DESARROLLO DE LA ESTRATEGIA iMO-NMPC: PRIMEROS PASOS PARA SU IMPLEMENTACIÓN EN DISPOSITIVOS INDUSTRIALES

A. Zabaljauregi, A. Alonso, M. Larrea, E. Irigoyen
EIB, UPV/EHU, {azabaljauregi001,aalonso198}@ikasle.ehu.eus,{m.larrea,eloy.irigoyen}@ehu.eus

Resumen

Este trabajo presenta la metodología que se está empleando, dentro del grupo de investigación de control inteligente (GICI) de la UPV/EHU, para el desarrollo de estrategias de control inteligente y su posterior implementación en plataformas de tiempo real. Mediante esta metodología se establece el procedimiento para la validación de dichas estrategias, no solamente desde el punto de vista de simulación, en forma de scripts, sino acercando estos desarrollos a diferente hardware industrial para su posterior implementación en tiempo real en forma de s-functions. El caso de uso que se presenta y que está siendo implementado actualmente es el de la estrategia iMO-NMPC, la cual integra bajo un esquema de control predictivo, algoritmos evolutivos para la optimización y redes neuronales para el modelado de sistemas. Esta metodología se desarrolla haciendo uso de la plataforma de simulación MATLAB/Simulink®. Estos desarrollos se podrán validar en hardware industrial, para lo cual se empleará la generación automática de código que proporciona dicha herramienta de simulación.

Palabras clave: iMO-NMPC, hardware industrial, *S-Function*, dinámicas complejas

1. INTRODUCCIÓN

En el mundo de la ingeniería existen muchos sistemas que presentan dinámicas complejas: procesos químicos, aplicaciones robóticas, vehículos aéreos, etc. Estos sistemas debido a su naturaleza no lineal, o a incertidumbres en su comportamiento, o al acoplamiento entre sus variables, u otras características que hacen compleja la tarea de diseñar un sistema de control específico, no son fácilmente controlables con técnicas de control clásicas. Adicionalmente, una linealización sobre un punto de operación tampoco basta para un control satisfactorio. Asimismo, frecuentemente la tarea de identificación y modelización del sistema a controlar, en base a un estudio matemático apoyado en las leyes que gobiernan el comportamiento de sus diferentes componentes, se ve dificultada al no tener acceso directo al mismo, por la gran cantidad de parámetros y limitaciones técnicas, o simplemente por razones económicas. Hoy en día, más que nunca, surge la necesidad de identificar y controlar estos sistemas con precisión debido a las especificaciones de fabricación y medidas de seguridad más estrictas. Una de las estrategias más empleadas en la industria es el Control Predictivo basado en Modelo (del inglés, MPC) [3] y su extensión para sistemas no lineales (NMPC) [2]; algunos ejemplos aplicados recientemente [18, 19, 16, 14, 6].

El desarrollo de nuevas estrategias de control necesita de un entorno de simulación suficientemente versátil como para abarcar el ciclo de desarrollo de los mismos, esto es; desde simulaciones básicas a pruebas de ejecución en plataformas hardware industriales. Existes técnicas como el diseño basado en modelo que contemplan el desarrollo de una estrategia de control en todas estas fases [1, 13, 21] que están apoyadas en herramientas asistidas por computadora.

El trabajo desarrollado y presentado en este documento se está realizando en el contexto de una novedosa estrategia en el ámbito del control predictivo no lineal denominada iMO-NMPC (*Intelligent Multi-objective Nonlinear Model Predictive Control*) [17]. Este trabajo da continuación a anteriores desarrollos realizados en esta línea [8] [20]. En este documento se presenta una evolución de los pasos dados en un esfuerzo de implementar esta técnica en una plataforma de simulación como MATLAB/ Simulink ®, la cual proporciona la posibilidad de implementar los desarrollos en distintas fases del diseño de controladores.

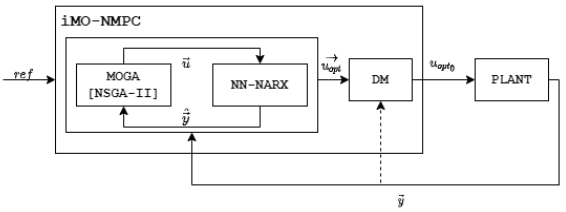


Figura 1: Esquema iMO-NMPC

https://doi.org/10.17979/spudc.9788497498418.0248

2. DESARROLLO

La estrategia de control inteligente iMO-NMPC propuesta en [17], la cual queda representada en la Figura 1, integra una base de MPC, paradigmas procedentes del ámbito de la Computación Inteligente como son los algoritmos evolutivos, redes neuronales artificiales (RNA) para la representación del modelo del sistema a controlar [7] y un *Decision Maker* que da soporte a la etapa de optimización. Mediante la propuesta aportada en este trabajo se pretende abordar el control de procesos industriales no lineales sujetos a perturbaciones, con complejas dinámicas acopladas y presencia de varios objetivos contrapuestos. Un primer paso en el desarrollo de este trabajo es llegar a comprender la función de cada componente de dicha estrategia. Dichos componentes quedan representados en el esquema del funcionamiento general de esta técnica en la Figura 1. De forma resumida, su funcionamiento general es el siguiente: el MOGA (*Multi-objective Genetic Algorithm*) (en el caso propuesto NSGA-II [5]), genera una población de individuos conformados por H_c cromosomas, es decir el número de acciones de control a aplicar en los instantes del horizonte de control, que se aplican al modelo neuronal de la planta, extendiendo la última acción de control hasta H_p, el horizonte de predicción. Estos individuos se evalúan según unos objetivos definidos (e.g el error $(ref - y)$, la variación de la acción de control Δu , etc.) y según su coste (y *crowding index* en NSGA-II) se clasificarán y se emplearán para obtener la siguiente generación mediante operadores genéticos de mutación y *crossover*. Mediante este proceso genético, a lo largo de las iteraciones, se minimizará el coste de esos objetivos. Los individuos dominantes que forman el frente de Pareto se entregan después al *Decision Maker*, para que según el contexto en el que se encuentre la planta elija una u otra solución partiendo del conocimiento de un agente experto. De la secuencia de control seleccionada se aplica solamente la primera acción y se vuelve a realizar la optimización iterada en el siguiente instante de muestreo una vez se realimenta la salida de la planta real.

2.1. APRENDIZAJE DE LAS HERRAMIENTAS DE *Simulink*

Se ha optado por MATLAB/Simulink para el desarrollo y simulación de el sistema de control propuesto, al ser un software de computación numérica y prototipado de algoritmos ampliamente utilizado en el ámbito de la ingeniería de control. Simulink, herramienta de modelización visual por bloques, cuenta con bloques de llamada a *S-Functions*, que permiten al usuario programar rutinas personalizadas para su ejecución en conjunción al resto de bloques propietarios. Se ha considerado emplear esta funcionalidad para llevar a cabo la tarea de implementar la estrategia, no solo porque otorga al usuario una mayor flexibilidad a la hora de programar, sino porque bajo condiciones específicas, permite generar el código de la simulación para su posterior ejecución en plataformas de tiempo real.

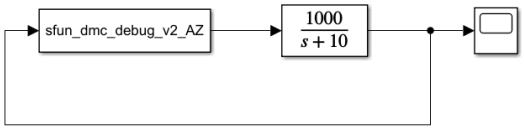
Simulink cuenta con varias formas de desarrollar estas *S-Functions* o funciones especiales, sin embargo el principal detalle a tener en cuenta es el lenguaje de programación; se pueden escribir dichas funciones de las siguientes maneras:

1. *Level 1 MATLAB S-Functions*: Una forma muy simple y limitada de programar diferentes *S-Functions* mediante *Flags* utilizando lenguaje de MATLAB. Pronto será obsoleto y desechado.
2. *Level 2 MATLAB S-Functions*: Proporciona acceso a un conjunto más amplio de la API de las *S-Functions* y admite generación de código. En esta opción también se emplea el lenguaje de programación de MATLAB. Para habilitar la generación de código, el usuario debe proveer un archivo `.tlc` (*Target Language Compiler*), donde se especifican las pautas de generación.
3. *MEX S-Functions*: Permite implementar algoritmos en formato C MEX *S-Functions* o programar *wrappers* como llamada a código ya existente escrito en *C,C++* o *Fortran*. En este caso, proveer un archivo `.tlc` para la generación de código es opcional.

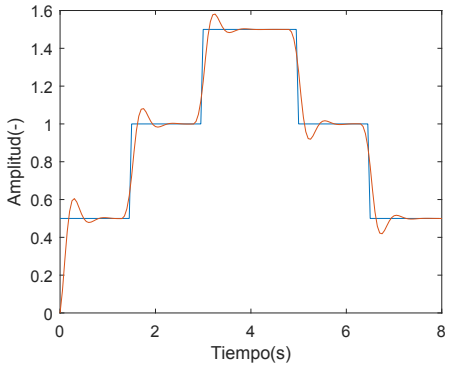
Se puede encontrar más información sobre el tema en la documentación proporcionada por MathWorks. Por razones de eficiencia en el flujo de desarrollo de los algoritmos y la posterior generación automática de los programas ejecutables, se ha optado por emplear la tercera alternativa. En el proceso de aprendizaje del API, se han implementado bloques de *S-Function* como controladores PI, PID y funciones de transferencia entre otros. Un desarrollo que vale la pena destacar es un control DMC (*Dynamic Matrix Control*) Figura 2, como primer paso a dar en la programación del control predictivo antes de afrontar la programación del iMO-NMPC. En dicha figura, la referencia que se emplea está establecida de antemano y no es necesaria introducirla externamente al bloque *sfun_dmc_debug_v2_AZ*.

En la Figura 2 se muestra también el resultado del control que realiza. Estos resultados deben tomarse como fases de aprendizaje del desarrollador, en

https://doi.org/10.17979/spudc.9788497498418.0248



(a) Esquema de bloques para DMC en Simulink



(b) Salida del sistema y referencia

Figura 2: Modelo de prueba DMC en Simulink

el marco del desarrollo de posteriores estrategias más complejas.

2.2. ESTUDIO DEL ALGORITMO MEDIANTE *scripts* DE MATLAB

Antes de pasar a la implementación en C, se ha estudiado el comportamiento de la estrategia iMO-NMPC mediante *scripts* de MATLAB, con diferentes modelos no-lineales como *benchmark*. Los modelos en cuestión han sido aplicados anteriormente en [11] y [10]. Las ecuaciones de los modelos no lineales empleados son las siguientes:

$$\begin{aligned} y_1(k+1) = \frac{1{,}5 \cdot y_1(k) \cdot y_1(k-1)}{1 + y_1(k)^2 + y_1(k-1)^2} + \\ +0{,}7 \sin\left[0{,}5\left(y_1(k) + y_1(k-1)\right)\right] \cdot \\ \cdot \cos\left[0{,}5\left(y_1(k) + y_1(k-1)\right)\right] + 1{,}2u_1(k) \end{aligned} \quad (1)$$

$$y_2(k+1) = u_2(k)^3 + \frac{y_2(k)}{1 + y_2(k)^2} \quad (2)$$

El script desarrollado reproduce los siguientes pasos expuestos esquemáticamente de la estrategia iMO-NMPC:

- Paso 1: Lectura del estado actual de la simulación.
- Paso 2: Generación de cromosomas (acciones de control).
- Paso 3: Proceso de optimización con predicciones de la RNA.
- Paso 4: Selección de acción de control a aplicar.
- Paso 5: Aplicar la acción de control y volver al Paso 1.

Para el apartado de la optimización (Paso 3) se ha hecho uso del algoritmo genético multiobjetivo `@gamultiobj` propietario de MATLAB. El `@gamultiobj` proporciona un frente de Pareto con múltiples soluciones entre las que se debe seleccionar la más adecuada.

El criterio empleado en este trabajo para la selección de la acción de control más adecuada (Paso 4), es el criterio de distancia mínima. Este criterio consiste en seleccionar aquella solución cuya distancia al origen de los ejes del frente de Pareto sea mínima.

En el contexto del control predictivo, como modelos de predicción, se han utilizado tanto modelos matemáticos inicialmente, como redes neuronales entrenadas para representar las ecuaciones (1) y (2). Las primeras sirven como referencia para comparar la bondad de la estrategia cuando se emplean modelos identificados. Una profundización sobre el empleo de redes neuronales para la identificación de dinámicas no lineales se presenta en [9]. También se han realizado pruebas con una red neuronal MIMO para representar ambas ecuaciones.

Por otro lado, en un esfuerzo de acelerar los tiempos de ejecución y gracias a que la función `@gamultiobj` permite una entrada vectorizada, se ha modificado el *script* existente para evaluar vectorialmente las acciones de control (cromosomas) de todos los individuos de la población en cada iteración y para cada instante de muestreo. Se muestra en la Figura 3, la evolución de las salidas de ambos sistemas bajo control, estando los sistemas representados por una misma red neuronal empleando un horizonte de control y predicción $H_c = H_p = 4$, 200 individuos de población y un tiempo de muestreo de 0.5 s. El uso del tiempo de muestreo en el script sirve como limitador del tiempo de búsqueda que se le permite al optimizador. Debido al tiempo de muestreo reducido y al equipo donde se ha realizado la simulación, en cada instante solo se llega a 30 iteraciones en la búsqueda de la acción de control optimizada. De la misma manera, cuanto mayor poder de procesamiento tenga el dispositivo donde se realice el cálculo permitirá gestionar poblaciones de individuos más grandes y un número de iteraciones superior. No obstante, se puede observar que se obtiene un resultado satisfactorio de control.

https://doi.org/10.17979/spudc.9788497498418.0248

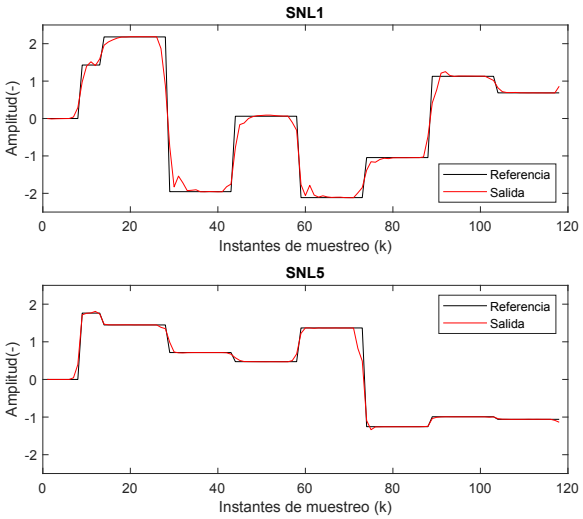


Figura 3: Evolución del control de los sistemas SNL5 (1) y SNL1 (2)

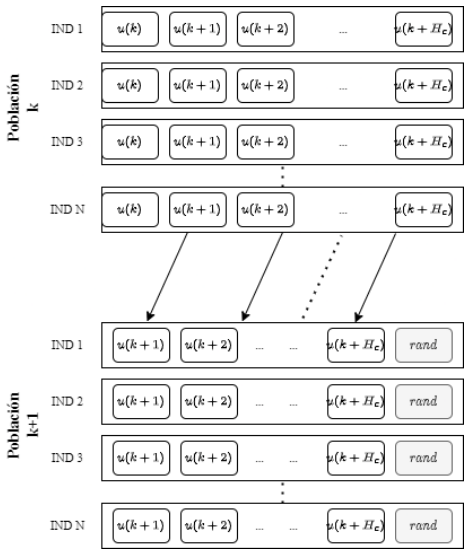


Figura 4: Esquema inicialización de individuos de la siguiente población

Se debe añadir en este apartado también, una de las pruebas heurísticas realizadas en la gestión de la inicialización de la población del siguiente instante de muestreo. Para inicializar los cromosomas de los individuos de la siguiente población se emplean los calculados para la población actual desplazados en un instante de muestreo tal y como se aprecia en la Figura 4. Para asignar un valor al último cromosoma de los individuos, simplemente se llama a una función que genera un número aleatorio entre los límites inferior y superior de la señal de control. Pese a que no se han realizado pruebas exhaustivas del efecto de esta inicialización, la técnica demuestra consistentemente reducir el tiempo de cómputo del control predictivo, aún siendo una reducción ligera, cuando se han realizado ejecuciones sin límite de tiempo en cada ciclo de iteración.

2.3. IMPLEMENTACIÓN EN *CMEX S-Function*

Finalmente, tras haber dado los pasos anteriores se expone en este apartado el desarrollo hasta el momento de la implementación en *S-Function* de iMO-NMPC. Partiendo del código de NSGA-II de Kalyanmoy Deb y ulteriores avances partiendo de [11], se han creado archivos adicionales, para encapsular ahí la lógica necesaria para el modelo de predicción y los criterios de selección, a parte del archivo *S-Function* principal, el cual es llamado por la rutina principal de simulación en Simulink mostrada en la Figura 5 (fuente: [15]). La lógica principal de la estrategia se divide en los pasos de la rutina recuadrados en azul. En el paso `mdlStart` se realiza la adjudicación de memoria, definición y asignación de variables; en el paso `mdlOutputs` se actualiza la salida con la acción de control seleccionada; finalmente en `mdlUpdate` se realiza la optimización para encontrar las soluciones del siguiente instante.

En la Figura 6 se muestran las respuestas de los sistemas (1) y (2) para $H_c = 4$, $H_p = 6$, una población de 200, número de generaciones 200 y con un tiempo de muestreo de 0.1 s. En este caso el tiempo de muestreo no limita el número de iteraciones realizadas, solamente sirve para dar una escala de tiempo a la simulación. Para ambos sistemas, se ha empleando su correspondiente modelo matemático para realizar las predicciones. Dichos resultados serán empleados como validación de posteriores desarrollos en los que se integrarán redes neuronales para la realización de las predicciones.

3. CONCLUSIONES

En este trabajo se han presentado los primeros pasos metodológicos para el desarrollo y validación de la estrategia de control inteligente iMO-NMPC empleando técnicas de diseño basada en modelo. Dichos pasos van orientados a la simulación en script, al diseño de los bloques de simulink con s-functions y por último a la validación de la estrategia en simulink. A través de este trabajo se han presentado de forma detallada los pasos y pruebas realizadas para la consecución del objetivo que es el análisis de la viabilidad del código desarrollado que posteriormente pueda ser implementado en plataformas hardware industriales.

Los resultados preliminares se han realizado sobre la estrategia de control iMO-NMPC y se han

https://doi.org/10.17979/spudc.9788497498418.0248

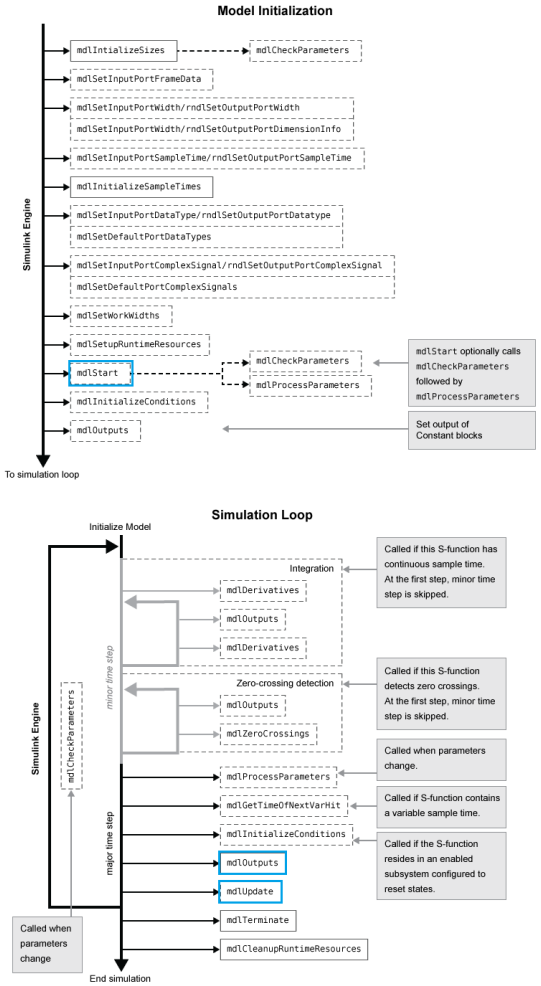
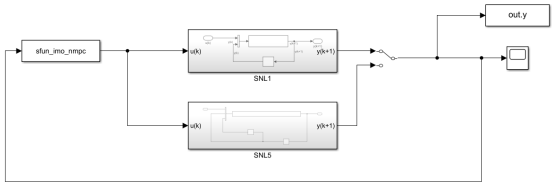


Figura 5: Interacción del Motor de Simulink con C S-Functions (fuente: Mathworks®)

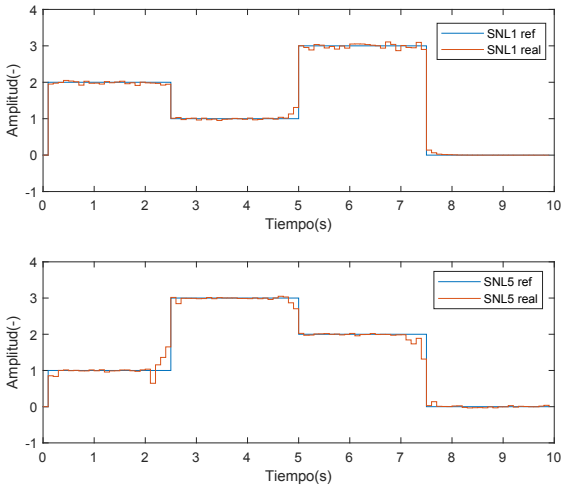
llevado a cabo simulaciones en base a scripts de matlab. La figura 3 refleja un control adecuado para los sistemas propuestos. Tras esta validación inicial, se procede al desarrollo de la s-function / CMEX cuyos resultados se pueden ver en la figura 6(b). Se puede apreciar que el control sigue siendo adecuado pese a reducir y acotar los tiempos de computo del algoritmo de búsqueda.

Un desarrollo clave en el futuro es el traslado de la ejecución a una plataforma de tiempo real para una validación y experimentación adicional, similar a lo realizado en [12]. Para esta labor Simulink cuenta con aplicaciones como *Simulink Desktop Real Time Simulink Real Time* o *Simulink PLC Coder*.

En un esfuerzo de asegurar la concurrencia del software, facilitar la depuración del código y aseguramiento de la periodicidad del cómputo en cada instante de muestreo, se plantea introducir una ejecución similar a una máquina de estados con algún mecanismo de *timeout* en la optimización



(a) Esquema de bloques para iMO-NMPC *S-Function* en Simulink



(b) Salida del sistema y referencia

Figura 6: Modelo de prueba *S-Function* de iMO-NMPC

del algoritmo genético.

Por otro lado queda por extender la utilidad del programa a sistemas MIMO multi-objetivo. Adicionalmente queda como línea futura, la adición de diferentes lógicas de *Decision Maker*, a parte del criterio de distancia mínima implementado en el momento, que estén basadas en técnicas de *Soft Computing* tales como la lógica difusa. El método de inicialización de la población, mencionada anteriormente, queda por ser implementada también en la versión programada en C.

Finalmente, queda como posible desarrollo futuro el análisis y la experimentación con técnicas de optimización mas recientes, como puede el NSGA-III [4], una versión mejorada sobre el NSGA-II, o MOEA/D [22, 23] un algoritmo evolutivo multi-objetivo basado en descomposición y operadores diferenciales.

Agradecimientos

Este trabajo se ha desarrollado en el marco del proyecto PID2020-120087GB-C22 financiado por el Ministerio de Ciencia e Innovación del Gobierno de España.
(AEI / http://dx.doi.org/10.13039/501100011033)

English summary

https://doi.org/10.17979/spudc.9788497498418.0248

iMO-NMPC STRATEGY DEVELOPMENT: FIRST STEPS FOR IMPLEMENTATION ON INDUSTRIAL PLATFORMS

Abstract

This work presents the methodology used by the Intelligent Control Research Group (GICI) at UPV/EHU, for the development of intelligent control strategies and their further implementation in real time platforms. This methodology establishes the procedure for the validation of these strategies, not only from the point of view of simulation, in the form of scripts, but also by bringing these developments to different industrial hardware for their subsequent implementation in real time in the form of s-functions. The use case presented and currently being implemented is the iMO-NMPC strategy, which integrates under a predictive control scheme, evolutionary algorithms for optimisation and neural networks for system modelling. This methodology is developed using the MATLAB/Simulink® simulation platform. These developments can be validated on industrial hardware, for which the automatic code generation provided by this tool will be used.

Keywords: NMPC, Industrial Hardware, *S-Function*, Complex Dynamics.

Referencias

[1] A. Bemporad. Model predictive control design: New trends and tools. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 6678–6683, 2006.

[2] E. Camacho and C. Bordons. *Nonlinear Model Predictive Control: An Introductory Review*. 2007.

[3] E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer, 2nd ed. 2004. corr. 2nd printing edition, Apr. 2007.

[4] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[6] A. Flores-Tlacuahuac, P. Morales, and M. Rivera-Toledo. Multiobjective nonlinear model predictive control of a class of chemical reactors. *Industrial & Engineering Chemistry Research*, 51(17):5891–5899, 2012.

[7] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[8] E. Irigoyen, E. Larzabal, J. J. Valera, M. Larrea, and A. Gutierrez. Primeros resultados de un control genético predictivo sobre maqueta de helicóptero (twinrotor). *Jornadas de Automática*, 2014.

[9] S. Jagannathan and F. Lewis. Identification of nonlinear dynamical systems using multilayered neural networks. *Automatica*, 32(12):1707 – 1712, 1996.

[10] M. Larrea, E. Irigoyen, and V. Gómez. Adding nonlinear system dynamics to levenberg-marquardt algorithm for neural network control. In *ICANN 2010*, volume 6354 of *Lecture Notes in Computer Science*, pages 352–357. 2010.

[11] M. Larrea, E. Larzabal, E. Irigoyen, J. J. V. García, and M. Dendaluce. Implementation and testing of a soft computing based model predictive control on an industrial controller. *J. Appl. Log.*, 13:114–125, 2015.

[12] M. Larrea, E. Larzabal, E. Irigoyen, J. Valera, and M. Dendaluce. Implementation and testing of a soft computing based model predictive control on an industrial controller. *Journal of Applied Logic*, 2014.

[13] W.-J. Li, D.-S. Li, and J.-W. Zhang. Model-based design and experimental validation of control system for a three-level inverter. *Electronics*, 11(13), 2022.

[14] G. Marchante, A. Acosta, A. González, J. Zamarreño, and V. Álvarez. Comfort constraints evaluation in predictive controller for energy efficiency. *RIAI Journal*, 18(2):146–159, abr. 2021.

[15] Mathworks. *How the simulink engine interacts with c s-functions*. https://es.mathworks.com/help/simulink/sfg/how-the-simulink-engine-interacts-with-c-s-functions.html.

[16] M. Schaaf. Hybrid model predictive control of a gravity separator with intermittent product extraction. *RIAI Journal*, 17(3):318–328, jul. 2020.

[17] J. Valera, V. Gómez, E. Irigoyen, F. Artaza, and M. Larrea. Intelligent multiobjective nonlinear model predictive control (imo-nmpc): Towards the 'on-line' optimization of highly complex control problems. *Expert Systems with Applications*, 39(7):6527 – 6540, 2012.

[18] S. Vaneshani and H. Jazayeri-rad. Nonlinear control of a chemical plant employing a combination of fuzzy logic and particle swarm optimization techniques. 2011.

[19] G. Veselov, A. Sklyarov, and J. V. Chávez. Non-linear control of a tracked robot. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 641–646, 2019.

[20] K. Viana, M. Larrea, E. Irigoyen, M. Diez, and A. Zubizarreta. MIMO neural models for a twin-rotor platform: Comparison between mathematical simulations and real experiments. In Álvaro Herrero and et al., editors, *15th Int. Conf. SOCO'20, Burgos, Spain, 2020*, volume 1268 of *Advances in Intelligent Systems and Computing*, pages 407–417. Springer, 2020.

[21] D. Walica and P. Noskievič. Application of the mil and hil simulation techniques in stewart platform control development. *Applied Sciences*, 12(5), 2022.

[22] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[23] A. Zhou, Q. Zhang, and G. Zhang. A multiobjective evolutionary algorithm based on decomposition and probability model. pages 1–8, 06 2012.

 https://doi.org/10.17979/spudc.9788497498418.0248