Study of an Adaptive Reservoir Computing Algorithm for Time Series Prediction based on the Kuramoto Model

Trabajo Final

Herramientas Computacionales 2023-I

Diego Esteban Quintero Rey

Ingeniería de Sistemas y Computación

Junio 13, 2023

Resumen:

This article explores the implementation and evaluation of an adaptive reservoir computing algorithm. The proposed algorithm employs a network of phase oscillators as the reservoir, allowing for effective processing of spatiotemporal information. The methodology involves generating time series, implementing the adaptive algorithm, and analyzing the results in terms of prediction accuracy, oscillator synchronization, and reservoir adaptability. The obtained results did not fully replicate the findings from the referenced article, suggesting potential issues with the adaptation mechanism and the solution method employed. Further investigation is needed to address these discrepancies and explore alternative approaches. The article provides insights into the implementation challenges and highlights the importance of examining library implementations to enhance the algorithm's performance. The limitations encountered and the need for additional research are discussed, along with a plan for future improvements. Overall, this work contributes to the understanding of adaptive reservoir computing and its potential for machine learning applications.

Introducción:

La computación de reservorios es un método para entrenar redes neuronales recurrentes que supera los desafíos asociados con la propagación inversa en el tiempo, como los gradientes que desaparecen o explotan. En este enfoque, el reservorio, que consiste en una red de neuronas recurrentes, se inicializa con pesos aleatorios y las conexiones entre las neuronas ocultas se mantienen fijas [1]. Luego, la capa de salida se entrena utilizando regresión lineal. Este artículo explora un modelo adaptativo de computación de reservorios propuesto en [2], que permite que la estructura del reservorio se adapte al problema sin requerir conocimientos expertos humanos.

El modelo de reservorio adaptativo propuesto emplea una red de N osciladores de fase como reservorio. La dinámica de los osciladores y los pesos de acoplamiento dentro del reservorio interactúan y evolucionan continuamente en respuesta a las entradas externas. La adaptación está guiada por dos ecuaciones: una para la evolución de las diferencias de fase entre los osciladores y otra para la actualización de los pesos de acoplamiento. El mecanismo adaptativo permite que el reservorio procese información espacio-temporal de manera efectiva. Para evaluar el rendimiento del algoritmo, los autores realizaron experimentos utilizando tres series temporales diferentes: NARMA10, MG17 y MSO12. Cada serie temporal se generó en base a ecuaciones y parámetros específicos. Las predicciones del algoritmo se compararon con la verdad absoluta y se calcularon diversas métricas como el error cuadrático medio (MSE) y el coeficiente de sincronización. Los gráficos y análisis resultantes tenían como objetivo replicar los hallazgos presentados en [2].

En este artículo, implementamos el algoritmo propuesto e intentamos reproducir los gráficos proporcionados en [2]. La metodología involucró la generación de la serie temporal, la implementación del algoritmo adaptativo y la realización de experimentos con valores de parámetros fijos. Los resultados se analizaron en términos de precisión de predicción, sincronización de los osciladores y las propiedades adaptativas del reservorio.

Los resultados obtenidos no replicaron completamente los hallazgos de [2]. Si bien los valores de MSE mostraron un rendimiento aceptable en términos de predicción, el coeficiente de sincronización no alcanzó el

nivel deseado. Además, los gráficos que comparan el MSE con el radio espectral ρ, la fuerza de acoplamiento λ y otras variables no exhibieron los patrones esperados observados en el artículo original.

A pesar de las desviaciones de los resultados esperados, este artículo proporciona ideas sobre la implementación y evaluación del algoritmo adaptativo de computación de reservorios. Las limitaciones encontradas y las discrepancias en los resultados obtenidos plantean la necesidad de una investigación adicional y la exploración de enfoques alternativos.

En las siguientes secciones, presentamos la metodología empleada, los resultados obtenidos y una discusión de los hallazgos. El código y los resultados experimentales detallados se pueden encontrar en el repositorio de GitHub adjunto (este6an13/rc-kuramoto-sad-rc (github.com)), lo que permite un examen más detallado y posibles mejoras en futuras investigaciones.

Marco teórico:

Computación de reservorios estática:

De acuerdo con [1], la computación de reservorios es un método para entrenar redes neuronales recurrentes sin propagación hacia atrás en el tiempo y los problemas asociados de gradientes que se desvanecen o explotan. Los pasos básicos son:

- Inicializar aleatoriamente los pesos de la red neuronal recurrente.
- Fijar los pesos de conexión ocultos.
- Entrenar la capa de salida lineal con regresión lineal.

El estado de la red neuronal recurrente de M neuronas está dado por sus activaciones ocultas $\mathbf{h} \in \mathbb{R}^M$ que están conectadas por una matriz fija e inicializada aleatoriamente $\mathbf{W}_\mathrm{h}$. Una secuencia de entrada $\mathbf{X}_{1:\tau}$ es embebida por una transformación lineal $\mathbf{W}_\mathrm{i}$ a un estado $\mathbf{h}_\tau$:

$$\mathbf{h}_1 = \sigma(\mathbf{W}_\mathrm{h}\mathbf{h}_0 + \mathbf{W}_\mathrm{i}\mathbf{x}_1)$$
$$\vdots$$
$$\mathbf{h}_\tau = \sigma(\mathbf{W}_\mathrm{h}\mathbf{h}_{\tau-1} + \mathbf{W}_\mathrm{i}\mathbf{x}_\tau)$$

Luego, el reservorio puede predecir linealmente la siguiente entrada $\mathbf{x}_{\tau+1}$ basada en el estado oculto $\mathbf{h}_\tau$:

$$\hat{\mathbf{x}}_{\tau+1} = \mathbf{W}_\mathrm{o}\mathbf{h}_\tau$$

Con $\mathbf{W}_\mathrm{o}$ mapeando de la activación oculta a la salida. Solo los parámetros de esta matriz de salida $\mathbf{W}_\mathrm{o}$ son entrenados. Así, los valores óptimos pueden ser obtenidos analíticamente usando regresión contraída con un valor de regularización λ:

$$\mathbf{W}_\mathrm{o} = (\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^T\mathbf{X}$$

Donde $\mathbf{H}$ son los estados ocultos, $\mathbf{I}$ es la matriz identidad y $\mathbf{X}$ son las salidas esperadas de la regresión.

Como modelo biológicamente inspirado, la computación de reservorios tiene ventajas únicas en el procesamiento de información espaciotemporal. Sin embargo, las arquitecturas típicas solo utilizan redes aleatorias estáticas o consideran la dinámica de las neuronas y la conectividad separadamente.

En [2] se propone un modelo de computación de reservorios cuya estructura se puede adaptar al problema sin conocimiento experto humano. Específicamente, se implementa un reservorio como una red de osciladores de fase adaptativos. En este sistema dinámico coevolucionario, la dinámica de los nodos y los pesos de acoplamiento en el reservorio constantemente interactúan y evolucionan juntos cuando son perturbados por la entrada externa.

Computación de reservorios adaptativa:

En [2] se propone un mecanismo que consiste en una red adaptativa de N osciladores perturbados por entradas externas, en la cual los pesos de acoplamiento entre los nodos y el estado de los elementos activos en los nodos interactúan y evolucionan juntos. Este es un mecanismo dinámico y adaptativo en el que los pesos de la matriz de adyacencia se van actualizando en cada iteración. Las dos fórmulas que guían esta adaptación son las siguientes:

$$\frac{d\theta_i}{dt} = \omega_i + \lambda \cdot \sum_{j=1}^{N} k_{ij} \cdot \sin\left(\theta_j - \theta_i + u(t)\right) \quad (1)$$

$$\frac{dk_{ij}}{dt} = -\epsilon \cdot \sin\left(\theta_j - \theta_i + \beta\right) \text{ con } |k_{ij}| \leq 1 \quad (2)$$

De esta manera, se tiene una red N nodos, cada uno con frecuencia natural ω_i. El parámetro λ es la fuerza de acoplamiento global. El valor k_{ij} es la entrada de la matriz de adyacencia K y $u(t)$ es la entrada de la serie de tiempo. Por otro lado, ϵ representa la escala de tiempo de la dinámica. Además, las entradas de la matriz de adyacencia son acotadas al rango $[-1, 1]$. En conclusión, la matriz de pesos K se adapta según los valores de las fases de los osciladores, es decir θ, que a su vez depende de la entrada de la serie de tiempo.

En [2] se presenta el siguiente algoritmo de entrenamiento. Este algoritmo es el mecanismo adaptativo presentado por los autores.

procedure RC
  Initialize K, neuronal states Θ, spectral radius ρ
  for $i = 1$ to L_{train} **do**
    Update the neuronal states Θ according to (1)
    if $i < L_{\text{adev}}$ **then**
      Update adjacency matrix K according to (2)
      Rescale spectral radius of matrix K: $\rho(K) = \rho$
    else
      Collect the neuronal states Θ
    end if
  end for
  Compute the output weight W_{out} by ridge regression
  for $i = 1$ to L_{test} **do**
    Update the neuronal states Θ according to (1)
    Collect prediction as $\hat{y}(t) = W_{\text{out}}\Theta(t)$
  end for
  Calculate the model error
end procedure

Metodología:

El objetivo del presente trabajo consistió en implementar el algoritmo propuesto por los autores e intentar replicar las gráficas que aparecen en [2].

Primero, se implementaron las funciones que generan las primeras tres series de tiempo utilizadas por los autores. Estas series de tiempo son NARMA10, MG17 y MSO12. Las ecuaciones que generan estas secuencias se presentan a continuación, en el orden mencionado

$$y(t+1) = 0.3 \cdot y(t) + 0.05 \cdot y(t) \cdot \left(\sum_{i=0}^{9} y(t-i)\right) + 1.5 \cdot u(t-9) \cdot u(t) + 0.1 \quad (4)$$

$$\frac{dy}{dt} = a \cdot \frac{y(t-\tau)}{1 + y(t-\tau)^n} + b \cdot y(t) \quad (5)$$

$$u(t) = \sum_{i=1}^{m} \sin(\varphi_i \cdot t) \quad (6)$$

La secuencia MG17 utilizó los siguientes parámetros: $a = 0.2, n = 10, b = -0.1, \tau = 17$. Se trata de una ecuación diferencial que se solucionó usando Runge-Kutta de orden 4.

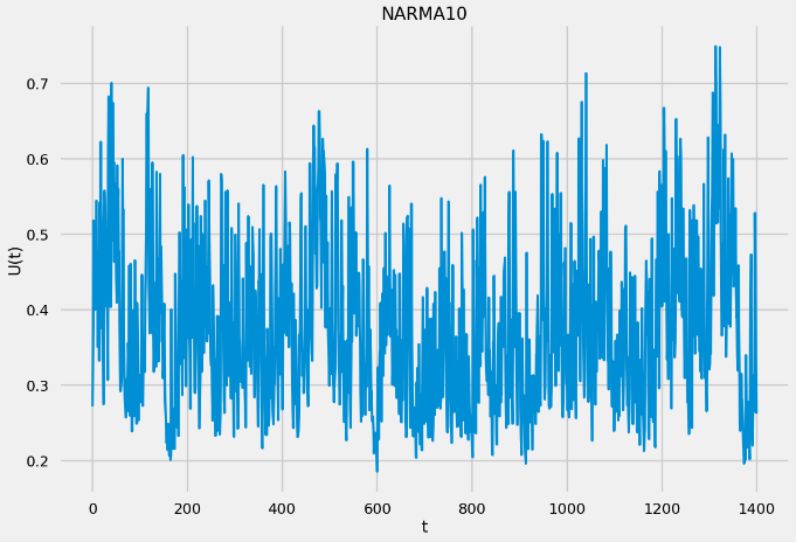
La secuencia MS12 utilizó los siguientes parámetros: $m = 12$ y las siguientes frecuencias φ_i:

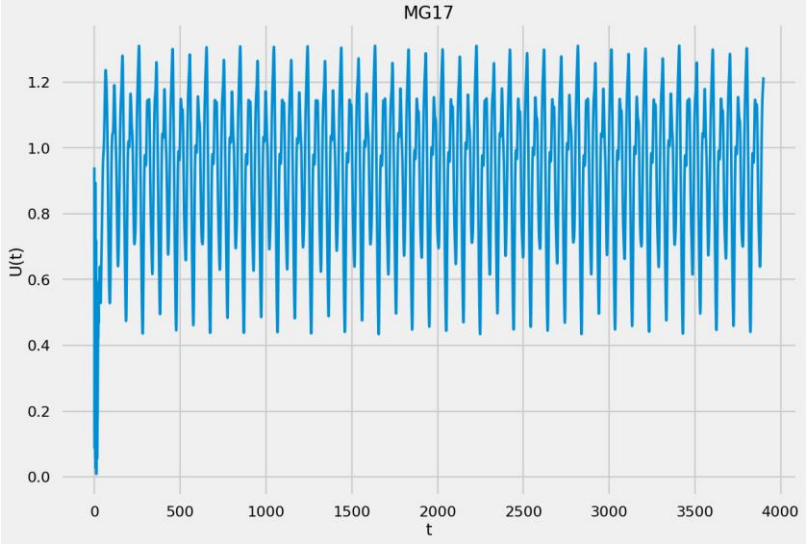
$$\varphi_1 = 0.2, \varphi_2 = 0.331, \varphi_3 = 0.42, \varphi_4 = 0.51, \varphi_5 = 0.63, \varphi_6 = 0.74,$$

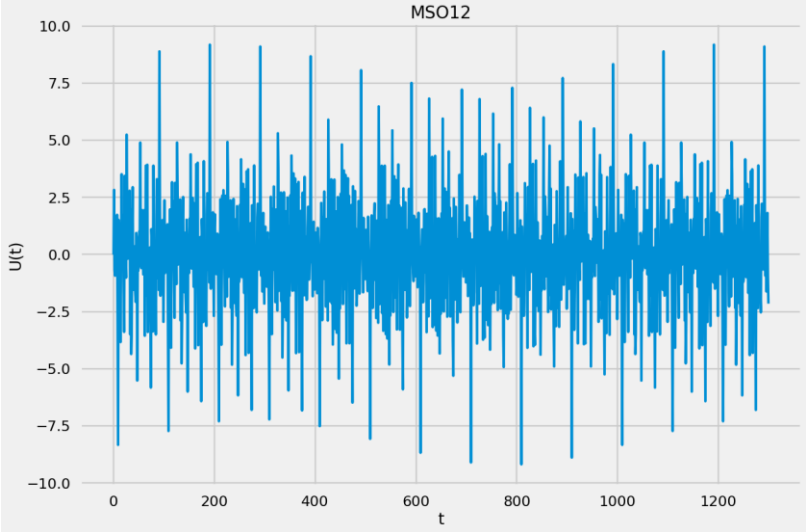
$$\varphi_7 = 0.85, \varphi_8 = 0.97, \varphi_9 = 1.08, \varphi_{10} = 1.19, \varphi_{11} = 1.27, \varphi_{12} = 1.32$$

Estas frecuencias fueron tomadas de [3].

A continuación, se presentan ejemplos de las secuencias generadas:







Posteriormente se implementó el algoritmo adaptativo propuesto por los autores. Las ecuaciones (1) y (2) se resuelven usando el método de Euler para ecuaciones diferenciales, obteniendo un valor nuevo de la solución en cada iteración. Una vez implementado se probó el algoritmo con unos parámetros fijos propuestos [2]. Estos parámetros son: radio espectral $\rho = 2.0$, density of $K = 0.05, \beta = \frac{\pi}{2}$, $\lambda = 4.0$, $\epsilon = 0.1$, $h = \Delta t = 1.0$.

Las funciones que componen el algoritmo son: generate_K_matrix, rescale_spectral_radius, generate_W_matrix, (que es en realidad un vector), univariate_ridge_regression, Euler_iteration (para resolver la ecuación (1)), Euler_matrix_iteration (para resolver la ecuación (2)) y train que es la función principal. También se tienen tres funciones auxiliares: MSE, kuramoto_synchrony y memory_capacity. Todos los detalles de la implementación, así como los resultados obtenidos tras ejecutar los experimentos están disponibles en el repositorio de GitHub: este6an13/rc-kuramoto-sad-rc (github.com).

Se realizó una ejecución con estos parámetros para cada una de las tres secuencias. Al final del entrenamiento se calculó el error cuadrático medio (MSE), se graficó la predicción obtenida por el algoritmo para los datos de prueba. Se calculó el coeficiente de sincronía r y se graficaron la matriz K resultante y las fases de los osciladores. Se usaron $N = 10$ osciladores para no incrementar significativamente el tiempo computacional.

Los valores para $L_{\text{adev}}, L_{\text{train}}$ y L_{test} fueron tomados de [2]. La longitud de la serie de tiempo se calculó como $n = L_{\text{train}} + L_{\text{test}}$. La siguiente tabla muestra los valores utilizados:

Serie de tiempo	L_{adev}	L_{train}	L_{test}	λ
NARMA10	100	900	500	4.0
MG17	100	2900	1000	1.0
MSO12	100	1200	100	4.0

En el código que acompaña a este proyecto se denomina a L_{adev} como a, a L_{train} como b. Como $n = L_{\text{train}} + L_{\text{test}}$, basta con definir a n para obtener a L_{test}.

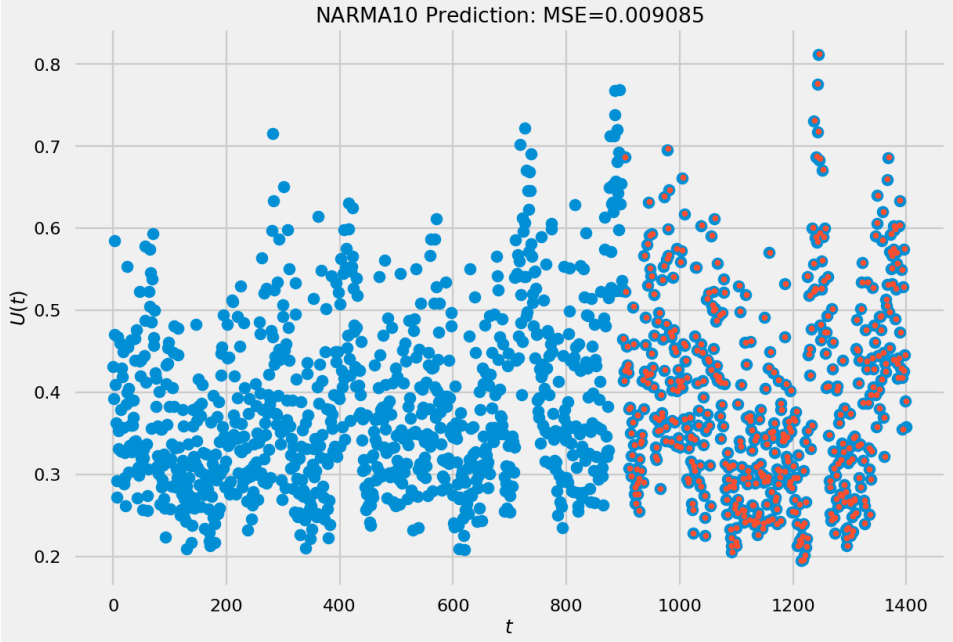
Se inicializaron las fases Θ como una matriz de ceros de tamaño $N \times n$. En realidad, no era necesario construir una matriz de este tamaño sino únicamente un vector de tamaño N pues solo se requieren las fases del tiempo anterior para resolver los sistemas. Sin embargo, se decidió usar la matriz para poder graficar los osciladores y ver si se sincronizaban. Las frecuencias naturales de cada oscilador se obtuvieron de una distribución normal estándar tal como se sugiere en [2].

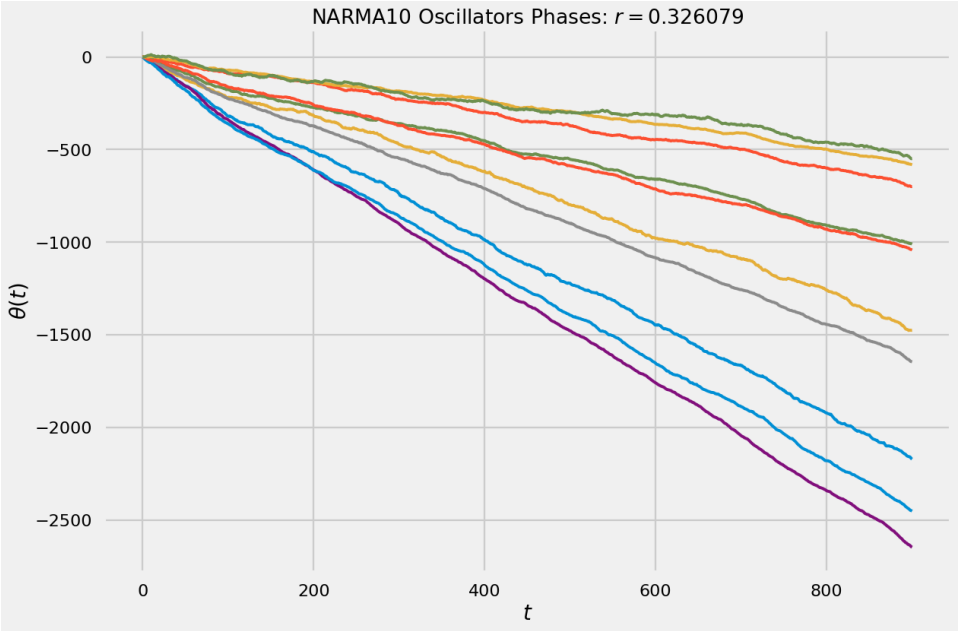
Una vez se ejecutaron estos experimentos, se intentaron replicar las gráficas del artículo. En particular, se intentó replicar las siguientes gráficas: MSE vs. λ vs. ρ, capacidad de memoria MC vs. λ vs. ρ, MSE vs. Dispersión de K (complemento de la densidad), MSE vs. β y los reservorios como redes para tres valores de β.

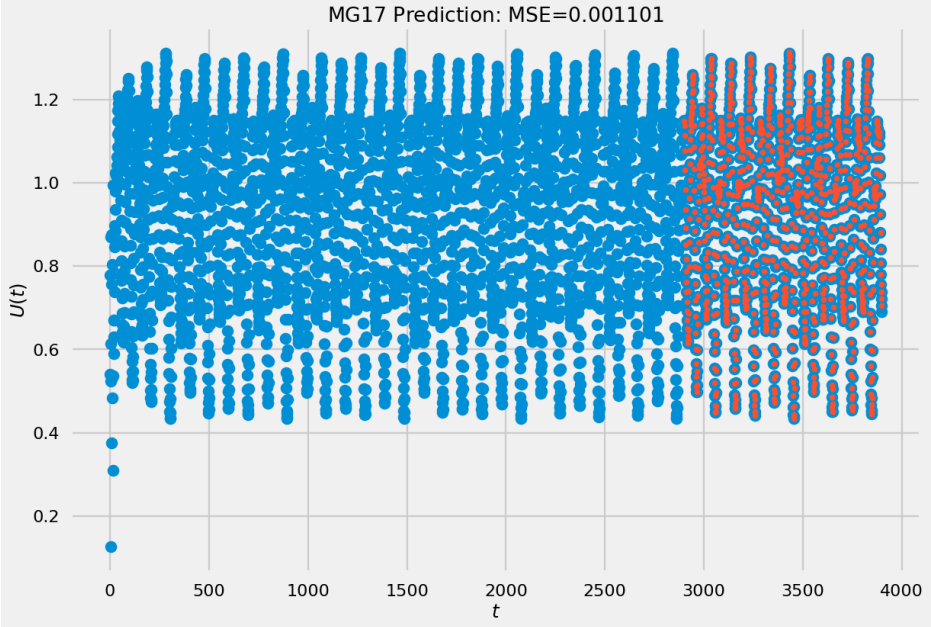
La siguiente sección de resultados presenta los resultados obtenidos tras implementar la metodología descrita.

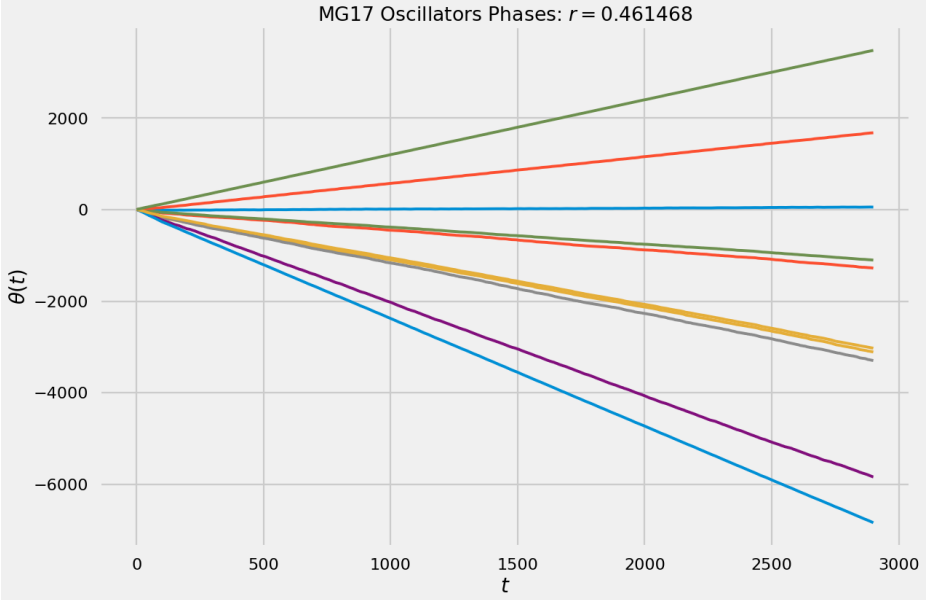
Resultados:

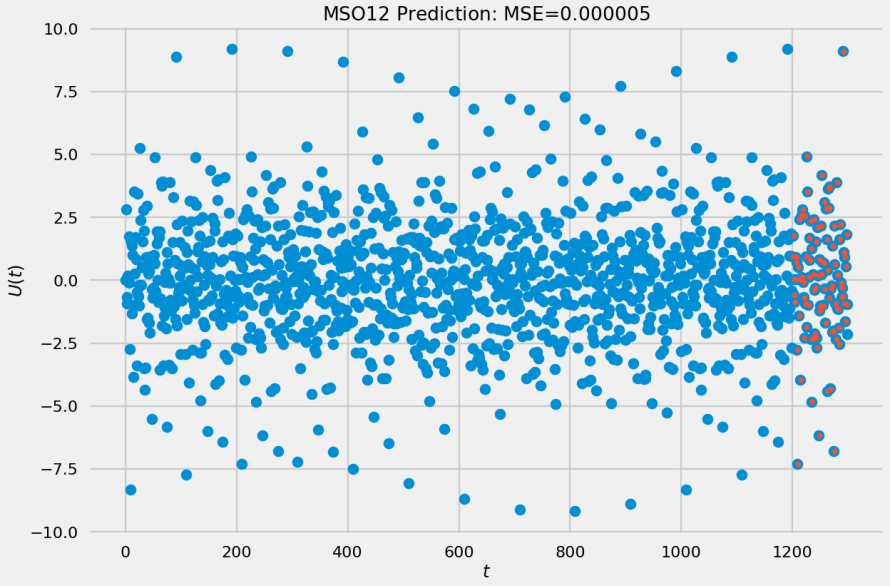
A continuación, se muestran las gráficas de la predicción para cada una de las series de tiempo, tras ejecutar el algoritmo con los parámetros fijos según se describió en la metodología. También se presentan las gráficas de los osciladores. Se observa que en ninguno de los casos se logra una sincronización de los osciladores. El coeficiente de sincronía no se aproxima a 1.0 en ninguna de estas ejecuciones. Sin embargo, se obtienen resultados aceptables para el MSE en cada ejecución.

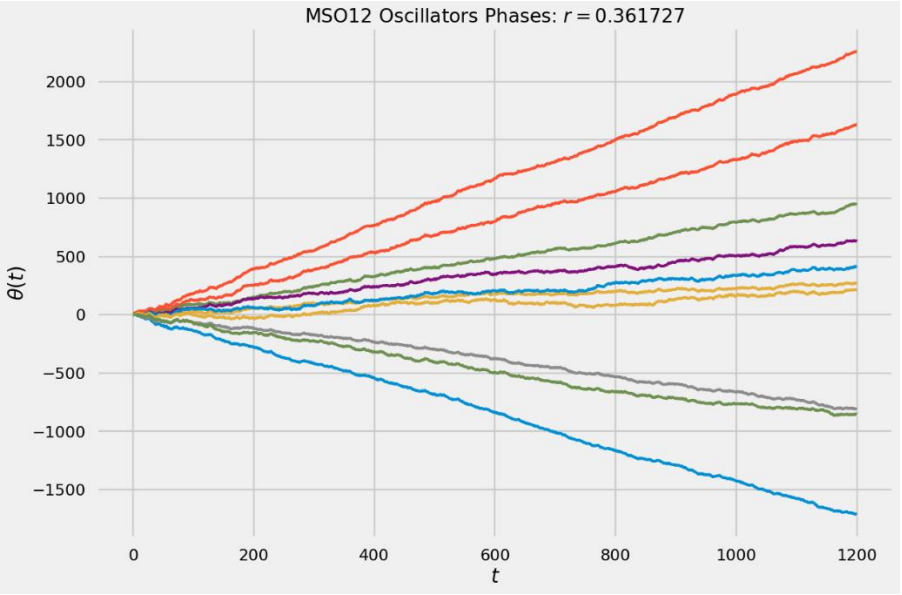




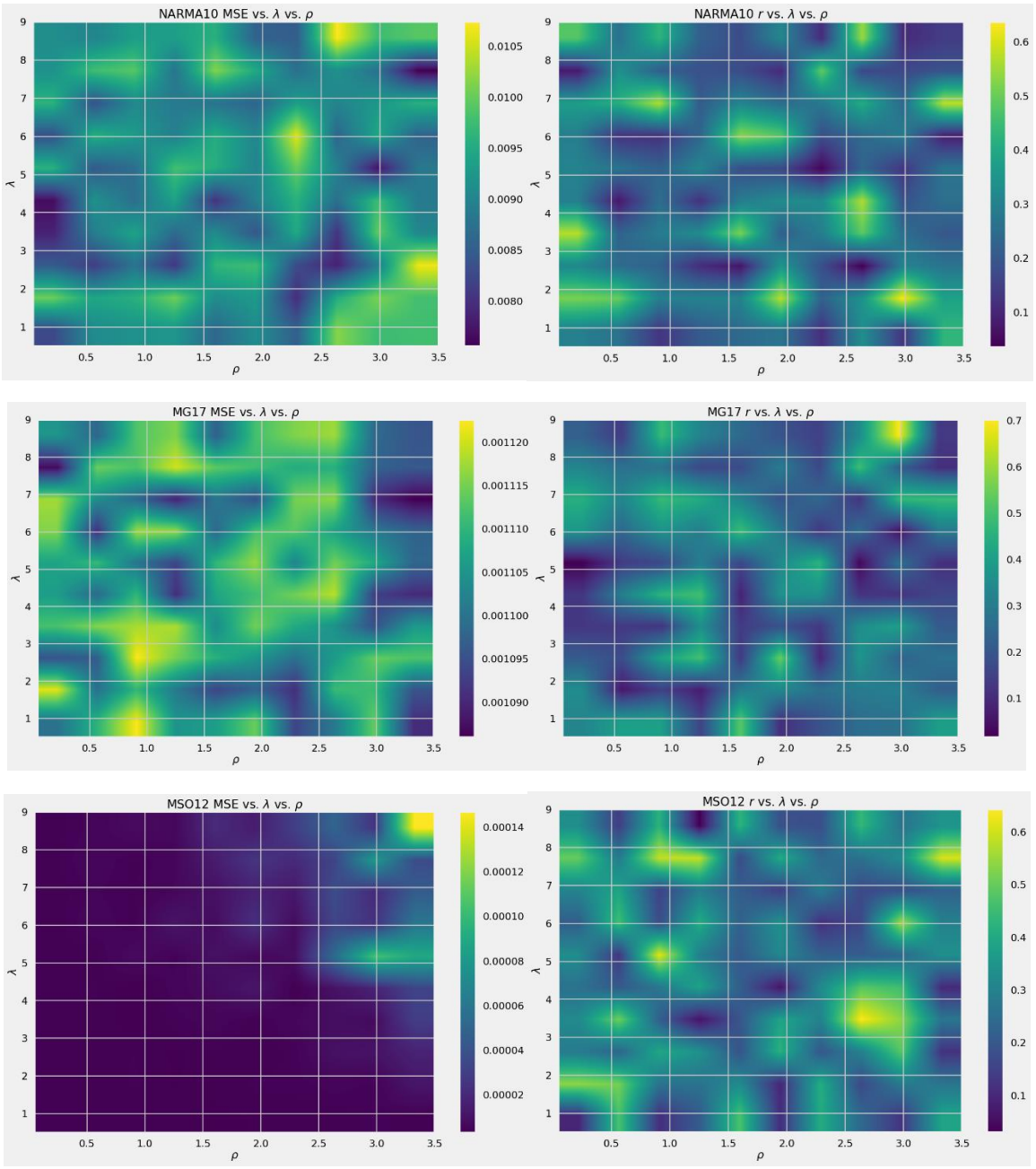

MG17 Prediction: MSE=0.001101
1.2
1.0
0.8
0.6
0.4
0.2
U(t)
0
500
1000
1500
2000
2500
3000
3500
4000
t


MG17 Oscillators Phases: r = 0.461468
2000
0
−2000
−4000
−6000
θ(t)
0
500
1000
1500
2000
2500
3000
t



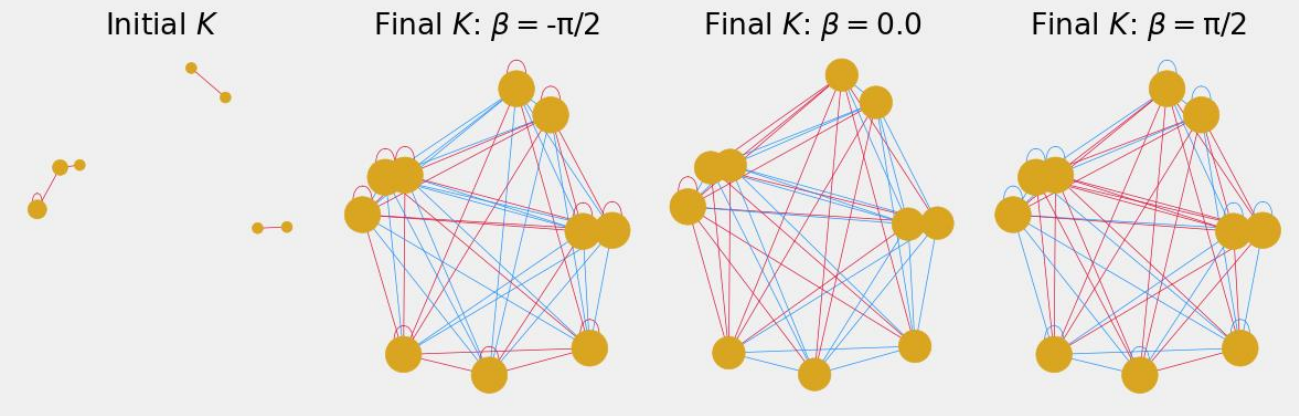


Ahora se presentan las gráficas de MSE vs. ρ vs. λ y r vs. ρ vs. λ. Estas gráficas pretenden mostrar la relación que existiría entre la sincronización de los osciladores y el desempeño en la predicción del algoritmo. Se obtuvieron ejecutando el algoritmo para diferentes combinaciones del radio espectral ρ y la fuerza de acoplamiento λ. En [2] se observa claramente esta relación al observar patrones muy similares entre ambas gráficas para cada una de las series de tiempo. Desafortunadamente, durante la ejecución de estos experimentos, no se consiguió replicar estos resultados. A continuación, se muestran las gráficas obtenidas.



De estas gráficas la que más se parece a las que se presentan en el artículo es la de MSE para la serie de tiempo MSO12. Sin embargo, se observa que la gráfica de r para esta misma serie no muestra el patrón de sincronía deseado. En el código que acompaña este trabajo se pueden encontrar las gráficas de capacidad de memoria, las cuáles tampoco muestran el comportamiento esperado, el cuál debería ser decreciente a medida que el radio espectral y la fuerza de acoplamiento crecen. Allí también se encuentran las gráficas de MSE vs. Dispersión de la matriz K, las cuales no muestran la tendencia decreciente que se presenta en el artículo. De igual manera ocurre con las gráficas de MSE y coeficiente de sincronía vs. β.

Finalmente se graficó el reservorio como una red antes y después del entrenamiento para tres valores de β. De manera consistente de acuerdo con los resultados obtenidos hasta el momento en este trabajo, el valor de β no parece incidir significativamente, más que en el signo de las sinapsis. Las sinapsis inhibitorias (negativas) se graficaron en color azul, mientras que las excitatorias, en color rojo. Obsérvese cómo la matriz inicia muy dispersa pues se usó una densidad de apenas 5% en estos experimentos. Pero termina como una matriz densa y en un grafo completo.



Conclusiones:

Es necesario investigar la razón por la cual los resultados obtenidos no coinciden completamente con el artículo [2]. Se plantea la posibilidad de que el mecanismo de adaptación, específicamente la actualización de la matriz K en cada iteración, no esté implementado correctamente, lo que podría afectar la sincronización de los osciladores. También se cuestiona si el método de solución utilizado, basado en una iteración por iteración, está funcionando adecuadamente.

Sería valioso explorar las implementaciones de bibliotecas de Python, como scipy, para verificar si es posible modificarlas y adaptarlas a este algoritmo. Dado que la función de la ecuación diferencial recibe múltiples parámetros, incluyendo aquellos que cambian en cada iteración, como la entrada de la serie de tiempo y la matriz K misma, se requiere una cuidadosa manipulación. Además, se plantea la posibilidad de que la forma en que se actualiza el radio espectral no esté implementada correctamente.

Es fundamental investigar y verificar la implementación del algoritmo, ya que existe la preocupación de que, durante el ajuste, los valores de la matriz puedan exceder los límites de [-1,1], sin que los autores del artículo mencionen cómo manejar esta situación. Se realizará una revisión exhaustiva de la literatura relacionada con la computación de reservorios para complementar la implementación realizada.

Se espera la aprobación y corrección del artículo por parte de los autores, ya que se han identificado ciertas impresiones de estilo y errores ortográficos, así como algunas lagunas en el contenido. Además, se investigará si los autores planean publicar el código utilizado en algún momento. Hasta la fecha, no se encuentra disponible, al igual que los videos referenciados en el artículo.

Finalmente, se continuará investigando este fascinante tema relacionado con el aprendizaje de máquina inspirado en la física. Este enfoque resulta innovador y ha demostrado resultados prometedores, como se pudo observar en las predicciones realizadas por el modelo en este trabajo.

Referencias:

[1] DNN - Tutorial 2 Part I: Physics inspired Machine Learning - https://uvadlc-notebooks.readthedocs.io/

[2] Z Zuo, Z. Gan, Y. Fan, V. Bobrovs, X. Pang, O. Ozolins. “Self-Evolutionary Reservoir Computer Based on Kuramoto Model” (2023).

[3] Ziqiang Li, Gouhei Tanaka. “Multi-reservoir echo state networks with sequence resampling for nonlinear time-series prediction” (2022).