Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXVII, págs. 1747-1756 (artículo completo)
A. Cardona, L. Garelli, J.M. Gimenez, P.A. Kler, S. Márquez Damián, M.A. Storti (Eds.)
Santa Fe, 5-7 Noviembre 2019

PROGRAMACIÓN ORIENTADA A OBJETOS APLICADA A SIMULACIONES DE FLUJOS DOMINADOS POR VORTICIDAD – PARTE 1: ASPECTOS DE IMPLEMENTACIÓN

OBJECT-ORIENTED PROGRAMMING APPLIED TO VORTICITY DOMINATED FLOWS SIMULATIONS – PART 1: IMPLEMENTATION ISSUES

Martín E. Pérez Segura [a,b], **Alejandro T. Brewer**[b], **Sergio Preidikman** [a,b]

[a]*Instituto de Estudios Avanzados en Ingeniería y Tecnología (IDIT) - CONICET, Universidad Nacional de Córdoba, Av. Vélez Sarsfield 1611, 5000 Córdoba, Argentina. idit@fcefyn.edu.ar, www.inv.idit.efn.uncor.edu.*

[b]*Dpto. de Estructuras, FCEFyN, Universidad Nacional de Córdoba, Av. Vélez Sarsfield 1611, 5000 Córdoba, Argentina. depestructuras@fcecyn.unc.edu.ar, http://www.efn.uncor.edu.ar*

Palabras clave: programación orientada a objetos, flujos dominados por vorticidad.

Resumen. A medida que los avances en la investigación científica e ingeniería impulsan la aparición de modelos cada vez más complejos, que exceden los métodos de análisis tradicionales, la multi-física computacional adquiere un papel más preponderante. Se han impuesto metas más ambiciosas en cuanto a la fidelidad de los modelos, promoviendo la sofisticación de las técnicas de análisis y generando nuevas exigencias para la modelación y para los proyectos de desarrollo de software. En este contexto, el paradigma de programación orientada a objetos constituye una alternativa importante a la hora de desarrollar software de simulación, debido a que facilita la implementación, lectura, mantenimiento, expansibilidad y flexibilidad del código resultante. Esto es, utilizando al "objeto" como elemento fundamental para el análisis y diseño del modelo de software. En consecuencia, es necesario conocer los principios y parámetros del paradigma a los fines de poder aprovechar las ventajas que proporciona por sobre otras alternativas. El presente trabajo proporciona un marco generalizado de simulación fluido-dinámica basado en el método de red de vórtices inestacionario y no lineal, y un ejemplo de desarrollo de software de simulación dentro del paradigma orientado a objeto.

Keywords: Object-Oriented Programming, Vorticity dominated flows.

Abstract. Due to progress in scientific research and engineering, computational multi-physics has developed a significant role as increasingly complex models that exceed traditional approaches have arisen. More ambitious goals have been imposed regarding the fidelity of the computational models, promoting more sophisticated analysis techniques and generating new requirements for modeling and for projects of software development. In this context, the object-oriented programming paradigm represents an important alternative when developing a simulation software, since it eases implementation, reading, maintenance, expandability and flexibility of the resulting code. All this is achieved using the "object" as the fundamental element for the analysis and design of the software model. In consequence, one must know the principles and parameters of the paradigm so as to be able to exploit its advantages over other alternatives. This effort provides a generalized framework for computational simulation of fluid dynamics based on the non-linear unsteady vortex lattice method, and an example of software development within the object-oriented paradigm.

1 INTRODUCCIÓN

Actualmente, la investigación y el desarrollo en el ámbito de la ingeniería y la tecnología han incorporado la simulación computacional como práctica habitual para alcanzar resultados. En este contexto, se han impuesto metas cada vez más ambiciosas en cuanto a la representatividad de los modelos computacionales, obligando la sofisticación de las técnicas de análisis en pos del detalle en la descripción de los fenómenos estudiados. En este sentido, la multi-física es la disciplina computacional que ha permitido la implementación de un enfoque particionado de fenómenos complejos, involucrando diversas áreas de la física, en un proceso de co-simulación basado en la combinación de modelos disímiles altamente especializados.

En lo que concierne a este trabajo, la co-simulación se enmarca en la aeroelasticidad computacional (CAE) como método de análisis de la interacción entre fluido y estructuras (FSI). Conceptualmente, se abordan por separado el problema aerodinámico y el problema estructural-dinámico, con modelos apropiados para cada uno de ellos, requiriendo de la incorporación de un método de interacción entre ambos. Varios fenómenos, tanto numerosos como diversos, podrían considerarse bajo el alcance de CAE, por lo que resulta imperativo poder adaptar fácilmente un software de simulación sin degradar sus características generales. El trabajo aquí descrito se enfoca en la parte aerodinámica del problema FSI y presenta una visión general del proceso de desarrollo de un software de simulación de propósito general basado en el Método de Red Vórtice No-lineal e Inestacionario (UVLM), considerando los conceptos mencionados anteriormente.

Al presente, existen otras herramientas capaces de llevar a cabo simulaciones aerodinámicas y satisfacer las demandas de co-simulación de la CAE y muchas han sido utilizadas en diferentes contextos. Sin embargo, el UVLM proporciona un balance razonable entre aplicabilidad y costo computacional en relación a los métodos de elementos de volumen (VEM) de la dinámica de fluidos computacional (CFD). Por ello, este método ha sido utilizado para el análisis de turbinas eólicas de eje horizontal (Maza, M. S., et al, 2014), aeronaves con alas que cambian de forma (Verstraete, M. L., et al, 2012), aeronaves con alas unidas (Pérez Segura, M. E., et al, 2018), micro vehículos aéreos (Verstraete, M. L., et al, 2015) y modelos de alas batientes (Roccia, B. A., et al, 2013), entre otros.

La construcción de una estructura de simulación computacional para la investigación en ingeniería no debe concebirse de forma aislada, sino como parte de un entorno de desarrollo e íntimamente relacionada con el lenguaje y el estilo de programación. A pesar de la gran difusión que la programación por procedimientos ha alcanzado, en muchos casos es necesario adoptar un enfoque más apropiado, especialmente para los grupos de trabajo de investigación en ingeniería. Luego, definir cuidadosamente el alcance de la implementación y el paradigma de programación se vuelve esencial. Entre otras posibilidades que han demostrado ser adecuadas, este esfuerzo adopta el Paradigma de Programación Orientada a Objetos (OOP) para establecer los lineamientos para la implementación de ULVM en el campo de co-simulación, a fin de aprovechar la confiabilidad y la versatilidad de un código modular. Este enfoque permite fructificar la posibilidad de cambiar de una aplicación a otra sin que dicho cambio conlleve un proceso de reprogramación significativo.

Como características adicionales para el software desarrollado, se acentúa la robustez y la generalidad, incluyendo mecanismos de pre-proceso que garantizan la correcta ejecución del método, reduciendo las exigencias y restricciones impuestas al usuario en cuanto a la definición de parámetros de simulación. Así también, con la adaptabilidad como premisa, se reduce al mínimo la información a proporcionar como datos de entrada, a los fines de admitir la construcción de simulaciones con independencia respecto de las características de los mismos. Igualmente, se permite desestructurar los datos de salida dando mutabilidad para generar

visualizaciones, pos-procesamientos, nuevas iteraciones, etcétera. En cuanto al desempeño, se prioriza la reducción de tiempos de ejecución implementando técnicas de Computación de Alto Desempeño (HPC) y aprovechando las posibilidades de ejecución en paralelo. Todo el desarrollo se realiza utilizando FORTRAN 2008 como lenguaje de programación, siguiendo con la tendencia del contexto de trabajo y debido a su versatilidad y condiciones tanto para la OOP como para técnicas de HPC (Pérez Segura, M. E., et al, 2018). No obstante, todas las conclusiones son independientes del lenguaje utilizado y, por ende, extrapolables a otros, siempre que soporten el paradigma de la OOP.

2 PARADIGMAS DE PROGRAMACIÓN

La implementación computacional de un modelo fluido-dinámico, al igual que con cualquier modelo físico de simulación, requiere un exhaustivo proceso de análisis y diseño previo a la implementación propiamente dicha. Aquí intervienen diversos niveles de abstracción que se concatenan progresivamente desde el fenómeno físico que se pretende representar hasta el software desarrollado como herramienta de simulación. Este proceso define lo que se conoce como Paradigma de Programación. Es decir, representa un enfoque particular que define los conceptos, los sistemas de abstracción y los procedimientos que integran la solución del problema en análisis, en el marco del desarrollo computacional. Van Roy (2012) define a los paradigmas de programación como métodos basados en una teoría matemática o en un conjunto coherente de principios que soporta una serie de conceptos, que los hacen apropiados para un determinado tipo de problema. La selección de un paradigma que enmarque un desarrollo de software implica desvincular la implementación de sus atributos computacionales. Esto es, retroceder hasta el origen del problema para reestructurar su concepción con métodos alternativos que sean más ventajosos, incluso antes de considerar un lenguaje o una plataforma para la implementación.

Un diagrama representativo de estos conceptos se muestra en la Figura 1. En términos generales, se puede concebir un software como la combinación de dos partes: una estructura de datos y un algoritmo que opera sobre ésta. El paradigma adoptado para el desarrollo guarda una estrecha relación con ambas, determina cómo se construye cada una y cómo se vinculan entre sí. Evidentemente, el contexto y la funcionalidad del código en desarrollo definen la factibilidad de adoptar un determinado paradigma, al igual que el lenguaje de programación a utilizar. Mientras que, la aceptación y el uso dentro de la comunidad de investigación también resultan influyentes. De ahí la importancia de adoptar un paradigma para estructurar y unificar los desarrollos. Existen numerosos paradigmas, alrededor de treinta de acuerdo con Van Roy (2012), que han surgido y mutado en función de los cambiantes requerimientos en el desarrollo de softwares, y pueden asociarse en dos grandes grupos. Por un lado, aquellos relacionados con la programación imperativa y, por otro, los relacionados con la programación declarativa.

De entre los paradigmas asociados a la programación imperativa, el paradigma de programación por procedimientos (PPP) es el más reconocido en el campo de la ingeniería y su reputación se basa en su demostrada capacidad para satisfacer las demandas de los desarrolladores por décadas, así como también su proximidad conceptual al lenguaje nativo de los procesadores. Sin embargo, el PPP presenta ciertas desventajas referidas a la cantidad de recursos que se consumen en la propagación en los cambios de estado a través de los procedimientos involucrados, la criticidad del almacenamiento de dichos estados y la gran dificultad de concebir abstracciones como procedimientos (Aguilar, L. J., 1996). En resumen, el PPP permite optimizar procedimientos (o algoritmos) pero muestra ciertas debilidades a la hora de construir estructuras de datos, limitando la intercambiabilidad de código.

Luego de su introducción en la década de 1960, la OOP se ha convertido en una alternativa significativa para el desarrollo de software de simulación en el campo de la investigación en

ingeniería. Booch, G. (2017) lo define como un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representan una instancia de alguna clase, y cuyas clases son todas miembros de una jerarquía de clases unidas mediante relaciones de herencia. Luego, la OOP permite la representación de los fenómenos de un modo simple y natural, reduciendo y simplificando el proceso de abstracción que comprende la transición entre las especificaciones en términos del problema a los requerimientos en términos del código. Este estilo satisface una necesidad real del desarrollo de software: brinda la posibilidad de manipular confiablemente diversos tamaños de problemas, proporcionando flexibilidad, mantenibilidad, y la capacidad de mutar cuando se producen cambios en los requerimientos.

Finalmente, dos aspectos conceptuales significativos intervienen en el desarrollo de una aplicación de simulación y ambos deben ser adecuadamente investigados. El primero de ellos está constituido por los principios y elementos del paradigma de guía y, el segundo, comprende las bases de los fenómenos bajo análisis y del modelo utilizado para su representación.

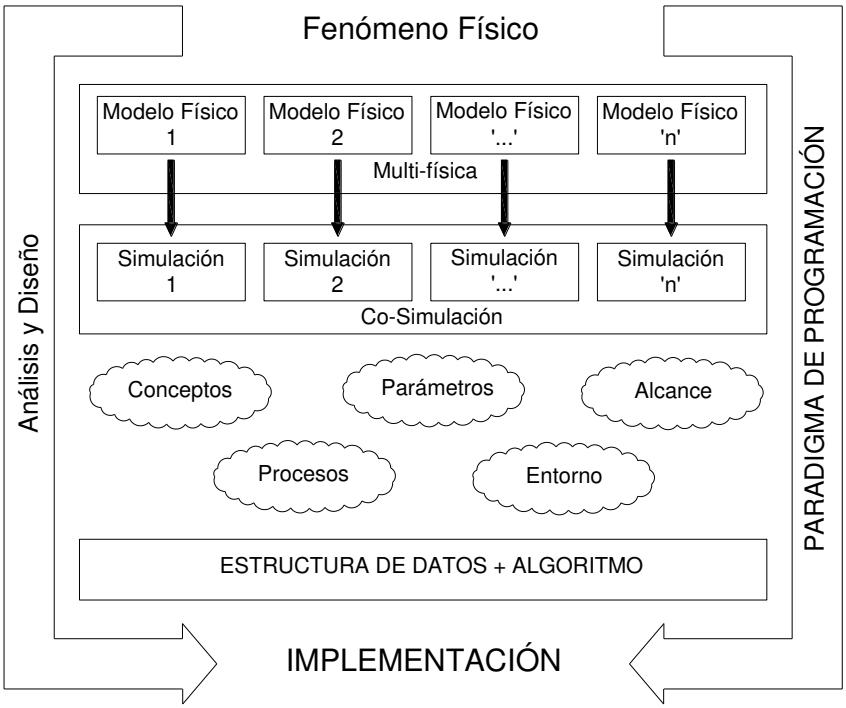

Figura 1: Aspectos de diseño y análisis de un paradigma de programación.

3 EL MODELO AERODINÁMICO: UVLM

El modelo aerodinámico detrás del ULVM, como la mayoría de los modelos físico-matemáticos para dinámica de fluidos, se construye a partir de las ecuaciones de Navier-Stokes, a las que luego se aplican ciertas hipótesis simplificativas. En primer lugar, el fluido es considerado incompresible lo que implica que la densidad permanece constante y garantiza un error en su estimación menor al 5% para números de Mach menores a 0.3 (Anderson, J. D. Jr, 2012). La segunda hipótesis consiste en definir al fluido como no viscoso, es decir, despreciar cualquier fenómeno de transporte molecular. Esta hipótesis es, en términos del número de Reynolds (Re), una solución asintótica a medida que $Re \to \infty$.

En este punto, y de acuerdo a la formulación matemática del ULVM, se introduce el campo de vorticidad. Luego, ambos campos, el de velocidad y el de vorticidad, coexisten como funciones del tiempo y del espacio relacionadas por una ley cinemática.

En general, para un fluido incompresible (o estrictamente para uno barotrópico), donde se desprecian los fenómenos de difusión y las fuerzas externas son conservativas, se satisfacen las leyes de Helmholtz. Esto significa que la vorticidad no puede crearse en el seno del fluido y que las distribuciones de vorticidad compactas, permanecen compactas. Por lo tanto, la vorticidad se origina sobre superficies sólidas debido a la condición de no deslizamiento, construyendo capas límites, y luego se transfiere al flujo formando estelas.

Bajo estas condiciones, la estructura y evolución del flujo se describe más convenientemente en términos del campo de vorticidad que en términos del campo de velocidades. Además, estas definiciones permiten analizar flujos con moderados a elevados (pero finitos) números de Reynolds. Maza, M. S. (2013) postula que a medida que el número de Reynolds se incrementa, la distribución de vorticidad es suficientemente compacta como para considerar dos zonas bien definidas en el dominio fluido: i) una porción pequeña que incluye el flujo rotacional (capas límites y estelas) con vorticidad no nula; y ii) la parte restante del dominio que se asume irrotacional.

La velocidad en cualquier punto del dominio fluido es la suma de tres componentes: la velocidad de la corriente libre, la velocidad producida por las capas límites y la velocidad asociada a las estelas. Estas dos últimas componentes se calculan en términos del campo de vorticidad utilizando la ley de Biot-Savart (Preidikman, S., 1998). El modelo se cierra al aplicar dos condiciones de contorno de acuerdo a las hipótesis planteadas: la condición de regularidad en el infinito y la condición de no penetración sobre superficies sólidas.

Desde el punto de vista del modelo numérico, las sábanas vorticosas (capas límites y estelas) se discretizan como una red de segmentos vorticosos de circulación constante. Las capas límites se identifican como “sábanas adheridas” debido a que se encuentran dinámicamente unidas a las superficies sólidas y sus segmentos se encuentran agrupados en anillos (paneles) con puntos de control y versores normales para el cálculo de cargas aerodinámicas.

La incógnita del problema es entonces la circulación de los segmentos y su determinación depende de la evaluación de las condiciones de contorno. Para ello, la condición de no penetración se impone en los puntos de control de los paneles de las sábanas adheridas que, en este punto, se denominan grillas aerodinámicas (GA). Así, se obtiene un sistema de ecuaciones algebraicas lineales en las circulaciones de los segmentos. Siguiendo a Preidikman, S. (1998), la dimensión del sistema puede reducirse sensiblemente si se definen circulaciones de anillo, asociadas a cada panel.

El tratamiento de las estelas es un poco diferente y conceptualmente responde a las leyes de Helmholtz. La circulación que se transfiere al flujo debe permanecer inalterada y moverse con la velocidad local del fluido como partículas materiales. Por lo tanto, en cada paso de tiempo la circulación de los segmentos es convectada desde los bordes de fuga de las superficies al fluido, formando la estela como una grilla de nuevos segmentos que se mueven con el fluido y conservan su circulación.

Hasta aquí el análisis determina la cinemática del modelo, por lo que el paso siguiente es determinar las fuerzas actuantes en las superficies inmersas en el fluido. Para ello, se utiliza la ecuación de Bernoulli inestacionaria que relaciona los campos cinemáticos (vorticidad y velocidad) con el salto de presión a través de las superficies (Preidikman, S., 1998).

4 IMPLEMENTACIÓN ORIENTADA A OBJETOS

El UVLM involucra entidades bien definidas para representar los fenómenos fluido-dinámicos, por lo que su enfoque naturalmente se identifica con el modelo orientado a objetos, de acuerdo a lo presentado en la Figura 2. En primer lugar, las GA se conforman por anillos rectilíneos, identificados como paneles, cada uno con un punto de control en su centro, un versor normal que le da orientación y un valor de vorticidad de anillo. Adicionalmente, cada anillo se

compone de segmentos vorticosos que conectan dos nodos y son definidos por su circulación. En segundo lugar, las GA convectan vorticidad al flujo originando estelas a partir de líneas de convección previamente definidas. Las estelas se forman por los mismos tipos de nodos y segmentos vorticosos que las GA, aunque, en esta versión del UVLM, no poseen anillos o paneles. Por otro lado, las superficies no sustentadoras se definen como grillas de contorno (GC). Éstas se componen exactamente igual a las AG, pero no generan estelas y sólo proveen una estructura para aplicar la condición de no penetración. Finalmente, un grupo de puntos extra se define en el dominio fluido para determinar la velocidad local si fuera necesario.

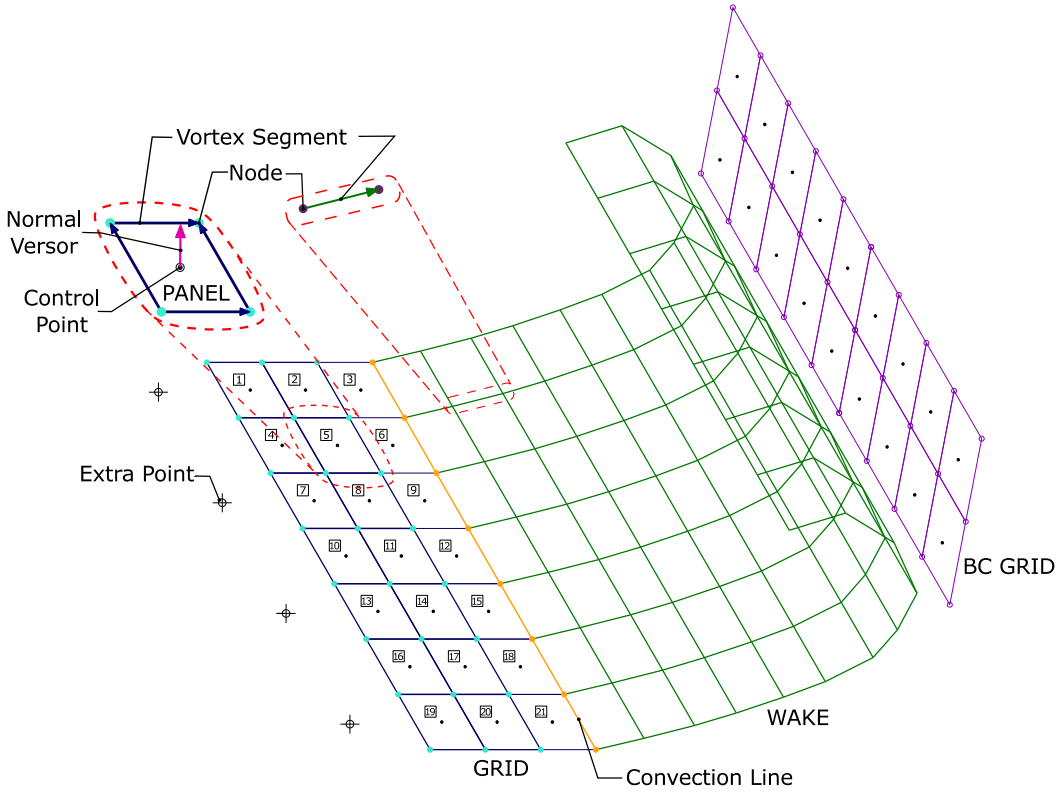

Figura 2: Entidades del modelo aerodinámico.

Con la descripción anterior, es posible interpretar a grandes rasgos cómo se construye el modelo orientado a objetos. El primer paso es el proceso de abstracción que consiste en representar las características internas y externas de cada entidad para describirla desde el punto de vista de la implementación. En la OOP, este proceso se materializa en la construcción de "clases" y "objetos" como instancias de las clases.

El segundo paso es encapsular los atributos internos de una clase, para separarlos de aquellos que se encuentran disponibles para el resto de ellas: los externos. La encapsulación es probablemente el aspecto más importante de la OOP y consiste en incluir todos los recursos necesarios para la funcionalidad de un objeto dentro de sí mismo. Igualmente, garantiza que dichos recursos se encuentren "ocultos" para el resto de los objetos y puedan ser accedidos sólo a través de procedimientos de vinculación. En términos del código, la encapsulación implica modularidad.

Una vez definidas las clases deben relacionarse entre sí. Estas relaciones ordenan las abstracciones dentro del diseño del modelo y son principalmente de dos tipos: i) generalización/especialización (también llamadas relaciones "es un"); ii) agregación (también llamadas relaciones "tiene"). En el primer tipo de relación, una nueva clase se deriva de una

existente por lo que, la segunda posee todas las características, atributos y procedimientos heredados de la clase preexistente, además de propiedades exclusivas que puedan añadirse. En contraste, la relación de agregación permite incluir un objeto como una propiedad de otro, es decir, el objeto de una clase se define como un atributo de una clase diferente.

Al aplicar los conceptos previos, es posible representar cada entidad del modelo aerodinámico definiendo una clase y generando tantos objetos de ella como sea necesario. Luego, el modelo se construye de forma descendente desde las entidades más inclusivas o generales hacia las más exclusivas o particulares, aprovechando así los conceptos de herencia y agregación. Al respecto, la Figura 3 presenta un *diagrama de clases* a través del *Lenguaje Unificado de Modelado* (UML) (Fowler. M., Kendall, S., 1999), exponiendo todas la estructura del código desarrollado.

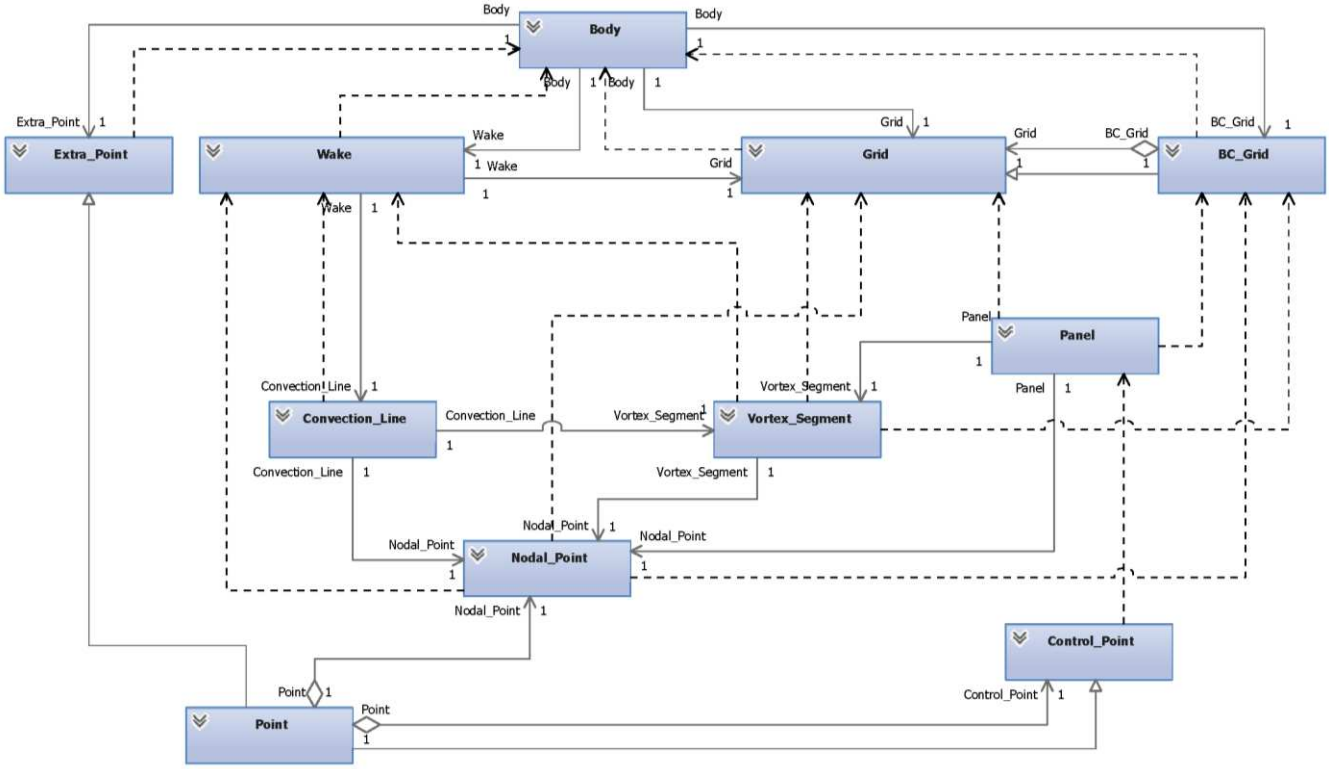

Figura 3: Diagrama de Clases (UML).

La implementación particular tratada en este trabajo organiza al modelo, en su nivel más alto, como un set de cuerpos (“bodies”). Cada cuerpo agrupa todas las entidades que de alguna manera se relacionan entre sí en el modelo físico. Así, un cuerpo se compone de las grillas, las estelas y los puntos extra (si los hubiera) que componen una porción específica del modelo. Por ejemplo, a la hora de simular una aeronave convencional un cuerpo podría instaurarse como “semi-ala izquierda”, éste contará con una GA, una estela, y puntos extra, y será distinto al cuerpo “semi-ala derecha” o “grupo de cola”. Luego, la interacción entre grillas, estelas y puntos extra se realiza a través de los cuerpos, facilitando la representación de interacciones multi-cuerpo.

Las GA se ubican en el siguiente nivel de organización, son parte de un cuerpo y se construyen con paneles (anillos vorticosos). Al mismo tiempo, cada grilla posee información adicional que no es requerida por sus paneles y, aunque puede ser accedida por otros objetos, los permisos de modificación son restringidos. Consiguientemente, una grilla puede mantener control de sus propios paneles y sus subcomponentes, y transferir sólo información específica a otros objetos.

Haciendo uso del concepto de herencia, la clase GA se extiende a una segunda clase que representa las GC. Esta nueva clase, comparte todos los aspectos (atributos y procedimientos) de su predecesora, dado que se construyen de forma similar, e incluye características propias como la velocidad de transpiración.

En el mismo nivel de organización que las GA, se define la clase estela ("wake"). Las estelas se conforman, como se mencionó, con un set de segmentos vorticosos y una línea de convección a la cual están asociadas. Así, una estela controla la posición de sus propios segmentos y añade a su estructura aquellos nuevos que sean convectados y, al mismo tiempo, provee datos para el cálculo de velocidades inducidas a otros objetos. Sin embargo, el aspecto más importante de la clase estela son sus procedimientos; en particular aquel destinado al proceso de convección donde se ejecuta la generación, movimiento y depreciación de los segmentos vorticosos que la forman. Por ende, la siguiente clase a mencionar son las líneas de convección que, como componentes de las estelas y entre otras características, definen el set de nodos desde donde se efectuará la convección.

Los componentes de las clases citadas anteriormente se determinan en un subnivel, estos son paneles, segmentos vorticosos y puntos. En primera instancia se define la clase abstracta (Aguilar, L. J., 1996) punto ("points") que se utiliza para derivar clases sucesivas. Este es el caso de los nodos ("nodes"), puntos de control ("control points") y los puntos extra ("extra points"), ya que todos ellos poseen los atributos de la clase punto y algunas características exclusivas agregadas. Por otro lado, los segmentos vorticosos poseen dos nodos en relación de agregación debido a la forma en la que se construyen y su principal característica es la circulación. Finalmente, los paneles se forman por un set de segmentos vorticosos, un punto de control y un versor normal, poseen una vorticidad de anillo y representan el bloque elemental para construir GA y GC:

5 VERIFICACIÓN DEL CÓDIGO

En esta sección se presenta un grupo de resultados del programa de computadora descrito, con el objetivo de exponer un breve proceso de verificación. En este sentido, se analiza el caso del arranque impulsivo de una placa plana que se discute ampliamente en la bibliografía especializada.

En esta simulación, un ala es puesta impulsivamente en movimiento rectilíneo y se calcula su coeficiente de sustentación (C_L) como una función del tiempo (t). En particular, el ala se representa como una placa plana rectangular y se analizan dos valores de alargamiento (AR). Además, la simulación se caracteriza por el parámetro adimensional $V_\infty \Delta t/c = 1/16$ y una grilla con cuatro paneles en la cuerda y trece paneles en la semi-envergadura. Los resultados se muestran en la Figura 4 a), donde la variable temporal, t^*, se encuentra adimensionalizada como $t^* = V_\infty \Delta t/c$, y se incluyen conjuntamente con los resultados de Katz, J. y Plotkin, 2001.

Un segundo ejemplo de la misma simulación se lleva a cabo para un tercer valor de alargamiento y se muestra en la Figura 4 b). En este caso, el coeficiente de sustentación se presenta en forma adimensional $C_L^* = C_L(t)/C_L(t = \infty)$ para capturar más significativamente el efecto transitorio.

Estos simples casos permiten identificar dos comportamientos que se superponen para conformar la curva de resultado. Inicialmente, una porción inestacionaria generada por el movimiento impulsivo gobierna la curva de C_L. Sin embargo, este transitorio decae rápidamente dejando la solución de estado estacionario a medida que el tiempo crece.

5.1 Medidas de performance

Haciendo énfasis en la incorporación de las directivas de HPC mencionadas, se implementa

la librería "*mkl_intel_threds*", que utiliza el modelo de ejecución de la API "*p-threads*", para la paralelización de hilos en núcleos múltiples (memoria compartida). Mientras que, para la resolución de sistemas de ecuaciones algebraicas se incorpora la subrutina *DGESV* de la librería *LAPACK*. Luego, se exponen en la Tabla 1 valores de referencia elementales medidos para ejecuciones de referencia (placa plana en arranque impulsivo), que alcanzan incrementos de velocidad promedio de 3.88X, respecto de las ejecuciones en un único núcleo.

Pasos de Tiempo	Nº Paneles		Tiempo de Ejecución		Eficiencia de Paralelismo Media
	Grilla	Estela	Media	Desviación	
60	3560	8940	172.3 (seg.)	9.2 (seg.)	61%

Tabla 1: Datos de ejecución. *Hardware: Intel Core i7-7700HQ (4-Core HT, Kaby Lake).*

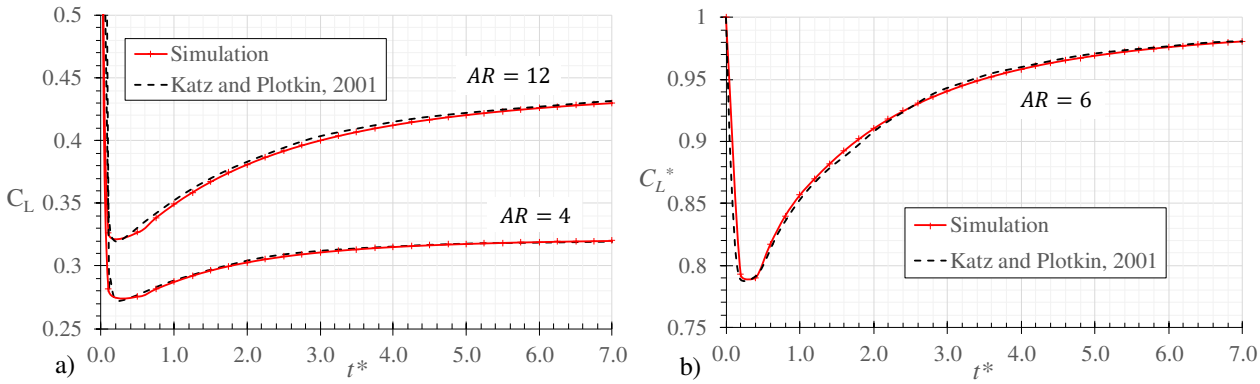

Figura 4: Coeficiente de sustentación para el arranque impulsivo de una placa plana.

6 COMENTARIOS FINALES

A lo largo del presente esfuerzo se construyó una herramienta computacional de simulación fluido-dinámica enmarcada en el paradigma de la OOP, concebida para el uso eficiente en un grupo de investigación ingenieril, adaptada para el acoplamiento con otros códigos de simulación, y haciendo énfasis en la performance y la validación. En este proceso se exploró una alternativa a la clásica programación procedural, procurando vencer las deficiencias que este paradigma impone para el desarrollo colaborativo y la reutilización de código en grupos de investigación. En general, estos conceptos son extrapolables a cualquier fenómeno físico que pueda encasillarse en la co-simulación, bajo condiciones similares a las expuestas.

El resultado general es una implementación de UVLM que permite evaluar las ventajas de la OOP, las cuales se hacen evidentes a la hora de incorporar nuevas características al código, modificar funcionalidades y agregar elementos de simulación. Asimismo, se reducen los tiempos de ejecución entre un 30% y 60% respecto a otros códigos similares implementados en el grupo de trabajo, los cuales no solo se especializan en una aplicación particular, sino que carecen de la posibilidad de flexibilizarse para extender el análisis. En particular, se reduce sensiblemente el tiempo destinado al pre-procesamiento y acondicionamiento de casos para la simulación, lo que representa un avance considerable en el ámbito de desarrollo. Esto es, alternar entre casos de estudio se reduce a un proceso de definición de geometría y mallado, como se muestra en la Segunda Parte de este trabajo (Pérez Segura, M. E., et al, 2019).

Por otro lado, el presente esfuerzo representa un punto de partida tanto para desarrollos en el área de la FSI como en aquellas relacionadas al desarrollo de herramientas computacionales, en cuanto al análisis de paradigmas. De aquí se desprenden varias líneas de trabajo que pueden seguirse, entre ellas se destacan: i) la implementación de variantes al UVLM, como estelas de partículas, densificación localizada de grillas, convección de paneles triangulares, convección

condicionada, etc.; ii) evaluar otras estructuras de datos que coloquen otros elementos (por ejemplo, paneles) como núcleos de la definición de clases; iii) analizar el balance entre el nivel de parametrización del código y su performance; iv) considerar la utilización de otros lenguajes de programación.

En resumen, el resultado obtenido es alentador y cumplimenta los objetivos propuestos, los cuales se materializan en una implementación del UVLM diseñada y construida desde el punto de vista del desarrollo colaborativo de software de simulación.

REFERENCIAS

Maza, M. S., Preidikman, S. and Flores, F. G. Unsteady and non-linear aeroelastic analysis of large horizontal-axis wind turbine. *International Journal of Hydrogen Energy*, 39(16), pp. 8813–8820. 2014.

Verstraete, M. L., Preidikman, S., Ceballos, L. R., y Massa, J. C., Un modelo estructural no-lineal de alas flexibles para vehículos aéreos no-tripulados con alas que mutan. *Mecánica Computacional*, Vol. XXXI, pp. 2657-2670, 2012.

Pérez Segura, M. E., Preidikman, S., y Beltramo, E., Análisis de las Características Aerodinámicas de Aeronaves con Configuración de Alas Unidas en Función de sus Parámetros Topológicos. *Mecánica Computacional*, Vol. XXXVI No 42, Multiphysics, pp. 1949-1958. 2018.

Verstraete, M. L., Preidikman, S., Roccia, B. A., and Mook, D. T. A New Numerical Model to Study the Nonlinear and Unsteady Aerodynamics of Bioinspired Morphing-Wing Concepts. *International Journal of Micro Air Vehicles*, 7(3), pp. 327–345. 2015.

Pérez Segura, M. E., Maza, M. S., Preidikman, S. Implementación Computacional del Método de Red de Vórtices Inestacionario: Una Versión Basada en los Paradigmas de Programación Orientada a Objetos y Co-Simulación. MSC Thesis, Facultad de Ciencias Exactas Físicas y Naturales, Universidad Nacional de Córdoba. Córdoba, Argentina. 2018.

Roccia, B. A., Preidikman, S., Massa, J. C., and Mook, D. T. A modified 3-D unsteady vortex-lattice method to model the aerodynamics of flapping wings in hover flight. *AIAA Journal*, 51(1), pp. 2628–2642. 2013.

Van Roy, P. Programming Paradigms for Dummies: What Every Programmer Should Know. *Artículo. Université Catholique de Louvain*. Francia, 2012

Aguilar, L. J., *Programación Orientada a Objetos*. McGraw Hill. España, 1996.

Booch, G., *Object-Oriented Analysis and Design with Application*, Third Edition. Addison-Wesley. 2007.

Anderson, J. D. Jr. *Fundamentals of Aerodynamics, Fifth Edition*. McGraw Hill. USA, 2010.

Maza, M. S., *Desarrollo de Herramientas Numéricas para la Simulación de la Interacción de Estructuras con un Fluido a Elevado Número de Reynolds*. Tesis de MSC, Facultad de Ingeniería, Universidad Nacional de Río Cuarto, 2013.

Preidikman S., *Numerical Simulations of Interactions Among Aerodynamics, Structural Dynamics, and Control Systems*. PhD thesis, Department of Engineering Science and Mechanics, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, 1998.

Fowler. M., Kendall, S., UML Distilled, Addison Wesley Longman, USA, 1997.

Katz, J. y Plotkin, A. *Low-Speed Arodynamics, Second Edition*. Cambridge University Press, USA, 2001.

Pérez Segura, M. E., Brewer, A. T., Preidikman, S. Programación Orientada a Objetos Aplicada a Simulaciones de Flujos Dominados por Vorticidad – Parte 2: Ejemplos de Aplicación. Trabajo aceptado para publicación. Congreso ENIEF 2019. Santa Fé, Argentina.