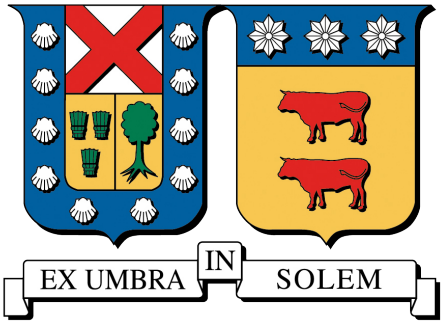
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE OBRAS CIVILES
SANTIAGO - CHILE



IMPLEMENTACIÓN Y VALIDACIÓN VIRTUAL DE PLATAFORMA DE COORDINACIÓN CLIENTE-SERVIDOR PARA SUBESTRUCTURACIÓN MULTITASA EN ENSAYOS DE SIMULACIÓN HÍBRIDA EN TIEMPO REAL

Tesis de Grado y Memoria de Título presentada por

Diego Nicolás Hipólito Mera Muñoz

como requisito parcial para optar al título de

Ingeniero Civil

y el grado de

Magíster en Ciencias de la Ingeniería Civil

Profesor Guía
Gastón Andrés Fermandois Cornejo

21 de abril de 2022

A mis abuelos . . .

Trinidad Del Carmen Ruiz Bravo y Domingo Muñoz Villegas. Abuelos, la huella que dejaron en mí y de la que tan orgulloso me siento, hace que siempre los sienta muy cerca, como una parte más de mi ser. Gracias por establecer las bases de quien soy ahora. Sé que siguen y guían mis pasos desde allí arriba.

AGRADECIMIENTOS

En primer lugar quiero agradecerme a mi mismo. Por creer en mí, por todo lo que este gran trabajo representa, por no tener días libres, por nunca rendirme y por siempre dar lo mejor de mí. Quiero agradecerme por ser yo en todo momento.

Familia y amigos, este logro es en gran parte gracias a ustedes, he logrado desarrollar con éxito un trabajo que en su minuto parecía imposible, pero gracias a sus aportes, amor y apoyo, lo complicado de alcanzar esta meta se ha notado menos.

Agradecer profundamente a mis padres Juana Muñoz e Hipólito Mera, junto con mis segundos padres Trinidad Muñoz y Miguel Robles, por el amor brindado durante mi vida y este proceso. Bárbara, Trinidad, Javiera, Fredy y Miguel, más que primos, mis hermanos. Han estado ahí presentes siempre, más aún cuando los he necesitado, les agradezco de corazón.

A mi pareja Anahi Rivera, eres lo que le faltaba a mi vida y un pilar fundamental para poder culminar este proceso. Me brindaste no solo tu apoyo y comprensión, sino también una infinita paciencia durante esos días donde parecía que nada me resultaba. Te amo con todo mi ser, eres mi inspiración.

A mis amigos Alejandra Bravo, Amada Quiroz, María-Ruth Fuentes, César Guzmán, Lukas Castillo, Camilo Saavedra, Francisco Sáez y Youseff Cid, quienes ya son parte de mi familia, gracias por todos estos años de amor y amistad. A mis amigos Belén Llanquilef, Lukas Mery, Alvaro Aravena, Karla Rivera y Daniel Paredes, con quienes viví cada paso de este largo proceso, compartiendo risas y llantos.

A mi profesor Gastón Fermandois por su apoyo, consejos y conocimiento. Y a mis compañeros del proyecto, Maria Elena Quiroz, Cristóbal Gálmez, Diego Rivera y Daniel Maurel con quienes ha sido todo un agrado y placer trabajar. Gracias por todos sus conocimientos, consejos y aportes. Un especial agradecimiento al profesor Claudio Sepúlveda por toda su ayuda, apoyo y consejos a lo largo estos años.

Gracias!!!

RESUMEN

La simulación híbrida en tiempo real (*RTHS* por su nombre en inglés Real-Time Hybrid Simulation) es una poderosa técnica ciberfísica rentable para realizar la evaluación de sistemas estructurales, el cual se basa en la metodología de subestructuración en la que los componentes difíciles de modelar se prueban en una configuración experimental mientras que el resto de la estructura se modela numéricamente.

En las últimas décadas, ha habido avances significativos en el desarrollo de modelos numéricos no lineales para capturar escenarios de respuesta inelástica y grandes desplazamientos para una evaluación estructural realista. Sin embargo, estos desarrollos están implementados en software de elementos finitos (FE). Por lo tanto, para trabajar con subestructuras más complejas y lograr resultados realistas, es fundamental integrar las capacidades del análisis de elementos finitos.

La motivación de este estudio es brindar un procedimiento general y manejable para que los investigadores en RTHS logren el acoplamiento de un software FE ampliamente utilizado con el software de control disponible en laboratorio. Esta coordinación utiliza la arquitectura cliente-servidor sobre una red informática. En este estudio, se eligieron el software *OpenSeesPy* y *Matlab/Simulink* como FE y software de control, respectivamente. A través de una serie de simulaciones, tanto locales como con hardware externo dedicado, se evaluó y validó la factibilidad de ensayar estructuras de forma precisa, eficiente y con comportamiento realista sin necesidad de una calibración manual del sistema de carga.

Palabras Claves: *simulación híbrida en tiempo real; cliente-servidor; transición entre múltiples tasas, compensación dinámica*

ABSTRACT

Real-Time Hybrid Simulation (*RTHS*) is a powerful and cost-effective cyber-physical technique for evaluating structural systems, which is based on the substructuring methodology, difficult-to-model components are tested in an experimental setup while the rest of the structure is modeled numerically.

In recent decades, there have been significant advances in the development of nonlinear numerical models to capture inelastic responses and large displacement scenarios for realistic structural assessment. However, these developments are implemented in finite element (FE) software. Therefore, to work with more complex substructures in RTHS and achieve realistic results, it is essential to integrate the capabilities of finite element analysis and allow for coordination of the substructures in real-time.

The motivation of this study is to provide a general and manageable procedure for researchers at RTHS to achieve coupling of widely used FE software with available control software in the laboratory. This coordination uses the client-server architecture on a computer network. In this study, *OpenSeesPy* and *Matlab/Simulink* software were chosen as FE and control software, respectively. Through a series of simulations, both local and with dedicated external hardware, the feasibility of testing structures accurately, efficiently, and with realistic behavior without the need for manual calibration of the loading system was evaluated and validated.

Keywords: *real-time hybrid simulation; client-server; multi-rate transition, dynamic compensation*

Índice de Contenidos

1. Introducción **1**
1.1. Objetivos Generales . . . 2
1.2. Objetivos Específicos . . . 2
1.3. Organización del documento . . . 2

2. Revisión de literatura **3**
2.1. Tipos de ensayo . . . 3
2.1.1. Cuasi-Estáticos . . . 3
2.1.2. Mesa Vibradora . . . 3
2.1.3. Simulación Híbrida . . . 4
2.1.3.1. Subestructuración . . . 6
2.2. Plataformas de coordinación para RTHS existentes . . . 10
2.2.1. UI-Simcor . . . 11
2.2.1.1. Clase MDL_RF . . . 11
2.2.1.2. Clase MDL_AUX . . . 11
2.2.2. UT-SIM . . . 12
2.2.2.1. Modulo de Integración . . . 13
2.2.2.2. Modulo de Subestructura . . . 14
2.2.2.3. Network Interface for Controllers (NICON) . . . 14
2.2.2.4. Network Interface for Console Applications (NICA) . . . 14
2.2.3. Mercury . . . 15
2.2.3.1. Elemento híbrido . . . 15
2.2.4. Collaborative Structure Analysis (CSA) . . . 16
2.2.4.1. Implementación de Coordinador . . . 17
2.2.4.2. Métodos de comunicación . . . 17
2.2.5. Matlab-OpenSees programación combinada . . . 18
2.2.6. OpenFresco . . . 19
2.2.6.1. Ensayos Locales . . . 21
2.2.6.2. Ensayos Distribuidos . . . 22
2.2.6.3. Ensayos Client-Server . . . 23
2.2.6.4. Ensayos FE-coupled . . . 24
2.3. Dinámica del sistema de transferencia . . . 24
2.3.1. Modelos de Actuadores Servo-Hidráulicos . . . 24
2.3.1.1. Carrion & Spencer (2007) . . . 25
2.3.1.2. MTS (Schellenberg, 2018) . . . 27
2.4. Métodos de compensación . . . 28
2.4.1. Métodos de modificación de desplazamiento comandado . . . 28
2.4.1.1. Compensación Adaptiva . . . 31
2.4.2. Métodos de correción de la fuerza . . . 34
2.5. Subestructura Numérica . . . 36
2.6. Problema de Elasticidad 2D . . . 39

2.6.1. Formulación Fuerte . . . 39
2.6.2. Formulación Débil . . . 39
2.6.3. Formulación de Elementos Finitos . . . 40
2.7. Formulación de Análisis Isogeométrico . . . 41
2.7.1. Espacios relevantes . . . 41
2.7.2. Funciones de forma . . . 42
2.7.3. Discretización isogeométrica . . . 43
2.7.4. Mapping (cambio de variables) . . . 43
2.7.5. Implementación para problemas de elasticidad 2D . . . 44
2.8. Discusión . . . 45

3. Coordinación de subestructuras en RTHS 46
3.1. Protocolos de Comunicación . . . 46
3.1.1. Antecedentes . . . 46
3.1.2. Descripción general de la API de socket . . . 46
3.1.3. TCP Sockets . . . 47
3.2. Transición entre tasas . . . 48
3.2.1. Predictor-Corrector . . . 48
3.2.2. Transición entre tasas basada en datos (filtro de Wiener) . . . 49
3.2.2.1. Filtro invariante de tiempo lineal . . . 50
3.2.2.2. Filtro variable en tiempo lineal . . . 51
3.2.2.3. Interpolación basada en monomios . . . 53
3.2.2.4. Implementación de la metodología. . . . 54
3.3. Python - Simulink . . . 55
3.3.1. Configuraciones en OpenSeesPy . . . 56
3.3.2. Configuración en Simulink . . . 57
3.3.2.1. Simulación Local . . . 58
3.3.2.2. Simulación Externa . . . 58
3.3.3. Plataforma vRTHS Cliente-Servidor . . . 59

4. Aplicaciones Numéricas 61
4.1. Criterios de Evaluación . . . 61
4.2. Implementación del sistema en tiempo real - OpenFresco . . . 62
4.3. Caso Estudio 1, Modelo 2D . . . 63
4.4. Caso Estudio 2, Modelo 3D . . . 65
4.5. Caso Estudio 3, Análisis Isogeométrico . . . 69
4.5.1. Modelamiento de la cepa . . . 70
4.5.2. Simulación híbrida . . . 70
4.5.3. Resultados . . . 72
4.6. Implementación del sistema en tiempo real - Plataforma vRTHS Cliente-Servidor . . . 82
4.7. Caso Estudio 4, Modelo 2D . . . 82
4.7.1. Subestructuras Numérica y Experimental . . . 82
4.7.2. Subestructura Experimental . . . 83
4.7.3. Resultados Subestructuración . . . 85
4.7.4. Resultados Simulaciones Locales . . . 86
4.7.5. Resultados Simulaciones Modo Externo . . . 89
4.8. Caso Estudio 5, Modelo 3D - OpenSeesPy . . . 92
4.8.1. Subestructuras Numérica y Experimental . . . 92
4.8.2. Resultados Simulaciones Modo Externo . . . 101

5. Conclusiones 103
5.1. Comentarios Generales . . . 103
5.2. Trabajo Futuro . . . 104

Bibliografía 105

A. Apéndice **112**

Índice de Tablas

2.1. Parámetros del modelo del actuador. (Carrion y Spencer, 2007) 27
2.2. Parámetros de los modelos de actuador de MTS. 27

4.1. Criterios de evaluación del desempeño de la vRTHS. 61
4.2. Periodos y Frecuencias Naturales para los primeros 5 modos de vibrar de la estructura 66
4.3. Valores de las propiedades del material y las dimensiones del modelo. 71
4.4. Valores de convergencia de cada método, definidos como el valor de desplazamiento máximo obtenido (en milímetros) con la malla más fina computada para cada método. 74
4.5. Comparación de los resultados de los indicadores de simulación con cada método. 78
4.6. Parametros para los modelos no lineales. 84
4.7. Tabla resumen subestructuración a diferentes tasas de muestreo 86
4.8. Tabla resumen de simulaciones locales para subestructura numérica lineal parámetros fijos . 88
4.9. Tabla resumen de simulaciones locales para subestructura numérica lineal parámetros adaptivos 88
4.10. Tabla resumen de simulaciones locales para subestructura numérica no lineal parámetros fijos. 89
4.11. Tabla resumen de simulaciones locales para subestructura numérica no lineal parámetros adaptivos. 89
4.12. Tabla resumen para simulaciones externas de subestructura numérica lineal parámetros fijos 91
4.13. Tabla resumen para simulaciones externas de subestructura numérica lineal parámetros adaptivos 92
4.14. Tabla resumen para simulaciones externas de subestructura numérica no lineal con parámetros fijos. 92
4.15. Tabla resumen para simulaciones externas de subestructura numérica no lineal con parámetros adaptivos. 92
4.16. Tabla resumen tiempos de simulación. 98
4.17. Tabla resumen de simulaciones externas modelo 3D de 498 GDL no lineal parámetros fijos. . 102
4.18. Tabla resumen de simulaciones externas modelo 3D de 498 GDL no lineal parámetros adaptivos. 102

Índice de Figuras

2.1. Familia de técnicas de simulación híbrida. 4
2.2. Procedimiento de retención en rampa utilizado en pruebas pseudodinámicas convencionales. (Spencer et al., 2007) 5
2.3. Predicción de respuesta para pseudodinámico continuo. (Spencer et al., 2007) 6
2.4. Ejemplo subestructuración y ciclo de resolución mediante simulación híbrida. 7
2.5. Función del framework de coordinación. 10
2.6. Arquitectura del framework UI-Simcor. (Kwon et al., 2007; Spencer et al., 2007) 12
2.7. Esquema resumen del framework UT-SIM. (Mortazavi et al., 2017) 13
2.8. Modelo numérico y físico en simulación híbrida (Mercury). (Saouma et al., 2012, 2014) . . 16
2.9. Sistema de análisis de subestructura (CSA). (Zhang et al., 2014) 17
2.10. Métodos de comunicación (CSA). (Zhang et al., 2014) 17
2.11. Arquitectura central del sistema de prueba híbrido basado en la programación combinada Matlab-OpenSees. (Xu et al., 2021) 18
2.12. Esquema resumen del framwork OpenFresco. (Schellenberg et al., 2009b) 19
2.13. Arquitectura de software OpenFresco para simulación local (izquierda) y distribuida (derecha). (Schellenberg et al., 2009b) 20
2.14. Esquema de Simulación disponibles para Ensayos Locales. 21
2.15. Esquema de Simulación disponibles para Ensayos Distribuidos. 22
2.16. Arquitectura de software de varios niveles. (Schellenberg et al., 2009b) 23
2.17. Control experimental SimFEAdapter. (Schellenberg et al., 2009b) 24
2.18. Modelo de actuador servo-hidráulico acoplado con un especimen físico 25
2.19. Implementación en Simulink del Modelo de actuador servo-hidráulico acoplado con un especimen físico. 26
2.20. Implementación en Simulink del Modelo de actuador servo-hidráulico largo de MTS. 27
2.21. Extrapolación general de desplazamiento comandado. 28
2.22. Problema de seguimiento en RTHS. 29
2.23. Arquitectura de control de la metodología propuesta. 31
2.24. Diagrama de flujo del compensador basado en modelo adaptativo (AMB). 32
2.25. Diagrama de flujo del compensador basado en modelo adaptativo discreto (dAMB). 34
2.26. Estimación de la fuerza correspondiente al desplazamiento deseado mediante el ajuste de curvas de las medidas. (Ahmadizadeh et al., 2008) 35
2.27. Clasificación de modelos para simular la respuesta inelástica de elementos estructurales. (Deierlein et al., 2010) 37
2.28. Espacio indicial generado por dos vectores de nudos iguales a $\Xi^1 = \{0,0,0,1,2,3,3,3\}$ y $\Xi^2 = \{0,0,1,1\}$, donde se destacan las regiones de área paramétrica distinta de cero (Nguyen et al., 2015). 41
2.29. Espacio paramétrico definido por intervalos entre nudos no nulos (Nguyen et al., 2015). . . . 42
2.30. Interpretación de los espacios utilizados en IGA (Nguyen et al., 2015). 43

3.1. Protocolo Cliente-Servidor (TCP) API 47
3.2. Algoritmo predictor-corrector en Stateflow. 48

3.3. Diagrama de bloques para la transición de múltiples velocidades basada en datos para simulación híbrida en tiempo real (DDMRT-RTHS) . . . 49
3.4. Funciones de base monomial. (Heath, 2001) . . . 54
3.5. Esquema de resolución para el cálculo de parámetros fijos. . . . 54
3.6. Esquema de resolución para el cálculo de parámetros variables. . . . 54
3.7. Implementación en Simulink de la metodología dentro de una RTHS virtual . . . 55
3.8. Ciclo de resolución implementado en OpenSeesPy para simulación. . . . 56
3.9. Bloques utilizados en una simulación local. . . . 58
3.10. Esquema de simulación para modo externo . . . 58
3.11. Bloques utilizados en una simulación externa. . . . 59
3.12. Ciclos de simulación e implementación del framework considerados en vRTHS. . . . 60

4.1. Ciclos de simulación considerados en vRTHS. . . . 62
4.2. Subestructuras a resolver en la vRTHS. . . . 63
4.3. Desplazamiento Objetivo v/s Medido e indicador J_2. . . . 63
4.4. Desplazamiento de Referencia v/s Medido e indicador J_4. . . . 64
4.5. Histéresis de la Subestructura experimental (Referencia v/s Medido) . . . 64
4.6. Ticks perdidos y métricas para 40 simulaciones . . . 65
4.7. Subestructuras a resolver en la vRTHS. . . . 66
4.8. Registros sísmicos El Centro 1940, estación El Centro. . . . 66
4.9. Desplazamiento Objetivo v/s Medido e indicador J_2. . . . 67
4.10. Desplazamiento de Referencia v/s Medido e indicador J_4 para grado de libertad vertical en Y. 67
4.11. Desplazamiento de Referencia v/s Medido e indicador J_4 para grado de libertad horizontal X. 68
4.12. Desplazamiento de Referencia v/s Medido e indicador J_4 para grado de libertad Horizontal Z. 68
4.13. Ticks perdidos y métricas para 40 simulaciones . . . 69
4.14. Ciclo resumen de la resolución de la vRTHS. . . . 70
4.15. Modelo del puente que se utiliza en vRTHS. . . . 71
4.16. Dimensiones de la cepa en estudio. . . . 71
4.17. Ciclo de resolución y ejecución de la vRTHS. Se incluyen las subestructuras numéricas y experimentales, compensación del retraso y error de señales, sensores de medición, modelo y controlador del actuador. . . . 72
4.18. Esquema del elementos utilizado para la comparación. . . . 73
4.19. Comparación de la tasa de convergencia con método IGA y FEM v/s ANSYS. . . . 73
4.20. Comparación de la tasa de convergencia con método IGA y FEM para el máximo desplazamiento (en mm) en cada eje de análisis según la cantidad de grados de libertad considerados. 74
4.21. Mallado utilizado en cada método para vRTHS. . . . 75
4.22. Sincronización de la vRTHS con IGA. . . . 75
4.23. Desempeño de la vRTHS con IGA. . . . 76
4.24. Histeresis cepa izquierda del puente con IGA. . . . 76
4.25. Ticks perdidos en la vRTHS con IGA. . . . 76
4.26. Sincronización de la vRTHS con FEM. . . . 77
4.27. Desempeño de la vRTHS con FEM. . . . 77
4.28. Histeresis cepa izquierda del puente con IGA. . . . 78
4.29. Ticks perdidos en la vRTHS con FEM. . . . 78
4.30. Comparación de los desplazamientos obtenidos mediante FEM e IGA en la vRTHS. La figura de la derecha realiza un acercamiento entre los 11.5 y 13 s. . . . 79
4.31. Comparación esfuerzo de Von Misses. Los niveles de deformación se encuentran amplificados por un factor de 5. . . . 79
4.32. Comparación esfuerzo de Von Misses. El rango de colores fue disminuido para mejorar la apreciación. . . . 80
4.33. Comparación de los esfuerzos de Von Misses máximos y mínimos obtenidos mediante FEM e IGA en la vRTHS. . . . 81
4.34. Diferencia entre los esfuerzos de Von Misses máximos y mínimos obtenidos mediante FEM e IGA en la vRTHS. . . . 81

4.35. Subestructuras y componentes de una simulación híbrida. 83
4.36. Relación fuerza vs desplazamiento para diferentes modelos no lineales. 84
4.37. Relación fuerza vs desplazamiento de la subestructura experimental para modelo numérico no lineal. 85
4.38. Implementación en Simulink de la subestructuración dentro de una RTHS virtual 85
4.39. Desplazamiento Objetivo v/s Medido e indicador J_2 86
4.40. Desplazamiento Objetivo v/s Medido e indicador J_2 87
4.41. Desplazamiento de Referencia v/s Medido e indicador J_4. 87
4.42. Histéresis de la Subestructura experimental (Referencia v/s Medido) 88
4.43. Dispositivos de hardware utilizados en el framework de comunicación propuesto. **(1):** PC host (subestructura numérica), **(2):** Enrutador (red LAN) y **(3):** Hardware externo (subestructura experimental) 89
4.44. Desplazamiento Objetivo v/s Medido e indicador J_2 90
4.45. Desplazamiento de Referencia v/s Medido e indicador J_4. 90
4.46. Histéresis de la Subestructura experimental (Referencia v/s Medido) 91
4.47. Subestructuras y componentes de una simulación híbrida. 93
4.48. Comparación Modelos de 192 GDL y 906 GDL. 93
4.49. Comparación Modelos de 498 GDL y 906 GDL. 93
4.50. Modelo de 192 GDL resuelto con 0, 3 y 10 iteraciones 94
4.51. Modelo de 498 GDL resuelto con 0, 3 y 10 iteraciones 95
4.52. Modelo de 906 GDL resuelto con 0, 3 y 10 iteraciones 95
4.53. Modelo de 192 GDL resuelto con 0, 3 y 10 iteraciones 96
4.54. Modelo de 498 GDL resuelto con 0, 3 y 10 iteraciones 97
4.55. Modelo de 906 GDL resuelto con 0, 3 y 10 iteraciones 97
4.56. Comparación respuesta modelo de 498 GDL con 2 y 5 puntos de integración 98
4.57. Norma de la prueba de convergencia considerando 2 puntos de integración. En el eje Y se indica la norma del test versus el tiempo [s] en el eje X 99
4.58. Norma de la prueba de convergencia considerando 5 puntos de integración. En el eje Y se indica la norma del test versus el tiempo [s] en el eje X 100
4.59. Diferencia en la respuesta global de la estructura al modificar el paso temporal de 0.02 a 0.03. 101
4.60. Desplazamiento Objetivo v/s Medido e indicador J_2 101
4.61. Desplazamiento de Referencia v/s Medido e indicador J_4. 102
4.62. Histéresis de la Subestructura experimental (Referencia v/s Medido) 102

1 | Introducción

Existen diferentes técnicas experimentales de ensayo estructural en ingeniería civil, siendo simulación híbrida la más moderna. Esta corresponde a una técnica con interfaz física y computacional que se usa para la evaluación experimental de sistemas estructurales complejos (Nakashima, 2020). La técnica radica en la metodología de subestructuración. En ella se clasifica como subestructura numérica al conjunto de componentes y materiales estructurales cuyo comportamiento es posible predecir, o aquellos que cuentan con modelos matemáticos calibrados y validados experimentalmente. Por otro lado, la subestructura experimental corresponde a los componentes difíciles de modelar, ya sea porque los modelos matemáticos no son capaces de entregar el comportamiento esperado o por el surgimiento de materiales o sistemas innovadores sin un modelo calibrado. Una técnica que se deriva de la simulación híbrida convencional es la simulación híbrida en tiempo real (RTHS en inglés), cuya característica principal es que todos los cálculos, mediciones y procesamiento de señales, deben realizarse en intervalos de tiempo muy cortos, típicamente menos de 1 milisegundo (Nakashima, 2001). Un aspecto crítico de las pruebas RTHS es que cualquier error experimental debe mitigarse durante la ejecución en tiempo real para evitar respuestas inexactas e inestables (Horiuchi et al., 1999). Por lo tanto, la comunidad de ingenieros se ha enfocado en el desarrollo de técnicas de compensación en RTHS, con suficiente rendimiento y garantías de robustez, tales que los experimentos de RTHS se lleven a cabo de manera segura.

La literatura revela que los modelos numéricos simples y lineales se utilizan la mayor parte del tiempo y se implementan directamente en la unidad del microcontrolador para su ejecución en tiempo real (Ashasi-Sorkhabi et al., 2015; Enokida, 2020; Fei et al., 2019; Karavasilis et al., 2011; Stavridis y Shing, 2010). La razón es que la mayoría de las investigaciones priorizaron el estudio de los métodos de compensación. Sin embargo, estos métodos ya han alcanzado la madurez suficiente y si se busca lograr una mayor fidelidad de simulación, es deseable tener un paquete de elementos finitos para modelar la subestructura numérica de manera más realista. En las últimas décadas, ha habido avances significativos en el desarrollo de elementos numéricos no lineales con la finalidad de capturar el comportamiento realista de las estructuras civiles. Sin embargo, estos modelos numéricos avanzados se implementan en un paquete de modelado de elementos finitos. Por lo tanto, un aspecto fundamental en RTHS es la integración de un software de análisis de elementos finitos (FE) con el software controlador utilizado en laboratorio. Hasta la fecha, algunos programas logran este objetivo de simulación híbrida, como UI-SIMCOR (Kwon et al., 2007), OpenFresco (Schellenberg et al., 2009a) (que es una versión mejorada de lo propuesto por (Takahashi y Fenves, 2006)), Mercury (Saouma et al., 2012), CSA (Zhang et al., 2014), UT-SIM (Mortazavi et al., 2017) y la técnica CS (Gu y Ozcelik, 2011). Aunque, los desarrolladores de UI-SIMCOR y Mercury ya no ofrecen soporte actualmente. Además, las técnicas CSA y CS se desarrollaron para trabajar en el acoplamiento de software de elementos finitos, no para trabajar con sistemas ciberfísicos. El resto de los programas de coordinación se desarrollaron originalmente para realizar pruebas pseudodinámicas; por lo tanto, no hay garantías que estos programas completen los cálculos requeridos de manera determinista dadas las limitaciones de tiempo para realizar las pruebas RTHS. Sin embargo, la literatura proporciona algunos ejemplos de realización de RTHS con OpenFresco como software coordinador (Li et al., 2021). El problema radica en la pequeña comunidad que apoya un desarrollo continuo y sostenido de OpenFresco, y la falta de documentación detallada para los métodos de integración/coordinación en las pruebas RTHS. Estos problemas dificultan el desarrollo y la implementación de pruebas RTHS con sistemas acoplados de alta fidelidad.

La motivación de este estudio es proporcionar un procedimiento razonable para que los investigadores

en RTHS logren el acoplamiento de un software de elementos finitos, en esta ocasión OpenSees (McKenna, 2011) y OpenSeesPy (Zhu et al., 2018), con el software de control implementado en Matlab/Simulink (Matlab, 2020). Como primer enfoque, y aproximación a la implementación en laboratorio, se utilizarán entornos virtuales (Silva et al., 2020). Por tanto, en este estudio se trabajará con RTHS virtuales. La coordinación entre tareas computacionales se realiza a través de protocolos de comunicación de internet utilizando la arquitectura cliente-servidor. De esta forma, se evaluará empíricamente la factibilidad de ensayar estructuras con comportamiento realista, de forma precisa y eficiente usando especímenes sin necesidad de una calibración manual del sistema de carga.

1.1. Objetivos Generales

El objetivo general de este trabajo es desarrollar e implementar un procedimiento de coordinación entre subestructuras (numérica y experimental), cuyo procedimiento sea sencillo y de fácil uso.

1.2. Objetivos Específicos

Los objetivos específicos del estudio se presentan a continuación:

i. Desarrollar e implementar una modelación numérica/física para simulación híbrida en tiempo real en el laboratorio.

ii. Desarrollar e implementar un procedimiento de coordinación sencillo que logre acoplar un software FE con el software de control y cualquier otro equipo presente en laboratorio.

ii. Implementar una técnica de compensación adaptiva con configuración robusta, que permita entregar suficientes garantías de estabilidad y precisión durante la ejecución en tiempo real del ensayo de simulación híbrida

iii. Validar, tanto la técnica propuesta de coordinación, como también el algoritmo de compensación dinámica adaptiva, mediante simulaciones virtuales de ensayos dinámicos para especímenes con propiedades tanto lineales, como no lineales e inelásticas.

1.3. Organización del documento

- El capítulo 2 presenta una revisión de literatura respecto a RTHS con un enfoque en el estado del arte de las plataformas de coordinación existentes para el acople entre las subestructuras numéricas y experimentales, actuadores y metodologías de compensación.
- El capítulo 3 describe en que consisten los protocolos de comunicación, transición entre tasas de las subestructuras. En conjunto, se exponen la transición entre tasas y la plataforma de coordinación desarrollada e implementada en esta tesis.
- El capítulo 4 contiene las validaciones numéricas de las metodologías propuestas tanto de compensación, como de comunicación y transición entre tasas. Se indican los criterios de evaluación para la validación de las simulaciones a realizar. Luego, ejemplos utilizando OpenFresco como plataforma de coordinación. Entre los modelos utilizados se tiene modelos resueltos mediante metodología de elementos finitos, como también mediante análisis isogeométrico. La sección 3.3 entrega ejemplos resultos con un framework de coordinación propuesto por el autor utilizando OpenSeesPy y Matlab/Simulink, en conjunto con una transición entre tasas basada en un filtro de Wiener implementada por el autor.
- El capítulo 5 proporciona las conclusiones de este estudio junto con posibles mejores y pasos a seguir para validad experimentalmente las distintas metodologías.

2 | Revisión de literatura

La protección de las estructuras civiles, incluidos el contenido material y los ocupantes humanos, es, sin duda, una prioridad mundial. El alcance de la protección puede variar desde una operación confiable y comodidad del ocupante, hasta la supervivencia humana y estructural. Estructuras civiles, incluidas las existentes y futuros edición, torres y puentes, deben estar adecuadamente protegidos de una variedad de eventos, incluyendo terremotos, vientos, olas y tráfico.

En el ámbito nacional, la elevada amenaza sísmica en Chile nos lleva a querer asegurar de la mejor forma posible que nuestra infraestructura civil pueda resistir de manera adecuada las solicitaciones sísmicas. Si bien contamos con modelos de simulación numérica para realizar diseños estructurales apropiados, estos deben ser siempre comprobados experimentalmente (Scheller et al., 1999). Por ejemplo, la mayoría de los modelos computacionales y relaciones constitutivas son desarrollados y evaluados en la base a resultados de ensayos experimentales.

Las pruebas de laboratorio son esenciales para estudiar el comportamiento de los sistemas estructurales, calibrar modelos matemáticos y desarrollar métodos de diseño robustos para lograr estructuras económicas y seguras. Actualmente existen diferentes técnicas experimentales siendo las más conocidas y utilizadas los (i) ensayos de carga estática, (ii) mesa vibradora, (iii) pseudo-dinámico o simulación híbrida (Blakeborough y Williams, 2001). Otro método de prueba específico que vale la pena mencionar es el método de prueba de fuerza efectiva (EFT); este es conceptualmente diferente al método de prueba pseudo-dinámico, pero de manera similar al método de simulación híbrida se puede aplicar a cualquier sistema estructural que pueda idealizarse con masas agrupadas (McCrum y Williams, 2016).

2.1. Tipos de ensayo

2.1.1. Cuasi-Estáticos

En los ensayos cuasi-estáticos la historia de carga o deformación se aplica a una velocidad suficientemente baja, de tal forma que se considera casi de forma estática. Los ensayos cuasi-estáticos monótonos y cuasi-estáticos cíclicos se clasifican dentro de esta categoría. La práctica actual del diseño de estructuras sometidas a carga sísmica está basada principalmente en resultados experimentales de elementos o sistemas estructurales sometidos a carga cíclica, utilizando tasas de deformación cuasi-estáticas. Estas tasas son significativamente menores que las asociadas a las frecuencias de la excitación sísmica (Shah et al., 1987). La fortaleza principal de los ensayos cuasi-estáticos recae en su relativa economía y practicidad.

2.1.2. Mesa Vibradora

El ensayo de mesa vibradora es el más realista ya que incluye el comportamiento dinámico de estructuras, pero es costoso y complejo, especialmente para grandes estructuras. Diming et al. (1999) y Krawinkler (2000) han reconocido que existen desventajas y limitaciones que aumentan el grado de complejidad en este tipo de ensayos, por ejemplo:

a) El costo elevado de instalación y mantenimiento de mesas vibratorias de gran tamaño para el ensayo de estructuras en escala real.

b) El tamaño del espécimen está limitado por la capacidad de la mesa vibratoria.

c) Los efectos de escala pueden ser importantes en la evaluación de la degradación y de los modos de falla locales de los ensayos a escala reducida. Se generan problemas asociados a reproducir las propiedades mecánicas de los materiales o sus efectos locales, tales como fractura o pandeo local, y los resultados podrían no ser representativos.

d) Los problemas de control debidos a la interacción mesa vibratoria-espécimen o por el cabeceo no deseado de la mesa vibratoria durante el movimiento.

e) El riesgo de daño de la mesa vibratoria y del equipo e instrumentación circundante por llevar estructuras a escala real hasta el colapso.

2.1.3. Simulación Híbrida

Respecto a los ensayos de Simulación Híbrida o Pseudo-Dinámicos, estos tienen una larga trayectoria desde hace más de 50 años (Hakuno et al., 1969). La simulación híbrida (HS por sus siglas en inglés, Hybrid Simulation) se presenta como una alternativa confiable, flexible y práctica para ensayos estructurales. Corresponde a una técnica moderna con interfaz física y computacional que se usa para la evaluación experimental de sistemas estructurales complejos (Nakashima, 2020). La técnica radica en la metodología de subestructuración. Se clasifica como subestructura numérica al conjunto de componentes y materiales estructurales cuyo comportamiento es posible predecir, o aquellos que cuentan con modelos matemáticos calibrados y validados experimentalmente. Mientras, la subestructura experimental corresponde a los componentes difíciles de modelar, ya sea porque los modelos matemáticos no son capaces de entregar el comportamiento esperado o por el surgimiento de materiales sin un modelo calibrado.

A continuación, en la Figura 2.1 se aprecia un esquema con la historia de estos tipos de ensayos.

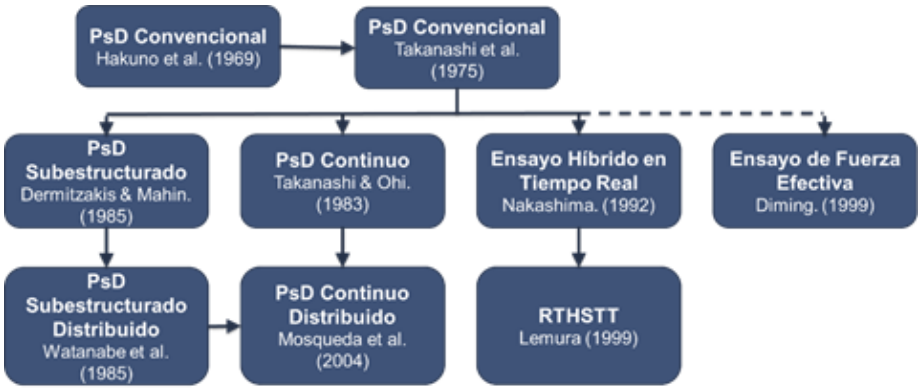


Figura 2.1: Familia de técnicas de simulación híbrida.

El concepto de la prueba de PsD fue propuesto por primera vez por Hakuno et al. (1969) en el que la excitación del actuador de un sistema de un único grado de libertad (SDOF) se combinó con la solución de la ecuación de movimiento en una computadora analógica. El método de prueba PsD fue implementado formalmente por Takanashi et al. (1975). Se consideró que el método era necesario para formar modelos analíticos más precisos para capturar la rigidez y el deterioro de la resistencia durante la carga del terremoto. Por lo tanto, el método PsD analiza la respuesta no lineal de una estructura durante un terremoto, pero con la no linealidad de rigidez modelada por una muestra de prueba física en lugar de un modelo analítico. Una prueba de PsD ofrece una forma poderosa de verificar el comportamiento de los sistemas estructurales resistentes a los sismos, y puede proporcionar datos valiosos para el desarrollo y la calibración de modelos

numéricos no lineales de estructuras y elementos. Permite una simulación sísmica realista sin la necesidad de equipos de prueba con calificación dinámica, y la baja velocidad del ensayo permite un monitoreo e inspección muy completos del estado de la estructura en cada etapa del terremoto.

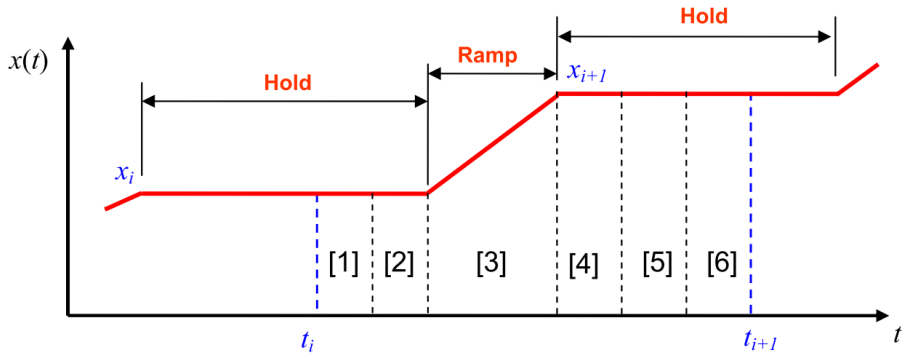


Figura 2.2: Procedimiento de retención en rampa utilizado en pruebas pseudodinámicas convencionales. (Spencer et al., 2007)

Durante el ensayo de simulación híbrida podemos apreciar seis zonas claras. Estas son:

1. Calcular desplazamiento objetivo
2. Enviar desplazamiento objetivo
3. Imponer el desplazamiento objetivo utilizando actuadores
4. Medir la fuerza restauradora y desplazamientos
5. Enviar fuerza y desplazamiento
6. Actualizar respuesta

Un problema con este tipo de ensayos es que se generan zonas y/o periodos de espera (hold), tal como se aprecia en la Figura 2.2. En este periodo el desplazamiento es mantenido constante, lo cual puede provocar fuerzas de relajación en el especimen ensayado y disminuir el realismo del experimento (Spencer et al., 2007). El método PsD Continuo, se diferencia de método convencional, ya que se lleva a cabo utilizando una escala de tiempo ampliada, la cual proporciona el tiempo requerido para la integración numérica de las ecuaciones de movimiento y la implementación de los desplazamientos de comando resultantes por los actuadores (Zhang et al., 2005). Es importante que la respuesta del material que se está probando sea independiente de la tasa de deformación, ya que el comportamiento dependiente del tiempo no se captura (Magonette, 2001). El esquema de integración de tiempo empleado debe ser eficiente, por lo tanto, los procedimientos de integración explícitos se utilizaron típicamente en las primeras investigaciones. Ahora, esta diferencia se aprecia sobre la Figura 2.3.

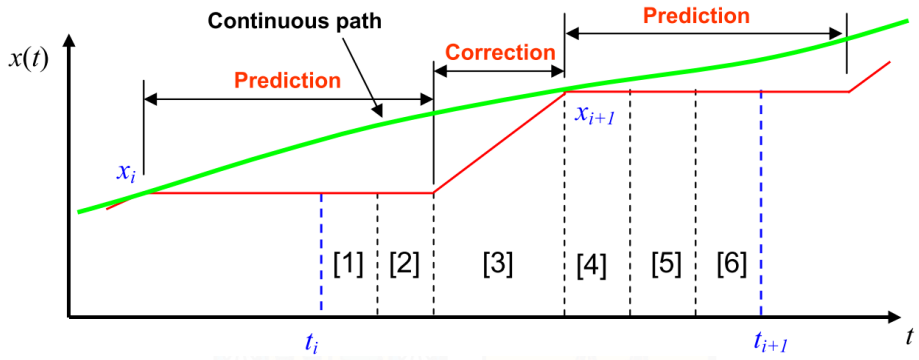


Figura 2.3: Predicción de respuesta para pseudodinámico continuo. (Spencer et al., 2007)

El principal desafío en el ensayo Pseudo Dinámico continuo es que después de que el desplazamiento objetivo es alcanzado, hay un periodo de tiempo en el cual los actuadores tienen que continuar moviéndose aun cuando el siguiente desplazamiento objetivo es desconocido. Una vez el siguiente desplazamiento objetivo es conocido, los actuadores deben corregir su camino y moverse hacia el desplazamiento objetivo.

Una técnica que se deriva de la simulación híbrida convencional es la simulación híbrida en tiempo real (RTHS por sus siglas en inglés, Real-Time Hybrid Simulation), cuya característica principal es que todos los cálculos, mediciones y procesamiento de señales, deben realizarse en intervalos de tiempo muy cortos, típicamente menos de 1 milisegundo Nakashima (2001).

Sin embargo, la RTHS es técnicamente desafiante porque la escala de tiempo en la que se trabaja requiere algoritmos complejos de control o compensación, algoritmos de sincronización y métodos especiales de integración. Un hardware de última generación también es indispensable para realizar pruebas híbridas de manera adecuada. Este último es más pronunciado en las pruebas híbridas en tiempo real porque tales pruebas operan a la velocidad real de la carga de excitación o cerca de ella.

La clave para RTHS es que los desplazamientos comandados se imponen en la muestra de prueba mediante actuadores hidráulicos en tiempo real. Durante la prueba, el cálculo, la comunicación y la dinámica de los actuadores servohidráulicos provocan un retraso de tiempo y un retraso entre los comandos de desplazamiento y las mediciones. Estos retrasos pueden introducir energía al algoritmo de integración, lo cual se puede se puede representar mediante amortiguamiento negativo equivalente, lo que daría lugar a inexactitudes e inestabilidades potenciales (Horiuchi et al., 1995).

2.1.3.1. Subestructuración

Uno de los pasos cruciales al momento de realizar simulación híbrida es la correcta elección de la subestructuración a realizar (Tena Colunga, 2007). Cuando se pretende analizar un gran y complejo sistema estructural mediante métodos matriciales, el número de grados de libertad que se requieren es tal que en muchas ocasiones sobrepasa la capacidad de nuestro computador. En estos casos se recomienda hacer una subdivisión de la estructura en porciones que puedan ser analizadas individualmente, para posteriormente resolver el problema completo con base en las condiciones de borde inherentes a la subdivisión realizada. A esta manipulación matemática se le conoce como subestructuración.

En el contexto de RTHS, en vez de resolver la ecuación de movimiento del dominio completo, este proceso de subestructuración se puede aplicar para subdividir el dominio en subdominios más pequeños de modo que el orden de los sistemas estructurales grandes y complejos se reduzca para eficiencia computacional. Cada subdominio se puede resolver de forma independiente, siempre que se aplique el acoplamiento entre componentes mediante condiciones de compatibilidad y equilibrio en sus interfaces (De Klerk et al., 2008).

El concepto de subestructuración en ensayos estructurales fue primero introducido por Dermitzakis y Mahin (Dermitzakis y Mahin, 1985), y posteriormente implementado por otros (Buchet y Pegon, 1994; Molina et al., 2002; Nakashima et al., 1990). El concepto de subestructura permite que la parte crítica de la estructura (típicamente con una alta respuesta no lineal) sea ensayada físicamente y el resto de la estructura sea modelada numéricamente simultáneamente. La fuerza y los desplazamientos de los actuadores se retroalimentan al modelo numérico, donde se utilizan para resolver la ecuación de movimiento del siguiente paso de tiempo e imponer un desplazamiento. Un problema que aparece es que, en la práctica, es difícil aplicar la gravedad en una prueba subestructurada y muy pocas pruebas al respecto han sido realizadas. Si las cargas de gravedad no se aplican físicamente a la subestructura, entonces los efectos $P-\Delta$ de segundo orden se tienen en consideración numéricamente por la transformación geométrica (Schellenberg et al., 2009a).

La subestructuración juega un rol muy importante en el campo de la dinámica estructural. Realizar análisis por componentes en lugar de todo el problema tiene ventajas, como:

1. Permite evaluar el comportamiento dinámico de estructuras que son demasiado complejas o grandes para ser analizadas como un entero.
2. Al analizar subsistemas, el comportamiento dinámico local puede ser reconocido de forma más fácil que analizando el sistema completo.
3. La subestructuración dinámica ofrece la posibilidad de combinar piezas modeladas (discretizadas o analíticas) y componentes experimentales.
4. Permite compartir y combinar subestructuras de diferentes grupos de proyectos.

Nakashima (2001) presentó el primer sistema capaz de realizar un ensayo en tiempo real. En una simulación híbrida en tiempo real la subestructura numérica explica la masa, la amortiguación y la rigidez de la parte modelada numéricamente, la respuesta de amortiguación e inercia de la subestructura física se mide en el ensayo en laboratorio junto con la fuerza restitutiva, y juntas estas comprenden la retroalimentación de fuerza al modelo numérico. Las pruebas se ejecutan en "tiempo real", lo que significa que todos los cálculos, mediciones analógicas y procesamiento digital de señales, debe realizarse en intervalos de tiempo muy cortos, típicamente menos de 1 milisegundo. Además, las condiciones de borde y compatibilidad también deben imponerse a velocidades rápidas, lo que significa que se requieren actuadores dinámicos para esta tarea. Por lo tanto, los requisitos fundamentales para realizar pruebas RTHS es la implementación de hardware y software rápidos para lograr un resultado estable y preciso.

Definamos una estructura y sus sub-estructuras como la que se muestra a continuación en la Figura 2.4

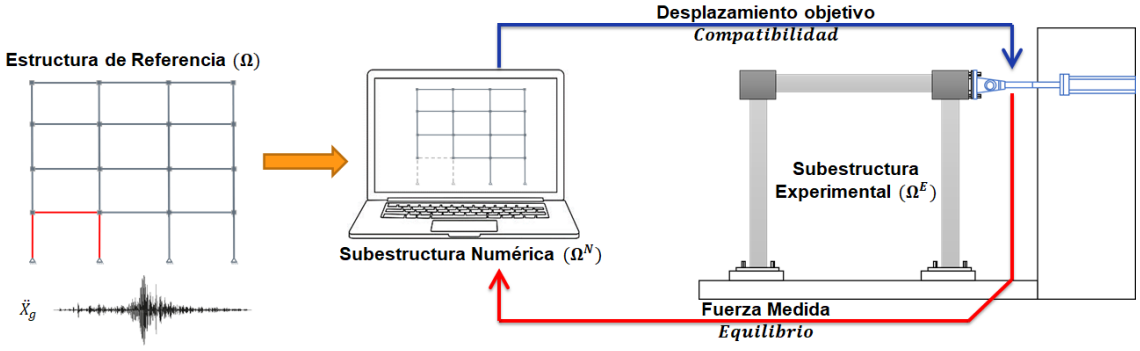


Figura 2.4: Ejemplo subestructuración y ciclo de resolución mediante simulación híbrida.

Si le damos un enfoque matemático, tenemos que las ecuaciones de movimiento que rigen la respuesta dinámica de la estructura sometida a una excitación cualquiera de entrada están representadas como

sigue:

$$M_r\ddot{u}(t) + C_r\dot{u}(t) + r(t) = P(t) \tag{2.1}$$

donde M_r y C_r son las matrices de masa y amortiguamiento del sistema de referencia respectivamente. $\ddot{u}(t)$ y $\dot{u}(t)$ los vectores de aceleración y velocidad, $r(t)$ el vector de fuerza de restauración estructural ($K_r u(t)$ en el caso lineal, donde u es el vector de desplazamiento y K_r la matriz de rigidez) y $P(t)$ el vector de fuerza externa aplicada al sistema. Para RTHS, la ecuación de movimiento se divide en componentes numéricos y experimentales como se ve en la figura anterior.

Consideremos ahora el sistema de dos estructuras mostrados previamente podemos escribir las ecuaciones de movimiento de estas dos subestructuras como se muestra a continuación:

$$M^N\ddot{u}^N + C^N\dot{u}^N + K^N u^N = P^N + g^N \tag{2.2}$$

$$M^E\ddot{u}^E + C^E\dot{u}^E + K^E u^E = P^E + g^E \tag{2.3}$$

Parámetros estructurales asociados con la subestructura experimental son indicados con el superíndice "E" y los asociados con la subestructura numérica son indicados con el superíndice "N". Los términos P^N y P^E son los vectores de fuerzas externas y los términos g^N y g^E son las fuerzas de acoplamiento.

Sean los vectores de desplazamiento asociados a la subestructura numérica y experimental como se muestran a continuación:

$$u^N = \begin{Bmatrix} u_i^N \\ u_b^N \end{Bmatrix} \qquad u^E = \begin{Bmatrix} u_i^E \\ u_b^E \end{Bmatrix} \tag{2.4}$$

De esta misma forma podemos escribir las fuerzas de acoplamiento:

$$g^N = \begin{Bmatrix} g_i^N \\ g_b^N \end{Bmatrix} \qquad g^E = \begin{Bmatrix} g_i^E \\ g_b^E \end{Bmatrix} \tag{2.5}$$

Los componentes previamente mostrados serán indicados con el subíndice "i" para denotar los grados de libertad internos y los asociados a las condiciones de borde serán indicados con el subíndice "b".
El principal supuesto de esta formulación es que las subestructuras están acopladas solo a través de las condiciones de borde. Por tanto, las fuerzas de acoplamiento de los grados de libertad interiores para cada subestructura deben ser iguales a cero.

$$g^N = 0_i^N \qquad g^E = 0_i^E \tag{2.6}$$

Para la resolución de este problema, y tal como se muestra en la Figura 2.4, dos condiciones deben cumplirse:

$$u_b^N = u_b^E \qquad (Compatibilidad) \tag{2.7}$$

$$g_b^N + g_b^E = 0 \qquad (Equilibrio) \tag{2.8}$$

Sustituyendo:

$$M^N\ddot{u}^N + C^N\dot{u}^N + K^N u^N = P^N + \begin{Bmatrix} 0_i^N \\ -g_b^E \end{Bmatrix} \tag{2.9}$$

$$g^E = \left\{ \begin{array}{c} 0_i^E \\ g_b^E \end{array} \right\} = M^E \ddot{u}^E + C^E \dot{u}^E + K^E u^E - P^E \tag{2.10}$$

$$u^N = \left\{ \begin{array}{c} u_i^E \\ u_b^N \end{array} \right\} \tag{2.11}$$

Este incluye todos los efectos potenciales asociados con fuerzas de restauración no lineales, amortiguación no lineal y fuerzas de inercia, junto con cualquier excitación externa que pueda ser inducida directamente a la subestructura experimental. Para obtener una solución admisible, se deben satisfacer la compatibilidad y el equilibrio para todos los grados de libertad y en todo momento.

Desde las ecuaciones previamente mostradas podemos notar que, dado un desplazamiento generado en la subestructura numérica se puede resolver la ecuación de la subestructura experimental y obtener así, la fuerza de acoplamiento que retorna a la ecuación de movimiento de la subestructura numérica, generándose un ciclo como el evidenciado en la Figura 2.4. Este concepto es aplicable a todo ensayo que considere subestructuración. La diferencia entre cada tipo radica en la velocidad a la cual la carga es aplicada. Esto influye directamente en la fuerza restitutiva de la subestructura experimental la cual debe ser calculada de forma diferente. Para clarificar, se muestran a continuación las ecuaciones de movimiento en base a la velocidad de carga.

Ensayo Lento:

$$M\ddot{u}_{i+1} + C\dot{u}_{i+1} + P_r^N(u_{i+1}, \dot{u}_{i+1}) + P_r^E(u_{i+1}) = P_{i+1} - P_{0,i+1} \tag{2.12}$$

En este ensayo se aprecia que la fuerza restitutiva depende exclusivamente del desplazamiento, por tanto, es calculada desde el producto de rigidez por desplazamiento.

Ensayo Rápido:

$$P_r^E(u_{i+1}) = P_{r,i+1}^E - M^E \ddot{u}_{i+1}^E - C^E \dot{u}_{i+1}^E \tag{2.13}$$

En un ensayo rápido aparecen términos extras que hacer referencia a fuerzas inerciales y a disipaciones de energía. Por lo que se hace necesario compensar la fuerza restitutiva estática con estas fuerzas dinámicas.

Ensayo en Tiempo Real:

$$M\ddot{u}_{i+1} + C\dot{u}_{i+1} + P_r^N(u_{i+1}, \dot{u}_{i+1}) + P_r^E(u_{i+1}, \dot{u}_{i+1}, \ddot{u}_{i+1}) = P_{i+1} - P_{0,i+1} \tag{2.14}$$

$$P_r^E(u_{i+1}, \dot{u}_{i+1}, \ddot{u}_{i+1}) = P_{r,i+1}^E + M^E \ddot{u}_{i+1}^E \tag{2.15}$$

Al ejecutar en tiempo real se debe añadir los efectos de las fuerzas inerciales y no solo compensar.

Ensayo de Mesa Vibradora:

$$P_r^E(u_{t,i+1}, \dot{u}_{t,i+1}, \ddot{u}_{t,i+1}) = P_{r,i+1}^E + M^E \ddot{u}_{t,i+1}^E \tag{2.16}$$

En este caso la diferencia radica en el subíndice "t", este hace referencia a que, para la resolución, se necesitan los valores totales (absolutos) tanto de desplazamientos, velocidades y aceleraciones.

2.2. Plataformas de coordinación para RTHS existentes

De suma importancia es el soporte computacional para una simulación híbrida. La esencia de la simulación híbrida es la dependencia de una prueba física en apoyo de una simulación numérica, que es incapaz de simularla numéricamente correctamente. En pocas palabras, y para casos complejos y altamente no lineales, la simulación híbrida con algo menos que un soporte computacional de última generación frustra el propósito de tal esfuerzo. Por lo tanto, la simulación híbrida (en tiempo real o de otro tipo) con un modelo simplista es, para todos los propósitos prácticos, irrelevante, ya que podría ser reemplazada por un modelo numérico más refinado.

En respuesta a esto y dado que se busca lograr una mayor fidelidad de simulación, es deseable contar con un paquete de elementos finitos para modelar la subestructura numérica de manera más realista (sección de fibra, comportamiento del material, modelo 3D). De aquí surge la idea, y necesidad, de acoplar el ensayo en laboratorio con un software de análisis de elementos finitos. Se han desarrollado varios frameworks para coordinación de simulación durante las últimas décadas para facilitar la implementación de simulaciones híbridas y multiplataforma.

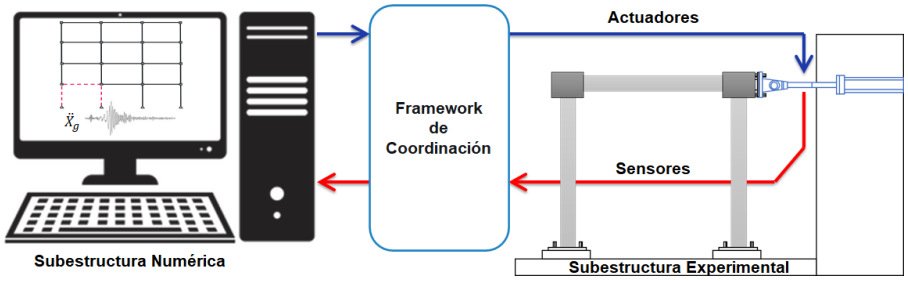


Figura 2.5: Función del framework de coordinación.

Tal como se aprecia en la Figura 2.5, la función principal de los frameworks de coordinación es proporcionar los medios necesarios para comunicar la subestructura numérica y el sistema de control en laboratorio para el ensayo del especimen. Algunos ejemplos se presentan a continuación:

- UI-SimCor (Kwon et al., 2007; Spencer et al., 2007)
- OpenFresco (Takahashi y Fenves, 2006; Schellenberg et al., 2009a,b)
- ISEE (Wang et al., 2007; Yang et al., 2007)
- HybridFEM (Karavasilis et al., 2010)
- Mercury (Saouma et al., 2012, 2014)
- Collaborative Structure Analysis (CSA) (Zhang et al., 2014)
- Celestina-Sim (Lamata Martinez et al., 2016)
- UT-SIM (Mortazavi et al., 2017)
- Ethernet-Based (Peroni et al., 2021)
- FMI-based (Blochwitz et al., 2011; Blockwitz et al., 2012; Andreas Junghanns et al., 2021)
- Matlab-OpenSees programación combinada (Xu et al., 2021)

A continuación, revisaremos más a detalle algunos de estos Frameworks de coordinación.

2.2.1. UI-Simcor

El concepto básico de UI-Simcor es que los modelos analíticos asociados con varias plataformas o especímenes experimentales se consideran un superelemento con muchos grados de libertad. Cada uno de estos elementos se resuelve en una sola computadora o en diferentes computadoras conectadas a través de la red. La Figura 2.6 ilustra la arquitectura general del marco UI-SimCor. La rutina principal que se muestra en la figura refuerza el equilibrio y realiza la integración dinámica del tiempo. En este proceso, el modelo estructural está completamente encapsulado como objetos de una clase. Por lo tanto, es sencillo agregar nuevos métodos o integración de tiempo para hacer cumplir el equilibrio estático.
Hay dos clases en UI-SimCor:

1.- MDL_RF (módulo de fuerza de restauración)

2.- MDL_AUX (módulo auxiliar)

2.2.1.1. Clase MDL_RF

Los objetos de la clase MDL_RF representan componentes estructurales. La principal funcionalidad de esta clase es la abstracción de los componentes estructurales en sitios remotos. Las rutinas principales, como los esquemas de integración dinámica, imponen el desplazamiento sobre los componentes estructurales y recuperan las fuerzas restauradoras sin considerar la comunicación con sitios remotos, independientemente de si los componentes son muestras experimentales o modelos analíticos. Esta abstracción permite una implementación excepcionalmente fácil de nuevas herramientas y componentes de simulación.

Otra funcionalidad importante de la clase MDL_RF es la comunicación. Cuando las rutinas de análisis principales imponen un desplazamiento en un componente estructural representado por un objeto de la clase MDL_RF, el objeto reformatea los datos para el protocolo preespecificado, abre conexiones a los sitios remotos y envía los datos reformateados. En la versión actual se implementan seis tipos de protocolos de comunicación. Estos se presentan en la siguiente sección. La clase MDL_RF incluye otras funcionalidades como comprobar la fuerza y las capacidades de desplazamiento en cada paso de tiempo. Además, el objeto de la clase MDL_RF muestra el estado de la comunicación y monitorea los valores comunicados en cada paso de tiempo.

2.2.1.2. Clase MDL_AUX

La clase MDL_AUX se utiliza para controlar hardware experimental que no sean actuadores. El objeto de esta clase tiene una función para enviar comandos preespecificados a sitios remotos. Al recibir el comando, los sitios remotos pueden tomar acciones como tomar fotografías o activar la adquisición de datos.

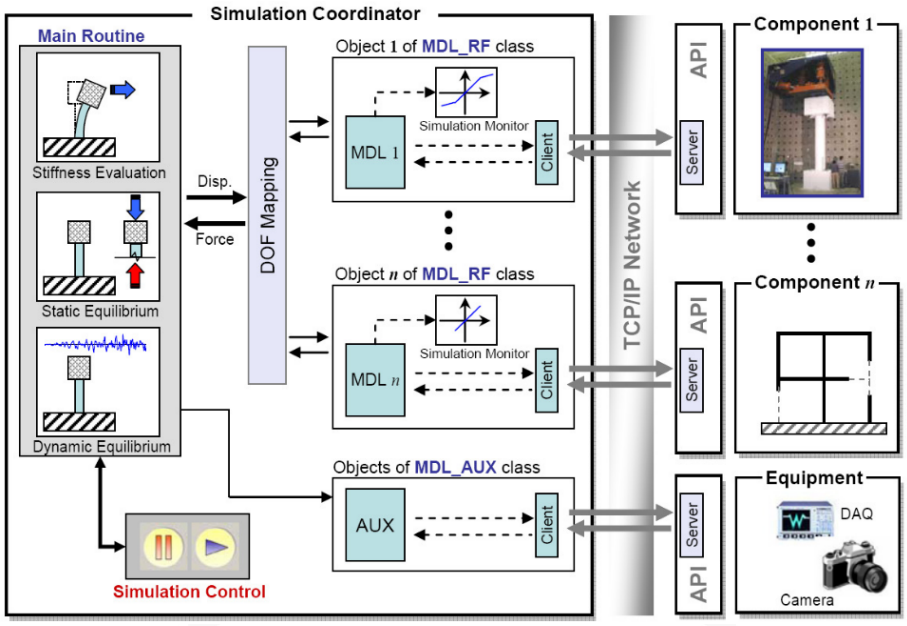


Figura 2.6: Arquitectura del framework UI-Simcor. (Kwon et al., 2007; Spencer et al., 2007)

2.2.2. UT-SIM

Framework de simulación híbrida desarrollado en la Universidad de Toronto, de ahí su nombre, University of Toronto Simulation (UT-SIM). El framework UT-SIM se puede utilizar para simulaciones multiplataforma numéricas distribuidas, así como simulaciones híbridas experimentales.

La evaluación del rendimiento de estructuras complejas que requieren estrategias de modelado avanzadas podría beneficiarse enormemente de las diferentes ventajas que ofrecen los programas de análisis estructural alternativos o incluso las pruebas de muestras físicas. Esto constituye la base de las simulaciones multiplataforma. En las simulaciones multiplataforma, el sistema estructural se descompone en un módulo de integración y varios módulos de subestructura numéricos y/o físicos de modo que la simulación integrada de la estructura puede beneficiarse de las distintas mejoras ofrecidas por diferentes paquetes de análisis estructural y/o ensayo de probetas físicas. Por tanto, tales simulaciones pueden ser puramente numéricas o numéricas-experimentales, como la simulación híbrida en tiempo real y la simulación híbrida pseudodinámica. El esquema de subestructuración en todas las aplicaciones se puede realizar estratégicamente para obtener la mayor ventaja de la simulación mientras se mantiene la eficiencia del análisis.

Los componentes clave en una simulación multiplataforma típica incluyen:

1. Un método de integración de tiempo numérico, denominado módulo de integración
2. Módulos de subestructura numérica
3. Controlador de actuador y muestras físicas en el caso de híbrido numérico-experimental simulaciones, conocidas como subestructuras físicas
4. Un método de comunicación para el intercambio de datos entre las subestructuras físicas y el módulo de integración.
5. Métodos de compensación de errores para la holgura en la configuración de prueba, en simulaciones híbridas experimentales y retardo en Simulaciones híbridas en tiempo real.

Todos estos componentes son apreciables en la Figura 2.7

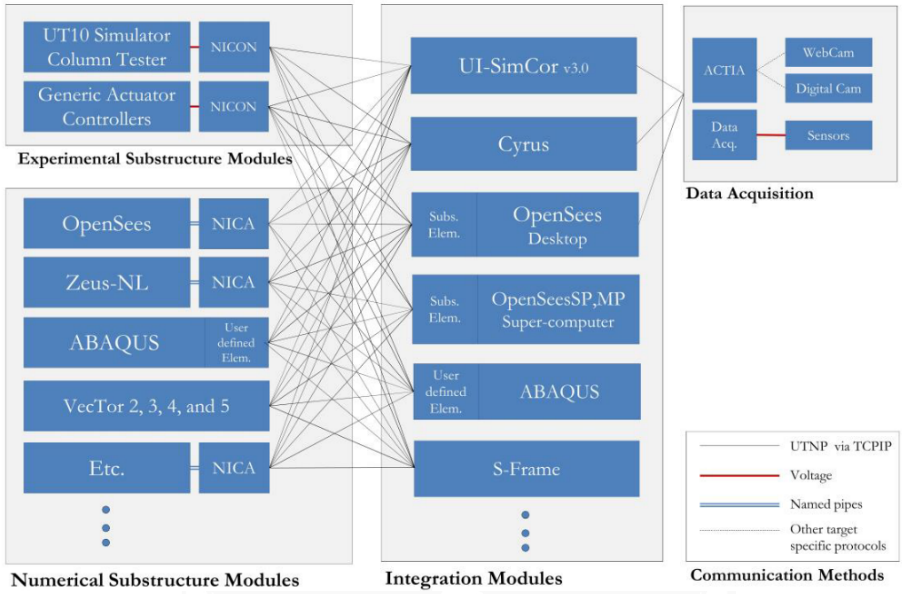


Figura 2.7: Esquema resumen del framework UT-SIM. (Mortazavi et al., 2017)

La característica distintiva del marco UT-SIM es que cualquier paquete de elementos finitos potencial y/o aparato de prueba físico puede integrarse en el marco. Esta capacidad de UT-SIM proviene del desarrollo de un método de comunicación generalizado, denominado Protocolo de red de la Universidad de Toronto (UTNP). El UTNP se compone de un formato de intercambio de datos y un protocolo de comunicación estandarizados.

La UTNP puede comunicar datos a través de TCP-IP o UTP. En la versión actual, el servidor (es decir, el módulo de subestructura) siempre espera una conexión y un comando del cliente (es decir, el módulo de integración). Una vez que el cliente está conectado y se recibe un comando, como imponer un desplazamiento o velocidad a un modelo numérico o una muestra física, el servidor envía resultados analíticos o experimentales al cliente.

El protocolo actual solo permite la comunicación secuencial, es decir, cuando un módulo envía datos, el otro módulo espera recibir los datos. Este protocolo de comunicación puede garantizar que ambos módulos puedan sincronizar completamente los datos en una etapa de carga determinada (paso de tiempo). Sin embargo, puede alargar el tiempo de simulación si la comunicación presenta retrasos que dependen de la distancia física o del tráfico de la red. El retraso de la comunicación no es un problema en las pruebas pseudo-dinámicas o simulaciones puramente numéricas. Sin embargo, si uno desea ejecutar simulaciones en tiempo real distribuidas geográficamente, el retraso será un problema y el protocolo debe mejorarse.

El trabajo de Mortazavi et al. (2017) nos entrega distintos ejemplos de verificación multi-plataforma de la aplicación de este framework, es decir, distintos softwares comunicados entre sí. Algunos ejemplos son OpenSees-OpenSees, OpenSees-Matlab, OpenSees-C++, OpenSees-ABAQUS, ABAQUS-ABAQUS, OpenSees-VecTor, etc.

2.2.2.1. Modulo de Integración

Un módulo de integración es el módulo de software principal, que actúa como el solucionador principal en la simulación y ejecuta el método de integración de tiempo numérico. Normalmente, el módulo de integración se utiliza para modelar la mayor parte de la estructura. En la red de comunicación, el módulo de integración actúa como cliente dentro de la red de simulación.
Los módulos de integración que se pueden utilizar actualmente dentro del marco UT-SIM incluyen:

- UI-SimCor v3.0 (Kwon et al., 2007)
- Cyrus (Sadeghian et al., 2015)
- S-FRAME (S-Frame Software Inc., 2013)
- OpenSees (McKenna, 2011)
- ABAQUS (Dassault-Systemes, 2013)

2.2.2.2. Modulo de Subestructura

Los módulos de subestructura son modelos numéricos y / o especímenes físicos, que representan una región relativamente pequeña de la estructura, cuya respuesta puede afectar críticamente el desempeño estructural general y, por lo tanto, puede necesitar una evaluación de respuesta realista. Por ejemplo, en un marco arriostrado excéntrico (EBF), la viga de enlace podría modelarse utilizando un paquete robusto de elementos finitos como ABAQUS, o incluso representarse mediante una muestra física, mientras que el resto del sistema estructural se modela con elementos del marco. Los paquetes de elementos finitos que se pueden utilizar actualmente incluyen:

- OpenSees
- ABAQUS
- VecTor (Vecchio et al., 1993; Vecchio, 2000, 2001)
- Aplicaciones de consola genéricas como MATLAB y C ++.

2.2.2.3. Network Interface for Controllers (NICON)

Uno de los componentes importantes en las simulaciones híbridas numérico-experimentales es el establecimiento de comunicación entre el controlador del actuador y el módulo de integración numérica. Como parte de este sistema de comunicación, los comandos de desplazamiento en el sistema de coordenadas cartesianas del modelo numérico deben transformarse en coordenadas de los actuadores y desplazamientos de retroalimentación. Además, las fuerzas del actuador deben volver a convertirse al sistema de coordenadas cartesianas del modelo. La interfaz de red para controladores (NICON) es un programa de interfaz de controlador generalizado que se ha desarrollado dentro del marco UT-SIM para este propósito, para facilitar la adopción de simulación híbrida experimental en instalaciones de pruebas estructurales (Kammula et al., 2013; Zhan y Kwon, 2015). NICON está programado en LabView, un software de diseño de sintaxis de programación gráfica, que se puede utilizar junto con el hardware de National Instrument (NI). NICON está programado para recibir comandos desde el módulo de integración a través de UTNP, generar comandos de voltaje analógico a los controladores del actuador y devolver las respuestas medidas al módulo de integración. La primera versión de NICON se había desarrollado para la simulación híbrida de sistemas de grado único de libertad (DOF). El programa está generalizado para que se pueda utilizar para varias configuraciones de prueba, como grados de libertad únicos, tres grados de libertad acoplados, seis grados de libertad acoplados y diez grados de libertad desacoplados. Las pruebas de validación se han llevado a cabo utilizando aparatos de prueba multiaxiales en la Universidad de Toronto.

2.2.2.4. Network Interface for Console Applications (NICA)

Para acoplar programas de análisis estructural como módulos de subestructura, es necesario implementar funcionalidades para comunicarse con un módulo de integración, imponer desplazamientos de destino y devolver las fuerzas de restauración de los DOF controlados al módulo de integración. Sin embargo, muchas herramientas de análisis estructural no tienen tales funcionalidades. Una forma de compensar esto es modificar el código fuente del software. Alternativamente, es necesario utilizar un programa de interfaz para permitir que el software se comunique a través de la red. La interfaz de red para aplicaciones de consola

(NICA) se ha desarrollado en la Universidad de Toronto con este fin. NICA proporciona intercambio de datos entre un módulo de integración y un módulo de subestructura numérico como ABAQUS, OpenSees y Zeus-NL.

2.2.3. Mercury

Mercury (Saouma et al., 2012, 2014) es un conjunto de dos programas idénticos: una versión de MATLAB para instrucción, creación de prototipos y evaluación previa a la prueba; y una versión C++ diseñada para integrarse en un sistema como LabVIEW o Simulink/xPC. Tanto las versiones MATLAB como C++ de Mercury admiten múltiples métodos de integración, incluidos los típicos métodos de solución explícitos e implícitos/iterativos.
El procesamiento del archivo de entrada se realiza utilizando el lenguaje de secuencias de comandos Lua (Ierusalimschy, 2003), que admite conceptos de programación comunes como funciones, matrices y bucles.

Para RTHS, la velocidad del cálculo es crítico; Con este fin, Mercury intenta mejorar la velocidad de cálculo mediante el uso de un solucionador matricial optimizado de terceros (la biblioteca Kernal matemática de Intel) y mediante el multiproceso de partes clave del código computacional en un sistema multiprocesador. El subproceso múltiple se realiza mediante un programador de subprocesos basado en tareas (Intel Threading Building Blocks), que proporciona equilibrio de carga automático, una característica útil cuando los requisitos computacionales se distribuyen de manera desigual en el modelo de elementos finitos.

Mercury se basa en implementaciones 2D de los elementos basados en la flexibilidad (fuerza), que se sabe que son más efectivos que los basados en la rigidez (es decir, requieren menos tiempo de CPU para lograr el mismo nivel de precisión)(Neuenhofer y Filippou, 1997) y modelos constitutivos y esquemas de integración generalmente aceptados.

Para la simulación híbrida, Mercury proporciona dos componentes relacionados, que combinan la simulación numérica con la plataforma de prueba física. El primero es el *elemento híbrido*, que convierte los desplazamientos y fuerzas de elementos numéricos en desplazamientos y fuerzas del actuador físico, y el segundo es la *interfaz de transferencia*, que transporta datos entre Mercury y la aplicación de alojamiento que controla la prueba física. Los dos componentes, independientes pero relacionados, se comunican a través de un mecanismo que fundamentalmente permite que diferentes partes del código lean y escriban valores compartidos referenciados por su nombre.

2.2.3.1. Elemento híbrido

El Elemento Híbrido aparece como un elemento como cualquier otro en el modelo numérico; tiene una matriz de rigidez del elemento (k_e) y produce fuerzas de restauración del elemento tal como lo haría un elemento estándar. En los elementos “regulares", éstos se relacionan mediante un modelo constitutivo definido matemáticamente. Por el contrario, falta un modelo de este tipo para el elemento híbrido (que es precisamente la razón por la que se realiza una simulación híbrida). Así, en esencia, el modelo constitutivo (que relaciona los desplazamientos con las fuerzas) para el elemento híbrido se determina experimentalmente. (Figura 2.8)

Los desplazamientos nodales de los elementos en el modelo estructural numérico se convierten en desplazamientos del actuador (p^n), que luego se aplican a la muestra física. En la otra dirección, los desplazamientos medidos del actuador (p^m) se convierten en desplazamientos nodales del elemento ($\bar{d}_e^d$), y las fuerzas medidas del actuador (A) se convierten en fuerzas de restauración del elemento deseadas ($\bar{f}_e^d$) para ser utilizadas en el modelo numérico . Cabe señalar que las fuerzas de restauración del elemento medidas incluyen todas las fuerzas producidas por la muestra: rigidez, amortiguación y fuerzas de inercia.

El elemento híbrido transfiere datos hacia y desde la muestra física mediante la interfaz externa. Durante cada incremento, iteración o paso de tiempo, el elemento híbrido enviará nuevos desplazamientos nodales del elemento aplicado a la muestra física y recibirá nuevas fuerzas de restauración del elemento

medidas y desplazamientos nodales del elemento de la muestra. Para algunas configuraciones, los desplazamientos nodales del elemento aplicados al elemento híbrido se pueden aplicar directamente a la muestra; sin embargo, en muchos casos, existen diferencias de orientación y escala, por lo que puede ser necesario aplicar una rotación y escala a los desplazamientos nodales del elemento y las fuerzas de restauración del elemento que se transfieren entre el elemento híbrido y la muestra física.

Finalmente, el usuario puede insertar múltiples elementos híbridos en un modelo Mercury y acoplarlos para separar muestras; esto permite la prueba simultánea de múltiples componentes en una prueba híbrida

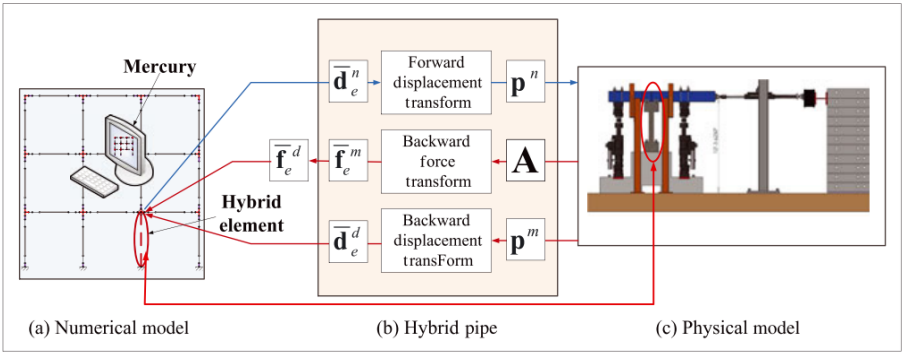


Figura 2.8: Modelo numérico y físico en simulación híbrida (Mercury). (Saouma et al., 2012, 2014)

2.2.4. Collaborative Structure Analysis (CSA)

El marco del sistema CSA (Zhang et al., 2014) es similar al del sistema de prueba híbrido P2P. En este, la estructura simulada se divide en varias subestructuras, y cada una de ellas se trata por igual. Las ecuaciones de movimiento para cada subestructura se formulan y resuelven de forma independiente y simultánea. La interacción entre las subestructuras se considera satisfaciendo las condiciones de compatibilidad y equilibrio en los límites.

El sistema CSA consta principalmente de dos módulos, “Coordinator” y “Partner”, como se muestra en la Figura 2.9. Para lograr la compatibilidad y el equilibrio en el límite, el Coordinador está equipado con un algoritmo basado en un método de iteración Quasi-Newton. Cada socio se comunica con el coordinador y tiene interfaces para comunicarse con el software FE para controlar el análisis. Dado que solo se utiliza una interfaz de I/O estándar, se pueden utilizar diferentes software de análisis de FE en varios análisis de subestructura siempre que se cumplan los estándares de la interfaz de I/O.

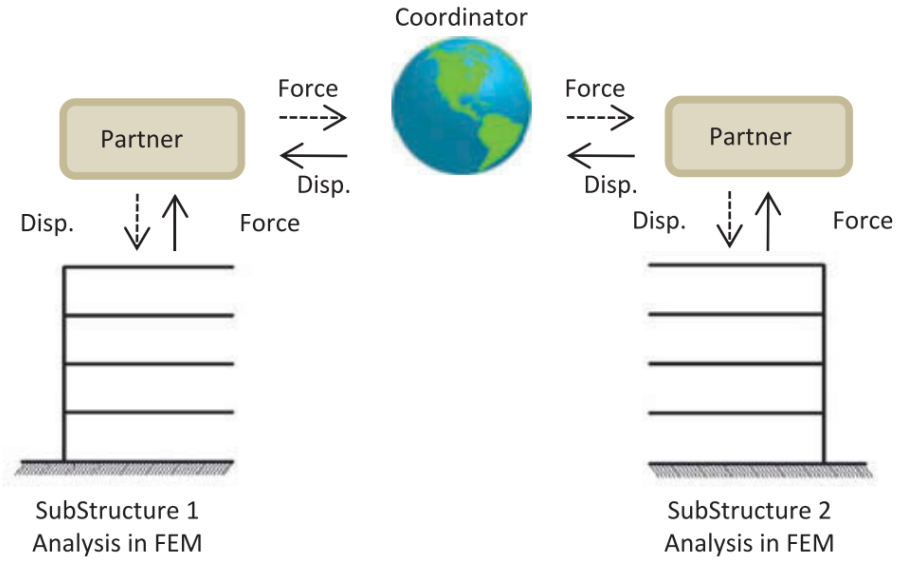


Figura 2.9: Sistema de análisis de subestructura (CSA). (Zhang et al., 2014)

2.2.4.1. Implementación de Coordinador

El núcleo de este sistema es el Coordinador y la función más importante para el Coordinador es obtener desplazamientos de límites correctos basados en las fuerzas desequilibradas. Para este propósito, se utiliza un método iterativo de Cuasi-Newton. En este estudio se utiliza el método Broyden-Fletcher-Goldfarb-Shanno (BFGS), uno de los métodos más populares de la familia Quasi-Newton (Pan et al., 2006).

2.2.4.2. Métodos de comunicación

Los métodos de comunicación se muestran en la Figura 2.10. Se utiliza un método de conexión basado en Internet para resolver la comunicación entre el "Coordinador" y el "Socio". Socket es un método de comunicación multiplataforma que es rápido y estable, lo que lo hace adecuado para este problema. Las pruebas en diferentes versiones de los sistemas operativos Windows y Linux han demostrado la alta eficiencia de este método (Tada et al., 2008).

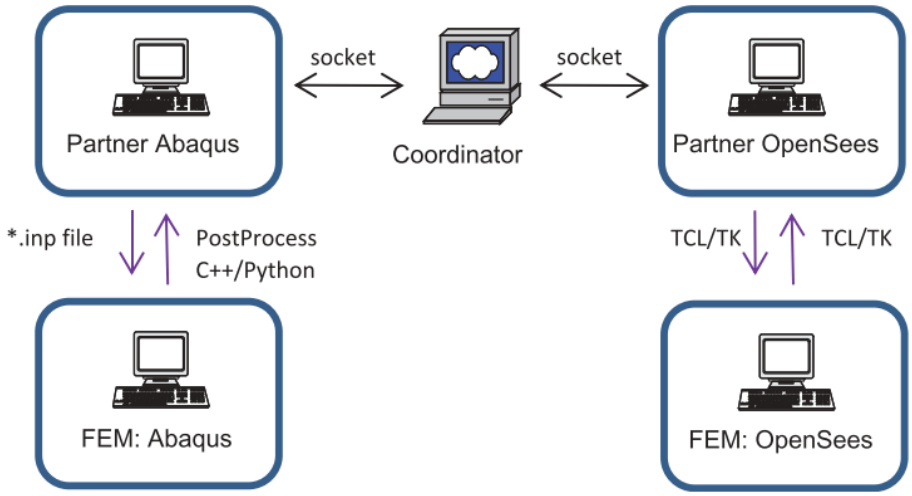


Figura 2.10: Métodos de comunicación (CSA). (Zhang et al., 2014)

2.2.5. Matlab-OpenSees programación combinada

La arquitectura central del sistema de prueba híbrido basado en la programación combinada Matlab-OpenSees (Xu et al., 2021) se ilustra en la Figura 2.11. La arquitectura del sistema está compuesta por el sistema Matlab-OpenSees y el sistema de actuador.

El sistema Matlab-OpenSees incluye el módulo OpenSees, el módulo Matlab y la interfaz Matlab-OpenSees. El módulo OpenSees es para el modelado y cálculo de subestructura numérica. El módulo Matlab se utiliza como intermediario para la transmisión de datos entre OpenSees y el sistema del actuador. Adicionalmente, se utiliza para trazar la respuesta de la subestructura experimental, esto puede facilitar el monitoreo durante el proceso de prueba. La interfaz Matlab-OpenSees es una interfaz de comunicación desarrollada en base al protocolo TCP/IP para realizar la transmisión de datos en tiempo real entre el módulo Matlab y el módulo OpenSees. El sistema de actuador consta de actuador, sistema de control de actuador y sensores que se utilizan para cargar la subestructura experimental, recolectar la fuerza de restauración de la subestructura experimental y transfiera los datos de prueba con el sistema Matlab-OpenSees a través de comunicación en serie.

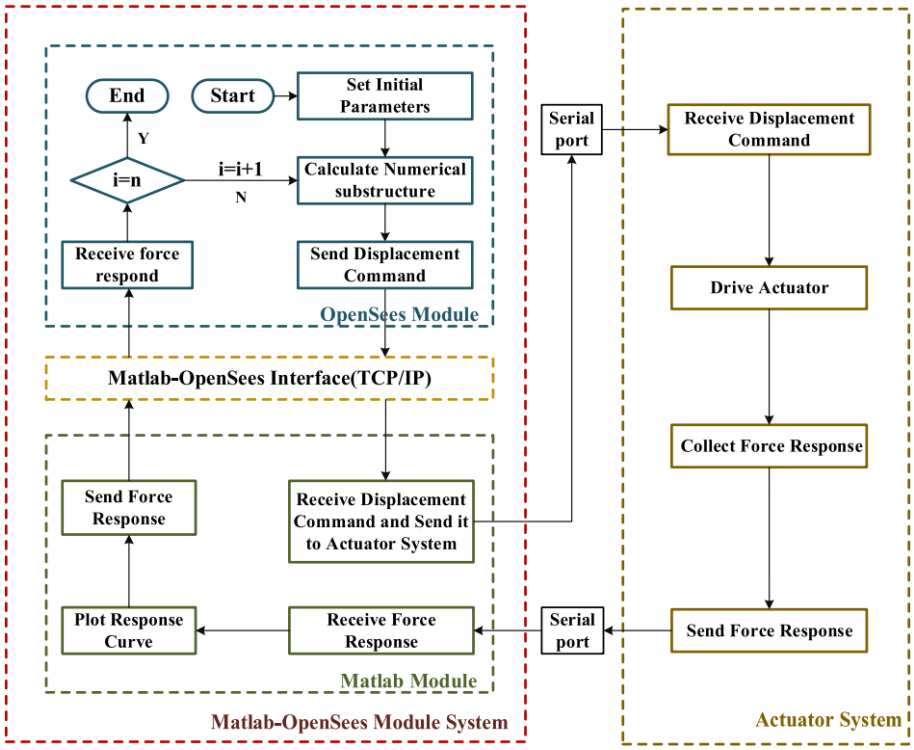


Figura 2.11: Arquitectura central del sistema de prueba híbrido basado en la programación combinada Matlab-OpenSees. (Xu et al., 2021)

Acorde a la Figura 2.11, el proceso detallado del sistema de prueba híbrido es el siguiente:

1) El módulo OpenSees establece los parámetros iniciales, establece un servidor para la comunicación TCP/IP y espera al cliente que el módulo Matlab establece para acceder. Una vez establecida la comunicación, sea $i = 1$.

2) El cálculo del modelo de subestructura numérico se realiza en el módulo OpenSees para obtener el desplazamiento entre capas del paso 1), y la señal de comando de desplazamiento se envía al módulo Matlab a través de la interfaz Matlab-OpenSees.

3) El módulo Matlab envía la señal de comando de desplazamiento recibida al sistema del actuador a través del puerto serie y el sistema del actuador impulsa el actuador para cargar la subestructura

experimental para realizar el comando de desplazamiento. Luego, la señal de respuesta de fuerza recopilada por el sensor se envía al módulo Matlab a través del puerto serie.

4) El módulo Matlab recibe la señal de respuesta de fuerza enviada por el sistema de actuador y luego envía la señal de respuesta de fuerza al módulo OpenSees a través de la interfaz Matlab-OpenSees.

5) Después de que el módulo OpenSees recibe la señal de respuesta de fuerza, si $i < n$ (pasos de análisis total), la fuerza de reacción de la subestructura experimental se aplica a la subestructura numérica, sea $i = i + 1$ y repita el proceso, pasos 2) al 5). Si $i = n$, se ejecuta el proceso 6).

6) La prueba híbrida ha terminado.

2.2.6. OpenFresco

En el pasado, cada implementación de simulación híbrida ha sido específica de un problema y altamente personalizada para el software y hardware disponible en un laboratorio determinado. Por lo tanto, los investigadores que deseaban realizar una simulación híbrida tenían que estar dispuestos a invertir un tiempo y un esfuerzo considerables para familiarizarse con los sistemas de control y adquisición de datos disponibles y para implementar los detalles específicos del problema. Por lo tanto, muchos investigadores se han mostrado reacios a utilizar la simulación híbrida para investigar el rendimiento de los sistemas estructurales en las condiciones mencionadas anteriormente. Además, ha sido un desafío, o muchas veces imposible, emplear paquetes de software de elementos finitos existentes para ejecutar simulaciones híbridas. Esto se debe a la falta de middleware que proporcione un enfoque estandarizado e independiente del entorno para conectar cualquier paquete de software de elementos finitos de la elección del usuario con las instalaciones del laboratorio donde se realizan las pruebas experimentales.

Para abordar esta importante necesidad en la ingeniería sísmica aparece el Open Framework for Experimental Setup and Control (OpenFresco (Takahashi y Fenves, 2006; Schellenberg et al., 2009a,b)). Este es un sistema de software para la simulación híbrida de sistemas estructurales, tal como se aprecia en la Figura 2.12. Para el usuario de ingeniería sísmica, OpenFresco permite combinar una amplia variedad de algoritmos de simulación híbridos, sistemas de control y de laboratorio, configuraciones experimentales y modelos de simulación computacional para una simulación híbrida específica.

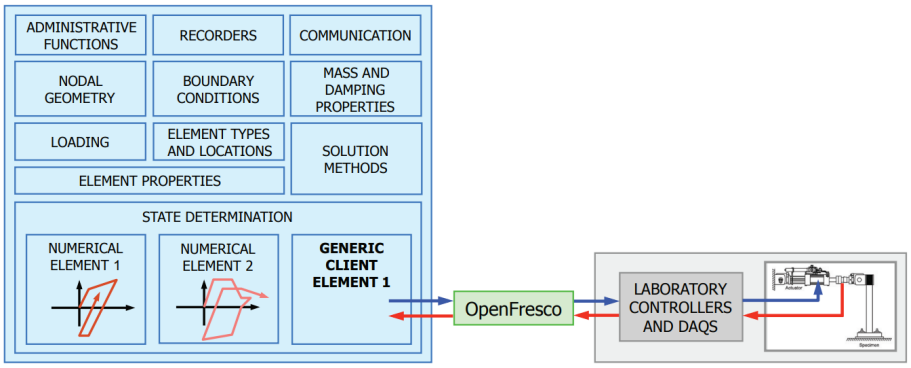


Figura 2.12: Esquema resumen del framwork OpenFresco. (Schellenberg et al., 2009b)

A continuación, en la Figura 2.13, se muestran las arquitecturas de ensayo disponible en OpenFresco. En estos se define un "Client" donde se define la subestructura numérica, un "Server" que representa la subestructura experimental y Openfresco se encarga del Middle, donde se definen múltiples variables dependiendo de si se trata de una simulación local o geográficamente distribuida

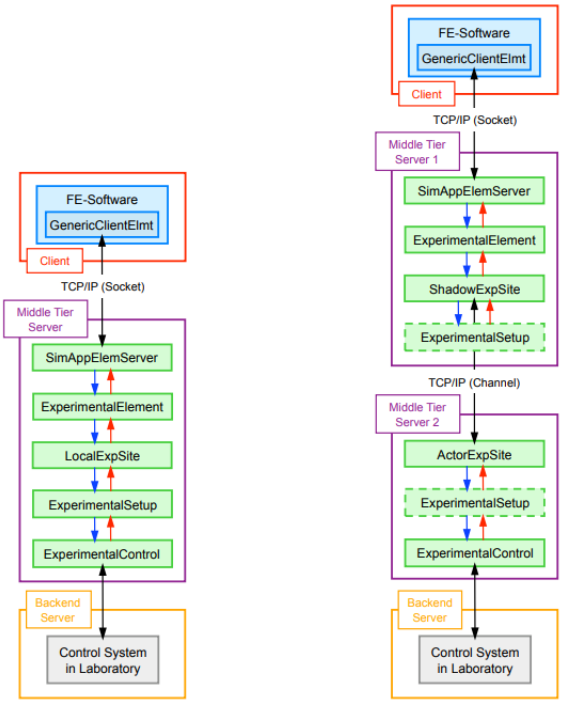
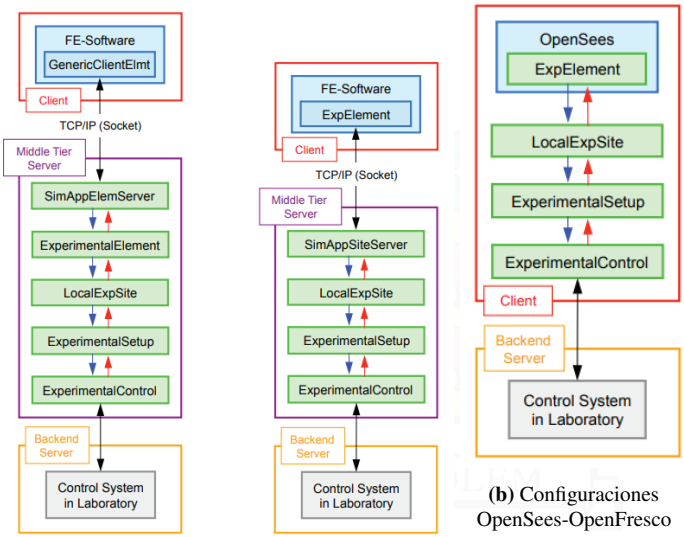


Figura 2.13: Arquitectura de software OpenFresco para simulación local (izquierda) y distribuida (derecha). (Schellenberg et al., 2009b)

Un aspecto importante de OpenFresco es que el software de elementos finitos usa una forma general de un elemento, denominado Elemento Genérico-Cliente. El usuario no tiene que crear un elemento experimental en el software de elementos finitos cuando se utiliza el Elemento Genérico-Cliente. El elemento genérico-cliente se puede implementar fácilmente en cualquier software de elementos finitos que permita elementos definidos por el usuario. Esta característica permite que OpenFresco se use con una amplia variedad de paquetes de software computacional, como OpenSees, Matlab, LS-DYNA, UI-SimCor, Abaqus y Simulink. Para los desarrolladores y usuarios avanzados, se pueden crear elementos experimentales para aplicaciones más sofisticadas, pero para la mayoría de los usuarios interesados en la simulación hibrida, el elemento genérico-cliente debería ser suficiente.

Adicionalmente, OpenFresco permite realizar ensayos distribuidos, una prueba distribuida consta de la arquitectura cliente-servidor de varios niveles que se muestra en la Figura 2.13 de la derecha. El primer servidor de nivel medio y el segundo nivel medio y back-end servidor que se ejecuta como tres procesos separados. Los procesos se pueden ejecutar en la misma computadora, pero en una aplicación práctica el cliente y los primeros procesos del servidor de nivel medio se ejecutarían en una máquina y los procesos del segundo servidor de nivel medio y backend se ejecutarían en un equipo diferente o maquina a través de una red. La configuración experimental se puede definir en el primer o segundo lado del servidor de nivel medio. Para interactuar con una variedad de paquetes de software de elementos finitos, OpenFresco utiliza el concepto de arquitectura de software de tres niveles y un elemento de cliente genérico en el software FE. Se puede implementar un elemento de cliente genérico en cualquier software FE, que proporciona una interfaz de programación de aplicaciones (API) que permite la adición de elementos definidos por el usuario. Además, un elemento de cliente genérico solo necesita implementarse una vez para cualquier software FE que se desee utilizar como controlador computacional para realizar simulaciones hibridas. Se han implementado elementos de cliente genéricos para OpenSees, Matlab, LS-DYNA, UISimCor, Abaqus y Simulink.

2.2.6.1. Ensayos Locales



(a) Configuraciones de tres niveles para implementación local. (Schellenberg et al., 2009b)

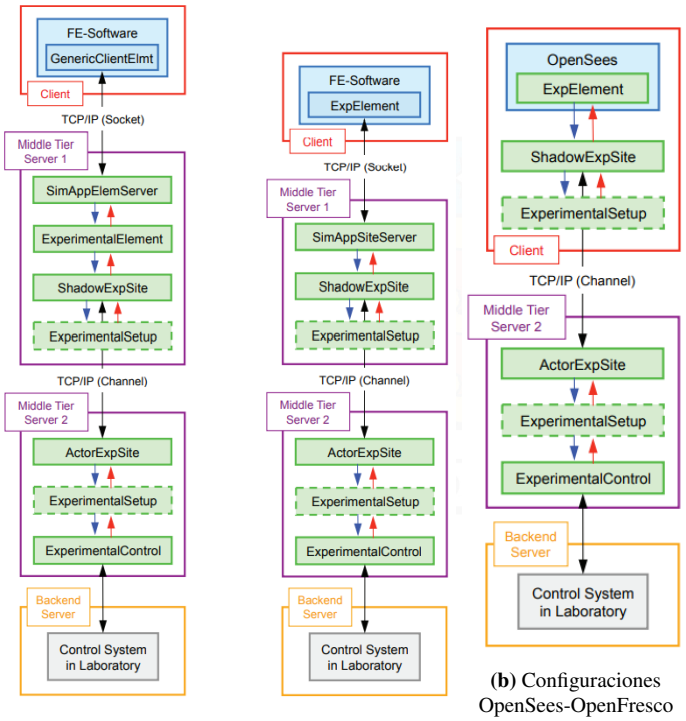
(b) Configuraciones OpenSees-OpenFresco para implementaciones locales. (Schellenberg et al., 2009b)

Figura 2.14: Esquema de Simulación disponibles para Ensayos Locales.

En el caso de una implementación local de OpenFresco, lo que significa que todos los procesos se ejecutan localmente en un laboratorio, hay dos configuraciones posibles disponibles, como se muestra en la Figura 2.14a. En la primera configuración, que es la más general, se debe agregar un elemento de cliente genérico al software de elementos finitos que se está utilizando para el análisis. El elemento de cliente genérico requiere solo el número de nodos, los grados de libertad a los que está conectado y el puerto a través del cual comunicarse como parámetros de entrada. Además, se puede implementar en el lenguaje del correspondiente software de análisis de elementos finitos. Por lo tanto, el intercambio de datos con el middleware OpenFresco se lleva a cabo a través del elemento de cliente genérico. La ventaja de esta primera configuración es que solo un elemento, el elemento cliente genérico, debe agregarse al software de análisis de elementos finitos y todos los elementos experimentales existentes en el marco del software OpenFresco pueden utilizarse para representar las partes experimentales de una estructura.

Para RTHS se puede utilizar la configuración OpenSees-OpenFresco para implementaciones locales 2.14b. Al ejecutarse, OpenSees se vincula dinámicamente con OpenFresco y adquiere acceso directo a todos los comandos de OpenFresco sin la necesidad de ninguna comunicación de red entre los dos. La ventaja de hacer uso de esta configuración, es la mayor rapidez en la transmisión de información entre los diferentes componentes disminuyendo así la latencia de datos.

2.2.6.2. Ensayos Distribuidos



(a) Configuraciones de cuatro niveles para la implementación distribuida. (Schellenberg et al., 2009b)

(b) Configuraciones OpenSees-OpenFresco para implementaciones distribuidas. (Schellenberg et al., 2009b)

Figura 2.15: Esquema de Simulación disponibles para Ensayos Distribuidos.

Como se puede ver en la Figura 2.15, en el caso de una implementación distribuida de OpenFresco, están disponibles dos configuraciones posibles muy similares (Figura 2.15a). Su principal diferencia es que el servidor de nivel medio se divide en dos unidades que se distribuyen en una red. Los sitios experimentales shadow y actuadores, que son subclases concretas de la clase base ExperimentalSite, gestionan las comunicaciones entre los dos niveles. La tercera configuración (Figura 2.15b) se genera de forma similar a la configuración local antes mostrada en la cuál la subestructura numérica se ejecuta al interior de OpenFresco. También es conocida como configuración OpenSees-OpenFresco para ensayos distribuidos.

2.2.6.3. Ensayos Client-Server

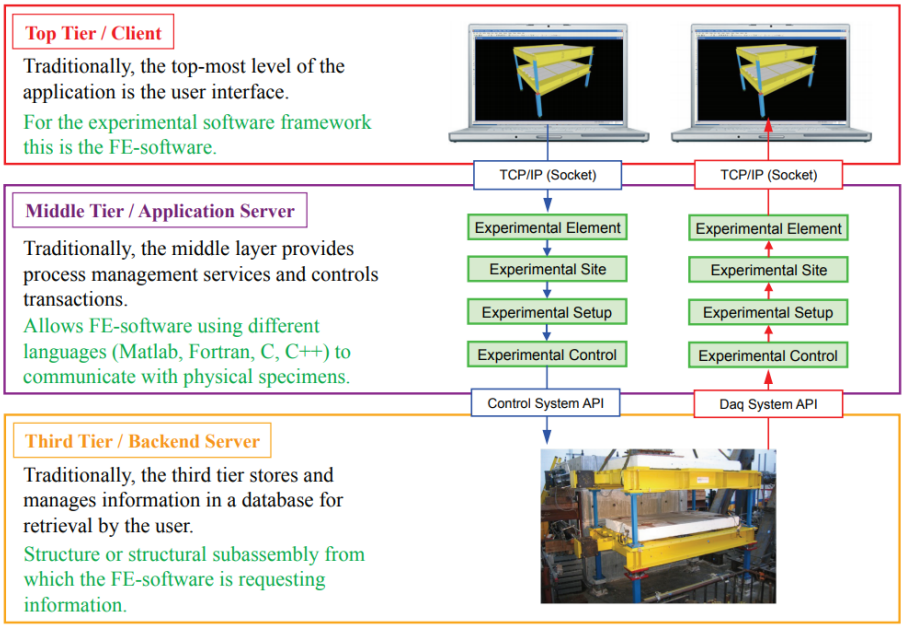


Figura 2.16: Arquitectura de software de varios niveles. (Schellenberg et al., 2009b)

La arquitectura para estos ensayos se muestra en la Figura 2.16. En esta, el software está dividido en tres capas compuestas por un cliente, un servidor de nivel medio (o aplicación) y un servidor backend. El servidor de nivel medio se encuentra entre la interfaz de usuario (cliente) y los componentes de administración de datos (servidor). La arquitectura de tres niveles se emplea normalmente siempre que se requiere un diseño de cliente / servidor distribuido que proporcione un mayor rendimiento, flexibilidad, capacidad de mantenimiento, reutilización y escalabilidad al tiempo que oculta al usuario la complejidad del procesamiento distribuido. Utilizando el socket proporcionado por la plataforma OpenFresco, el elemento genérico definido por el usuario se comunica con el servidor del elemento de aplicación de simulación a través de una conexión TCP/IP.

Debido a que el servidor de nivel medio está dividido en dos unidades, la clase ExperimentalSetup, que es responsable de la transformación entre los grados de libertad del elemento y del actuador, se puede ubicar en la primera unidad del servidor de nivel medio (generalmente ejecutándose en la misma máquina que el cliente de análisis de elementos finitos) o en la segunda unidad de servidor de nivel medio (que normalmente se ejecuta en el laboratorio), pero no en ambos lados simultáneamente.

2.2.6.4. Ensayos FE-coupled

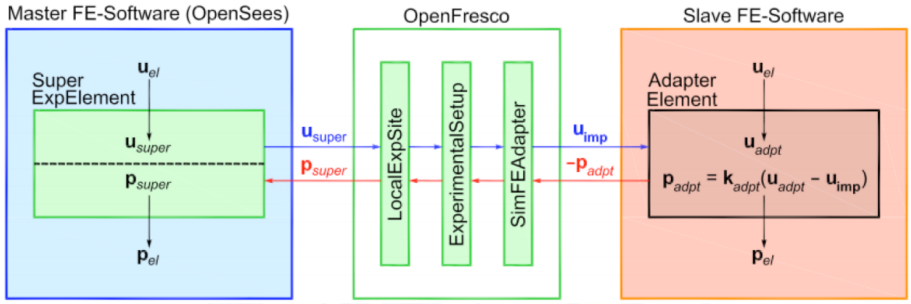


Figura 2.17: Control experimental SimFEAdapter. (Schellenberg et al., 2009b)

El método de simulación FE-coupled propuesto se muestra en la Figura 2.17. El modelo numérico completo se divide en dos subestructuras. La primera se conoce como el modelo maestro (Master), que se compone de toda la estructura, excepto los elementos que están planificados para ser probados experimentalmente. La segunda subestructura se conoce como el modelo esclavo (Slave), que también es un modelo numérico, y representa la subestructura experimental. Para las simulaciones FE-coupled en este estudio, los modelos maestro y esclavo están modelados en OpenSEES e interactúan juntos a través de OpenFresco.

Este enfoque proporciona la ventaja importante de que todos los códigos conectados se ejecutan continuamente sin la necesidad de apagar y reiniciar, disminuyendo el tiempo de análisis significativamente. La velocidad de simulaciones acopladas donde se usan uno o más códigos de software FE juntos, incluidos los casos en los que se incorporan subestructuras físicas en el modelo FE para una prueba de simulación híbrida, son al menos un orden de magnitud más rápido que los enfoques convencionales que se basan en apagar y reiniciar el análisis en los programas acoplados en cada paso del tiempo de integración.

Para realizar estas simulaciones se debe utilizar el control experimental SimFEAdapter. SimFEAdpater permite a los usuarios simular subconjuntos físicos y muestras en cualquier software de elementos finitos de su elección. Los elementos adaptadores que actúan como interfaces para el software esclavo de elementos finitos, en el que se simulan los subconjuntos, se han implementado hasta ahora para OpenSees, LS-DYNA y Abaqus. Recomiendo hacer uso del elemento experimental generic en el master, y para acoplar, el elemento adapter en el slave.

Los ensayos Client-Server y FE-coupled se utilizan de forma previa a realizar los ensayos en laboratorio con la finalidad de verificar la técnica de subestructuración, validar el esquema de integración y pasos de tiempo para la prueba real.

2.3. Dinámica del sistema de transferencia

2.3.1. Modelos de Actuadores Servo-Hidráulicos

Los actuadores servo-hidráulicos se utilizan a menudo como sistema de transferencia en RTHS debido a su capacidad para imponer desplazamientos en las frecuencias de interés para la evaluación sísmica. En la literatura podemos encontrar disponibles muchos modelos lineales y no lineales para actuadores servo-hidráulicos (Dyke et al., 1995; Carrion y Spencer, 2007; Maghareh et al., 2018; Silva et al., 2020). En este estudio se describen a continuación los dos modelos de actuadores estudiados.

2.3.1.1. Carrion & Spencer (2007)

Este modelo fue desarrollado por Spencer et al. (2007) y es equivalente a los modelos presentados por Dyke et al. (1995) y Maghareh et al. (2018). Estos modelos se basan en las mismas ecuaciones físicas, pero presentan variaciones en como los modelos agrupan los diagramas de bloques. El modelo seleccionado incluye representaciones matemáticas de los diferentes componentes del sistema de prueba, incluido el actuador, la dinámica de la servo-válvula, el flujo de la servo-válvula y el controlador. El diagrama de bloques de este sistema servo-hidráulico junto con el especimen físico se presenta en la Figura 2.18.

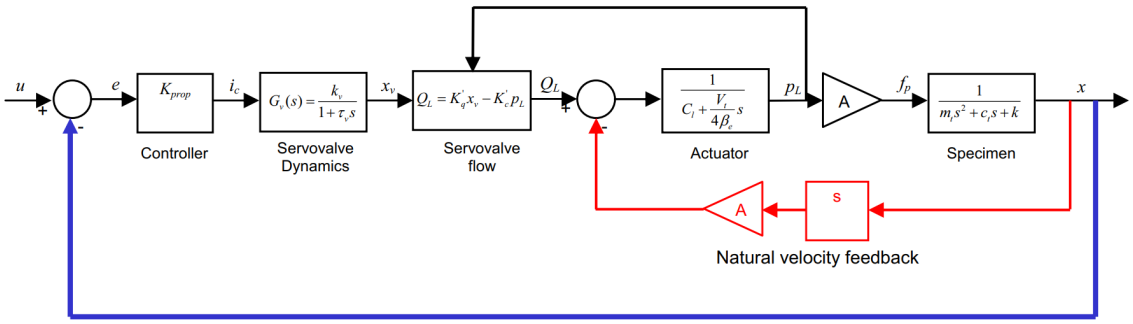


Figura 2.18: Modelo de actuador servo-hidráulico acoplado con un especimen físico

La entrada del sistema de transferencia es el desplazamiento objetivo u, y luego, el error e entre el desplazamiento real x y u, es decir, $e = u - x$, se envía al controlador interno del sistema servohidráulico para generar la entrada i_c de la servo-válvula:

$$i_c = K_p e = K_p(u - x) \tag{2.17}$$

donde K_p es la ganancia del controlador proporcional interno. En algunos sistemas servo-hidráulicos, en lugar del control proporcional, se implementa un control proporcional-integral-derivativo (PID). En esos casos, la entrada de la servoválvula se obtiene mediante:

$$i_c(s) = \left(K_p + \frac{K_i}{s} + K_d s\right) \tag{2.18}$$

donde K_i es la ganancia integral y K_d es la ganancia derivada. Sin embargo, en esta sección, solo se considera el control proporcional. Este controlador interno no debe confundirse con el método de compensación dinámica implementado en un controlador externo. El controlador interno es parte del sistema servo-hidráulico, mientras que el controlador externo se usa generalmente para compensar la dinámica servo-hidráulica, la cuál incluye su control interno.

Una vez que se genera la entrada de la servo-válvula, la dinámica de la servo-válvula se puede modelar como una función de transferencia $G_v(s)$ que genera el desplazamiento de la válvula x_v:

$$x_v(s) = G_v(s) i_c(s) = \frac{K_v}{1 - \tau s} i_c(s) \tag{2.19}$$

donde k_v es la ganancia de la servo-válvula y τ es la constante de tiempo de la servo-válvula. Entonces, la ecuación de continuidad que gobierna el flujo de la servo-válvula:

$$Q_L = K_q x_v - K_c' p_L \tag{2.20}$$

donde K_q es la ganancia de flujo de la válvula, K_c' es la ganancia de presión de flujo de la válvula y p_L es la caída de presión. Por otro lado, el flujo en el actuador se define por:

$$Q_L = A\dot{x} + C_l p_L + \frac{V}{4\beta}\dot{p}_L \tag{2.21}$$

donde A es el área del pistón del actuador, C_l es el coeficiente de fuga en el pistón, V es el volumen total de fluido y β es el módulo volumétrico del fluido. Entonces, se puede obtener un modelo de función de transferencia del actuador:

$$p_L = \frac{1}{C_l + \frac{V}{4\beta}s}\left(Q_L(S) - Asx(s)\right) \tag{2.22}$$

Alternativamente, combinando las ecuaciones de continuidad 2.20 y 2.21, se obtiene una ecuación para la fuerza del pistón del actuador f_p:

$$f_p = Ap_L \tag{2.23}$$

$$\dot{f}_p = \frac{4\beta}{V}\left(AK_q k_v i_c - (K_c' + C_l)f_p - A^2\dot{x}\right) \tag{2.24}$$

Es importante notar que la fuerza del actuador se ve directamente afectada por la respuesta física de la muestra,específicamente, por la retroalimentación de velocidad. Finalmente, la respuesta de la muestra se rige por la fuerza de equilibrio entre el pistón del actuador y la ecuación de movimiento para un especimen físico lineal queda.

$$m\ddot{x} + c\dot{x} + kx = f_p \tag{2.25}$$

donde m es la masa total del especimen físico y el pistón, c es el coeficiente de amortiguamiento viscoso del especimen y el pistón, y k es la rigidez del especimen. La ecuación de movimiento del espécimen físico en la ecuación 2.25 se puede expresar en el dominio de Laplace:

$$\frac{1}{ms^2 + cs + k}f_p(s) \tag{2.26}$$

A continuaciónn, en la Figura 2.19 se aprecia la implementación de este modelo en el ambiente Matlab/Simulink.

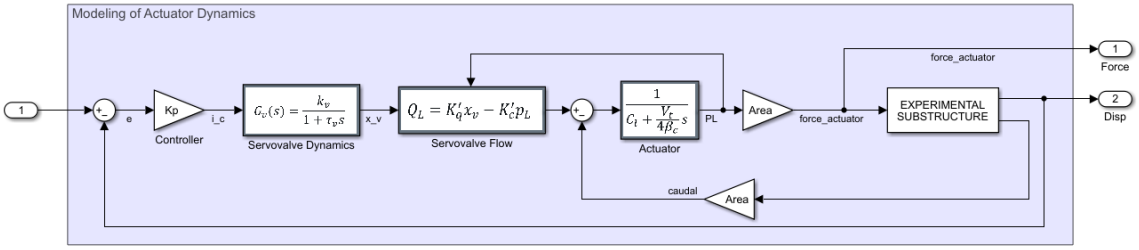


Figura 2.19: Implementación en Simulink del Modelo de actuador servo-hidráulico acoplado con un especimen físico.

Los parámetros utilizados para este modelo son los entregados por Carrion y Spencer (2007). Se muestran a continuación en la tabla 2.1.

Tabla 2.1: Parámetros del modelo del actuador. (Carrion y Spencer, 2007)

Componentes	Parámetros	Valor	Unidad	Descripción
Controlador	K_p	3	mA/in	Ganancia proporcional
Servo-válvula	τ_v	0,00332	s	Constante de tiempo de la servoválvula
	K_q	23,01	$in^3/s/mA$	Ganancia de la válvula
	K'_c	$1{,}36x10^{-5}$	$in^3/s/psi$	Ganancia de presión de flujo de la válvula
Actuador	A	0,751	in^2	Área del pistón
	C_l	$5{,}89x10^{-6}$	$in^3/s/psi$	Coeficiente de fuga del actuador
	V_t	44,66	in^3	Volumen total de fluido en las cámaras del actuador
	β_e	95958	psi	Módulo de volumen de aceite efectivo

2.3.1.2. MTS (Schellenberg, 2018)

MTS Systems Corporation ha desarrollado un modelo detallado de Simulink para ser utilizado en los actuadores servohidráulicos fabricados por MTS para diferentes actuadores estáticos y dinámicos (Schellenberg, 2018). Este modelo incluye un modelo C MEX con una S-Function para el conjunto actuador/válvula, un modelo Simulink para el acumulador y también un modelo Simulink para la carga útil. La implementación de este modelo se muestra a continuación en la Figura 2.20. Aquí apreciamos de igual forma que el actuador presentado en la sección 2.3.1.1 se realiza el control del actuador mediante control proporcional.

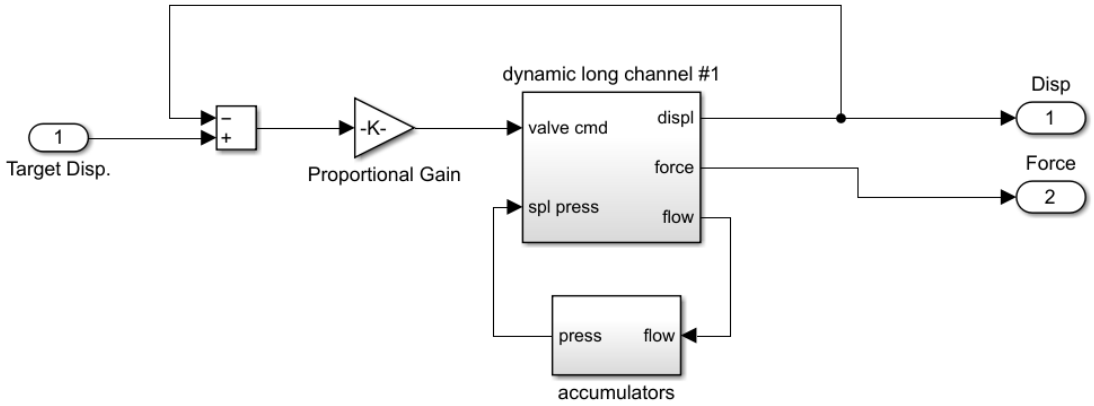


Figura 2.20: Implementación en Simulink del Modelo de actuador servo-hidráulico largo de MTS.

MTS proporciona tres modelos de actuadores hidráulicos los cuáles clasifica como modelo corto, largo y estático. Las principales diferencias entre estos se muestran en la Tabla 2.2.

Tabla 2.2: Parámetros de los modelos de actuador de MTS.

Modelo	Corto		Largo		Estático		Unidad
Rango Desplazamiento	-22	22	-42	42	-72	72	$[in]$
Rango Velocidad	-22.0808	22.0808	-30.3109	30.3109	-1.3408	0.8844	$[in/s]$
Rango Fuerza	-222.725	222.725	-162.25	162.25	-220.07	333.645	$[kips]$
Frecuencia de la columna de aceite	39.9936		32.0214		31.9473		$[Hz]$
Área Pistón	55		75.5		93.85		$[in]$

Se aprecia que tanto el actuador corto como el largo presentan capacidad para RTHS, no así el actuador

denominado estático, el cuál está pensado para ensayos de carga lenta o pseudo-estáticos. Los parámetros necesarios, junto con la implementación en Simulink se encuentra disponible en el código fuente de OpenFresco y es posible descargarlo desde el GitHub personal del PhD. Andreas Schellenberg (Schellenberg, 2018). Puede obtener el código fuente de OpenFresco directamente en GitHub realizando un fork o descargando el código fuente de OpenFresco desde el repositorio directamente en su computadora.

2.4. Métodos de compensación

Se han desarrollado varios procedimientos para compensar el retraso en simulaciones híbridas. En esta sección, se revisan los procedimientos populares. En general, estos procedimientos se pueden clasificar en dos grupos (Ahmadizadeh et al., 2008): (i) procedimientos que modifican o compensan el desplazamiento comandado y (ii) procedimientos que corrigen la fuerza medida.

2.4.1. Métodos de modificación de desplazamiento comandado

En un procedimiento de compensación de retraso típico que se muestra en la Figura 2.21, el desplazamiento comandado en el tiempo t, denotado por x', se predice antes del desplazamiento deseado x_1 por el retardo esperado, τ. Dado que se envía un desplazamiento comandado predicho al actuador, se espera que la respuesta del actuador esté próxima al desplazamiento deseado x_1 al final del paso de tiempo de integración numérica, Δt.

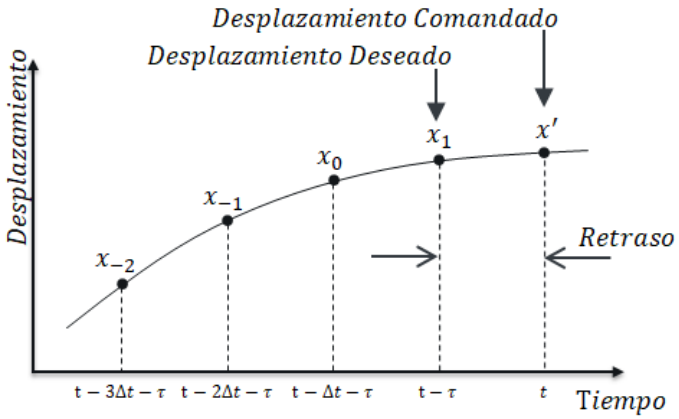


Figura 2.21: Extrapolación general de desplazamiento comandado.

El procedimiento de compensación de retraso más utilizado para la simulación híbrida fue introducido por Horiuchi et al. (1999). En su procedimiento recomendado, se ajusta un polinomio de tercer orden a los últimos cuatro desplazamientos deseados y se utiliza para predecir el desplazamiento de comando antes del tiempo de simulación mediante extrapolación al polinomio ajustado. Horiuchi y Konno (2001a) propusieron un algoritmo de predicción diferente asumiendo una variación lineal de aceleración, que también proporciona una variación de desplazamiento de tercer orden. Dado que la mayoría de los procedimientos de integración que se utilizan actualmente en la simulación híbrida son explícitos (por ejemplo, el método beta de Newmark (1959)), la velocidad y la aceleración en el tiempo de desplazamiento deseado $t-\tau$ permanecen desconocidas hasta que esté disponible la retroalimentación de fuerza en el desplazamiento deseado. Por lo tanto, el desplazamiento comandado debe determinarse con base en los estados en los tiempos $t-\tau-\Delta t$ y $t-\tau-2\Delta t$, utilizando las siguientes ecuaciones:

$$\ddot{x}' = \ddot{x}_0 + \frac{\Delta t + \tau}{\Delta t}(\ddot{x}_0 - \ddot{x}_{-1}) = \left(2 + \frac{\tau}{\Delta t}\right)\ddot{x}_0 - \left(1 + \frac{\tau}{\Delta t}\right)\ddot{x}_{-1} \tag{2.27}$$

$$x' = x_0 + (\Delta t + \tau)\dot{x}_0 + \frac{1}{3}(\Delta t + \tau)^2\ddot{x}_0 + \frac{1}{6}(\Delta t + \tau)^2\ddot{x}' \tag{2.28}$$

donde x, $\dot{x}$ y $\ddot{x}$ son desplazamiento, velocidad y aceleración, respectivamente, y los subíndices denotan el número de paso de integración. El subíndice cero corresponde al paso de tiempo anterior, cuya velocidad y aceleración se conocen, y las variables primarias indican una cantidad predicha.

Alternativamente, el desplazamiento comandado compensado por retraso x' se puede determinar usando el mismo procedimiento de integración numérica usado para determinar x_1 a partir de los estados disponibles en el tiempo $t - \tau - \Delta t$. Por ejemplo, usando el método explícito de Newmark $\left(\beta = 0, \gamma = \frac{1}{2}\right)$, las siguientes ecuaciones cinemáticas suponiendo una aceleración constante se pueden usar para predecir los desplazamientos deseados y comandado:

$$x_1 = x_0 + \Delta t\dot{x}_0 + \frac{1}{2}\Delta t^2\ddot{x}_0 \tag{2.29}$$

$$x' = x_0 + (\Delta t + \tau)\dot{x}_0 + \frac{1}{2}(\Delta t + \tau)^2\ddot{x}_0 \tag{2.30}$$

Cabe señalar que la ecuación 2.30 se puede derivar de la ecuación 2.28 para el caso de aceleración constante. Es decir, la ecuación 2.28 tiene un término adicional dado por:

$$x'_{Eq.2,28} - x'_{Eq.2,30} = \frac{1}{6}(\Delta t + \tau)^3\frac{\ddot{x}_0 - \ddot{x}_{-1}}{\Delta t} \approx \frac{1}{6}(\Delta t + \tau)^3\dddot{x}_{0,-1} \tag{2.31}$$

que es un término de tercer orden, que resulta del supuesto de variación de aceleración lineal.

Sin embargo, la sincronización del desplazamiento es muy difícil de lograr en tiempo real. El sistema de transferencia es un sistema dinámico en sí mismo y, además, interactúa con la muestra física, produciendo discrepancias entre los desplazamientos ordenados y medidos. Este fenómeno se conoce como interacción control-estructura (Dyke et al., 1995). Por lo tanto, para reducir el error de sincronización, se implementa un controlador entre la subestructura numérica y el sistema de transferencia. En la Figura 2.22, se esquematiza el problema de seguimiento. Un desplazamiento objetivo x_t proviene de la subestructura numérica y se procesa en el controlador para generar la señal de comando x_c para el sistema de transferencia con la subestructura experimental adjunta. El subsistema compuesto por el sistema de transferencia y el espécimen físico se denomina aquí como planta de control. Finalmente, se mide el desplazamiento x_m logrado y se envía de vuelta al controlador. El objetivo de la compensación dinámica es reducir el error de sincronización $e = x_t - x_m$, y se han propuesto varios métodos en la literatura.

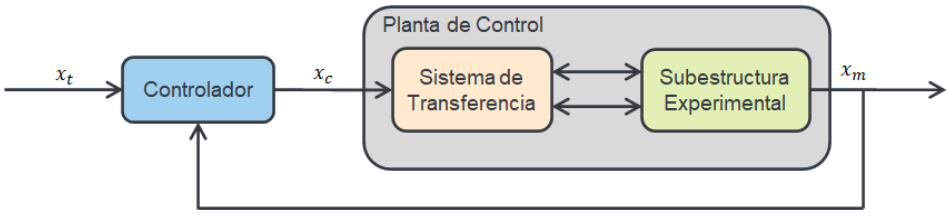


Figura 2.22: Problema de seguimiento en RTHS.

Los errores de sincronización se pueden clasificar como errores de amplitud o retraso. En 1996, Horiuchi et al. (1995). demuestran que el retardo del actuador es equivalente a introducir amortiguamiento negativo al sistema híbrido (Horiuchi et al., 1995). Si el amortiguamiento negativo supera el amortiguamiento inherente del sistema, la respuesta se vuelve inestable, es decir, los desplazamientos tienden a crecer sin límite. Por lo tanto, los métodos de compensación anteriores sirven para compensar el retraso del actuador. El método más utilizado se conoce como extrapolación de polinomios (Horiuchi et al., 1995, 1999). En

este método, el retraso del actuador se asume como un valor constante, por lo que la respuesta del actuador después del retraso de tiempo se estima a través de la extrapolación de polinomios, luego, en lugar de ordenar el desplazamiento deseado, se ordena el desplazamiento estimado y, debido al retraso, la respuesta lograda es cercana a la deseada

Sin embargo, el tiempo de retraso de un actuador no es un valor constante, depende de las frecuencias comandadas. En consecuencia, se implementaron métodos más sofisticados. Chen (2007) propuso la utilización de la función de transferencia discreta de primer orden para modelar la dinámica del actuador y compensarla a través de la compensación inversa (Chen, 2007). Otro enfoque similar basado en la función de transferencia de primer orden es el compensador de adelanto de fase implementado por Zhao et al. (2003).

Las estrategias de control más complejas son las técnicas de compensación basadas en modelos, donde se aplican diferentes herramientas de control basadas en un modelo de orden superior del actuador conectado a la subestructura experimental (Carrion y Spencer, 2007). Phillips y Spencer, Jr. (2011) propusieron un esquema de compensación de retroalimentación hacia adelante. En este método, el controlador feedforward se basa en el inverso de la planta de control, por lo que su función es compensar la dinámica modelada de la planta y solo toma como entrada el desplazamiento deseado. Por otro lado, en paralelo, un controlador de retroalimentación toma el error de seguimiento para generar una señal de comando, mejorando la robustez de la compensación. El controlador de retroalimentación consta de un regulador gaussiano cuadrático lineal (LQG por su nombre en inglés, Linear Quadratic Gaussian). El controlador LQG está compuesto por un filtro Kalman diseñado para estimar los estados de la planta de control y un Regulador Cuadrático Lineal diseñado para minimizar el error de seguimiento.

Las estrategias basadas en modelos han demostrado un excelente desempeño en RTHS, especialmente si se dispone de un modelo preciso de la planta de control. Sin embargo, la incertidumbre en el modelo identificado puede afectar la precisión del controlador. Además, las estrategias basadas en modelos generalmente se basan en sistemas lineales invariantes en el tiempo, mientras que las propiedades reales de la planta de control pueden exhibir comportamientos no lineales o propiedades variables en el tiempo durante la prueba. Por este motivo, los investigadores han aplicado técnicas de control más sofisticadas, como el control no lineal y el control adaptativo.

Xu et al. (2019) propusieron el uso de la técnica no lineal llamada control de modo deslizante (SMC por su nombre en inglés, Sliding Mode Control) en combinación con el predictor de avance adaptativo basado en polinomios mejorados (IAPF) (Xu et al., 2019). Khalil (2002) también introdujo el uso del control de modo deslizante en el sistema de control robusto autoajustable propuesto (SRCSys por su nombre en inglés, Self-tuning Robust Control System) propuesto, donde el control consta de dos capas, una capa de robustez basada en control de modo deslizante y una capa de adaptación para mejorar la precisión durante un prueba. El Control de Modo Deslizante consiste en obligar a la planta de control a alcanzar y mantener una variedad deslizante en un tiempo finito. La variedad está diseñada para lograr el objetivo de control (Khalil, 2002). El Control de Modo Deslizante es una herramienta de control robusta y flexible, sin embargo, su diseño se basa en la función de Lyapunov para garantizar la convergencia asintótica, por lo tanto, para cumplir con los requisitos de RTHS en poco tiempo, la ley de control debe calibrarse cuidadosamente.

El control adaptativo consiste en una combinación de ley de control y una ley adaptativa. La ley de control genera la señal de comando tomando el objetivo y los desplazamientos de medición, y depende de un conjunto de parámetros de control. Por otro lado, la ley adaptativa ajusta los parámetros de control utilizando la información obtenida durante la prueba (Ioannou y Baldi, 2010).

A continuación se describen métodos eficaces y sencillos que pueden clasificarse como feedforward adaptativo. La Serie Temporal Adaptativa (ATS por su nombre en inglés Adaptive Time Series) propuesta por Chae et al. (2013) consiste en aproximar la relación entrada-salida de la planta de control como una expansión en serie de Taylor. El orden del controlador depende del número de coeficientes en la expansión, pero los autores recomiendan dos o tres términos. La adaptación se basa en el método de mínimos cuadrados y se implementa utilizando los datos medidos y ordenados durante la prueba. Aunque el método de mínimos cuadrados proporciona una excelente estimación de los parámetros requeridos, su cálculo involucra la inversión de una matriz en cada paso de tiempo, lo que puede ser un problema restrictivo considerando los

pequeños pasos de tiempo usados en RTHS. Por ello, una modificación de la ATS fue propuesta por Palacio-Betancur y Gutierrez Soto (2019) nombrada como Serie Temporal Adaptativa Condicional (CATS por su nombre en inglés, Conditional Adaptive Time Series). La arquitectura de control y adaptación es equivalente al ATS, pero en lugar del método convencional de mínimos cuadrados, se introdujo el método recursivo de mínimos cuadrados. El método recursivo es más rentable, pero requiere la selección de parámetros de diseño menos intuitivos.

Por otro lado, Chen et al. (2015) propusieron el compensador basado en el modelo adaptativo (AMB por su nombre en inglés, Adaptive Model-based), que consiste en un feedforward adaptativo combinado con un controlador LQG convencional. El feedforward adaptativo consiste en el inverso de un modelo de dominio de frecuencia de la planta de control. Después de la implementación, la arquitectura resulta muy similar a la ATS y CATS, pero con una ley adaptativa diferente. En este caso, se define una ley adaptativa de gradiente.

Los métodos de feedforward adaptativos anteriores (ATS, CATS y AMB) se formulan en base a modelos dinámicos de tiempo continuo, pero para la implementación de controladores digitales, se requieren formas discretas. Como alternativa, algunos métodos se formulan directamente en tiempo discreto. Wang et al. (2020) propusieron el compensador adaptativo de dos etapas, donde el retraso estimado se compensa a través de la extrapolación polinomial clásica, mientras que el retraso no compensado o la dinámica no modelada se compensan utilizando un enfoque de avance adaptativo discreto con adaptación basada en el método recursivo de mínimos cuadrados.

2.4.1.1. Compensación Adaptiva

La clave de RTHS es que los desplazamientos ordenados se imponen a la muestra de prueba mediante actuadores hidráulicos en tiempo real. Durante la prueba, la comunicación, el control y la dinámica del actuador servohidráulico provocan un retraso de tiempo y un retraso entre los comandos de desplazamiento y la medición. Estos retrasos y retrasos pueden introducir una amortiguación negativa en la simulación híbrida, lo que puede generar inexactitudes e inestabilidades potenciales.

Por tanto, de todos los métodos revisados para manejar y resolver estos problemas, se decidió utilizar una Compensación basada en modelos adaptativos (AMBC por su nombre en inglés Adaptive model-based compensation). Las dos metodologías que se describen a continuación fueron desarrolladas e implementadas por el M.Sc. Cristóbal Gálmez Villaseca (Gálmez, 2021).

La arquitectura básica de la compensación basada en modelos adaptativos se presenta en la Figura 2.23, donde el desplazamiento a imponer se denomina como desplazamiento objetivo x_t. Un control anticipativo que depende de un conjunto de parámetros A toma el desplazamiento objetivo para generar la señal a comandar x_c para la planta de control. En consecuencia, la planta de control logra un desplazamiento medido x_m. El objetivo es minimizar el error de sincronización $e = x_m - x_t \rightarrow 0$.

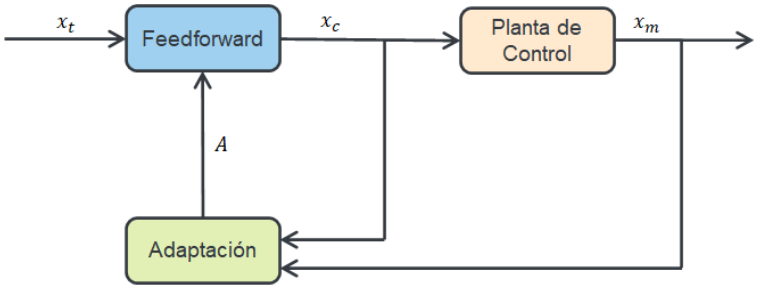


Figura 2.23: Arquitectura de control de la metodología propuesta.

2.4.1.1.1 Metodología de compensación basada en modelos adaptativos (AMB)

Esta primera metodología recibe el nombre de Compensación basada en modelos adaptativos (AMBC) y se basa en el método basado en el modelo adaptativo presentado por Chen et al. (2015) con las modificaciones presentadas por Fermandois et al. (2020). La formulación consiste en estimar la planta mediante una función de transferencia de tercer orden sin ceros, como se muestra en la Ec. (2.32).

$$x_m = G_p(s)x_c = \left(\frac{1}{a_3 s^3 + a_2 s^2 + a_1 s + a_0}\right) x_c \tag{2.32}$$

donde x_m es el desplazamiento medido por los sensores. La inversa de esta función de transferencia se usa para generar la señal de comando usando el desplazamiento del objetivo como entrada, como se muestra en la Ec. (2.33). Las derivadas de x_t se obtienen con diferencias finitas.

$$x_c = G_p^{-1} x_t = (a_3 s^3 + a_2 s^2 + a_1 s + a_0)x_t = a_3 \dddot{x}_t + a_2 \ddot{x}_t + a_1 \dot{x}_t + a_0 x_t \tag{2.33}$$

Los parámetros adaptativos $a_i, i = 0, 1, 2, 3$, deben ajustarse durante la prueba para lograr una buena compensación. Esta adaptación se formula a través de un modelo de ecuación diferencial proporcionado en la Ec. (2.34), donde $x_m^{(i)}$ es la i-ésima derivada de x_m asociada con el parámetro adaptativo a_i.

$$\dot{a}_i = \gamma_i \left(\frac{x_c - [a_3 a_2 a_1 a_0][\dddot{x}_m \ddot{x}_m \dot{x}_m x_m]^T}{1 + [\dddot{x}_m \ddot{x}_m \dot{x}_m x_m][\dddot{x}_m \ddot{x}_m \dot{x}_m x_m]^T} \right) x_m^{(i)} \tag{2.34}$$

En este método, un filtro Butterworth elimina el ruido de alta frecuencia. El filtro se aplica a la señal ordenada para sincronizar con la señal medida filtrada. Esta metodología requiere la calibración de ganancias adaptativas. Para ello se sigue el procedimiento dado en el artículo original (Fermandois et al., 2020), pero con ciertas modificaciones: (i) aceleración del suelo escalada al 100 %; (ii) múltiples estructuras de un solo grado de libertad con un período natural entre 0,6 y 4[s]; y (iii) la masa, la rigidez, la amortiguación y las condiciones iniciales del controlador adaptativo se generaron aleatoriamente mediante el método del hipercubo latino para cubrir una gama más amplia de estructuras, obteniendo ganancias más robustas. La calibración de las ganancias adaptativas óptimas consiste en un esquema de optimización global a través de la optimización del enjambre de partículas, que minimiza el indicador de desempeño J_2 explicado más adelante en la Sección 4.1. Además, las condiciones iniciales se definen desde el actuador sin muestra.

La implementación del compensador AMB se resume en el diagrama de flujo de la Figura 2.24.

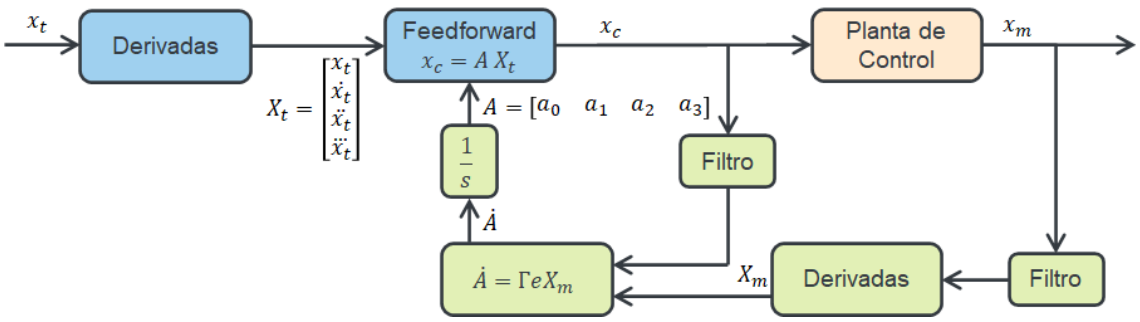


Figura 2.24: Diagrama de flujo del compensador basado en modelo adaptativo (AMB).

2.4.1.1.2 Metodología de compensación discreta basada en modelo adaptativo (dAMB)

El método de compensación dinámica presentado en esta sección se basa en el modelo adaptativo basado en Chen et al. (2015) con las modificaciones presentadas en Fermandois et al. (2020), pero la implementación en forma discreta se basa en el trabajo presentado por Wang et al. (2020).

La planta de control se puede representar en el dominio de la frecuencia mediante una función de transferencia sin ceros y un número limitado de polos como se muestra en la Ec. (2.35):

$$x_m = G_p(s)x_c = \frac{1}{a_3 s^3 + a_2 s^2 + a_1 s + a_0} x_c \tag{2.35}$$

donde a_i con $i = 0, 1, 2, 3$ son los coeficientes de la función de transferencia $G_p(s)$; s es la variable de Laplace y x_m es el desplazamiento medido. Para compensar la dinámica de la planta de control, se implementa el inverso de $G_p(s)$ para generar la señal de comando a partir del desplazamiento objetivo como se muestra en la Ec. (2.36):

$$x_c = G_p^{-1}(s)x_t = (a_3 s^3 + a_2 s^2 + a_1 s + a_0)x_t \tag{2.36}$$

donde x_t es el desplazamiento objetivo de la subestructura numérica. Observe que la función de transferencia presentada en la ecuación. (2.36) es incorrecto, sin embargo, el controlador se puede implementar usando derivadas en diferencias finitas como se muestra en la Ec. (2.37):

$$\begin{bmatrix} x[i] \\ \dot{x}[i] \\ \ddot{x}[i] \\ \dddot{x}[i] \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/\Delta t & -1/\Delta t & 0 & 0 \\ 1/\Delta t^2 & -2/\Delta t^2 & 1/\Delta t^2 & 0 \\ 1/\Delta t^3 & -3/\Delta t^3 & 3/\Delta t^3 & -1/\Delta t^3 \end{bmatrix} \begin{bmatrix} x[i] \\ x[i-1] \\ x[i-2] \\ x[i-3] \end{bmatrix} \tag{2.37}$$

donde Δt es el período de muestreo de la simulación y $x[i]$ es un desplazamiento en el paso de tiempo i. Por lo tanto, la señal de comando para el actuador se puede obtener de la Ec. (2.38):

$$x_c = [a_0\ a_1\ a_2\ a_3][x_t\ \dot{x}_t\ \ddot{x}_t\ \dddot{x}_t]^T \tag{2.38}$$

Luego, para la implementación, las ecuaciones (2.37) y (2.38) se combinan para generar el filtro FIR presentado en Eq. (2.39):

$$x_c[i] = F^T X_t^{tap} = [f_0\ f_1\ f_2\ f_3] \begin{bmatrix} x_t[i] \\ x_t[i-1] \\ x_t[i-2] \\ x_t[i-3] \end{bmatrix} \tag{2.39}$$

donde $x_c[i]$ es la señal de comando en el paso de tiempo i; $F = [f_0\ f_1\ f_2\ f_3]^T$ son los coeficientes del filtro FIR y $X_t^{tap} = [x_t[i]\ x_t[i-1]\ x_t[i-2]\ x_t[i-3]]^T$ son los desplazamientos objetivo de retardos marcados.

Dado que el controlador inicial no considera la interacción con la subestructura experimental, los parámetros de control $F = [f_0\ f_1\ f_2\ f_3]^T$ se actualizan durante la prueba con el método recursivo de mínimos cuadrados. La señal ordenada x_c puede estimarse mediante la ecuación. (2.40):

$$\hat{x}_c[i] = F^T X_m^{tap} = [f_0\ f_1\ f_2\ f_3] \begin{bmatrix} x_m[i] \\ x_m[i-1] \\ x_m[i-2] \\ x_m[i-3] \end{bmatrix} \tag{2.40}$$

donde $\hat{x}_c$ es la señal de comando estimada y $X_m^{tap} = [x_m[i]\ x_m[i-1]\ x_m[i-2]\ x_m[i-3]]^T$ son los retrasos marcados del desplazamiento medido. Por lo tanto, con la ecuación. (2.40) y la señal de comando x_c se puede obtener un error de estimación indirecto de los parámetros f_i como se muestra en la Ec. (2.41):

$$e = x_c - F^T X_m^{tap} \tag{2.41}$$

donde e es el error estimado de los parámetros identificados F. Además, dado que el ruido en el desplazamiento medido puede afectar el proceso de adaptación, la señal x_m es filtrada por un filtro Butterworth. Por lo tanto, la señal de comando x_c también se filtra para sincronizar con la señal medida filtrada x_m. Luego, los parámetros f_i se actualizan con la Eq. (2.42):

$$F[i] = F[i-1] + \frac{P[i-1]X_m^{tap}}{\rho + (X_m^{tap})^T P[i-1] X_m^{tap}} e \tag{2.42}$$

donde ρ es el factor de olvido y P es una matriz de covarianza relacionada con los datos X_m^{tap} que se actualiza de acuerdo con la Ec. (2.43):

$$P[i] = \frac{1}{\rho}\left[I - \frac{P[i-1]X_m^{tap}}{\rho + (X_m^{tap})^T P[i-1] X_m^{tap}} (X_m^{tap})^T\right] P[i-1] \tag{2.43}$$

donde se debe definir la matriz de covarianza inicial P para inicializar el método RLS.

La implementación del compensador dAMB se resume en el diagrama de flujo de la Fig. 2.25.

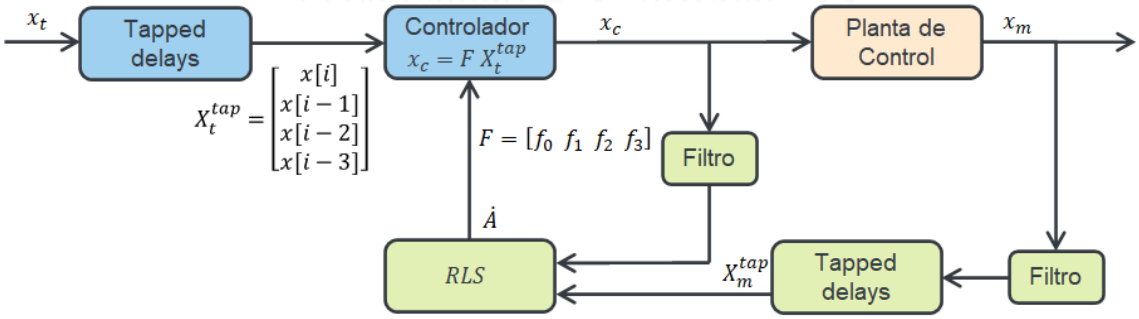


Figura 2.25: Diagrama de flujo del compensador basado en modelo adaptativo discreto (dAMB).

La matriz de covarianza inicial P_0 tiene un gran impacto en la capacidad de adaptación del compensador. Los valores pequeños pueden provocar una adaptación lenta y los valores extremadamente altos pueden provocar una adaptación inestable. La matriz de covarianza inicial P_0 se selecciona típicamente en la literatura como una matriz diagonal. En este trabajo,la siguiente matriz es utilizada:

$$P_0 = 10^{10} I_{4\times4} \tag{2.44}$$

donde I es la matriz identidad.

2.4.2. Métodos de correción de la fuerza

Como alternativa a los procedimientos de modificación del desplazamiento comandado, o en combinación con ellos, se pueden aplicar correcciones a las señales de fuerza medidas. La medición precisa de las fuerzas es importante porque este valor se retroalimenta al modelo numérico. La compensación de retraso se puede realizar en las señales de medición de fuerza utilizando los mismos procedimientos de extrapolación polinomial descritos en la sección anterior para los desplazamientos. Sin embargo, es más difícil ajustar polinomios a los datos de fuerza medidos debido a la contaminación por ruido.

Se considera un enfoque diferente para la corrección de la fuerza medida buscando el tiempo en el que se logra el desplazamiento deseado y su fuerza correspondiente dentro de los datos medidos. Como se

muestra en la Figura 2.26, dos polinomios de segundo orden se ajustan a las últimas medidas de fuerza y desplazamiento en el sistema de coordenadas del actuador, como:

$$r(t) = a_r t^2 + b_r t + c_r \tag{2.45}$$

$$u(t) = a_u t^2 + b_u t + c_u \tag{2.46}$$

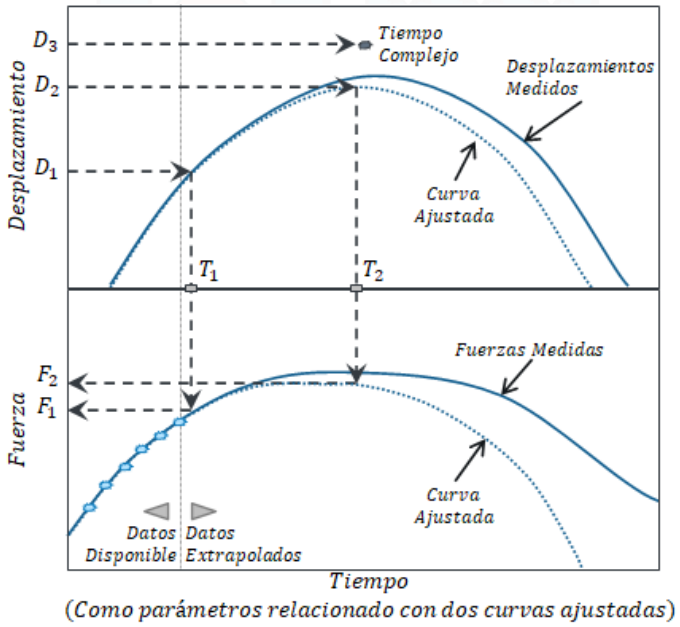


Figura 2.26: Estimación de la fuerza correspondiente al desplazamiento deseado mediante el ajuste de curvas de las medidas. (Ahmadizadeh et al., 2008)

Usando el polinomio de desplazamiento, el tiempo correspondiente al desplazamiento deseado u_n^d se puede determinar como:

$$t_d = \frac{-b_u \pm \sqrt{b_u^2 - 4a_u(c_u - d_n^d)}}{2a_u} \tag{2.47}$$

a partir del cual se elegirá el valor más cercano al tiempo de paso actual. La fuerza se puede encontrar reemplazando el tiempo mencionado anteriormente en el polinomio de fuerza dado por la ecuación 2.45. Hay que tener en cuenta que si la demora en una prueba de desplazamiento controlado se compensa en exceso, este procedimiento solo interpolará los datos de fuerza medidos. Este procedimiento elimina la necesidad de una estimación exacta del retraso, ya que busca directamente la fuerza correspondiente al instante en que se logran los desplazamientos deseados. Como resultado, este procedimiento se puede combinar con un procedimiento de extrapolación de desplazamiento para compensar los errores de seguimiento antes de que el retraso estimado alcance el valor real.

Usar el tiempo para relacionar los dos polinomios ajustados tiene varias ventajas. Primero, dado que los puntos de datos medidos están igualmente espaciados en el tiempo, la determinación de los coeficientes polinomiales ajustados es computacionalmente eficiente. Además, los efectos de las no linealidades de la muestra serán menos pronunciados en los historiales de tiempo en comparación con las curvas de fuerza-desplazamiento, lo que proporciona un ajuste de curvas de mejor calidad.

Claramente, el error de predicción usando los polinomios ajustados aumenta cuando se extrapola más de los puntos ajustados. Para minimizar los errores, se pueden imponer límites a la variación del tiempo dentro de un paso para evitar extrapolaciones extremadamente grandes. El límite de extrapolación debe seleccionarse lo suficientemente grande para compensar el retraso máximo esperado en una simulación.

Un problema en el uso de procedimientos de compensación de fuerza es que la Ecuación 2.47 puede resultar en un valor de tiempo complejo, particularmente durante el subimpulso del desplazamiento en las inversiones de desplazamiento (Figura 2.26). Al insertar el tiempo resultante con valores complejos en el polinomio de fuerza ajustado, se obtiene un valor de fuerza complejo. Si el desplazamiento deseado permanece cerca del pico de desplazamiento medido, la parte imaginaria de la fuerza compleja resultante es pequeña y su parte real aumenta proporcionalmente más allá del pico de fuerza con el componente complejo. En consecuencia, el valor absoluto de la fuerza compleja, que es aproximadamente igual a su parte real, proporciona una estimación adecuada de la fuerza. La aplicación de este procedimiento de corrección de fuerza en pasos con componentes de fuerza imaginarios pequeños ha demostrado ser ventajosa para minimizar errores experimentales en comparación con simulaciones sin correcciones en pasos con tiempo de valor complejo.

Para mitigar la suboscilación sistemática de los desplazamientos durante las inversiones de desplazamiento, se puede aplicar un “gain” al desplazamiento comandado compensado por retraso. La ganancia se puede calcular en línea en base a comparaciones previas de picos de desplazamiento medidos y deseados con promediado de ventana móvil para suavizar los resultados. Dado que la fuerza de restauración de la muestra adjunta tiende a mover el pistón del actuador de regreso a su punto de equilibrio, la ganancia de la señal comandada se aplica a la posición del pistón con respecto a este punto, denominado desplazamiento cero. Como el desplazamiento cero de una muestra no lineal puede derivar de su posición original, el punto de equilibrio de fuerza cero también se determina en tiempo real.

Finalmente, cabe mencionar que las mediciones de fuerza contienen una cantidad significativa de ruido. La mayor parte de este ruido se puede eliminar mediante el uso de filtros de paso bajo adecuados a expensas de un retraso adicional del sistema. En cualquier caso, usar el número mínimo de puntos para ajustar polinomios probablemente resulte en un ajuste contaminado por el ruido presente en las mediciones. Para evitar esta situación, se recomienda utilizar puntos adicionales para ajustar el polinomio. El esfuerzo computacional adicional para usar más puntos es pequeño, ya que típicamente los puntos medidos están igualmente espaciados en el tiempo.

2.5. Subestructura Numérica

Los modelos de componentes estructurales inelásticos se pueden diferenciar por la forma en que la plasticidad se distribuye en el dominio del elemento, tal como se aprecia en la Figura 2.27. Para este trabajo se seleccionan los modelos en base a secciones fibra. La rigidez de estos elementos se calcula mediante integración numérica utilizando la rigidez seccional de un número reducido de secciones, siendo estas los puntos de integración a lo largo del elemento. Se proporcionan secciones tipo fibra de hormigón armado, acero o elásticas. La sección elegida en esta ocasión es una sección fibra de hormigón armado. En esta se utilizan fibras tipo quad y capas tipo straight. El comando quad se utiliza para construir un objeto de elemento FourNodeQuad que utiliza una formulación isoparamétrica bilineal. El comando straight se utiliza para construir una capa recta de armaduras. Adicionalmente, el tipo de elemento al que se asignará la sección generada anteriormente es el denominado dispBeamColumn. Este elemento se basa en la formulación de desplazamiento, asumiendo un campo de deformaciones a lo largo del elemento asociado a un elemento lineal elástico Finalmente, se puede configurar la forma en que se aplica la amortiguación al modelo, esta puede ser mediante amortiguación modal o tipo Rayleigh.

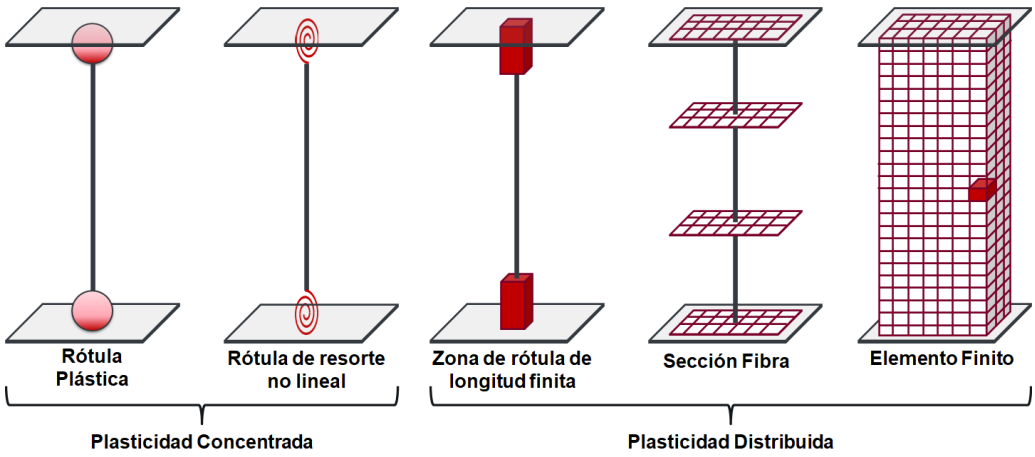


Figura 2.27: Clasificación de modelos para simular la respuesta inelástica de elementos estructurales. (Deierlein et al., 2010)

Adicionalmente, previo al análisis transiente, se realiza un análisis estático mediante el cual se aplican cargas gravitacionales, como los pesos de los pisos.

Los algoritmos de integración de tiempo más populares para RTHS son el método de diferencia central (CDM por su nombre en inglés central difference method) (Nakashima et al., 1992; Horiuchi et al., 1999) y el método explícito de Newmark (Newmark, 1959; Igarashi et al., 2000). Chang (2002) propuso un algoritmo explícito que tiene un excelente rendimiento de estabilidad y no presenta disipación numérica. Combescure y Pegon (1997) propone el llamado método AlphaOS, esto proviene del uso del $\alpha - method$ el cual utiliza el método de división del operador. Ricles y Chen (2008) utilizaron el método de mapeo de polos para desarrollar un algoritmo CR incondicionalmente estable en el que el desplazamiento y la velocidad en cada paso de integración se pueden calcular explícitamente. Gui et al. (2014) desarrollaron una familia de algoritmos explícitos con un parámetro controlado λ, en el que se encuentra que ciertas subfamilias son incondicionalmente estables para cualquier estado del sistema. Kolay y Ricles (2014) propusieron el método explícito KR-α con disipación de energía numérica controlable sobre la base del método generalized-α (Chung y Hulbert, 1993).

Los esquemas de integración de tiempo numérico utilizados en este trabajo son los llamados método AlphaOS, Newmark Explícito y Newmark. A continuación, una breve explicación matemática del método AlphaOS. El supuesto básico del método es que el comportamiento dinámico de la estructura se puede representar mediante un modelo de parámetros discretos que tiene solo un número finito de grados de libertad. Las ecuaciones de movimiento para este modelo se pueden expresar en términos de un sistema de ecuaciones diferenciales ordinarias de segundo orden igual al discutido en la sección 2.1.3.1, ecuación 2.1.

La solución para los nuevos pasos se obtiene en base a los pasos anteriores. A continuación, la duración T para la que se evaluará la respuesta estructural se divide en intervalos de tiempo iguales Δt. Conociendo el vector de desplazamiento u_n, el vector de velocidad $\dot{u}_n$ y el vector de aceleración $\ddot{u}_n$ en el momento t_n, el valor de estos vectores se obtiene en $\alpha - method$ en el momento $t_{n+1} = t_n + \Delta t$ según:

- Paso Predictor:

$$\tilde{u}_{n+1} = u_n + \Delta t \dot{u}_n + \frac{\Delta t^2}{2}(1 - 2\beta)\ddot{u}_n \tag{2.48}$$
$$\tilde{\dot{u}}_{n+1} = \dot{u}_n + \Delta t(1 - \gamma)\ddot{u}_n$$

- Paso Corrector:

$$u_{n+1} = \tilde{u}_{n+1} + \Delta t^2 \beta \ddot{u}_{n+1} \tag{2.49}$$
$$\dot{u}_{n+1} = \tilde{\dot{u}}_{n+1} + \Delta t \gamma \ddot{u}_{n+1}$$

- Verificación del sistema de ecuaciones de equilibrio discreto en tiempo "$\alpha - shifted''$

$$M\ddot{u}_{n+1} + (1+\alpha)C\dot{u}_{n+1} - \alpha C\dot{u}_n + (1+\alpha)r_{n+1} - \alpha r_n = (1+\alpha)P_{n+1} - \alpha P_n \tag{2.50}$$

los parámetros β y γ se eligen de la siguiente manera

$$\begin{aligned} \beta &= \frac{(1-\alpha)^2}{4} \\ \gamma &= \frac{(1-2\alpha)}{2} \end{aligned} \tag{2.51}$$

donde $\alpha \in [-1/3, 0]$ permite el ajuste de la amortiguación numérica del $\alpha - method$.

El método es implícito ya que u_{n+1} depende de $\ddot{u}_{n+1}$ relacionado con r_{n+1} que es una función de u_{n+1}. Luego, la verificación de Ec. 2.50, cuando la relación entre r y u se vuelve no lineal, implica un procedimiento iterativo. Cuando $\alpha = 0$, el esquema numérico degenera en el conocido esquema de integración temporal implícita (regla trapezoidal) $\beta - Newmark$, que todavía requiere, en principio, un procedimiento iterativo.

Es posible implementar este método sin iterar utilizando un método de división de operadores. Este método mantiene la estabilidad del esquema, quedando implícito para la parte elástica de la respuesta, pero no requiere ninguna iteración, siendo explícito para la parte no lineal de la respuesta. El método de división del operador (OS por su nombre en inglés operator splitting) se basa en la siguiente aproximación de la fuerza de restauración r_{n+1}.

$$r_{n+1}(u_{n+1}) \approx K^{|}u_{n+1} + (\tilde{r}_{n+1}(\tilde{u}_{n+1}) - K^{|}\tilde{u}_{n+1}) \tag{2.52}$$

Donde $K^{|}$ es una matriz de rigidez, idealmente con un valor cercano a la matriz elástica, pero nunca mayor o igual a la rigidez tangente actual de la estructura. Esta condición proporciona y asegura que el método sea incondicionalmente estable.

Usando la aproximación dada de la fuerza restauradora, es posible reemplazar las ecuaciones 2.48 y 2.49, para obtener $\ddot{u}_{n+1}$ como respuesta del sistema lineal.

$$\hat{M}\ddot{u}_{n+1} = \hat{P}_{n+1} \tag{2.53}$$

donde,

$$\hat{M} = M + \gamma\Delta t(1+\alpha)C + \beta\Delta t^2(1+\alpha)K^{|} \tag{2.54}$$

$$\hat{P}_{n+1} = (1+\alpha)P_{n+1} - \alpha P_n + \alpha\tilde{r}_n - (1-\alpha)\tilde{r}_{n+1} + \alpha V\tilde{u}_n - (1+\alpha)C\tilde{u}_{n+1} + \alpha(\gamma\Delta tC + \beta\Delta t^2K^{|})\ddot{u}_n \tag{2.55}$$

Tal como se menciona previamente, también se hará uso del método de Newmark. A continuación se muestran las denominadas ecuaciones de Newmark que permiten resolver el sistema dinámico mostrado en la ecuación 2.1.

$$u_{n+1} = u_n + \Delta t\dot{u}_n + \frac{(\Delta t)^2}{2}\left[(1-2\beta)\ddot{u}_n + 2\beta\ddot{u}_{n+1}\right] \tag{2.56}$$

$$\dot{u}_{n+1} = u_n + \Delta t[(1-\gamma)\ddot{u}_n + \gamma\ddot{u}_{n+1}] \tag{2.57}$$

γ y β definen la variación de la aceleración en el intervalo de tiempo.

- $\gamma = \frac{1}{2}$ y $\beta = \frac{1}{4}$ - Aceleración promedio constante.

- $\gamma = \frac{1}{2}$ y $\beta = \frac{1}{6}$ - Aceleración lineal.

El sistema de ecuaciones previa representa un sistema de ecuaciones no lineales el cual puede ser resuelto mediante métodos iterativos tales como el método de Newton-Raphson.

El método implícito de Newmark no se recomienda para simulación híbrida en tiempo real ya que las iteraciones pueden no converger y podrían dar como resultado cargas y descargas artificiales. De igual forma podrían provocar incrementos de desplazamiento no uniformes (oscilaciones de velocidad y aceleración en cada intervalo de tiempo). Finalmente, al ser iterativo su aplicación en tiempo real se vuelve difícil debido al tiempo de convergencia que puede tomar la resolución. Por tanto, para utilizar el método Newmark se eligen dos configuraciones, estas son trabajar con un número fijo de iteraciones (10) y la metodología explícita. Para la resolución del método explícito de Newmark, se siguen los siguientes cuatro pasos:

1. Calcular los desplazamientos

$$u_{n+1} = u_n + \Delta t \dot{u}_n + \frac{(\Delta t)^2}{2} \ddot{u}_n$$

2. Imponer los desplazamientos u_{n+1} y calcular/medir la fuerza r_{n+1}
3. Calcular las aceleraciones

$$[M + \Delta t \gamma C] \ddot{u}_{n+1} = P_{n+1} - r_{n+1} - C[\dot{u}_n + \Delta t(1-\gamma)\ddot{u}_n]$$
$$M_{eff} \ddot{u}_{n+1} = P_{eff}$$

4. Calcular las velocidades

$$\dot{u}_{n+1} = u_n + \Delta t[(1-\gamma)\ddot{u}_n + \gamma \ddot{u}_{n+1}]$$

Se vuelve explícito al definir $\beta = 0$.

2.6. Problema de Elasticidad 2D

2.6.1. Formulación Fuerte

Desde la cinemática, equilibrio diferencial y relación constitutiva, considere un dominio Ω con bordes Γ, particionado en Γ_u y Γ_t, donde en el primero existen desplazamientos prescritos y en el segundo existen tracciones prescritas. Se tiene que $\Gamma = \Gamma_t \bigcup \Gamma_u$, $\Gamma_t \bigcap \Gamma_u = \emptyset$. La formulación fuerte de elasticidad corresponde a:

$$\left.\begin{array}{rll} \text{Encuentre } \boldsymbol{u} \in C^2(\Omega, \mathbb{R}^2) \text{ tal que} & & \\ \text{div } \boldsymbol{\sigma} + \mathbf{b} = 0, & x \in \Omega & \\ \boldsymbol{\sigma} = \mathbb{C} : \boldsymbol{\varepsilon}, & x \in \Omega & \\ \boldsymbol{\varepsilon} = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T), & x \in \Omega & \\ \boldsymbol{u} = \bar{\boldsymbol{u}}, & x \in \Gamma_u & \\ \boldsymbol{\sigma}\mathbf{n} = \bar{\mathbf{t}}, & x \in \Gamma_t & \end{array}\right\} (S)$$

donde $\mathbb{C}$ es un tensor de elasticidad de cuarto orden, $\mathbf{b}$ contiene las fuerzas por unidad de volumen, $\bar{\boldsymbol{u}}$ son los desplazamientos prescritos en la frontera de Dirichlet Γ_u, $\bar{\mathbf{t}}$ son las tracciones prescritas en la frontera de Neumann Γ_t, $\boldsymbol{\sigma}$ es el tensor de esfuerzos, $\boldsymbol{\varepsilon}$ es el tensor de deformaciones y $\mathbf{n}$ es el vector normal.

2.6.2. Formulación Débil

Se definen los siguientes espacios:

$$\mathcal{S} = \{\boldsymbol{u} \ : \ u_i \in H^1(\Omega, R), \text{ tal que } u_i|_{\Gamma_{u_i}} = \bar{u}_i \ i = 1, ..., n_{sd}\}$$
$$\mathcal{V} = \{\boldsymbol{\delta u} \ : \ \delta u_i \in H^1(\Omega, R), \text{ tal que } \delta u_i|_{\Gamma_{u_i}} = 0 \ i = 1, ..., n_{sd}\}$$

Entonces, la formulación débil del problema de elasticidad es:

$$\left.\begin{array}{c}\text{Encuentre } \boldsymbol{u} \in \mathcal{S} \text{ tal que} \\ \displaystyle\int_\Omega \boldsymbol{\varepsilon}(\boldsymbol{\delta u}) : \boldsymbol{\sigma} dx = \int_\Omega \boldsymbol{\delta u} \cdot \boldsymbol{b} dx + \sum_{\alpha=1}^{n_{sd}} \int_{\Gamma_{t_\alpha}} \delta u_\alpha \bar{t}_\alpha dS, \quad \forall \boldsymbol{\delta u} \in \mathcal{V}\end{array}\right\} (W)$$

donde n_{sd} corresponde a la dimensión espacial del problema. Luego, se puede escribir en su formulación variacional como:

$$\left.\begin{array}{c}\text{Encuentre } \boldsymbol{u} \in \mathcal{S} \text{ tal que} \\ a(\boldsymbol{\delta u}, \boldsymbol{u}) = F(\boldsymbol{\delta u}), \quad \forall \boldsymbol{\delta u} \in \mathcal{V}\end{array}\right\} (W)$$

donde:

$$a(\boldsymbol{\delta u}, \boldsymbol{u}) = \int_\Omega \boldsymbol{\varepsilon}(\boldsymbol{\delta u})^T : \mathbb{C} : \boldsymbol{\varepsilon}(\boldsymbol{u}) dx$$
$$F(\boldsymbol{\delta u}) = \int_\Omega \boldsymbol{\delta u} \cdot \boldsymbol{b} dx + \sum_{\alpha=1}^{n_{sd}} \int_{\Gamma_{t_\alpha}} \delta u_\alpha \bar{t}_\alpha dS$$

2.6.3. Formulación de Elementos Finitos

Sea e_i el vector de base canónica en dirección i. Se define:

$$\mathcal{V}^h = \boldsymbol{\delta u}^h = \delta u_i^h \boldsymbol{e}_i \ : \ \delta u_i^h(\boldsymbol{x}) = \sum_{A \in \eta_{u_i}} N_A(\boldsymbol{x}) \delta u_{iA}$$
$$\mathcal{S}^h = \{\boldsymbol{u}^h = \boldsymbol{w}^h + g^h \ : \ \boldsymbol{w}^h \in \mathcal{V}^h \ y \ \boldsymbol{g^h} = g_i^h \mathbf{e}_i, \text{ con } g_i^h(\mathbf{x}) = \sum_{A \in \eta_{g_i}} N_A(\boldsymbol{x}) g_{iA}\}$$

La formulación FEM para el problema de elasticidad (W) es equivalente a

$$\left.\begin{array}{c}\text{Dada } \mathbf{K} \in \mathbb{R}^{MxM} \text{ y } \mathbf{F} \in \mathbb{R}^M, \text{ encuentre } \mathbf{u} \in \mathbb{R}^M \text{ tal que} \\ \mathbf{Ku} = \mathbf{F}\end{array}\right\} (L^{FEM})$$

Donde **u** es el vector de desplazamientos nodales,

$$\begin{aligned} K_{PQ} &= a(N_A \boldsymbol{e}_i, N_B \boldsymbol{e}_j) \\ F_P &= F(N_A \boldsymbol{e}_i) - a(N_A \boldsymbol{e}_i, \boldsymbol{g^h}) \end{aligned}$$

y P=ID(i,A), Q=ID(j,B) (con ID la matriz de destinación).

En este trabajo de utiliza una discretización mediante elementos Tri6, donde las funciones de forma en el espacio isoparamétrico están dadas por:

$$\hat{N}_1^e(\xi) = \xi_1(2\xi_1 - 1)$$
$$\hat{N}_2^e(\xi) = \xi_2(2\xi_2 - 1)$$
$$\hat{N}_3^e(\xi) = \xi_3(2\xi_3 - 1)$$
$$\hat{N}_4^e(\xi) = 4\xi_1\xi_2$$
$$\hat{N}_5^e(\xi) = 4\xi_2\xi_3$$
$$\hat{N}_6^e(\xi) = 4\xi_3\xi_1$$

Para la resolución FEM se emplea el *Partial Differential Equation Toolbox* de Matlab.

2.7. Formulación de Análisis Isogeométrico

La formulación por análisis isogeométrico consiste en utilizar los llamados vectores de nudos (*Knot Vectors*) para definir la región del problema. Además, plantea otro espacio de funciones base que pueden ser utilizadas para resolver el problema de Galerkin.

2.7.1. Espacios relevantes

- Espacio indicial: Este espacio consiste en darle un valor de coordenada a cada nudo en cada vector de nudo, sin importar si es que el nudo se repite dentro del vector. Esto genera un espacio con elementos que podrían tener un área paramétrica igual a cero y otros que no. En este caso, se trabajará solo con elementos que posean un área paramétrica distinta de cero, por lo que este espacio no será necesario.

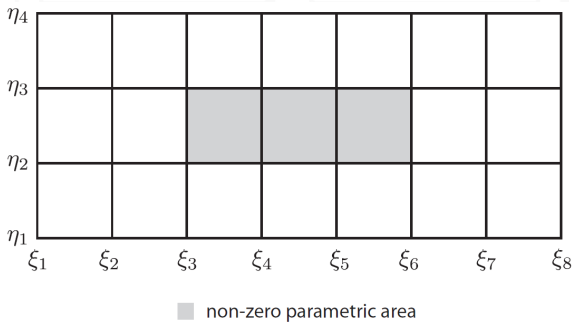


Figura 2.28: Espacio indicial generado por dos vectores de nudos iguales a $\Xi^1 = \{0, 0, 0, 1, 2, 3, 3, 3\}$ y $\Xi^2 = \{0, 0, 1, 1\}$, donde se destacan las regiones de área paramétrica distinta de cero (Nguyen et al., 2015).

- Espacio paramétrico: Consiste en los intervalos que no son cero entre los valores de los nudos del vector de nudos. Para el vector de nudos considerado anteriormente, en la Figura 2.29 se puede ver como se representa este espacio, el cual puede ser reducido a un cuadrado unitario mediante una normalización apropiada. Por ende, el espacio paramétrico se define como $\hat{\Omega} \subset \mathbb{R}^{d_p}$, donde d_p corresponde a las dimensiones de la región, con un set de coordenadas paramétricas $\xi = (\xi, \eta) = (\xi^1, \xi^2) \in \hat{\Omega}$ ($d_p = 2$). Si se normaliza, entonces $\hat{\Omega} = [0, 1]^{d_p}$.
  De la Figura 2.29 se puede ver que los elementos de área paramétrica no nula se encuentran rodeados por líneas de nudos con intervalos no nulos. Esto permite definir un vector de nudos únicos, el que es un subconjunto de Ξ, y se define como muestra la Ecuación 2.58, donde n_s corresponde al número de valores de nudos únicos.

$$\mathcal{S} = \{\xi_1, \xi_2, \cdots, \xi_{n_s}\} \quad \xi_i \neq \xi_{i+1} \text{ for } 1 \leq i \leq n_s - 1 \tag{2.58}$$

  Con esto definido, se puede generalizar el área de cada elemento como:

$$\hat{\Omega}^e = [\xi_i, \xi_{i+1}] \otimes [\eta_j, \eta_{j+1}], \quad 1 \leq i \leq n_s^1 - 1, \quad 1 \leq j \leq n_s^2 - 1 \tag{2.59}$$

  donde n_s^1 y n_s^2 indican el número de nudos únicos en los vectores de nudos de las direcciones ξ y η respectivamente, lo que lleva a un método de numeración de elementos tal que:

$$e = j(n_s^1 - 1) + i \tag{2.60}$$

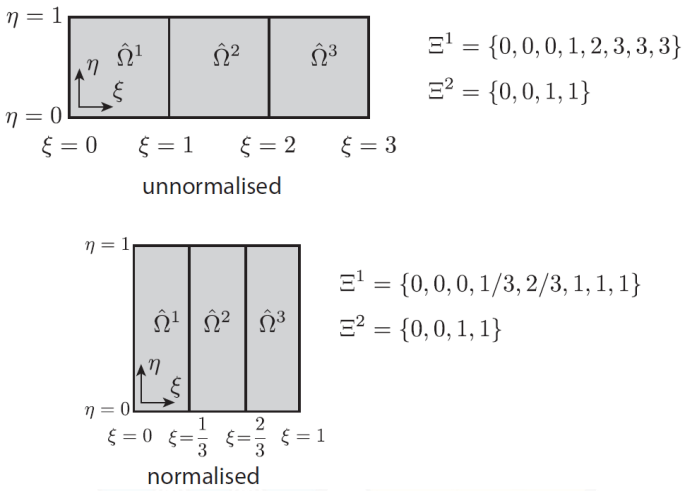


Figura 2.29: Espacio paramétrico definido por intervalos entre nudos no nulos (Nguyen et al., 2015).

- Espacio físico: Mediante la sumatoria de los puntos de control por las funciones base de los vectores de nudos asociados, es posible llevar las coordenadas paramétricas a un espacio físico. Para crear un espacio tridimensional, basta con que los puntos de control estén expresados en coordenadas (x, y, z). Para este trabajo, los puntos de control solo poseen coordenadas (x, y) ya que la figura que se está representando solo posee 2 dimensiones. En este espacio se puede evidenciar la posición de los puntos de control y apreciar que estos no coinciden con la posición de la curva NURBS a la que definen.
- Parent space: Los espacios anteriores son inherentes de las curvas *B-splines* y NURBS, pero para el análisis que se debe hacer se requiere de un espacio adicional, referido como *parent space* dado por $\tilde{\Omega} = [-1, 1]^{d_p}$. Las rutinas de integración numérica son las que necesitan de este espacio. En general se definen para el intervalo $[-1, 1]$.

2.7.2. Funciones de forma

Las funciones base de las NURBS en una dimensión están dadas por:

$$R_{i,p}(\xi) = \frac{N_{i,p}(\xi) w_i}{\sum_{\hat{i}=1}^{n} N_{\hat{i}}(\xi) w_{\hat{i}}} \tag{2.61}$$

donde $\{N_i\}_{i=1}^{n}$ es el set de funciones base *B-splines* de orden p definidas recursivamente por la fórmula de Cox-de-Boor, con $p = 0$:

$$N_{i,0}(\xi) = \begin{cases} 1 & \text{si} \quad \xi_i \leq \xi \leq \xi_{i+1} \\ 0 & \text{en otro caso} \end{cases} \tag{2.62}$$

para polinomios $1 \leq p$:

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_i + 1} N_{i+1,p-1}(\xi) \tag{2.63}$$

y $\{w_i\}_{i=1}^{n}$ son los pesos de las NURBS dadas generalmente por un programa CAD.

En dos dimensiones, las funciones base están dadas por:

$$R_{i,j}^{p,q}(\xi, \eta) = \frac{N_i(\xi) M_j(\eta) w_{i,j}}{\sum_{\hat{i}=1}^{n} \sum_{\hat{j}=1}^{m} N_{\hat{i}}(\xi) M_{\hat{j}}(\eta) w_{\hat{i},\hat{j}}} \tag{2.64}$$

2.7.3. Discretización isogeométrica

Se deben escribir las discretizaciones *B-splines* y NURBS en coordenadas isoparamétricas, quedando para cada elemento e:

$$\boldsymbol{x}^{\boldsymbol{e}}(\tilde{\boldsymbol{\xi}}) = \sum_{a=1}^{n_{en}} \boldsymbol{P}_{\boldsymbol{a}}{}^{\boldsymbol{e}} R_a^e(\tilde{\boldsymbol{\xi}}) \tag{2.65}$$

donde a es un índice de función de base local, $n_{en} = (p+1)^{d_p}$ es el número de funciones de base diferentes a cero sobre el elemento e, y $\boldsymbol{P}_{\boldsymbol{a}}^{\boldsymbol{e}}$, R_a^e son los puntos de control y funciones base de las NURBS asociadas al índice a, respectivamente. Se emplea la misma notación de elementos finitos para las matrices de conectividad, $A = \text{IEN}(a, e)$.

Los puntos de control globales y locales se relacionan mediante $\boldsymbol{P}_{\boldsymbol{A}} \equiv \boldsymbol{P}_{\text{IEN}(\boldsymbol{a},\boldsymbol{e})} \equiv \boldsymbol{P}_{\boldsymbol{a}}^{\boldsymbol{e}}$, y de igual manera para R_a^e. El campo de desplazamientos u se puede discretizar como:

$$\boldsymbol{u}^{\boldsymbol{e}}(\tilde{\boldsymbol{\xi}}) = \sum_{a=1}^{n_{en}} \boldsymbol{d}_{\boldsymbol{a}}^{\boldsymbol{e}} R_a^e(\tilde{\boldsymbol{\xi}}) \tag{2.66}$$

donde $\boldsymbol{d}_{\boldsymbol{a}}^{\boldsymbol{e}}$ representa una variable de control nodal.

2.7.4. Mapping (cambio de variables)

El uso de NURBS para la discretización de los elementos añade un espacio de trabajo que no se encuentra en la metodología FEM tradicional: el espacio paramétrico, como se puede apreciar en la Figura 2.30.

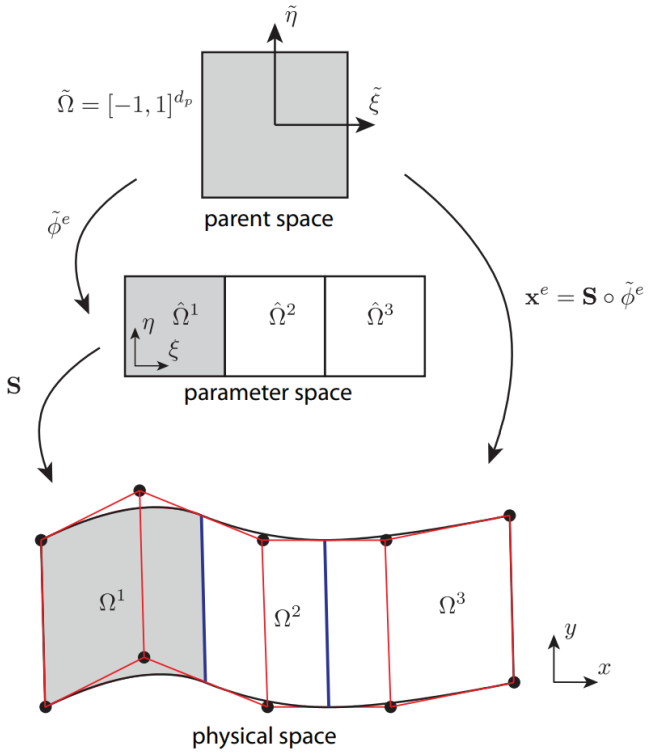


Figura 2.30: Interpretación de los espacios utilizados en IGA (Nguyen et al., 2015).

Por lo tanto, se deben realizar dos cambios de coordenadas: $\tilde{\phi}^e : \tilde{\Omega} \rightarrow \hat{\Omega}^e$ y $\mathbf{S} : \hat{\Omega} \rightarrow \Omega$. El paso desde $\mathbf{x}^e : \tilde{\Omega} \rightarrow \Omega^e$ está dado por la composición de $\mathbf{S} \circ \tilde{\phi}^e$.

En el caso de $d_p = d_s = 2$, un elemento definido por $\hat{\Omega}^e = [\xi_i, \xi_{i+1}] \otimes [\eta_i, \eta_{i+1}]$ es convertido mediante:

$$\tilde{\phi}^e(\tilde{\boldsymbol{\xi}}) = \left\{ \begin{array}{l} \frac{1}{2}[(\xi_{i+1} - \xi_i)\tilde{\xi} + (\xi_{i+1} + \xi_i)] \\ \frac{1}{2}[(\eta_{j+1} - \eta_j)\tilde{\eta} + (\eta_{j+1} + \eta_j)] \end{array} \right\} \tag{2.67}$$

$$|\boldsymbol{J}_{\tilde{\boldsymbol{\xi}}}| = \frac{1}{4}(\xi_{i+1} + \xi_i)(\eta_{j+1} + \eta_j) \tag{2.68}$$

En el caso de $d_p = d_s = 2$, la matriz de transformación Jacobiana está dada por:

$$J_{\xi} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial x}{\partial \eta} \\ \dfrac{\partial y}{\partial \xi} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} \tag{2.69}$$

donde las componentes son calculadas como:

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}} = \sum_{a=1}^{n_{en}} \boldsymbol{P_a}^e \frac{\partial R_a^e(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \tag{2.70}$$

La conversión $x^e : \tilde{\Omega} \rightarrow \Omega^e$ está dada por:

$$\boldsymbol{x}^e(\tilde{\boldsymbol{\xi}}) = \sum_{a=1}^{n_{en}} \boldsymbol{P_a}^e R_a^e(\tilde{\boldsymbol{\xi}}) \tag{2.71}$$

$$|J| = |J_{\xi}||J_{\tilde{\xi}}| \tag{2.72}$$

$$\int_{\Omega} f(\boldsymbol{x}) d\Omega = \sum_{e=1}^{n_{el}} \int_{\tilde{\Omega}} f(\tilde{\boldsymbol{\xi}}) |J| d\tilde{\Omega} \tag{2.73}$$

donde la integral final es calculable con la cuadratura de Gauss.

2.7.5. Implementación para problemas de elasticidad 2D

Considere un dominio Ω con bordes Γ, particionado en Γ_u y Γ_t, donde en el primero existen desplazamientos prescritos y en el segundo existen tracciones prescritas. Se tiene que $\Gamma = \Gamma_t \bigcup \Gamma_u$, $\Gamma_t \bigcap \Gamma_u = \emptyset$. La formulación débil es:

$$\left. \begin{array}{c} \text{Encuentre } \boldsymbol{u} \in \mathcal{S} \text{ tal que} \\ \displaystyle\int_{\Omega} \boldsymbol{\varepsilon}(\boldsymbol{\delta u}) : \boldsymbol{\sigma} dx = \int_{\Gamma_t} \boldsymbol{\delta u} \cdot \bar{\mathbf{t}} d\Gamma + \int_{\Omega} \mathbf{b} \cdot \boldsymbol{\delta u} d\Omega, \quad \forall \boldsymbol{\delta u} \in \mathcal{V} \end{array} \right\} (W)$$

Utilizando el método de Galerkin, donde la misma función de forma $R_a(\tilde{\xi})$ es usada para u y δu:

$$\boldsymbol{u}(\boldsymbol{x}) = \sum_{A=1}^{n_{np}} R_A(\tilde{\boldsymbol{\xi}}) \boldsymbol{u_A}, \quad \boldsymbol{\delta u}(\boldsymbol{x}) = \sum_{A=1}^{n_{np}} R_A(\tilde{\boldsymbol{\xi}}) \boldsymbol{\delta u_A} \tag{2.74}$$

Luego, se puede plantear el set de ecuaciones discretas estándar $Ku = F$ con:

$$\mathbf{K}_{AB} = \int_{\Omega} \mathbf{B_A}^T \mathbf{D} \mathbf{B_B} d\Omega, \qquad \mathbf{f}_A = \int_{\Gamma_t} R_A \bar{\boldsymbol{t}} d\Omega + \int_{\Omega} R_A \mathbf{b} d\Omega, \qquad A, B = 1, 2, ..., n_{np} \tag{2.75}$$

En dos dimensiones, la matriz B_A está dada por:

$$\mathbf{B_A} = \begin{bmatrix} R_{A,x} & 0 \\ 0 & R_{A,y} \\ R_{A,y} & R_{A,x} \end{bmatrix} \tag{2.76}$$

Para la implementación del método en el software Matlab, se utilizan como referencia los códigos provistos por Nguyen et al. (2015); Gantner y Praetorius (2020) (IGAFEM) y Vuong et al. (2010) (ISOGAT), junto con algunas funciones auxiliares.

2.8. Discusión

Desafortunadamente, los frameworks UI-SimCor (Kwon et al., 2007; Spencer et al., 2007), Hybrid-FEM (Karavasilis et al., 2010), Mercury (Saouma et al., 2012, 2014), ISEE (Wang et al., 2007; Yang et al., 2007) y Celestina-Sim (Lamata Martinez et al., 2016) no presentan soporte actualmente, por lo que están desactualizados. Otros, como CSA (Zhang et al., 2014), se desarrollaron para el acoplamiento estructural de forma puramente numérica y no están pensado para implementación en laboratorio. En el trabajo presentado por Peroni et al. (2021), el controlador ELSAREC y DAQ pueden proporcionar capacidades en tiempo real en este momento, pero esto aún no se aplica a RTHS. El uso de FMI (Blochwitz et al., 2011; Blockwitz et al., 2012; Andreas Junghanns et al., 2021) ha sido extenso en la industria automotriz y de aviación, pero a nuestro leal saber y entender, además del trabajo desarrollado por Cláudio Gomes et al. (2021), aún no ha habido ninguna aplicación del FMI al campo de las pruebas sísmicas híbridas.

Los que se utilizan ampliamente en la actualidad son OpenFresco (Takahashi y Fenves, 2006; Schellenberg et al., 2009a,b) y UT-SIM (Mortazavi et al., 2017). Sin embargo, tienen ciertas limitaciones. El marco de simulación de la Universidad de Toronto (marco UT-SIM) integra varias herramientas de análisis estructural y muestras experimentales para simulaciones híbridas pseudodinámicas (PsD), es decir, no está destinado a la simulación híbrida en tiempo real. OpenFresco, por otro lado, está diseñado y pensado para su uso en conjunto con una *memoria RAM compartida* (SCRAMNet +) y con un xPc Target. Además, no permite la comunicación con otros sistemas de control (como el sistema de control utilizado en este documento) porque no tiene interfaces de comunicación. Por tanto, para quienes no disponen de este equipamiento específico, es difícil, si no imposible, realizar una simulación híbrida en tiempo real.

Un trabajo novedoso y prometedor es el propuesto por Xu et al. (2021). Este trabajo enfrenta y da solución a los mismos problemas aquí evidenciados para OpenFresco. Se considera el software Matlab como un intermediario de comunicación entre el software OpenSees y el sistema del actuador. Esto, debido al fácil desarrollo de interfaces de forma independiente con numerosos software y sistemas de actuador. Se considera el protocolo TCP/IP con esta finalidad. El sistema de comunicación Opensees-Matlab puede comunicarse con la mayoría de los sistemas de control. Sin embargo, a pesar de que esta metodología tiene capacidad en tiempo real, aún no ha sido probada.

Estos problemas motivan el desarrollo de una nueva metodología de acoplamiento entre la subestructura numérica y experimental que permita, y garantice, tiempo real, pero considerando una fácil implementación en el laboratorio y sin necesidad de una gran variedad de equipos, o marcas específicas.

La compensación seleccionada a utilizar es la denominada compensación adaptiva. Adicionalmente, los modelos numéricos a utilizar, tal como se menciona, se implementan en OpenSees/OpenSeesPy y son evaluados considerando diferentes integradores. El Problema de Elasticidad 2D y la Formulación de Análisis Isogeométrico serán utilizados en el Caso de Estudio 3.

3 | Coordinación de subestructuras en RTHS

3.1. Protocolos de Comunicación

3.1.1. Antecedentes

Los sockets tienen una larga historia en protocolos de comunicación entre computadoras. Su uso se originó con ARPANET en 1971 y luego se convirtió en una API en el sistema operativo Berkeley Software Distribution (BSD) lanzado en 1983 llamado Berkeley sockets.

Cuando Internet despegó en la década de 1990 con la World Wide Web, también lo hizo la programación de redes. Los navegadores y servidores web no eran las únicas aplicaciones que aprovechaban las redes recién conectadas y usaban sockets. Se generalizaron las aplicaciones cliente-servidor de todos los tipos y tamaños.

Hoy en día, aunque los protocolos subyacentes utilizados por la API de socket han evolucionado a lo largo de los años y hemos visto otros nuevos, la API de bajo nivel sigue siendo la misma.

El tipo más común de aplicaciones de socket son las aplicaciones cliente-servidor, donde un lado actúa como servidor y espera las conexiones de los clientes. Este es el tipo de aplicación que se utilizará. Más específicamente, la API de sockets para sockets de Internet, a veces llamados sockets Berkeley o BSD. También hay sockets de dominio Unix, que solo se pueden usar para comunicarse entre procesos en el mismo host.

3.1.2. Descripción general de la API de socket

Los sockets y la API de socket se utilizan para enviar mensajes a través de una red. Proporcionan una forma de comunicación entre procesos (Inter-Process Communication IPC). La red puede ser una red local lógica a la computadora o físicamente conectada a una red externa, con sus propias conexiones a otras redes. El módulo de socket de Python proporciona una interfaz para la API de sockets de Berkeley. Las funciones y métodos de la API de socket principal en este módulo son *socket(), bind(), listen(), accept(), connect(), connect_ex(), send(), recv(), and close()*.

Python proporciona una API conveniente y consistente que se asigna directamente a estas llamadas al sistema, sus contrapartes C. Como parte de su biblioteca estándar, Python también tiene clases que facilitan el uso de estas funciones de socket de bajo nivel.

3.1.3. TCP Sockets

Se debe crear un objeto socket usando socket.socket () y especificar el tipo de socket como socket.SOCK_STREAM. Cuando se hace, el protocolo predeterminado que se utiliza es el Protocolo de control de transmisión (TCP).

Algunas de las ventadas de utilizar TCP (Transmission Control Protocol) son:

- Es confiable: los paquetes caídos en la red son detectados y retransmitidos por el remitente.
- Tiene entrega de datos en orden: su aplicación lee los datos en el orden en que fueron escritos por el remitente.

Por el contrario, los sockets del User Datagram Protocol (UDP) no son confiables y los datos leídos por el receptor pueden estar desordenados con respecto a las escrituras del remitente. Esto es importante ya que las redes son un sistema de entrega de mejor esfuerzo. No hay garantía de que sus datos lleguen a su destino o de que recibirá lo que se le envió. Los dispositivos de red (por ejemplo, enrutadores y conmutadores) tienen un ancho de banda finito disponible y sus propias limitaciones inherentes al sistema. Tienen CPU, memoria, buses y búfer de paquetes de interfaz, al igual que nuestros clientes y servidores. TCP le evita tener que preocuparse por la pérdida de paquetes, los datos que llegan fuera de orden y muchas otras cosas que suceden invariablemente cuando se comunica a través de una red. En el diagrama expuesto en la Figura 3.1 vemos la secuencia de llamadas a la API de socket y el flujo de datos para TCP:

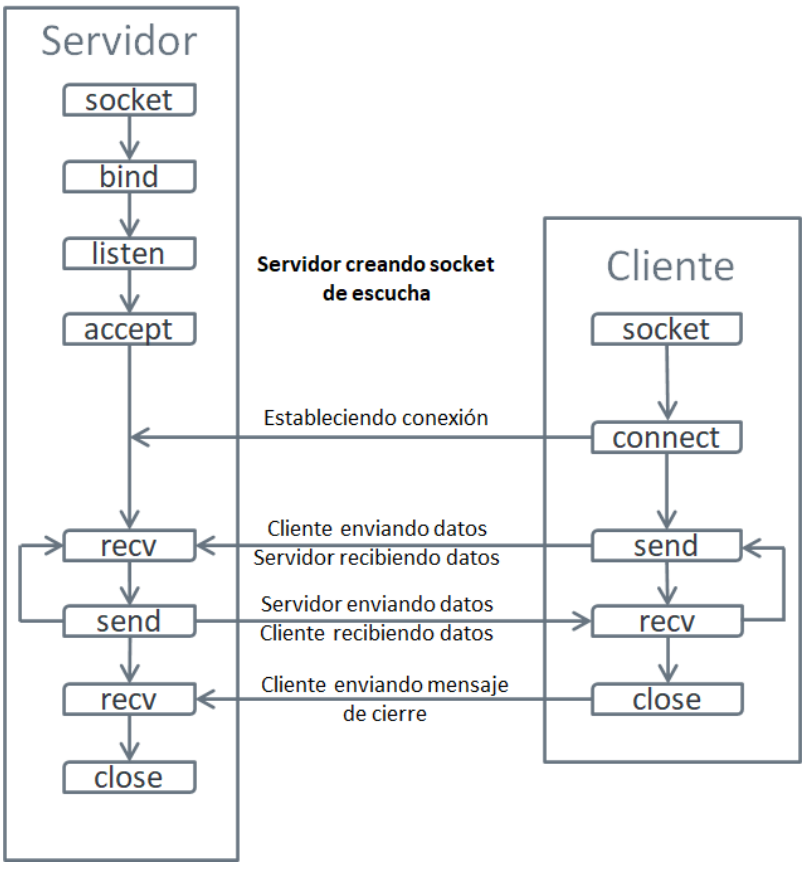


Figura 3.1: Protocolo Cliente-Servidor (TCP) API
(Fuente: University of Toronto Simulation (UT-SIM))

La columna de la izquierda de la Figura 3.1 representa al servidor. En el lado derecho está el cliente. Comenzando en la columna superior izquierda, tenga en cuenta las llamadas a la API que hace el servidor

para configurar un socket de “escucha":

- socket()
- bind()
- listen()
- accept()

Un socket de escucha hace exactamente lo que parece. Escucha las conexiones de los clientes. Cuando un cliente se conecta, el servidor llama a accept () para aceptar o completar la conexión.

El cliente llama a connect() para establecer una conexión con el servidor e iniciar el protocolo de enlace de tres vías. El paso del handshake es importante ya que asegura que cada lado de la conexión sea accesible en la red, en otras palabras, que el cliente pueda llegar al servidor y viceversa. Puede ser que solo un host, cliente o servidor, pueda llegar al otro. En el medio está la sección de ida y vuelta, donde los datos se intercambian entre el cliente y el servidor mediante llamadas a send () y recv (). En la parte inferior, el cliente y el servidor cierran () sus respectivos sockets.

3.2. Transición entre tasas

Existen en la literatura múltiples metodologías y técnicas para la transición entre tasas. En estas destacan el algoritmo predictor-corrector (Mosqueda et al., 2005; Schellenberg et al., 2009b), AMRI (Maghareh et al., 2016), mrRTHS (Waldbjoern et al., 2021) y los métodos clásicos de extrapolación polinomial (Bonnet et al., 2007; Wallace et al., 2005; Horiuchi y Konno, 2001b).

3.2.1. Predictor-Corrector

El algoritmo predictor-corrector (Mosqueda et al., 2005; Schellenberg et al., 2009b), se programó en Simulink y Stateflow dentro del entorno MATLAB como se muestra en la Figura 3.2 (Matlab, 2020). Los casos estudio 1, 2 y 3 utilizan esta transición entre tasas para su ejecución.

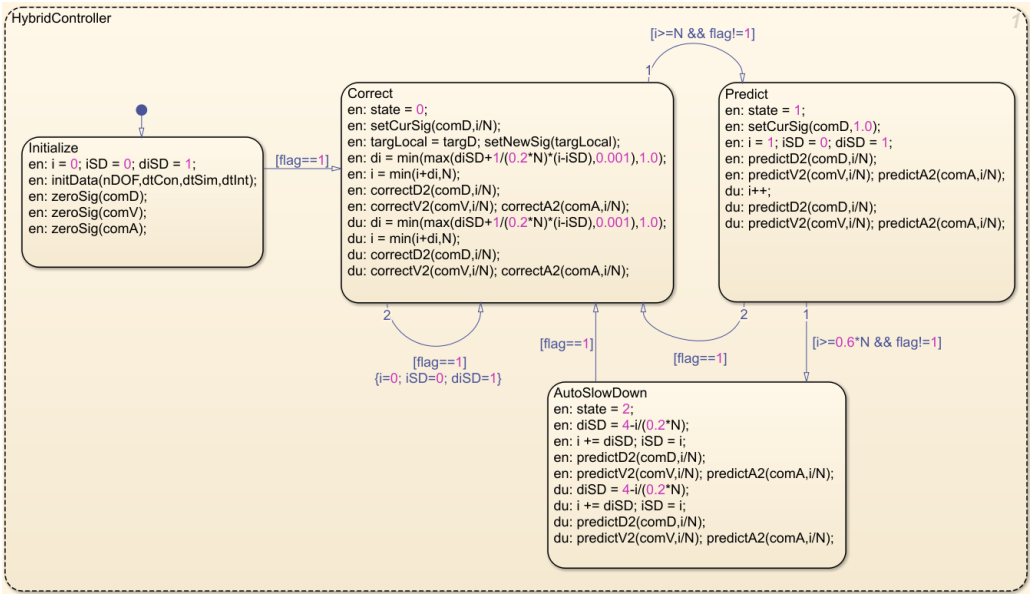


Figura 3.2: Algoritmo predictor-corrector en Stateflow.

Se ajusta un polinomio a los desplazamientos objetivo recibidos del modelo híbrido para generar una señal de comando continua a la velocidad deseada. El orden del polinomio de Lagrange ajustado se puede cambiar simplemente en el diagrama del codificador de flujo de estado antes de ejecutar una prueba híbrida. Para estas simulaciones, se seleccionaron polinomios de segundo orden para desplazamientos, velocidades y aceleraciones.

Para sincronizar la ejecución no determinista del análisis OpenSees/OpenFresco con la ejecución determinista del sistema de control, el algoritmo predictor-corrector realiza las siguientes tareas:

1) Mientras el software de análisis resuelve las ecuaciones de movimiento para el nuevo desplazamiento objetivo, el algoritmo genera desplazamientos de comandados basados en la predicción directa de polinomios.
2) Una vez que se ha recibido el nuevo desplazamiento objetivo, el algoritmo cambia al modo de corrección donde genera desplazamientos comandados que impulsan la respuesta hacia el nuevo desplazamiento objetivo.
3) Si el nuevo desplazamiento objetivo no se recibe dentro del 60 % del tamaño del paso de tiempo de simulación, el algoritmo ralentiza gradualmente los desplazamientos comandados hasta que se recibe el nuevo desplazamiento objetivo. Por lo tanto, para lograr una ejecución en tiempo real de la prueba híbrida sin ralentizaciones, el software de análisis OpenSees necesita calcular un nuevo desplazamiento objetivo en menos de $0,6 \times \Delta t$.

3.2.2. Transición entre tasas basada en datos (filtro de Wiener)

Esta sección describe los métodos propuestos para realizar la transición de múltiples velocidades para la simulación híbrida en tiempo real. La idea principal es realizar una estimación probabilística simple del siguiente punto con la frecuencia de muestreo menor utilizando propiedades locales y globales, y luego realizar una interpolación incluyendo este punto para obtener los puntos para la frecuencia de muestreo mayor. Esto se denomina filtro de transición de múltiples velocidades basado en datos (DDMRT por su sigla en inglés, Data-driven multi-rate transitioning). La Figura 3.3 muestra el esquema general de lo que sería una simulación híbrida en tiempo real virtual. Aquí claramente apreciamos los bloques que trabajan a una tasa rápida y a una tasa lenta. Adicionalmente, tenemos el bloque de retención de orden cero (ZOH, por su nombre en inglés Zero-Order Hold). Este bloque retiene su entrada para el período de muestra que especifique, es decir, entrega una salida a una tasa de tiempo específicada, en este caso, a una baja frecuencia de muestreo. Esta metodología fue propuesta por los Prof. Fernando Gómez y Gastón Fermandois (Mera et al., 2022), e implementada y validada por el autor. Los casos estudio 4 y 5 utilizan esta transición entre tasas para su ejecución.

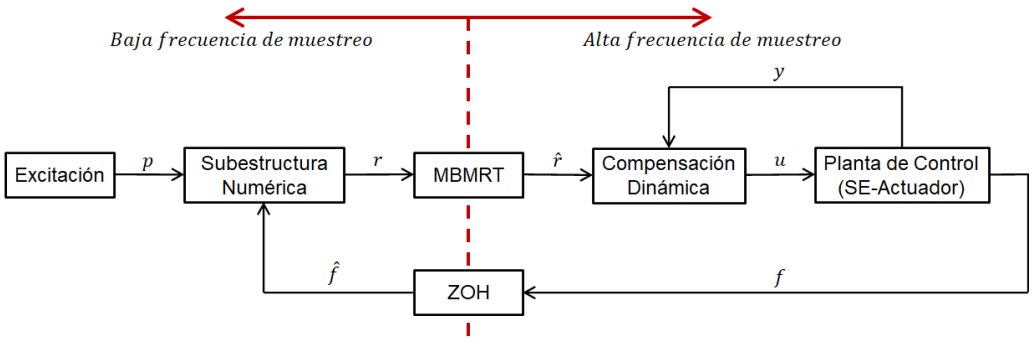


Figura 3.3: Diagrama de bloques para la transición de múltiples velocidades basada en datos para simulación híbrida en tiempo real (DDMRT-RTHS)

Las siguientes subsecciones describen las estrategias para estimar el siguiente punto de datos de la

señal utilizando filtros discretos.

3.2.2.1. Filtro invariante de tiempo lineal

Sea $u(t)$ la señal escalar de interés en tiempo continuo. La señal se discretiza con la relación de muestreo f_1 dando la secuencia de valores $\{u_0, u_1, u_2, u_3, \ldots\}$, y el objetivo es estimar el siguiente punto de la secuencia dados los puntos anteriores de la secuencia. Debido a la facilidad de implementación y la necesidad de estimaciones en línea, se propone un filtro lineal invariante en el tiempo (LTI) de respuesta al impulso finito causal $\mathbf{w}$ para estimar el siguiente punto de la siguiente manera:

$$\hat{u}_n = \sum_{i=1}^{L} w_i u_{n-i} = w_1 u_{n-1} + w_2 u_{n-2} + \cdots + w_L u_{n-L} \tag{3.1}$$

donde $\hat{u}_n$ es la estimación de u_n, $\mathbf{w} = [w_1, w_2, \ldots, w_L]^{\mathrm{T}}$ es el filtro y L es la longitud del filtro. Los coeficientes del filtro se calculan al principio y la estimación del siguiente punto se realiza en tiempo lineal. Además, la longitud del filtro suele ser pequeña y, en consecuencia, la implementación del filtro requiere muy pocos recursos, lo cual es una limitación importante en las aplicaciones en tiempo real.

Suponiendo que la señal $u(t)$ es un proceso estocástico, entonces el filtro se puede calcular de manera que se minimice el error cuadrático medio e con una regularización de Tikhonov:

$$\begin{aligned} e(\mathbf{w}) &= \mathbb{E}((u_n - \hat{u}_n)^2) + \lambda \|\mathbf{w}\|^2 \\ &= \mathbb{E}(u_n^2) - 2\sum_{i=1}^{L} w_i \mathbb{E}(u_n u_{n-i}) + \sum_{i=1}^{L}\sum_{j=1}^{L} w_i w_j \mathbb{E}(u_{n-i} u_{n-j}) + \lambda \left(\sum_{j=1}^{L} w_i^2 \right) \end{aligned} \tag{3.2}$$

donde $\lambda > 0$ es el factor de regularización.

La forma cuadrática anterior tiene un mínimo local, que se calcula haciendo que las derivadas parciales con respecto a los coeficientes de filtro sean iguales a 0. Para simplificar, $\mathbf{I}$ es la matriz de identidad $L \times L$, y la siguiente $L \times L$ matriz $\mathbf{R}$ y $L \times 1$ vector $\mathbf{r}$ se definen de la siguiente manera:

$$R_{ij} = \mathbb{E}(u_{n-i} u_{n-j}), \quad r_i = \mathbb{E}(u_n u_{n-i}), \quad \text{for} \quad i, j = 1, 2, \ldots, L \tag{3.3}$$

Entonces, el filtro óptimo es el sentido del error cuadrático medio está dado por:

$$(\mathbf{R} + \lambda \mathbf{I})\mathbf{w} = \mathbf{r} \quad \Longrightarrow \quad \mathbf{w} = (\mathbf{R} + \lambda \mathbf{I})^{-1}\mathbf{r} \tag{3.4}$$

Cuando u es un proceso estocástico débilmente estacionario con función de correlación conocida, la matriz $\mathbf{R}$ es semidefinida positiva y en algunos casos puede ser singular. Por lo tanto, $\mathbf{R} + \lambda \mathbf{I}$ es invertible y positivo definido para cualquier λ positivo, entonces, la solución anterior proporciona el mínimo global del problema. Este filtro se conoce como filtro de Wiener regularizado causal. El factor de regularización λ debe ser un pequeño número real positivo, los valores adecuados se muestran en los ejemplos de la siguiente sección.

Para un proceso estocástico, los operadores de expectativa en la definición de las matrices requieren la función de distribución de probabilidad para cualquier colección de puntos en el proceso. Para un proceso estacionario de sentido amplio, solo se necesita la función de correlación del proceso. Sin embargo, la señal no es un proceso estocástico y las propiedades anteriores no están disponibles. Alternativamente, los valores esperados pueden evaluarse aproximadamente en un sentido estadístico si está disponible una realización

de la señal. Claramente, la realización de la señal no está disponible, en cambio, se usa una realización aproximada del proceso $\tilde{u}_i$ para estimar la expectativa de la siguiente manera

$$\tilde{\mathbb{E}}(u_i u_j) = \frac{1}{M}\sum_{k=1}^{M}(\tilde{u}_{i+k}\tilde{u}_{j+k}) \tag{3.5}$$

El filtro propuesto incluye características globales y locales, lo que mejora la precisión de la estimación; las características globales están dadas por las propiedades estadísticas utilizadas en el cálculo de los coeficientes, y las características locales están dadas por los últimos L puntos de la señal. Bajo el supuesto de una estacionalidad débil del proceso, los coeficientes del filtro son invariantes en el tiempo. Esta suposición es justa para sistemas lineales sujetos a excitaciones de media cero o sistemas no lineales con respuesta de media cero. La idoneidad de esta suposición se mostrará en los ejemplos de la siguiente sección.

Como se indicó anteriormente, se necesita una realización aproximada para calcular los coeficientes de filtro. En el marco de simulación híbrida, la subestructura numérica y las excitaciones del sistema están fácilmente disponibles y, en consecuencia, no se necesitan aproximaciones en estos componentes. Por otro lado, se desconoce el modelo de la subestructura física y el objetivo principal de la prueba es evaluar el desempeño de este componente. No obstante, se puede utilizar una aproximación del componente físico basada en geometría conocida, propiedades nominales del material y un modelo constitutivo razonable. Entonces, la realización aproximada $\tilde{u}_i$ se puede obtener numéricamente y fuera de línea usando una aproximación de la subestructura física.

La suposición en el enfoque propuesto es que la compensación en la respuesta debido al modelo físico incorrecto no afecta en gran medida la precisión del filtro dado que se basa en propiedades estadísticas globales; la idoneidad de esta suposición se mostrará en los ejemplos. Además, debido a que el análisis se realiza off line, la duración de la realización puede ser tan larga como sea necesario y el intervalo de tiempo del análisis tan pequeño como sea necesario para obtener la precisión adecuada. Finalmente, para RTHS multi axial (*maRTHS*, por su nombre en inglés Multiaxial Real-Time Hybrid Simulation), se requiere la estimación de una señal vectorial, luego, el método propuesto se aplica por separado a cada entrada; Debido a la naturaleza del filtro propuesto, se requieren pocos recursos computacionales para implementar el filtro en tiempo real.

3.2.2.2. Filtro variable en tiempo lineal

Para sistemas lineales, el método anterior proporciona estimaciones precisas como se muestra en los ejemplos de la siguiente sección. El enfoque anterior también produce estimaciones precisas para sistemas con no linealidades leves. Sin embargo, para sistemas con grandes no linealidades, especialmente para sistemas con deformaciones residuales, el enfoque anterior puede perder precisión. Por lo tanto, en esta subsección, se presenta una alternativa utilizando un filtro de variación en el tiempo lineal (LTV), que se adapta en cada paso de tiempo.

Como en la sección anterior, el objetivo es estimar el siguiente punto de la secuencia dados los puntos anteriores de la secuencia. Debido a la facilidad de implementación y la necesidad de estimaciones en línea, se propone un filtro variable en el tiempo de respuesta al impulso finito causal $\mathbf{w}_n$ para estimar el siguiente punto de la siguiente manera:

$$\hat{u}_n = \sum_{i=1}^{L} w_{n,i}u_{n-i} = w_{n,1}u_{n-1} + w_{n,2}u_{n-2} + \cdots + w_{n,L}u_{n-L} \tag{3.6}$$

donde $\hat{u}_n$ es la estimación de u_n, $\mathbf{w}_n = [w_{n,1}, w_{n,2}, \ldots, w_{n,L}]^{\mathrm{T}}$ es la estimación actual del filtro y L es la longitud del filtro. La diferencia con el filtro LTI es que ahora los coeficientes del filtro se actualizan en cada paso, y luego la estimación del siguiente punto de la señal se realiza como antes. Además, la longitud del filtro suele ser pequeña y, en consecuencia, la implementación del filtro requiere pocos recursos, lo que constituye una

limitación importante en las aplicaciones en tiempo real. Vale la pena notar que el filtro LTV requiere más recursos que el filtro LTI debido al paso de actualización.

Proponiendo el mismo objetivo que en el filtro LTI de minimizar el error cuadrático medio, entonces la función objetivo es nuevamente la forma cuadrática mostrada anteriormente, pero los coeficientes del filtro ahora se actualizan en cada paso. Luego, los coeficientes se pueden actualizar en cada paso usando la iteración de descenso de gradiente, donde se usan gradientes de las funciones objetivo. Aquí, el algoritmo de mínimos cuadrados medios con fugas (LMS) se utiliza para actualizar el coeficiente de filtro de la siguiente manera:

$$\begin{aligned} e_n &= u_n - \hat{u}_n \\ \mathbf{w}_{n+1} &= (1 - \mu\lambda)\mathbf{w}_n + \mu e_n \mathbf{U}_n \end{aligned} \tag{3.7}$$

donde $\mathbf{U}_n = [u_{n-1}, u_{n-2}, \ldots, u_{nL}]^{\mathrm{T}}$ y $\mu > 0$ es la tasa de aprendizaje.

El paso de actualización requiere solo sumas vectoriales y multiplicaciones escalares, que se pueden realizar en tiempo lineal. En consecuencia, el paso de actualización, que se realiza en línea, no requiere muchos recursos computacionales.

La tasa de aprendizaje controla la convergencia del método. Una tasa de aprendizaje pequeña implica que es más probable que el filtro converja, pero convergerá más lentamente. Para garantizar la convergencia del algoritmo, la tasa de aprendizaje debe satisfacer la siguiente condición:

$$0 < \mu < \frac{2}{\lambda + \psi(\mathbf{R})} \tag{3.8}$$

donde $\psi(\mathbf{R})$ denota el valor propio más grande de la matriz $\mathbf{R}$.

Si la tasa de aprendizaje satisface la condición anterior, el filtro eventualmente convergerá; sin embargo, el uso de un punto inicial arbitrario para el filtro puede requerir demasiadas iteraciones, tal vez incluso más que el número de pasos de tiempo disponibles. Por lo tanto, se necesita un punto inicial adecuado para el filtro para acelerar la convergencia del sistema. Un buen punto inicial para el filtro es el filtro LTI propuesto anteriormente, y esta elección garantizará una precisión aceptable desde la primera iteración. La idoneidad de esta declaración se mostrará en los ejemplos.

El filtro propuesto incluye características globales y locales, lo que mejora la precisión de la estimación; las características globales están dadas por las propiedades utilizadas en el cálculo de los coeficientes de filtro y las características locales están dadas por los últimos L puntos de la señal. En este caso, las propiedades estadísticas del sistema no son necesarias en la actualización de los coeficientes de filtro. Sin embargo, las propiedades estadísticas se utilizan para obtener una estimación inicial de los coeficientes del filtro como el filtro LTI.

El supuesto en el enfoque propuesto es que la compensación en la respuesta debido al modelo físico incorrecto no afecta en gran medida la precisión de la estimación inicial porque se basa en propiedades estadísticas globales, y cualquier inexactitud se mejorará en la actualización.

Además, debido a que el análisis para la estimación inicial se realiza fuera de línea, la duración de la realización puede ser tan larga como sea necesario y el intervalo de tiempo del análisis tan pequeño como sea necesario para obtener la precisión adecuada. Finalmente, para emph maRTHS, se requiere la estimación de una señal vectorial, luego, el método propuesto se aplica por separado a cada entrada; Debido a la naturaleza del filtro propuesto, se requieren pocos recursos computacionales para implementar el filtro en tiempo real.

3.2.2.3. Interpolación basada en monomios

Una vez que se calcula la estimación del siguiente punto de datos de la señal $\tilde{u}_n$ en el momento t_n utilizando el método descrito en las secciones anteriores, se necesitan puntos de datos adicionales para la mayor frecuencia de muestreo. Se puede utilizar una función de interpolación para obtener estos puntos intermedios de la siguiente manera

$$u(t) = f(t, \tilde{u}_n, u_{n-1}, \ldots, u_{n-l}, t_n, t_{n-1}, \ldots, t_{n-l}) \quad \text{for} \quad t_{n-1} < t < t_n \tag{3.9}$$

donde se utilizan $l + 1$ puntos de datos anteriores $\tilde{u}_n, u_{n-1}, \ldots, u_{n-l}$, los puntos de datos se dan en los tiempos $t_n, t_{n-1}, \ldots, t_{n-l}$.

El tipo de interpolación más simple y común utiliza polinomios. Para un entero dado $k \geq 0$, denotamos por $\mathbb{P}_k$ el conjunto de todos los polinomios de grado k como máximo, definidos en un intervalo dado. Con la suma y la multiplicación escalar definidas de la manera obvia, $\mathbb{P}_k$ forma un espacio vectorial de dimensión $k + 1$. En esta ocasión se selecciona una metodología de interpolación basada en monomios. En esta, para interpolar n puntos de datos, elegimos $k = n - 1$ para que la dimensión del espacio coincida con el número de puntos de datos. La base más natural para $\mathbb{P}_{n-1}$, el espacio vectorial de polinomios de grado como máximo $n - 1$, se compone de los primeros n *monomios*,

$$\phi_j(t) = t^{j-1}, \quad j = 1, \ldots, n \tag{3.10}$$

para el cual un polinomio dado $p_{n-1} \in \mathbb{P}_{n-1}$ tiene la forma

$$p_{n-1}(t) = x_1 + x_2 t + \ldots + x_n t^{n-1} \tag{3.11}$$

En la base monomial, el vector $\mathbf{x}$ de coeficientes del polinomio interpolando los puntos de datos $(t_i, y_i),\ i = 1, \ldots, n$, viene dado por el sistema lineal $n \times n$

$$\mathbf{Ax} = \begin{bmatrix} 1 & t_1 & \ldots & t_1^{n-1} \\ 1 & t_2 & \ldots & t_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & \ldots & t_n^{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \mathbf{y} \tag{3.12}$$

Una matriz de esta forma, cuyas columnas son potencias sucesivas de alguna variable independiente t, se llama *matriz de Vandermonde*. Suponga que $t_i, i = 1, \ldots n$, son todos distintos y que $\mathbf{A}$ es la matriz de Vandermonde correspondiente. Ahora bien, si $\mathbf{Az = 0}$ para algún $z \in \mathbb{R}^n$, entonces el polinomio cuyos coeficientes están dados por $\mathbf{z}$, que es de grado como máximo $n - 1$, tiene n ceros y por lo tanto debe ser el polinomio cero, es decir, $\mathbf{z = 0}$. Nosotros concluir que una matriz de Vandermonde es necesariamente no singular siempre que los t_i sean todos distintos y, por lo tanto, exista el interpolante polinomial.

La interpolación de polinomios y la evaluación de polinomios son inversas entre sí en el siguiente sentido: si $\mathbf{A}$ es una matriz de Vandermonde como se acaba de definir, entonces calcular el producto matriz-vector $\mathbf{Ax}$ evalúa en n puntos el polinomio cuyos coeficientes están dados por las componentes de $\mathbf{x}$, mientras que calcular el producto matriz-vector $\mathbf{A}^{-1}\mathbf{y}$ (resolviendo el sistema lineal $\mathbf{Ax=y}$) determina los coeficientes del polinomio cuyos valores en n puntos están dados por las componentes de $\mathbf{y}$.

Resolver el sistema $\mathbf{Ax=y}$ utilizando un solucionador de ecuaciones lineales estándar para determinar los coeficientes del polinomio de interpolación requiere trabajo $O(n^3)$. Además, cuando se usa la base monomial, la matriz $\mathbf{A}$ de Vandermonde resultante a menudo está mal condicionada, especialmente para polinomios de alto grado. La razón de esto se ilustra en la Figura 3.4, en la que los primeros monomios se grafican en el intervalo [0; 1]. Estas funciones son progresivamente menos distinguibles a medida que aumenta el grado, lo que hace que las columnas de la matriz de Vandermonde sean casi linealmente dependientes. Para

la mayoría de las opciones de puntos de datos t_i, el número de condición de la matriz de Vandermonde crece al menos exponencialmente con el número de puntos de datos n. Por lo tanto, aunque la matriz de Vandermonde es necesariamente no singular en teoría, se vuelve arbitrariamente mal condicionada a medida que n crece y, por lo tanto, los coeficientes polinómicos resultantes se vuelven muy sensibles; para n suficientemente grande, la matriz de Vandermonde se vuelve singular dentro de la precisión de trabajo. Dados estos argumentos, en la práctica se limita el valor de n.

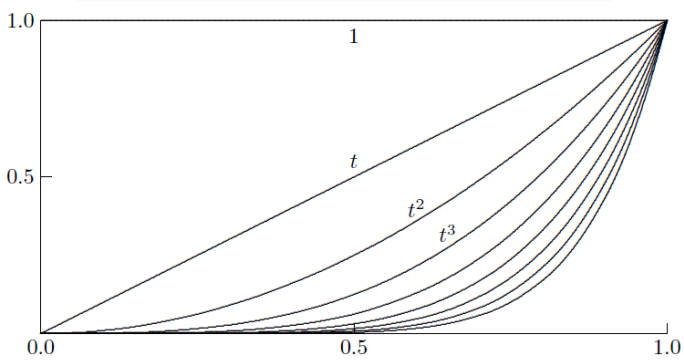


Figura 3.4: Funciones de base monomial. (Heath, 2001)

3.2.2.4. Implementación de la metodología.

La implementación de esta metodología fue llevado a cabo en Matlab/Simulink (Matlab, 2020). A continuación, en la Figura 3.5 y 3.6 se muestran los esquemas generales para el cálculo de los parámetros W fijos y adaptivos, respectivamente.

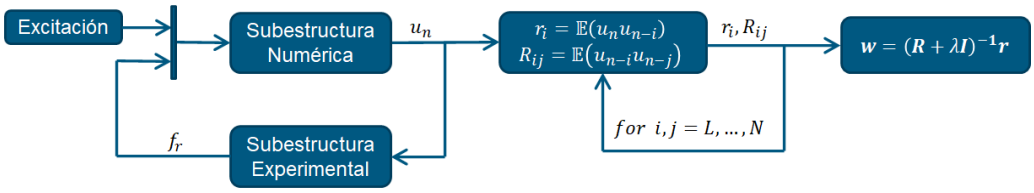


Figura 3.5: Esquema de resolución para el cálculo de parámetros fijos.

En la Figura 3.5 se aprecia que los parámetros W son calculados de forma offline tomando en consideración solo la respuesta de la subestructura numérica.

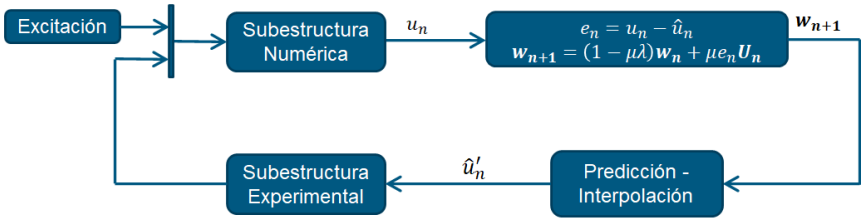


Figura 3.6: Esquema de resolución para el cálculo de parámetros variables.

Para el caso adaptivo, se tiene como punto inicial del ciclo los parámetros W obtenidos como parámetros fijos. En la Figura 3.6 se muestra el como se adaptan estos parámetros en base a una tasa de aprendizaje (μ) y el error (e_n) entre el valor medido versus el valor real.

A continuación, en la Figura 3.7 se muestra la implementación de la metodología dentro de un ciclo de resolución para una simulación híbrida en tiempo real virtual. Es apreciable que se obtiene un diagrama de bloques similar al expuesto en la Figura 3.3. Se aprecia que no existe una retroalimentación entre los bloques, esto es causado por el uso de bloque Goto y From. El bloque Goto pasa su entrada a sus bloques From correspondientes, es decir, permiten pasar una señal de un bloque a otro sin conectarlos realmente. En consecuencia, nos permite tener una simulación mucho más limpia y sin tantos conectores.

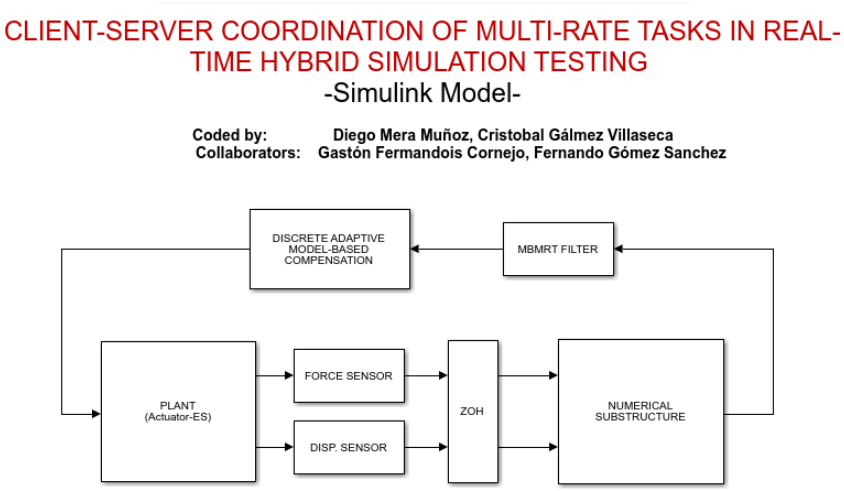


Figura 3.7: Implementación en Simulink de la metodología dentro de una RTHS virtual

3.3. Python - Simulink

En este caso se trabajará con Simulink y la subestructura numérica se encontrará implementada en Python (Van Rossum y Drake, 2012), específicamente en OpenSeesPy (Zhu et al., 2018), dado que este proporciona los medios apropiados, y de sencilla utilización, tanto para establecer comunicación entre sistemas como para la modelación mediante elementos finitos de la subestructura numérica. Adicionalmente, su uso nos permitirá el trabajar el modo externo de Simulink. Para esto, una Raspberry Pi 4 Model B se considerará como hardware externo.

Estas simulaciones consideran el uso de:

- Transición entre tasas basada en datos (Filtro de Wiener) (Subsección 3.2.2)
- Metodología de compensación basada en modelos adaptivos discretos (dAMB) (Subsección 2.4.1.1.2)
- Modelo de actuador servo-hidráulico de Carrion & Spencer (Subsección 2.3.1.1)

En secciones previas analizamos el funcionamiento de Cliente-Servidor. A continuación se muestra como implementar esto en los softwares mencionados. Antes de resolver el ciclo de comunicación, se deben realizar configuraciones previas, estas incluyen definir tanto la IP como el puerto a través del cual se realizará la comunicación. Se decide usar como la dirección IP 0,0,0,0 o también conocida como la dirección no especificada ya que OpenSeesPy actúa como un servidor. Cuando un servidor escucha las conexiones en la dirección 0,0,0,0, significa que el servidor escuchará las conexiones en todas las direcciones IP disponibles en la computadora. En cuanto al puerto de comunicación, se utiliza el 8090 ya que no está reservado para aplicaciones específicas.

El código para enviar y recibir información considera el uso del módulo *structure*. Este módulo realiza conversiones entre valores de Python y estructuras de C representadas como objetos de bytes de Python. Se utiliza para manejar datos binarios almacenados en archivos o desde conexiones de red. Además, se utiliza el módulo con el formato big endian, ya que Simulink envía la información en este formato.

3.3.1. Configuraciones en OpenSeesPy

Se deben realizar ciertas configuraciones en OpenSeesPy en el momento del modelado para asegurar una resolución iterativa, es decir, con envío y recepción de información en cada paso de tiempo. Estas configuraciones se realizarán al resolver la simulación transiente.

Como en cualquier modelo OpenSees, los comandos constraints(), numberer(), system(), test(), algorithm(), integrator() y analysis('Transient') deben usarse para resolver el modelo a través del análisis transiente. Se recomienda que el integrador sea de tipo explícito. En el caso de un tipo implícito, se recomienda el uso de un número fijo de iteraciones. Una vez que estos son definidos por el usuario, se ingresa un bucle for, delimitado entre 0 y el número de datos contenidos en el registro sísmico utilizado (nPts). Dentro de este ciclo suceden cuatro cosas:

i La fuerza de restauración se recibe de la subestructura experimental.

ii La fuerza se impone en en GDL de interfaz (En este caso en la dirección x).

iii La subestructura numérica se resuelve usando el comando analyze() para un solo paso de tiempo. El comando remove() se utiliza para eliminar componentes del modelo. En este caso, el patrón de carga del registro sísmico y la fuerza puntual aplicada. Esto se hace para liberar memoria durante la ejecución.

iv El paquete de datos de desplazamientos de interés se envía a Matlab/Simulink.

Un diagrama del ciclo implementado para este propósito se muestra en la Figura 3.8 a continuación. Un punto importante a destacar es que como se ve en la Figura 3.8 (Parte (ii) Imponer fuerza y parte (iv) Obtener y enviar el desplazamiento para el nodo en cuestión), es posible realizar la subestructuración de tal manera que la compatibilidad de desplazamientos y equilibrio de fuerzas se lleve a cabo, en este ejemplo $2D$, tanto para desplazamiento en las dos direcciones principales (X y Y) y momento en el eje Z (M_z). Por lo tanto, permite RTHS con múltiples actuadores. Asimismo, esto se puede ampliar fácilmente al caso tridimensional.

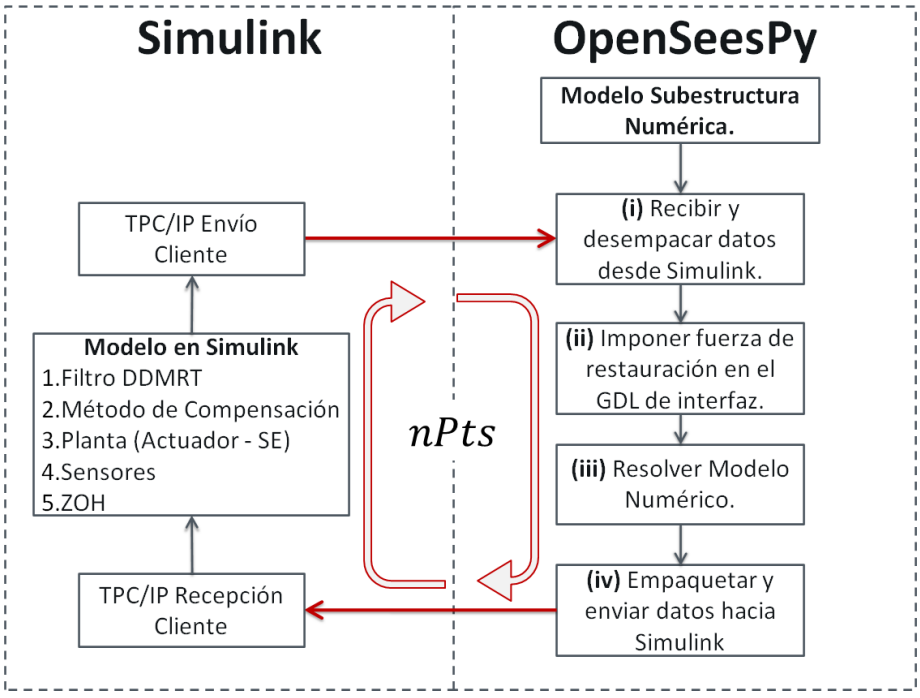


Figura 3.8: Ciclo de resolución implementado en OpenSeesPy para simulación.

3.3.2. Configuración en Simulink

En Matlab/Simulink se utilizan dos modos de simulación para validar la metodología propuesta. El primero consiste en simulaciones locales en una computadora, mientras que el segundo consiste en simulaciones en modo externo, asegurando así la ejecución en tiempo real, pero requiriendo un hardware externo para su resolución.

Simulink Desktop Real-Time (SLDRT) proporciona un kernel en tiempo real para ejecutar modelos de Simulink. Incluye una biblioteca de bloques de controladores de E/S que proporcionan conexiones entre dispositivos de E/S físicos y su modelo en tiempo real. Puede ejecutar su modelo en modo Normal, Acelerador o Externo, según la frecuencia de muestreo que necesite. Los modos Normal y Acelerador se utilizan para las simulaciones locales. A continuación se muestran algunas ventajas, desventajas y diferencias al utilizar Simulink Desktop Real-Time en modo Normal, Acelerador y Externo.

- **Simulación en tiempo real en modo normal:** Si necesita una frecuencia de muestreo moderada de hasta 1 kHz, utilice el modo Normal. En estos modos, el modelo se ejecuta “normalmente"dentro de Simulink. Solo los controladores del módulo de E/S se ejecutan en el kernel en tiempo real. Esto significa que en el modo Normal, SLDRT no garantiza el rendimiento en tiempo real. Sin embargo, es capaz de informar al usuario cuando no se puede lograr un rendimiento en tiempo real mediante el uso de la función Ticks perdidos.
  - Las ventajas del modo normal SLDRT son:
    - No se requieren toolboxes adicionales.
    - La posibilidad de utilizar bloques que no admiten la generación de código.
    - La posibilidad de utilizar cualquier S-Function y llamar a binarios externos.
    - La posibilidad de utilizar solucionadores de pasos variables.
  - Las limitaciones del modo normal SLDRT son
    - No hay garantía de tiempo real (solo una detección si se cumplen o no las restricciones en tiempo real)
    - Rendimiento limitado de hasta 1 kHz.
- **Simulación en tiempo real del Modo Acelerador:** Si ve demasiados tics perdidos mientras ejecuta su modelo en Modo Normal, esto puede significar que su modelo es demasiado grande o complejo para ejecutarse en el tiempo de muestra deseado. Esta es una situación en la que ejecutar en modo Acelerador puede ayudar, que es una extensión del Modo Normal y tiene las mismas ventajas y limitaciones.
- **Simulación en tiempo real del modo externo:** Si necesita una frecuencia de muestreo más alta, utilice el modo externo. En el modo externo, el modelo, el solucionador y los controladores se convierten a código C, se integran en un ejecutable en tiempo real y se ejecutan en el kernel en tiempo real. Esto significa que todo el modelo se está ejecutando en el kernel en tiempo real y el ejecutable está completamente sincronizado con el reloj en tiempo real. Dependiendo de la complejidad del modelo y el número y tipo de E/S, se puede utilizar hasta 20 kHz de frecuencia de muestreo.
  - Las ventajas del modo externo SLDRT son:
    - Garantía de tiempo real.
    - Buen rendimiento de hasta 20 kHz.
    - Recomendado para el funcionamiento con hardware de E/S externo en tiempo real.
  - Las limitaciones del modo externo SLDRT son:
    - Se requiere el toolbox Simulink Coder.
    - Los modelos solo deben usar bloques que admitan la generación de código

- Soporte limitado para S-Functions y sin soporte para binarios externos.
- Se requiere un solucionador de paso fijo.

Además, tenemos que configurar el modo de tareas de Simulink. Esto significa que debemos especificar si Simulink ejecuta bloques con tiempos de muestreo periódicos de forma individual o en grupos. En la sección Solver de la configuración de Simulink, debemos dejar la opción de "Treat each discrete rate as a separate task" sin seleccionar. Es decir, especifica que todos los bloques se procesan a través de cada etapa de simulación juntos.

3.3.2.1. Simulación Local

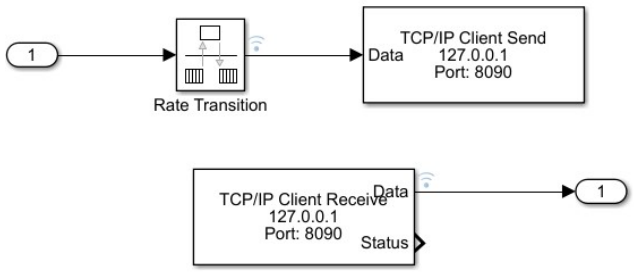


Figura 3.9: Bloques utilizados en una simulación local.

Los bloques de recepción y envío de TCP/IP, que se pueden ver en la Figura 3.9, configuran y abren una interfaz a la dirección remota especificada usando el protocolo TCP/IP. Son bloques de simulink nativos y no admiten el modo externo. Es necesario utilizar bloques en tiempo real. Se utiliza el bloque de transición de velocidad que se puede ver, ya que los bloques de comunicación no admiten señales continuas. Además, se está trabajando en un sistema discreto. La dirección IP que se muestra corresponde a la IP local de cualquier computadora.

3.3.2.2. Simulación Externa

Una simulación de modo externo establece un canal de comunicación entre Simulink en su computadora de desarrollo (host) y el hardware de destino que ejecuta el archivo ejecutable creado por el proceso de generación y compilación de código.

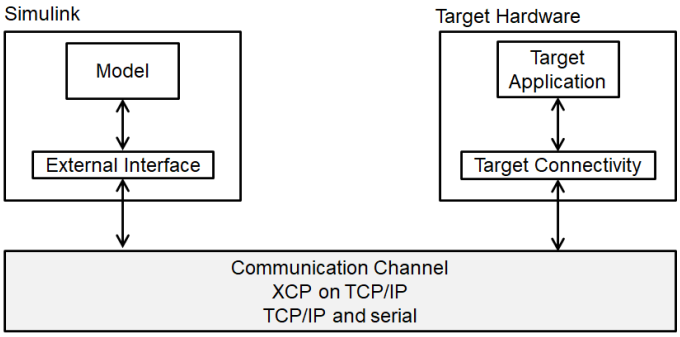


Figura 3.10: Esquema de simulación para modo externo

Como se puede ver en la Figura 3.10, la capa de transporte de bajo nivel del canal maneja la transmisión física de mensajes. Simulink y el código de modelo generado son independientes de esta capa. La capa de

transporte y su código de interfaz están aislados en módulos separados que formatean, transmiten y reciben mensajes y paquetes de datos.

Las simulaciones en modo externo se ejecutan en una Raspberry Pi 4 Modelo B. Las configuraciones para el caso se muestran a continuación.

3.3.2.2.1 Raspberry Pi

La Raspberry Pi es una computadora de bajo costo del tamaño de una tarjeta de crédito que se conecta a un monitor de computadora o TV y utiliza un teclado y un mouse estándar. Es capaz de hacer todo lo que esperaría que hiciera una computadora de escritorio, desde navegar por Internet y reproducir videos de alta definición, hasta crear hojas de cálculo, procesar textos y jugar. Además, la Raspberry Pi tiene la capacidad de interactuar con el mundo exterior y se ha utilizado en una amplia gama de proyectos de creadores digitales.

El primer paso para configurar la Raspberry Pi es descargar e instalar el paquete de soporte de Simulink para Raspberry Pi. Este debe instalarse en una tarjeta micro sd siguiendo las instrucciones dadas por Mathworks (MathWorks, 2019). Este paquete le permite desarrollar algoritmos que se ejecutan de forma independiente en su Raspberry Pi. El paquete de soporte amplía Simulink con bloques para impulsar la E/S digital Raspberry Pi y leer y escribir datos desde ellos. Después de crear su modelo de Simulink, puede simularlo y descargar el algoritmo completo para su ejecución independiente en el dispositivo. Una capacidad particularmente útil que ofrece Simulink es la capacidad de ajustar los parámetros en vivo desde su modelo de Simulink mientras el algoritmo se ejecuta en el hardware.

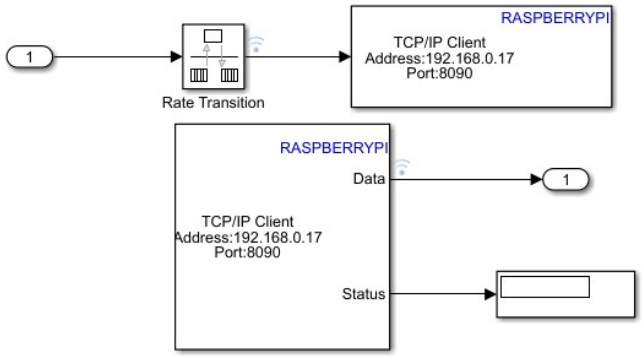


Figura 3.11: Bloques utilizados en una simulación externa.

El paquete de soporte de Simulink para hardware Raspberry Pi le permite crear y ejecutar modelos de Simulink en hardware Raspberry Pi. El paquete de soporte incluye una biblioteca de bloques Simulink, como los que se muestran en la Figura 3.11, para configurar y acceder a los periféricos de E/S e interfaces de comunicación. Estos bloques que se muestran aquí se utilizan para comunicarse con el modelo numérico ejecutado y desarrollado en OpenSeesPy. Esta vez la IP utilizada corresponde a la del ordenador donde se está ejecutando el modelo numérico en OpenSeesPi.

3.3.3. Plataforma vRTHS Cliente-Servidor

La implementación de la simulación híbrida en tiempo real virtual (vRTHS por sus siglas en inglés, virtual Real-Time Hybrid Simulation) bajo la arquitectura Cliente-Servidor se ilustra en la Figura 3.12. Las instrucciones detalladas de instalación y ejecución del software se proporcionan en el repositorio complementario de GitHub Mera y Fermandois (2021). Brevemente, el primer requisito es iniciar el servidor

SN-EF, que está esperando a que se inicie la simulación (aparecerá en la pantalla un mensaje de "*Esperando a que se inicie Simulink*"). Luego, se debe ejecutar Simulink Cliente, el cual solicita un desplazamiento objetivo a OpenSeesPy para la ejecución del primer paso temporal. Se utiliza un esquema de integración explícito, como el método Alpha-OS (SN) y el método Dormand-Prince (Simulink Client). Los intervalos de muestreo utilizados para los procesos rápido (cliente y espécimen físico) y lento (SN-EF servidor) son 1/4000 y 2/100 [s], respectivamente (es decir, $0 < T_{Fast} < T_{Slow}$). Para resumir, los bucles deben correr desde la parte superior a la inferior de las capas: (1) el servidor SN-EF; y (2) Cliente de Simulink y muestra física.

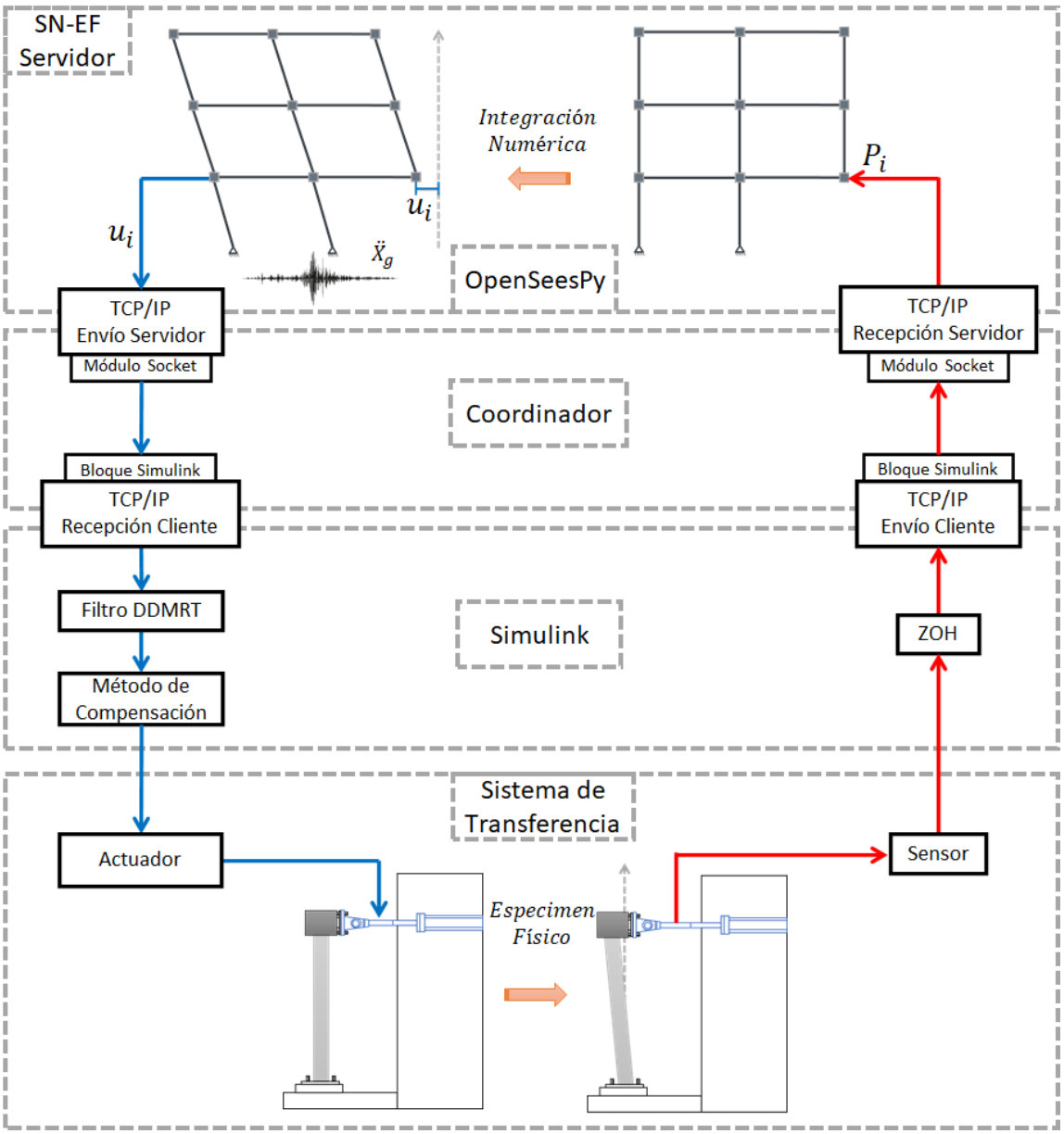


Figura 3.12: Ciclos de simulación e implementación del framework considerados en vRTHS.

4 | Aplicaciones Numéricas

4.1. Criterios de Evaluación

El rendimiento del vRTHS se evalúa a través de cuatro indicadores de rendimiento, que se muestran en la Tabla 4.1. J_2 es el error cuadrático medio normalizado entre el desplazamiento objetivo (x_t) y el desplazamiento medido (x_m); este indicador mide el error de sincronización. J_4 es el error cuadrático medio normalizado entre el desplazamiento de referencia (x_{ref}) y el desplazamiento medido (x_m), que mide el error entre la simulación y la estructura de referencia. Tanto los índices J_2 como J_4 se proporcionaron en Silva et al. (2020). Además, τ corresponde al indicador de retardo obtenido con el índice de evaluación de frecuencia (Guo et al., 2014), que mide el retardo de sincronización (en milisegundos) entre x_t y x_m. Finalmente, MT son los ticks perdidos del reloj en tiempo real en Simulink Desktop Real-Time, que se discutirán en la sección de implementación del sistema en tiempo real. El límite superior establecido por el experimentador es de 500.

Tabla 4.1: Criterios de evaluación del desempeño de la vRTHS.

Criterio	Descripción	Definición	Unidad
J_2	Raíz cuadrada media normalizada del error de seguimiento	$J_2 = \sqrt{\frac{\sum_{i=1}^{N}[x_m(i)-x_t(i)]^2}{\sum_{i=1}^{N}[x_t(i)]^2}} \cdot 100$	%
J_4	Raíz cuadrada media normalizada del error respecto a la referencia	$J_4 = \sqrt{\frac{\sum_{i=1}^{N}[x_m(i)-x_{ref}(i)]^2}{\sum_{i=1}^{N}[x_{ref}(i)]^2}} \cdot 100$	%
τ	Retraso de tiempo entre las señales de entrada y salida evaluadas a la frecuencia equivalente	$\tau = -\frac{\phi_0}{2\pi f^{eq}}$	ms
MT	Ticks perdidos del reloj en tiempo real y sus puntos de datos asociados.	Calculado internamente por Simulink.	*integer*

4.2. Implementación del sistema en tiempo real - OpenFresco

Para los casos estudio 1, 2 y 3, tres programas informáticos serán utilizados. Primero, Matlab/Simulink (Matlab, 2020) proporciona los medios más convenientes para implementar algoritmos de control para RTHS. Sin embargo, Matlab no es la mejor opción para modelar sistemas estructurales con comportamiento no lineal dado que no proporciona una biblioteca de elementos y materiales adecuados para el modelado estructural. Por lo tanto, se prefiere el software OpenSees (McKenna, 2011) para modelar subestructuras numéricas. Por último, el marco de código abierto para configuración y control experimentales (OpenFresco) (Schellenberg et al., 2009a,b) permite la comunicación desde/hacia OpenSees y un modelo de Simulink que se ejecuta en tiempo real, y combina una amplia variedad de algoritmos de simulación híbridos, sistemas de control y de laboratorio, configuraciones experimentales y modelos de simulación computacional para una simulación híbrida específica.

Estas simulaciones consideran el uso de:

- Transición entre tasas utilizando el algoritmo predictor-corrector (Subsección 3.2.1)
- Metodología de compensación basada en modelos adaptivos (AMB) (Subsección 2.4.1.1.1)
- Modelo de actuador servo-hidráulico de Carrion & Spencer (Subsección 2.3.1.1)

Finalmente, los valores obtenidos para los gains (Γ_i) y las condiciones iniciales (a_i) necesarios para el AMB son:

- Γ_i=diag($10^{6,03}$ $10^{4,23}$ $10^{1,48}$ $10^{-1,10}$)
- a_i = [1,0 $1{,}9 \cdot 10^{-2}$ 10^{-4} 0,0]

La implementación de vRTHS se ilustra en la Figura 4.1. Las instrucciones detalladas de instalación y ejecución del software se proporcionan en el repositorio complementario de GitHub (Fermandois y Mera, 2021). Brevemente, el primer requisito es iniciar el servidor SE-EF, que está esperando que comience la simulación. Luego, se debe ejecutar Simulink Cliente que solicita un desplazamiento de destino a OpenSees para la ejecución del primer paso temporal. Finalmente, el servidor SN-EF que contiene la subestructura numérica realiza la integración numérica y comienza la simulación. Un esquema de integración explícito, como el método de diferencia central (SN-EF) y el método Dormand-Prince (Simulink Client). Los intervalos de muestreo utilizados para los procesos rápido (cliente y servidor SE-EF) y lento (servidor SN-EF) son 1/1024 y 20/1024 [s], respectivamente (es decir, $0 < T_{Fast} < T_{Slow}$). Para resumir, los bucles deben ejecutarse desde las capas internas hacia las externas: (1) servidor SE-EF (verde); (2) Cliente Simulink (naranja); y (3) el servidor SN-EF (azul grisáceo).

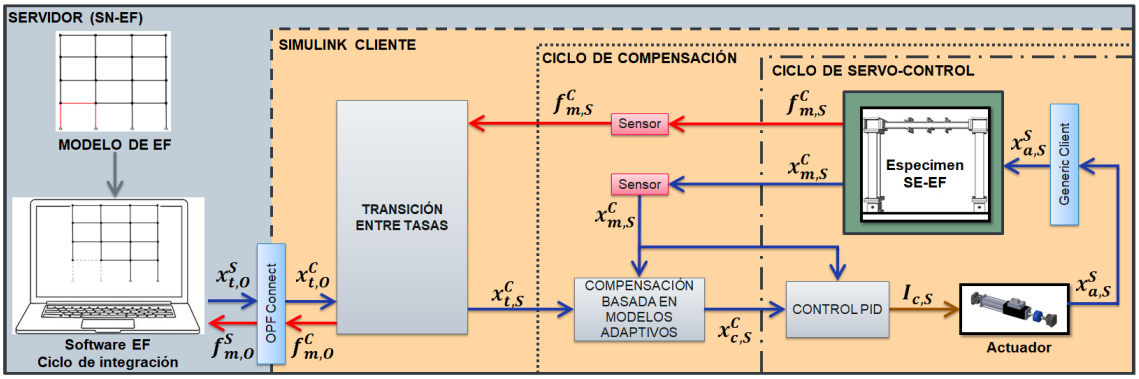


Figura 4.1: Ciclos de simulación considerados en vRTHS.

4.3. Caso Estudio 1, Modelo 2D

En este caso se selecciona como estructura de referencia un marco plano de cuatro pisos con tres vanos, con un total de 52 grados de libertad (16 laterales, 16 verticales y 20 rotacionales). El modelo utiliza elementos beam-column utilizando como material, acero Giuffre-Menegotto-Pinto con endurecimiento por deformación isotrópica con una rigidez inicial de 5,6$[kip/in]$. En consecuencia, el período fundamental del primer modo es $T_1 = 1{,}9[s]$. La estructura de referencia se divide en subestructura numérica (NS) y subestructura experimental (ES), como se muestra en la Figura 4.2. Un marco del primer piso se toma como ES, mientras que el resto se modela como NS. Dado que está diseñado para funcionar solo con un actuador uniaxial, los grados de libertad límite se establecen para que sean solo desplazamientos horizontales.

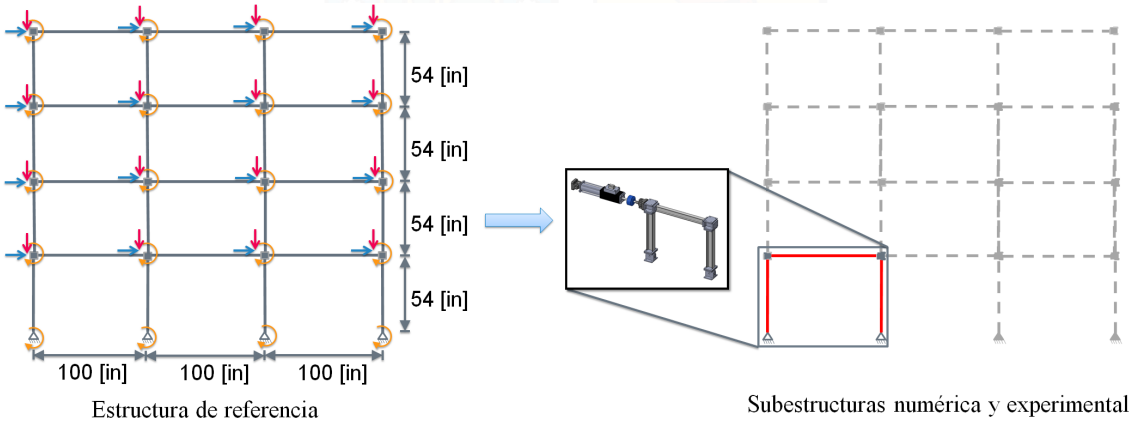


Figura 4.2: Subestructuras a resolver en la vRTHS.

Los resultados de la simulación se presentan gráficamente a continuación. Para las comparaciones de desplazamiento, el nodo a la izquierda del primer piso (nodo superior izquierdo del ES) se selecciona como representativo. En la Figura 4.2, el desplazamiento medido se compara con el desplazamiento objetivo para evaluar el método de compensación. Con $J_2 = 0{,}759[\,\%]$ y $\tau = 0{,}1713[msec]$, se logra un buen seguimiento, lo que permite una prueba estable.

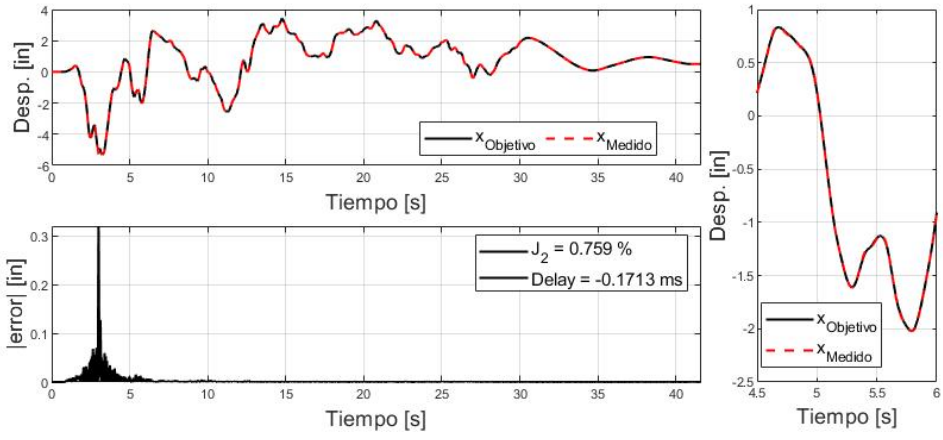


Figura 4.3: Desplazamiento Objetivo v/s Medido e indicador J_2.

Mientras tanto, la Figura 4.4 muestra una comparación entre el desplazamiento medido y el desplazamiento de referencia. El indicador J_4 alcanza un valor de 3,76[%], lo que significa que se obtuvieron

resultados correctos y precisos. El desplazamiento de referencia se obtiene de un modelo realizado íntegramente en el software OpenSees.

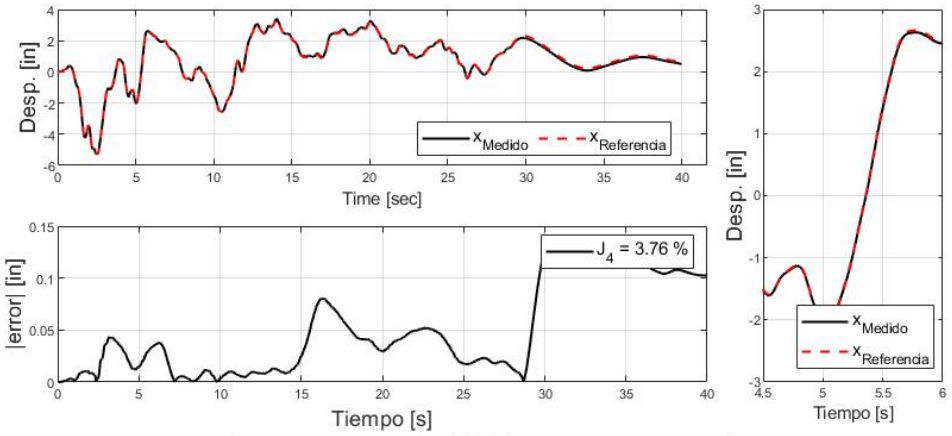


Figura 4.4: Desplazamiento de Referencia v/s Medido e indicador J_4.

Finalmente, en la Figura 4.5 se agrega una comparación en términos globales de la estructura. Para este propósito, se hace la gráfica de desplazamiento de piso del techo versus cortante de la base. Nuevamente se observa una buena concordancia de los resultados medidos con los de referencia, validando así la metodología de subestructuración y compensación.

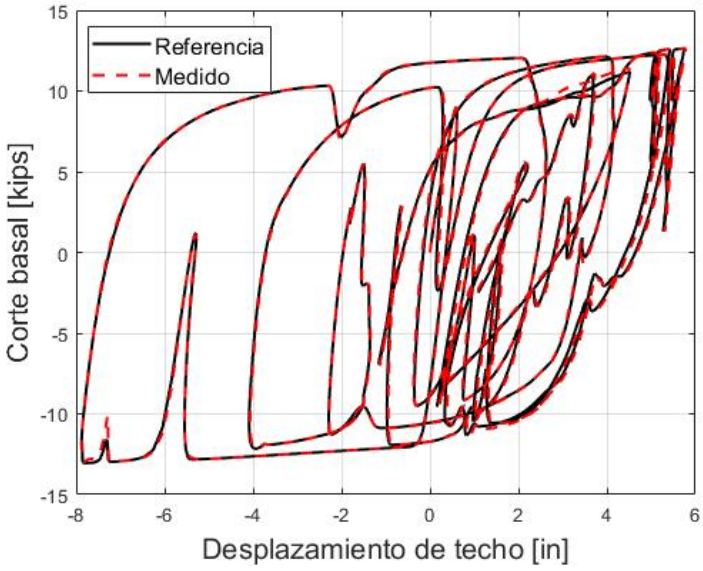


Figura 4.5: Histéresis de la Subestructura experimental (Referencia v/s Medido)

La confiabilidad de la simulación se evalúa mediante un conjunto de 40 ejecuciones de simulación. Las simulaciones se llevaron a cabo en un notebook MSI, modelo GF75 Thin 9RCX (procesador Intel Core i7-9750H de 2.6 GHz, con 8 Gb de RAM), ejecutando Windows 10 versión 20H2, ejecutando Simulink Desktop Real-Time en Modo Acelerador y bajo temperatura de la habitación controlada ($19°C$). Los resultados se muestran en la Figura 4.6, que presenta estadísticas de ticks perdidos durante la ejecución en tiempo real. El valor máximo de tics perdidos es 426, que es el peor de los casos. Por lo tanto, nunca se excede el límite de 500

(establecido por el experimentador) y se logra un protocolo de comunicación rápido y eficiente. Se observa que los ticks perdidos se concentran en los primeros 3 a 4$[s]$. Por lo tanto, el registro sísmico se rellena con ceros durante los primeros 5$[s]$ para verificar que la excitación no causa este fenómeno. Esta sobrecarga de comunicación probablemente se deba al inicio del protocolo de comunicación entre aplicaciones.

Para la simulación de estado estable en tiempo real (es decir, tiempos después de 4$[s]$), las estadísticas de ticks perdidos se reducen significativamente. El gráfico inferior derecho de la Figura 4.6 muestra que en el peor de los casos se alcanza una máxima de 18 tikcs perdidos y una mediana de cero. Los picos de ticks perdidos se generan en intervalos regulares de 20/1024$[s]$, coincidiendo con la duración del proceso de velocidad lenta (EF-SN Servidor). Es apreciable que es un número bajo y estable de ticks perdidos, en comparación con los picos anteriores, por lo que se asume una mayor precisión y exactitud en los resultados después del inicio 4$[s]$ de la simulación. Esto fomenta el uso de un registro sísmico con 5$[s]$ de relleno de ceros al inicio, lo que garantiza que no se pierda información crítica.

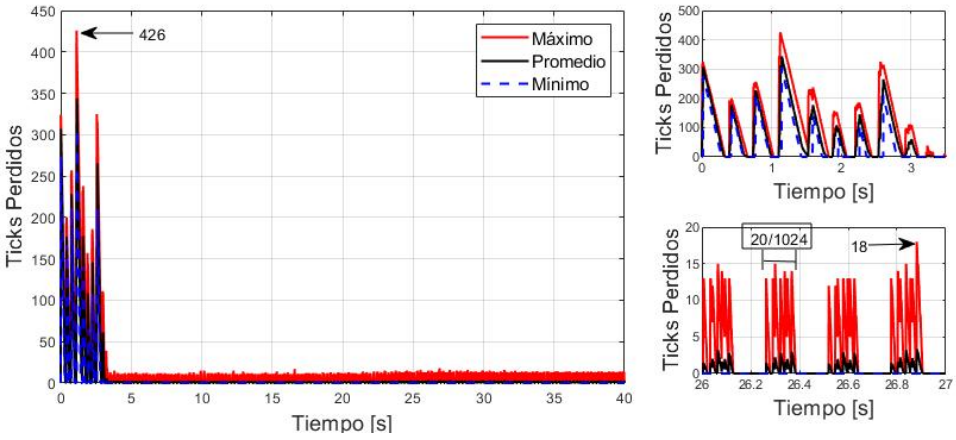


Figura 4.6: Ticks perdidos y métricas para 40 simulaciones

4.4. Caso Estudio 2, Modelo 3D

Para la simulación siguiente, y buscando evaluar modelo numéricos más complejos, se trabaja con una estructura tridimensional. En la Figura 4.7 se muestran las subestructuras trabajadas. Aquí un modelo numérico de 5 pisos y 2 vanos es utilizado. Por otro lado, la subestructura experimental corresponde a una diagonal ubicada en el primer piso sobre el plano XZ. La subestructura numérica fue generada trabajando secciones de tipo fibra e incluyendo una rigidez torsional lineal desacoplada de los demás GDL. Las vigas son $W40x297$ con un peso propio de 297 $\left[\frac{lbf}{ft}\right]$ mientras que las columnas son $W40x503$ con un peso propio de 503 $\left[\frac{lbf}{ft}\right]$. Se consideró de forma adicional una sobrecarga para oficina con equipos de 5$[KPa]$ correspondiente a 104,427 $\left[\frac{lbf}{ft^2}\right]$ y carga muerta extra (Losa) de 2$[KPa]$ correspondiente a 41,7709 $\left[\frac{lbf}{ft^2}\right]$. Estos valores son obtenidos desde la norma chilena (Nch. 1537 del 2009) de Diseño estructural - Cargas permanentes y cargas de uso (INN, 2009).

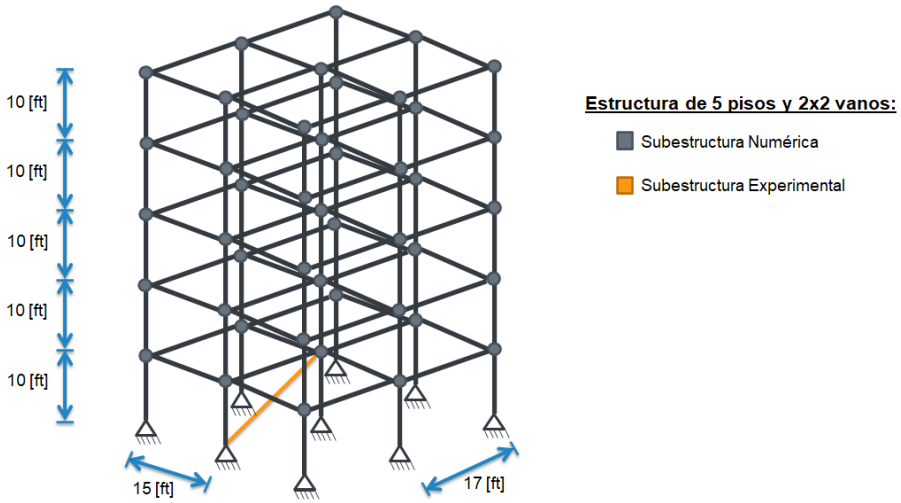


Figura 4.7: Subestructuras a resolver en la vRTHS.

Al ser un modelo tridimensional, es posible considerar registros sísmicos de entrada en múltiples direcciones. En este caso se seleccionan las dos componentes del registro *El Centro*, estas se muestran en la Figura 4.8.

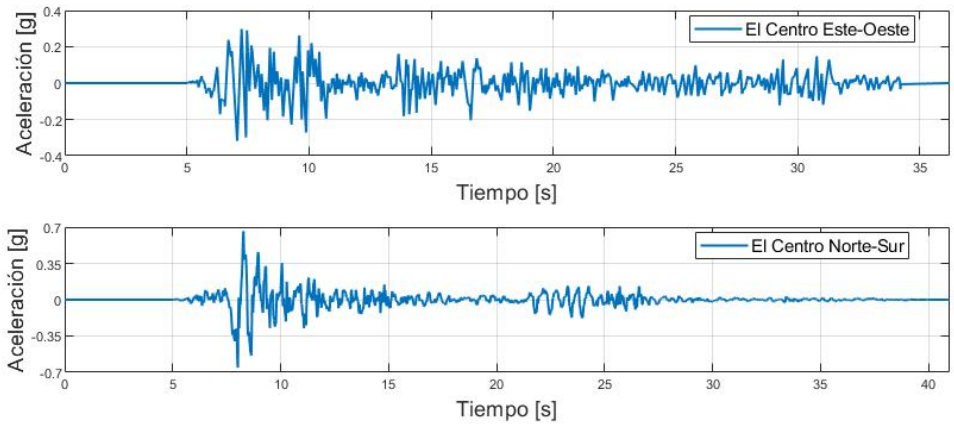


Figura 4.8: Registros sísmicos El Centro 1940, estación El Centro.

Posterior a un análisis modal de la estructura se obtienen los periodos y frecuencias natuales de esta. Los resultados para los primeros 5 modos se muestran a continuación.

Tabla 4.2: Periodos y Frecuencias Naturales para los primeros 5 modos de vibrar de la estructura

Modo	Periodo [s]	Frecuencia [Hz]
1	0.7092	1.4101
2	0.2814	3.554
3	0.2049	4.8811
4	0.1136	8.8009
5	0.0815	12.2681

A continuación, se presentan gráficamente los resultados de la simulación. En la Figura 4.9, el

desplazamiento medido se compara con el desplazamiento objetivo para evaluar el método de compensación. Con $J_2 = 0{,}48249[\,\%]$ y $\tau = 0{,}00711[msec]$, se logra un buen seguimiento, lo que permite una prueba estable.

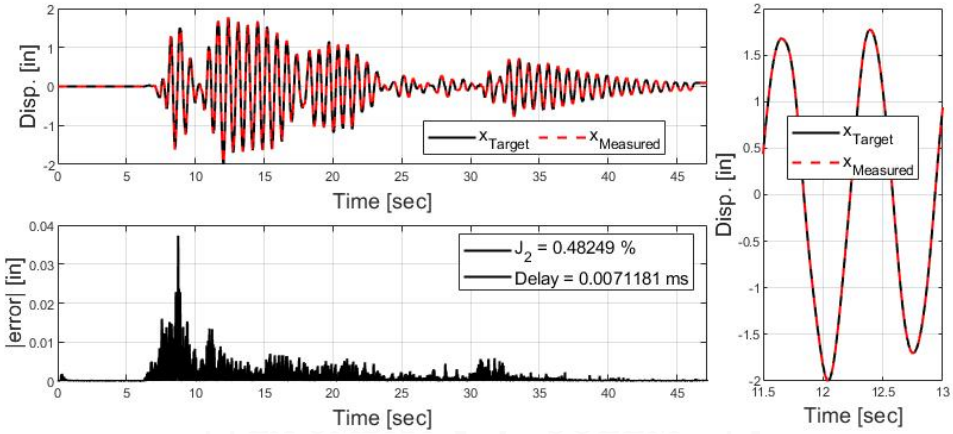


Figura 4.9: Desplazamiento Objetivo v/s Medido e indicador J_2.

Por otro lado, las Figuras 4.10, 4.11 y 4.12 muestran una comparación entre el desplazamiento medido y el desplazamiento de referencia. La figura 4.10 muestra la comparación respecto al eje Y, es decir, en el plano sobre el cuál no actúa la diagonal. Por consiguiente, se obtiene un error de máquina.

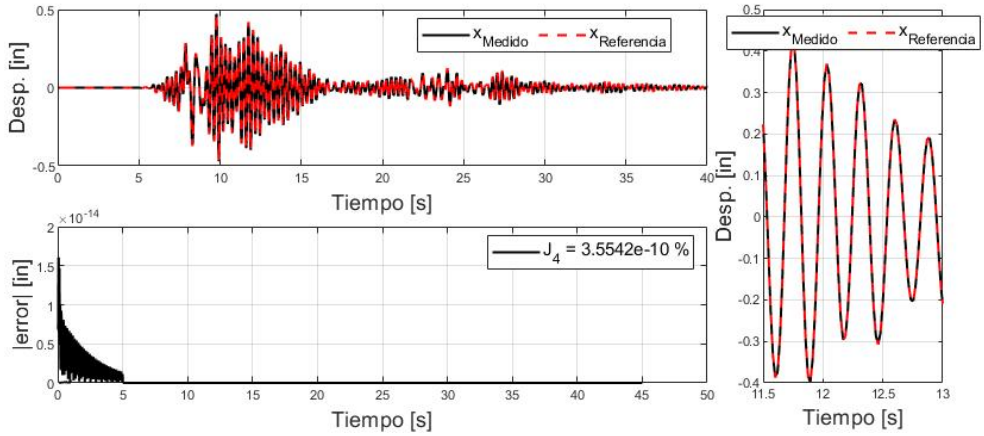


Figura 4.10: Desplazamiento de Referencia v/s Medido e indicador J_4 para grado de libertad vertical en Y.

Las Figuras 4.11 y 4.12 corresponden al desplazamiento en el plano sobre el cuál se encuentra la diagonal, por tanto, son los resultados que nos interesa analizar. El indicador J_4 alcanza un valor de $0{,}1175[\,\%]$ en dirección X y $0{,}25[\,\%]$ en dirección Z, lo que significa que se obtuvieron resultados correctos y precisos. El desplazamiento de referencia se obtiene de un modelo realizado íntegramente en el software OpenSees.

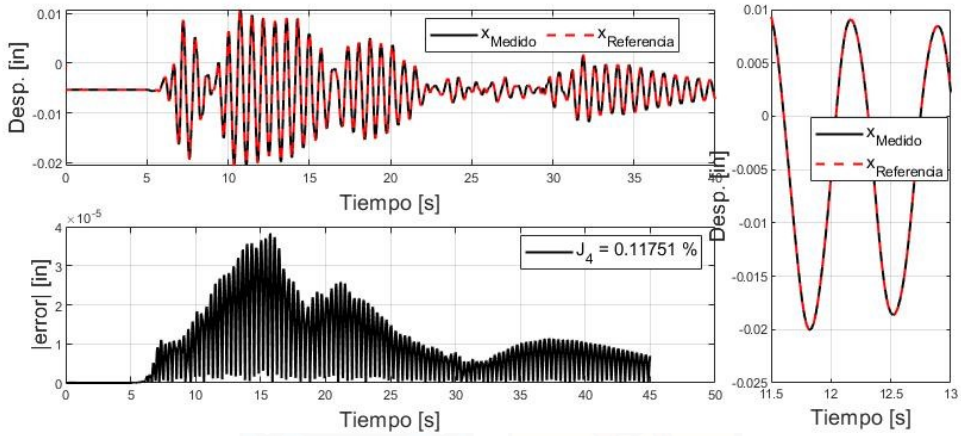


Figura 4.11: Desplazamiento de Referencia v/s Medido e indicador J_4 para grado de libertad horizontal X.

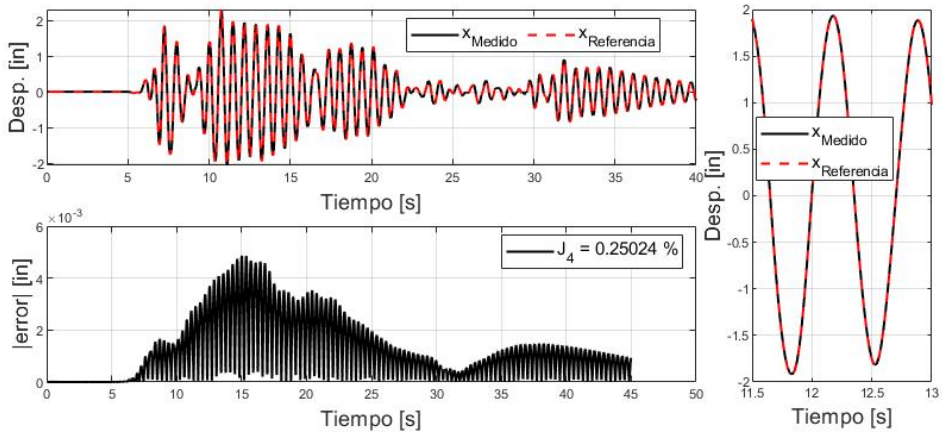


Figura 4.12: Desplazamiento de Referencia v/s Medido e indicador J_4 para grado de libertad Horizontal Z.

Nuevamente, la confiabilidad de la simulación se evalúa mediante un conjunto de 40 ejecuciones de simulación. Los resultados se muestran en la Figura 4.13, que presenta estadísticas de ticks perdidos durante la ejecución en tiempo real. El valor máximo de tics perdidos es 405 indicando que el límite de 500 no es excedido. Se observa que los ticks perdidos, al igual que para el caso bidimensional, se concentran en los primeros 3 a 4$[s]$. Por lo tanto, el registro sísmico se rellena con ceros durante los primeros 5$[s]$ para verificar que la excitación no causa este fenómeno. Esta sobrecarga de comunicación probablemente se deba al inicio del protocolo de comunicación entre aplicaciones.

Para tiempos superiores a 4$[s]$, las estadísticas de ticks perdidos se reducen significativamente. El gráfico inferior derecho de la Figura 4.13 muestra que en el peor de los casos se alcanza nuevamente una máxima de 18 tikcs perdidos y una mediana de cero. De igual forma al caso 2D, los picos de ticks perdidos se generan en intervalos regulares de $20/1024[s]$, coincidiendo con la duración del proceso de velocidad lenta (EF-SN Servidor).

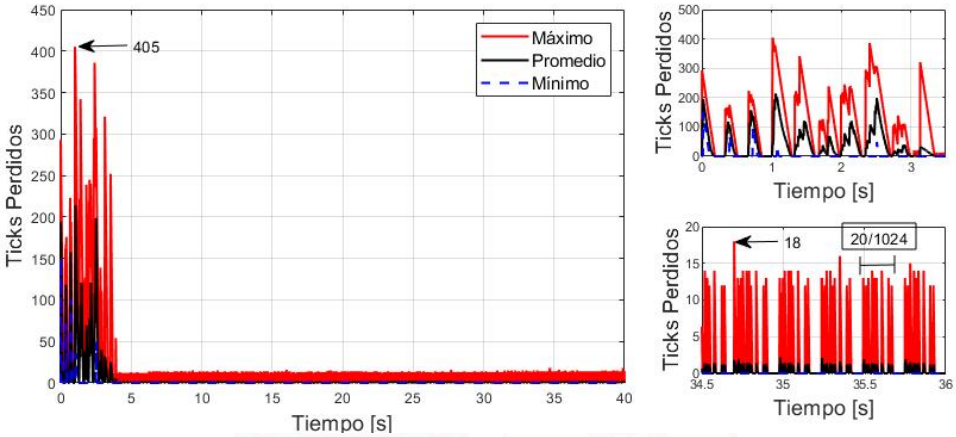


Figura 4.13: Ticks perdidos y métricas para 40 simulaciones

4.5. Caso Estudio 3, Análisis Isogeométrico

Tal como se mencionó en capítulos previos, se suelen utilizar entornos netamente virtuales para simulaciones previas a los ensayos en laboratorio (Silva et al., 2020). Hasta la fecha, para trabajar con subestructuras más complejas y realistas se utilizan herramientas de análisis con métodos de elementos finitos, por ejemplo, el software de análisis de elementos finitos OpenSees (McKenna, 2011). Sin embargo, este posee algunas limitantes en su utilización. Tales como el no permitir realizar un mallado de forma automática, por lo que el ingreso de los elementos se debe hacer de forma manual y uno a la vez, definiendo nodos y su conectividad. Adicionalmente, los elementos disponibles son escasos (Tri3, Quad4, Tetrahedro y Elementos fibra), por lo que si se desea trabajar con otro elemento se debe incorporar al código fuente y generar un nuevo ejecutable, proceso que puede resultar engorroso y difícil de realizar. A pesar de sus capacidades, el uso de OpenSees se ha visto relegado a estructuras lineales simplificadas (e.g. edificios de corte (Ghaffary, 2019)).

Por otro lado, el análisis isogeométrico (IGA) surge como alternativa al método de elementos finitos, con la ventaja de que las funciones de base son generadas a partir de la geometría original, por lo que es capaz de conservar la geometría exacta del elemento en todas las etapas del análisis. Esto se logra utilizando funciones de base generadas desde NURBS (*Non-Uniform Rational B-Splines*) Hughes et al. (2005), lo que permite representar objetos curvos con precisión, como se verifica en Lu (2009). Cabe destacar que, con el método de elementos finitos (FEM), esto solamente se puede aproximar realizando un mallado extremadamente fino.

Dadas estas características del método de IGA, se propone la hipótesis de que su precisión en aplicaciones con *vRTHS* mejorará respecto al análisis con FEM. Sin embargo, hasta la fecha no se cuenta con literatura que vincule los estudios de simulación híbrida con las técnicas de análisis de IGA. Luego, los objetivos de este estudio son comparar el desempeño de las metodologías FEM e IGA y explicitar la versatilidad de la plataforma de coordinación permitiendo conectar modelos personalizables y/o no tradicionales. Para esto, se resolverá una estructura correspondiente a un puente de hormigón de cuatro vanos, donde se considerará una de las pilas como subestructura experimental utilizando el programa Matlab/Simulink (Matlab, 2020), dado que proporciona los medios más convenientes para implementar algoritmos de control para RTHS. Lo expuestos en las secciones 2.6 y 2.7 será implementado y utilizado en este caso estudio.

4.5.1. Modelamiento de la cepa

Se utiliza el Software Rhinoceros 3D Nurbs (2006) para el diseño de la cepa, el cual está basado en NURBS, desde el cual es posible extraer la información de la geometría en función de: (i) las coordenadas de los puntos de control, (ii) los pesos de los puntos y (iii) los vectores uKnot y vKnot. Estos últimos son requeridos al momento de emplear el algoritmo de resolución con el método IGA.

Por otro lado, desde el mismo programa se puede extraer como superficie la geometría en formato `.stl`, el cual ese puede importar directamente desde Matlab para la resolución por FEM.

4.5.2. Simulación híbrida

En términos generales, lo que se busca resolver es un ciclo de la forma mostrada en la Figura 4.14. En este ejemplo, y tal como se menciona previamente, se resolverá un puente de 4 vanos. Para efectos prácticos, se simplifica el análisis a 2 dimensiones, siendo la dirección longitudinal del puente la dirección a analizar.

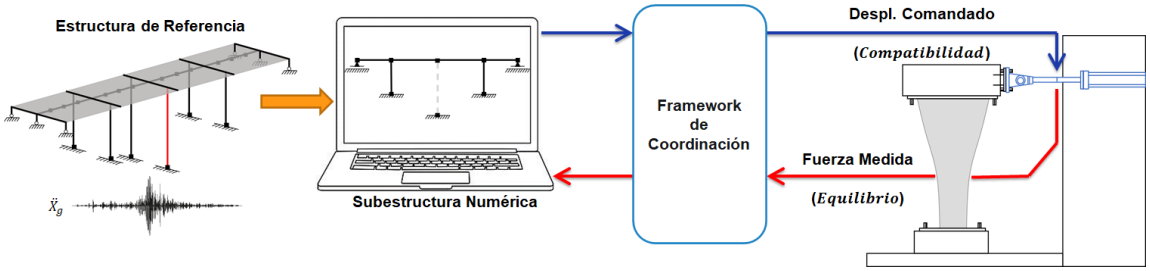
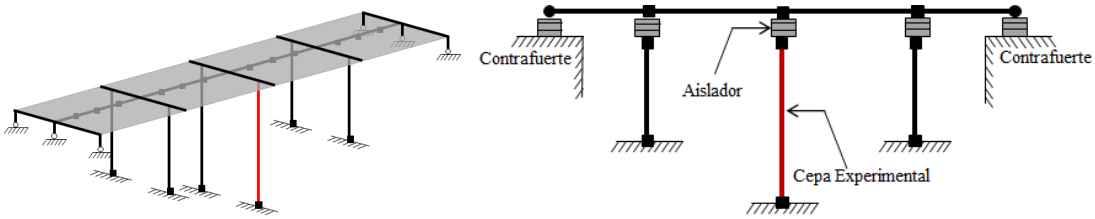


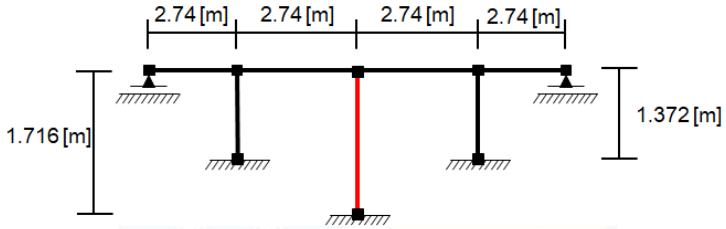
Figura 4.14: Ciclo resumen de la resolución de la vRTHS.

Modelo de la estructura:

Se trabaja con un modelo de un puente con tres cepas en la dirección longitudinal y dos en la dirección transversal, como se muestra en la Figura 4.15(a). En este estudio, la estructura se analiza en dirección transversal, como se muestra en la Figura 4.15(b) y cuyas dimensiones se aprecian en la Figura 4.15(c). Las propiedades mecánicas del material se presentan en la Tabla 4.3. En cuanto a los elementos numéricos en OpenSees, las cepas son modeladas a través de elementos beam-column, los aisladores y bisagras en la parte superior de las columnas con elementos zero-length.



(a) Modelo 3D del puente en estudio. **(b)** Simplificación del modelo a 2D.



(c) Dimensiones del puente a trabajar.

Figura 4.15: Modelo del puente que se utiliza en vRTHS.

Propiedades	Valor
Largo tramos	2.74 m
Área contribuyente	0.3 m^2
Altura columna experimental	1.716 m
Altura columnas numéricas	1.372 m
Espesor columnas	0.35 m
Densidad hormigón	2.423.000 kg/m^3
Módulo elasticidad hormigón	2.3414e10 N/m^2

Tabla 4.3: Valores de las propiedades del material y las dimensiones del modelo.

La geometría de las cepas del puente cuentan con un diseño curvo. Esta es apreciable en la Figura 4.16, junto con sus dimensiones respectivas.

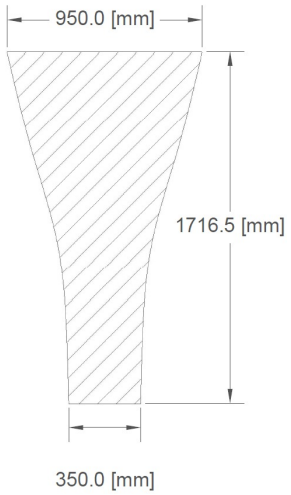


Figura 4.16: Dimensiones de la cepa en estudio.

Finalmente, la configuración de la simulación híbrida virtual se muestra en la Figura 4.17. Como se

aprecia, esta Figura es similar a la Figura 4.1 mostrada en la sección 4.2. La principal diferencia radica en la subestructura experimental, en esta ocasión se resuelve un elemento curvo mediante análisis isogeométrico implementado en el ambiente Matlab/Simulink, por otro lado, la figura original presenta a la subestructura experimental siendo resuelta en el software de elementos finitos OpenSees y comunicando a través de una S-Function. Con esta nueva figura se puede explicar brevemente en qué consiste la simulación: las subestructuras numéricas y experimentales (SN y SE, respectivamente) deben comunicarse en tiempo real, lo cual es manejado por un software coordinador, Simulink en este caso. En cada paso de tiempo del análisis, la SN resuelve la ecuación de movimiento a través de la integración numérica y produce un desplazamiento objetivo (x_t) en la interfaz. Esta señal se impone a la SE en el laboratorio a través de un sistema de carga, en este caso, se utiliza un modelo numérico, tanto del controlador, como del actuador hidráulico. Un aspecto crítico en RTHS, es que la aplicación del desplazamiento del objetivo debe realizarse en tiempo real; cualquier retraso en la comunicación puede provocar resultados inexactos y una posible inestabilidad, es por esto que se trabaja con un algoritmo de compensación que se encarga de corregir y evitar estos problemas. Después de que se impone este desplazamiento, la fuerza de restauración se mide a partir de la muestra y se devuelve al integrador de la SN para el siguiente paso.

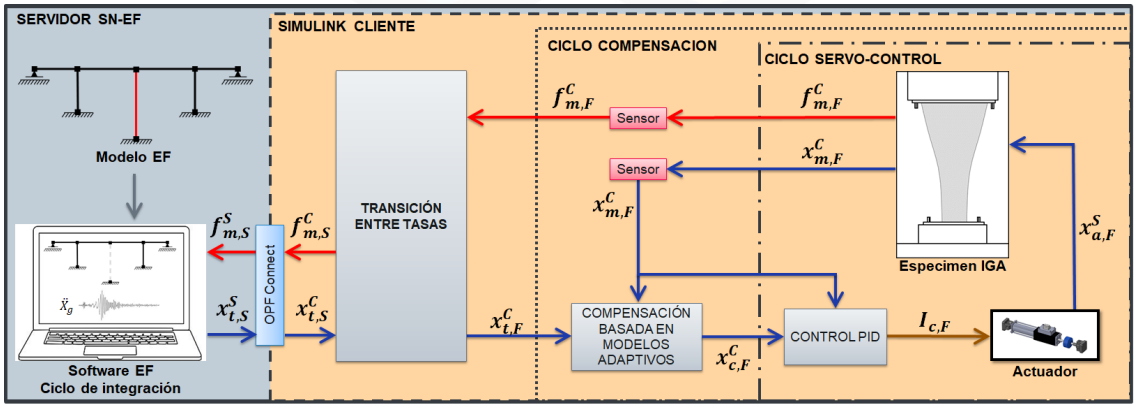


Figura 4.17: Ciclo de resolución y ejecución de la vRTHS. Se incluyen las subestructuras numéricas y experimentales, compensación del retraso y error de señales, sensores de medición, modelo y controlador del actuador.

4.5.3. Resultados

Comprobación implementación de los métodos:

Para la verificación de los resultados obtenidos mediante los métodos IGA y el código empleado para FEM en Matlab, se realizó la comparación con el software ANSYS (P.C., 1982). ANSYS es un programa informático de elementos finitos para la solución de análisis de ingeniería estructural y de transferencia de calor. Dado que este programa está basado en elementos finitos, y estos no capturan de buena forma los elementos curvos, se decide utilizar un elemento trapezoidal, tal como se aprecia en la Figura 4.18, el cual tiene las mismas dimensiones mostradas en la Tabla 4.3.

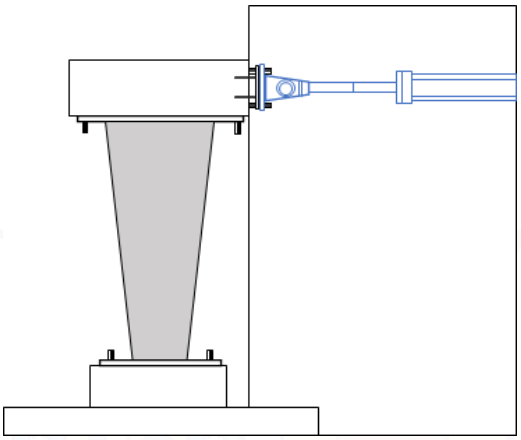


Figura 4.18: Esquema del elementos utilizado para la comparación.

En la Figura 4.19 se muestran los resultados obtenidos en la comparación, usando una malla de elementos de 1 cm de tamaño máximo en el programa ANSYS. Se aprecia claramente que la implementación tanto de IGA como de FEM convergen al mismo valor que ANSYS, sin embargo, nuevamente IGA converge más rápido que el código elaborado de FEM. Para el caso de FEM es importante menciona el grado de la función de interpolación que se utilizó. En esta ocasión se trabajo con una función cuadrática.

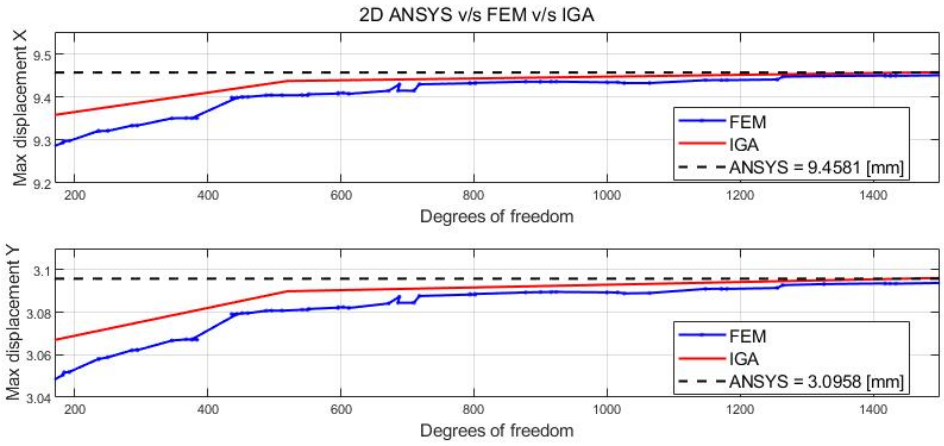


Figura 4.19: Comparación de la tasa de convergencia con método IGA y FEM v/s ANSYS.

Tasas de convergencia:

Para estudiar la tasa de convergencia con los métodos FEM e IGA se utilizarán los datos de la Figura 4.19, pero en esta ocasión se modificará a una escala logarítmica y en el eje Y se modificará acorde a la ecuación 4.1. Esto se aprecia en la Figura 4.20.

$$Y = \frac{|Desp. - Desp.ANSYS|}{Desp.ANSYS} \tag{4.1}$$

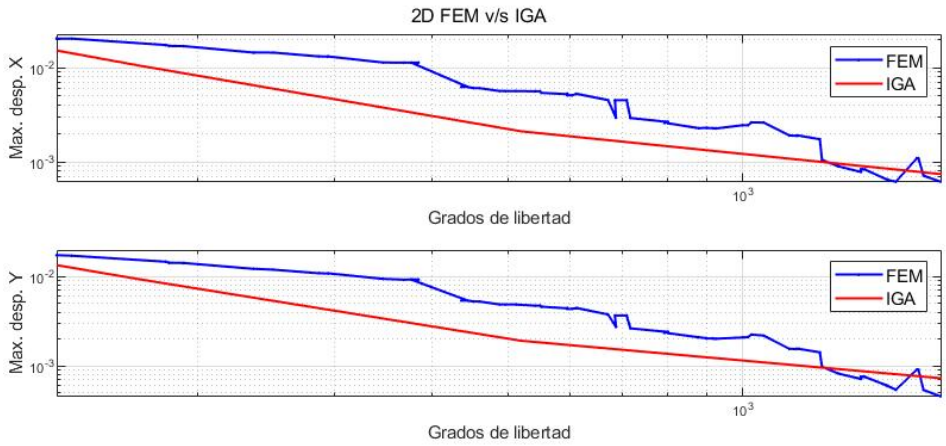


Figura 4.20: Comparación de la tasa de convergencia con método IGA y FEM para el máximo desplazamiento (en mm) en cada eje de análisis según la cantidad de grados de libertad considerados.

De la Figura 4.20 se puede apreciar que la pendiente, y la tasa de convergencia, es similar para ambas metodologías. Sin embargo el método IGA requiere menor cantidad de grados de libertad para alcanzar un resultado con un menor error respecto al esperado, lo que se traduce en una mayor precisión con tamaños de malla más gruesos.

Por otro lado, en la Tabla 4.4 se indican los valores a los que converge cada método. Estos valores de convergencia se obtienen con el desplazamiento máximo calculado con la malla más fina considerada con cada uno (6682 grados de libertad para FEM y 1800 para IGA). Cabe señalar que se obtienen valores que difieren a la centésima de milímetro, siendo IGA el método que entrega desplazamientos más cercanos al obtenido mediante ANSYS.

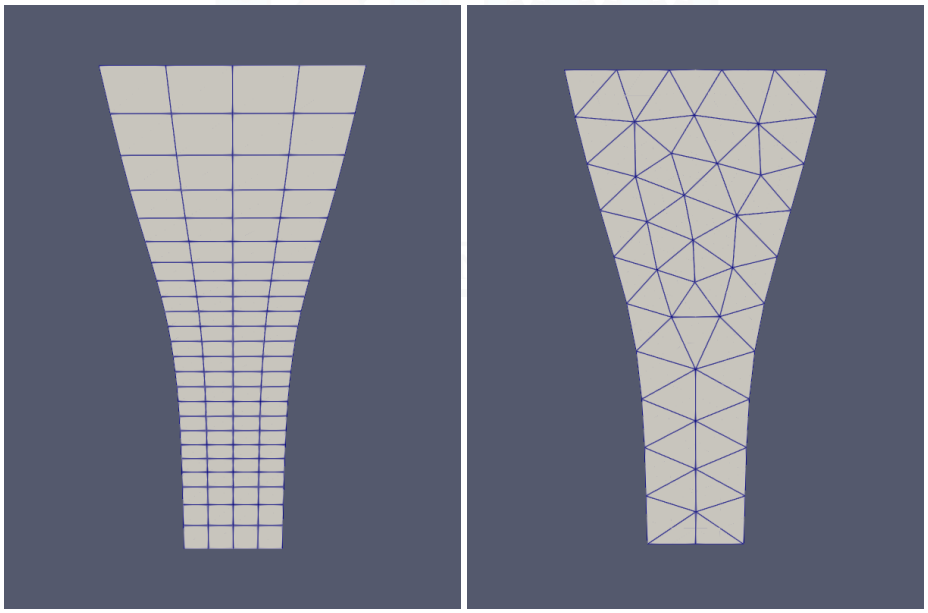
Método	IGA	FEM	ANSYS
Eje X	9.4651	9.4717	9.4581
Eje Y	3.0981	3.1001	3.0958

Tabla 4.4: Valores de convergencia de cada método, definidos como el valor de desplazamiento máximo obtenido (en milímetros) con la malla más fina computada para cada método.

vRTHS:

La cantidad de grados de libertad considerados para la simulación con el método IGA es de 312 y con FEM es de 316, con el mallado que se presenta en la Figura 4.21. Con este mallado, es posible obtener la rigidez equivalente del objeto que se utiliza para la vRTHS, que corresponde a 64.7 Ns^2/mm en el caso de FEM y 64.0 Ns^2/mm en el caso de IGA.

En esta sección se presentan los resultados obtenidos para cada método estudiado: FEM e IGA. Finalmente, se realiza una comparación de los resultados entre ambos.



(a) IGA. **(b)** FEM.

Figura 4.21: Mallado utilizado en cada método para vRTHS.

- **Método IGA:**

Los resultados de sincronización se encuentran en la Figura 4.22, el desempeño de la vRTHS respecto a la estructura de referencia se muestra en la Figura 4.23, en la Figura 4.24 se aprecia la histéresis de la cepa izquierda del puente, mientras que en la Figura 4.25 se obtienen los ticks perdidos durante la simulación.

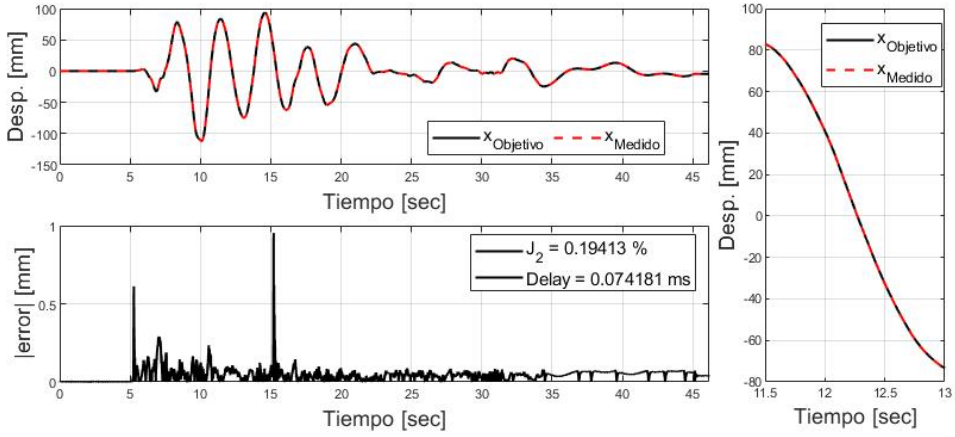


Figura 4.22: Sincronización de la vRTHS con IGA.

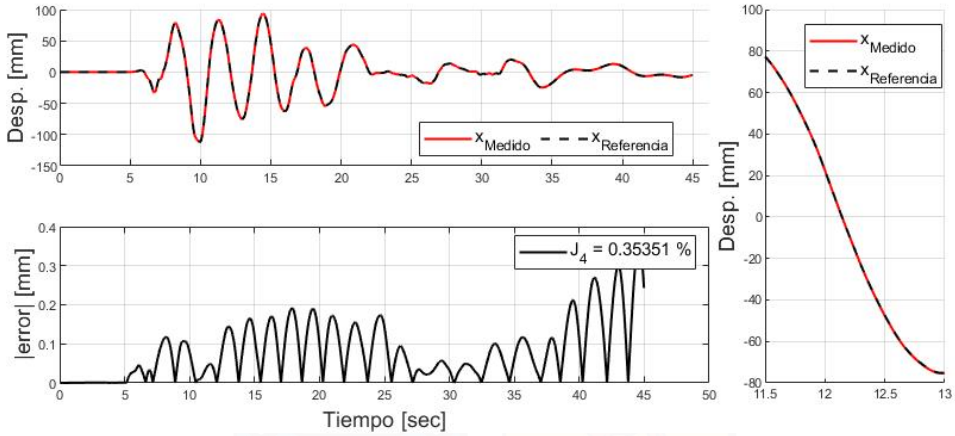


Figura 4.23: Desempeño de la vRTHS con IGA.

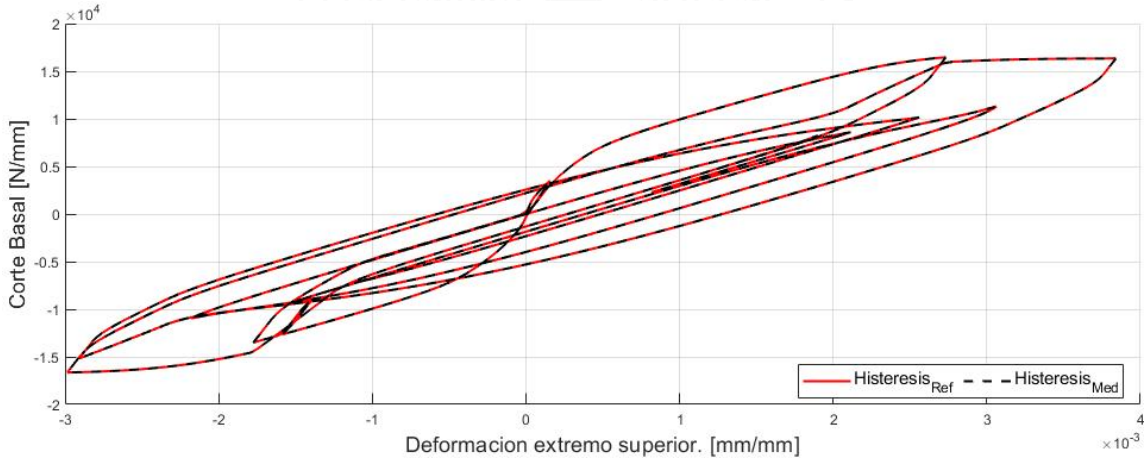


Figura 4.24: Histeresis cepa izquierda del puente con IGA.

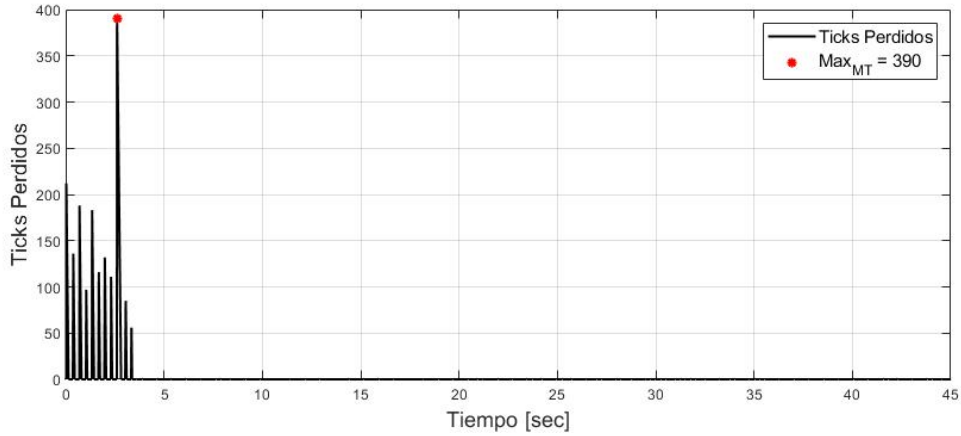


Figura 4.25: Ticks perdidos en la vRTHS con IGA.

- **Método FEM:**

Los resultados de sincronización se encuentran en la Figura 4.26, el desempeño de la vRTHS respecto a la estructura de referencia se muestra en la Figura 4.27, en la Figura 4.28 se aprecia la histeresis de la cepa izquierda del puente, mientras que en la Figura 4.29 se obtienen los ticks perdidos durante la simulación.

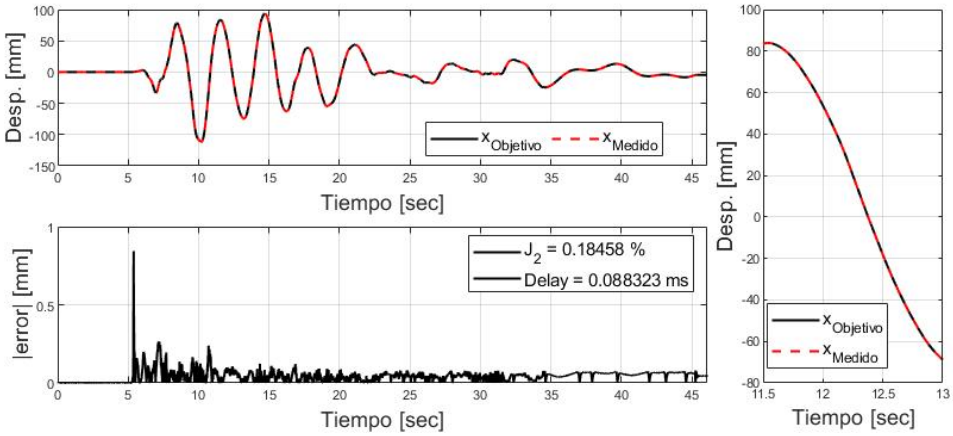


Figura 4.26: Sincronización de la vRTHS con FEM.

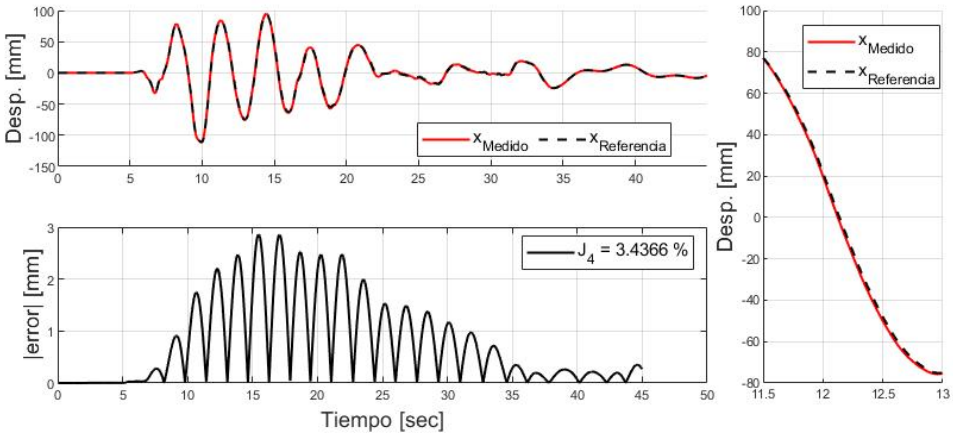


Figura 4.27: Desempeño de la vRTHS con FEM.

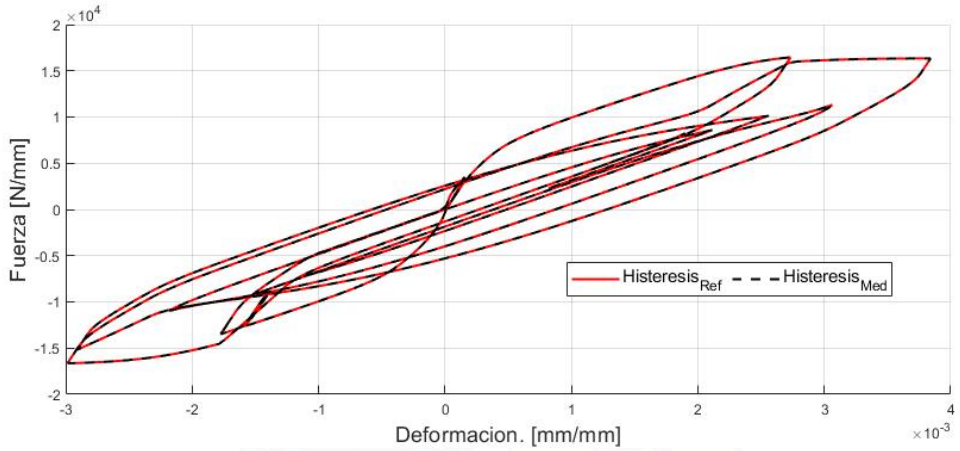


Figura 4.28: Histeresis cepa izquierda del puente con IGA.

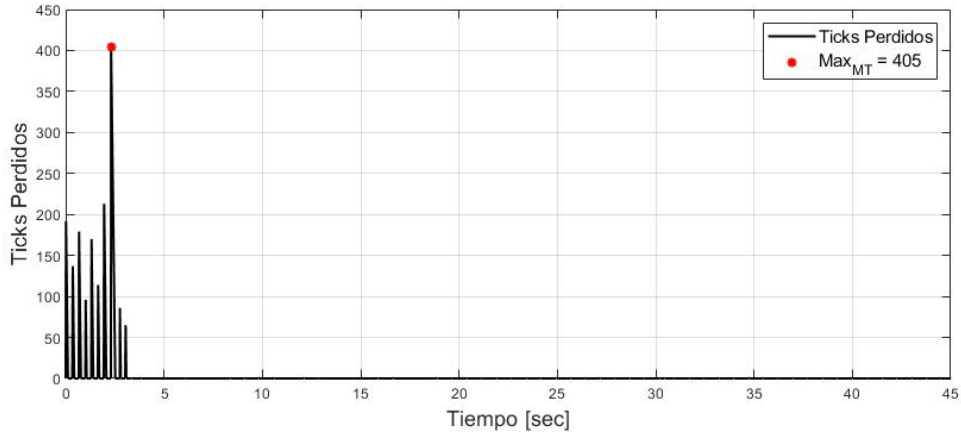


Figura 4.29: Ticks perdidos en la vRTHS con FEM.

- **Comparación:**

En la Tabla 4.5 se presenta un resumen de los resultados obtenidos con cada método. Por otro lado, en la Figura 4.30 se presenta una comparación de los desplazamientos obtenidos.

Método	IGA	FEM
J_2 %	0.194	0.185
J_4 %	0.354	3.4366
τ [ms]	0.074	0.088
Ticks Perdidos	390	405

Tabla 4.5: Comparación de los resultados de los indicadores de simulación con cada método.

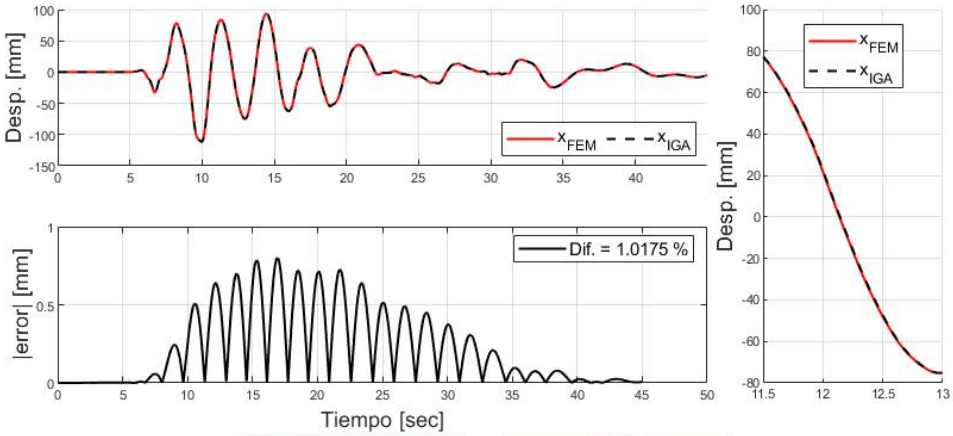
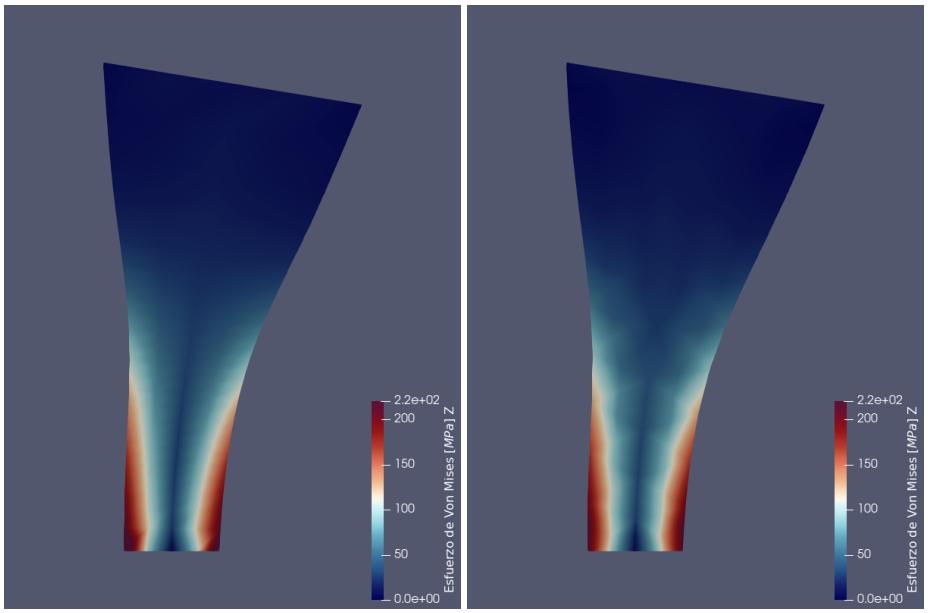


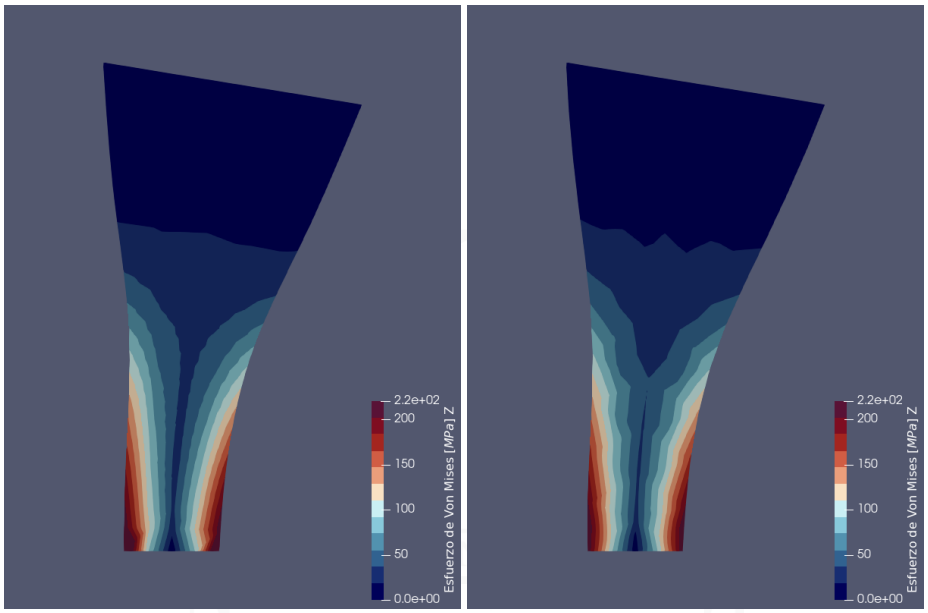
Figura 4.30: Comparación de los desplazamientos obtenidos mediante FEM e IGA en la vRTHS. La figura de la derecha realiza un acercamiento entre los 11.5 y 13 s.

A continuación, en la Figura 4.31 se muestra una comparación en cuanto a los esfuerzos de Von Misses. En la Figura 4.32 se hace más apreciable la diferencia entre la suavización de tensiones para ambos métodos. Importante mencionar que estos modelos son del tipo lineal.



(a) IGA. **(b)** FEM.

Figura 4.31: Comparación esfuerzo de Von Misses. Los niveles de deformación se encuentran amplificados por un factor de 5.



(a) IGA. **(b)** FEM.

Figura 4.32: Comparación esfuerzo de Von Misses. El rango de colores fue disminuido para mejorar la apreciación.

Es importante mencionar que debido a la diferencia en tamaño del mallado de ambos métodos no es directo el cálculo de alguna métrica. De igual forma podemos comparar las normas de ambas matrices. En este caso, utilizando la norma de Frobenius. La norma de Frobenius es una extensión de la norma euclidiana para $K^{n\times n}$ y proviene del producto interno de Frobenius en el espacio de todas las matrices. La norma de Frobenius de una matriz $m \times n$ (con m,$n \geq 2$) queda definida por:

$$||X||_F = \sqrt{\sum_{i=1}^{m}\sum_{i=1}^{m}|a_{ij}^2|}$$

Obtenemos los siguientes resultados:

- $||X||_{FEM} = 6{,}3856 \times 10^4$
- $||X||_{IGA} = 6{,}5154 \times 10^4$

Como es apreciable, las normas de las dos matrices son comparables. Otra comparación que podemos realizar, es respecto a los máximos y mínimos esfuerzos de Von Misses para la simulación. En la Figura 4.33 se aprecia una comparación entre estos; se realiza un zoom entre los 12 y 24 [s] dado que es el intervalo donde la diferencia es mas notoria.

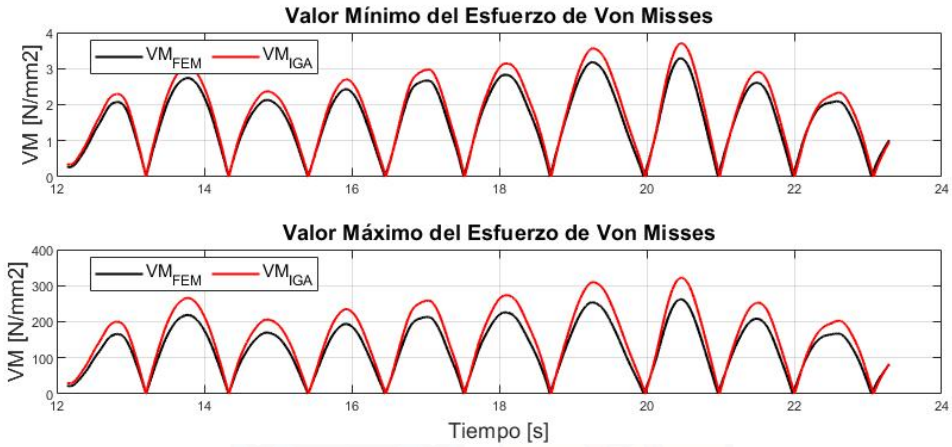


Figura 4.33: Comparación de los esfuerzos de Von Misses máximos y mínimos obtenidos mediante FEM e IGA en la vRTHS.

Para este mismo rango se obtiene la diferencia entre los valores máximos y mínimos entre ambas metodologías, tal como se aprecia en la Figura 4.34.

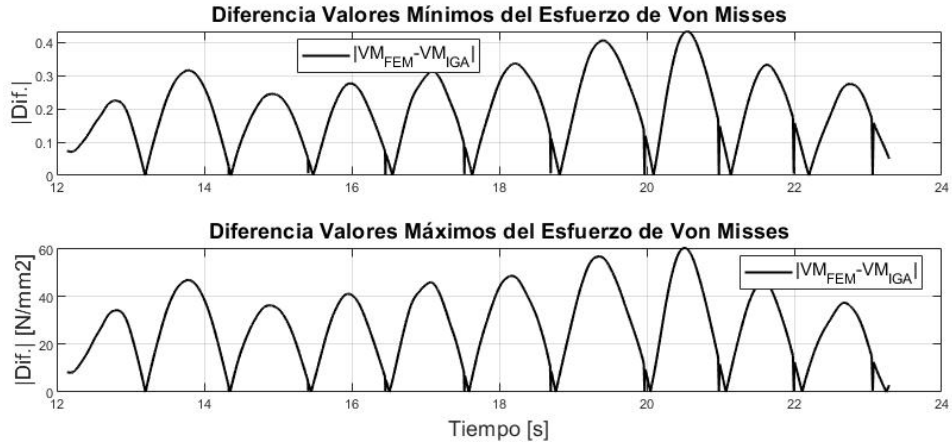


Figura 4.34: Diferencia entre los esfuerzos de Von Misses máximos y mínimos obtenidos mediante FEM e IGA en la vRTHS.

Si bien el modelamiento de la cepa es un proceso simple para cualquier persona familiarizada con programas de tipo CAD, el proceso de extracción de los datos de la geometría presenta un desafío, debido a que existe muy poca información actualmente del proceso de exportación de la geometría que describen las NURBS. Este procedimiento es esencial en el caso de que se quiera aprovechar las ventajas del uso de la resolución de problemas mediante IGA, puesto a que es capaz de representar estas geometrías de manera "exacta", a diferencia de FEM que realiza una aproximación polinomial para dicho propósito. Sin embargo, para geometrías predeterminadas, es posible su directa representación mediante un código en Matlab sin requerimiento de utilizar un programa CAD.

A través de las tasas de convergencia obtenidas para los métodos analizados IGA y FEM, se puede apreciar que IGA requiere de menor refinamiento para lograr resultados más precisos, lo que se traduce en un menor costo computacional. Sin embargo, actualmente no se cuenta con códigos optimizados para la resolución de estos problemas mediante IGA, por lo que requieren de mayor tiempo de cómputo que la metodología FEM.

Por otro lado, se puede apreciar del gráfico de la Figura 4.20 que el resultado de convergencia difiere levemente con ambas metodologías (al valor de la centésima de milímetro). Esto puede deberse a que existe alguna aproximación en cómo interpreta Matlab la geometría curva de este objeto. Cabe destacar que esta diferencia disminuye al utilizar una geometría recta como en el caso del modelo trapezoidal utilizado para la comprobación.

De igual forma, en la Figura 4.31 se observa una buena concordancia en cuanto a niveles de esfuerzos de Von Misses, llegando en ambos casos a un máximo aproximado de 220 $[N/mm^2]$, pero se aprecia nuevamente un mejor resultado para la metodología IGA, entregando valores mucho más suavizados (Figura 4.32) que su contraparte FEM. En ambos casos se aplicó una suavización de tensiones, pero dado el tamaño de elemento en FEM aún son apreciables los elementos Tri6 utilizados. Analizando ahora los esfuerzos máximos y mínimos en todos los instantes de tiempo, se observa una mayor diferencia entre las metodologías, llegando a apreciarse diferencias máximas de hasta 60 $[N/mm^2]$ (Figura 4.34).

La gran diferencia de ambas metodologías proviene de la interpretación que da cada una a la geometría del objeto de análisis, es decir, las funciones de forma. Esto implica en el caso de IGA el uso de un espacio dimensional extra para la resolución del problema de elasticidad. Sin embargo, el proceso de ensamblaje de las matrices de rigidez y de fuerzas es prácticamente el mismo para ambos procedimientos.

En este sentido, ambas metodologías de análisis entregan resultados muy similares en cuanto a los desplazamientos del espécimen (con un error NRMSE entre ambos métodos de un 1 %) en la vRTHS, lo que se explica con la diferencia de un 1 % de las rigideces equivalentes. En cuanto al desempeño de la vRTHS, no existe gran diferencia entre ambos métodos, ya que los indicadores de la simulación presentan variabilidad debido al ruido de los sensores. De igual forma, es apreciable un mayor error para la metodología FEM en lo que a la comparación con la referencia se refiere.

Queda demostrado que en la eventualidad de que se necesite la utilización de un método personalizable, como lo es IGA, la plataforma lo permite. Queda abierta la posibilidad de realizar este estudio con otras metodologías como lo son Mesh Free method, Discrete Elements methods, etc.

4.6. Implementación del sistema en tiempo real - Plataforma vRTHS Cliente-Servidor

La implementación de vRTHS a utlizar para los casos estudio 4 y 5 es el mostrado en la sección 3.3.3 e ilustrado en la Figura 3.12. Tal como se mencionó previamente, las instrucciones detalladas de instalación y ejecución del software se proporcionan en el repositorio complementario de GitHub Mera y Fermandois (2021).

4.7. Caso Estudio 4, Modelo 2D

4.7.1. Subestructuras Numérica y Experimental

Uno de los pasos cruciales al realizar una simulación híbrida es la elección correcta de la subestructuración (Dermitzakis y Mahin, 1985). En el contexto de RTHS, en lugar de resolver la ecuación de movimiento del dominio completo, este proceso de subestructuración se puede aplicar para subdividir el dominio en subdominios más pequeños de modo que el orden de los sistemas estructurales grandes y complejos se reduzca para la eficiencia computacional. Cada subdominio se puede resolver de forma independiente, siempre que el acoplamiento entre componentes se aplique en sus interfaces (De Klerk et al., 2008).

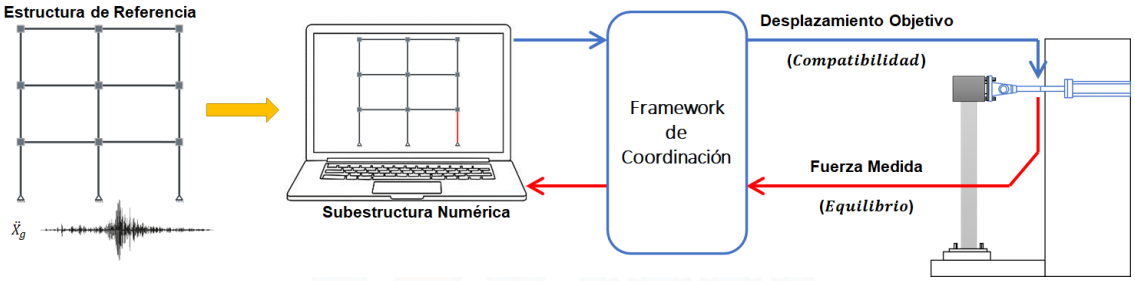


Figura 4.35: Subestructuras y componentes de una simulación híbrida.

En este estudio se elige como estructura de referencia un marco plano de tres pisos con dos vanos, con un total de 30 grados de libertad (9 laterales, 9 verticales y 12 rotacionales). La estructura de referencia se divide en la subestructura numérica (NS) y la subestructura experimental (ES), como se muestra en la Figura 4.35. Una columna del primer piso se toma como ES, mientras que el resto se modela como NS. Dado que está diseñado para funcionar solo con un actuador uniaxial, los grados de libertad límite se establecen para que sean solo desplazamientos horizontales.

4.7.2. Subestructura Experimental

Para la subestructura se trabajan cinco casos diferentes (un caso lineal, cuatro casos no lineales). Las no linealidades se modelan de acuerdo con lo establecido por Wang et al. (2001). Los modelos no lineales se basan en el modelo de Bouc-Wen con algunas modificaciones para incluir variaciones de resistencia y rigidez.

La fuerza experimental se calcula con la siguiente expresión:

$$f_e = r_e + c_e\dot{x}_m + m_e\ddot{x}_m \tag{4.2}$$

donde c_e y m_e son el amortiguamiento experimental y la masa respectivamente, x_m es el desplazamiento medido y r_e es una fuerza restitutiva no lineal descrita por el modelo de Bouc-Wen modificado:

$$\dot{r}_e = R_k k_e \left(1 - \left|\frac{r_e}{f_y}\right|^N (\eta_1 sign(r_e\dot{x}_m) + \eta_2)\right)\dot{x}_m \tag{4.3}$$

donde R_k es un factor de degradación de la rigidez, k_e es la rigidez elástica inicial y f_y es la fuerza de fluencia. N, eta_1 y eta_2 son parámetros que controlan la forma de histéresis. El factor de degradación de la rigidez y la degradación de la fuerza de fluencia se calculan con los modelos de degradación de Mostaghel.

$$R_k = e^{-\alpha H} \tag{4.4}$$

$$f_y = f_{y0}(1 + \beta H)^{-1} \tag{4.5}$$

donde $alpha$ es un parámetro que controla la degradación de la rigidez, $beta$ controla la degradación de la resistencia y f_{y0} es la fuerza de fluencia inicial. Finalmente, H es la energía histerética disipada que se puede calcular en forma incremental como:

$$\Delta H = \left(\frac{r_e + (r_e + \Delta r_e)}{2}\right)\left(\Delta x_m - \frac{\Delta r_e}{R_k k_e}\right) \tag{4.6}$$

Además, para considerar un efecto de endurecimiento, se agrega un resorte cúbico en paralelo con una rigidez cúbica constante k_h:

$$f_e = k_h x_m^3 + r_e + c_e \dot{x}_m + m_e \ddot{x}_e \tag{4.7}$$

Se definen cuatro subestructuras experimentales con los parámetros presentados en la Tabla 4.6

Tabla 4.6: Parametros para los modelos no lineales.

Caso no lineal	$f_{y0}[Kips]$	N	η_1	η_2	α	β	$k_h[Kips/in^3]$
1	0.058	0.1	0.1	0.9	0	0	0.005
2	0.058	0.5	0.5	0.5	0	0	0.005
3	0.058	0.5	0.5	0.5	0.5	0.5	0.005
4	0.29	1.5	0.25	0.75	0.2	0.2	0.005

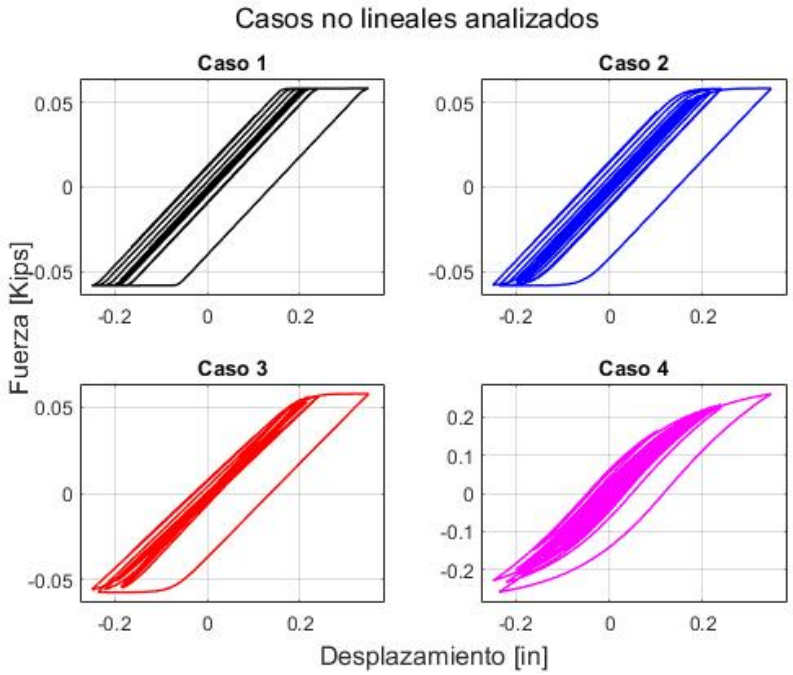


Figura 4.36: Relación fuerza vs desplazamiento para diferentes modelos no lineales.

Los casos analizados que se pueden ver en la Figura 4.36 consisten en un bilineal, bilineal suavizado cerca de la fluencia, bilineal suavizado con degradación de la rigidez y no lineal con endurecimiento post fluencia.

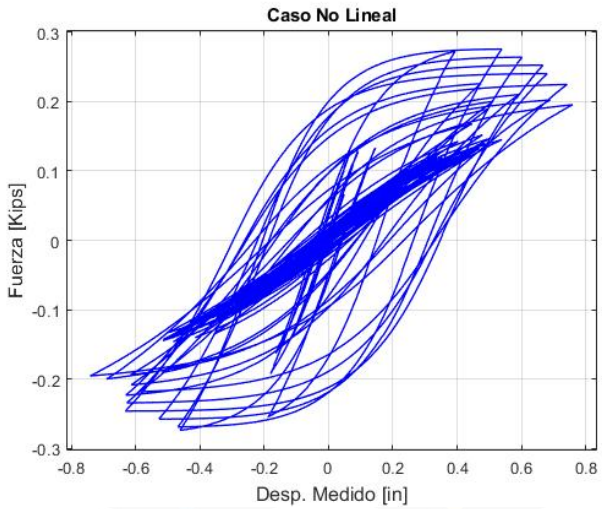


Figura 4.37: Relación fuerza vs desplazamiento de la subestructura experimental para modelo numérico no lineal.

Para demostrar la flexibilidad, modularidad y aplicabilidad del marco de coordinación para una subestructura numérica compleja y con comportamiento realista, también se resuelve para una subestructura numérica no lineal. En este caso, se considera una sección de fibra de hormigón armado considerando elementos tipo *quad* para hormigón y *straight* para el acero de refuerzo. Además, se utilizaron vigas y columnas del tipo *Displacement Beam-Column*. La consideración de los parámetros del caso no lineal 4 da como resultado la histéresis que se muestra en la Figura 4.37.

4.7.3. Resultados Subestructuración

Como se mencionó previamente en la subsección 3.3.3, se seleccionaron tasas de muestreo de $2/100$ para la subestructura numérica y de $1/4000$ para la subestructura experimental. Para la elección de estas tasas se llevaron a cabo una serie de simulaciones a diferentes tasas de muestreo con la finalidad de evaluar y seleccionar la diferencia de tasas óptima. En la Figura 4.38 se muestra la implementación en Simulink de la simulación para resolver el problema de subestructuración.

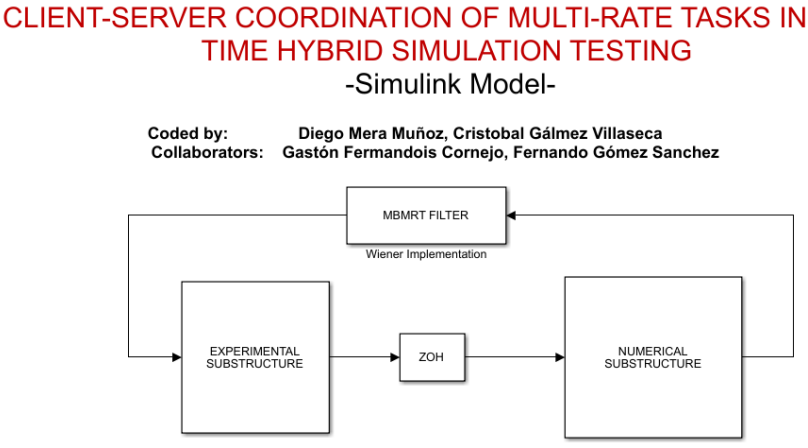


Figura 4.38: Implementación en Simulink de la subestructuración dentro de una RTHS virtual

La tasa de muestreo de la subestructura numérica fue definida como Δt_n y fue fijada a la tasa del registro sísmico utilizado. El registro de El Centro de 1940 fue seleccionado, el cuál posee una tasa de

muestreo de 2/100. La tasa de muestreo de la subestructura experimental se denominó Δt_e y se propuso evaluar tres casos distintos. Estos son Δt_e igual 1/2000, 1/4000 y 1/8000, es decir, diferencias de tasas entre la subestructura numérica y la experimental $\left(\frac{\Delta t_n}{\Delta t_e}\right)$ de 40, 80 y 160.

A continuación, en la Figura 4.39 se muestra el resultado para una de las simulaciones realizadas. En este caso se seleccionó una simulación local, con una diferencia de tasa de 80 y con parámetros del filtro DDMRT invariantes o fijos. En esta figura se aprecia la comparación entre el desplazamiento numérico que ingresa el filtro DDMRT y el desplazamiento que sale de este. Se observa un error entre ambas señales de un 2,954[%] y un retraso en las señales de 0,865[ms]

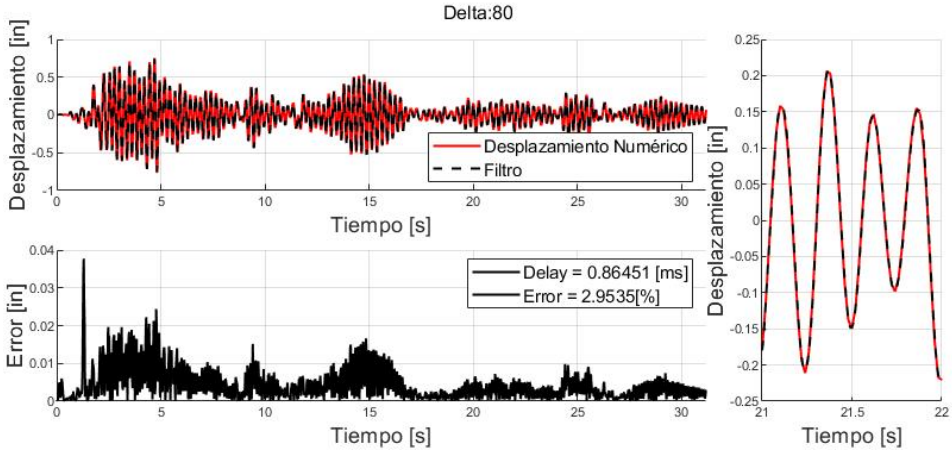


Figura 4.39: Desplazamiento Objetivo v/s Medido e indicador J_2

En la Tabla 4.7 se muestra el resumen de todas las simulaciones realizadas. Se incluyen la simulaciones tanto locales como en modo externo ejecutándose en la Raspberry Pi 4 Modelo B, junto con parámetros del filtro DDMRT tanto fijos como adaptivos. Es claramente apreciable, que a medida aumenta la diferencia de tasas, aumenta el error entre ambas señales. Este resultado era el esperado dado que requiere un mayor tiempo, cantidad de procesamiento y cálculos, para generar una señal a menor tasa. Por consiguiente, induce un retraso entre estas. Finalmente, se selecciona una diferencia de tasas de 80, tanto para parámetros fijos como adaptivos, dado que en ningún caso el error supero el 5[%], y para el autor, es un indicador de un buen resultado.

Tabla 4.7: Tabla resumen subestructuración a diferentes tasas de muestreo

	Parámetros Fijos				Parámetros adaptivos			
	Local		Externa		Local		Externa	
$\Delta t_n/\Delta t_e$	J_2 [%]	$delay$ [ms]	J_2 [%]	$delay$ [ms]	J_2 [%]	$delay$ [ms]	J_2 [%]	$delay$ [ms]
40	1.713	0.197	0.938	0.308	0.833	0.193	0.919	0.350
80	2.954	0.865	3.775	1.188	3.046	0.672	3.800	1.161
160	17.332	1.548	11.688	2.091	11.970	1.536	11.634	2.000

4.7.4. Resultados Simulaciones Locales

Dada la gran extensión que abarcaría incluir todas las figuras, se decide incluir solo las de una simulación. Las siguientes figuras están considerando una subestructura numérica lineal elástica. En este caso, se selecciona el caso 4 no lineal con parámetros adaptivos. Las simulaciones locales se ejecutan en un notebook MSI GF75 modelo Thin 9RCX. La figura 4.40 nos muestra la comparación entre el desplazamiento objetivo (x_t) y el desplazamiento medido (x_m), junto con su indicador correspondiente J_2. Los resultados muestran un error NRMSE de 1,34 % junto con un retraso entre señales de 0,04[ms]. Estos resultados nos

permiten validar la simulación en términos de compensación de retardo entre las señales producidas por la dinámica y el retardo del actuador.

La figura 4.41 nos muestra la comparación entre la señal medida en nuestro vRTHS versus el desplazamiento de referencia de la estructura. Se ve un error NRMSE de aproximadamente 1,672 %, resultado que indica una resolución correcta del vRTHS.

Finalmente, la Figura 4.42 nos muestra la comparación entre la histéresis de la subestructura experimental en nuestro vRTHS versus la de referencia.

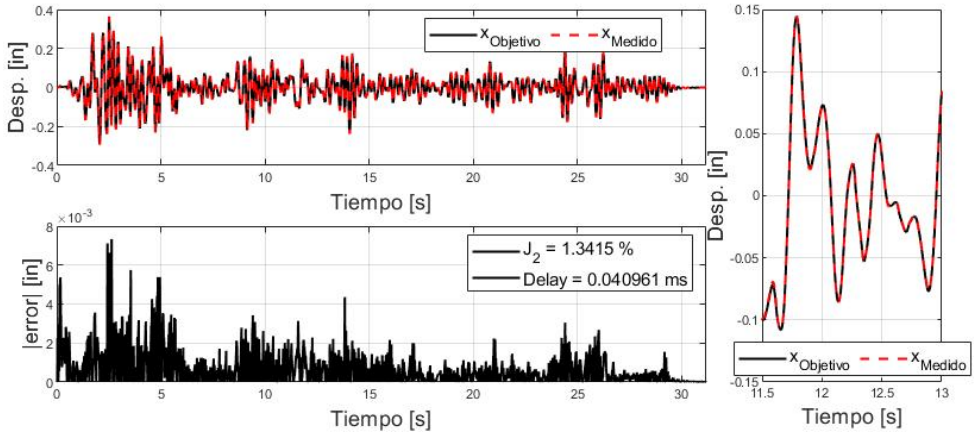


Figura 4.40: Desplazamiento Objetivo v/s Medido e indicador J_2

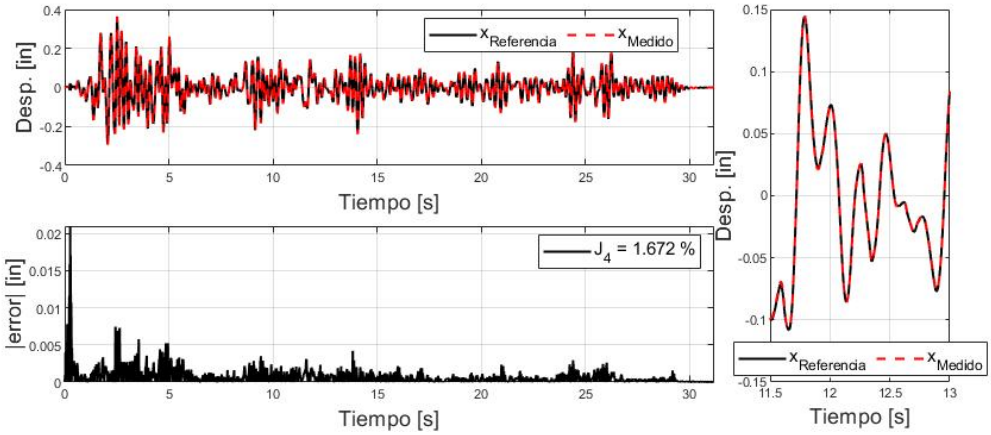


Figura 4.41: Desplazamiento de Referencia v/s Medido e indicador J_4.

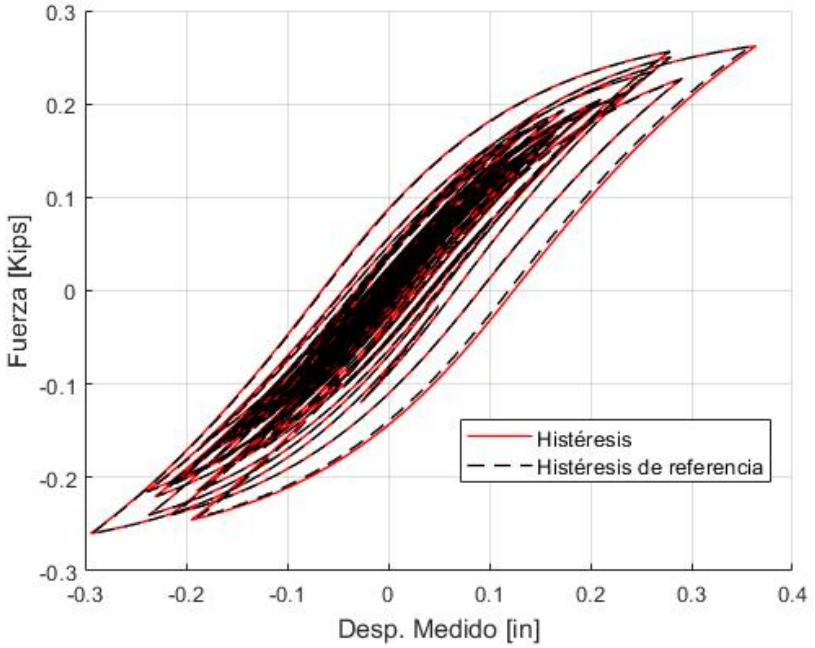


Figura 4.42: Histéresis de la Subestructura experimental (Referencia v/s Medido)

Las tablas 4.8 y 4.9 muestra los resultados obtenidos con el filtro con parámetros fijos y variables respectivamente, para los 5 casos analizados. Se muestran los errores J_2, J_4 (Error de desplazamiento y fuerza), junto con el retraso entre el desplazamiento objetivo (x_t) y el desplazamiento medido (x_m). De todos los casos analizados, el mayor error alcanzó un valor de aproximadamente 2,6 %. Analizando los ticks perdidos, se puede apreciar que no se superó el límite establecido de 500, es decir, la simulación se realiza sin ningún problema. Esto nos muestra que la simulación se realizó correctamente y con precisión. Además, nos muestra la aplicabilidad y modularidad del marco de comunicación generado.

Tabla 4.8: Tabla resumen de simulaciones locales para subestructura numérica lineal parámetros fijos

Caso	J_2 [%]	$J_{4,Desp}$ [%]	$J_{4,Fuerza}$ [%]	$delay$ [ms]	MT
Lineal	1.3136	1.4018	1.3486	0.040235	43
Caso 1	1.3397	1.3878	2.1977	0.040909	22
Caso 2	1.3397	1.5052	1.8432	0.040911	32
Caso 3	1.3396	1.5047	2.5962	0.040869	29
Caso 4	1.2884	1.3864	1.5713	0.041286	20

Tabla 4.9: Tabla resumen de simulaciones locales para subestructura numérica lineal parámetros adaptivos

Caso	J_2 [%]	$J_{4,Desp}$ [%]	$J_{4,Fuerza}$ [%]	$delay$ [ms]	MT
Lineal	1.3136	1.4485	1.3971	0.040244	23
Caso 1	1.3142	1.2730	2.1304	0.040278	27
Caso 2	1.3143	1.4237	1.7772	0.040279	40
Caso 3	1.3142	1.4231	2.5266	0.040237	32
Caso 4	1.3415	1.6720	1.9886	0.040961	23

Considerando ahora una subestructura numérica de tipo no lineal, anteriormente explicada, se obtienen los resultados que se muestran a continuación. Adicionalmente, se realizó un análisis del efecto de considerar

otro tipo de integrador en OpenSeesPy, comparando el desempeño del método AlphaOS (Combescure y Pegon, 1997) versus Newmark (Newmark, 1959) con un número fijo de iteraciones. En esta ocasión 3 iteraciones son seleccionadas.

Las tablas 4.10 y 4.11 muestran los resultados de los 3 casos analizados. Se muestran los errores J_2 (Error de seguimiento), J_4 (Error de desplazamiento y fuerza), junto con el retraso entre el desplazamiento objetivo (x_t) y el desplazamiento medido (x_m). Se aprecia que en todos los casos se obtienen buenos y prometedores resultados. Algo importante a tener en cuenta en estas tablas, es la diferencia considerable en el retraso entre los diferentes tipos de integradores. Como era de esperar, el método de Newmark, al ser implícito, induce un retraso mayor que el método explícito de AlphaOS. A pesar de esta diferencia, las simulaciones son estable y se logran buenos resultados, permitiendo el uso de ambos métodos de integración.

Tabla 4.10: Tabla resumen de simulaciones locales para subestructura numérica no lineal parámetros fijos.

Caso	Integrador	J_2 [%]	$J_{4,Desp}$ [%]	$J_{4,Fuerza}$ [%]	$delay$ [ms]	MT
Lineal	Alpha OS (Explícito)	2.4821	2.6022	0.6363	0.05849	56
No Lineal	Alpha OS (Explícito)	2.3612	2.4409	6.7456	0.05203	18
No Lineal	Newmark (Implícito)	2.6025	4.2585	4.0738	0.14262	30
No Lineal	Newmark (Explícito)	1.0116	3.0539	5.0528	0.01370	29

Tabla 4.11: Tabla resumen de simulaciones locales para subestructura numérica no lineal parámetros adaptivos.

Caso	Integrador	J_2 [%]	$J_{4,Desp}$ [%]	$J_{4,Fuerza}$ [%]	$delay$ [ms]	MT
Lineal	Alpha OS (Explícito)	2.4869	2.6069	0.63635	0.05867	52
No Lineal	Alpha OS (Explícito)	2.0237	2.0293	2.8414	0.04705	54
No Lineal	Newmark (Implícito)	2.6112	2.7799	0.6476	0.14340	32
No Lineal	Newmark (Explícito)	1.0154	1.3081	3.7039	0.01385	16

4.7.5. Resultados Simulaciones Modo Externo

Al igual que en el caso de las simulaciones locales, se utiliza un notebook MSI modelo GF75 Thin 9RCX, pero esta vez solo esta encargado de ejecutar la subestructura numérica que se ejecuta en OpenSeesPy. Adicionalmente, se utiliza una Raspberry Pi 4 Modelo B como hardware externo dedicado, en el que se ejecuta el modelo de Simulink donde se implementa el sistema de control. Estos dos sistemas están conectados a través de una red LAN directamente mediante cables Ethernet, como se muestra en la Figura 4.43.

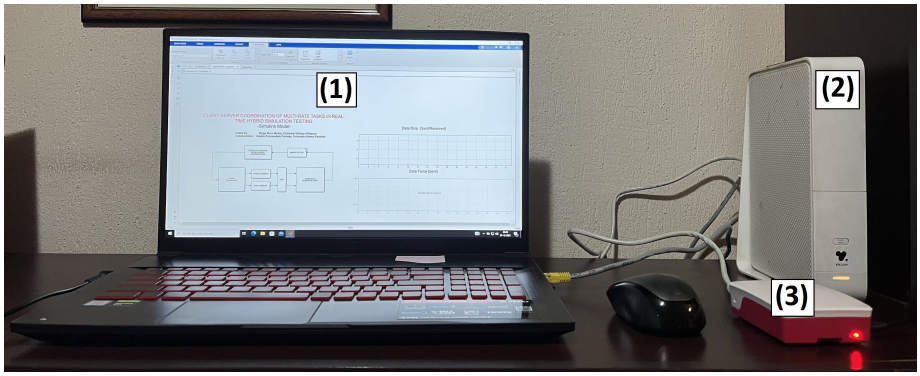


Figura 4.43: Dispositivos de hardware utilizados en el framework de comunicación propuesto. **(1):** PC host (subestructura numérica), **(2):** Enrutador (red LAN) y **(3):** Hardware externo (subestructura experimental)

En esta ocasión, para las figuras en modo externo que se muestran a continuación, se selecciona el caso con subestructuras, tanto numérica como experimental, no lineales, con parámetros fijos para el filtro

DDMRT. La figura 4.44 nos muestra la comparación entre el desplazamiento objetivo (x_t) y el desplazamiento medido (x_m), junto con su indicador correspondiente J_2. Los resultados muestran un error NRMSE de 0,53 % junto con un retraso entre señales de 0,008[ms]. Estos resultados nos permiten validar la simulación externa en términos de compensación de retardo entre las señales producidas por la dinámica y el retardo del actuador. Adicionalmente, se valida el framework de comunicación propuesto para simulaciones que intercomunican procesos ejecutados en diferentes máquinas. Este es un indicador de su aplicabilidad al laboratorio, donde se trabaja con un dSpace como hardware externo dedicado.

La figura 4.45 nos muestra la comparación entre la señal medida en nuestra vRTHS externa versus el desplazamiento de referencia de la estructura. Se ve un error NRMSE de aproximadamente 1,22 %, resultado que indica una resolución correcta y precisa de la vRTHS.

Finalmente, la Figura 4.46 nos muestra la comparación entre la histéresis de la subestructura experimental en nuestra vRTHS externa versus la de referencia.

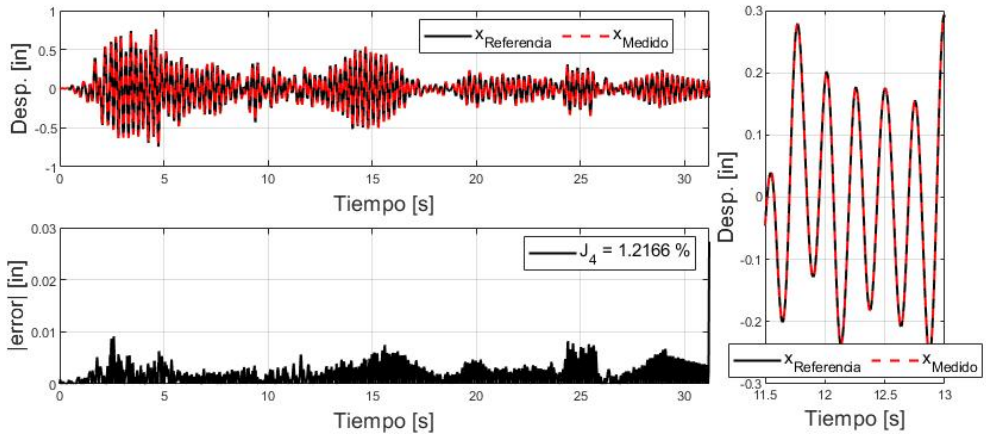


Figura 4.44: Desplazamiento Objetivo v/s Medido e indicador J_2

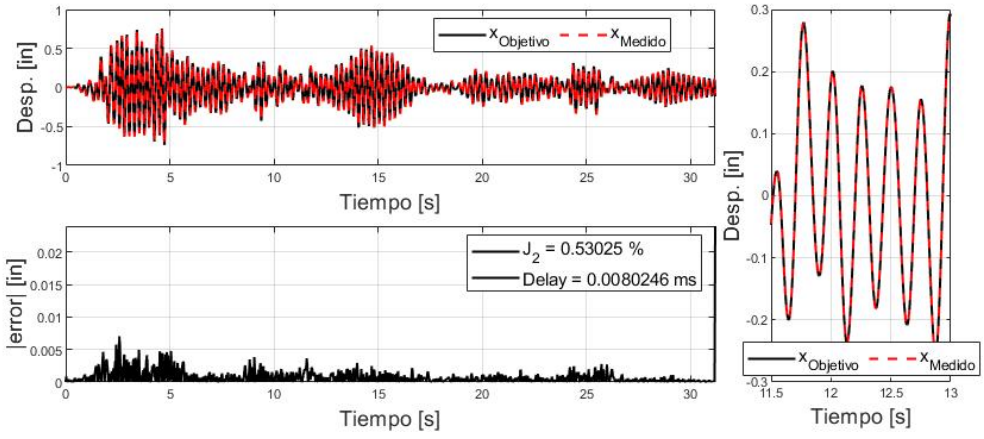


Figura 4.45: Desplazamiento de Referencia v/s Medido e indicador J_4.

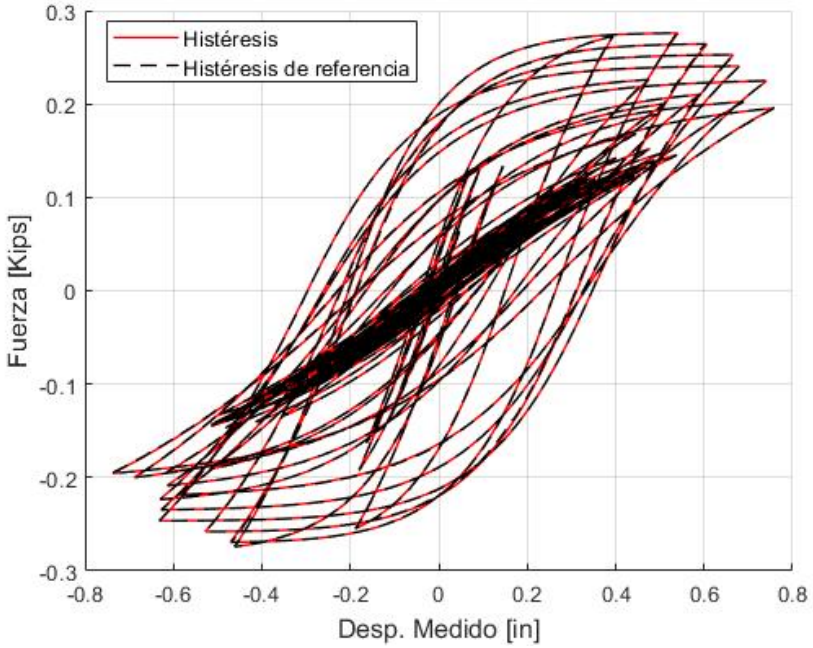


Figura 4.46: Histéresis de la Subestructura experimental (Referencia v/s Medido)

Al igual que las simulaciones locales, los resultados de los 5 casos de estudio, considerando una subestructura numérica lineal, se resumen en las tablas 4.12 y 4.13 donde se muestran los resultados obtenidos con el filtro con parámetros fijos y variables respectivamente. Se aprecian los errores J_2 (Error de seguimiento), J_4 (Error de desplazamiento y fuerza), junto con el retraso entre el desplazamiento objetivo (x_t) y el desplazamiento medido (x_m). De todos los casos analizados, el mayor error alcanzó un valor de aproximadamente 2,43 %. Esto nos muestra que la simulación se realizó correctamente y con precisión. Además, nos muestra la aplicabilidad y modularidad del marco de comunicación generado.

Se nota que para las simulaciones externas, tanto los errores como el retardo entre señales disminuyen, esto proviene de una limitación existente en las simulaciones locales, específicamente, en cuanto al uso del bloque de sincronización en tiempo real y su capacidad para perder ticks, es decir, puntos de información. De todos modos, al ser una cantidad tan pequeña la perdida en esta ocasión, este efecto no es altamente apreciable.

Tabla 4.12: Tabla resumen para simulaciones externas de subestructura numérica lineal parámetros fijos

Caso	J_2 [%]	$J_{4,Desp}$ [%]	$J_{4,Fuerza}$ [%]	$delay$ [ms]
Lineal	1.2836	1.4476	1.3810	0.03994
Caso 1	1.2777	1.3811	2.4245	0.03972
Caso 2	1.2947	1.5303	2.0647	0.04045
Caso 3	1.2879	1.5215	2.4226	0.04015
Caso 4	1.2824	1.4716	1.5694	0.03983

Tabla 4.13: Tabla resumen para simulaciones externas de subestructura numérica lineal parámetros adaptivos

Caso	J_2 [%]	$J_{4,Desp}$ [%]	$J_{4,Fuerza}$ [%]	$delay$ [ms]
Lineal	1.2772	1.4857	1.4210	0.03951
Caso 1	1.2826	1.3967	2.4320	0.04002
Caso 2	1.2787	1.5167	2.0540	0.03982
Caso 3	1.2786	1.5133	2.4142	0.03955
Caso 4	1.2879	1.7298	1.9684	0.04029

Asimismo, las tablas 4.14 y 4.15 resumen los resultados obtenidos considerando una subestructura numérica no lineal para parámetros del filtro DDMRT fijos y adaptivos, respectivamente. Se ven buenos resultados para todas las simulaciones. Una vez más, hay un aumento en el retraso debido al uso de los métodos de integración de Newmark y Newmark Explícito, ya sea implícito o explícito. Adicionalmente, se observa que es el que proporciona los peores indicadores, especialmente en lo que respecta a la fuerza restauradora de la subestructura experimental.

Tabla 4.14: Tabla resumen para simulaciones externas de subestructura numérica no lineal con parámetros fijos.

Caso	Integrador	J_2 [%]	$J_{4,Disp}$ [%]	$J_{4,Force}$ [%]	Delay [ms]
Lineal	Alpha OS (Explícito)	0.7444	1.4546	1.3101	0.00876
No Lineal	Alpha OS (Explícito)	0.5303	1.2166	1.4549	0.00803
No Lineal	Newmark (Implícito)	2.2678	2.3810	1.6416	0.09709
No Lineal	Newmark (Explícito)	0.9229	1.1590	3.7399	0.01040

Tabla 4.15: Tabla resumen para simulaciones externas de subestructura numérica no lineal con parámetros adaptivos.

Caso	Integrador	J_2 [%]	$J_{4,Disp}$ [%]	$J_{4,Force}$ [%]	Delay [ms]
Lineal	Alpha OS (Explícito)	1.6973	1.7464	1.5521	0.02647
No Lineal	Alpha OS (Explícito)	0.5317	0.6949	1.2332	0.00799
No Lineal	Newmark (Implícito)	1.3648	1.3710	3.7066	0.10145
No Lineal	Newmark (Explícito)	0.9265	0.9208	3.7158	0.01053

4.8. Caso Estudio 5, Modelo 3D - OpenSeesPy

4.8.1. Subestructuras Numérica y Experimental

Para probar la flexibilidad y aplicabilidad del marco de coordinación propuesto, se desarrolla una extensión del problema previamente resuelto en la sección 4.7. En esta ocasión, el mismo modelo se amplía a tres dimensiones. Este conserva las propiedades, tanto de materiales, como de dimensiones del problema original. Al igual que para el caso bidimensional con subestructura numérica no lineal, el caso seleccionado para la subestructura experimental es el caso 4 mostrado en la sección 4.7.2.

La idea del modelo tridimensional es producir un aumento significativo en el costo computacional del modelo, generando "estrés" en la velocidad de integración necesaria para ser considerado RTHS. Con esta finalidad, 3 modelos distintos para el caso 3D son generados, los cuáles son expuestos en la Figura 4.47. El primero considera un único elemento finito para representar tanto las vigas como las columnas, el segundo considera 2 elementos, es decir, un nodo intermedio y el tercer modelo considera 3 elementos, es decir, 2 nodos intermedios. En el caso bidimensional original, había 30 grados de libertad, en los casos tridimensionales, este número sufre un aumento significativo, alcanzando los 162, 498 y 906 grados de libertad respectivamente.

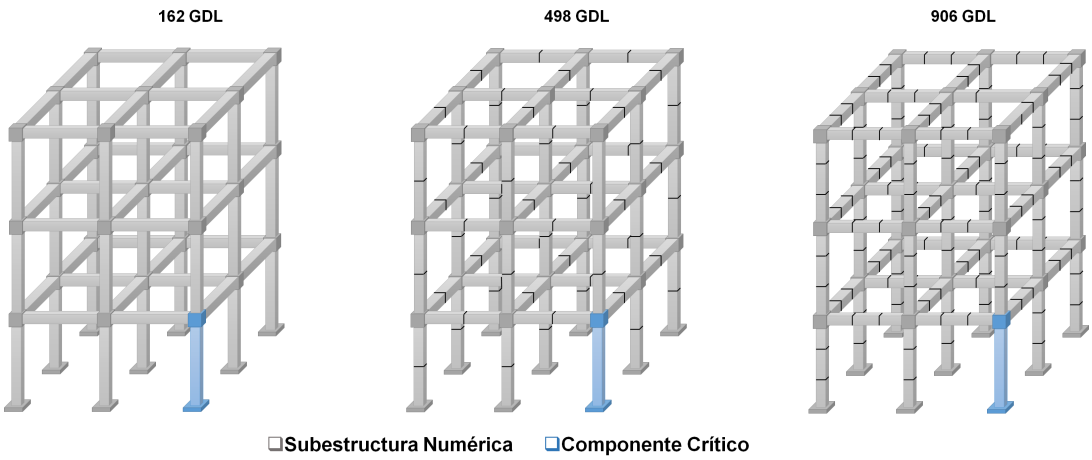


Figura 4.47: Subestructuras y componentes de una simulación híbrida.

Como primer paso, se realiza una comparación en cuanto a la respuesta para los modelos previamente mostrados. Para esto, se considera como punto de comparación el modelo con 908 GDL.

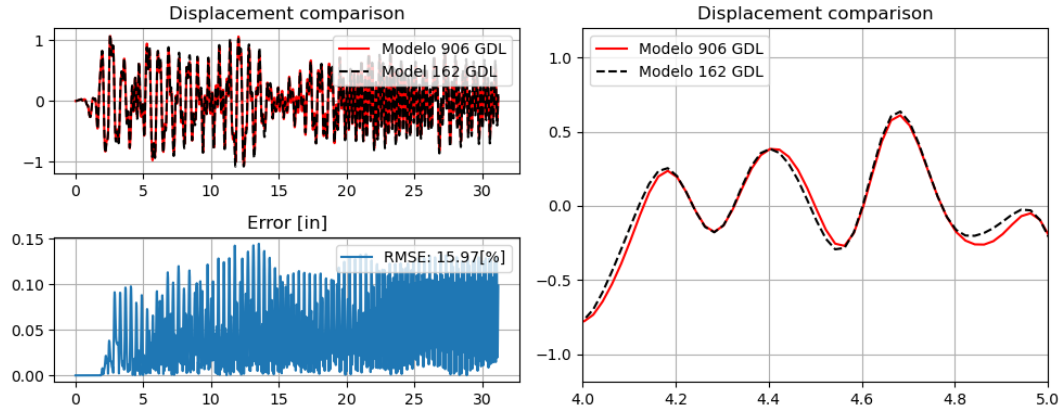


Figura 4.48: Comparación Modelos de 192 GDL y 906 GDL.

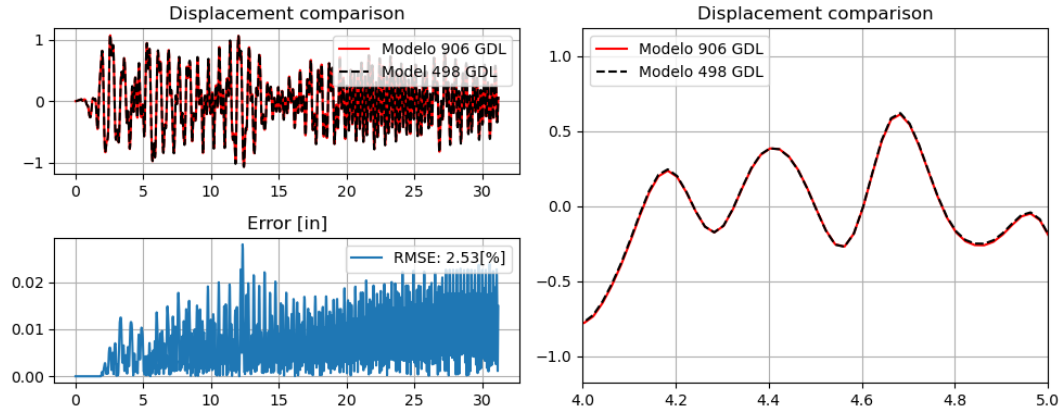
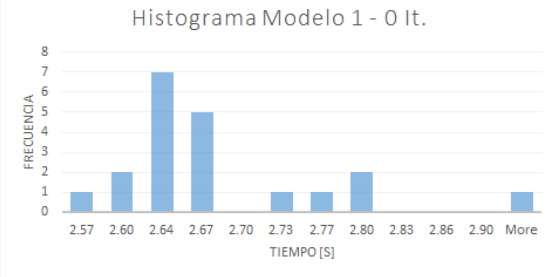


Figura 4.49: Comparación Modelos de 498 GDL y 906 GDL.

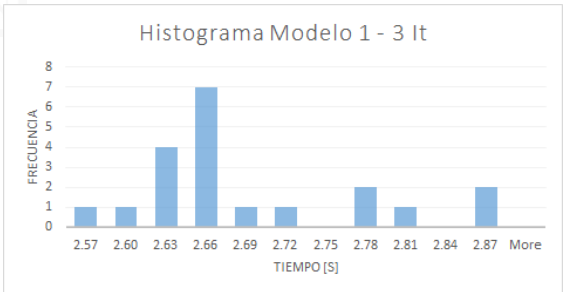
En la Figura 4.48 se observa una diferencia importante (cercana al 16 %), al comparar las respuestas del modelo de 162 GDL con el de 906 GDL. Esta diferencia se reduce a cerca de un 2.5 % cuando comparamos los modelos de 498 GDL y el de 906 GDL, tal como se aprecia en la Figura 4.49.

Adicionalmente, y previo a la resolución de la vRTHS se lleva a cabo un estudio paramétrico de estos casos. Dado el considerable aumento de GDL, no se puede asegurar que la resolución del modelo se lleve a cabo en tiempo real, por lo mismo, se estudia el tiempo de resolución de cada uno de estos modelos. Se resuelve el modelo numérico utilizando el integrador de Newmark para 0, 3 y 10 iteraciones. Para cada caso se resuelve 20 veces y se obtienen los siguientes histogramas del tiempo de simulación. Estos histogramas fueron generados para caso de forma independiente y consideran 10 intervalos de tiempo de simulación. La idea detrás de esto es facilitar la observación de algún tipo de distribución en los tiempos de simulación de cada modelos con las consideraciones mencionadas.

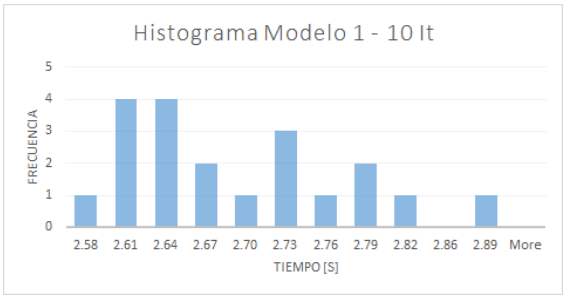
Los casos presentados a continuación corresponden a la resolución de los 3 modelos considerando 2 puntos de integración por elemento. En la Figura 4.50 se muestran los histogramas para el modelo de 192 GDL resuelto mediante el método de integración de Newmark con 0, 3 y 10 iteraciones. Figura 4.50 (a), (b) y (c) respectivamente. Sea $\bar{t}[s]$ y $\sigma[s]$ el tiempo medio y la desviación estándar respectivamente.



(a) Modelo de 192 GDL resuelto con 0 iteraciones ($\bar{t}$ = 2,662[s] - σ = 0,082[s])



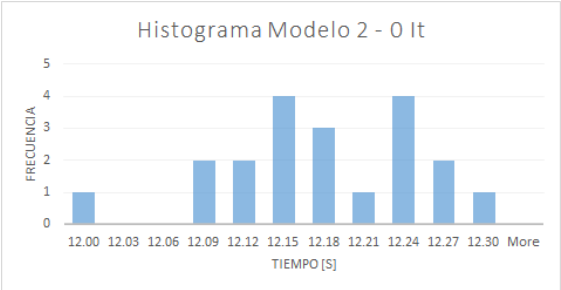
(b) Modelo de 192 GDL resuelto con 3 iteraciones ($\bar{t}$ = 2,681[s] - σ = 0,085[s])



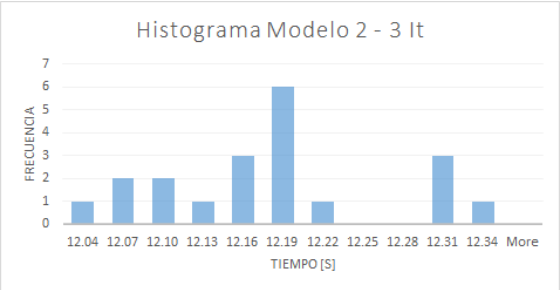
(c) Modelo de 192 GDL resuelto con 10 iteraciones ($\bar{t}$ = 2,686[s] - σ = 0,083[s])

Figura 4.50: Modelo de 192 GDL resuelto con 0, 3 y 10 iteraciones

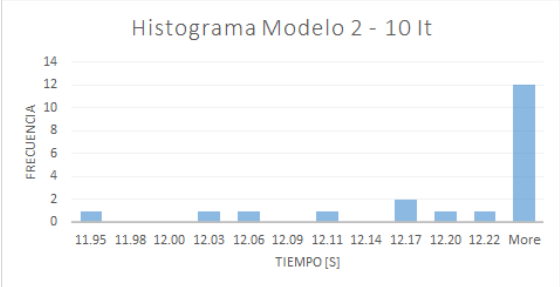
En la Figura 4.51 se muestran los histogramas para el modelo de 498 GDL resuelto mediante el método de integración de Newmark con 0, 3 y 10 iteraciones. Figura 4.51 (a), (b) y (c) respectivamente.



(a) Modelo de 498 GDL resuelto con 0 iteraciones ($\bar{t}$ = 12,168[s] - σ = 0,071[s])



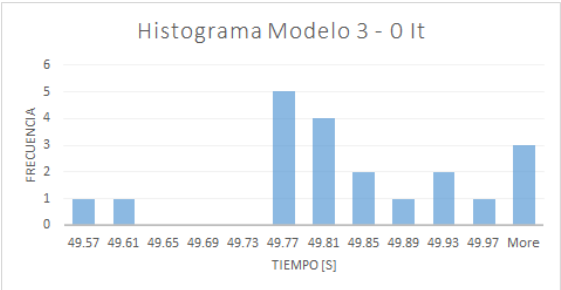
(b) Modelo de 498 GDL resuelto con 3 iteraciones ($\bar{t}$ = 12,165[s] - σ = 0,079[s])



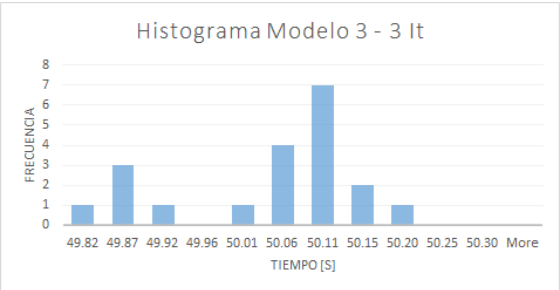
(c) Modelo de 498 GDL resuelto con 10 iteraciones ($\bar{t}$ = 12,208[s] - σ = 0,107[s])

Figura 4.51: Modelo de 498 GDL resuelto con 0, 3 y 10 iteraciones

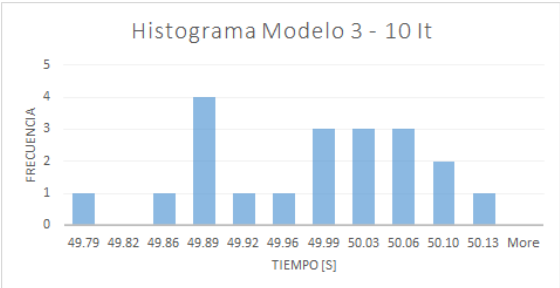
En la Figura 4.52 se muestran los histogramas para el modelo de 906 GDL resuelto mediante el método de integración de Newmark con 0, 3 y 10 iteraciones. Figura 4.52 (a), (b) y (c) respectivamente.



(a) Modelo de 906 GDL resuelto con 0 iteraciones ($\bar{t}$ = 49,819[s] - σ = 0,125[s])



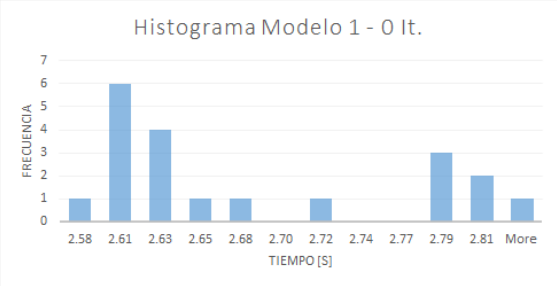
(b) Modelo de 906 GDL resuelto con 3 iteraciones ($\bar{t}$ = 50,015[s] - σ = 0,148[s])



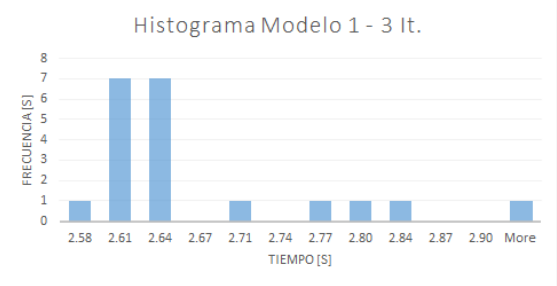
(c) Modelo de 906 GDL resuelto con 10 iteraciones ($\bar{t}$ = 49,966[s] - σ = 0,089[s])

Figura 4.52: Modelo de 906 GDL resuelto con 0, 3 y 10 iteraciones

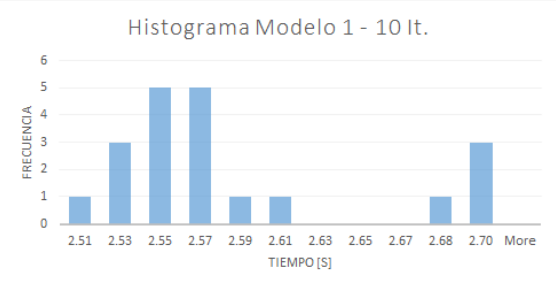
Los casos presentados a continuación corresponden a la resolución de los 3 modelos considerando 5 puntos de integración. En la Figura 4.53 se muestran los histogramas para el modelo de 192 GDL resuelto mediante el método de integración de Newmark con 0, 3 y 10 iteraciones. Figura 4.53 (a), (b) y (c) respectivamente.



(a) Modelo de 192 GDL resuelto con 0 iteraciones ($\bar{t}$ = 2,673[s] - σ = 0,085[s])



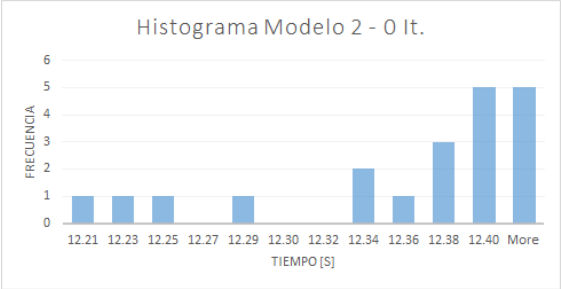
(b) Modelo de 192 GDL resuelto con 3 iteraciones ($\bar{t}$ = 2,652[s] - σ = 0,084[s])



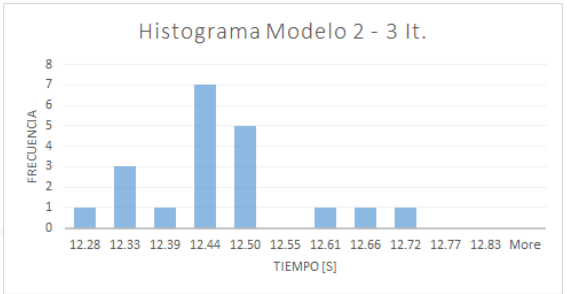
(c) Modelo de 192 GDL resuelto con 10 iteraciones ($\bar{t}$ = 2,575[s] - σ = 0,060[s])

Figura 4.53: Modelo de 192 GDL resuelto con 0, 3 y 10 iteraciones

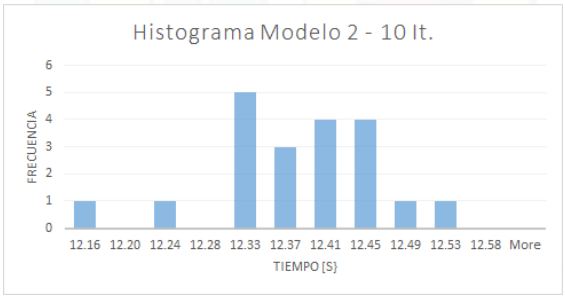
En la Figura 4.54 se muestran los histogramas para el modelo de 498 GDL resuelto mediante el método de integración de Newmark con 0, 3 y 10 iteraciones. Figura 4.54 (a), (b) y (c) respectivamente.



(a) Modelo de 498 GDL resuelto con 0 iteraciones ($\bar{t}$ = 12,386[s] - σ = 0,117[s])



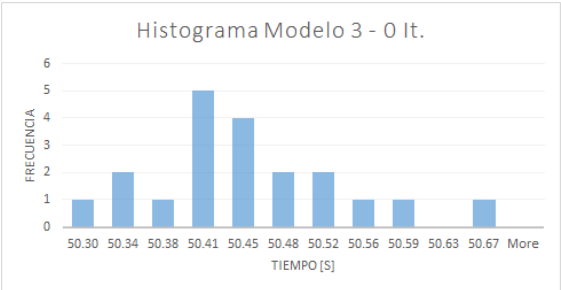
(b) Modelo de 498 GDL resuelto con 3 iteraciones ($\bar{t}$ = 12,437[s] - σ = 0,105[s])



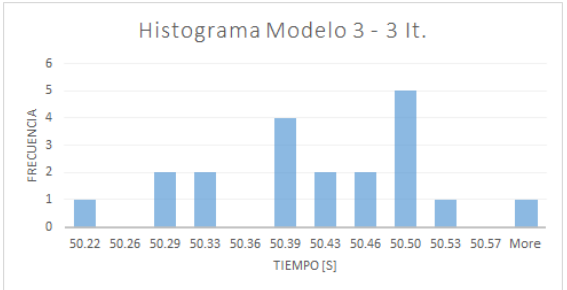
(c) Modelo de 498 GDL resuelto con 10 iteraciones ($\bar{t}$ = 12,359[s] - σ = 0,085[s])

Figura 4.54: Modelo de 498 GDL resuelto con 0, 3 y 10 iteraciones

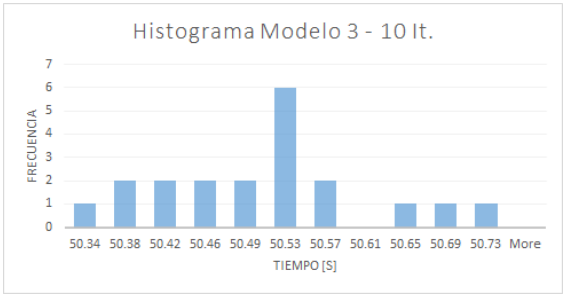
En la Figura 4.55 se muestran los histogramas para el modelo de 906 GDL resuelto mediante el método de integración de Newmark con 0, 3 y 10 iteraciones. Figura 4.55 (a), (b) y (c) respectivamente.



(a) Modelo de 906 GDL resuelto con 0 iteraciones ($\bar{t}$ = 50,432[s] - σ = 0,083[s])



(b) Modelo de 906 GDL resuelto con 3 iteraciones ($\bar{t}$ = 50,409[s] - σ = 0,094[s])



(c) Modelo de 906 GDL resuelto con 10 iteraciones ($\bar{t}$ = 50,496[s] - σ = 0,101[s])

Figura 4.55: Modelo de 906 GDL resuelto con 0, 3 y 10 iteraciones

Analizando los histogramas previamente mostrados, no se aprecia tendencia ni distribución de los datos para ningún modelo. De igual forma, la dispersión de los datos no es significativa como para esperar una gran diferencia entre simulaciones. Se concluye que estos depende de la máquina utilizada para su ejecución y de las aplicaciones ejecutándose en esta a la hora de la simulación.

En la tabla 4.16 se aprecian los tiempos promedio de ejecución de cada modelos para todos los casos expuestos previamente.

Tabla 4.16: Tabla resumen tiempos de simulación.

	2 Puntos de Integración			5 Puntos de Integración		
Modelo	0 It.	3 It.	10 It	0 It.	3 It.	10 It
162 GDL	2.66	2.68	2.69	2.67	2.65	2.58
498 GDL	12.17	12.17	12.21	12.38	12.44	12.36
906 GDL	49.82	50.02	49.97	50.43	50.41	50.50

Dado que el modelo de 498 GDL nos entrega una buena respuesta, en comparación con el modelo de 908 GDL, y además un tiempo de resolución menor a la duración del registro sísmico, se decide trabajar con este. Finalmente, se analiza la influencia en la respuesta de la cantidad de puntos de integración seleccionados. Se comparan las respuestas para 2 y 5 puntos de integración.

Es apreciable en la Figura 4.56 que no existe una diferencia considerable en cuanto a la respuesta considerando 2 o 5 puntos de integración. Esto es respaldado por el estudio realizado por Scott et al. (2004) en el cual se indica que 2 puntos de integración para elementos del tipo Displacement-Based es suficiente para capturar la respuesta con precisión.

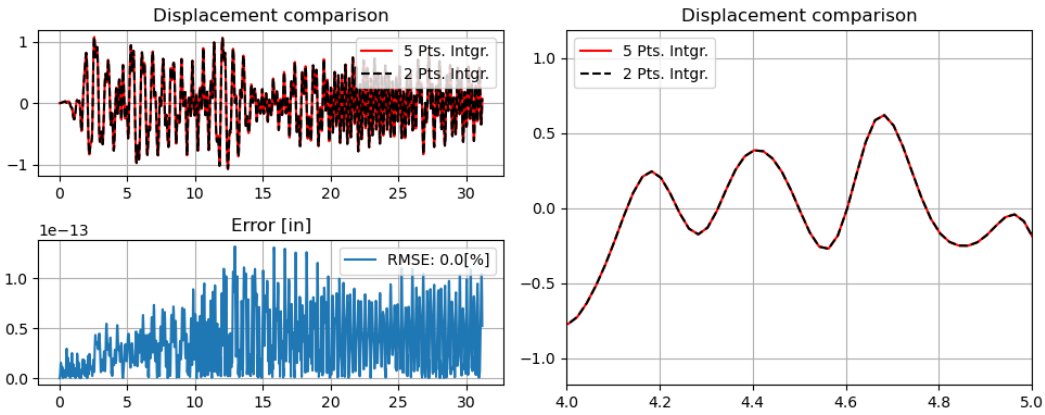


Figura 4.56: Comparación respuesta modelo de 498 GDL con 2 y 5 puntos de integración

De igual forma, y haciendo uso del comando *testNorm()*, se analiza la norma de la prueba de convergencia para el último paso de análisis. La prueba utilizada corresponde a *EnergyIncr*, esta utiliza el producto escalar del vector solución y la norma del lado derecho de la ecuación matricial para determinar si se alcanzó la convergencia. El significado físico de esta cantidad depende del integrador y del controlador de restricciones elegido. Por lo general, aunque no siempre, es igual al desequilibrio de energía en el sistema. A continuación, se aprecian las normas para el modelo de 498 GDL considerando 2 y 5 puntos de integración.

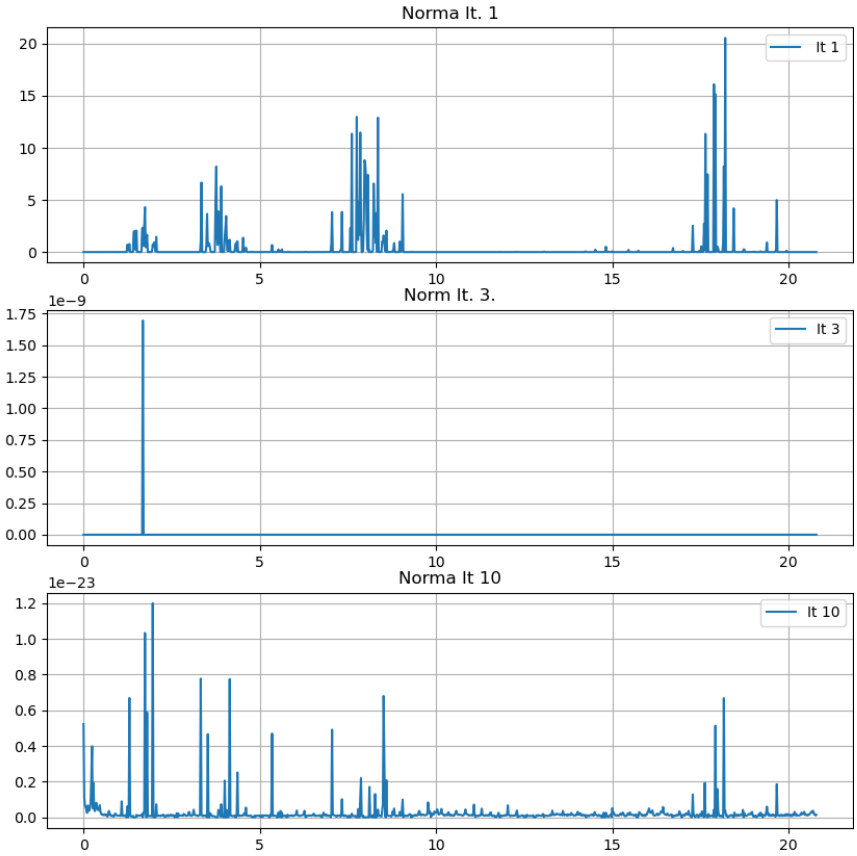


Figura 4.57: Norma de la prueba de convergencia considerando 2 puntos de integración. En el eje Y se indica la norma del test versus el tiempo [s] en el eje X

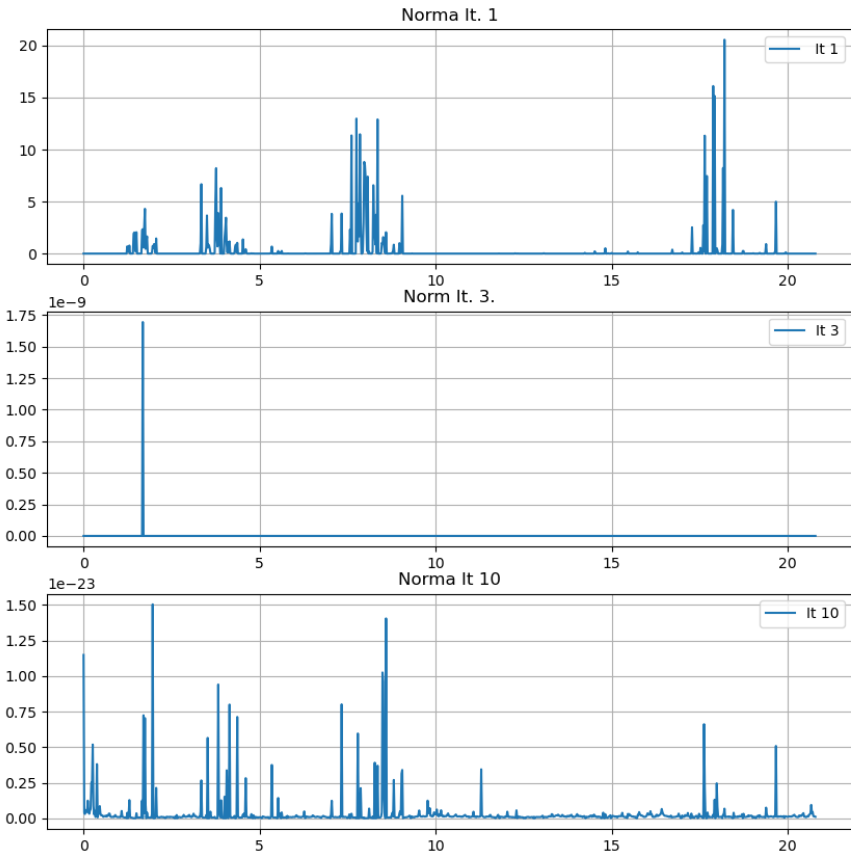


Figura 4.58: Norma de la prueba de convergencia considerando 5 puntos de integración. En el eje Y se indica la norma del test versus el tiempo [s] en el eje X

Analizando las Figuras 4.57 y 4.58, se aprecia que el valor entregado por el *testNorm()* se ve reducido desde ordenes de magnitud de 2×10^{1} a valores del orden de 1×10^{-9} luego de la tercera iteración. En consecuencia, podemos concluir que independiente de la cantidad de puntos de integración, con 3 pasos de integración es suficiente para alcanzar una precisión aceptable de la respuesta.

Una restricción importante que no se ha mencionado es que cada paso de integración del modelo numérico debe resolverse en un tiempo menor al estipulado (δt_n). No basta con que el modelo en el tiempo global se resuelva en menos tiempo que el de duración del registro sísmico. Para enfrentar esta limitación, se trabaja con un $\delta t_n = 0{,}03$. Esta limitación proviene de los recursos computacionales disponibles, en este caso, el notebook MSI GF75 Thin 9RCX. Si bien esto puede parecer una buena solución para evitar las restricciones que impone el tiempo real, se debe considerar la pérdida de información en el registro sísmico y la afectación en la respuesta global y precisión del modelo. Tal como se aprecia en la Figura 4.59, a pesar de que el cambio de paso temporal es pequeño, su impacto en la respuesta es significativa produciendo una diferencia entre las respuestas de más de un 87.41 %.

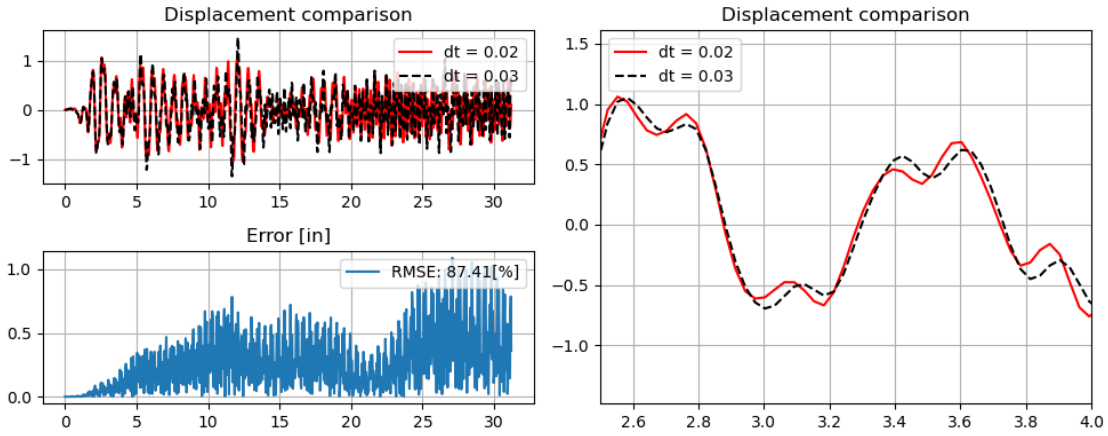


Figura 4.59: Diferencia en la respuesta global de la estructura al modificar el paso temporal de 0.02 a 0.03.

Finalmente, y considerando todo lo previamente expuesto, se resolverá el modelo de 498 GDL considerando 2 puntos de integración, con un paso temporal de 0.03 [s], se utilizará el integrador de Newmark con 3 iteraciones y será resuelto únicamente para simulaciones en modo externo ejecutándose en la Raspberry Pi 3 Model B.

4.8.2. Resultados Simulaciones Modo Externo

Se muestran a continuación los resultados obtenidos con parámetros fijos del filtro DDMRT. La figura 4.60 nos muestra la comparación entre el desplazamiento objetivo (x_t) y el desplazamiento medido (x_m), junto con su indicador correspondiente J_2. Los resultados muestran un error NRMSE de 2,1 % junto con un retraso entre señales de 0,22[ms]. Estos resultados nos permiten validar la simulación en términos de compensación de retardo entre las señales producidas por la dinámica y el retardo del actuador.

La figura 4.61 nos muestra la comparación entre la señal medida en nuestro vRTHS versus el desplazamiento de referencia de la estructura. Se ve un error NRMSE de aproximadamente 2,13 %, resultado que indica una resolución correcta del vRTHS.

De la misma forma, la Figura 4.62 nos muestra la comparación entre la histéresis de la subestructura experimental en nuestra vRTHS versus la de referencia.

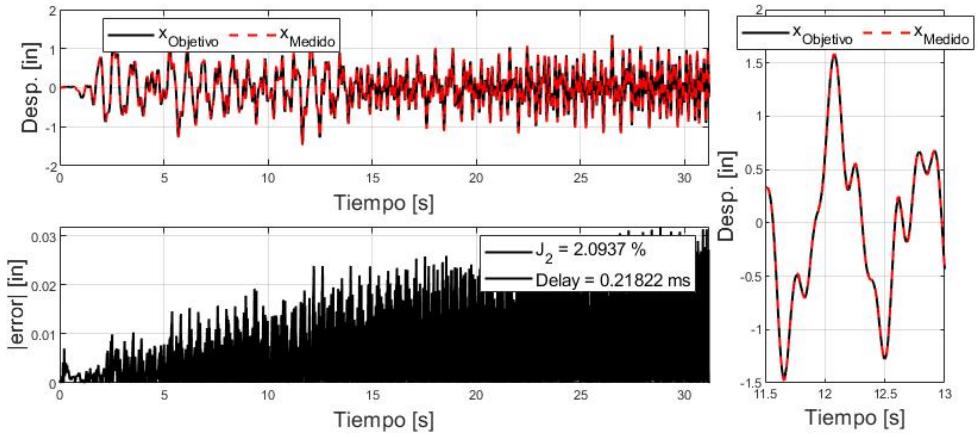


Figura 4.60: Desplazamiento Objetivo v/s Medido e indicador J_2

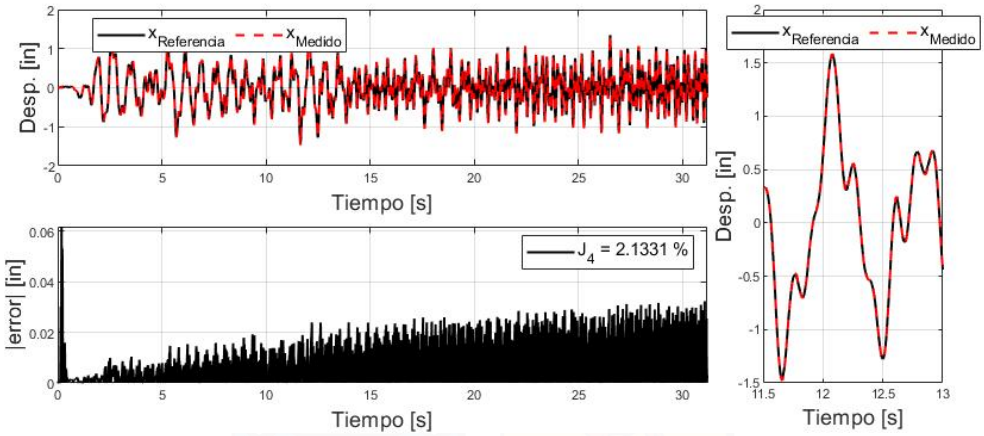


Figura 4.61: Desplazamiento de Referencia v/s Medido e indicador J_4.

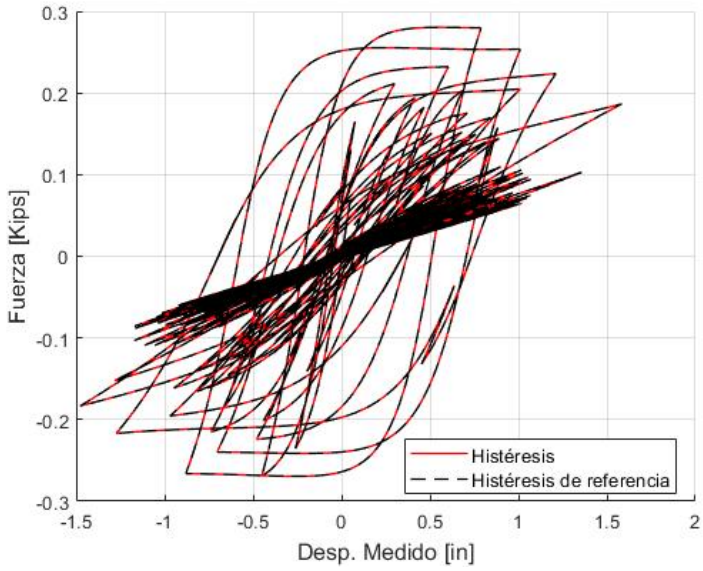


Figura 4.62: Histéresis de la Subestructura experimental (Referencia v/s Medido)

Finalmente, los resultados se resumen en las Tablas 4.17, y 4.18. Como es apreciable, se obtienen buenos resultados para todos los casos.

Tabla 4.17: Tabla resumen de simulaciones externas modelo 3D de 498 GDL no lineal parámetros fijos.

Caso	Integrador	J_2 [%]	$J_{4,Desp}$ [%]	$J_{4,Fuerza}$ [%]	$delay$ [ms]
No Lineal	Newmark (Explícito)	2.0937	2.1331	3.5275	0.21822

Tabla 4.18: Tabla resumen de simulaciones externas modelo 3D de 498 GDL no lineal parámetros adaptivos.

Caso	Integrador	J_2 [%]	$J_{4,Desp}$ [%]	$J_{4,Fuerza}$ [%]	$delay$ [ms]
No Lineal	Newmark (Explícito)	2.1396	5.1559	3.8818	0.22504

5 | Conclusiones

5.1. Comentarios Generales

Este estudio presenta el desarrollo de los componentes fundamentales para coordinación de subestructuras en una simulación híbrida virtual en tiempo real. Un factor importante que determina la capacidad de vRTHS para representar un comportamiento realista del sistema de referencia es la fidelidad de la subestructura numérica. Por lo tanto, una plataforma de coordinación basado en la arquitectura Cliente-Servidor es propuesto, desarrollado e implementado. De igual forma, se implementa una técnica de compensación basada en modelos adaptativos para simulación híbrida en tiempo real. El aspecto novedoso del método propuesto es que las condiciones iniciales del controlador se formulan independientemente de la muestra física conectada al actuador. Adicionalmente, un método para realizar la transición de múltiples velocidades en simulación híbrida en tiempo real es implementado.

Las simulaciones fueron separadas en dos grandes grupos. (i) Se considera la utilización de la plataforma de coordinación OpenFresco. Este fue utilizado e implementado mediante el protocolo de comunicación Cliente-Servidor para acoplar un software de análisis de elementos finitos (OpenSees) con el sistema de control en tiempo real (Matlab/Simulink). (ii) Las siguientes simulaciones fueron resueltas utilizando un la plataforma vRTHS Cliente-Servidor desarrollada por el autor, para acoplar un software de análisis de elementos finitos (OpenSeesPy) con el sistema de control en tiempo real (Matlab/Simulink).

Dentro de la simulaciones utilizando OpenFresco, tenemos tres casos estudio. 1 y 2 corresponden a modelos bi y tridimensionales. Estos consideran una subestructura experimental resuelta en otro archivo OpenSees, es decir, un modelo de elementos finitos. Los resultados expuestos muestran buenos resultados, tanto en sincronización como en comparación con los modelos de referencia. Adicionalmente, se provee evidencia suficiente para validar el protocolo de comunicación y su aplicabilidad para comunicar diferentes subestructuras. El caso estudio 3 utiliza una subestructura experimental resuelta en Simulink, pero mediante análisis isogeométrico, es decir, integra el análisis de elementos finitos en herramientas de diseño CAD basadas en NURBS convencionales. Este modelo, para el caso analizado, nos demuestra una ventaja tanto para capturar estructuras con geometrías complejas, como para converger con una menor cantidad de grados de libertad que los modelos de elementos finitos. Adicionalmente, se probó su aplicabilidad a vRTHS abriendo la oportunidad a incluir modelos personalizables. Se reconoce la falta de códigos altamente optimizados, permitiendo no solo mejorar la rapidez de simulación, sino que, de igual forma, el uso de mallados más finos que los que se han empleado en este trabajo con el fin de aumentar la precisión de los resultados.

Si bien OpenFresco se desenvuelve de buena manera en ambiente virtuales locales, no fue posible su utilización en modo externo con un hardware dedicado. La causa es la forma en que se encuentra pensada la plataforma. Este se diseño para ser utilizado en hardware con una memoria compartida (SCRAMNet+) el cuál elimina la necesidad de conectar directamente los equipos dado que todos se ejecutan en la misma ram y un xPc Target como hardware externo. Esta metodología de trabajo disminuye la latencia de datos, pero limita su uso a laboratorios que cuenten con estos equipos en específico. Como solución, se desarrolló e implemento una plataforma de coordinación.

La validación del protocolo de comunicación cliente-servidor propuesto se logra para los vRTHS locales y externos. Adicionalmente, se verifica su aplicabilidad a subestructuras, tanto numéricas como

experimentales, de tipo lineal y no lineal. Para el caso externo, el método Cliente-Servidor se implementa en vRTHS a través de una red con microcontrolador dedicado (cliente/Simulink), ejecutándose en una Raspberry Pi 4 Modelo B, y máquinas FEA (servidores/OpenSeesPy) en una configuración de laboratorio. Se proporcionan dos ejemplos (2D y 3D) de ingeniería sísmica/estructural para ilustrar su implementación y validación de software, obteniendo buenos y prometedores resultados.

Tanto la técnica de compensación como el método de transición entre tasas implementado fueron validados y demuestran un excelente rendimiento de seguimiento y robustez en todos los escenarios propuestos. Un punto importante a destacar, tal como se vio en la subsección 4.7.3, la transición induce un error que alcanza valores máximos cercanos a un 4[%]. Estos errores se vieron reducidos en las subsecciones 4.7.4 y 4.7.5, esto nos indica que la compensación basa en modelos adaptivos no solo compensa el retraso producto de la dinámica del actuador, sino que adicionalmente, ayuda a compensar el retraso producto de la transición entre las tasas de las subestructuras numéricas y experimentales.

5.2. Trabajo Futuro

Algunos de los estudios futuros se muestran a continuación:

- Validación experimental de la metodología de coordinación propuesta. Un punto importante a destacar, es que una plataforma de coordinación, que sea independiente de los componente que se tengan en laboratorio, permitiría expandir y diversificar los ensayos de Simulación Híbrida en Tiempo Real.
- Implementación del algoritmo de coordinación en hardware en tiempo real, como por ejemplo, dSpace MicroLabBox. Adicionalmente, la implementación en laboratorio.
- Al tratarse de una metodología basada en el protocolo de comunicación Cliente-Servidor, es aplicable a cualquier programa de análisis de elementos finitos que posea la capacidad para enviar y recibir señales. A causa de lo mismo, es posible extenderlo a RTHS distribuido geográficamente. Sin embargo, esto implica un estudio en profundidad de la compensación del retraso producto del tiempo que le toma a la señal viajar entre locaciones geográficamente distintas.
- Estudio de múltiples actuadores. Como bien fue mencionado, la metodología de comunicación propuesta presenta la potencialidad para ser utilizado con múltiples actuadores. El principal desafío de esto, es el control de los actuadores ya que se debe considerar la interacción con la muestra física y entre los actuadores.
- Optimización de códigos para metodología IGA. Esto permitiría resolver estructuras considerando su forma fidedigna su geometría, y no aproximada como al utilizar FEM. Adicionalmente, y como ya se comprobó, se lograrían obtener resultados más precisos utilizando un mallado más grueso que con mediante FEM.

Bibliografía

Ahmadizadeh, M; Mosqueda, G; y Reinhorn, A.M. (2008). Compensation of actuator delay and dynamics for real-time hybrid structural simulation. *Earthquake Engineering and Structural Dynamics*, (37), 21 – 42. (document), 2.4, 2.26

Andreas Junghanns; Cláudio Gomes; Christian Schulze; Klaus Schuch; Pierre R.; Matthias Blaesken; Irina Zacharias; Andreas Pillekeit; Karl Wernersson; Torsten Sommer; Christian Bertsch; Torsten Blochwitz; y Masoud Najafi (2021). The Functional Mock-up Interface 3.0 - New Features Enabling New Applications. *Proceedings of 14th Modelica Conference 2021, Linköping, Sweden, September 20-24, 2021*, 181, 17–26. 2.2, 2.8

Ashasi-Sorkhabi, Ali; Malekghasemi, Hadi; y Mercan, Oya (2015). Implementation and verification of real-time hybrid simulation (RTHS) using a shake table for research and education. *JVC/Journal of Vibration and Control*, 21(8), 1459–1472. 1

Blakeborough, By A y Williams, M S (2001). The development of real-time substructure testing. (pp. 1869–1891). 2

Blochwitz, T.; Otter, M.; Arnold, M.; Bausch, C.; Clauss, C.; Elmqvist, H.; Junghanns, A.; Mauss, J.; Monteiro, M.; Neidhold, T.; Neumerkel, D.; Olsson, H.; Peetz, J.-V.; y Wolf, S. (2011). The Functional Mockup Interface for Tool independent Exchange of Simulation Models. *Proceedings from the 8th International Modelica Conference, Technical Univeristy, Dresden, Germany*, 63(March), 105–114. 2.2, 2.8

Blockwitz, Torsten; Otter, Martin; Akesson, Johan; Arnold, Martin; Clauss, Christoph; Elmqvist, Hilding; Friedrich, Markus; Junghanns, Andreas; Mauss, Jakob; Neumerkel, Dietmar; Olsson, Hans; y Viel, Antoine (2012). Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models. *Proceedings of the 9th International MODELICA Conference, September 3-5, 2012, Munich, Germany*, 76(July 2015), 173–184. 2.2, 2.8

Bonnet, Paul A; Lim, C N; Williams, M S; Blakeborough, A; Neild, S A; Stoten, D P; y Taylor, C A (2007). Real-time hybrid experiments with Newmark integration, MCSmd outer-loop control and multi-tasking strategies. *Earthquake Engineering & Structural Dynamics*, 36(1), 119–141. 3.2

Buchet, P y Pegon, P (1994). Psd Testing With Substructuring :implementation and use. *JRC special publication*, No.I.94.25(February). 2.1.3.1

Carrion, Juan E y Spencer, Billie F (2007). *Model-based Strategies for Real-time Hybrid Testing. Technical Report NSEL-006*. Technical report, Urbana, IL. (document), 2.3.1, 2.3.1.1, 2.1, 2.4.1

Chae, Yunbyeong; Kazemibidokhti, Karim; y M. Ricles, James (2013). Adaptive time series compensator for delay compensation of servo-hydraulic actuator systems for real-time hybrid simulation. *Earthquake Engineering and Structural Dynamics*, (42), 1697–1715. 2.4.1

Chang, Shuenn Yih (2002). Explicit pseudodynamic algorithm with unconditional stability. *Journal of Engineering Mechanics*, 128(9), 935–947. 2.5

Chen, Cheng (2007). Development and numerical simulation of hybrid effective force testing method. *Ph.D. Dissertations*, (pp. 344). 2.4.1

Chen, Pei-Ching; Chang, Chia-Ming; Spencer, Jr., Billie F.; y Tsai, Keh-Chyuan (2015). Adaptive model-based tracking control for real-time hybrid simulation. *Bulletin of Earthquake Engineering*, 13(6), 1633–1653. 2.4.1, 2.4.1.1.1, 2.4.1.1.2

Chung, J. y Hulbert, G. M. (1993). A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized-α method. 2.5

Cláudio Gomes; Giuseppe Abbiati; y Peter Gorm Larsen (2021). Seismic Hybrid Testing using FMI-based Co-Simulation. *Proceedings of 14th Modelica Conference 2021, Linköping, Sweden, September 20-24, 2021*, 181, 287–295. 2.8

Combescure, Didier y Pegon, Pierre (1997). -Operator Splitting Time Integration Technique for Pseudodynamic Testing Error Propagation Analysis. *Soil Dynamics and Earthquake Engineering*, 16(7-8), 427–443. 2.5, 4.7.4

Dassault-Systemes (2013). Documentation (Abaqus), 2013. *Abaqus User's Guide*, (pp. 1138). 2.2.2.1

De Klerk, D.; Rixen, D. J.; y Voormeeren, S. N. (2008). General framework for dynamic substructuring: History, review, and classification of techniques. *AIAA Journal*, 46(5), 1169–1181. 2.1.3.1, 4.7.1

Deierlein, Gregory G; Reinhorn, Andrei M; y Willford, Michael R (2010). Nonlinear Structural Analysis For Seismic Design: A Guide for Practicing Engineers. *National Standards and Technology(NIST), U.S. Department of Commerce*, (4), 36. (document), 2.27

Dermitzakis, S. N. y Mahin, S.A. (1985). Development of substructuring techniques for on-line computer controlled seismic performance testing. *Report to the National Science Foundation*, (UCB/EERC-85/04), 1–170. 2.1.3.1, 4.7.1

Diming, J; Shield, C; French, C; Bailey, F; y Clarck, A (1999). Effective force testing: A method of seismic simulation for structural testing. *Journal of Structural Engineering*, 125(September), 1028–1037. 2.1.2

Dyke, S. J .; Spencer Jr, B. F.; Quast, P.; y Sain, M. (1995). The Role of Control-Structure Interaction in Protective System Design. *Journal of Engineering Mechanics*, 121(2)(FEBRUARY 1995), 322–338. 2.3.1, 2.3.1.1, 2.4.1

Enokida, Ryuta (2020). Basic examination of two substructuring schemes for shake table tests. *Structural Control and Health Monitoring*, 27(4), 1–23. 1

Fei, Zhu; Jinting, Wang; Feng, Jin; y Liqiao, Lu (2019). Control performance comparison between tuned liquid damper and tuned liquid column damper using real-time hybrid simulation. *Earthquake Engineering and Engineering Vibration*, 18(3), 695–701. 1

Fermandois, G; Galmez, C; y Valdebenito, M (2020). Optimal Gain Calibration of Adaptive Model-Based Compensation for Real-Time Hybrid Simulation Testing. *17th World Conference on Earthquake Engineering (17WCEE), Sendai, Japan*. 2.4.1.1.1, 2.4.1.1.1, 2.4.1.1.2

Fermandois, Gastón y Mera, Diego (2021). Çlient-Server application for vRTHS", 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4646490. 4.2

Gantner, Gregor y Praetorius, Dirk (2020). Adaptive IGAFEM with optimal convergence rates: T-splines. *Computer Aided Geometric Design*, 81, 101906. 2.7.5

Ghaffary, Azin (2019). Comprehensive nonlinear seismic performance assessment of MR damper controlled systems using virtual real time hybrid simulation. *The structural design of tall and special buildings*, 1606(January), 1–18. 4.5

Gu, Quan y Ozcelik, Ozgur (2011). Integrating opensees with other software with application to coupling problems in civil engineering. *Structural Engineering and Mechanics*, 40(1), 85–103. 1

Gui, Yao; Wang, Jin Ting; Jin, Feng; Chen, Cheng; y Zhou, Meng Xia (2014). Development of a family of explicit algorithms for structural dynamics with unconditional stability. *Nonlinear Dynamics*, 77(4), 1157–1170. 2.5

Guo, Tong; Chen, Cheng; Xu, Weijie; y Sanchez, Frank (2014). A frequency response analysis approach for quantitative assessment of actuator tracking for real-time hybrid simulation. *Smart Materials and Structures*, 23(4). 4.1

Gálmez, C (2021). Adaptive Dynamic Compensation For Real-Time Hybrid Simulation Testing. Master's thesis, Universidad Técnica Federico Santa María. 2.4.1.1

Hakuno, M; Shidawara, M; y Hara, T (1969). Dynamic destructive test of a cantilever beam controlled by an analog-computer. *Trans Jpn Soc Civ Eng*, (171), 1–9. 2.1.3, 2.1.3

Heath, Michael T (2001). *Scientific Computing, An Introductory Survey*. (document), 3.4

Horiuchi, T.; Inoue, M.; Konno, T.; y Namita, Y. (1999). Real-time hybrid experimental system with actuator delay compensation and its application to a piping system with energy absorber. *Earthquake Engineering and Structural Dynamics*, 28(10), 1121–1141. 1, 2.4.1, 2.4.1, 2.5

Horiuchi, T. y Konno, T. (2001a). A new method for compensating actuator delay in real-time hybrid experiments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 359(1786), 1893–1909. 2.4.1

Horiuchi, Toshihiko y Konno, Takao (2001b). A new method for compensating actuator delay in real-time hybrid experiments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 359(1786), 1893–1909. 3.2

Horiuchi, Toshihiko; Nakagawa, Masaki; Sugano, Masahuru; y Konno, Takao (1995). Development of a real-time hybrid experimental system with actuator delay compensation. *In In Proc. 11th World Conf. Earthquake Engineering*, (pp. 660). 2.1.3, 2.4.1

Hughes, T.J.R.; Cottrell, J.A.; y Bazilevs, Y. (2005). Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39), 4135–4195. 4.5

Ierusalimschy, Roberto (2003). *Programming in Lua*. 2.2.3

Igarashi, Akira; Iemura, Hirokazu; Suwa, Takanori; Horiuchi, Toshihiko; Inoue, Masahiko; y Konno, Takao (2000). Development of substructured shaking table test method. *12th World Conference on Earthquake Engineering*, (pp. 1–7). 2.5

INN (2009). *NCh 1537 of86 mod2009*. Technical report, Instituto Nacional de Normalización, Chile. 4.4

Ioannou, Petros y Baldi, Simone (2010). *Robust Adaptive Control*. 2.4.1

Kammula, Viswanath; Erochko, Jeffrey; Kwon, Oh-Sung; y Christopoulos, Constantin (2013). Measuring bias in structural response caused by ground motion scaling. *Earthquake Engineering and Structural Dynamics*. 2.2.2.3

Karavasilis, Theodore L; Ricles, James; y Seo, Choung-Yeol (2010). *HybridFEM : A program for dynamic time history analysis of 2D inelastic framed structures and real-time hybrid simulation.* Technical Report 610. 2.2, 2.8

Karavasilis, Theodore L.; Ricles, James M.; Sause, Richard; y Chen, Cheng (2011). Experimental evaluation of the seismic performance of steel MRFs with compressed elastomer dampers using large-scale real-time hybrid simulation. *Engineering Structures*, 33(6), 1859–1869. 1

Khalil, Hassan K. (2002). *Nonlinear Systems*. 2.4.1

Kolay, C. y Ricles, J.M. (2014). Development of a family of unconditionally stable explicit direct integration algorithms with controllable numerical energy dissipation. *Earthquake Engng Struct. Dyn.*, (43), 1361–1380. 2.5

Krawinkler, H. (2000). A perspective on experimental research in earthquake engineering. *Journal of Earthquake Technology*, 37(iii-iv). 2.1.2

Kwon, O; Elnashai, A S; Spencer, B F; y Park, K (2007). UI-SIMCOR: A global platform for hybrid distributed simulation. *9th Canadian Conference on Earthquake Engineering*, (June), 139–149. (document), 1, 2.2, 2.6, 2.2.2.1, 2.8

Lamata Martinez, Ignacio; Santacana, Ferran Obon; Williams, Martin S.; Blakeborough, Anthony; y Dorka, Uwe E. (2016). Celestina-Sim: Framework to Support Distributed Testing and Service Integration in Earthquake Engineering. *Journal of Computing in Civil Engineering*, 30(1), 04014119. 2.2, 2.8

Li, Tengfei; Su, Mingzhou; Sui, Yan; y Ma, Lei (2021). Real-time hybrid simulation on high strength steel frame with Y-shaped eccentric braces. *Engineering Structures*, 226(September 2020), 111–369. 1

Lu, Jia (2009). Circular element: Isogeometric elements of smooth boundary. *Computer Methods in Applied Mechanics and Engineering*, 198(30), 2391–2402. 4.5

Maghareh, Amin; Silva, Christian E; y Dyke, Shirley J (2018). Servo-hydraulic actuator in controllable canonical form : Identification and experimental validation. *Mechanical Systems and Signal Processing*, 100, 398–414. 2.3.1, 2.3.1.1

Maghareh, Amin.; Waldbjorn, Jacob P.; Dyke, Shirley J.; Prakash, Arun; y Ozdagli, Ali I. (2016). Adaptive multi-rate interface: development and experimental verification for real-time hybrid simulation. *Earthquake Engineering Structural Dynamics*, (45), 1411–1425. 3.2

Magonette, G. (2001). Development and application of large-scale continuous pseudo-dynamic testing techniques. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 359(1786), 1771–1799. 2.1.3

MathWorks, Inc. (2019). Getting Started with Simulink Support Package for Raspberry Pi Video. 3.3.2.2.1

Matlab (2020). *Matlab Primer version 9.9.0 (R2020b)*. Natick, Massachusetts: The MathWorks Inc. 1, 3.2.1, 3.2.2.4, 4.2, 4.5

McCrum, D. P. y Williams, M. S. (2016). An overview of seismic hybrid testing of engineering structures. *Engineering Structures*, 118, 240–261. 2

McKenna, Frank (2011). OpenSees: A framework for earthquake engineering simulation. *Computing in Science and Engineering*, 13(4), 58–66. 1, 2.2.2.1, 4.2, 4.5

Mera, Diego y Fermandois, Gastón (2021). Çlient-Server coordination of multi-rate tasks in real-time hybrid simulation testing", 2021. [Online]. Available: https://github.com/DiegoNHMeraM/ClientServer_OpenSeesPy. 3.3.3, 4.6, A

Mera, Diego; Gomez, Fernando; y Fermandois, Gaston (2022). Data-driven multi-rate transitioning for real-time hybrid simulation tests with large-scale numerical substructures. Manuscript in preparation. 3.2.2

Molina, F. J.; Verzeletti, G.; Magonette, G.; Buchet, Ph; Renda, V.; Geradin, M.; Parducci, A.; Mezzi, M.; Pacchiarotti, A.; Federici, L.; y Mascelloni, S. (2002). Pseudodynamic tests on rubber base isolators with numerical substructuring of the superstructure and strain-rate effect compensation. *Earthquake Engineering and Structural Dynamics*, 31(8), 1563–1582. 2.1.3.1

Mortazavi, Pedram; Huang, Xu; Kwon, Oh Sung; y Christopoulos, Constantin (2017). An overview of the university of toronto simulation (UT-SIM) framework and its application to the performance assessment of structures. *International Conference on Advances in Experimental Structural Engineering*, 2017-Septe(September), 503–523. (document), 1, 2.2, 2.7, 2.2.2, 2.8

Mosqueda, Gilberto; Stojadinovic, Bozidar; y Mahin, Stephen (2005). Implementation and accuracy of continuous hybrid simulation with geographically distributed substructures. *Technical Report EERC 2005-02, Earthquake Engineering Research Center, University of California, Berkeley, Berkeley, CA.*, (January), 168. 3.2, 3.2.1

Nakashima, M. (2001). Development , potential , and limitations of real-time online (pseudo-dynamic) testing. (Aci 1985), 1851–1867. 1, 2.1.3, 2.1.3.1

Nakashima, M. (2020). Hybrid simulation: An early history. *Earthquake Engineering and Structural Dynamics*, (March), 1–14. 1, 2.1.3

Nakashima, M; Ishida, M; y Ando, K (1990). Integration techniques for substructure Pseudo dynamic test. *4th U.S. national conference of earthquake engineering. Earthquake Engineering Research Institute, Palm Springs, CA*. 2.1.3.1

Nakashima, Masayoshi; Kato, Hiroto; y Takaoka, Eiji (1992). Development of real[U+2010]time pseudo dynamic testing. *Earthquake Engineering Structural Dynamics*, 21(1), 79–92. 2.5

Neuenhofer, Ansgar y Filippou, Filip C. (1997). Evaluation of Nonlinear Frame Finite-Element Models. *Journal of Structural Engineering*, 123(7), 958–966. 2.2.3

Newmark, Nathan M. (1959). Method of Computation for Structural Dynamics. 2.4.1, 2.5, 4.7.4

Nguyen, Vinh Phu; Anitescu, Cosmin; Bordas, Stéphane P.A.; y Rabczuk, Timon (2015). Isogeometric analysis: An overview and computer implementation aspects. *Mathematics and Computers in Simulation*, 117, 89–116. (document), 2.28, 2.29, 2.30, 2.7.5

Nurbs, Modelador (2006). Rhinoceros: Modelador NURBS para Windows. *Europe*. 4.5.1

Palacio-Betancur, Alejandro y Gutierrez Soto, Mariantonieta (2019). Adaptive tracking control for real-time hybrid simulation of structures subjected to seismic loading. *Mechanical Systems and Signal Processing*, 134, 106345. 2.4.1

Pan, Peng; Tomofuji, Hiroshi; Wang, Tao; Nakashima, Masayoshi; Ohsaki, Makoto; y Mosalam, Khalid M. (2006). Development of peer to peer (P2P) internet online hybird test system. *Earthquake Engineering and Structural Dynamics*, 35(7), 867–890. 2.2.4.1

P.C., Kohnke (1982). *ANSYS. In: Brebbia C.A. (eds) Finite Element Systems*. Heidelberg: Springer, Berlin. 4.5.3

Peroni, Marco; Pegon, Pierre; Molina, Francisco Javier; Buchet, Philippe; Peroni, Marco; Pegon, Pierre; Molina, Francisco Javier; Buchet, Philippe; Pegon, Pierre; y Molina, Francisco Javier (2021). Ethernet-Based Servo-Hydraulic Real-Time Controller and DAQ at ELSA for Large Scale Experiments ELSA for Large Scale Experiments. *Journal of Earthquake Engineering*, 00(00), 1–32. 2.2, 2.8

Phillips, Brian M y Spencer, Jr., Billie F. (2011). Model-Based Feedforward-Feedback Tracking Control for Real-Time Hybrid Simulation. *NSEL Report Series*, (August). 2.4.1

Ricles, James M. y Chen, Cheng (2008). Development of direct integration algorithms for structural dynamics using discrete control theory. *Journal of Engineering Mechanics*, 134(8), 676–683. 2.5

S-Frame Software Inc. (2013). S-FRAME and S-STEEL R11.0 Reference Manual. 2.2.2.1

Sadeghian, Vahid; Kwon, Oh Sung; y Vecchio, Frank (2015). An integrated framework for the analysis of mixedtype reinforced concrete structures. *COMPDYN 2015 - 5th ECCOMAS Thematic Conference on Computational Methods in Structural Dynamics and Earthquake Engineering*, (May), 1011–1023. 2.2.2.1

Saouma, Victor; Haussmann, Gary; Kang, Dae-Hung; y Ghannoum, Wassim (2014). Real-Time Hybrid Simulation of a Nonductile Reinforced Concrete Frame. *Journal of Structural Engineering*, 140(2), 04013059. (document), 2.2, 2.2.3, 2.8, 2.8

Saouma, Victor; Kang, Dae-Hung; y Haussmann, Gary (2012). A computational finite-element program for hybrid simulation. *Earthquake Engineering and Structural Dynamics*, (41), 375–389. (document), 1, 2.2, 2.2.3, 2.8, 2.8

Schellenberg, Andreas; Kim, Hong K; Takahashi, Yoshikazu; Fenves, Gregory L; y Mahin, Stephen A (2009a). OpenFresco Command Language Manual. (July). 1, 2.1.3.1, 2.2, 2.2.6, 2.8, 4.2

Schellenberg, Andreas H. (2018). Github. (document), 2.3.1.2, 2.3.1.2

Schellenberg, Andreas H; Mahin, Stephen A; y Fenves, Gregory L (2009b). Advanced Implementation of Hybrid Simulation. PEER Report 2009/104. *Peer 2009/104*, (November), 286. (document), 2.2, 2.2.6, 2.12, 2.13, 2.14a, 2.14b, 2.15a, 2.15b, 2.16, 2.17, 2.8, 3.2, 3.2.1, 4.2

Scheller, Joern; Constantinou, Michael C; y Hall, Ketter (1999). *Response History Analysis of Structures with Seismic Isolation and Energy Dissipation Systems : Verification Examples for Program SAP2000 by*, volume 14260-4300. 2

Scott, Michael H.; Franchin, Paolo; Fenves, Gregory L.; y Filippou, Filip C. (2004). Response Sensitivity for Nonlinear Beam–Column Elements. *Journal of Structural Engineering*, 130(9), 1281–1288. 4.8.1

Shah, Surendra P.; Wang, Ming Liang; y Chung, Lan (1987). Model concrete beam-column joints subjected to cyclic loading at two rates. *Materials and Structures*, 20(2), 85–95. 2.1.1

Silva, Christian E.; Gomez, Daniel; Maghareh, Amin; Dyke, Shirley J.; y Spencer, Billie F. (2020). Benchmark control problem for real-time hybrid simulation. *Mechanical Systems and Signal Processing*, 135, 106381. 1, 2.3.1, 4.1, 4.5

Spencer, Jr.; Elnashai, Amr S.; Kwon, Oh-Sung; Park, Kyu-Sik; y Nakata, Narutoshi (2007). The UI-SimCor Hybrid Simulation Framework. *2nd World Forum on Collaborative Research in Earthquake Engineering*, (pp. 9 pages). (document), 2.2, 2.1.3, 2.3, 2.2, 2.6, 2.3.1.1, 2.8

Stavridis, Andreas y Shing, P. B. (2010). Hybrid testing and modeling of a suspended zipper steel frame. *Earthquake Engineering and Structural Dynamics*, 39(2), 187–209. 1

Tada, M; Tamai, H; Ohgami, K; Kuwahara, S; y Horimoto, A (2008). Analytical Simulation Utilizing Collaborative Structural Analysis System. *14th World Conference on Earthquake Engineering (14WCEE)*, (X), 1–8. 2.2.4.2

Takahashi, Yoshikazu y Fenves, Gregory L. (2006). Software framework for distributed experimental-computational simulation of structural systems. *Earthquake Engineering and Structural Dynamics*, 35(3), 267–291. 1, 2.2, 2.2.6, 2.8

Takanashi, Koichi; Udagawa, Kuniaki; Seki, Masutaro; Okada, Tsuneo; y Tanaka, Hisashi (1975). Non-linear Earthquake Response Analysis of Structures by a Computer-Actuator On-Line System. *Bull. of Earthquake Resistant Struc. Res. Ctr.*, 8(March 1975), 17. 2.1.3

Tena Colunga, Arturo (2007). *Analisis De Estructuras Con Metodos Matriciales*. Mexico: Editorial Limusa. 2.1.3.1

Van Rossum, Guido y Drake, Fred L (2012). Python Tutorial. *Python Software Foundation*, (pp. 1–136). 3.3

Vecchio, F. J. (2000). Disturbed Stress Field Model for Reinforced Concrete: Formulation. 2.2.2.2

Vecchio, F. J. (2001). Disturbed Stress Field Model for Reinforced Concrete: Implementation. 2.2.2.2

Vecchio, F. J.; Collins, M.P.; y Members, ASCE (1993). Compression Response of Cracked Reinforced Concrete. *Structural Engineering and Mechanics*, 119(12). 2.2.2.2

Vuong, A.-V.; Heinrich, Ch.; y Simeon, B. (2010). Isogat: A 2d tutorial matlab code for isogeometric analysis. *Computer Aided Geometric Design*, 27(8), 644–655. Advances in Applied Geometry. 2.7.5

Waldbjoern, Jacob P.; Maghareh, Amin; Ou, Ge; Dyke, Shirley J.; y Stang, Henrik (2021). Multi-rate Real Time Hybrid Simulation operated on a flexible LabVIEW real-time platform. *Engineering Structures*, 239(April 2020). 3.2

Wallace, M I; Wagg, D J; y Neild, S A (2005). An adaptive polynomial based forward prediction algorithm for multi-actuator real-time dynamic substructuring. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 461(2064), 3807–3826. 3.2

Wang, Chi-hsiang; Foliente, Greg C.; Sivaselvan, Mettupalayam V.; y Reinhorn, Andrei M. (2001). Hysteretic Models for Deteriorating Inelastic Structures. *Journal of Engineering Mechanics*, 127(11), 1200–1202. 4.7.2

Wang, Kung Juin; Tsai, Keh Chyuan; Wang, Shiang Jung; Cheng, Wei Choung; y Yang, Yuan Sen (2007). ISEE: Internet-based simulation for earthquake engineering- Part II: The application protocol approach. *Earthquake Engineering and Structural Dynamics*, 36(15), 2307–2323. 2.2, 2.8

Wang, Zhen; Yan, Xueqi; Ning, Xizhan; y Wu, Bin (2020). Test Verification of Two-stage Adaptive Delay Compensation Method for Real-time Hybrid Simulation. *Unknown yet*, (pp. 1–17). 2.4.1, 2.4.1.1.2

Xu, Dan; Zhou, Huimeng; Shao, Xiaoyun; y Wang, Tao (2019). Performance study of sliding mode controller with improved adaptive polynomial-based forward prediction. *Mechanical Systems and Signal Processing*, 133, 106263. 2.4.1

Xu, Zhao Dong; Dong, Yao Rong; Chen, Shi; Guo, Ying Qing; Li, Qiang Qiang; y Xu, Ye Shou (2021). Development of hybrid test system for three-dimensional viscoelastic damping frame structures based on Matlab-OpenSees combined programming. *Soil Dynamics and Earthquake Engineering*, 144(February), 106681. (document), 2.2, 2.2.5, 2.11, 2.8

Yang, Yuan Sen; Hsieh, Shang-Hsien; Tsai, Keh Chyuan; Wang, Shiang Jung; Wang, Kung Juin; Cheng, Wei Choung; y Hsu, Chuan-Wen (2007). ISEE: Internet-based Simulation for Earthquake Engineering—Part I: Database approach. *Earthquake Engineering and Structural Dynamics*, (36), 2291–2306. 2.2, 2.8

Zhan, Hu y Kwon, Oh-Sung (2015). Actuator controller interface program for pseudo- dynamic hybrid simulation. *2015 World Congress on Advances in Structural Engineering and Mechanics*, (pp. 1–13). 2.2.2.3

Zhang, Yu; Pan, Peng; Gu, Quan; Yang, Jun; y Deng, Kailai (2014). Development of Collaborative Structure Analysis (CSA) system and its application to investigate effects of soil-structure interaction. *Journal of Earthquake Engineering*, 18(7), 1151–1169. (document), 1, 2.2, 2.2.4, 2.9, 2.10, 2.8

Zhang, Yunfeng; Sause, Richard; Ricles, James M.; y Naito, Clay J. (2005). Modified predictor-corrector numerical scheme for real-time pseudo dynamic tests using state-space formulation. 2.1.3

Zhao, J.; French, C.; Shield, C.; y Posbergh, T. (2003). Considerations for the development of real-time dynamic testing using servo-hydraulic actuation. *Earthquake Engineering and Structural Dynamics*, 32(11), 1773–1794. 2.4.1

Zhu, Minjie; McKenna, Frank; y Scott, Michael H. (2018). OpenSeesPy: Python library for the OpenSees finite element framework. *SoftwareX*, 7, 6–11. 1, 3.3

A | Apéndice

El código fuente utilizado en esta tesis se encuentra disponible en el repositorio personal del autor, accesible desde el siguiente link https://github.com/DiegoNHMeraM/ClientServer_OpenSeesPy o desde la referencia Mera y Fermandois (2021)