Implementar esta red neuronal desarrollada en sistemas de seguridad activa, de manera que dichos sistemas sean capaces de dar un aviso al conductor o efectuar acciones correctivas en los componentes del vehículo cuando exista peligro de inestabilidad dinámica del mismo.
- Tal y como se explica en el apartado 5.4, se debe considerar la instalación de sensores de desplazamiento en el vehículo experimental para medir las deformaciones de las suspensiones para que de esta manera la red neuronal propuesta pueda ser validada considerando dichas deformaciones.
- Realizar maniobras experimentales adicionales, tales como una maniobra de giro en J o un cambio de sentido realizando un giro de 180° de manera que la red neuronal sea entrenada con más datos diferentes y por tanto sea capaz de estimar el ángulo de balanceo en un rango más amplio de posibles situaciones de riesgo.
- Dado que el campo de las redes neuronales está en continuo desarrollo, se puede considerar realizar un estudio similar al propuesto utilizando las redes que se desarrollen en los próximos años.

Bibliografía

[1] D. E. L. Interior, “Ministerio del interior,” 2006.

[2] A. Soltani, A. Goodarzi, M. H. Shojaeefard, and A. Khajepour, “Vehicle dynamics control using an active third-axle system,” *Veh. Syst. Dyn.*, vol. 3114, no. August 2015, pp. 1–22, 2014.

[3] J. Marzbanrad, G. Soleimani, M. Mahmoodi-k, and A. H. Rabiee, “Development of fuzzy anti-roll bar controller for improving vehicle stability,” pp. 3856–3865, 2015.

[4] V. Jorge, “An Algebraic Approach for Maximum Friction Estimation,” in *IFAC Proceedings Volumes*, 2010, vol. 43, no. 14, pp. 885–890.

[5] L. Zhao and Z. Liu, “Vehicle State Estimation with Friction Adaptation for Four-Wheel Independent Drive Electric Vehicle *,” vol. 2014, no. 61104060, pp. 4518–4522, 2014.

[6] H. Eric Tseng, L. Xu, and D. Hrovat, “Estimation of land vehicle roll and pitch angles,” *Veh. Syst. Dyn.*, vol. 45, no. 5, pp. 433–443, 2007.

[7] K. Nam, S. Member, and S. Oh, “Estimation of Sideslip and Roll Angles of Electric Vehicles Using Lateral Tire Force Sensors Through RLS and Kalman Filter Approaches,” vol. 60, no. 3, pp. 988–1000, 2013.

[8] K. Nam, S. Oh, H. Fujimoto, and Y. Hori, “Direct roll moment control for electric vehicles based on roll angle observer and lateral tire force control,” *8th Int. Conf. Power Electron. - ECCE Asia "Green World with Power Electron. ICPE 2011-ECCE Asia*, pp. 2681–2686, 2011.

[9] E. N. Sanchez, L. J. Ricalde, R. Langari, and D. Shahmirzadi, “Rollover Prediction and Control in Heavy Vehicles Via Recurrent High Order Neural Networks,” *Intell. Autom. Soft Comput.*, vol. 17, no. May 2013, pp. 95–107, 2013.

[10] C. Lin, W. Liu, and H. Ren, “Neural network-PID control algorithm for semi-active suspensions with magneto-rheological damper,” pp. 4432–4445, 2015.

[11] S. Cong and Y. Liang, “PID-Like Neural Network Nonlinear Adaptive Control for Uncertain Multivariable Motion Control Systems,” *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 3872–3879, 2009.

[12] R. Goldman, M. El-Gindy, and B. Kulakowski, “Development of a software-based rollover warning device,” *Int. J. Heavy Veh. Syst.*, vol. 12, no. 4, p. 1, 2005.

[13] Z. Man, H. R. Wu, S. Liu, X. Yu, and S. Member, “A New Adaptive Backpropagation Algorithm Based on Lyapunov Stability Theory for Neural Networks,” vol. 17, no. 6, pp. 1580–1591, 2006.

[14] R. Hecht-Nielsen, “the Backpropagation Neural Network,” *Processing*, vol. 1, pp. 593–605, 1989.

[15] S. Valero, J. Aparicio, C. Senabre, M. Ortiz, J. Sancho, and a. Gabaldon, "Comparative analysis of self organizing maps vs. multilayer perceptron neural networks for short-term load forecasting," *2010 Mod. Electr. Power Syst.*, no. 1, pp. 1–5, 2010.

[16] Hilera, J.R., Martínez, V. J. Redes neuronales artificiales. *Fundamentos, modelos y aplicaciones*. España: Rama, 1995.

[17] T. Kohonen, "An introduction to neural computing," *Neural Networks*, vol. 1, no. 1, pp. 3–16, Jan. 1988.

[18] R. Hecht-Nielsen, "Applications of counterpropagation networks," *Neural Networks*, vol. 1, no. 2, pp. 131–139, Jan. 1988.

[19] D. J. Matich, "Redes Neuronales: Conceptos Básicos y Aplicaciones.," *Historia Santiago.*, p. 55, 2001.

[20] Haykin, S. Neural Networks, a comprehensive foundation. Singapore: Pearson Education, 2001.

[21] *The Graphical User of BikeSim, Carsim, and TruckSim.* Mechanical Simulation Corporation, Michigan, pp. 92, 2009.

Anexo A

Desarrollo del modelo de furgoneta mediante el software TruckSim

TruckSim se basa en la creación de conjuntos de datos para cada componente del vehículo tales como sistema de suspensión, sistema de dirección, masa suspendida, tren de potencia, etc. La Figura A-1 muestra el conjunto de vehículos que contiene TruckSim. Se pueden disponer desde una combinación de vehículos cargados, hasta unidades principales con dos, tres y cuatro ejes.

Una vez que una de esas opciones es seleccionada, se puede crear un nuevo conjunto de datos para el camión/furgoneta pulsando la fecha y nombrando el conjunto de datos tal como se muestra en la Figura A-2, en este caso se usa el nombre de ¨Large European Van_Mercedes¨. Una vez que el conjunto de datos es creado, aparece un recuadro azul como se muestra en la Figura A-3. Pulsando este recuadro aparece una pantalla (Figura A-4) donde se introducen los diferentes parámetros del vehículo. Es necesario crear un conjunto de datos individual para cada parámetro.

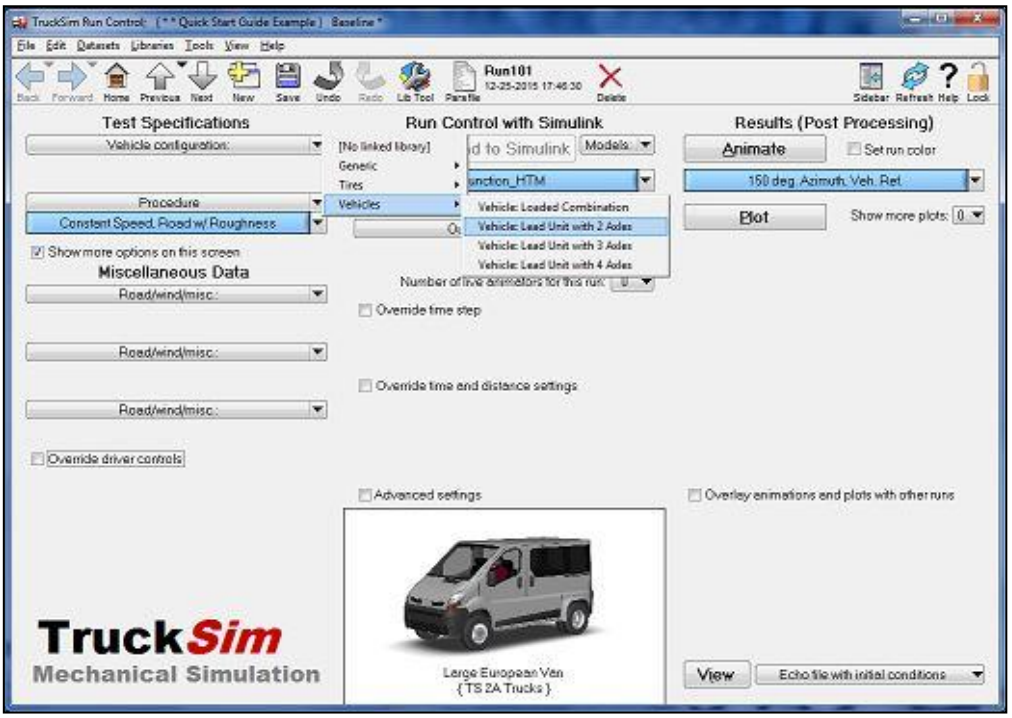


Figura A-1. Conjunto de vehículos

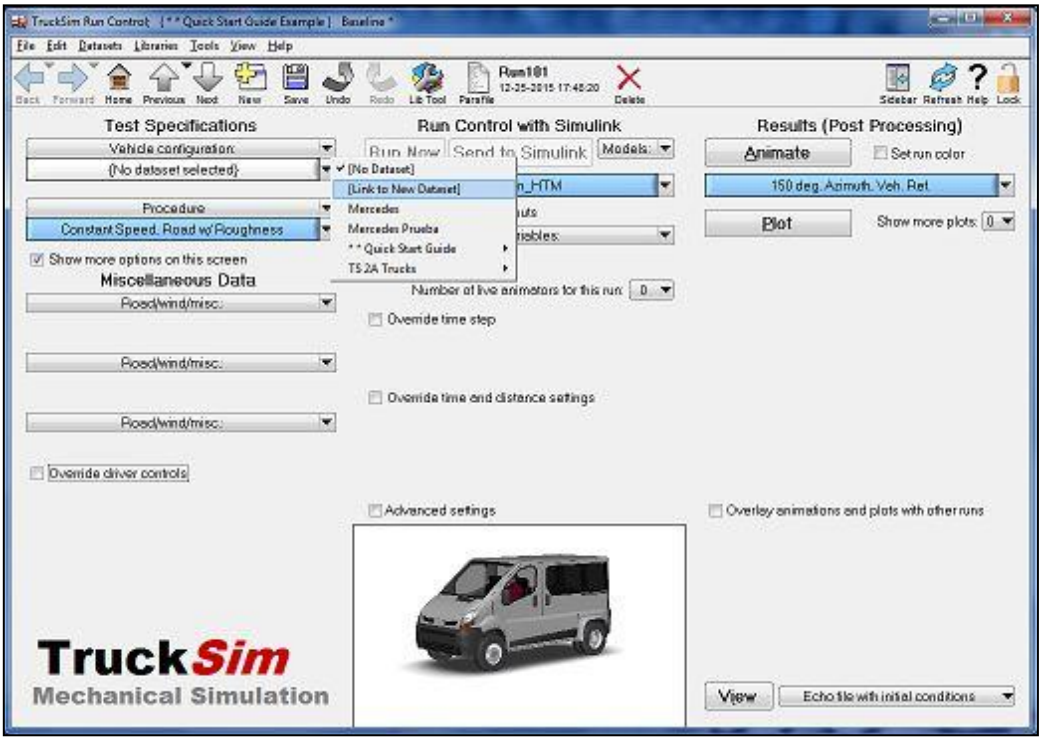


Figura A-2. Creación del conjunto de datos

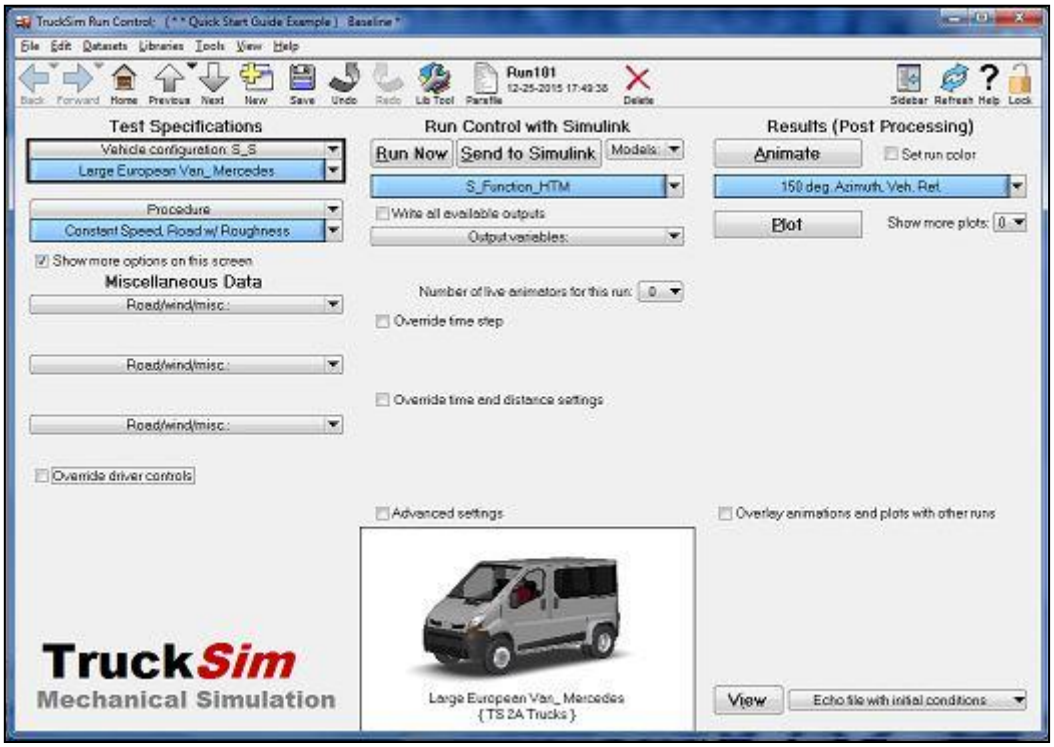


Figura A-3. Ejemplo del conjunto de datos Large European Van_Mercedes

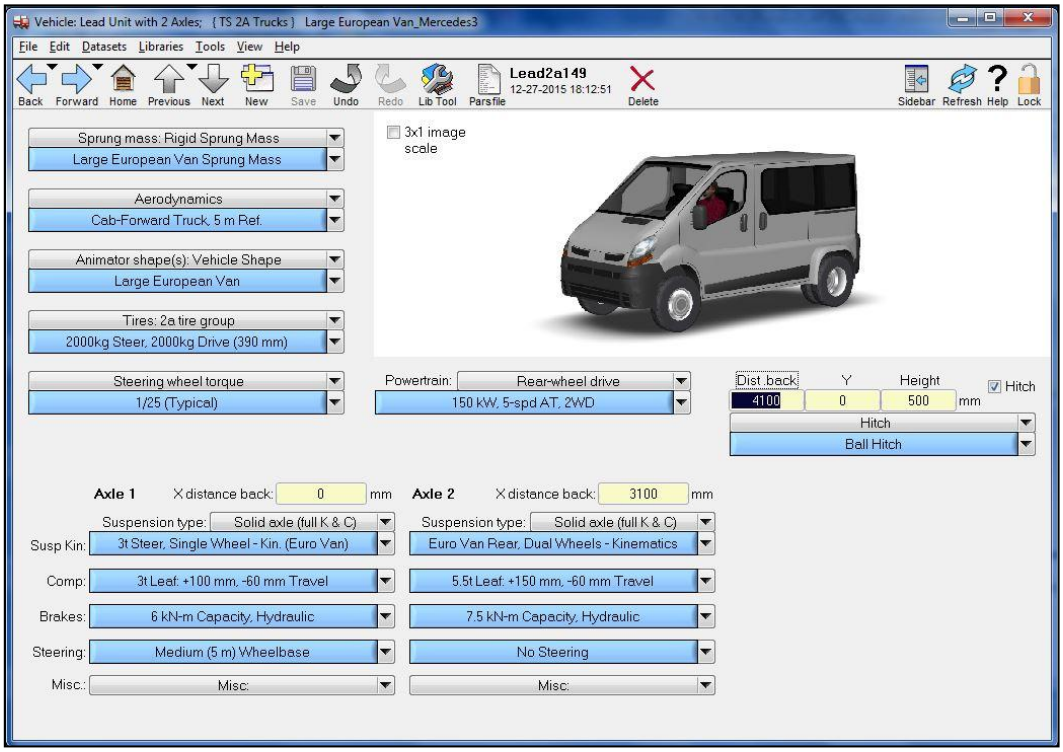


Figura A-4. Parámetros del modelo Large European Van_Mercedes

- Introducción de los parámetros de la furgoneta

La Figura A-4 muestra todos los parámetros que componen el modelo, siendo estos la masa suspendida, suspensiones, cinemática de dirección, llantas, frenos, aerodinámica y controles de animación. En los siguientes pasos se muestra como se modelan cada una de estas opciones.

La obtención de estos valores óptimos se consiguió mediante un trabajo de ajuste de cada uno individualmente. Tal y como se explica en los siguientes apartados, tan solo se ajustan los parámetros que realmente afectan a las dinámicas del vehículo.

1) Masa suspendida, inercia del vehículo y centro de gravedad

La localización del centro de gravedad (cg) de la masa suspendida siempre esta referenciada con respecto al eje delantero. Por definición, la masa suspendida del vehículo consiste en toda la masa soportada por la suspensión y la mitad del peso de los componentes de la suspensión tales como las ballestas y amortiguadores.

El sistema de coordenadas usado por TruckSim se muestra en la Figura A-5.

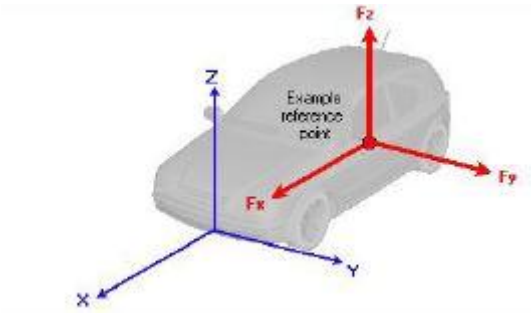


Figura A-5. Sistema de coordenadas usado por TruckSim [21]

Los valores de inercia se referencian con respecto a los tres ejes coordenados. El eje X es el eje de balanceo, el eje Y es el de cabeceo y el Z es el de guiñada.

Para introducir los datos mostrados en la Tabla 7 al modelo se pulsa la flecha del recuadro mostrado en la Figura A-4 con el nombre de ¨Sprung Mass¨ y se selecciona la opción ¨Vehicle: Lead Unit Sprung Mass¨. A continuación se pulsa la flecha que aparece en la parte inferior y en la categoría ¨2A Trucks: Sprung Mass¨ se escoge la opción de ¨Large European Van Sprung Mass¨. Una vez realizados estos pasos se pulsa sobre el recuadro azul con el nombre anterior y se accede a la pantalla mostrada en la Figura A-6 donde se introducen los mencionados datos.

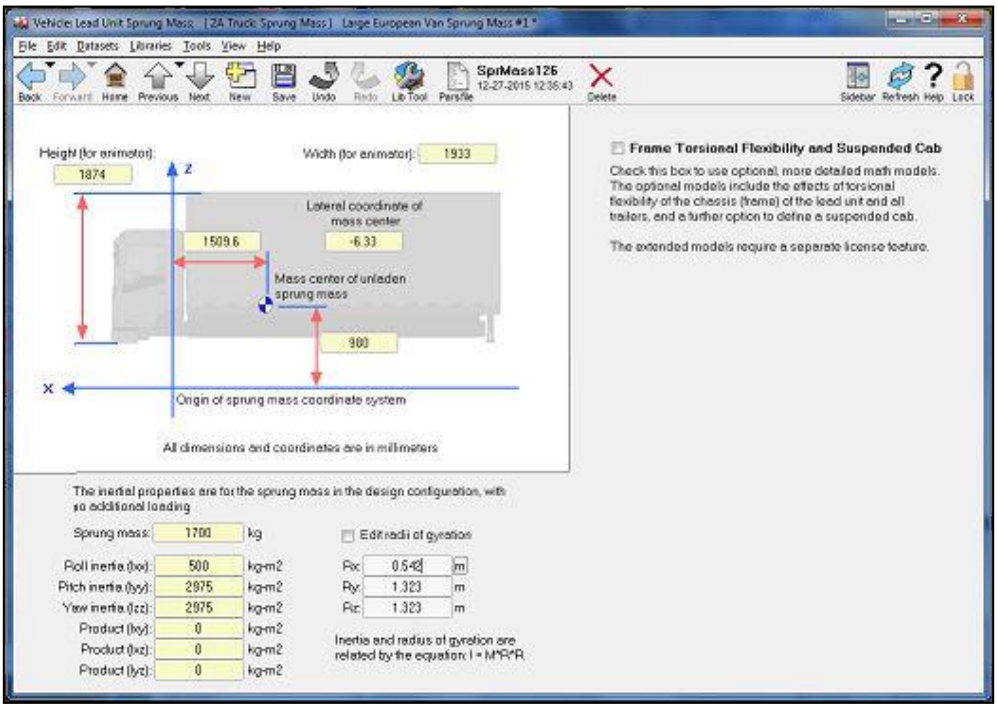


Figura A-6. Masa suspendida, cg e inercias del modelo

2) Tren de potencia y transmisión

El motor usado por este modelo será el que TruckSim ofrece por defecto para este tipo de furgoneta. Como se puede apreciar en la Figura A-4 en el apartado ¨Powertrain¨, es un vehículo de tracción trasera (¨Rear-wheel drive¨), con una potencia nominal de 150 KW y 5 velocidades. Pulsando el recuadro azul de dicho apartado se accede a la pantalla en la cual se pueden modelar los datos del motor, caja de transmisión, convertidor de par y diferenciales delantero y trasero (Figura A-7).

La variación del par del motor en función de las revoluciones por minuto (rpm) del motor y de la posición del acelerador se muestra en la Figura A-8. Las líneas de colores representan las diferentes posiciones del acelerador.

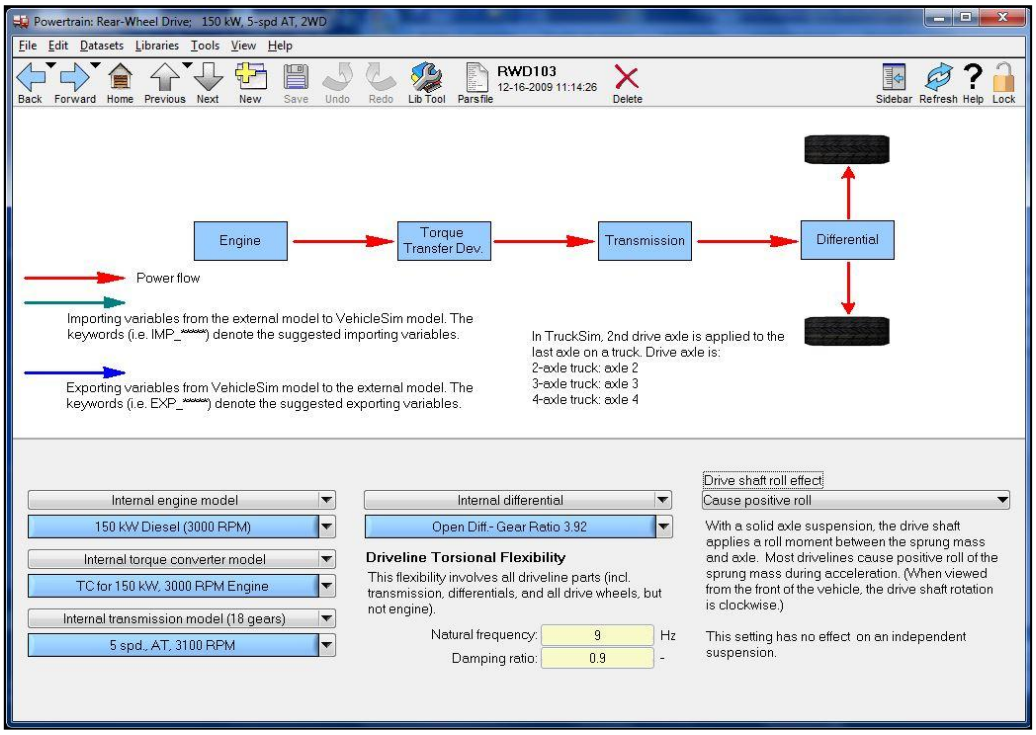


Figura A-7. Módulo del tren de potencia

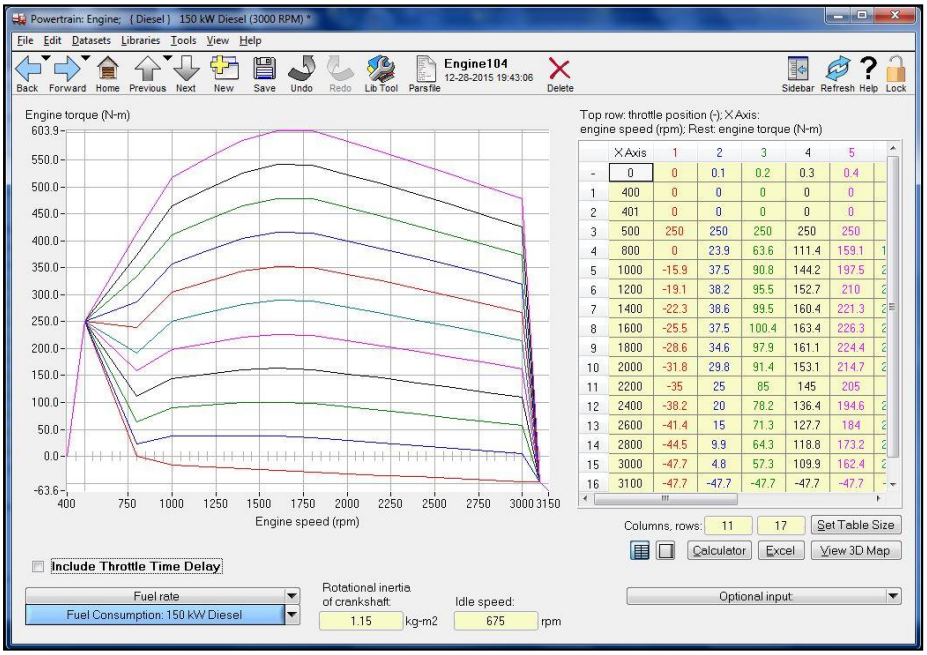


Figura A-8. Variación del par del motor en función de las rpm del motor y de la posición del acelerador

El modelo de convertidor hidráulico de par que utiliza esta furgoneta, el cual transmite el par desde el motor hasta la transmisión, se puede modelar accediendo a la pantalla de la Figura A-9. La gráfica superior muestra el inverso de la capacidad del convertidor de par en función de la relación de velocidad, mientras que la gráfica inferior muestra la amplificación de par en función de la relación de velocidad.

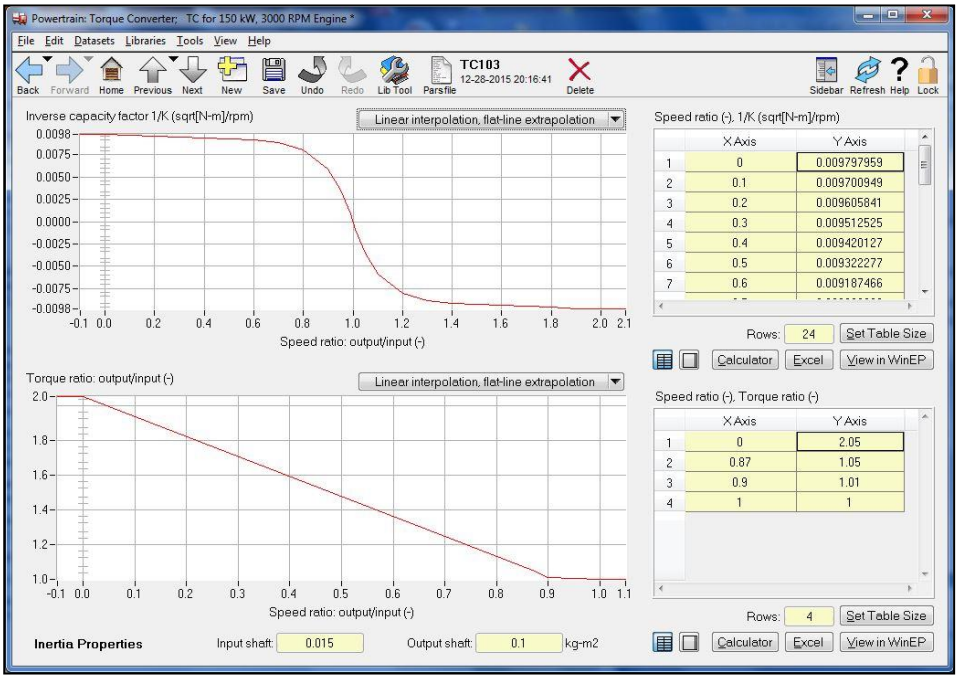


Figura A-9. Parámetros del convertidor de par

Los valores intermedios de ambas gráficas se calculan mediante un método de extrapolación lineal, extrapolación de línea plana de TruckSim (el método usado se aprecia en la parte superior de ambas gráficas).

La caja de cambios de la transmisión, tal y como se mencionó anteriormente, consiste en 5 velocidades con las cuales proporciona una amplia variedad de velocidades. Se mantienen los datos de relación de transmisión, inercia y eficiencia de cada marcha que proporciona TruckSim por defecto, los cuales se muestran en la Figura A-10. En esta figura también se observa que el tiempo para cambiar de una marcha a otra es de 0,25 s (¨shift duration: 0,25 sec¨).

La Figura A-11 muestra una instantánea de la secuencia de cambios para la primera relación de transmisión en función de la posición del acelerador.

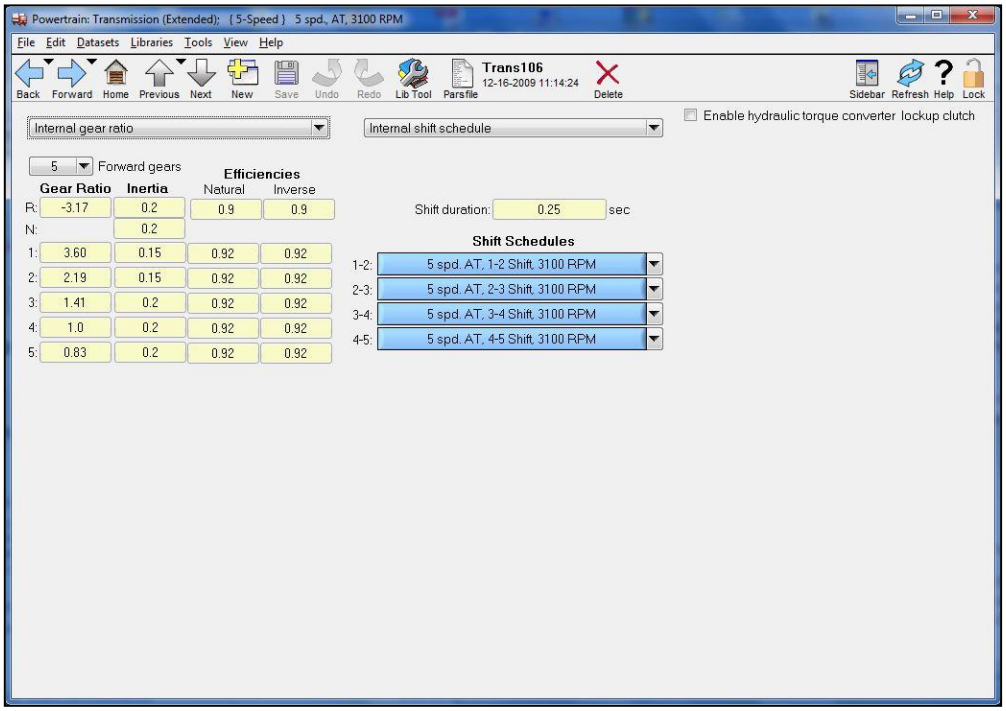


Figura A-10. Caja de cambios de 5 velocidades

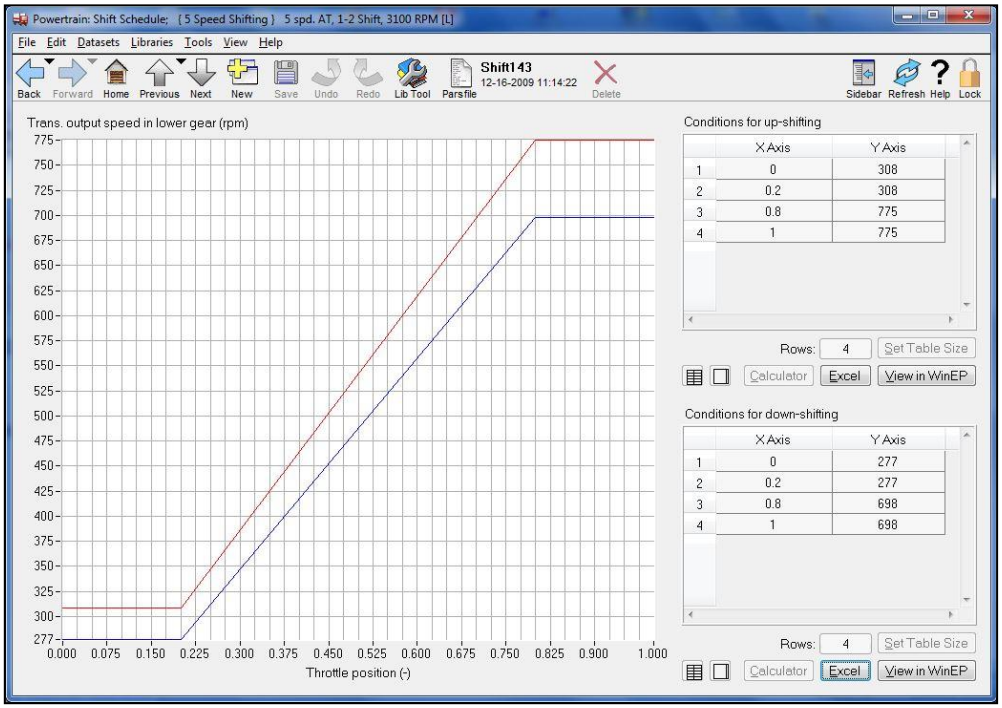


Figura A-11. Secuencia de cambios para la primera marcha

3) Sistema de dirección y modelo de par de frenado

Sistema de dirección: Los parámetros del sistema de dirección afectan significativamente a la dinámica del vehículo. La Figura A-12 muestra una instantánea de TruckSim del módulo del sistema de dirección. A esta pantalla se accede eligiendo una opción de las disponibles del modulo ¨Steering¨ de la Figura A-4 y pulsando sobre esa opción. Una vez que se accede a dicha pantalla se modifican los distintos componentes, hasta alcanzar los valores que se muestran en dicha figura y que se incluyen en la Tabla 8.

El módulo ¨kingpin geometry¨ que aparece en la esquina inferior derecha de la pantalla consiste en la inclinación lateral, el desplazamiento lateral, la coordenada X y el ángulo de avance. Estos parámetros ayudan a definir la geometría de los diferentes vínculos de dirección.

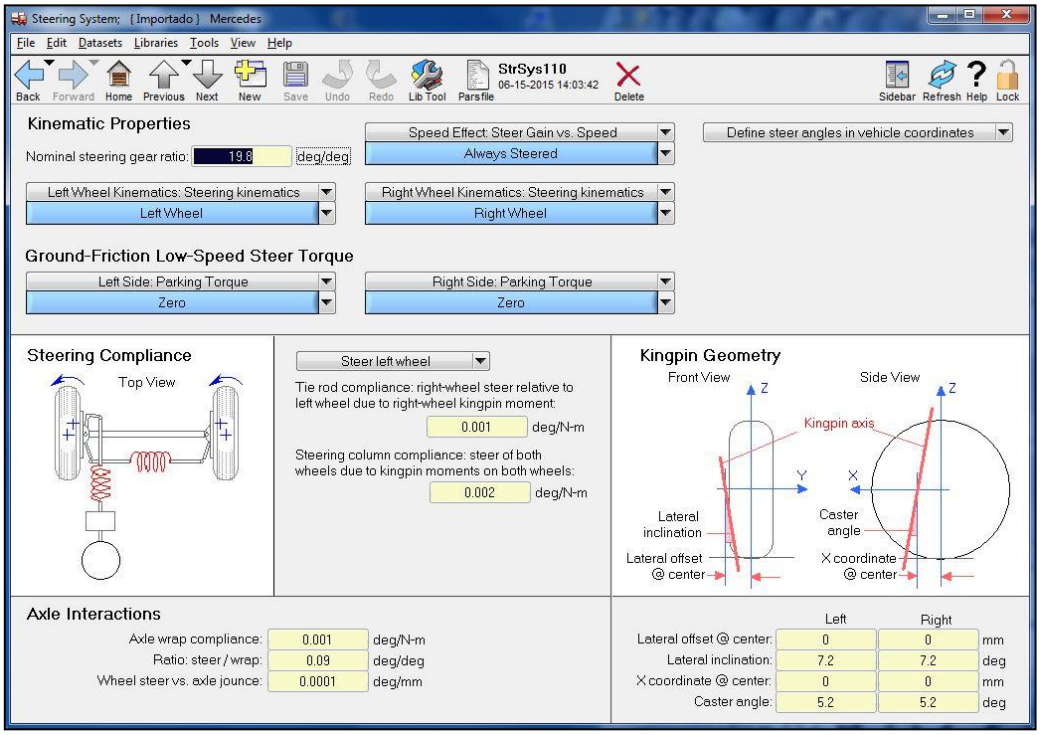


Figura A-12. Modulo del sistema de dirección

Después se necesita especificar una relación de transmisión para la caja de cambios, la cual en el modelo propuesto es de 19,8. Basándose en la relación de transmisión TruckSim calcula el ángulo de salida de la caja de cambios como sigue:

$$Angulo\ de\ salida\ de\ la\ caja\ de\ cambios = \frac{Angulo\ del\ volante}{Relación\ de\ transmisión}$$

Este ángulo es luego usado como una entrada para una tabla de búsqueda que grafica la cinemática de dirección para las ruedas izquierda y derecha como se muestra en las Figuras A-13 y A-14, respectivamente. Estas gráficas tienen en cuenta el error de Ackermann[3]. Tal y como se observa la gráfica es lineal hasta cierto rango de ángulo de dirección y después se curva debido al error de Ackermann en grandes ángulos de dirección.

[3] El principio de Ackermann enuncia que cuando un vehículo gira en una curva, los ejes de todas las ruedas deben concurrir en un punto, el centro instantáneo de rotación. El cuadrilátero de Ackermann permite que esta condición se satisfaga con una aproximación bastante buena.

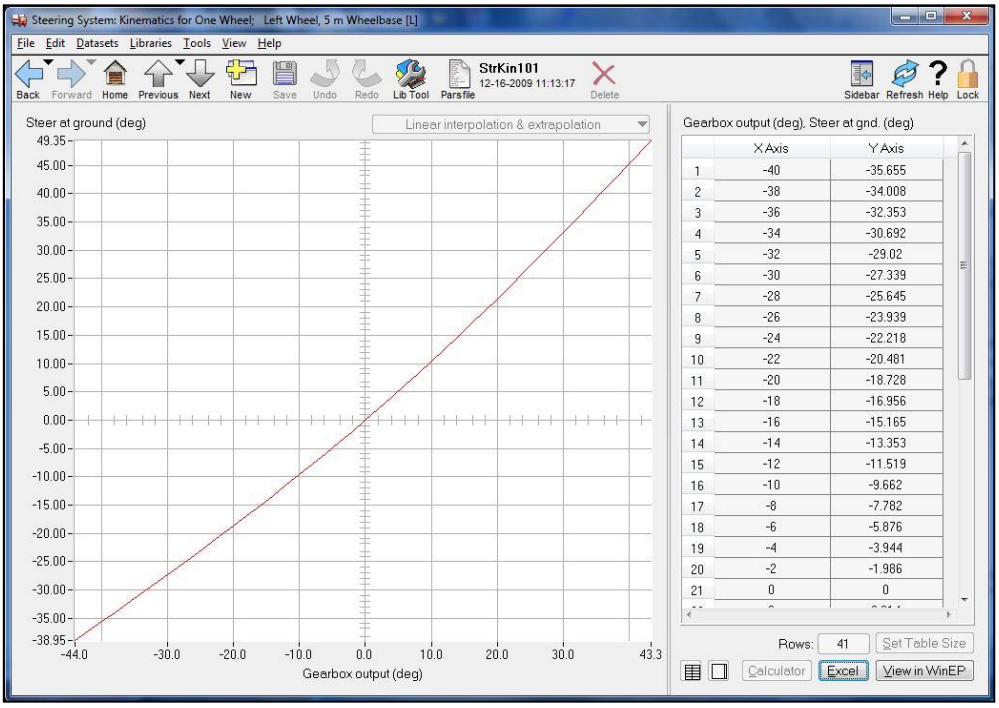


Figura A-13. Cinemática de dirección de la rueda izquierda

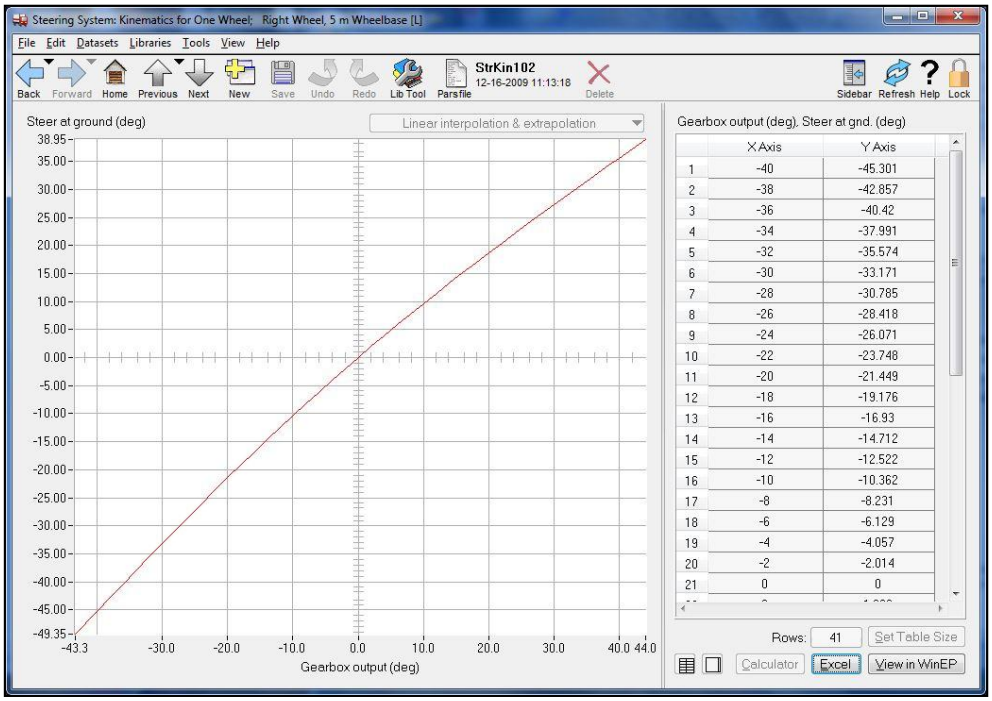


Figura A-14. Cinemática de dirección de la rueda derecha

Modelo de par de frenado: Este modelo de furgoneta incorpora frenos de disco hidráulicos de 6 KN-m de capacidad de frenado para el eje delantero y de 7,5 KN-m para el eje trasero. La Figura A-15 muestra una instantánea del módulo de freno para el eje delantero.

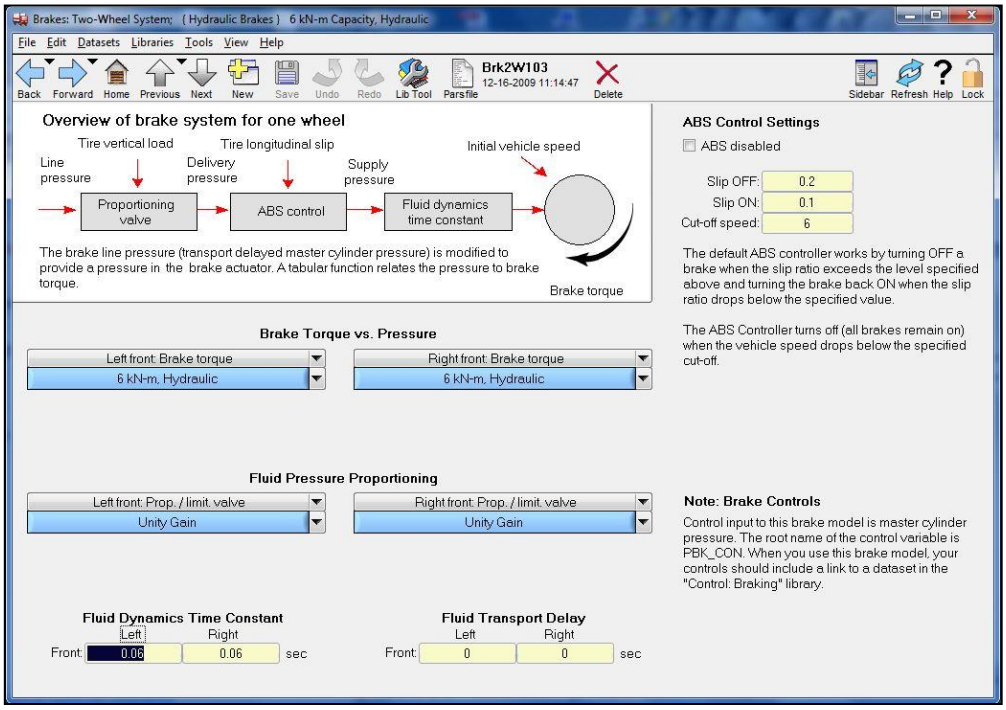


Figura A-15. Modulo de freno

También es necesario especificar el tiempo constante de la cámara de freno y el retraso en el transporte del fluido. Como la furgoneta utiliza un sistema electrónico de frenado (EBS ≡ Electronic Braking System) el retraso en el transporte se toma como 0. El tiempo de la cámara de freno según la definición de TruckSim, es el tiempo requerido para que la presión aumente un 33,33% (un tercio) en la cámara de freno. Estos valores están tabulados en la siguiente tabla.

Tabla 23. Tiempo de la cámara de frenos y retraso en el transporte

Parámetro	Valor	Unidad
Tiempo de la cámara de freno (eje delantero)	0,06	s
Tiempo de la cámara de freno (eje trasero)	0,06	s
Retraso en el transporte del fluido (ambos ejes)	0	s

4) Cinemática de suspensión

Este punto cubre todos los parámetros importantes de la suspensión. Tiene un gran número de entradas. La Figura A-16 muestra el módulo de suspensión de TruckSim.

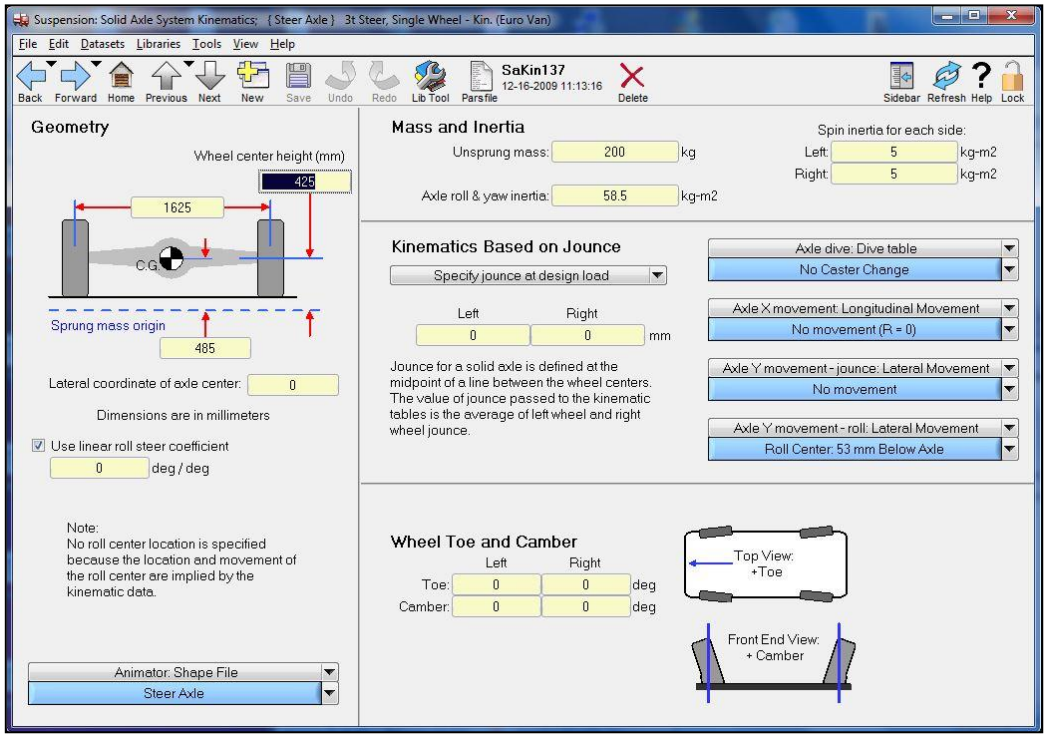


Figura A-16. Módulo de suspensión del modelo

Los parámetros más importantes se explican a continuación:

- Masa no suspendida (¨Unsprung Mass¨): Consiste en la masa del eje, de la rueda, de la llanta y la mitad de la masa de la suspensión de ballesta y de los amortiguadores.

- Eje de balanceo e inercia de guiñada (¨Axle Roll and Yaw Inertia¨): Es la inercia combinada del eje, rueda y llanta alrededor del eje de balanceo (Eje X) o del eje de guiñada (Eje Z). Esta inercia se calcula alrededor del cg del eje. La inercia de balanceo y la de guiñada tienen prácticamente el mismo valor. Por tanto cualquiera de las dos se puede introducir en TruckSim. Los valores no deben sumarse.

- Dirección de balanceo (¨Roll Steer¨): Esta es la cantidad con la que la rueda izquierda de la furgoneta sería dirigida en función del balanceo del vehículo. Este parámetro es muy importante ya que un cambio en el ángulo de dirección podría afectar la aceleración lateral y la velocidad de guiñada. El signo de este parámetro afecta a la dirección a la cual las ruedas son dirigidas.

- Inercia de giro (¨Spin Inertia¨): Este es el momento de inercia de la masa de rotación en el sistema de eje tales como engranajes diferenciales y semiejes. La inercia de giro de la combinación de rueda y llanta se considera en modulo de llanta de TruckSim.

- Eje de picado (¨Axle Dive¨): Es una tabla que representa la rotación de los componentes de la suspensión en función de la compresión de la suspensión. Un ángulo de picado positivo implica

una rotación en sentido anti horario de la suspensión alrededor de un eje lateral según se ve desde el lado izquierdo.

- Movimiento longitudinal del eje debido al vaivén de la suspensión (¨Axle longitudinal movement due to Suspension Jounce¨): Es una tabla que representa el movimiento horizontal del centro de la rueda o del centro del eje en función de la compresión o vaivén de la suspensión.

- Movimiento lateral del eje debido al vaivén de la suspensión (¨Axle lateral movement due to Suspension Jounce¨): Es una tabla que representa el movimiento lateral hacia la izquierda del centro del eje en función de la compresión o vaivén de la suspensión.

- Movimiento lateral del eje debido al balanceo del eje (¨Axle lateral movement due to Axle Roll¨): Es una tabla que representa el movimiento lateral hacia la izquierda del centro del eje debido al balanceo de la suspensión. El eje de balanceo es positivo cuando la rueda derecha se mueve hacia abajo con respecto a la masa suspendida.

Los valores que adopta el modelo para los parámetros del módulo de suspensión para el eje conducido y para el eje tractor están incluidos en la Tabla 9.

5) Componentes de la suspensión

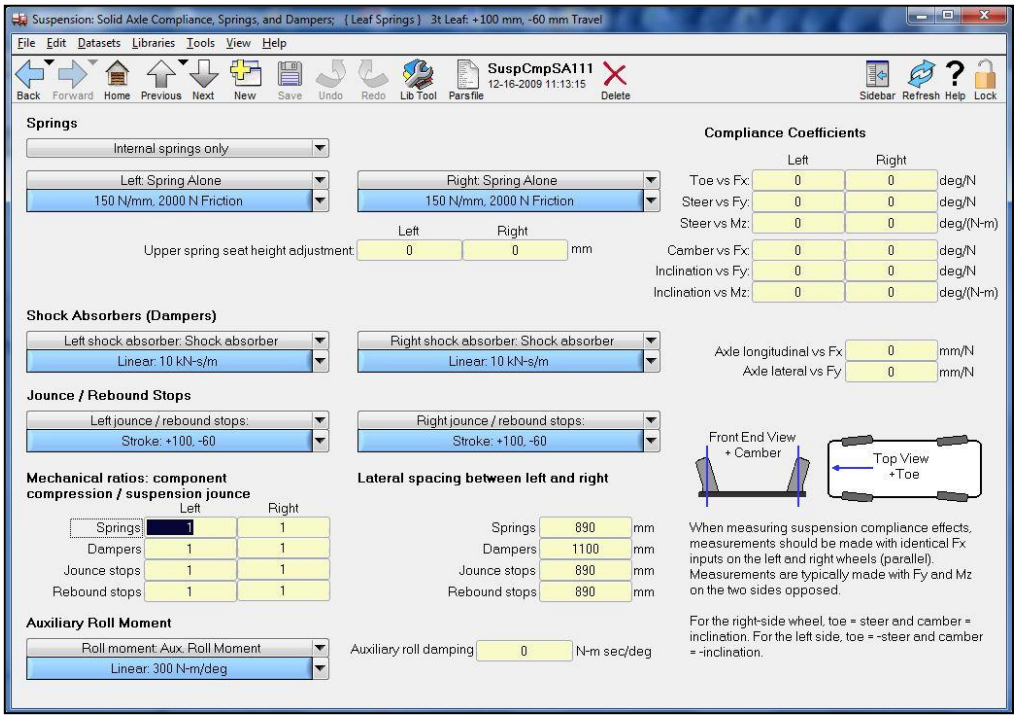


Figura A-17. Componentes de la suspensión

En este punto se modelan varios componentes del sistema de suspensión tales como la ballesta, el amortiguador, los coeficientes de suspensión, las relaciones mecánicas y el momento auxiliar de balanceo. La Figura A-17 muestra una instantánea del módulo de componentes para el eje conducido

A continuación se explican brevemente los parámetros más importantes mostrados en la Figura A-17:

- Toe vs Fx: Representa el movimiento hacia el interior o el movimiento hacia fuera de las ruedas izquierda y derecha simultáneamente. Se define como positiva cuando la rueda se dirige hacia dentro, hacia el eje de balanceo del vehículo (Eje X) y negativa cuando se dirige hacia afuera. Fx se define como el efecto puntera, porque en una suspensión simétrica las dos ruedas serán dirigidas ambas hacia dentro o hacia afuera debido a la fuerza aplicada. Un valor positivo quiere decir que una fuerza longitudinal positiva causará que las ruedas sean dirigidas hacia dentro.

- Steer vs Fy: Este parámetro induce algún ángulo de dirección debido a las fuerzas laterales que actúan sobre toda la superficie de contacto. Se define positivo cuando la rueda se somete a una rotación positiva alrededor del eje Z (hacia la izquierda). En este caso, la fuerza aplicada causará que las ruedas de una suspensión simétrica sean dirigidas ambas a la izquierda o a la derecha. Un valor positivo para este parámetro significa que una fuerza lateral positiva (hacia la izquierda) causaría que las ruedas sean dirigidas hacia la izquierda.

- Steer vs Mz: Este parámetro es similar al anterior con similares convenciones de signos para el ángulo de dirección. Sin embargo, el momento de alineación aplicado hará que las ruedas de una suspensión simétrica sean dirigidas ambas a la izquierda o a la derecha. Un valor positivo para este parámetro significa que un momento de alineación positivo (en contra de las agujas del reloj alrededor del eje Z) causará que las ruedas sean dirigidas hacia la izquierda.

- Camber vs Fx: Es la cantidad de la parte superior de la rueda que se inclina hacia el exterior o hacia el interior desde el plano de la rueda. Este parámetro se define positivo cuando la rueda se inclina hacia el exterior en la parte superior y negativa cuando se inclina hacia el interior. Un valor positivo para este parámetro significa que una fuerza longitudinal positiva causará que las ruedas se inclinen hacia afuera en la parte superior.

- Inclination vs Fy: La inclinación es el ángulo al cual el eje de dirección es inclinado con respecto al plano del centro de la rueda. Una fuerza lateral hace variar ligeramente esta inclinación debido a las conformidades en los vínculos de dirección. Una inclinación positiva significa que la rueda se inclina hacia la derecha lo cual es una rotación positiva alrededor del eje X. Un valor positivo para este parámetro significa que una fuerza lateral positiva causará que la rueda se incline hacia la derecha. Estos valores pueden tener diferentes signos para las ruedas izquierda y derecha ya que la inclinación puede variar independientemente.

- Inclination vs Mz: Un valor positivo para este parámetro significa que un momento de alineación positivo (en contra de las agujas del reloj alrededor del eje Z) causará que la rueda se incline hacia la derecha.

- Axle longitudinal vs Fx: Esta es una medida del desplazamiento del eje longitudinal medido desde el centro de la rueda. El desplazamiento normalmente ocurre en la misma dirección que la fuerza. Por tanto un valor positivo para este parámetro significa que una fuerza longitudinal positiva provoca que eje se mueva hacia adelante en la dirección positiva del eje X.

- Axle lateral vs Fy: Esta es una medida del desplazamiento del eje lateral medido desde el centro de la rueda. Un valor positivo para este parámetro significa que una fuerza lateral positiva (hacia la izquierda) provoca que el eje se mueva hacia adelante en la dirección positiva del eje Y (hacia la izquierda del vehículo).

- Relación mecánica: Se define como la relación del componente de compresión para el rebote de la suspensión. TruckSim utiliza el valor del componente de compresión para determinar el valor de la fuerza de rebote tope. La fuerza de rebote tope es igual al rebote de la suspensión multiplicado por la relación mecánica.

- Momento auxiliar de balanceo/ rigidez: Esta es la contribución del momento de balanceo a otros componentes auxiliares de suspensión tales como el estabilizador de balanceo y el marco de rigidez. El momento auxiliar de balanceo actúa en paralelo con la suspensión para aumentar (a veces disminuir) la rigidez al balanceo global del vehículo. Las Figuras A-18 y A-19 ilustran cómo se calcula la rigidez al balanceo debido a las ballestas.

Rigidez al balanceo total = rigidez al balanceo debido a las ballestas + rigidez al balanceo auxiliar

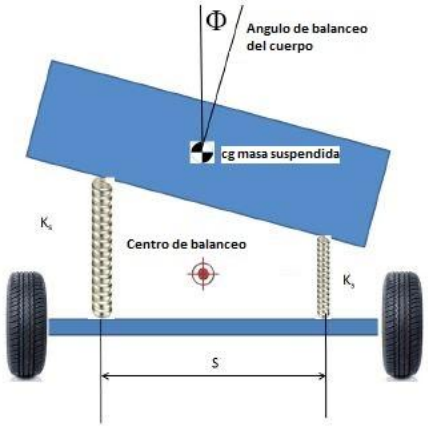


Figura A-18. Rigidez al balanceo debido a los muelles de suspensión

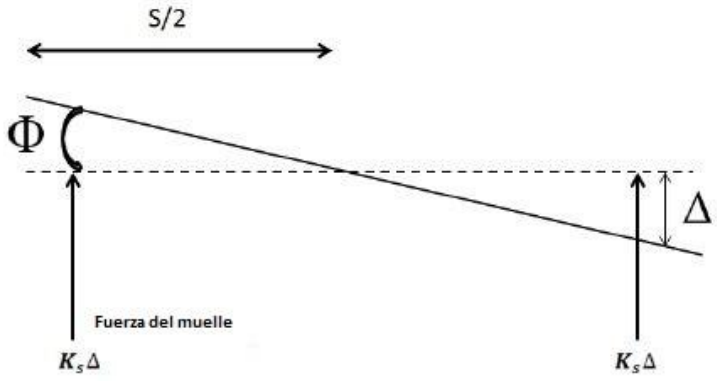


Figura A-19. Representación esquemática de la fuerza del muelle

$$\sin\phi = \frac{\Delta}{S/2} \approx \phi \ \ (aproximación\ de\ ángulo\ pequeño) \qquad (A.1)$$

$$\Delta = \phi\frac{S}{2} \qquad (A.2)$$

$$K_{\phi\_muelle}.\phi = Momento\ de\ balanceo = \left(2.K_s.\phi.\frac{S}{2}\right)\frac{S}{2} \qquad (A.3)$$

$$K_{\phi\_muelle} = K_s.\frac{S^2}{2} \qquad (A.4)$$

$$K_{total} = K_{\phi\_muelle} + K_{aux} \qquad (A.5)$$

Como se puede apreciar en la Ecuación A.5, el momento total de balanceo aumenta con un aumento en separación lateral entre los muelles dado que esto incrementa el brazo.

- Rebound Stops: La función de esta parada es prevenir un excesivo movimiento o compresión de las ballestas. El tope de suspensión actúa como un muelle virtual y proporciona una fuerza de reacción extra debido a su naturaleza elástica. Normalmente el tope de suspensión produce un fuerza de valor 0 hasta cierto valor de compresión y cuando la compresión es superior al valor límite, empieza a producir fuerzas de reacción.

- Shock Absorbers & Leaf Spring: Estos componentes se encuentran en la mayoría de los vehículos pesados. Los muelles proporcionan flexibilidad vertical al aislar el chasis de las superficies irregulares. Tambien resiste el balanceo del chasis y reacciona a las fuerzas de reacción producidas por las ruedas debido a la fricción con la carretera.

En las tablas 10 y 11 se muestran los valores tomados para los anteriores parámetros para ambos ejes.

Como se observa en la Figura A-17, también deben definirse las relaciones mecánicas para los componentes de las suspensiones para ambos ejes. Estos valores se incluyen en la Tabla 12.

Por otra parte, la distancia longitudinal entre ambos ejes se puede observar en la Figura A-4, y tiene un valor final de 3550 mm.

6) Neumáticos

Neumáticos: La Figura A-20 muestra una instantánea del módulo del neumático en TruckSim.

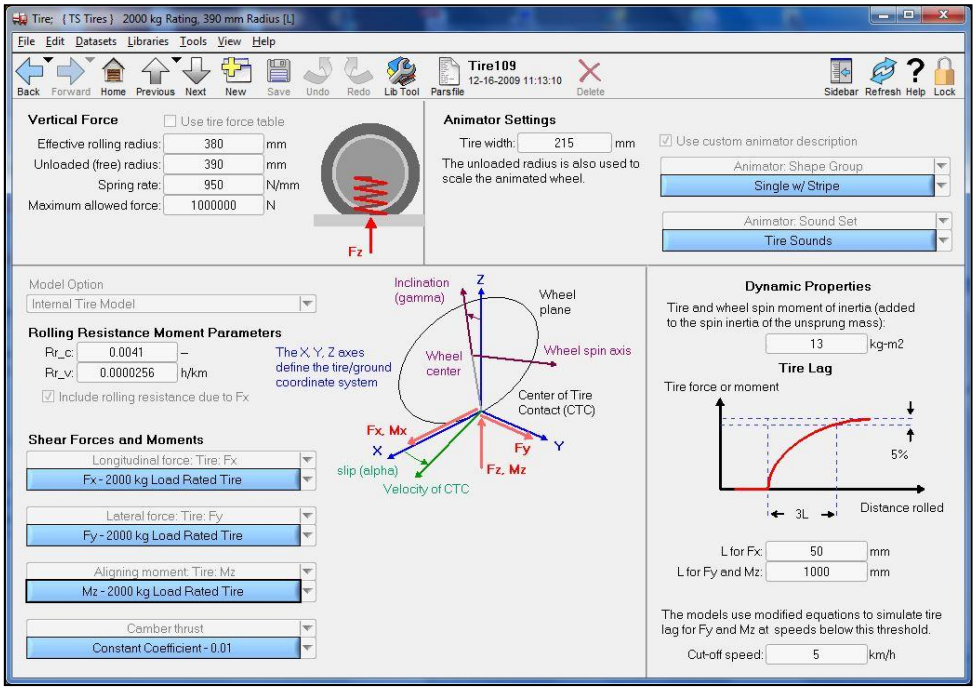


Figura A-20. Módulo del neumático

Las características principales que definen las fuerzas del neumático son las siguientes:

- Fuerza longitudinal del neumático vs relación de deslizamiento (Figura A-21)
- Fuerza lateral del neumático vs ángulo de deslizamiento (Figura A-22)
- Momento de alineación del neumático vs ángulo de deslizamiento (Figura A-23)

Los demás parámetros para los neumáticos se presentan en la siguiente tabla:

Tabla 24. Valores de los parámetros de los neumáticos

Parámetro	Valor	Unidad
Radio de curvatura efectiva	380	mm
Radio descargado	380	mm
Tasa de elasticidad	950	N/mm
Anchura del neumático	215	mm
Longitud de relajación: L - Fx	50	mm
Longitud de relajación: L – Fy & Mz	1000	mm
Momento de giro de los neumáticos	13	Kg-m^2
Máxima fuerza permitida	1000	kN

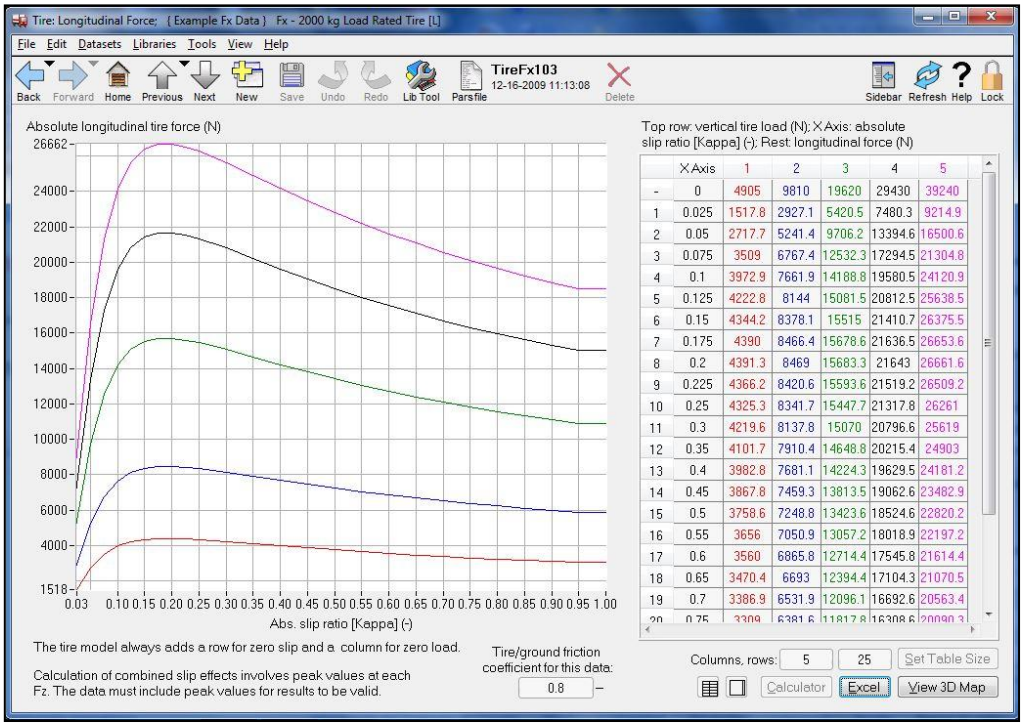


Figura A-21. Fuerza longitudinal del neumático vs relación de deslizamiento

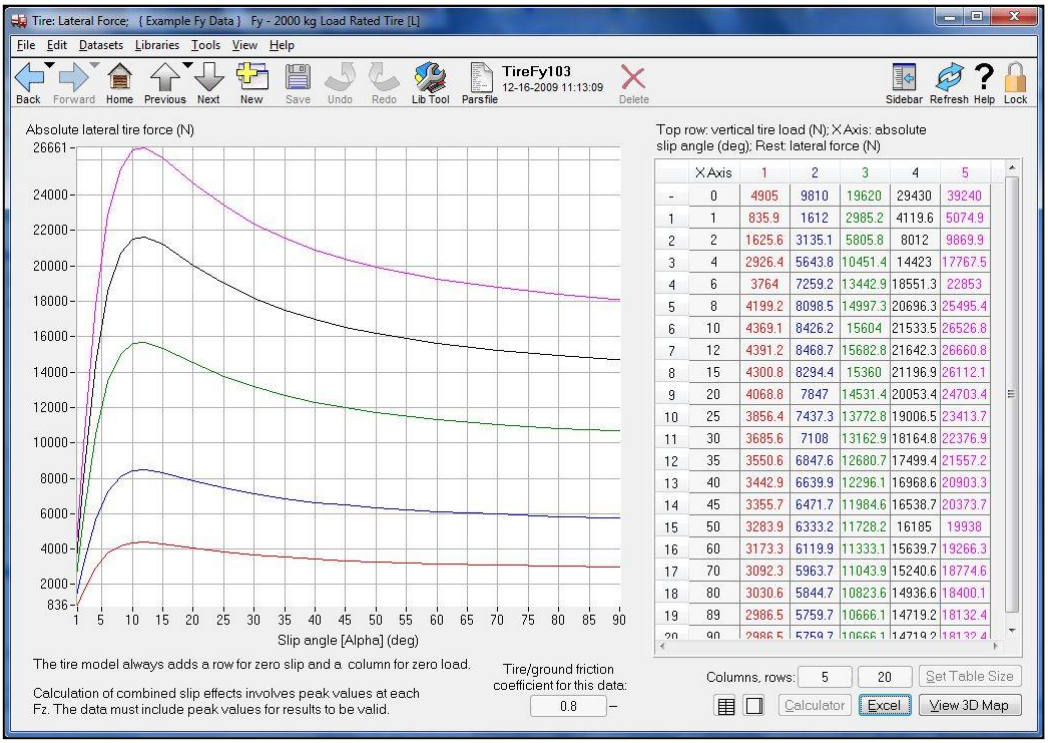


Figura A-22. Fuerza lateral del neumático vs relación de deslizamiento

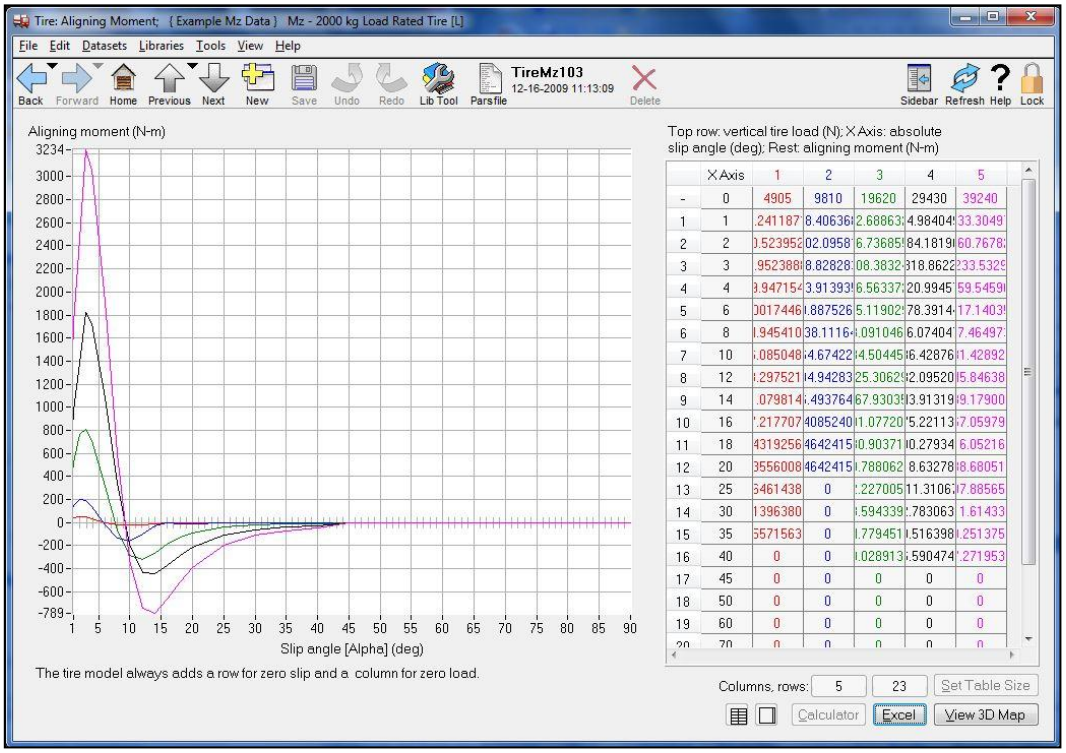


Figura A-23. Momento de alineación de los neumáticos vs ángulo de deslizamiento

Por último se define como se modela el siguiente parámetro en TruckSim:

- Coeficiente de fricción del suelo/neumáticos: Este es el coeficiente de fricción que se midió para el neumático cuando éste se puso a prueba para generar los datos. Este valor se especifica en las características de la fuerza longitudinal y lateral del neumático. En la parte inferior de las Figuras A-21 y A-22 se observa este parámetro.

TruckSim utiliza este valor como coeficiente de referencia para escalar la fuerza de los neumáticos representados por las tablas de búsqueda como se muestra anteriormente. El escalado se realiza en base al coeficiente de fricción de la carretera. Este coeficiente se introduce en la pantalla de ¨procedimientos¨ (Figura A-24) del menú principal del modelo.

$$Factor\ de\ escala = \frac{Coeficiente\ de\ fricción\ de\ la\ carretera}{Coeficiente\ de\ fricción\ de\ referencia}$$

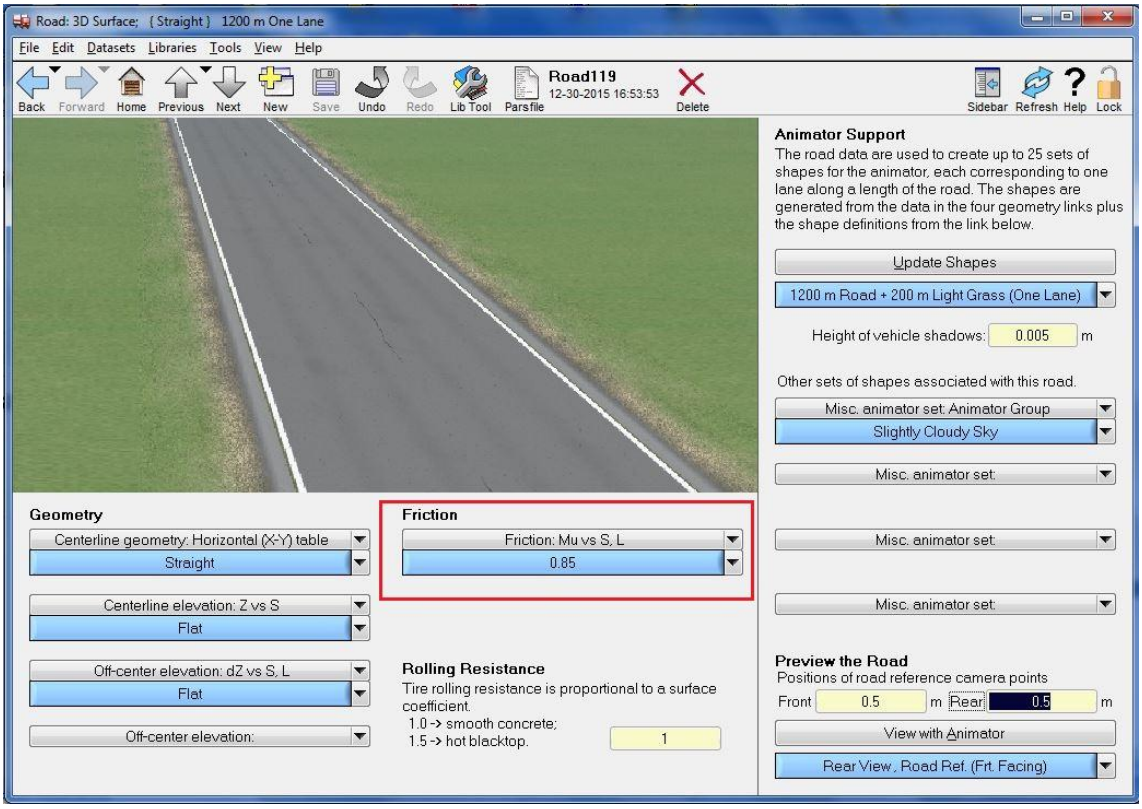


Figura A-24. Coeficiente de fricción de la carretera

- Introducción de los parámetros de la simulación

En este apartado del anexo se trata con el ajuste de la velocidad del vehículo, el control de frenado, el control de la dirección, el control del cambio de marcha, las características de la carretera y la configuración de la animación para la simulación.

La Figura A-25 muestra las opciones disponibles para establecer el control de la velocidad. En este módulo de TruckSim el término ¨station¨ hace referencia a la distancia recorrida por el vehículo en la respectiva carretera seleccionada. Para una carretera recta, esta distancia se refiere a la distancia longitudinal y para una carretera circular, se refiere a la distancia circunferencial. Tambien se puede apreciar en dicha figura, bajo las condiciones de arranque y parada (¨Start and Stop conditions¨), que el usuario puede especificar el momento predeterminado de arranque y parada del vehículo.

Si se selecciona una carretera circular, seleccionando la opción ¨Road forward¨ el vehículo traza una trayectoria en contra de las agujas del reloj, mientras que si se selecciona la opción ¨Road reverse¨ la trayectoria será en el sentido de las agujas del reloj.

Las Figuras A-26 y A-27 muestran las dos diferentes opciones que TruckSim ofrece para especificar el control del frenado. En una situación de frenado real, el conductor deja de acelerar, desembraga y pisa el pedal del freno. Esto puede ser simulado usando la segunda opción como se muestra en la Figura A-28.

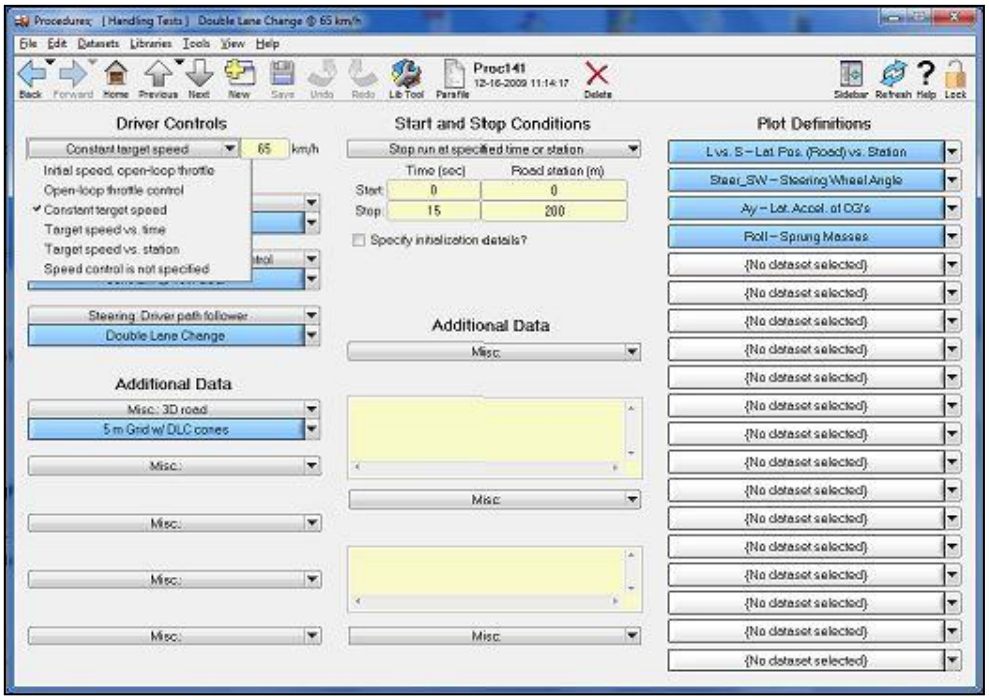


Figura A-25. Opciones del control de velocidad

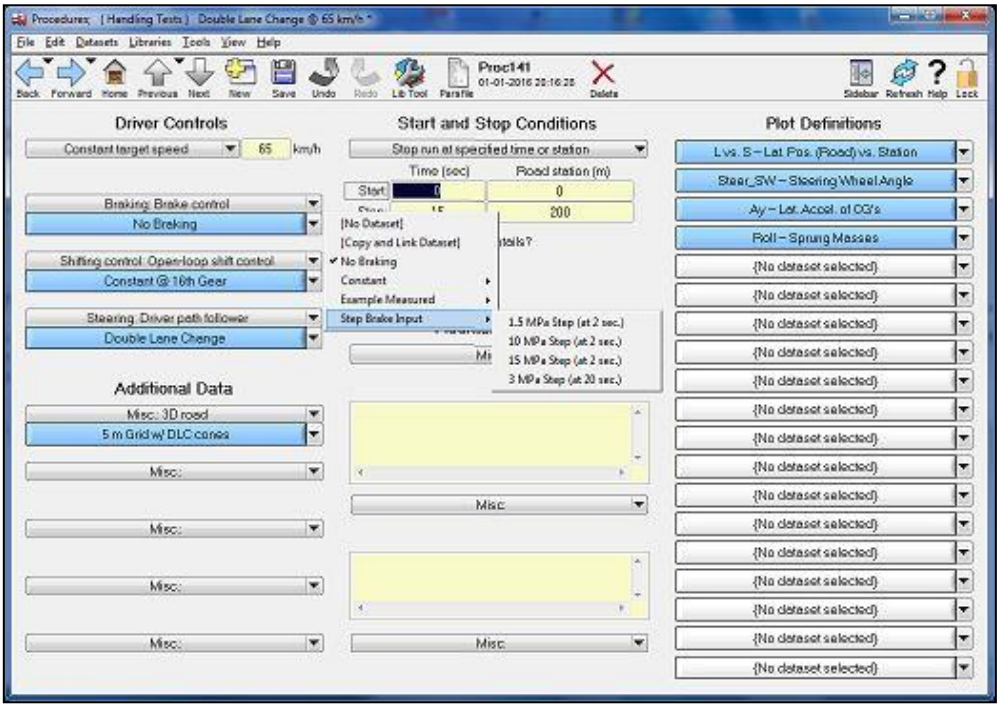


Figura A-26. Control del freno, opción 1

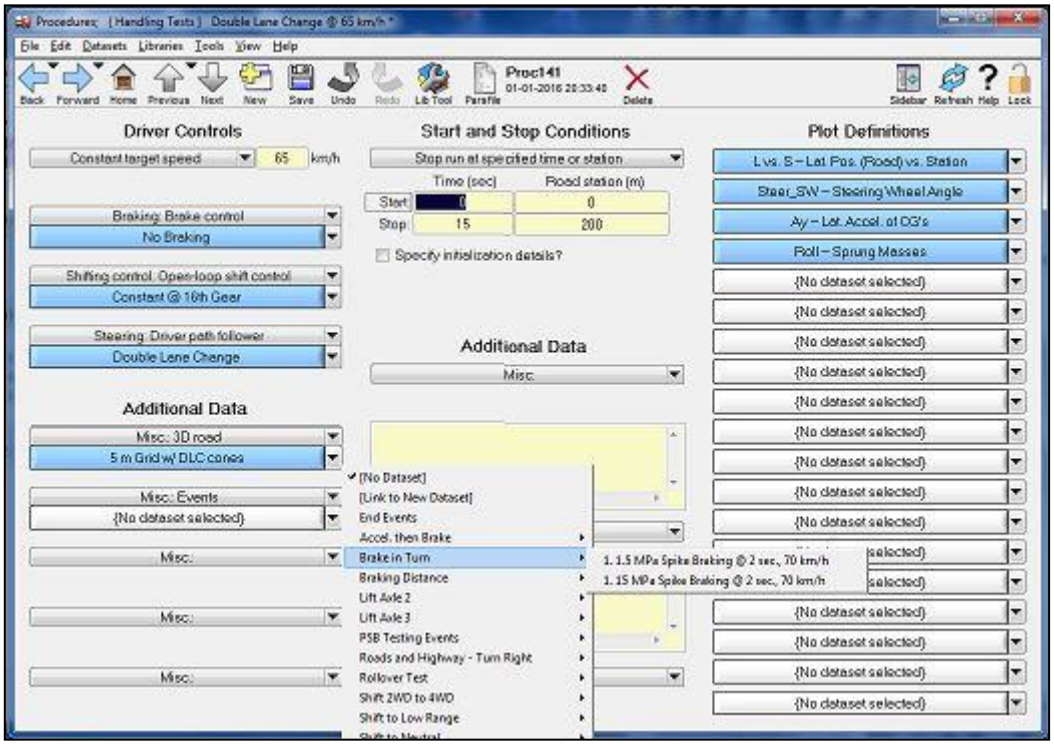


Figura A-27. Control del freno, opción 2

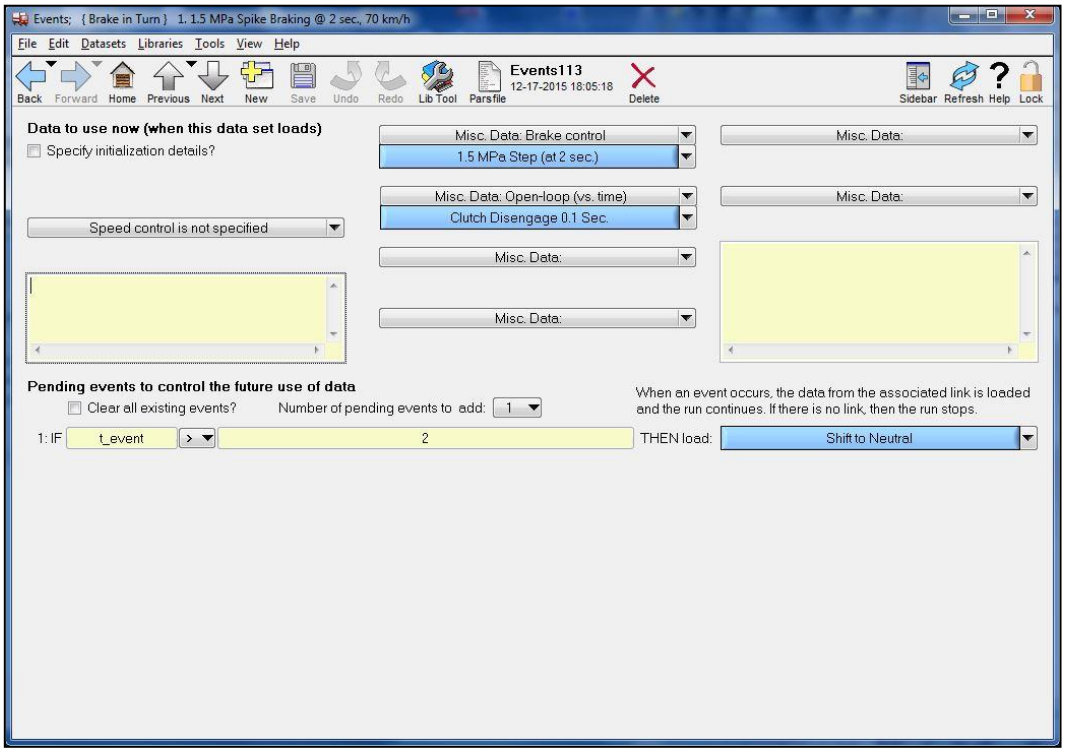


Figura A-28. Conjunto de datos del freno de la opción 2

En esta pantalla, básicamente se tienen que introducir el número de eventos que ocurren durante el procedimiento, el momento en el cual el evento ocurre y la acción que se debe realizar durante ese instante de tiempo. La variable ¨t_event¨ indica el tiempo de simulación actual. En esta figura, se lleva a cabo una fuerte frenada sobre una superficie durante 2 s. En ese momento el embrague se desengancha, se establece el punto muerto y se realiza la frenada.

Para el control del cambio de marcha TruckSim proporciona dos opciones para configurar la palanca de cambios, una llamada de bucle abierto y otra de bucle cerrado. Estas dos opciones se muestran en la Figura A-29.

El control de bucle cerrado es similar a una caja de transmisión automática en la cual el programa decide que engranajes seleccionar, dependiendo de la velocidad y aceleración requerida desde el control en TruckSim.

Los ajustes de animación permiten al usuario especificar las dimensiones virtuales del vehículo, las características de la carretera y el ambiente para la simulación. La Figura A-30 muestra donde se configuran estas dimensiones y la apariencia del vehículo para la simulación. Estas dimensiones no tienen nada que ver con las dimensiones reales del vehículo y por tanto no afectan de ninguna manera a la dinámica del vehículo.

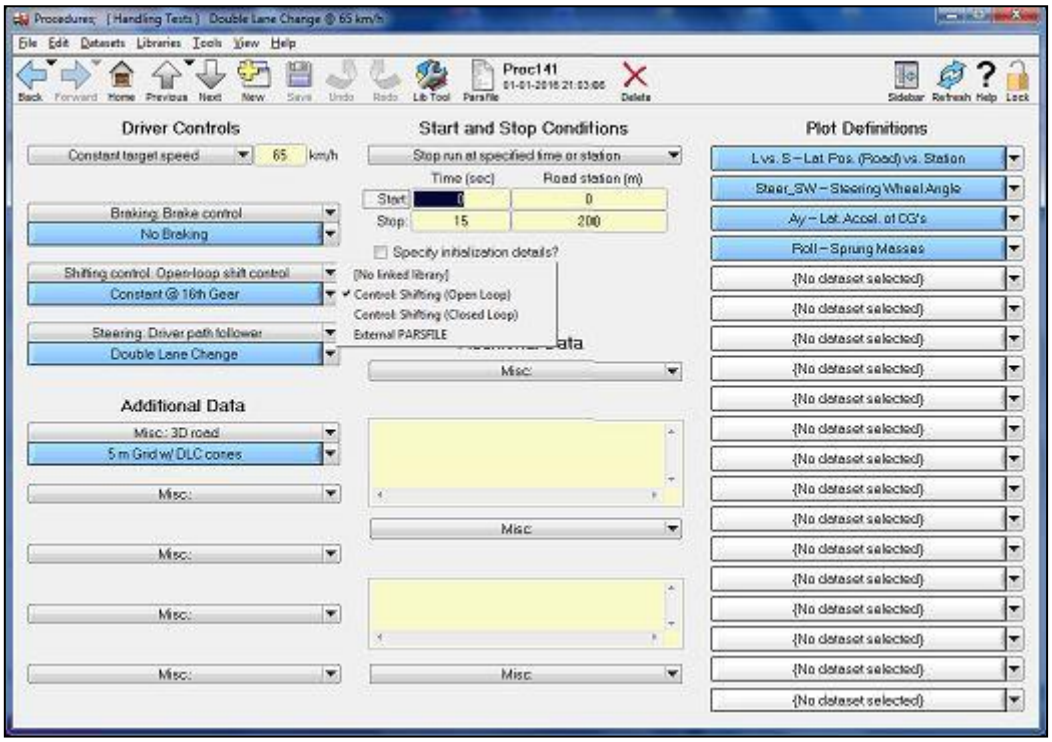


Figura A-29. Control del cambio - Bucle abierto o cerrado

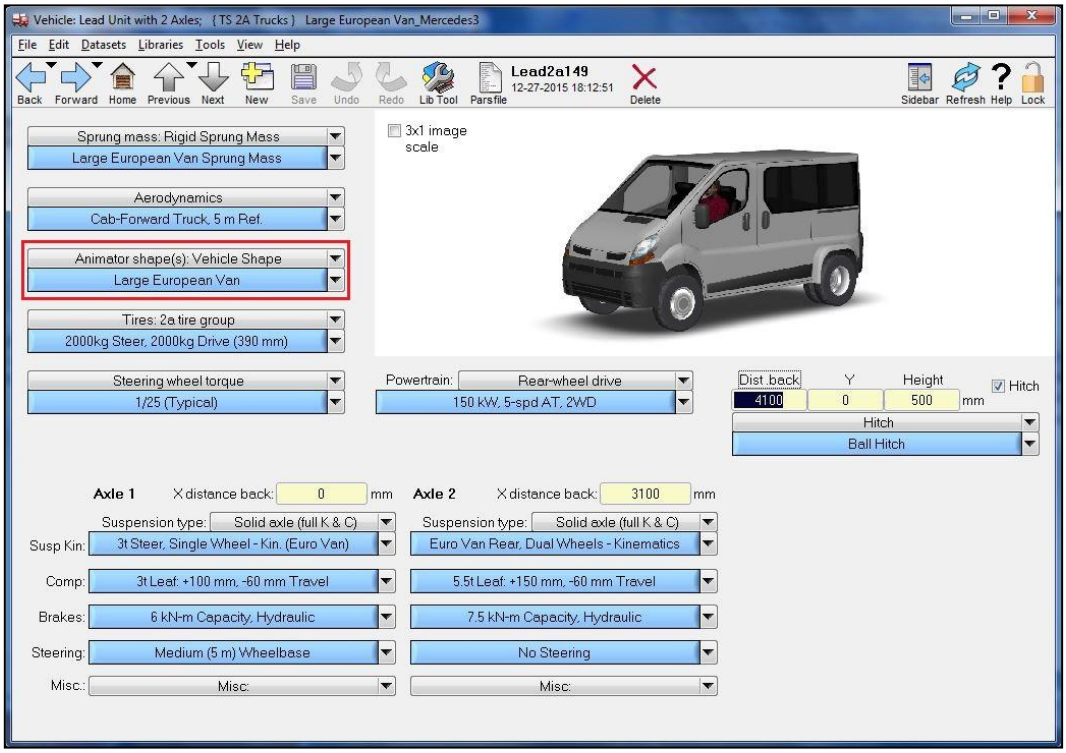


Figura A-30. Controles de animación del vehículo

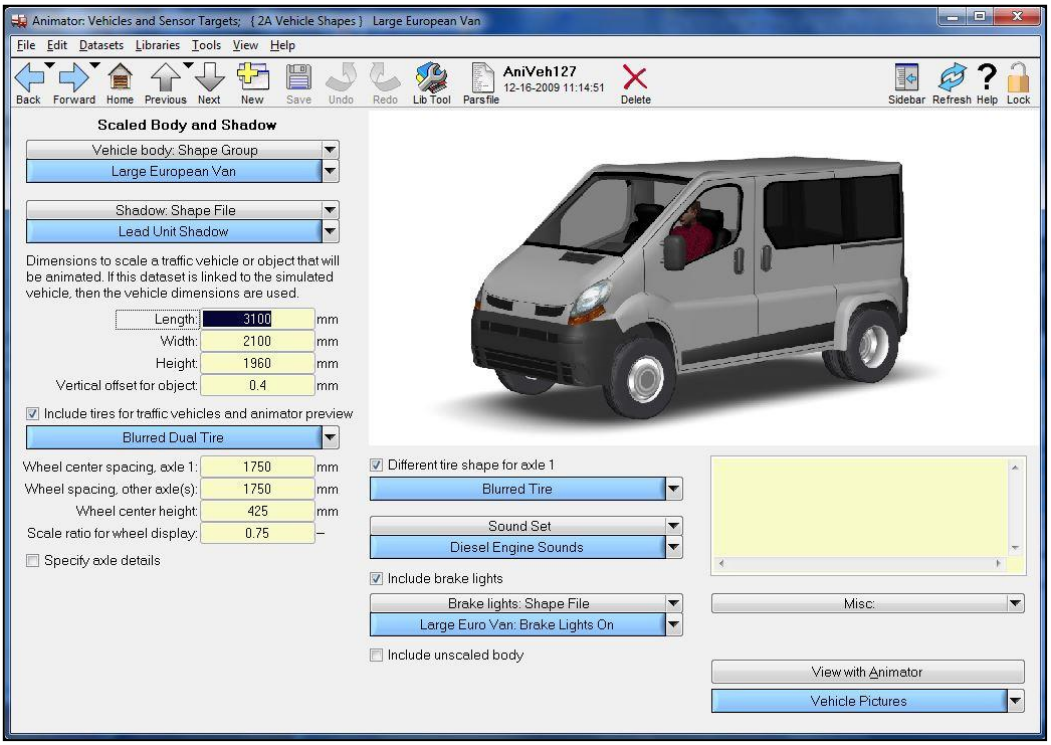


Figura A-31. Parámetros del control de animación

Para todos estos componentes, se tiene que especificar un archivo que esté contenido en uno de los directorios raíz de TruckSim y la ruta de acceso se elige en el lugar indicado en la Figura A-32.

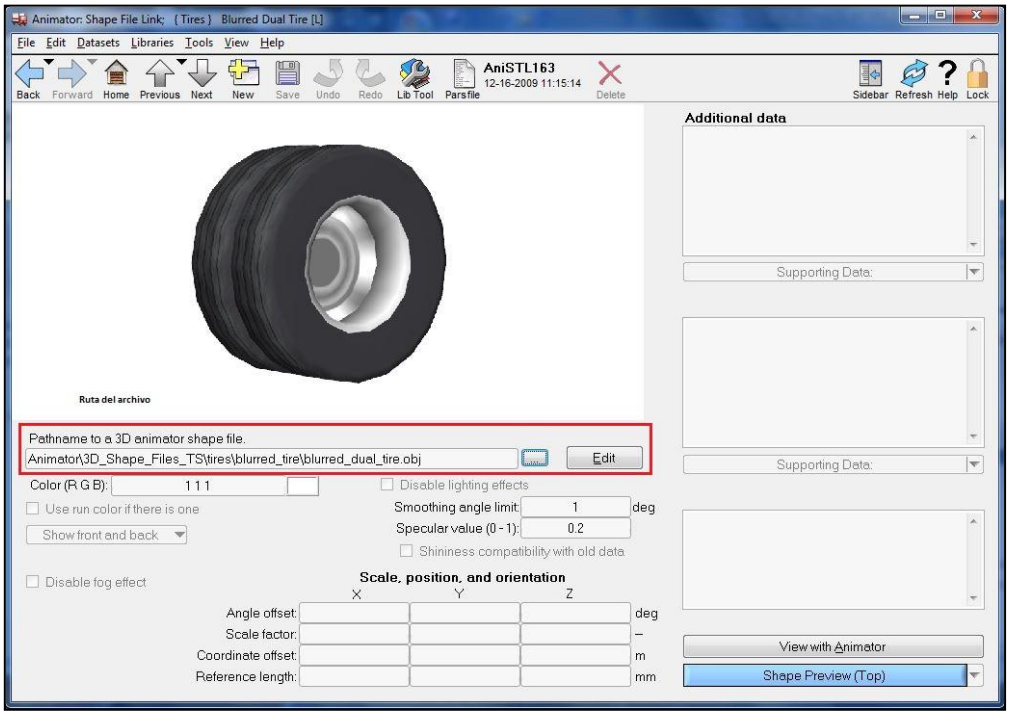


Figura A-32. Animación – Especificar ruta del archivo

Las Figuras A-33 y A-34 muestran los controles de animación que existen para la carretera. La Figura A-34 muestra la forma de modelar las características de los materiales de la carretera. Los términos ¨L Start¨ y ¨L Stop¨ se usan para especificar las coordenadas laterales de la carretera. Los términos ¨S Start¨ y ¨S Stop¨ especifican la distancia longitudinal de la carretera. Como se aprecia en la figura la anchura real de la carretera es de 20 metros (10 m /-10m) y la longitud total es de 500 m. A ambos lados de la carretera se tendrá césped con una anchura de 90 m (esto se puede apreciar visualmente en la Figura A-34). Los números que aparecen a la izquierda del tipo de material, 0.878 0.878 0.878 para el caso del césped, especifican el código de color para esa superficie.

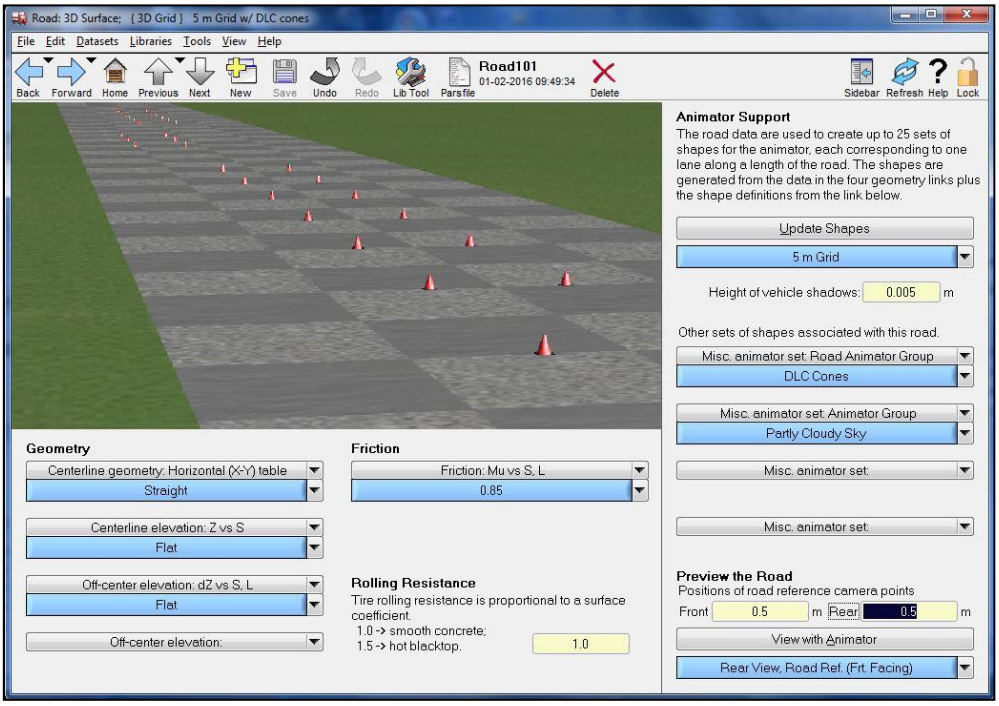


Figura A-33. Controles de animación de la carretera

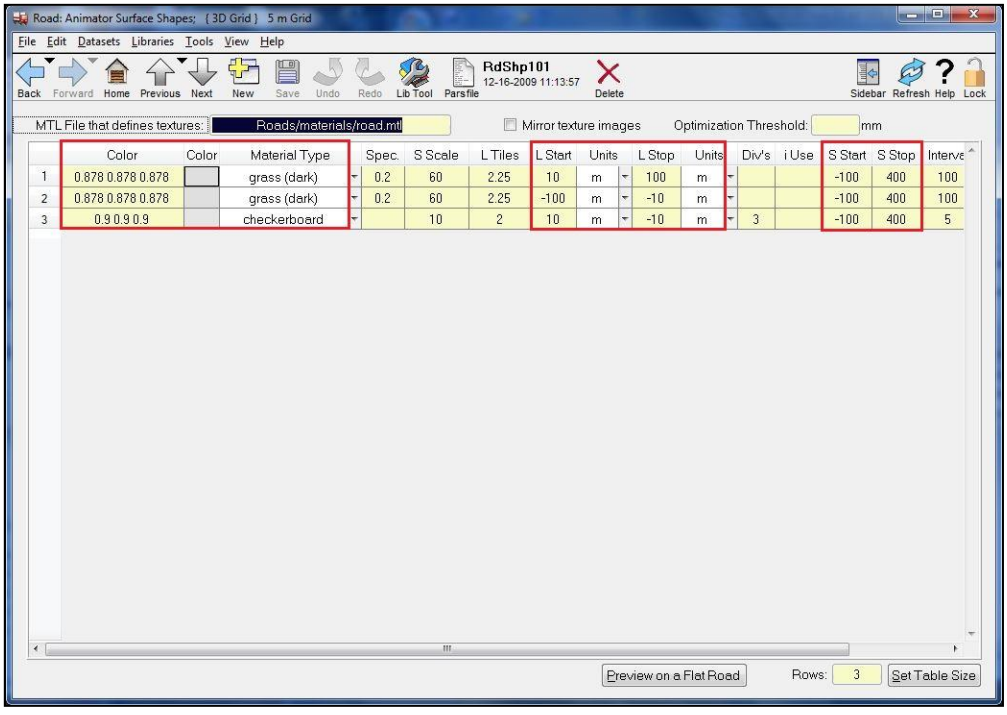


Figura A-34. Código de color, selección de material y ajustes de distancia para la carretera

Proceso de validación del modelo de furgoneta

En este apartado se explica el procedimiento a seguir para validar el anterior modelo preparado en TruckSim. Este proceso se realiza usando las siguientes maniobras dinámicas: doble cambio de carril y simple cambio de carril. Los resultados de la simulación serán comparados con las curvas obtenidas experimentalmente.

Estas maniobras dinámicas se usan para validar varias variables dinámicas del vehículo tales como la aceleración lateral, la velocidad de guiñada y el ángulo de balanceo. Analizando estas variables del vehículo se verifican la dinámica lateral, longitudinal, de balanceo y de guiñada del vehículo.

A continuación se presenta una breve descripción de estas maniobras.

Maniobra de doble cambio de carril

Esta es una de las maniobras que más comúnmente realizan los vehículos en condiciones reales cuando se mueven por las autopistas. En esta maniobra, el vehículo se desplaza de un carril a otro y luego regresa de nuevo al mismo carril. Este cambio de carril se hace rápidamente. Esta maniobra se usa ampliamente para evaluar el módulo de control de estabilidad de guiñada en el que el controlador impide que el vehículo tenga sobreviraje o subviraje, lo que normalmente ocurre en superficies con bajo coeficiente de fricción. Se realiza a velocidad constante y con la condición de tener el acelerador soltado. Es decir, cuando el vehículo alcanza la velocidad objetivo, el conductor suelta el acelerador y el vehículo va en punto muerto el resto del tiempo.

La Figura A-35 muestra el perfil del ángulo de dirección que se obtiene cuando se lleva a cabo esta maniobra y la Figura A-36 muestra la trayectoria del vehículo.

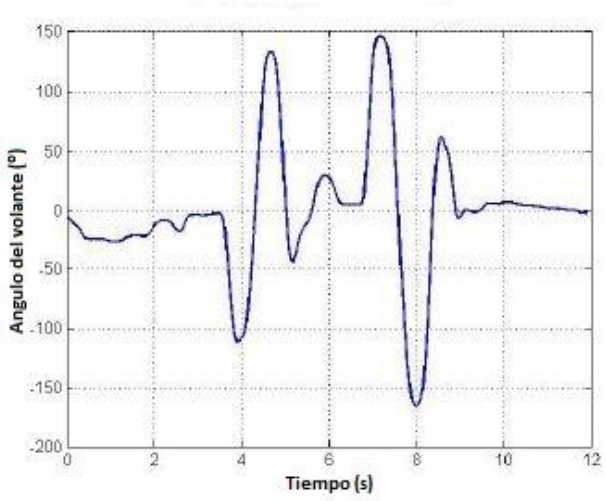


Figura A-35. Doble cambio de carril - Perfil de dirección

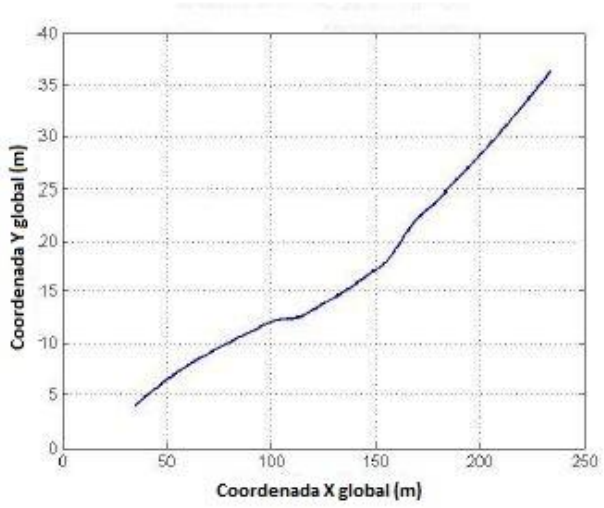


Figura A-36. Doble cambio de carril - Trayectoria del vehículo

La trayectoria del vehículo muestra una desviación muy pequeña respecto del camino en línea recta. Esto es debido a que la magnitud dada para la entrada de dirección del vehículo es pequeña, lo que resulta en una magnitud pequeña para el ángulo de la rueda.

Maniobra de simple cambio de carril

Esta maniobra es la simplificación de la de doble cambio de carril. El vehículo cambia únicamente una vez de carril. Al igual que la de doble cambio de carril, se realiza a velocidad constante y con el acelerador soltado.

- Proceso de validación

En este apartado se explican los pasos a seguir para validar el modelo de furgoneta realizado en TruckSim a partir de las curvas experimentales disponibles.

Para que TruckSim interactúe con el entorno de Simulink se elige el modelo propio de Simulink tal y como se muestra en la Figura A-37.

Una vez que se crea el modelo se debe especificar una ruta de acceso para este modelo tal y como se muestra en la figura A-38. Una vez realizado este, se definen las variables de salida que se desee que se muestren (en la Figura A-39 se selecciona el ángulo de balanceo a modo de ejemplo). De igual manera, en la Figura A-40 se definen las entradas al modelo. En esta simulación las entradas que se introducen al modelo son el ángulo de dirección y la velocidad longitudinal, teniendo esta última entrada un valor inicial de 60 Km/h, tal y como se observa en la figura.

Tras estos pasos se vuelve a la pantalla principal del modelo y se pulsa el botón ¨Send to Simulink¨ para acceder al entorno del mismo. La Figura A-41 muestra este entorno modificado

para representar en tres gráficas distintas el ángulo de balanceo del modelo de TruckSim, el de la furgoneta real y una gráfica conjunta de ambas.

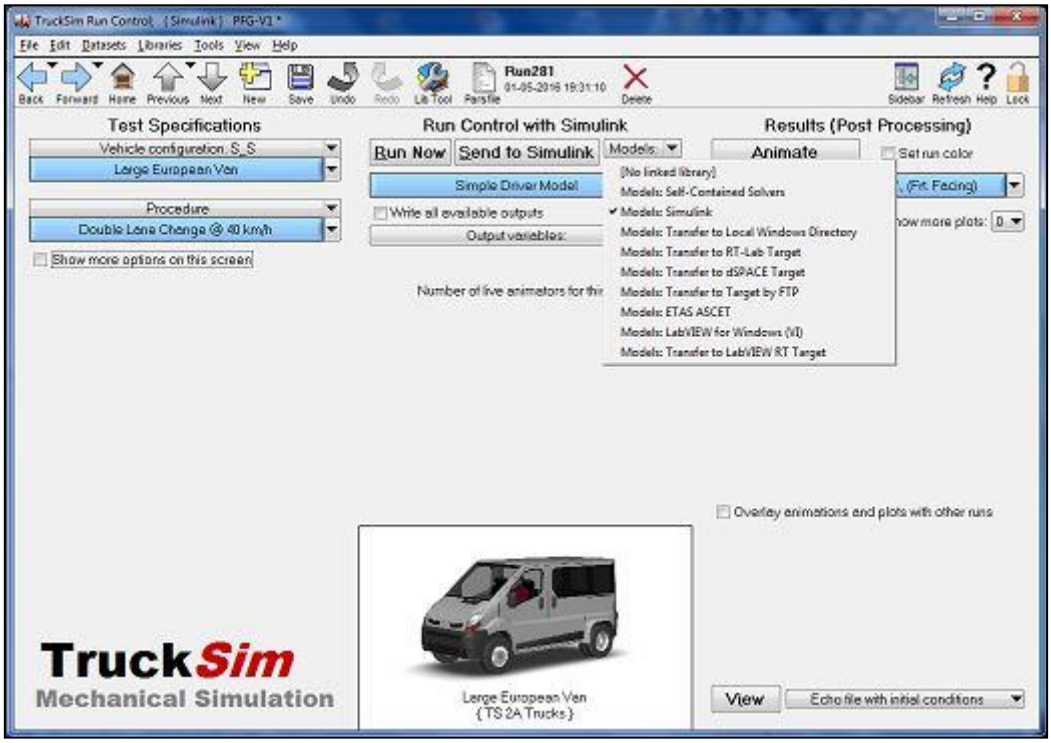


Figura A-37. Opción de Simulink

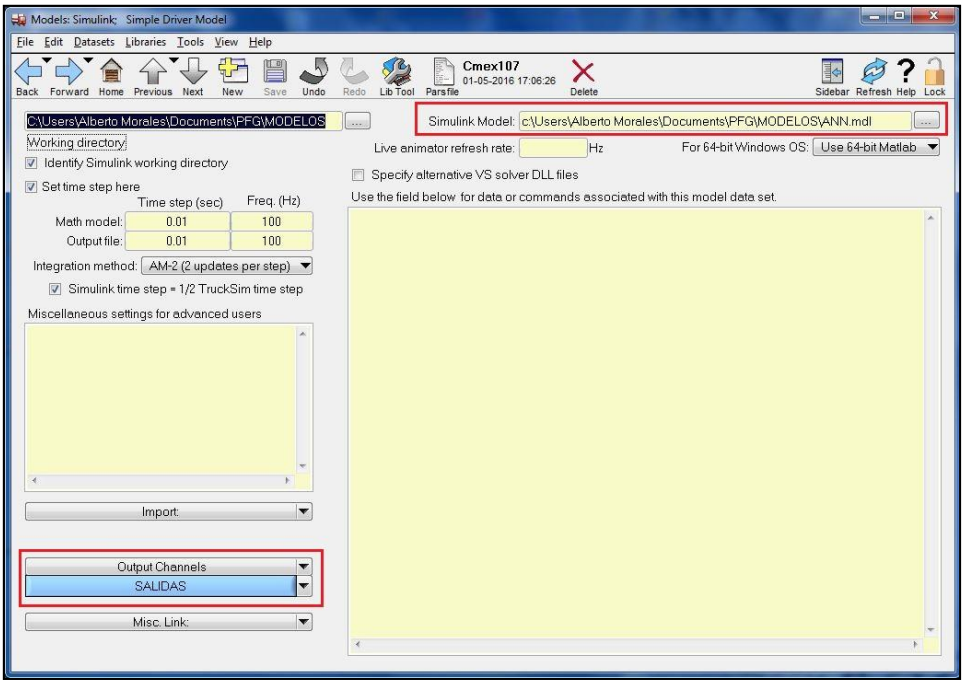


Figura A-38. Ruta de acceso del modelo de Simulink / Definición de salidas

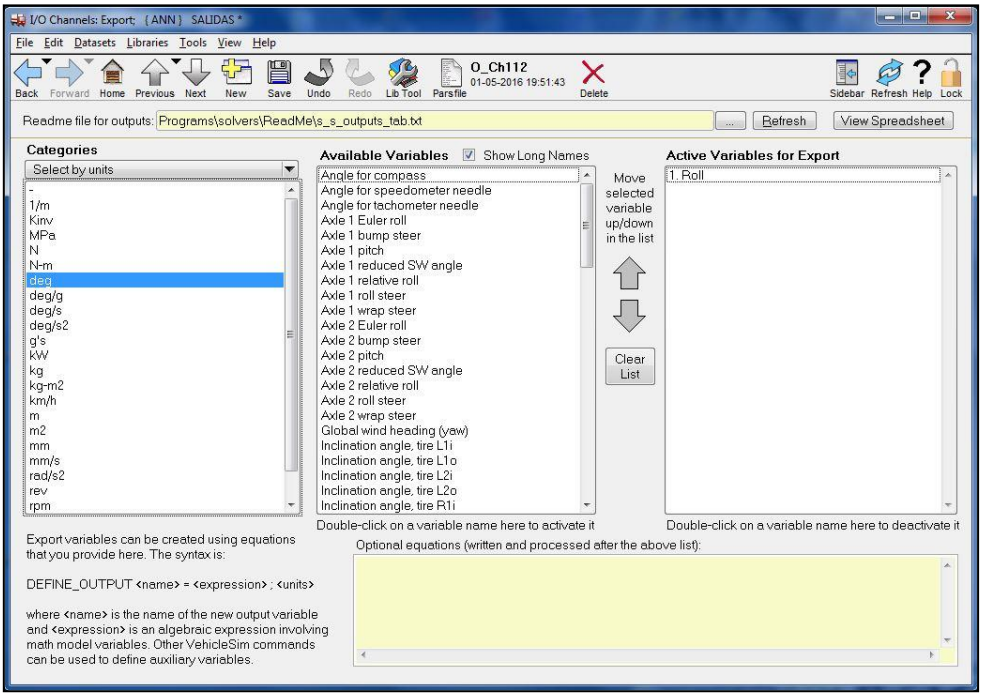


Figura A-39. Variables de salida

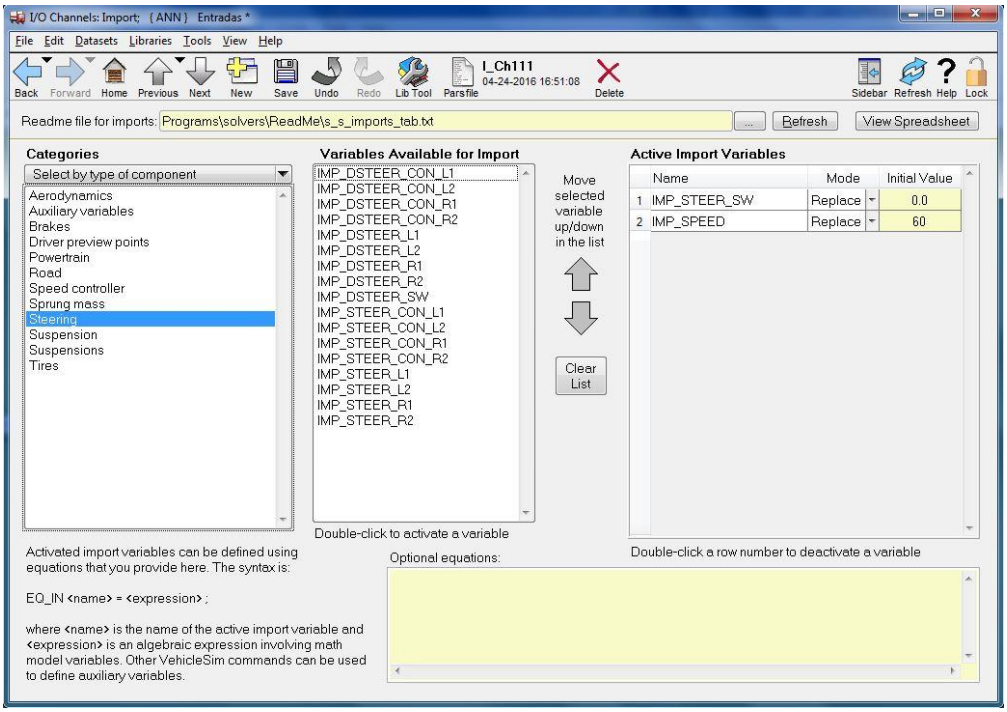


Figura A-40. Variables de entrada

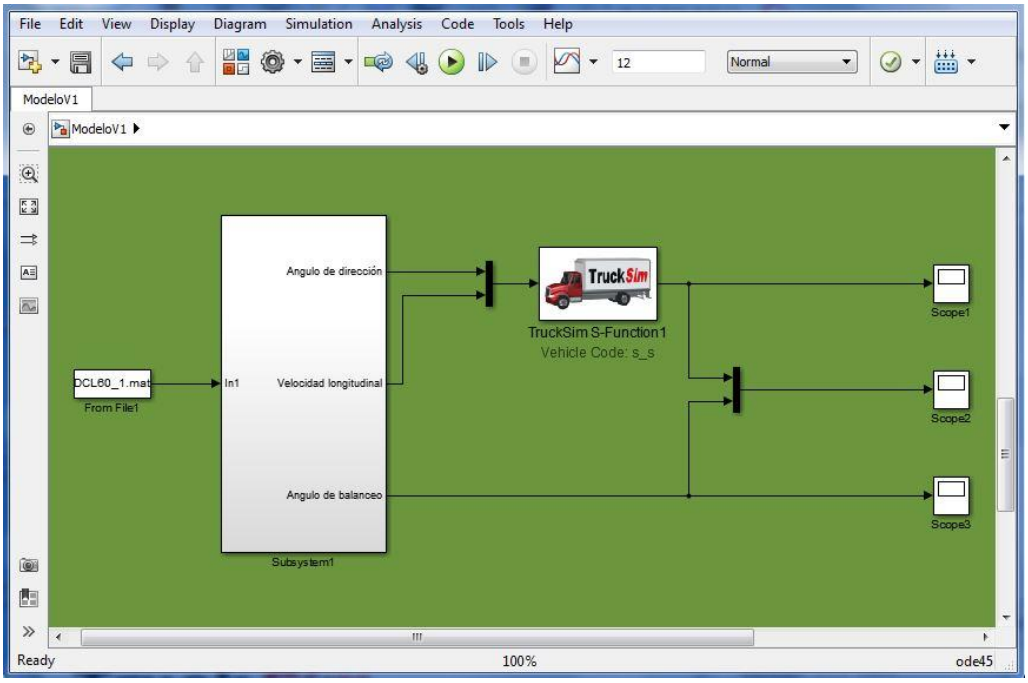


Figura A-41. Entorno de Simulink

Para introducir los datos disponibles para cada maniobra experimental, los cuales deben estar guardados en la misma ruta de acceso que se definió anteriormente para el modelo de Simulink, en el entorno se accede a la biblioteca y se introduce el bloque denominado ¨From File¨. Una vez introducido el bloque pulsando doble click sobre él se define el nombre de la maniobra a utilizar (DCL60_1 en el ejemplo). Por último pulsando el botón ¨Run¨ de la Figura A-41 se inicia el proceso de simulación cuyos resultados se grafican conjuntamente con las experimentales para realizar el proceso de validación.

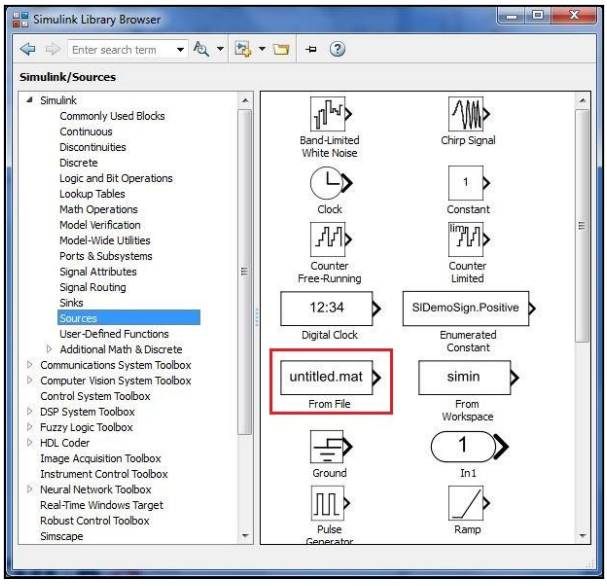


Figura A-42. Biblioteca de Simulink

Una vez que se representan las curvas para cada variable, para calcular el error correspondiente, escribiendo en la ventana de comandos de Matlab la palabra ¨workspace¨ aparece una ventana (Figura A-43) en la que se pueden obtener los valores de la variable representada tanto para el vehículo real como para el modelo de TruckSim (Figura A-44).

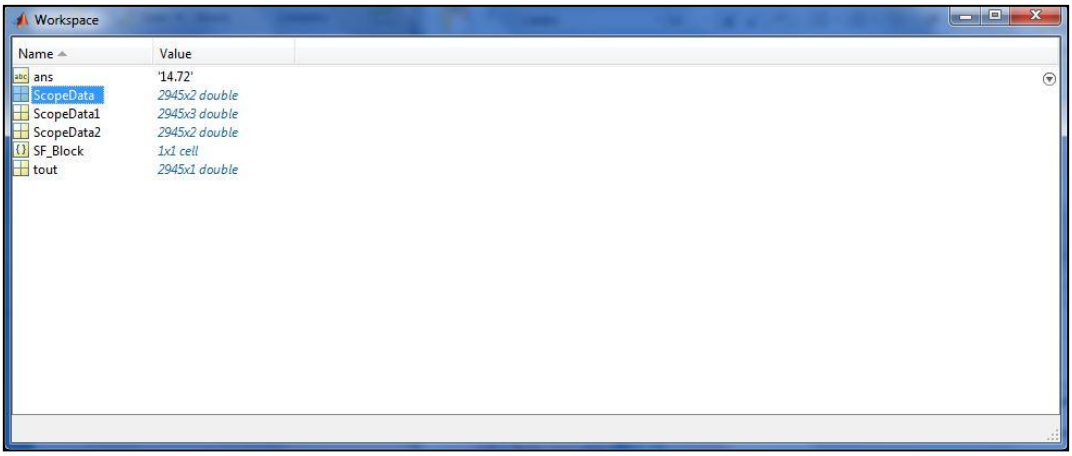
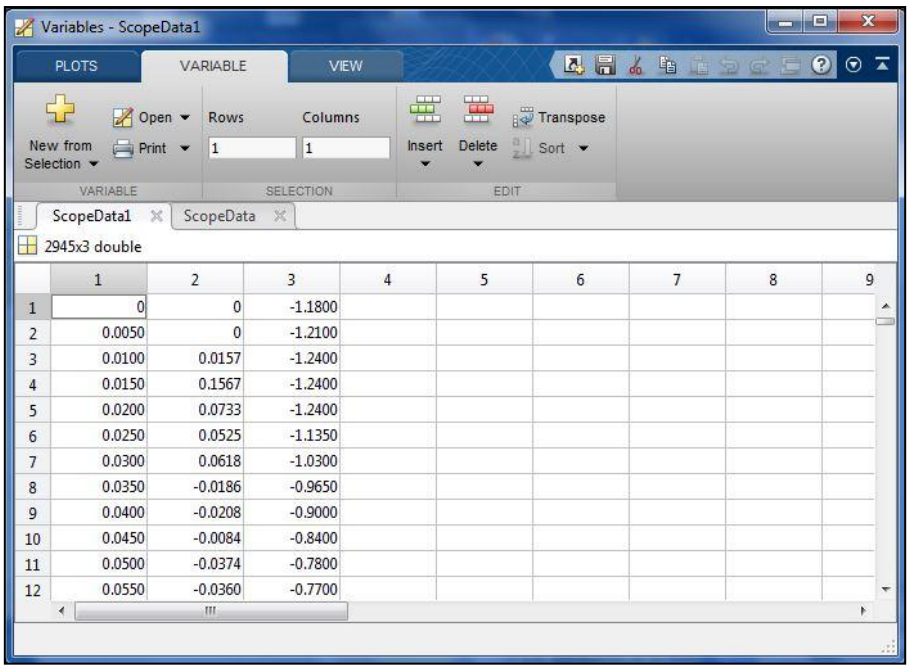


Figura A-43. Workspace de Simulink



	1	2	3	4	5	6	7	8	9
1	0	0	-1.1800						
2	0.0050	0	-1.2100						
3	0.0100	0.0157	-1.2400						
4	0.0150	0.1567	-1.2400						
5	0.0200	0.0733	-1.2400						
6	0.0250	0.0525	-1.1350						
7	0.0300	0.0618	-1.0300						
8	0.0350	-0.0186	-0.9650						
9	0.0400	-0.0208	-0.9000						
10	0.0450	-0.0084	-0.8400						
11	0.0500	-0.0374	-0.7800						
12	0.0550	-0.0360	-0.7700						

Figura A-44. Valores de las variables

Con estos valores es con los que se realiza la validación del modelo, tal y como se ha explicado en el Capitulo 5.

Anexo B

Codificación de la red neuronal en Matlab

Tal y como se explica en el Capítulo 2 existen diversas posibilidades a la hora de configurar parámetros como pueden ser la función de activación, el algoritmo de entrenamiento utilizado, etc.

En este Anexo se explican los comandos utilizados para la creación de la red neuronal y el motivo de la elección de cada uno de los parámetros.

1) Creación de la red neuronal

El comando para crear la red neuronal es el siguiente:

net = network(A,B,C,D,E,F)

Las letras situadas entre los paréntesis se corresponden con los siguientes parámetros:

A: Número de entradas.

B: Número de capas. Este siempre será 2 (una capa oculta y otra capa de salida).

C: Conexiones de sesgo con capa

D: Conexiones de las entradas con cada capa

E: Conexiones entre las capas

F: Conexiones de las capas con la salida

net = network(5, 2, [1 ; 1],[1 1 1 1 1; 0 0 0 0 0], [0 0; 1 0], [0 1]);

En este ejemplo propuesto, que se ilustra en la Figura B-1, se decide que haya 5 entradas y 2 capas. El vector [1; 1] establece que tanto la capa oculta como la de salida posean conexiones de sesgo (cuadrado azul con la letra b en la Figura B-1). La matriz [1 1 1 1 1; 0 0 0 0 0] establece que las cinco entradas posean conexiones con la capa oculta pero no con la de salida. La matriz [0 0; 1 0] establece que exista la salida de la capa oculta se multiplique por un peso para entrar a la función de activación de la capa oculta. El vector [0 1] establece que la salida tan solo se conecte con la capa de salida.

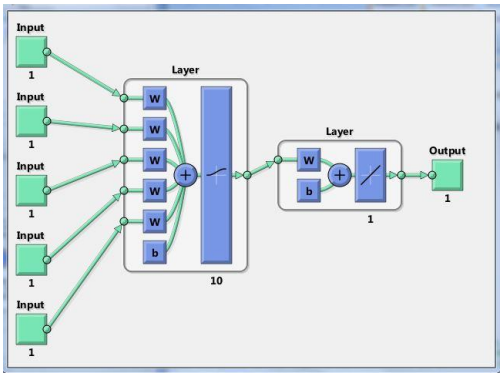


Figura B-1. Red neuronal configurada

2) Función de activación

El comando para definir esta función es el siguiente:

net.layers{*NUMERO DE LA CAPA*}.transferFcn = '*NOMBRE DE LA FUNCION*';

En el apartado 2.5.2 se definen las principales funciones de activación que se utilizan en las redes neuronales. Los nombres en Matlab de estas funciones son ¨*purelin*¨, ¨*logsig*¨, ¨*tansig*¨; estos nombres corresponden a la función lineal, a la función sigmoidea y a la función tangente hiperbólica respectivamente.

Tras realizar una serie de entrenamientos probando todas las combinaciones posibles de estas funciones en las dos capas, se determina que el menor error se consigue usando una función ¨*logsig*¨ en la capa oculta y una función ¨*purelin*¨ en la capa de salida.

3) Configuración de la red neuronal

El comando para configurar la red es el siguiente:

net = configure(net, *NOMBRE DE LAS ENTRADAS*,*NOMBRE DEL OBJETIVO*);

Este proceso establece los tamaños de las entradas y las salidas. Con este comando también se inician los pesos de la red. En el código las entradas reciben el nombre de ¨*inputs*¨ y el objetivo el de ¨*outputs*¨.

4) Parámetros del entrenamiento de la red

En este punto se distinguen varios comandos.

- Función de entrenamiento: la sintaxis para esta función es la siguiente:

  net.trainFcn = '*NOMBRE DE LA FUNCION DE ENTRENAMIENTO';*

  Tal y como se menciona en el apartado 2.1, existen distintas funciones de entrenamiento para la red neuronal. Así, ¨*trainlm*¨ es una función que actualiza los valores de los pesos de acuerdo con la optimización de Levenberg-Marquardt; ¨*trainbr*¨ utiliza un proceso de regularización llamado Bayesian; ¨*trainscg*¨ es una función que actualiza los valores de los pesos de acuerdo con el método del gradiente conjugado escalado (scg: scaled conjúgate gradient); por último ¨*trainrp*¨ actualiza los valores de los pesos de acuerdo con el algoritmo de propagación hacia atrás elástico (Rprop).

  Tras realizar una serie de entrenamientos con cada una de las funciones, se determina, tal y como se esperaba, que la función que mejor resultados ofrece es la ¨*trainlm*¨.

- Función de rendimiento:

  net.performFcn = '*NOMBRE DE LA FUNCION DE RENDIMIENTO';*

  Se utiliza la función de rendimiento MSE. Esta función mide el rendimiento de acuerdo con el error cuadrático medio.

- Parámetros de la función de entrenamiento

  De los distintos parámetros que contiene la función de entrenamiento seleccionada se configuran el número máximo de iteraciones y el número máximo de fallos en la validación. La sintaxis para configurar estos parámetros es:

  net.trainParam.epochs = *NUMERO DE ITERACIONES*;
  net.trainParam.max_fail = *NUMERO DE FALLOS*;

- División de los datos para un óptimo entrenamiento de la red neuronal

  La práctica general es la división de los datos en conjuntos de datos de entrenamiento, test y validación. Existen cuatro funciones diferentes para la división de los datos (¨*dividerand*¨, ¨*divideblock*¨, ¨*divideint*¨, ¨*divideind*¨). La sintaxis para definir la función de división es:

  net.divideFcn = *'NOMBRE DE LA FUNCION DE DIVISION';*

  En el código propuesto se utiliza la función ¨*dividerand*¨, la cual asigna un 70% de los datos al conjunto de entrenamiento, y un 15% de los datos a los otros dos conjuntos.

5) Entrenamiento de la red neuronal

La sintaxis del comando que realiza el entrenamiento de la red es la siguiente:

[net, tr] = train (net, *NOMBRE DE LAS ENTRADAS*, *NOMBRE DEL OBJETIVO*);

El entrenamiento de la red se realiza en función a los parámetros anteriormente descritos.

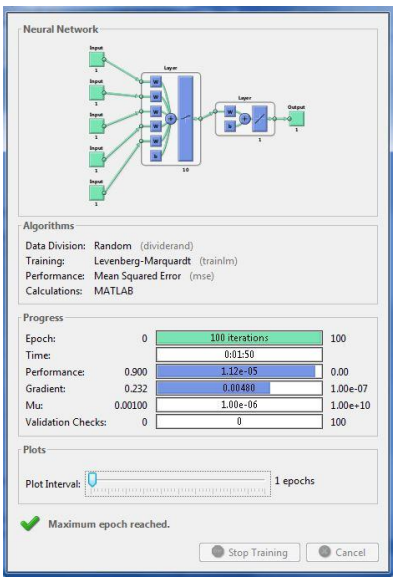


Figura B-2. Entrenamiento de la red neuronal completado

6) Obtención de resultados

➢ Valores calculados por la red neuronal

Para obtener los valores que estima la red para la variable en cuestión se utiliza el siguiente comando:

NOMBRE DE LA SALIDA ESTIMADA = net (inputs)

Con este comando una vez que la red neuronal ha sido entrenada, en el espacio de trabajo de Matlab aparece la variable con el nombre creado, la cual contiene los datos de la salida que proporciona la red.

➢ Representación gráfica de las actuaciones de entrenamiento, validación y test

La sintaxis para obtener esta representación es la siguiente:

plotperform(tr)

Se obtiene la gráfica que se muestra en la Figura B-3, en la cual se puede observar como disminuye el error en la estimación de la variable objetivo a medida que aumenta el número de iteraciones como consecuencia del progresivo ajuste de los pesos de las conexiones. Se aprecia que el error mínimo se consigue en la iteración número 100 y tiene un valor de $1{,}397310^{-5}$.

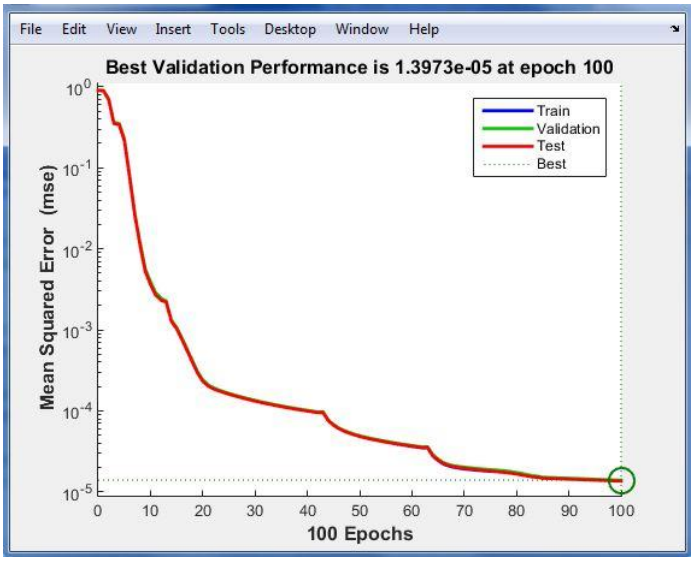


Figura B-3. Representación gráfica de la disminución del MSE

➢ Representación gráfica del ajuste de la salida con el objetivo de la red

La sintaxis para obtener esta representación es la siguiente:

```
plotregression(NOMBRE DEL OBJETIVO,NOMBRE DE LA SALIDA ESTIMADA)
```

La Figura B-4 muestra la gráfica que se obtiene con dicho comando:

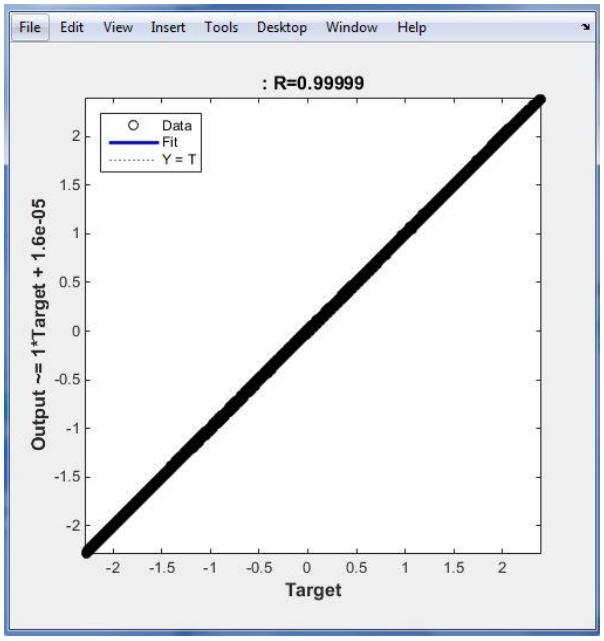


Figura B-4. Representación gráfica del ajuste

- Archivo .net de la red neuronal entrenada

Una vez realizado el entrenamiento de la red neuronal en el espacio de trabajo de Matlab se muestran los resultados, así como las entradas usadas a la red. En la Figura B-5 se muestra esta ventana en la que aparece el archivo .net que se debe guardar para posteriormente poder simular dicha red en el entorno de Simulink.

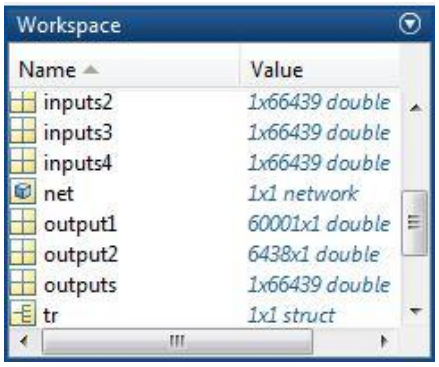


Figura B-5. Espacio de trabajo de Matlab

Las siguientes líneas incluyen el código desarrollado para la red neuronal:

```
%% Códigos para la ejecución de la red neuronal

close all, clear all, clc, format compact

%*************************************************************************
%                              IMPORTAR VARIABLES
%*************************************************************************
% Se solicita el número de entradas que se desea que tenga la red
A = input('Introduzca el número de entradas: ');
% Condicional para el número de entradas
if A==1;
 variable1=input('Introduzca el numero de la primera variable de entrada: ');
 % FuncionVariable1(variable1) llama a la función local FuncionVariable 1
 % la cual devuelve el valor de la variable inputs1
 [inputs1]= FuncionVariable1(variable1);
  inputs = inputs1;
  net = network(1,2,[1;1],[1;0],[0 0;1 0],[0 1]);
 % En el network se define: el número de entradas, el número de capas,
 % las conexiones de sesgo con cada capa, las conexiones de las entradas con
 % cada capa, los pesos de las capas y las conexiones de la salidas con cada
 % capa
elseif A==2;(. . .)
elseif A==3;(. . .)
elseif A==4;(. . .)
elseif A==5;(. . .)
elseif A==6;(. . .)
elseif A==7;(. . .)
elseif A==9;
variable1=input('Introduzca el numero de la primera variable de entrada: ');
 variable2=input('Introduzca el numero de la segunda variable de entrada: ');
 variable3=input('Introduzca el numero de la tercera variable de entrada: ');
 variable4=input('Introduzca el numero de la cuarta variable de entrada: ');

 variable5=input('Introduzca el numero de la quinta variable de entrada: ');
 variable6=input('Introduzca el numero de la sexta variable de entrada: ');
 variable7=input('Introduzca el numero de la séptima variable de entrada: ');
 variable8=input('Introduzca el numero de la octava variable de entrada: ');
 variable9=input('Introduzca el numero de la novena variable de entrada: ');
   [inputs1]= FuncionVariable1(variable1);
   [inputs2]= FuncionVariable2(variable2);
   [inputs3]= FuncionVariable3(variable3);
   [inputs4]= FuncionVariable4(variable4);
   [inputs5]= FuncionVariable5(variable5);
   [inputs6]= FuncionVariable6(variable6);
   [inputs7]= FuncionVariable7(variable7);
   [inputs8]= FuncionVariable8(variable8);
   [inputs9]= FuncionVariable7(variable9);
   inputs =
{inputs1;inputs2;inputs3;inputs4;inputs5;inputs6;inputs7;inputs8;inputs9};

net= network(9,2,[1;1],[1 1 1 1 1 1 1 1 1;0 0 0 0 0 0 0 0 0],[0 0;1 0],[0 1]);
end

% Se importan los datos de la hoja de Excel para el objetivo de la red
% neuronal
filename = 'MatrizRoll2.xls';
% El objetivo siempre seran estas
target1 = xlsread(filename,'A2:A60002');
target2 = xlsread(filename,'B2:B6439');

targets = [target1;target2];
```

```
% Para que tanto la entrada como el objetivo sean un vector fila
targets = transpose(targets);


%*************************************************************************
%                       PROGRAMACION DE LA RED NEURONAL
%*************************************************************************
% Se define la tipología y la función de transferencia
% Número de neuronas en la capa oculta
net.layers{1}.size = 2;
% Función de transferencia de la capa oculta
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'purelin';
% Configuración de la red; Con este comando también se inicializan los pesos
% y los sesgos de la red.
net = configure(net,inputs,targets);
% Representación de la red configurada
view(net);

% Parámetros para el entrenamiento y cálculo de la respuesta de la red
% Entrenamiento de la red
net.performFcn = 'mse';
net.trainFcn = 'trainlm';
net.trainParam.epochs = 100;
net.trainParam.max_fail = 100;
% Los datos de entrada y objetivo se dividen aleatoriamente
% en conjuntos de datos de entrenamiento, test y validación.
net.divideFcn = 'dividerand';

%*************************************************************************
%               ENTRENAMIENTO Y RESULTADOS DE LA RED NEURONAL
%*************************************************************************
% El comando train realiza el entrenamiento de la red
[net,tr] = train (net,inputs,targets);


% plotperform representa las actuaciones de entrenamiento, validación y
prueba.
plotperform(tr)

% Respuesta de la red después del entrenamiento
final_output = net (inputs)

% Plotregression representa la regresión lineal de los objetivos en la
% relación con la salida
plotregression(targets,final_output)

%*************************************************************************
%               REINICIALIZACION DE LOS PESOS DE LA RED NEURONAL
%*************************************************************************
B = input('¿Desea reinicializar los pesos de la red?(si/no): ','s');
if B=='si';
% Se inicializan de nuevo los valores de los pesos para obtener un
% resultado más exacto
net = init(net)
net = train (net,inputs,targets);
final_output1 = net (inputs)
else B=='no';
end

disp('FIN')
```