XLIV Jornadas de Automática 2023

https://jautomatica.es

Simulación dinámica distribuida con adición de eventos

Gutiérrez, F.[a,b *], Mazaeda, R.[a,c], Zamarreño, J.M.[a,d]

[a] *Grupo de Control y Supervisión de Procesos. Universidad de Valladolid (UVa). Institute of Sustainable Processes. C/ Dr. Mergelina s/n, 47011 Valladolid*
[b] fgutrod@autom.uva.es, [c] rogelio.mazaeda@uva.es, [d] jmzamarreno@uva.es

To cite this article: Gutiérrez, F., Mazaeda, R., Zamarreño, J.M. 2023. A distributed simulation dynamic with events addition XLIV Jornadas de Automática, 405-410. https://doi.org/10.17979/spudc.9788497498609.405

Resumen

Este artículo describe un administrador de simulación dinámica distribuida, empleando lenguaje de programación orientada a objetos y una implementación posterior en PYTHON. El administrador. que utiliza el estándar de interoperabilidad OPC UA para comunicarse con los diferentes servidores que constituyen una simulación distribuida, actúa como cliente configurable. Los servidores OPC UA se implementan utilizando una herramienta de modelado de procesos y simulación externa. Para que dicha herramienta tenga un funcionamiento acorde a las necesidades de la simulación, debe disponer de alguna forma de control de la frecuencia de intercambio de datos a medida que el proceso evoluciona.

En la presente contribución se describe el desarrollo de una estrategia de adaptación de la frecuencia de intercambio de información entre las diferentes unidades de simulación mediante la adición de eventos. Mediante esta estrategia, se lleva a cabo la configuración de las condiciones de simulación, la medición del error y la adaptación del tiempo de intercambio, de manera análoga a los algoritmos de control de paso empleados tradicionalmente por algunos algoritmos de integración dinámica.

Palabras clave: simulación, OPC UA, intercambio, eventos, error.

A distributed simulation dynamic with events addition

Abstract

This article describes a distributed dynamic simulation manager, using object-oriented programming language and a further implementation in PYTHON. The administrator, which uses the OPC UA interoperability standard to communicate with the different servers that make up a distributed simulation, acts as a configurable client. OPC UA servers are implemented using an external simulation and process modeling tool. In order for this tool to function according to the needs of the simulation, it must have some form of control of the frequency of data exchange as the process evolves.

This contribution describes the development of a strategy to adapt the frequency of information exchange between the different simulation units by adding events. By means of this strategy, the configuration of the simulation conditions, the measurement of the error and the adaptation of the exchange time are carried out, analogously to the step control algorithms traditionally used by some dynamic integration algorithms.

Keywords: simulation, OPC UA, exchange, events, error.

1. Introducción

En la actualidad, el desarrollo de la computación a nivel industrial y su influencia imponen una tendencia hacia la automatización. El ajuste entre la relación del modelado y la simulación por ordenador está cambiando la concepción del diseño industrial, de tal forma que el modelo físico y digital tienden a la integración, obteniendo un sistema ciberfísico. Esta transformación queda enmarcada en el paradigma de Industria 4.0. En este contexto, surge el concepto, cuyo alcance aún está por definir, de gemelo digital. El principal objetivo es lograr un uso más amplio más allá de labores tradicionales de recopilación, supervisión y control de datos. En el diseño de un modelo matemático debe lograrse un equilibrio entre el nivel de detalle, con gran impacto a nivel computacional, y la rapidez en la ejecución.

*Autor para correspondencia: fgutrod@autom.uva.es

La simulación precisa y en un tiempo aceptable de grandes modelos dinámicos es un doble reto. Por un lado, es habitual disponer de diferentes subprocesos de simulación procedentes de diversos desarrolladores, con lo cual es necesario lograr un equilibrio entre un conjunto de unidades heterogéneas entre sí. Por otro lado, la versatilidad de estas unidades requiere de una serie de restricciones temporales durante la ejecución, que debe estar sincronizada, lo que se traduce en un elevado costo computacional.

Un posible enfoque es la co-simulación, llevando a cabo la división de un modelo complejo en varios submodelos. Existe un interés significativo en este campo, como se recoge en (Santos, 2005, 2008), (Sun, 2022), (Tamellin, 2022), (Sadjina, 2017), (Haid, 2022). Cada submodelo puede encontrarse en un equipo diferente, con la posibilidad de emplear diferentes algoritmos de integración numérica ejecutados a diferentes velocidades. Esta estrategia es factible si se dispone de submodelos desacoplados, pero en el contexto de una alta integración, es necesario poder intercambiar información regularmente. Al disponer de una estrategia de co-simulación, debe establecerse equilibrio entre velocidad y capacidad de respuesta, cuanto puede avanzar el tiempo en cada agente (h_{macro}) hasta que los requisitos de estabilidad numérica y precisión requieran de un intercambio de información.

Esta estrategia corresponde a una ampliación de (Mazaeda, 2023), logrando ajustar el avance de la simulación distribuida, mediante la modificación dinámica del tamaño de paso h_{macro} en tiempo real por parte de una unidad central que actúa como mánager de co-simulación. El ajuste se realiza teniendo en cuenta las posibles restricciones dinámicas, el error cometido, y para aumentar su versatilidad, se emplea la experiencia descrita en (Gutiérrez, 2022).

2. Descripción del problema

2.1. Estrategia y arquitectura propuesta

Una estrategia de co-simulación eficaz debe gestionar la precisión y la estabilidad numérica de la simulación distribuida, facilitando una ejecución rápida. Cada agente lleva a cabo una integración numérica propia mientras que el mánager coordina el proceso, controlando el error.

La integración numérica de sistemas dinámicos es un campo muy amplio (Cellier, 2006). El problema de valores iniciales (IVP), se resuelve actualizando de forma iterativa el estado de la simulación a partir de las condiciones iniciales. El tamaño del paso de integración, h_{micro}, y el orden de integración influyen en la precisión. Puesto que se emplean métodos iterativos explícitos con valores calculados exclusivamente a partir de valores anteriores, si h_{micro} es demasiado grande, pueden aparecer oscilaciones y problemas de estabilidad numérica, problema complejo que requiere de algoritmos implícitos de mayor coste computacional.

Una posible arquitectura para co-simulación aparece en la Figura 1. La integración numérica se realiza de forma independiente en cada unidad para el período de tiempo específico, h_{macro}. Después, el mánager recopila los datos e intercambia los valores requeridos. El proceso se repite para todo el horizonte de simulación.

Figura 1: Arquitectura de co-simulación propuesta

El mánager se enfrenta a los mismos problemas descritos, a nivel diferente y con menos información. En general, la única información disponible es la de los valores actuales y los de anteriores iteraciones. Tiene la capacidad de decidir si implementará un esquema de integración numérica implícita o explícita, el valor que se le dará a cada agente durante el siguiente ciclo de h_{macro} y su duración.

El empleo de una estrategia implícita requiere del restablecimiento de cada agente a un estado anterior y el posterior reinicio, sin adelantar tiempo de simulación. Como inconvenientes señalar que esta estrategia requiere de elevado coste computacional, y en no pocas ocasiones el agente no será susceptible de ser reiniciado. Como alternativas, se puede seguir una estrategia de mantenimiento de valores (ZOH) o el empleo de extrapolación.

El control del paso h_{macro} también es una alternativa viable, al existir diferentes dinámicas de simulación. Para periodos de dinámicas rápidas, se precisa h_{macro} pequeño para cumplir compromiso entre precisión y estabilidad, mientras que en dinámicas lentas basta un h_{macro} amplio. En simuladores monolíticos es posible la integración de algoritmos de diferente orden, como pueden ser por ejemplo DASPK o RK45 (Fehlberg, 1969), sin embargo, en co-simulación se requiere de una estimación del error al no poder solicitar el mánager la integración de un paso dos veces con órdenes diferentes.

2.2. Modelo empleado

Para ilustrar la idea principal, se utilizará un modelo eléctrico dinámico simple, mostrado en la Figura 2.

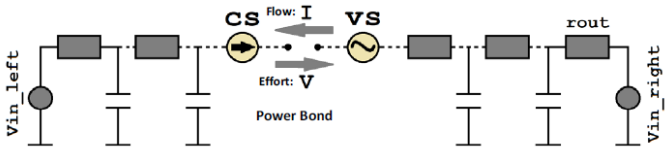

Figura 2: Modelo de ensayo.

El circuito consta de la unión de diez elementos, a los que se añaden dos resistencias adicionales y fuentes de corriente y tensión colocadas en ambos extremos. Se lleva a cabo una partición por la mitad, quedando integrados en una co-simulación. El modelo original monolítico completo también está disponible, pero solo como referencia. El modelado se lleva a cabo en EcosimPro/Proosis (Empresarios Agrupados, 2021). Se definen las clases de forma acausal, para ensamblar el modelo a través de puertos, concebidos como transmisores de potencia. Se dispone de un conjunto de ecuaciones para establecer la causalidad computacional. Posteriormente, es

posible exportar el modelo a servidores OPC UA, actuando como agentes de simulación distribuida.

La fuente de corriente y tensión en el lado izquierdo y derecho, se representan para indicar que la causalidad computacional de ambos submodelos debe ser compatible con las variables intercambiadas. En cada paso h_{macro}, el mánager lee la salida de tensión de la parte izquierda y la escribe en la fuente de tensión virtual de la derecha. De la misma manera, la corriente de la derecha se escribe en la fuente de corriente virtual de la parte izquierda.

El empleo de corriente y tensión como variables intercambiadas en el modelo propuesto se ha elegido a título de ejemplo, siendo extrapolable a otros sistemas físicos, como intercambio de variables en ambas direcciones, correspondientes a un flujo y/o esfuerzo.

2.3. *Efecto del tamaño de hmacro*

El esquema de intercambio variable discutido en la sección anterior corresponde a la estrategia ZOH. La simulación avanza asumiendo un error inevitable entre la simulación monolítica y la distribuida.

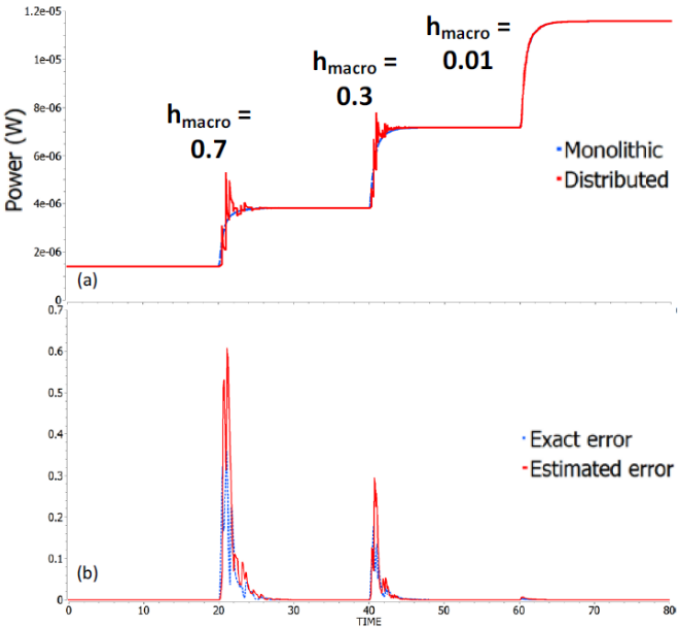

Figura 3: Efecto del tamaño de h_{macro} en el error. (a) Potencia transmitida entre monolítica y distribuida. (b) Error exacto frente al error estimado.

La Figura 3(a) muestra que esta diferencia se puede reducir hasta casi desaparecer con h_{macro} menores. La Figura 3(b) muestra la evolución del error en la transmisión de potencia a través de la frontera. El error exacto corresponde a la diferencia entre el valor correspondiente a una simulación y la solución analítica de las ecuaciones del modelo. El término error exacto corresponde aquí a una aproximación definida en (1) como la diferencia entre el valor de la potencia en la simulación monolítica, o de referencia (p_m), y distribuida (p_{dist}). No se tiene en cuenta el error intrínseco del método numérico de integración empleado. Dado que existe una discrepancia entre los valores de potencia (p_{dist}) calculados por las partes izquierda (p_{left}) y derecha (p_{right}), se toma su promedio.

$$error_{exact} = \left|\frac{p_m - p_{dist}}{p_m}\right| = \left|\frac{p_m - \frac{p_{left} + p_{right}}{2}}{p_m}\right| \quad (1)$$

2.4. *Evaluación del error.*

Disponer de una simulación monolítica de referencia no es realista. Es necesaria una expresión alternativa para la estimación del error. Para ello, se evalúa la estimación del error en base a la diferencia de potencia transmitida entre ambas partes, como muestra (2).

$$error_{est} = \left|2\frac{p_{left} - p_{right}}{p_{left} + p_{right}}\right| \quad (2)$$

La Figura 3.b muestra una correlación muy consistente entre los errores "exactos" y estimados. Una forma muy similar de usar la transmisión de potencia y de estimar el error aparece en (Santos, 2008).

Notar en este punto que tanto (1) como (2) son ecuaciones que emplean la terminología del modelo desarrollado. Ambas ecuaciones son extrapolables a un conjunto mayor de divisiones, especificando entre qué unidades de intercambio se produce cada error.

3. Control de h_{macro}

El error estimado empleado se puede utilizar como una medida del ajuste de la co-simulación. Un error mayor indica que el sistema está en un punto de dinámica más rápida que aconseja una reducción de h_{macro}. El mánager debe controlar el error y reducirlo por debajo de la tolerancia admitida (ϵ) adaptando h_{macro}. Un controlador PI típico puede hacer el trabajo. Empleamos el mismo esquema que el algoritmo de integración numérica Runge-Kutta-Fehlberg (RK45) (Fehlberg, 1969).

$$h_{macro_i} = h_{macro_{i-1}}\left(\frac{\varepsilon}{error_{est}}\right)^{1/(order+1)} \quad (3)$$

En la ecuación (3), se calcula el nuevo valor del tamaño h_{macro} para la siguiente iteración ($h_{macro(i)}$) a partir del anterior ($h_{macro(i-1)}$), la tolerancia (ϵ) y el error estimado ($error_{est}$). También depende del orden del error. El algoritmo de integración RK45, por ejemplo, es un algoritmo de cuarto orden, por lo que se utiliza la quinta raíz. Para nuestro esquema de error estimado ZOH explícito, se supone que el orden es uno. Además, el valor dado por la (3) está limitado entre dos valores: h_{macro_max} y h_{macro_min}. El uso de h_{macro_min} evita llegar a un valor cero. La importancia del límite superior (h_{macro_max}) se explicará con más detalle en la siguiente sección.

La Figura 4 ilustra el algoritmo de control de h_{macro} descrito. En la Figura 4(a), se muestra la evolución del valor medio de la tensión de intercambio y el de la simulación monolítica para valores decrecientes de la tolerancia (ϵ). La figura 4(b) ilustra la evolución correspondiente del tamaño del paso h_{macro}. Los cambios abruptos en la tensión del modelo de la izquierda exigen una tasa de intercambio más rápida. El

tamaño del paso se reduce rápidamente, para volver a aumentar más tarde cuando se alcanza una situación estática. Observe que al límite superior (h_{macro_max}) se le ha dado un valor de 0.7, de tal forma que en intervalos de dinámica cuasi-estacionaria, h_{macro} corresponde a este valor máximo configurado.

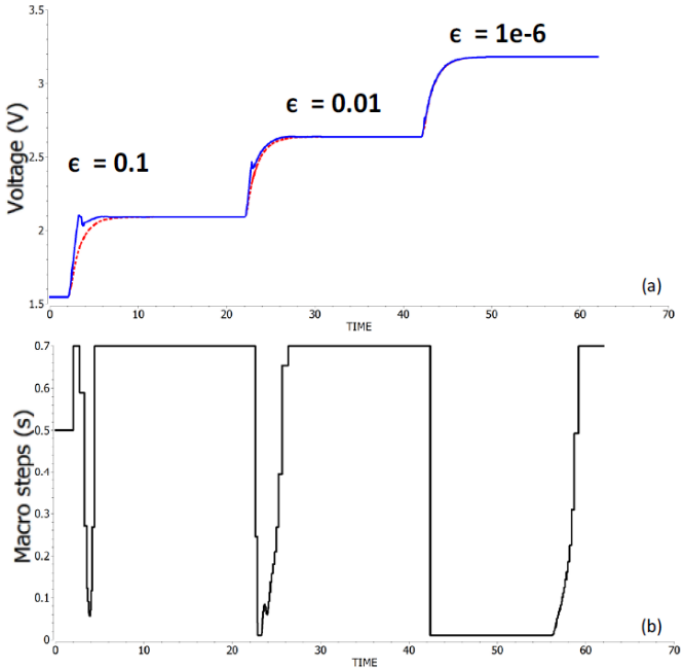

Figura 4: Control de h_{macro} (a) Intercambio de potencia para tolerancias decrecientes. (b) Evolución del tamaño de h_{macro} según intervalo.

4. Control de h_{macro} con eventos

4.1. Planteamiento

En el diseño de una simulación distribuida, cuanto mayor sea el valor del límite superior (h_{macro_max}), mayor será también la aceleración alcanzable por la misma. En una simulación distribuida, la existencia de diferentes sub-simulaciones requiere, de forma eventual, el intercambio de datos en base un cambio en la dinámica de una o ambas particiones.

En la Figura 5 se ilustra el problema. El algoritmo de control de h_{macro} está en su lugar, con una tolerancia de 1e-6, pero con un límite superior demasiado grande, igual a 10 para h_{macro}. Posteriormente, las condiciones cambian abruptamente en la parte izquierda, pero el mánager puede responder mucho más tarde, hasta 10 segundos más tarde, en el peor de los casos, al acabar el ciclo actual.

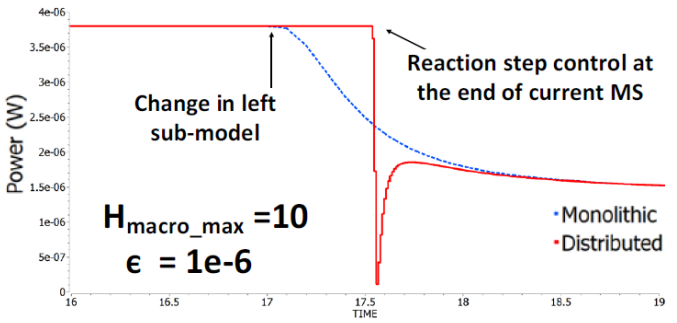

Figura 5: Efecto de un alto valor del límite de h_{macro}.

4.2. Adición de eventos

En el período entre dos pasos de integración h_{macro} sucesivos, las sub-simulaciones se ejecutan de forma independiente y el mánager de simulación no tiene información de posibles cambios. Esta es la causa del problema de falta de capacidad de respuesta descrito. Una posible solución es la incorporación de eventos. La idea es que cada uno de los agentes de simulación participantes realice un seguimiento interno de los cambios de las variables. En el momento en que se detecta un cambio significativo, se desencadena un evento para requerir la atención del mánager.

Al recibir el evento de cualquiera de las partes, el mánager ordena el intercambio. Cada partición puede determinar el error de forma interna. Para el caso que nos ocupa, la partición izquierda, por ejemplo, puede utilizar la (4) para comparar el valor actual de la variable potencia en la frontera con una anterior. La frecuencia con la que se realiza es propia de cada agente, pero en cualquier caso no afecta de forma apreciable a la velocidad de la simulación ya que no es enviada al mánager.

$$error_{left(j)} = p_{left(j)} - p_{left(j-1)} \quad (4)$$

El evento se activa cuando el error local supera un umbral preconfigurado, según se indica en (5).

$$event_{left(j)} = error_{left(j)} > threshold \quad (5)$$

En las Figuras 6 y 7, se ilustran las ventajas de la estrategia de eventos descrita y la interacción con el algoritmo de control de h_{macro}. En la Figura 6(a), el primer cambio dinámico ocurre con un umbral de evento algo alto de 1. Sin embargo, el error no es tan grande porque el límite superior efectivo del paso h_{macro} tenía, en ese momento, un valor bajo de 0.7.

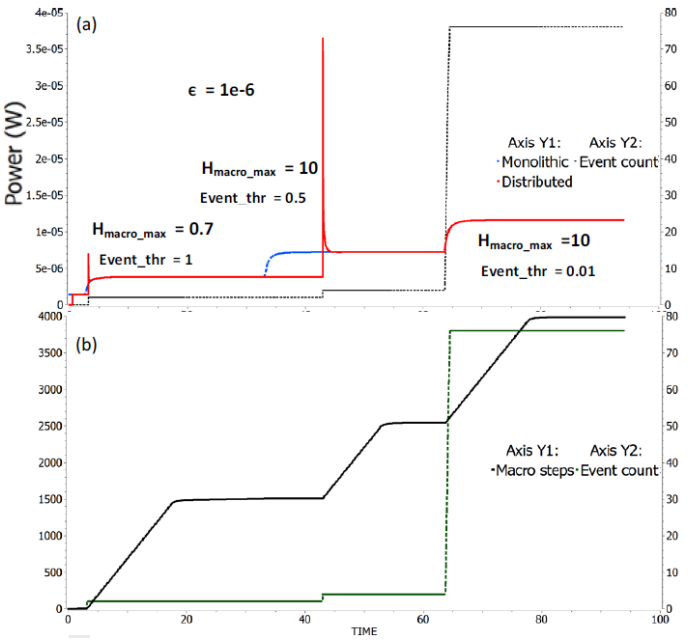

Figura 6: Control de h_{macro} con eventos. (a) Error de potencia transmitida para diferentes valores del límite, event_thr, y h_{macro_max}. (b) Pasos h_{macro} y costo

El segundo cambio, hacia la mitad del horizonte de simulación, ocurre con un umbral más bajo y sensible de 0.5, pero insuficiente para disparar el evento a tiempo. El último cambio se maneja perfectamente con un umbral de evento mucho más sensible de 0.01. Superpuesto en la figura 6(a) y en la figura 6(b) se representa el recuento agregado de todos los eventos. La figura 6(b) también muestra la evolución del número de pasos h_{macro} necesarios. Se puede notar que un umbral más sensible obviamente aumenta el número total de eventos activados, pero ese recuento de eventos sigue siendo muy pequeño en comparación con el número de pasos h_{macro}

En la figura 7, se muestran las ventajas de la adición de eventos. La tasa de pasos h_{macro} de la figura 7(a), inversamente proporcional a la velocidad de la co-simulación, disminuye de forma rápida. La razón es que la simulación se encuentra en un estado estable y el mecanismo de eventos permite tener un paso macro muy grande de 30 segundos. Observar que la simulación sigue respondiendo al cambio repentino que ocurre, aproximadamente, en el tiempo de simulación de 160 segundos. La Figura 7(b) muestra que la diferencia de error entre la simulación de referencia monolítica "exacta" y la estimada permanece bajo control.

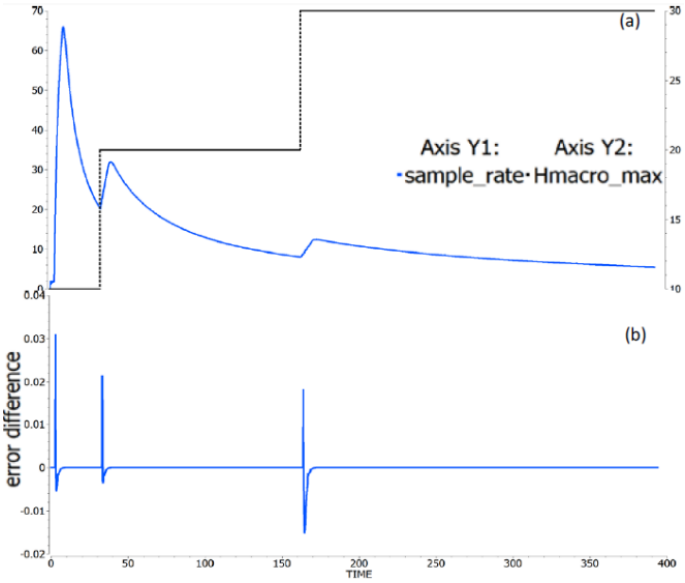

Figura 7: (a) Frecuencia de muestreo y (b) evolución del error con eventos.

5. Integración en PYTHON

El algoritmo de co-simulación se ha desarrollado mediante el empleo de EcosimPro/Proosis. Esta herramienta permite el exportado posterior mediante el empleo de OPC-UA. El modelo matemático programado a nivel computacional se incluye junto con el método de integración numérica elegido. Automáticamente se agrega un wrapper que incluye la funcionalidad de un servidor OPC (Zamarreño, 2014).

Se ha diseñado un marco de co-simulación configurable utilizando el estándar de la industria de procesos OPC-UA como tecnología subyacente, dotando a la herramienta de versatilidad. El diseño emplea el modelo de cliente-servidor.

El cliente configurable se ha implementado en Python 3 (Python Foundation, 2023). Las unidades de simulación SA de la arquitectura corresponden a servidores OPC-UA, mientras que la unidad gestora corresponde a un cliente OPC que lleva a cabo las labores de control de la simulación, mediante la suscripción a cada servidor OPC-UA involucrados. Un esquema puede verse en la Figura 8:

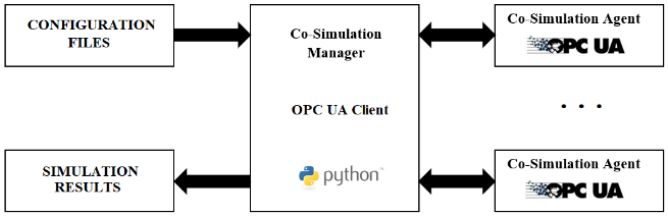

Figura 8: Arquitectura de integración en PYTHON y OPC UA

La base de esta integración queda descrita en (Gutiérrez, 2022). A partir de esta arquitectura, el cliente diseñado se encarga de leer los ficheros de configuración de cada servidor, con el rol activo de generar estructuras de intercambio de datos compatibles y el posterior inicio de la suscripción a cada servidor. El mánager de simulación solicita a cada servidor los valores actuales de las variables interconectadas. Una vez se ha realizado el intercambio de los datos y la posible modificación de h_{macro}, solicita a cada servidor la integración numérica para la siguiente iteración.

Mediante el empleo de esta arquitectura es posible llevar a cabo una estrategia de ampliación del modelo de planta empleado, es posible establecer comunicaciones entre las diferentes unidades de simulación mediante la implementación de una red de computadoras comunicadas a través de una red TCP-IP. El empleo de OPC UA permite no sólo un aumento de la complejidad del modelo, sino la interacción con los servidores en base al uso de eventos. En este punto cabe señalar que, si bien en esta contribución se ha empleado OPC UA, existe una alternativa superior y más específica es la que proporciona el estándar Functional Mock-Up interface (FMI) (Modelica Association Project, 2022). La versión actual de FMI, la versión 3.0, dispone la compatibilidad con el manejo de eventos externos.

6. Conclusiones y líneas de trabajo futuro

En la presente contribución se describe un marco explícito para la administración de una co-simulación que implementa un control del tamaño de paso h_{macro}, añadiendo la posibilidad de incorporar eventos.

La estrategia propuesta es razonable, para ello se ha ilustrado el funcionamiento con un ejemplo sencillo basado en componentes eléctricas, de tal forma que se ha dividido el modelo en dos subtareas. Mediante el empleo de EcosimPro/Proosis es posible implementar el algoritmo descrito. El algoritmo tiene la posibilidad de extenderse para manejar cualquier número de agentes de co-simulación interconectados, gracias al empleo de herramientas de utilidades de amplia difusión a nivel computacional e industrial, como pueden ser PYTHON y OPC-UA. De esta manera, se logra aumento de la versatilidad y su aplicación en diferentes campos.

El cálculo del error presentado a título de ejemplo presenta dos limitaciones. Por un lado, al emplear para el cálculo los valores en instantes sucesivos, es posible que el error se anule si en dos pasos sucesivos se produce una subida y una bajada de este. Una alternativa puede ser llevar a cabo una estimación del error mediante interpolación. Por otro lado, el intercambio bidireccional de variables entre diferentes unidades de simulación requiere de un mayor número de pruebas con el objetivo de fortalecer el algoritmo.

El empleo de OPC-UA para la presente contribución está basado en la amplia disponibilidad del estándar a nivel industrial. En la actualidad, existe una alternativa más potente y específica proporcionada por el estándar FMI. La versión 3 de este estándar permite la gestión de eventos en contraposición con versiones anteriores. La principal cuestión a la hora de emplear FMI 3 radica en que no todas las aplicaciones actuales son capaces de interactuar con él, de tal forma que una posible estrategia para lidiar con el problema sería el desarrollo de una pasarela intermedia capaz de interactuar con FMI 3, aprovechando la experiencia adquirida. A título de ejemplo, EcosimPro/Proosis proporciona FMI 3.0 con limitaciones a la hora de llevar a cabo la gestión de eventos.

A la hora de emplear FMI es también necesario resolver el problema de la realización de pruebas en un entorno de ejecución consistente en un solo núcleo. Existen incompatibilidades en la ejecución en paralelo de dos sub-simulaciones basadas en FMI para la misma unidad. Cabe esperar que la ejecución en paralelo a través de sistemas multi núcleo o en una red proporcione soluciones a este hecho, si bien es necesario llevar a cabo una serie de pruebas para dotar de consistencia a la implementación.

El desarrollo de esta estrategia y la posterior integración mediante el empleo de EcosimPro//Proosis y Python admite la adición de una interfaz de interacción con el potencial usuario. Como trabajos actualmente en desarrollo se está tratando de integrar dicha interfaz basándose en diversas librerías de código abierto actualmente disponibles para Python.

Es necesario en un futuro el desarrollo de un marco de pruebas más complejo para evaluar problemas subyacentes. La división de una planta industrial compleja mediante fronteras basadas en la transmisión de potencia no resulta limitante. Se logra de esta manera un procedimiento sistemático extensible a diferentes aplicaciones.

Agradecimientos

Trabajo financiado y desarrollado gracias a la ayuda del Proyecto PID2021-123654OB-C31 financiado por MCIN/AEI /10.13039/501100011033 / FEDER, UE.

El trabajo fue apoyado además por el Gobierno Regional de la Junta de Castilla y León, UE-FEDER, con referencia CL-EI-2021-07, UIC 233.

Referencias

Cellier, F. E. and Kofman, E. (2006) *Continuous system simulation*, Springer Science Business Media.

Empresarios Agrupados S.A. (2021) EcosimPro/Proosis, Madrid.

Erwin Fehlberg (1969) Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems, *National aeronautics and space administration*, 315.

Gutiérrez F., Mazaeda R., Zamarreño J.M. (2022) Adaptación dinámica de frecuencia de intercambio de datos en una simulación distribuida. XLIII Jornadas de Automática: libro de actas, pp.522-529. https://doi.org/10.17979/spudc.9788497498418.0522

Haid, T., Watzenig, D. and Stettinger, G. (2022) Analysis of the model-based corrector approach for explicit cosimulation, *Multibody System Dynamics*, Springer Science and Business Media B.V., 55(1–2), pp. 137–163

Mazaeda, R., Gutiérrez, F., Zamarreño, J. M. (2023) Event Enhanced Distributed Dynamical Simulation. 22nd IFAC World Congress, 2023. Yokohama.

Modelica Association Project (2022) FMI: Mock-Up Interface: The leading standard to exchange dynamic simulation models. *https://fmi-standard.org/*.

Python Software Foundation. Python Language Reference, version 3. Disponible en http://www.python.org

Sadjina, S., Kyllingstad, L. T., Skjong, S. and Pedersen, · Eilif (2017) Energy conservation and power bonds in co-simulations: non-iterative adaptive step size control and error estimation, *Engineering with Computers*, 33, pp. 607–620.

Santos, R. A., Normey-Rico, J. E., Gómez, A. M., Arconada, L. F. A. and de Prada Moraga, C. (2005) OPC based distributed real time simulation of complex continuous processes, *Simulation Modelling Practice and Theory*, 13(7), pp. 525–549.

Santos, R. A., Normey-Rico, J. E., Gómez, A. M., Arconada, L. F. A. and Moraga, C. de P. (2008) Distributed continuous process simulation: An industrial case study, *Computers and Chemical Engineering*, Elsevier Ltd, 32(6), pp. 1195–1205.

Sun, H., Li, W., Zheng, L., Ling, S. and Fu, W. (2022) Adaptive co-simulation method and platform application of drive mechanism based on Fruit Fly Optimization Algorithm, *Progress in Nuclear Energy*, Elsevier Ltd, 153.

Tamellin, I., Richiedei, D., Rodríguez, B. and González, F. (2022) Eigenstructure assignment and compensation of explicit co-simulation problems, *Mechanism and Machine Theory*, Pergamon, 176, p. 105004.

Zamarreño, J. M., Mazaeda, R., Caminero, J. A., Rivero, A. J. and Arroyo, J. C. (2014) A new plug-in for the creation of OPC servers based on EcosimPro© simulation software, *Simulation Modelling Practice and Theory*, 40, pp. 86–94.