Universidad Zaragoza
1542

# Trabajo Fin de Grado
# ANEXOS

## Título del trabajo:

Redes bayesianas informadas por la física aplicadas a sistemas estructurales

## English tittle:

Physically-informed neural networks applied to structural systems


Autor/es

Pablo Falcón Gutiérrez

Director/es

Beatriz García Moya

Elías Cueto Prendes


Ingeniería mecánica


Universidad de Zaragoza

Escuela de ingeniería y arquitectura


2022-2023

# Contenido

# Anexo A: Fundamentos teóricos

Las redes neuronales son modelos computacionales basado en el funcionamiento biológico de las neuronas permitiendo así la clasificación y predicción de datos. En el caso de este trabajo, el desarrollo y programación de las redes se fundamentará en el Teorema de Bayes para dar un enfoque estocástico al aprendizaje.

## 1. Teorema de Bayes

La base en la que se sustentan las redes neuronales bayesianas, como su propio nombre indica, es el Teorema de Bayes. El Teorema de Bayes se presenta como:

(A.1) $$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

Siendo:

$P(\theta|D)$: Probabilidad a posteriori (o *posterior*)

$P(D|\theta)$: Probabilidad condicionada o verosimilitud (o *likelihood*)

$P(\theta)$: Probabilidad a priori (o *priori*)

$\mathrm{P(D)}$: Probabilidad marginal (o *evidence*)

La probabilidad de los datos, debido al teorema de la probabilidad, se aproxima como la suma así que se transforma como la integral al ser un término continuo.

(A.2) $$P(\theta|D) = \frac{P(D|\theta)\,P(\theta)}{\int P(D|\theta)\,d\theta}$$

El Teorema de Bayes ajusta las distribuciones de probabilidad de los parámetros gracias a los datos. Es decir, siempre se parte de una creencia inicial conocida de los parámetros (priori) y mediante el teorema de bayes se actualiza a la distribución posterior.

$$\text{distribución priori} \rightarrow \text{datos} \rightarrow \text{distribución posterior}$$

El problema de este enfoque reside en el cálculo de la evidencia, ya que no es posible calcular su integral porque la solución no es analítica y, además, su complejidad aumenta conforme crece la profundidad y los parámetros de la red.

Las variaciones de $\theta$ solo afectarán a los términos $P(D|\theta)$ y $P(\theta)$

(A.3) $$P(\theta|D) \propto P(D|\theta)\,P(\theta)$$

## 2. Divergencia de Kullback-Leibler [5]

Dada la inviabilidad del cálculo de $P(D) = \int P(D|\theta)\,d\theta$, se debe encontrar una manera alternativa de conseguir la distribución de la posterior. Ésto se consigue mediante la

aproximación de la posterior como una nueva función que es ajustada mediante un proceso de optimización.

(A.4) $$q(\theta) \approx P(\theta|D)$$

Siendo:

$q(\theta)$: Aproximación de distribución de probabilidad

$P(\theta|D)$: Probabilidad de la posterior

Por lo tanto, para poder conseguir que esta aproximación sea válida, se necesita un paso intermedio donde validar cómo de parecida es la distribución posterior a la aproximación propuesta.

La Divergencia de Kullback-Leibler (Divergencia de KL) establece una forma cuantitativa de medir la "distancia" entre dos distribuciones de probabilidad. Si el resultado de la divergencia es nulo, entonces las dos distribuciones son idénticas. Por lo tanto, el objetivo será minimizar la divergencia. La divergencia de KL se puede expresar como:

(A.5) $$KL(q(\theta) \,||\, P(\theta|D)) = \mathbb{E}(\ln q(\theta)) - \mathbb{E}(\ln P(\theta|D))$$

Para tener un buen contexto de la divergencia de KL es importante conocer algunas de sus propiedades:

- No es simétrica:

(A.6) $$KL(q(\theta) \,||\, P(\theta|D)) \neq KL(P(\theta|D) \,||\, q(\theta))$$

- Siempre es positiva:

(A.7) $$KL(q(\theta) \,||\, P(\theta|D)) > 0$$

Siguiendo con el desarrollo de la divergencia de KL, en la ecuación (A.8) vuelve a aparecer el término $\mathbb{E}(\ln P(\theta|D))$, que no se puede resolver.

(A.8) $$\mathrm{K}L(q(\theta) \,||\, P(\theta|D)) = \mathbb{E}(\ln q(\theta)) - \mathbb{E}(\ln P(\theta, D)) + \ln P(D)$$

## 3. ELBO (Evidence Lower BOund) [5]

Modificando la ecuación (A.8) aparece la función llamada "límite inferior de la evidencia" o también llamada en inglés *Evidence Lower BOund* (ELBO) que transformará el problema en uno de optimización:

(A.9) $$\begin{aligned} \mathrm{K}L(q(\theta) \,||\, P(\theta|D)) &= \mathbb{E}(\ln q(\theta)) - \mathbb{E}(\ln P(\theta|D)) \\ &= \mathbb{E}(\ln q(\theta)) - \mathbb{E}(\ln P(\theta, D)) + \ln P(D) \\ &= -\left(\frac{\ln P(\theta,D)}{\ln q(\theta)}\right) + \ln P(D). \end{aligned}$$

Reorganizando la ecuación (A.9):

(A.10) $$ELBO = \mathbb{E}\left(\frac{\ln P(\theta, D)}{\ln q(\theta)}\right) = \ln P(D) - \mathrm{KL}(q(\theta) \,||\, P(\theta|D))$$

Ahora el problema de optimización se convierte en la maximización de la ELBO para poder minimizar la divergencia de KL entre las dos distribuciones, que es el objetivo principal. Al maximizar la ELBO también se está maximizando la evidencia $\ln P(D)$.

(A.11) $$argmax\ ELBO = argmax\ \mathbb{E}\left(\frac{\ln P(\theta, D)}{\ln q(\theta)}\right)$$

Una vez que termina el problema de optimización se obtiene la distribución de la aproximación a la posterior $q(\theta *)$.

Una posible interpretación gráfica para el problema de optimización de la ELBO quedaría reflejada en la figura (A.9). Se observa cómo la distribución aproximada trata de minimizar su divergencia de KL hasta que ya no se lo permite su dominio.

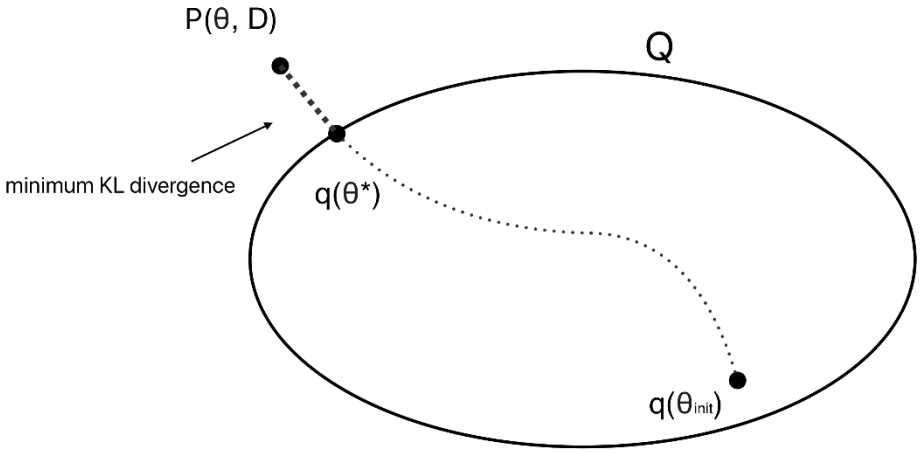


Figura A.1 Representación gráfica ELBO

Siendo:

- Q: Dominio de las distribuciones aproximadas $q(\theta)$
- $q(\theta *)$: Distribución aproximada optimizada
- $q(\theta_{init})$: Distribución aproximada inicial
- $P(\theta, D)$: Distribución real

## 4. Redes neuronales

Las redes neuronales son modelos matemáticos inspirados en las conexiones neuronales humanas. Estos modelos tienen la capacidad de aprender por sí solos a través de información que se muestra (bases de datos) y así poder predecir resultados con datos que no habían visto anteriormente.

Este tipo de modelos tienen una ventaja muy importante ya que no hay que preocuparse de programarlos explícitamente para poder resolver un problema. A la vez

esto tiene otra gran desventaja y es que una vez que se tiene el modelo no se ha conseguido dar una explicación a los resultados, por lo tanto, muchas veces son impredecibles.

A continuación, se comentarán las dos redes neuronales que se van a tratar en este trabajo, las redes neuronales artificiales y las redes neuronales bayesianas.

## 4.1. Redes neuronales artificiales

Las redes neuronales artificiales también llamadas *Artificial Neural Network* (ANN) son modelos deterministas ya que ante una misma entrada siempre se le asocia una salida.

Para poder entender mejor la estructura de las ANN conviene fijarse con mayor detalle en la unidad más básica, la neurona artificial. El nombre que recibe no es casualidad y ésta, al igual que la neurona biológica tiene unas conexiones de entrada y otras de salida.

Cada neurona está compuesta de unos pesos (w) y unos sesgos o *bias* (b), que son parámetros de la neurona y se actualizan durante la etapa de entrenamiento de la red. El procesamiento de información en una neurona se expresa como una suma ponderada de los pesos más el término *bias*:

(A.12) $$z = \sum_i w_i \, x_i \, + b$$

Siendo:

i: número de entradas

$x_i$: dato de entrada en la neurona

$w_i$: peso

$b$: sesgo

Para la salida de la red se le puede añadir una función no lineal, esta es llamada función de activación. Tiene la función de introducir no linealidades para aprender relaciones y patrones complejos.

(A.13) $$y = f(z) = f(\sum_i w_i \, x_i \, + b)$$

En la figura A.2 se muestran algunas de las funciones de activación más comunes y que además son detalladas a continuación:

- Escalón: La función de activación escalón, también conocida como función de paso, asigna un valor de salida de 1 si la entrada es mayor o igual a cero, y un valor de salida de 0 en caso contrario. Puede ser útil para problema de clasificación binaria.

- ReLU: llamada así por su nombre en inglés "*Rectified Linear Unit*". Es la función más utilizada para las redes neuronales. Mapea los valores negativos a cero y mantiene los valores positivos sin modificar.
- Sigmoide: también conocida como función logística, transforma los valores de entrada en un rango entre 0 y 1
- Softmax: se utiliza comúnmente en problemas de clasificación multiclase.

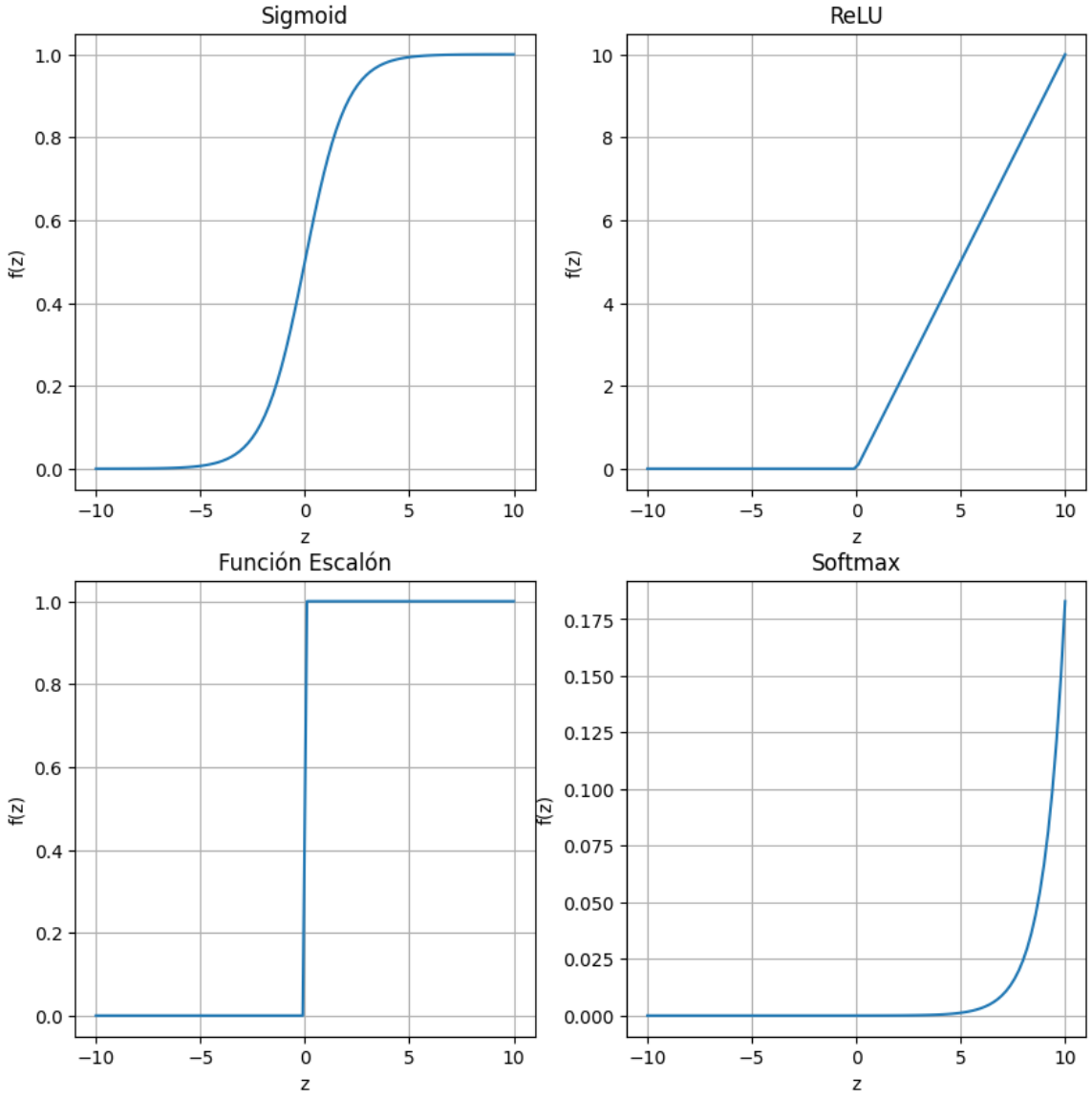


Figura A.2 Funciones de activación

A su vez la neurona se puede agrupar en varias neuronas formando la estructura llamada capa. Por último, las capas se agrupan y así forman la red neuronal. Se diferencian 3 tipos de capas dentro de la red neuronal:

- Capa de entrada
- Capas ocultas
- Capa de salida

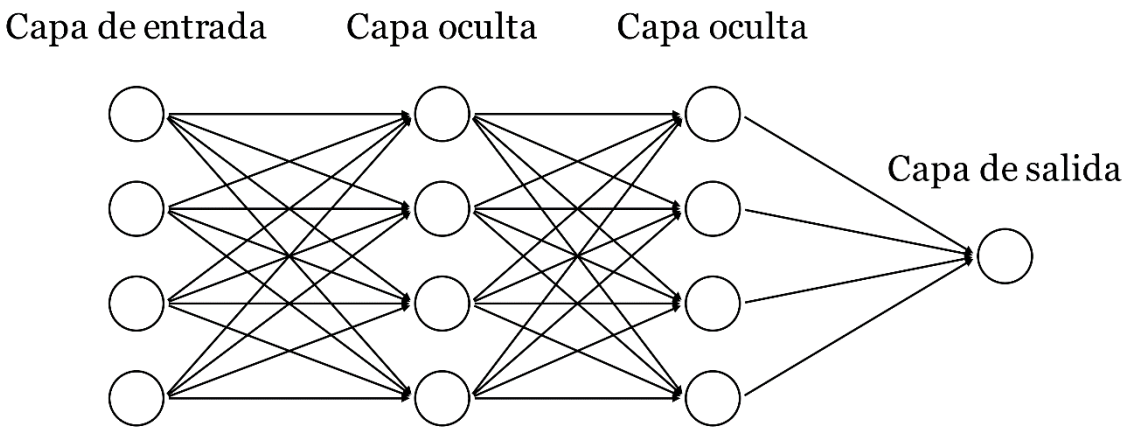


Figura A.3 Capas de la red

Cuanto mayor es el número de capas ocultas mayor es la complejidad de la red. Esto hace que se adapte mejor a funciones complicadas. Sin embargo, aumentarlas en exceso puede provocar sobreajuste o también llamado en inglés *overfitting*.

Para poder predecir un resultado (*forward pass*) se procesan los datos a través de cada una de las neuronas de la red como aparece en la figura A.3. Por último, la capa de salida proporciona un resultado que será la predicción de nuestro modelo.

Para poder optimizar la red debemos añadir una función de coste para que evalúe el error que estamos cometiendo en cada predicción que se realiza con la red. El más común y que además será utilizado en el proyecto es el error medio cuadrático o *mean squared error* (MSE):

(A.14)
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Siendo:

$n$: Número de muestras

$y_i$: Predicción del modelo o salida de la red

$\hat{y}_i$: Etiqueta de los datos de entrada

El error obtenido se propaga en sentido contrario (hacia atrás, llamado *Backward propagation*) para así poder modificar los parámetros convenientemente. Para determinar la penalización que debemos usar para cada uno de los parámetros $\theta$ se hace uso de las derivadas parciales. La técnica consiste en derivar la función de coste respecto a cada uno de los parámetros para poder encontrar el error proporcional que ha realizado cada parámetro $\frac{\partial L}{\partial \theta}$. Conocer el gradiente de la función de coste en función de sus parámetros se puede asemejar a conocer la dirección en la que cada parámetro debería de cambiar para poder minimizar la función de coste.

Para poder minimizarla de forma controlada se hace uso de un hiperparámetro llamado tasa de aprendizaje o *learning rate* (lr).

(A.15)
$$\theta_{i+1} = \theta_i + \alpha * \frac{\partial L}{\partial \theta_i}$$

Siendo:

$L$: Función de coste

$\theta_i$: Parámetro en iteración i

$\alpha$ : Tasa de aprendizaje

Cuando se realiza esta operación numerosas veces se llega al mínimo de la función de coste. A veces no tiene por qué alcanzar el mínimo global y se puede atascar en un mínimo local. En la figura A.4 se muestra cómo con el mismo número de iteraciones, pero diferente tasa de aprendizaje se fuerza a que el modelo llegue a un mínimo local.

Se observa como el ante un *learning rate* más alto (figura A.4 a) se puede evitar más fácilmente los mínimos locales que con uno más bajo (figura A.4 b).

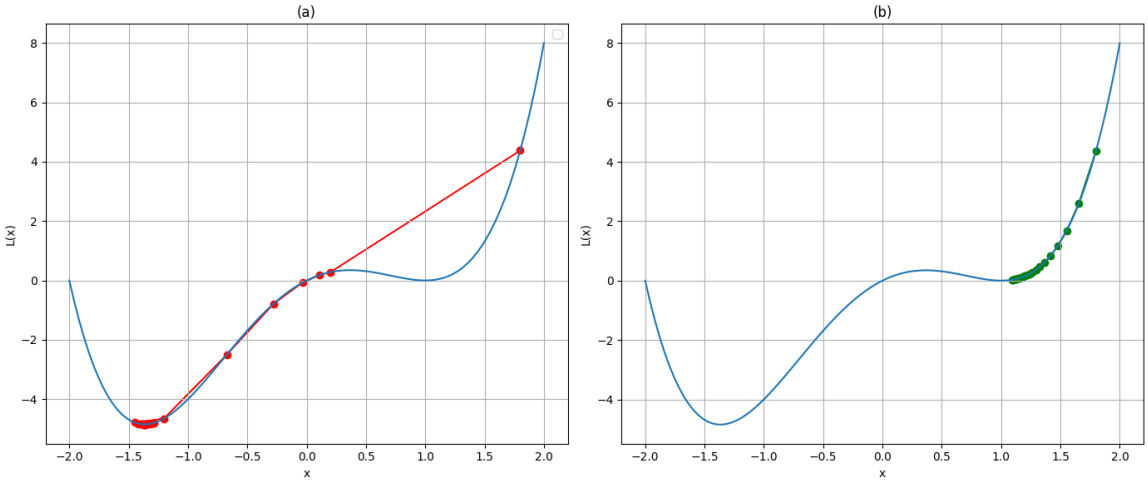


Figura A.4 Optimización global vs local

En la figura A.5 se aprecia la importancia de este hiperparámetro. Con un ejemplo tan sencillo como el citado es fácil de explicar los 3 posibles casos que nos pueden aparecer en el entrenamiento de una red neuronal. Para poder realizar una buena comparación se realiza el mismo número de iteraciones:

- *Learning rate* alto, figura A.5.a: No llega a alcanzar el óptimo porque el paso que se da es demasiado grande, se alcanzan óptimos inestables. Se debe de reducir el *learning rate*.
- *Learning rate* bajo, figura A.5.b: En este caso, el *learning rate* es demasiado bajo y el algoritmo de descenso del gradiente da pasos pequeños en cada iteración. Esto

puede llevar a una convergencia lenta y a quedar atrapado en mínimos locales en lugar del mínimo global. Se debe de aumentar el *learning rate*.

- *Learning rate* óptimo, figura A.5.c: Se encuentra el *learning rate* adecuado para el problema, el mínimo encontrado no es inestable, se adapta en todo momento al gradiente de la función de coste y alcanza el mínimo global.

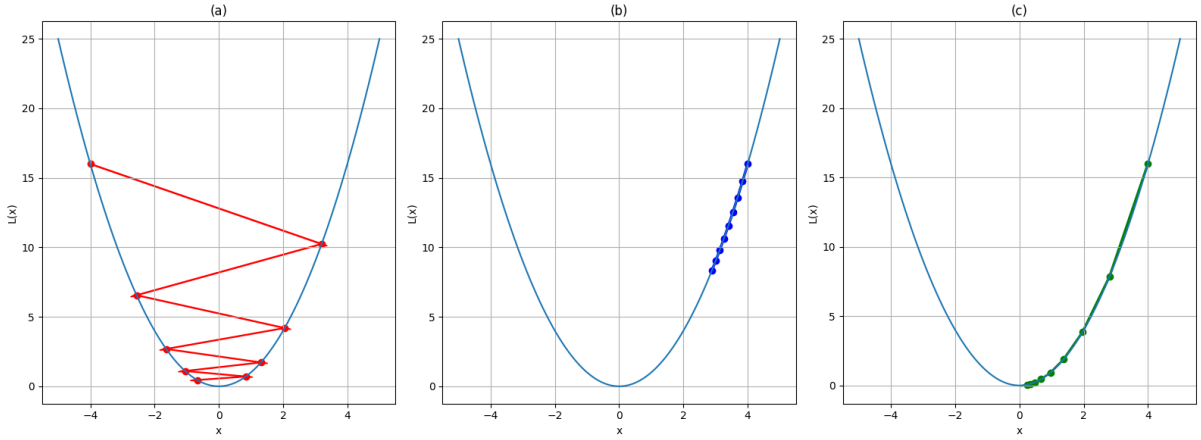


Figura A.5 Parametrización del learning rate

## 4.2.Redes neuronales bayesianas

Las redes neuronales bayesianas o *bayesian neural network* (BNN) se basan en las ANN. La gran diferencia es que las BNN están basadas en un modelo probabilístico. Concretamente los pesos de las neuronas ya no se tratan como números fijos que se aprenden, sino que pasan a ser distribuciones de probabilidad.

Este enfoque le aporta nuevas ventajas:

- Aparición de la incertidumbre: Debido a que el modelo es estocástico (ante una misma entrada la salida es diferente debido a la introducción de la aleatoriedad) se puede realizar varias predicciones del modelo y asociar una incertidumbre a cada predicción. Esta característica es muy importante ya que se sabe la predicción y se posee una incertidumbre asociada por lo que así se sabe lo fiable que es esa predicción.
- Disminución del sobreajuste: Generalizan más eficientemente que las redes neuronales normales.
- Aprendizaje con datos limitados: No necesitan bases de datos tan grandes como las redes neuronales normales ya que con una pequeña parte de los datos son capaces de generalizar adecuadamente el resto de los datos.

### 4.2.1. Neurona

La nueva definición de los pesos de las neuronas se formula en base a la probabilidad:

(A.16) $$\epsilon \sim \mathcal{N}(0,1)$$

Siendo:

- $\epsilon$: Distribución de probabilidad normal que modela la incertidumbre en cada predicción.

(A.17) $$z = \mu + \log(1 + e^{\rho}) * \epsilon$$

(A.18) $$\theta = (\mu, \rho)$$

Siendo:

- $\mu$: la media de la distribución de probabilidad normal de los parámetros
- $VARIANZA = e^{\rho}$

El intervalo de confianza es la nueva característica que se le añade a este modelo. Para poder calcularlo se sigue el siguiente procedimiento:

1. Se realizan varias predicciones del modelo y se almacenan los resultados.
2. Los resultados son procesados para extraer la media y desviación típica.
3. Se establece el nivel de confianza asociado al intervalo, generalmente entre el 90% y 95%.
4. Por último, se obtienen los rangos del intervalo, estos están centrados en la media de la distribución normal.

## Anexo B: Parámetros de la red

En la memoria del trabajo de fin de grado aparecen 3 tipos de parámetros, van a estar diferenciados por el tipo de red donde se usen:

- Parámetros que comparten las dos redes:
  - max_epoch: Equivale al número de épocas que se realizan en un modelo. Durante una época se realiza la predicción de los ejemplos a entrenar y además se realiza una actualización de los parámetros en función del error cometido en esa primera predicción. El valor de este parámetro puede depender del tipo de red, en general para las redes BNN no es necesario tantas épocas como para las ANN. Además, también puede variar según la dificultad del modelo a entrenar.
  - train_percent: Indica la parte de la base de datos total que sirve para entrenar la red neuronal. El resto de los datos serán destinados para comprobar los resultados.
  - lr: tasa de aprendizaje o *learning rate*, indica el paso de avance en cada actualización de los parámetros. (Explicado con más detalle en anexo A apartado 4.1 Redes neuronales bayesianas)
  - hidden_vec: Define el tamaño de la red neuronal, el número dentro de los corchetes indica el tamaño de cada capa y el numero por el que se multiplica indica el número de capas de la red.
  - miles: Indica en el número de épocas exacto en el que se reduce la tasa de aprendizaje para así no tener que partir de una tasa de aprendizaje muy pequeña.
  - gamma: Es un valor por el que se multiplica la tasa de aprendizaje justo en el momento que lo indique el parámetro *miles*. El valor de este parámetro siempre será menor a la unidad.
  - lambda_d - lambda_deg - lambda_r: Durante el entrenamiento, al sumar el error cometido por las condiciones de degeneración (deg), el error de las predicciones (d) y la regularización de los pesos (reg) aparecen estos parámetros para ponderar la importancia que se le quiere dar a cada término. Esta necesidad es debido a que si un término es mucho más pequeño que el otro la red optimizará los parámetros para que solamente disminuya el de mayor magnitud porque el resto van a ser despreciados.
- Parámetros propios de las redes bayesianas:
  - lambda_kl: Es idéntico a los parámetros lambda_d y lambda_deg pero este error solo aparece en las redes neuronales bayesianas. Va asociado al error cometido en la distribución de Kullback-Leibler.
  - n_samples: En las redes neuronales convencionales en cada época solo se realiza una predicción de los datos de entrenamiento, pero en las redes bayesianas al tener cierta estocasticidad se permite realizar más de una

predicción. Por lo tanto este parámetro indicará el número de predicciones a realizar, al tener varias predicciones se realiza la media.

- n_samples_ELBO: Este parámetro indica el número de muestras que se quiere que la red prediga en la fase de comprobación de los resultados.
- CI: Indica la probabilidad de que los valores predichos se encuentren en el intervalo de confianza.

- Parámetros propios de las redes convencionales:
  - net_init: Método usado para la inicialización de los parámetros de la red.

# Anexo C: Mapas de variables de estado secundarios

## 1. Red ANN

### 1.1.Placa sin fallos – modelo gemelo digital

Se aportan gráficas secundarias algunas de ellas como la deformación y las tensiones en dirección x y dirección y son imprescindibles para construir el mapa del módulo de Young:

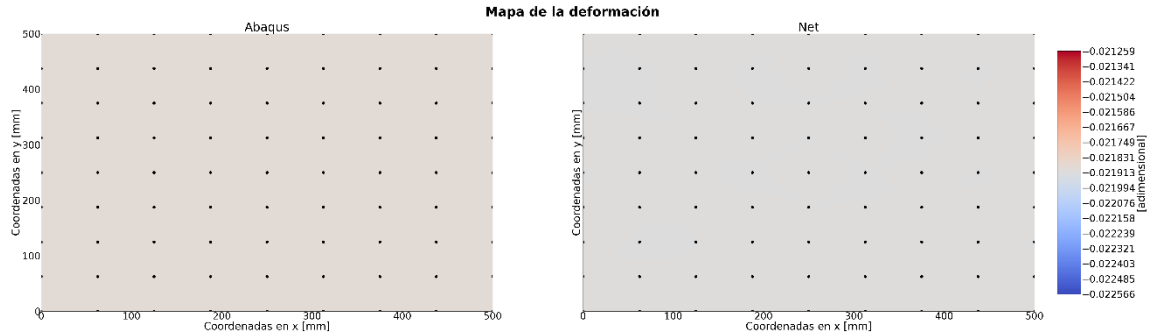


Figura C.1 Mapa de la deformación de la placa sin fallo. Gemelo digital.

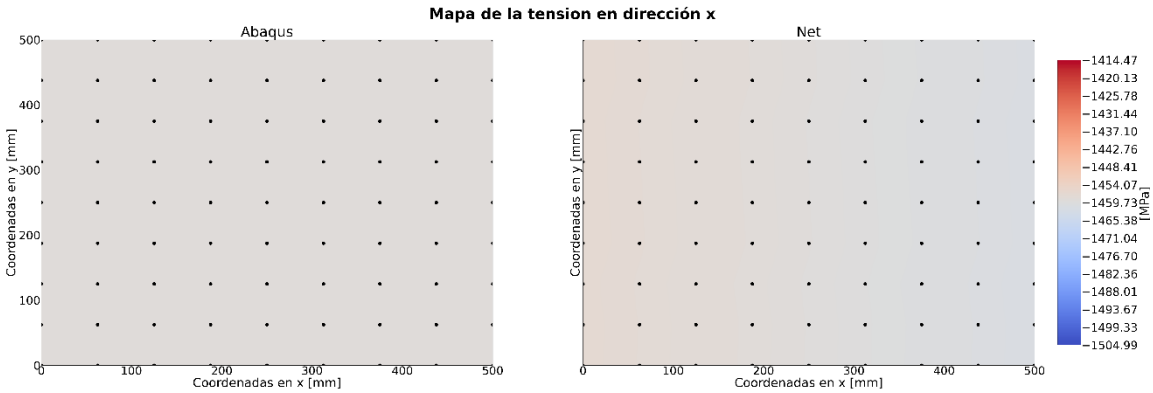


Figura C.2 Mapa de la tensión en dirección x de la placa sin fallo. Gemelo digital.

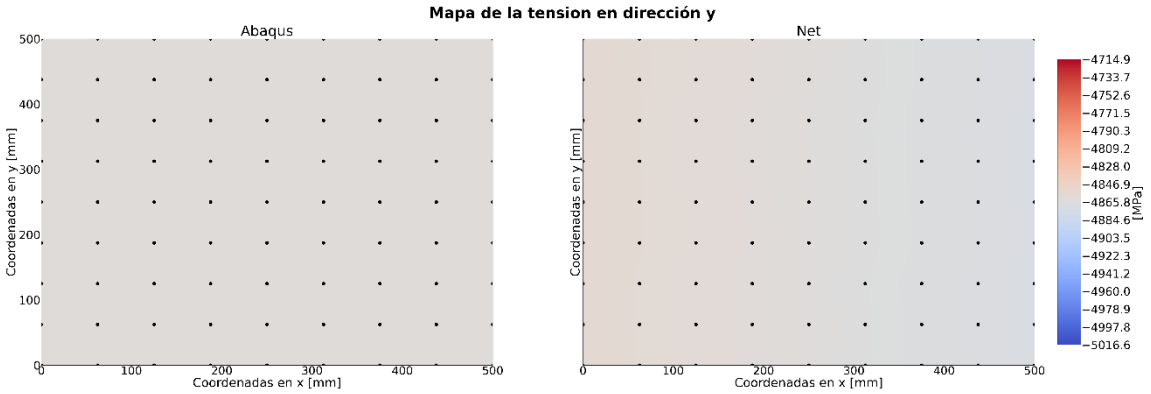


Figura C.3 Mapa de la tensión en dirección y de la placa sin fallo. Gemelo digital.

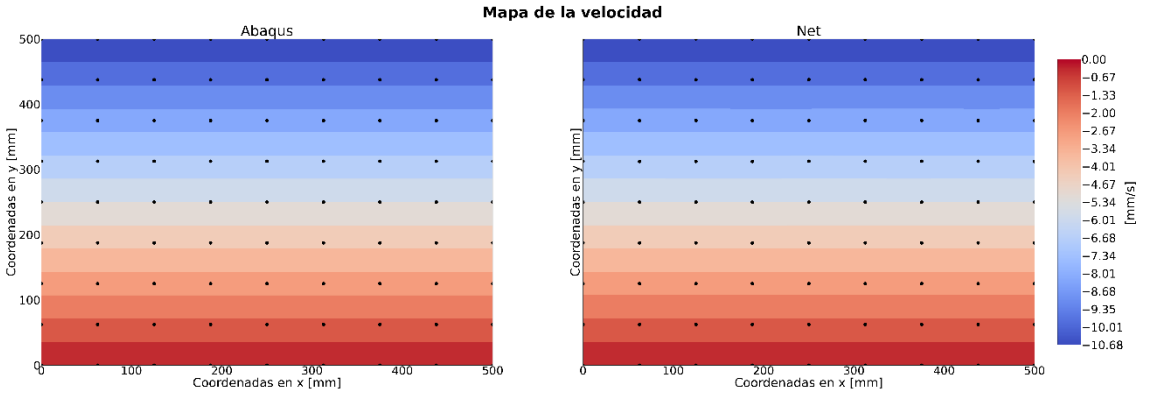


Figura C.4 Mapa de la velocidad de la placa sin fallo. Gemelo digital.

## 1.2. Placa con fallos – modelo gemelo digital

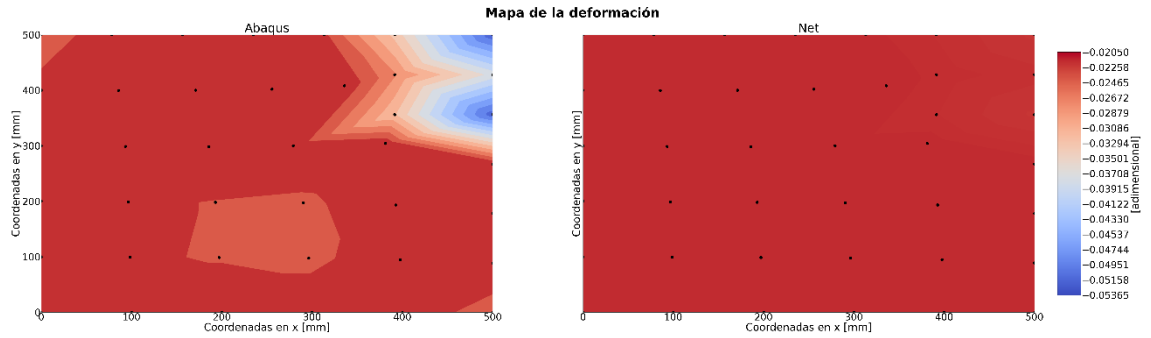


Figura C.5 Mapa de la deformación de la placa con fallo. Gemelo digital.

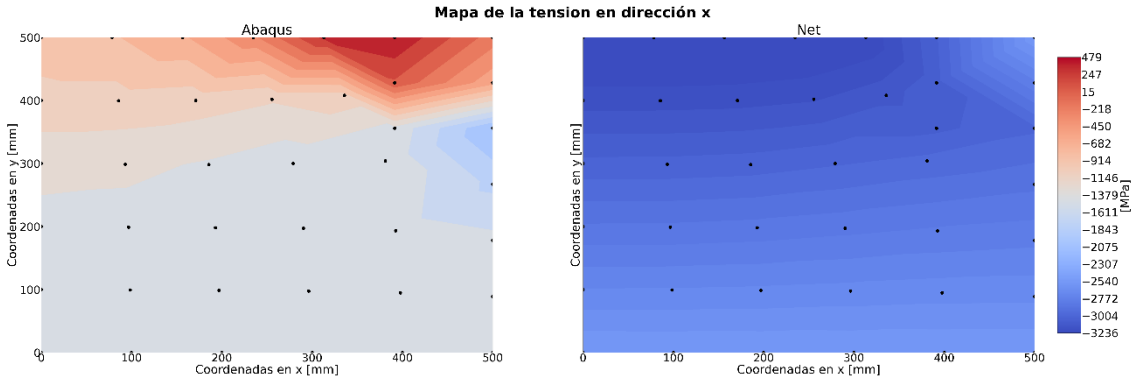


Figura C.6 Mapa de la tensión en dirección x de la placa con fallo. Gemelo digital.

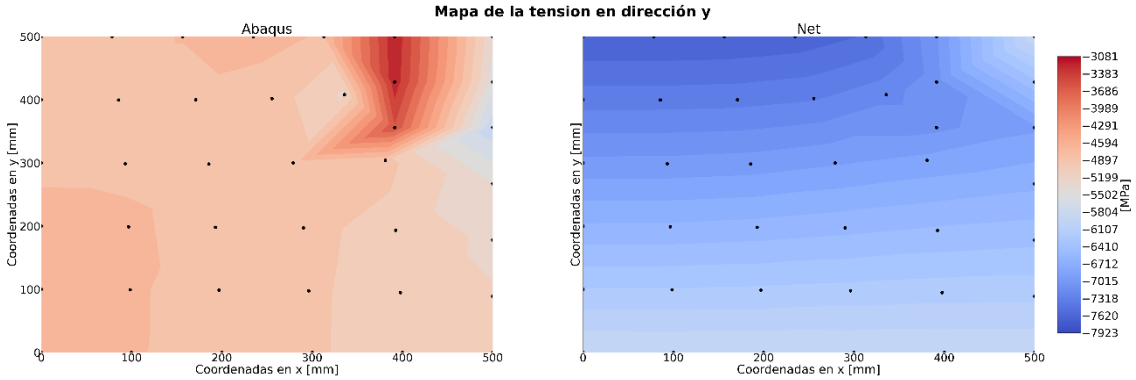


Figura C.7 Mapa de la tensión en dirección y de la placa con fallo. Gemelo digital.

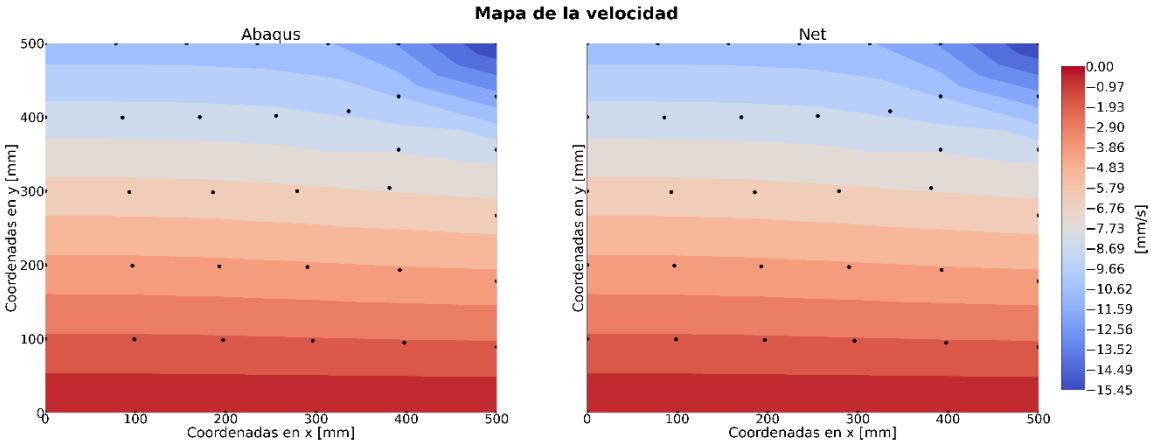


Figura C.8 Mapa de la velocidad de la placa con fallo. Gemelo digital.

## 1.3. Placa con fallos – modelo gemelo híbrido

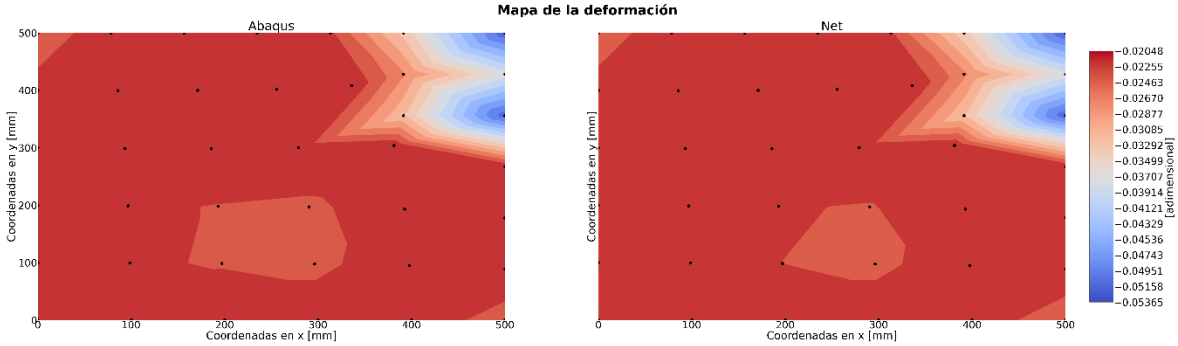


Figura C.9 Mapa de la deformación de la placa con fallo. Gemelo híbrido.

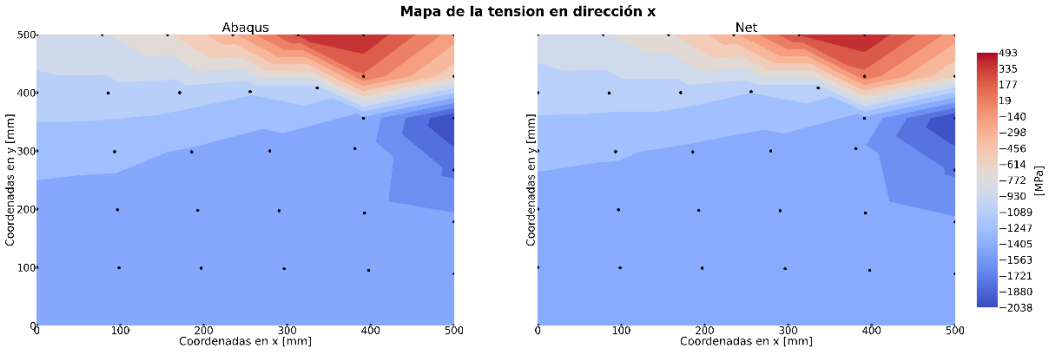


Figura C.10 Mapa de la tensión en dirección x de la placa con fallo. Gemelo híbrido.

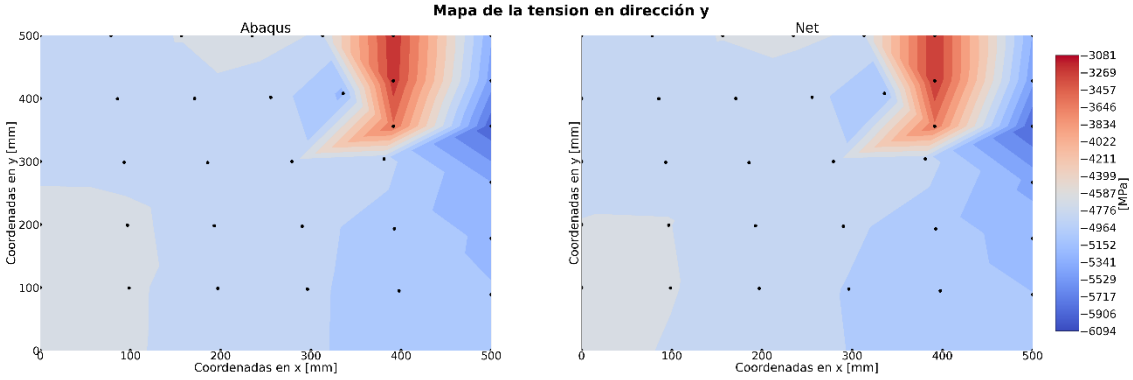


Figura C.11 Mapa de la tensión en dirección y de la placa con fallo. Gemelo híbrido.

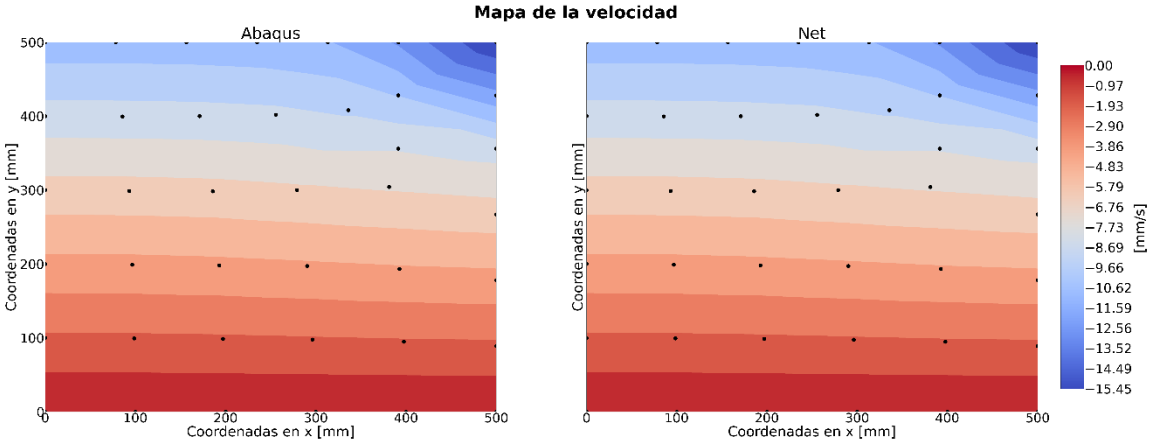


Figura C.12 Mapa de la velocidad de la placa sin fallo. Gemelo digital.

## 1.4.Placa con fallos – modelo gemelo híbrido – Test

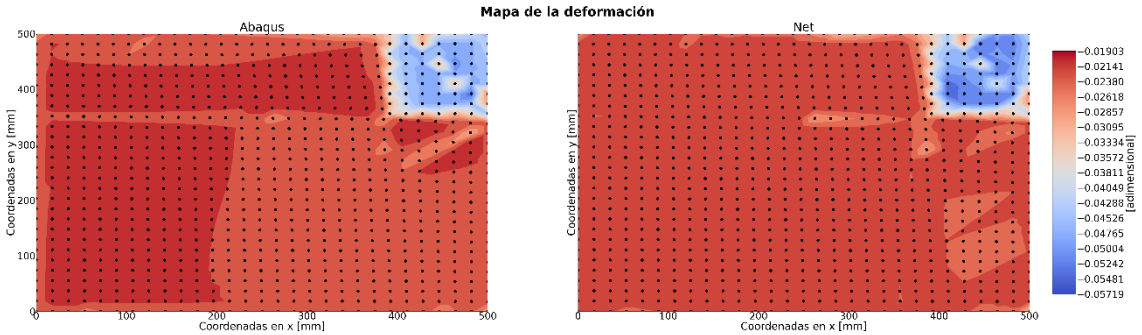


Figura C.13 Mapa de la deformación de la placa con fallo. Gemelo híbrido.

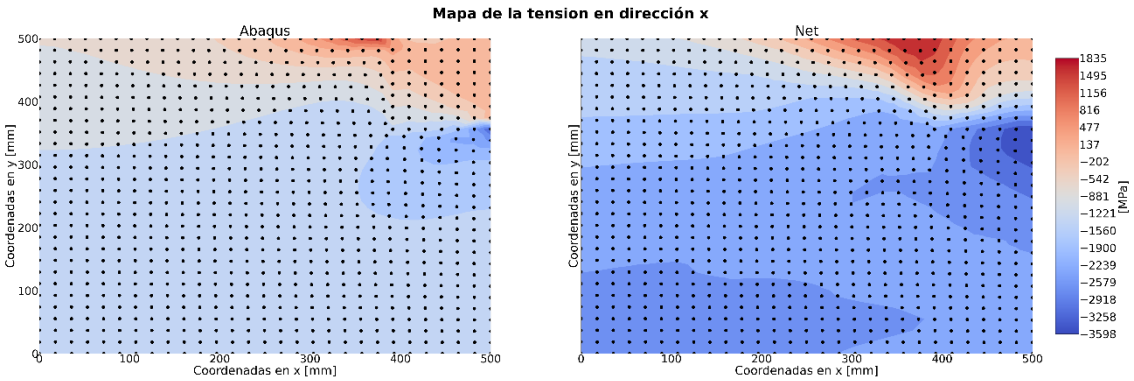


Figura C.14 Mapa de la tensión en dirección x de la placa con fallo. Gemelo híbrido.

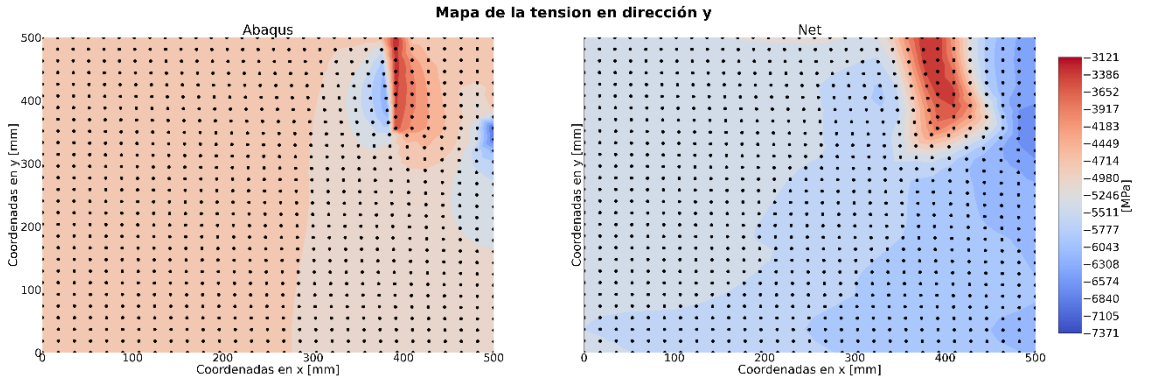


Figura C.15 Mapa de la tensión en dirección y de la placa con fallo. Gemelo híbrido.

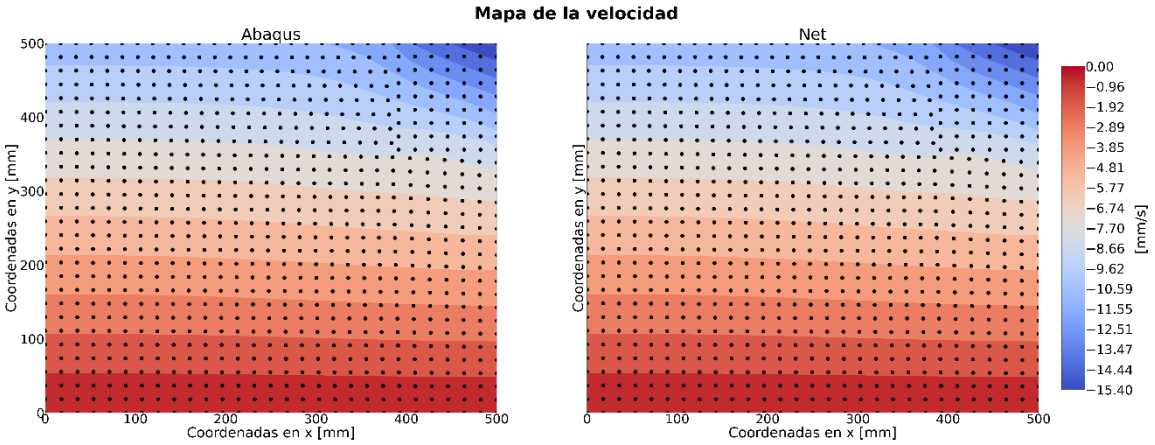


Figura C.16 Mapa de la velocidad de la placa sin fallo. Gemelo digital.

## 2. Red BNN

### 2.1.Placa sin fallos – modelo gemelo digital

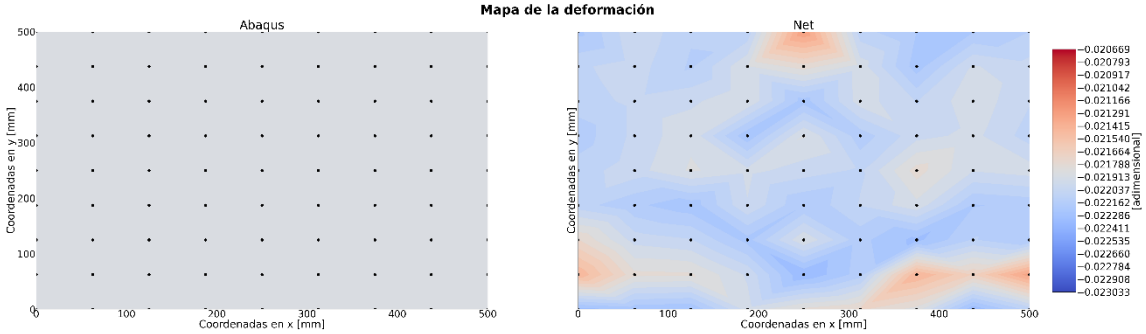


Figura C.17 Mapa de la deformación de la placa sin fallo. Gemelo digital.

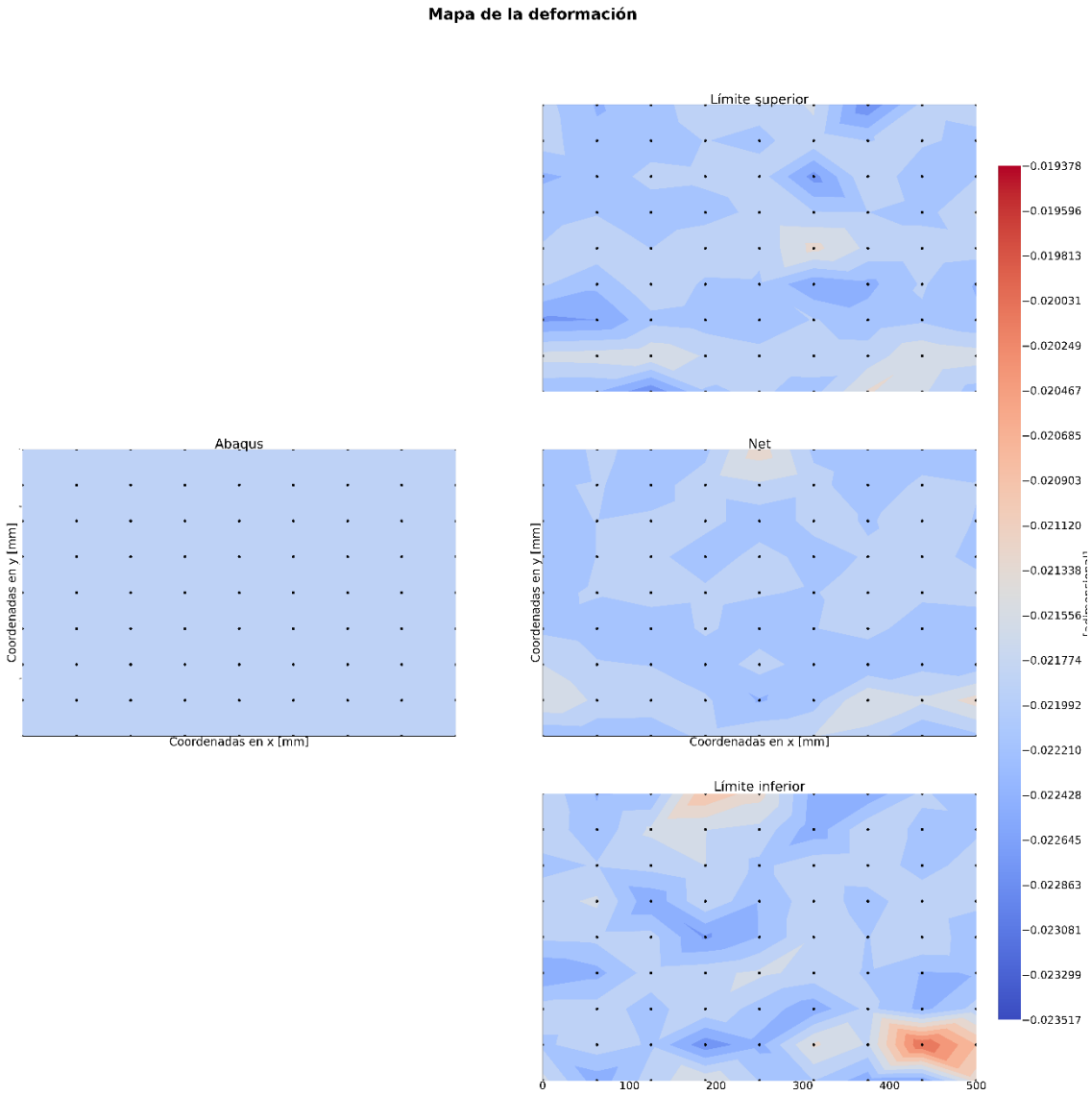


Figura C.18 Mapa de la deformación de la placa sin fallo. Intervalo de confianza al 95%. Gemelo digital.

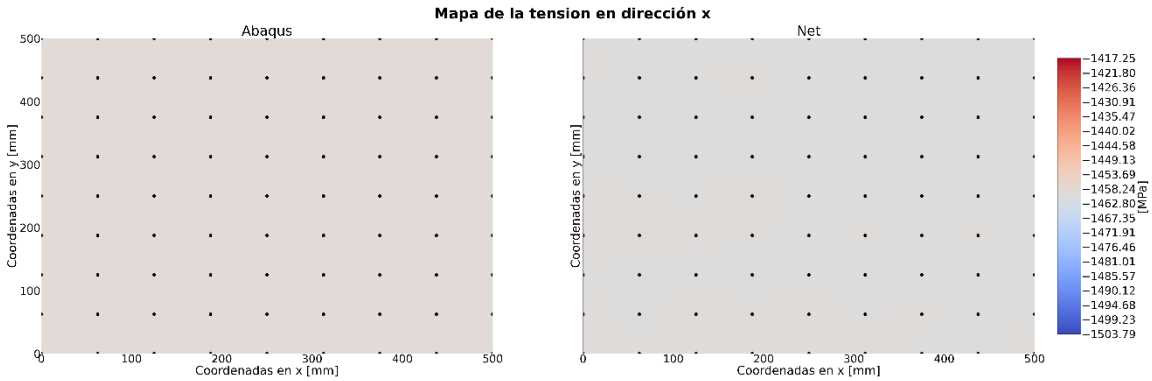


Figura C.19 Mapa de la tensión en dirección x de la placa sin fallo. Gemelo digital.

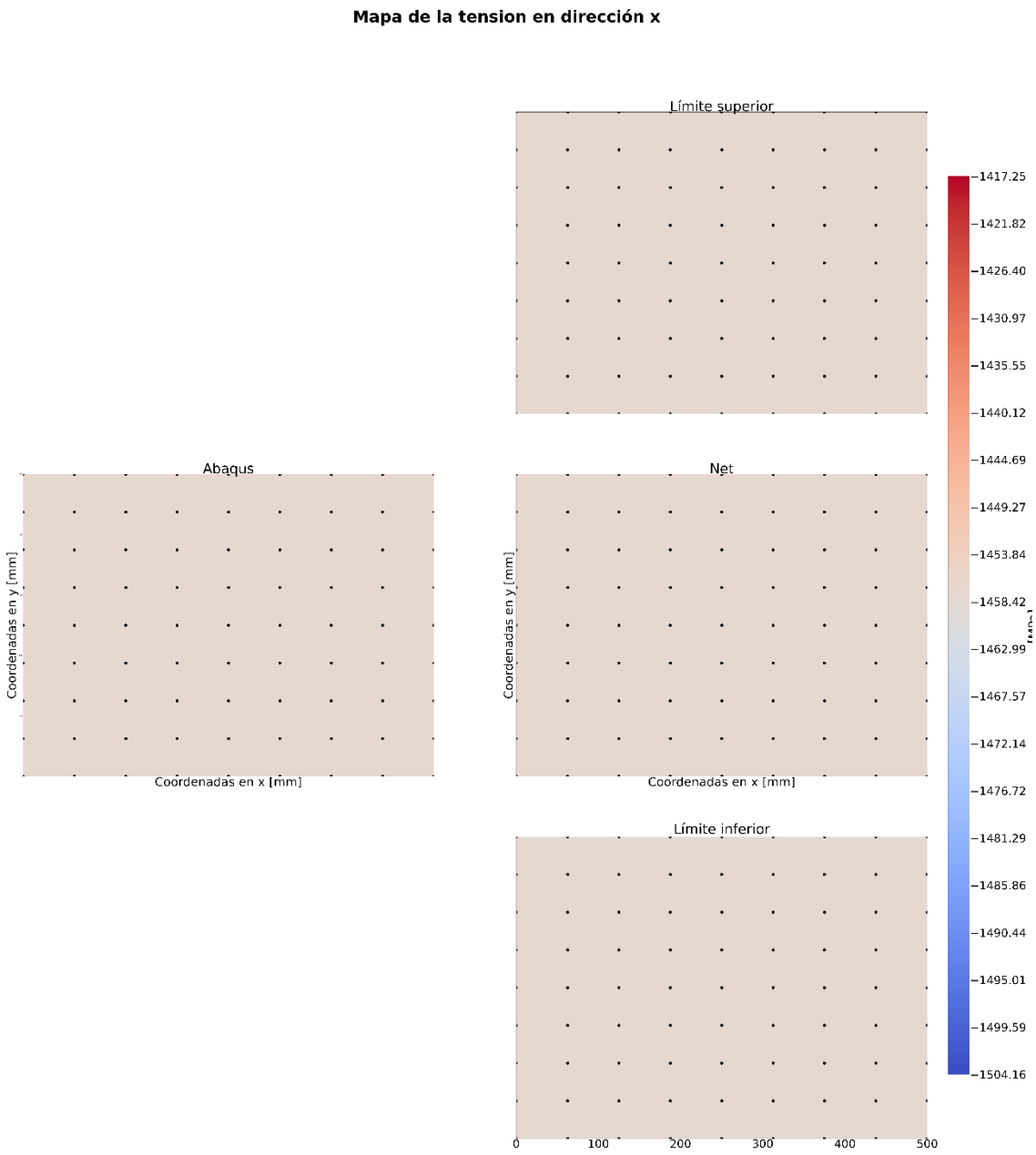


Figura C.20 Mapa de la tensión en dirección x de la placa sin fallo. Intervalo de confianza al 95%. Gemelo digital.


Figura C.21 Mapa de la tensión en dirección y de la placa sin fallo. Gemelo digital.

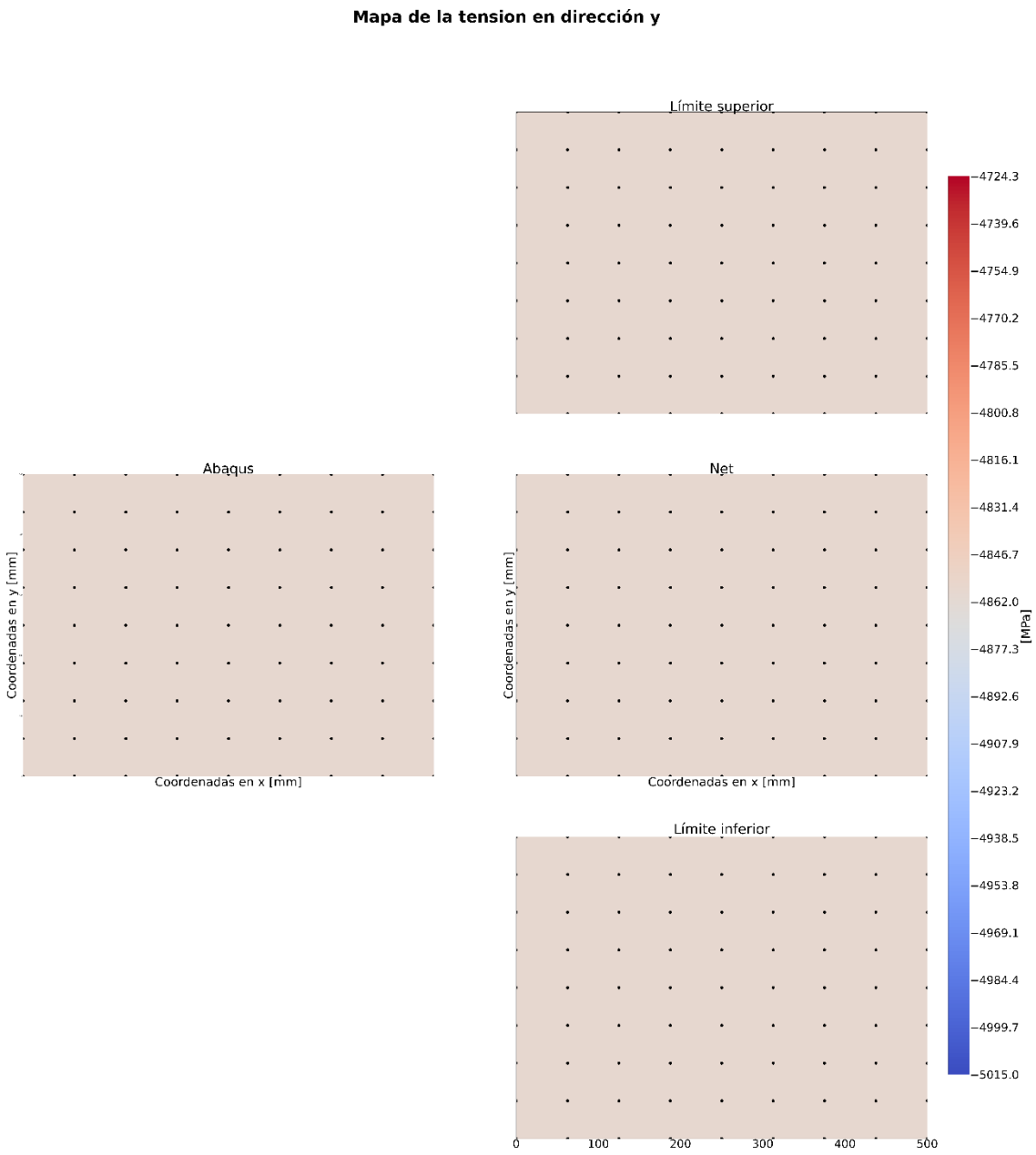


Figura C.22 Mapa de la tensión en dirección y de la placa sin fallo. Intervalo de confianza al 95%. Gemelo digital.

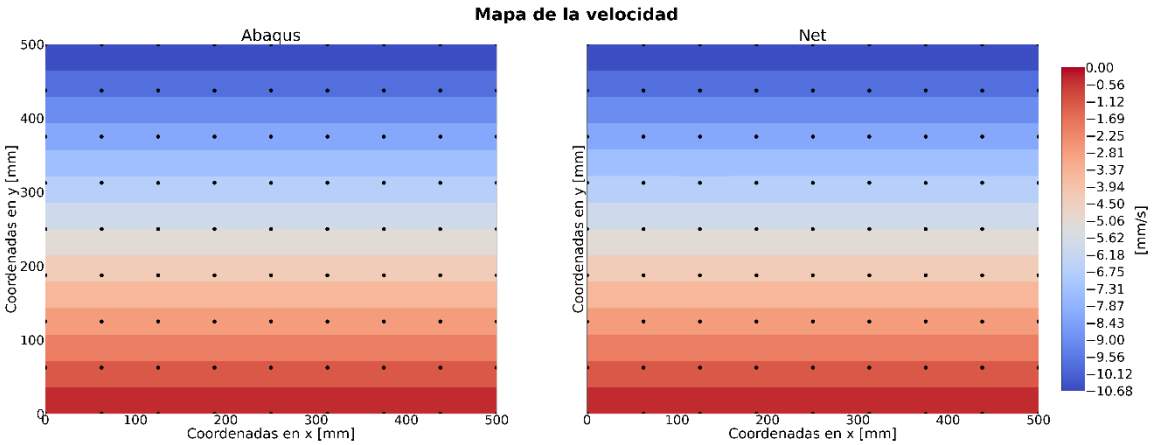


Figura C.23 Mapa de la velocidad de la placa sin fallo. Gemelo digital.

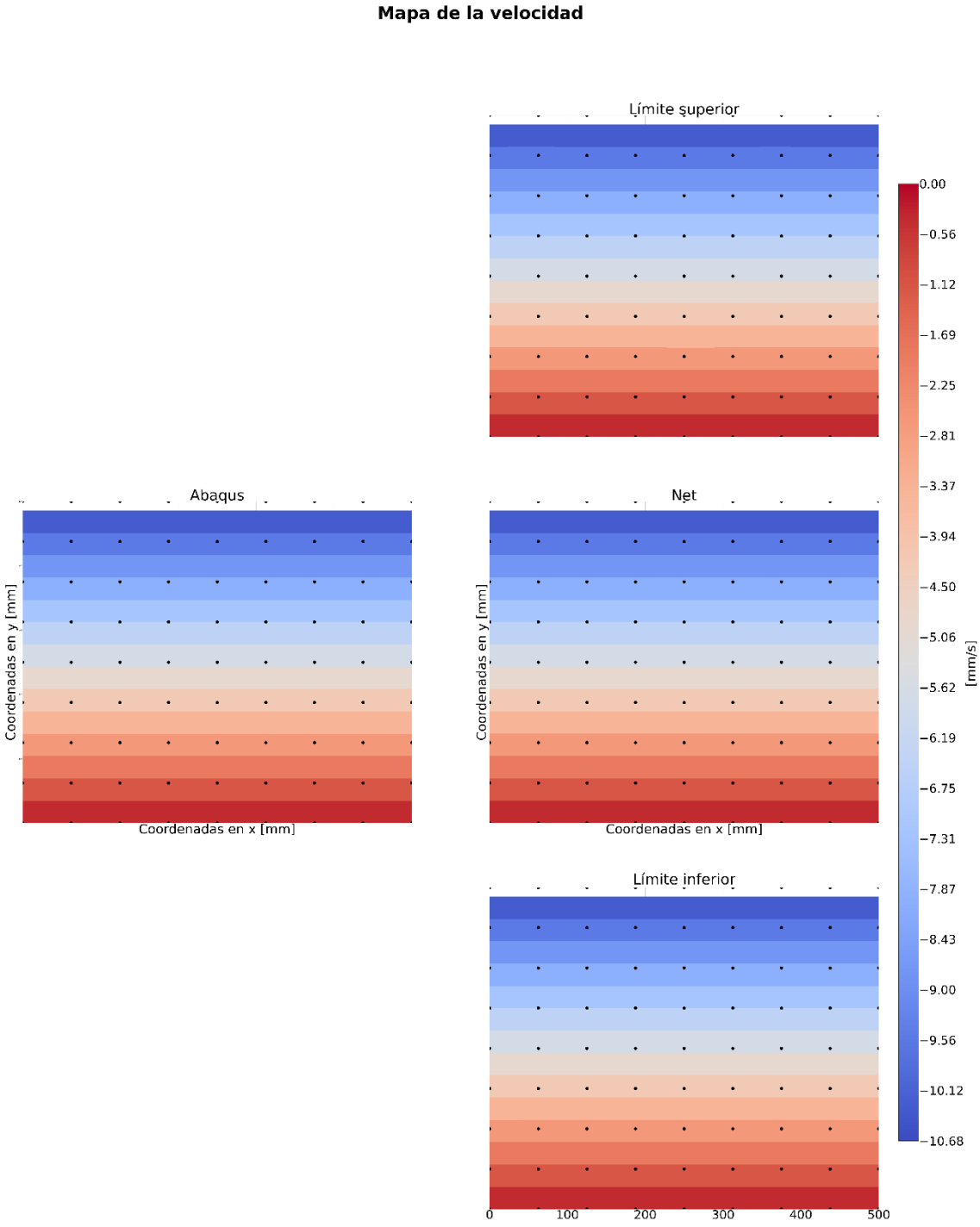


Figura C.24 Mapa de la velocidad de la placa sin fallo. Intervalo de confianza al 95%. Gemelo digital.

# 1 Glosario de abreviaturas

- TFG: Trabajo de Fin de Grado
- SPNN: *Structure-Preserving Neural Networks*
- IA: Inteligencia Artificial
- SHM: *Structural Health Monitoring*
- BNN: *Bayesian Neural Network*
- ANN: *Artificial Neural Network*
- KL: Kullback-Leibler
- lr: *Learning Rate* o tasa de aprendizaje
- GENERIC: *General Equation for the Non-Equilibrium ReversibleIrreversible Coupling*
- *ELBO: Evidence Lower BOund*
- ReLU: *Rectified Linear Unit*
- GT: *Ground Truth*
- MSE: *Mean squared Error*
- GD: Gemelo Digital
- GH: Gemelo Híbrido

# 2 Bibliografía

[5] https://uvadoc.uva.es/bitstream/handle/10324/57976/TFG-G5976.pdf?sequence=1