TESIS CARRERA DE MAESTRÍA EN INGENIERÍA

ACOPLAMIENTO MULTIESCALA EN CÁLCULOS FLUIDODINÁMICOS

Ing. Federico Agustín Caccia
Maestrando

Dr. Enzo Dari
Director

Miembros del Jurado
Dr.F. Teruel (Instituto Balseiro, Universidad Nacional de Cuyo)
Dr.P. Zanocco (Instituto Balseiro, Universidad Nacional de Cuyo)

13 de Julio de 2017

Departamento de Mecánica Computacional – Centro Atómico Bariloche

Instituto Balseiro
Universidad Nacional de Cuyo
Comisión Nacional de Energía Atómica
Argentina

Índice de símbolos

CFD: Fluidodinámica Computacional (*Computational Fluid Dynamics*)

DNS: Simulación numérica directa (*Direct Numerical Simulation*)

FEM: Método de Elementos Finitos (*Finite Element Method*)

GPL: Licencia Pública General (*Public General License*)

MIMD: Múltiples Instrucciones, Múltiples Datos (*Multiple Instruction, Multiple Data*)

MPI: Interfaz de Paso de Mensajes (*Message Passing Interface*)

PDE: Ecuación con Derivadas Parciales (*Partial Differential Equation*)

RANS: Promedio de Reynolds de Navier-Stokes (*Reynolds-Averaged Navier–Stokes*)

SISD: Una Instrucción, Un Dato (*Single Instruction, Single Data*)

SIMD: Una Instrucción, Múltiples Datos (*Single Instruction, Multiple Data*)

SSP: Segundo Sistema de Parada

Índice de contenidos

Índice de figuras

Índice de tablas

Resumen

Los análisis de ingeniería actuales exigen estudios en sistemas cada vez más complejos. Éstos, a su vez, involucran subsistemas de características disímiles: principalmente diferentes tamaños y parámetros característicos. Por ejemplo, en los sistemas termohidráulicos es posible identificar distintos regímenes de flujo en tanques o en cañerías. En ciertas ocasiones solo es de interés el detalle en algunos componentes, necesitando modelar el resto del sistema para conservar la dinámica global. En este trabajo se estudia una técnica que permite acoplar el modelado detallado de sistemas fluídicos bi- y tri- dimensionales con sistemas fluídicos más sencillos uni-dimensionales o cero-dimensionales. Cada subsistema se halla acoplado a los demás mediante los valores que toman las variables en las interfaces que comparten entre sí. El problema a resolver se reduce entonces a un sistema de ecuaciones cuyo tamaño depende de la cantidad de incógnitas en cada interfaz. Estas ecuaciones dependen, a su vez, de la física de cada subsistema y en general resultan ser no lineales. Debido a esta característica, se investigan diferentes métodos de resolución iterativa. Sobre el final del trabajo se extiende la técnica a acoples multifísicos y se muestran algunos ejemplos de acoplamiento neutrónico-termohidráulico.

Palabras clave: ACOPLAMIENTO FUERTE, MODELADO MULTIESCALA, FLUIDODINÁMICA COMPUTACIONAL, MÉTODO DE ELEMENTOS FINITOS, ACOPLAMIENTO NEUTRÓNICO-TERMOHIDÁULICO.

Abstract

Current engineering analyzes require studies in increasingly complex systems. These, in turn, involve subsystems of dissimilar characteristics: mainly different sizes and characteristic parameters. For example, in thermohydraulic systems it is possible to identify different flow regimes in tanks or pipelines. On some occasions it is only interesting to detail in some components, needing to model the rest of the system to preserve the global dynamics. In this work we study a technique that allows the coupling of the detailed modeling of bi- and three-dimensional fluidic systems with simplified one-dimensional or zero-dimensional fluidic systems. Each subsystem is coupled to the others by the values that the variables take on the interfaces they share with each other. The problem to be solved is then reduced to a system of equations whose size depends on the number of unknowns in each interface. These equations, in turn, depend on the physics of each subsystem and in general turn out to be non-linear. Due to this characteristic, different iterative resolution methods are investigated. On the end of the work the technique is extended to multiphysical couplings and some examples of neutron-thermohydraulic coupling are shown.

Keywords: STRONG COUPLING, MULTISCALE MODEL, COMPUTATIONAL FLUID DYNAMICS, FINITE ELEMENT METHOD, NEUTRONIC-TERMAL-HYDRAULIC COUPLINGS

Capítulo 1

Introducción

"No se involucre en problemas parciales, siempre tome vuelo hacia donde hay una vista libre sobre el gran problema único, incluso cuando esta visión todavía no sea clara."

— Ludwig Wittgenstein, 1889-1951

1.1. Motivación

La creciente sofisticación en los análisis de ingeniería demanda el estudio de sistemas cada vez más complejos. Un ejemplo actual de esto es el modelado de grandes componentes termohidráulicos de geometría muy compleja en la industria nuclear. Es notable la presencia de subsistemas de caracterísiticas muy diferentes: principalmente diferentes tamaños y regímenes de flujos. Si bien se necesita modelar y entender el sistema completo, solo es de interés el detalle en algunos subsistemas. Algunos, como las tuberías, se hallan muy bien caracterizados por modelos simple (ODE's). Otros, en cambio, requieren un análisis detallado de flujo, y por ello es necesaria la simulación fluidodinámica computacional (CFD).

En este marco se justifica el desarrollo de una técnica numérica que permita desglosar el problema general para analizar cada subsistema por separado mediante condiciones de borde dinámicas. Como referencia a este enfoque se citan los trabajos desarrollados por J. S. Leiva y G. C. Buscaglia (2006) [1], P.J. Blanco et al. (2010) [2] y J. S. Leiva et al. (2011) [3].

1.2. Abordaje del modelado

1.2.1. Desglosado del sistema original en subsistemas acoplados

Dado un sistema S en un dominio Ω con borde Γ, es posible desglosar este dominio en N particiones y analizar diferentes subsistemas $S_i, i = 1, ..., N$ por separado, acoplados entre sí mediante condiciones de borde en las uniones (método de descomposición disjunta de dominios [4]). Las condiciones de borde originales del problema, impuestas sobre la curva Γ, ahora se imponen sobre cada fragmento de la curva. La Figura 1.1 presenta el esquema propuesto. La notación utilizada es la siguiente:

- S_i representa al subsistema i, $i = 1, ..., N$.
- $U^k_{i,j}$ es la unión k entre subsistemas i y j, $k = 1, ..., K_{i,j}$.
- $I^l_{S_i}$ es la interfaz local l del subsistema i, $l = 1, ..., L_i$.
- Γ_i es la porción de frontera exterior en el subsistema N, $\Gamma_1 \cup \Gamma_2 \cup ... \cup \Gamma_i ... \cup \Gamma_N = \Gamma$. Notar que Γ_j puede ser nula para algún S_j.
- $(x_m)^{I_l}_{S_i}$ es el valor de la variable x_m en la interfaz l del subsistema i, $m = 1, ..., M_i$.
- $(\bar{x})^{I_l}_{S_i}$ es el vector de incógnitas $\{x_1, x_2, ..., x_{M_i}\}$ en la interfaz l del subsistema i.

En principio, existen tantas incógnitas como valores de variables en cada interfaz. Sin embargo, es posible notar que la unión $U^k_{i,j}$ que relaciona los sistemas S_i y S_j mediante las interfaces $I^{l_1}_{S_i}$ e $I^{l_2}_{S_j}$ respectivamente, define una relación de continuidad [1]entre las incógnitas $(x_m)^{I_{l_1}}_{S_i}$ y $(x_m)^{I_{l_2}}_{S_j}$, de tal forma que:

$$(x_m)^{I_{l_1}}_{S_i} = (x_m)^{I_{l_2}}_{S_j} \tag{1.1}$$

Estas relaciones reducen a la mitad la cantidad de incógnitas. Las demás ecuaciones se encuentran a partir del modelo de estudio de cada subsistema. Sean $(F_m)^l_i$ las relaciones funcionales que calculan el valor de las incógnitas $(x_m)^{I_l}_{S_i}$ en la interfaz l del subsistema i, a partir del valor de otras incógnitas y de los datos de contorno sobre la frontera exterior Γ_i:

$$(x_m)^{I_l}_{S_i} = (F_m)^l_i \left((\bar{x})^{I_1}_{S_i}, (\bar{x})^{I_2}_{S_i}, ..., (\bar{x})^{I_{L_i}}_{S_i}, (\alpha_i(\Gamma_i) \right) \tag{1.2}$$

[1] Las incógnitas que representan derivadas normales en la interfaz de acople pueden tomar signos opuestos según la convención. Por ejemplo, si el flujo de calor es una incógnita, y se define como flujo positivo a aquel que es saliente del subsistema, entonces la condición de continuidad implicará que:

$$(q")^{I_{l_1}}_{S_i} = -(q")^{I_{l_2}}_{S_j}$$

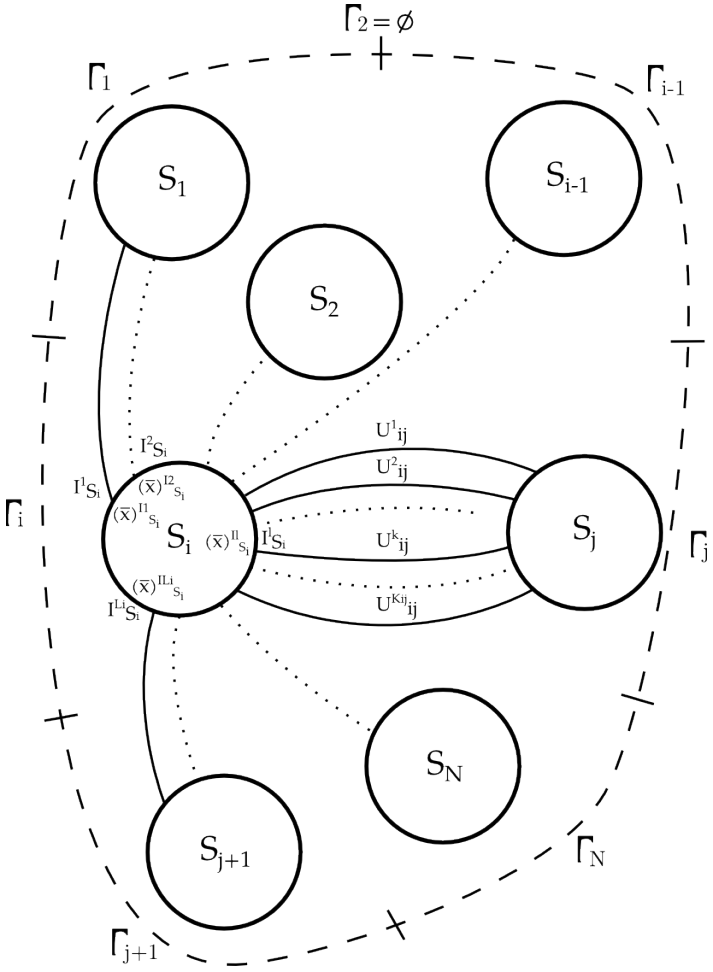


Figura 1.1: Esquema de subsistemas de estudio relacionados mediante condiciones de borde dinámicas en interfaces de acoplamiento.

donde $(\alpha_i(\Gamma_i)$ representa las condiciones de borde impuestas sobre la curva Γ_i. Notar que algunas de las dependencias pueden anularse dependiendo del modelo de estudio utilizado en cada subsistema [2].

1.2.2. Sistema de ecuaciones a resolver

Restando el lado derecho en (1.2) a ambos lados, se obtienen relaciones del tipo:

$$(R_m)_i^l = (x_m)_{S_i}^{I_l} - (F_m)_i^l \left((\bar{x})_{S_i}^{I_1}, (\bar{x})_{S_i}^{I_2}, ..., (\bar{x})_{S_i}^{I_{L_i}}, (\alpha_i(\Gamma_i) \right) \tag{1.3}$$

para cada residuo en la incógnita m de cada interfaz l del subsistema i. Debido a las relaciones de continuidad (1.1), es posible descartar ecuaciones, seleccionándolas de tal

[2] Cuando la expresión es más sencilla y solo involucra el valor de otro tipo de condición de borde en la misma interfaz, la relación funcional recibe el nombre de operador *Steklov-Poincaré*. En matemática, el operador *Steklov-Poincaré* mapea el valor de una condición de borde de una PDE elíptica en un dominio al valor de otra condición de borde (por ejemplo, una condición de borde de tipo *Dirichlet* en una condición de borde de tipo *Neumann*). Usualmente, cualquiera de las dos condiciones determinan la solución.

forma que el sistema de ecuaciones en cada subsistema quede bien planteado, como se explica más adelante. La convergencia fuerte de los subsistemas acoplados necesita que los residuos sean nulos. El sistema global de ecuaciones restante se reduce a la siguiente expresión:

$$\bar{R} = \bar{0} \tag{1.4}$$

donde $\bar{R}$ es el vector de residuos de las ecuaciones seleccionadas. En general estas relaciones son no lineales, y por lo tanto se resuelven de forma iterativa, a partir de un primer vector solución propuesto. Debido a que en cada subsistema solo han sobrevivido ciertas relaciones funcionales, van a tomar como dato solo algunos de estos valores propuestos. Es necesario por tanto seleccionar las ecuaciones de modo que el problema resulte bien planteado. Por ejemplo, si interesara calcular el campo de temperaturas en un subsistema dado, serían incógnitas la temperatura y el flujo de calor en cada una de sus interfaces. Sin embargo ambas no pueden ser impuestas como dato en el mismo subsistema. En general, cada subsistema debe recibir condiciones o bien de tipo *Dirichlet*, o bien de tipo *Neumann*, para cada ecuación.

1.2.3. Métodos numéricos para la resolución de sistemas de ecuaciones de residuos

Existen diferentes estrategias para hallar las raíces del sistema (1.4). La forma clásica es el conocido método *Dirichlet-to-Neumann*, que resuelve mediante iteraciones de tipo *Piccard*. La ventaja de este método es que es sencillo de implementar, ya que es un método explícito y por tanto, los valores calculados por algún subsistema son inmediatamente impuestos como condición de borde a otros subsistemas. Sin embargo, tiene algunas desventajas. Por ejemplo, las condiciones de borde que son datos en cada subsistema no pueden elegirse arbitrariamente. Las interfaces $I_{S_i}^{l_1}$ y $I_{S_j}^{l_2}$ comunes a cada unión $U_{i,j}^{k}$ deben alternar condiciones de borde de tipo *Dirichlet* y de tipo *Neumann* para las ecuaciones que relacionan las mismas variables de estado en cada subsistema.

El siguiente ejemplo esclarece lo expresado. Se desea resolver la ecuación de calor con condiciones de borde homogéneas, para encontrar el campo de temperaturas u en una barra unidimensional de longitud L, fuente interna de energía f y conductividad térmica k:

$$\begin{cases} -k\Delta u = f \\ u|_{\partial\Omega} = 0 \end{cases} \tag{1.5}$$

mediante el método de descomposición disjunta de dominios (ver Figura 1.2. El dominio original $[0, L]$ es particionado en los subdominios $[0, c]$ y $[c, L]$. Para que cada problema

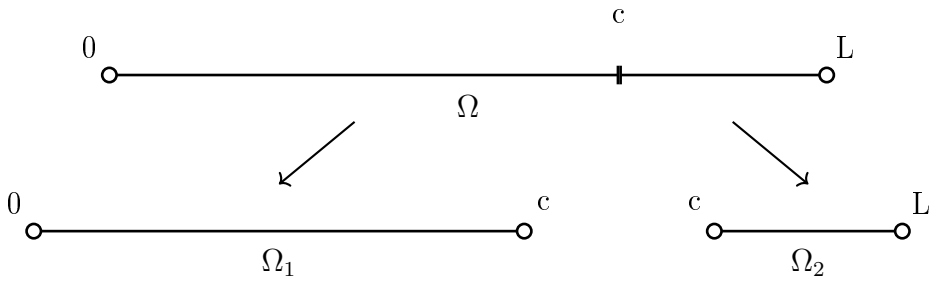


Figura 1.2: Descomposición disjunta de dominios en el cálculo del campo de temperatura a lo largo de una barra unidimensional.

quede bien planteado es necesario imponer una condición de borde extra en el punto de interfaz a cada lado. En el método *Dirichlet-to-Neumann*, es necesario decidir qué subsistema va a resolverse en primera instancia, a partir de algún valor supuesto en el borde. Si arbitrariamente se decidiera imponer una temperatura (condición *Dirichlet* al borde del primer subsistema, luego de realizar el cálculo de temperaturas quedaría definido un flujo calórico a través de su interfaz de conexión con el segundo subsistema. Este valor de flujo luego debe ser utilizado para imponerse como condición de borde al segundo subsistema, en el cuál se resolverá el campo de temperaturas y quedará definida una nueva temperatura en la interfaz de acople. Esta temperatura se impone nuevamente al primer subsistema, y el cálculo continúa así hasta que los sucesivos valores de las variables acopladas convergan. Si inicialmente se hubiera impuesto una condición de *Neumann* en el primer subdominio, necesariamente al segundo subdominio debe imponérsele una condición de *Dirichlet*. Es decir, al utilizar el método *Dirichlet-to-Neumann*, la elección de un tipo de frontera en un subdominio dado determina el tipo de frontera en el subdominio contiguo, para las ecuaciones que relacionan las mismas variables de estado en ambos subsistemas. Otra desventaja de este método es que en general requiere demasiadas iteraciones para converger [5]. En algunos problemas el método puede quedar estancado, iterando en series de valores que se repiten en ciclo. Y en otros casos el método es divergente (sin ir más allá, para un cierto conjunto de parámetros del problema ejemplo analizado, el método diverge, ver [6]).

Otra forma de resolver el problema es aplicando métodos para encontrar raíces a funciones vectoriales no lineales (sistema de ecuaciones de residuos 1.4 en función de las variables de acople incógnitas), como por ejemplo, el método *Newton-Raphson*. Haciendo un desarrollo de Taylor de las ecuaciones de residuos alrededor del punto $\bar{x}_n$, truncando los términos superiores al primer orden, y evaluando en $x = x_{n+1}$ se tiene:

$$\bar{R}(\bar{x}) = \bar{R}(\bar{x}_n) + \nabla\bar{R}(\bar{x}_n)(\bar{x}_{n+1} - \bar{x}_n) \tag{1.6}$$

donde $\nabla\bar{R}(\bar{x}_n)$ es la matriz jacobiana del sistema J evaluada en el punto $\bar{x}_n$, donde el elemento J_{ij} debe evaluarse como $J_{ij} = \frac{\partial R_i}{\partial x_j}$. Suponiendo que x_{n+1} tiende a la raíz

buscada se ha de cumplir que $\bar{R}(\bar{x}_{n+1}) = 0$. Sustituyendo en 1.6 y operando algebraicamente se llega a la siguiente expresión:

$$\bar{R}(\bar{x}_n) = -J(\bar{x}_n)(\bar{x}_{n+1} - \bar{x}_n) \tag{1.7}$$

que es el método de iterativo $Newton-Raphson$ con orden de convergencia cuadrática. Para hallar la solución $\bar{x}_{n+1}$ simplemente hay que resolver el sistema:

$$\bar{x}_{n+1} = \bar{x}_n - J^{-1}(\bar{x}_n)\bar{R}(\bar{x}_n) \tag{1.8}$$

El problema es que para utilizar este método se requiere la construcción de la matriz jacobiana en cada iteración, lo cual es demasiado costoso. Una sencilla aproximación mediante diferencias finitas de primer orden de cada elemento de la matriz jacobiana requiere numerosas ejecuciones de códigos clientes [3]. Si bien estas evaluaciones son independientes entre sí y por lo tanto altamente paralelizables, este método es excesivamente costoso.

Una forma alternativa y elegante de resolver el sistema de ecuaciones planteado en 1.4 es mediante métodos *quasi-Newton*. Estos métodos tienen convergencia superlineal [7], y por lo tanto presentan una ventaja frente a los métodos mediante iteraciones de *Piccard*. La característica principal de estos métodos es que aproximan la matriz jacobiana sin necesidad de realizar evaluaciones extras, y por lo tanto tienen también un punto de ventaja frente al método de *Newton-Raphson*. El método de *Broyden* es uno de ellos [8]. También existe otra variante del método, el método *Broyden ortonormal*, que asegura la convergencia en N iteraciones para sistemas lineales [7], donde N es la dimensión de la matriz jacobiana.

Existe también un conjunto de métodos conocidos como métodos *Newton-Krylov* para la resolución del sistema 1.4. Estos métodos no requieren el cálculo de matriz jacobiana ya que resuelven el sistema mediante gradientes descendientes, gradientes conjugados y otras técnicas que también serán exploradas en el presente trabajo. Sin embargo, en cada paso de descenso requieren evaluaciones de residuos, y por tanto son altamente costosos.

[3] Cada elemento $J_{ij} = \frac{\partial R_i}{\partial x_j}$ puede aproximarse mediante diferencias finitas a primer orden como:

$$J_{ij} \approx \frac{R_i(x_j + \Delta x_j) - R_i(x_j)}{\Delta x_j} \tag{1.9}$$

La construcción de la matriz jacobiana con éste método requiere una evaluación del vector residuo $\bar{R}$ en el punto $\bar{x}$ y luego N evaluaciones extras, donde N es la cantidad de incógnitas. Es decir, en total, se requieren $N + 1$ ejecuciones de cada código cliente.

1.2.4. Problemas de evolución temporal

En problemas de avance temporal la estrategia de selección de variables que son datos o incógnitas en cada interfaz puede variar en la evolución. Cabe resaltar también que no existe necesidad de que los cálculos en cada subsistema utilicen el mismo paso temporal. Cualquiera de los códigos podría utilizar subdivisiones del paso temporal del otro código, y establecer el acople sólo en pasos de tiempo determinados. Los valores para las condiciones de borde dinámicas entre los pasos de tiempo en los que efectivamente se intercambian datos, pueden interpolarse con valores previos.

Notar que la herramienta es incondicionalmente estable debido a que en cada paso de tiempo se asegura la convergencia fuerte de los valores de las variables en las interfaces de acople.

1.3. Objetivos y estructura de trabajo

Considerando la motivación y la formulación precedente, queda establecido el siguiente objetivo general de la maestría:

> *Implementar el acoplamiento fuerte entre modelos dimensionalmente heterogéneos, resolviendo cada subdominio por separado con códigos particulares, e imponiendo la interacción entre ellos sólo mediante condiciones de borde.*

La estructura de la tesis es detallada a continuación. En el presente capítulo se han planteado las ecuaciones que surgen al dividir sistemas complejos mediante interfaces con condiciones de borde dinámicas, y se han presentado alternativas numéricas para la resolución de las mismas. En el Capítulo 2 se describen las diferentes formas de implementar el acoplamiento entre códigos que han sido investigadas e implementadas, se presenta la estructura de comunicación definida y se describe la arquitectura de acoplamiento necesaria a ser implementada en los códigos comunicados por paso de mensajes. En el Capítulo 3 se muestran algunas aplicaciones de la herramienta estudiada, presentando distintos códigos utilizados para realizar cálculos fluidodinámicos. La primera aplicación es un sistema fluídico cerrado que se estudia subdividiéndolo en dos subsistemas, con dos interfaces de acople cada uno. El movimiento del fluido está gobernado por fuerzas naturales y por lo tanto interesa acoplar variables de caudal, presión, temperatura y flujo de calor en cada interfaz. El siguiente sistema de estudio es el vaciado del tanque reflector del reactor de investigación RA-10. Interesa analizar los tiempos de descarga ya que el mismo es diseñado como Segundo Sistema de Parada (SSP). Se abordan distintos modelos multiescala del mismo, para estudiar el detalle fluídico tridimensional en un componente del sistema, acoplando con condiciones de borde dinámicas a modelos cero-dimensionales que representan el resto del sistema. Al

final del capítulo se presenta un estudio de redes hidráulicas con múltiples componentes, para demostrar la eficiencia de la herramienta en acoples de mayor escala. En éstos últimos ejemplos comentados, las variables incógnitas en las interfaces de acoplamiento son velocidades (o caudales) y fuerzas (presiones y tensiones de corte). En el Capítulo 4 se extiende la técnica de acople a problemas que involucran otros modelos físicos. Se describe el código maestro de acople multifísico desarrollado y se presentan ejemplos de aplicación en el problema neutrónico-termohidráulico.

Capítulo 2

Estrategia de acoplamiento

"Construimos demasiadas murallas y no suficientes puentes."
— Isaac Newton, 1643-1727

2.1. Paradigma maestro-esclavo

El modelo de comunicación utilizado en el trabajo recibe el nombre de *maestro-esclavo* [9]. En este modelo, existe un programa *maestro* que tiene el control unidireccional sobre los demás programas, que actúan bajo el rol de *esclavos*. Cada código *esclavo* se encarga de calcular el valor de las incógnitas en las interfaces de acople mediante las ecuaciones 1.2. El código *maestro* recibe estos valores y con ellos resuelve las ecuaciones de residuos 1.3. En base a estos residuos propone[1] nuevos valores para las incógnitas en las interfaces de acople y se los envía a sus *esclavos*. Así también, es función del código *maestro* enviarles órdenes de comenzar el cálculo en un dato paso de evolución, reiniciar el cálculo o incluso abortar.

Cabe notar que cada código *esclavo* podría ejecutarse en varios procesos, paralelizando sus cálculos, ya sea mediante memoria compartida como mediante memoria distribuida. Incluso podrían lanzarse diversos procesos del código *maestro* en la resolución de algún problema. Debido a la complejidad en destinatarios de mensajes, cantidades y tipos de variables a compartir, es necesario definir una estrategia clara de comunicación que permita acoplar diversos códigos de manera genérica, segura y eficaz. La estrategia definida es comentada en la sección 2.3.

[1] Esta propuesta reside en el método de resolución de ecuaciones no lineales seleccionado, ver sección 1.2.3.

2.2. Códigos maestros utilizados

En este trabajo se utilizaron dos códigos maestros para la resolución de los problemas. El primer código *maestro* utilizado es el código **Coupling** desarrollado por XXX en XXX. **Coupling** permite acoplar códigos mediante funciones del estándard Interfaz de Paso de Mensajes (MPI por sus siglas en inglés, *Message Passing Interface*). Esta estrategia requiere el código fuente de los programas *esclavos* para implementar las funciones adecuadas. El código maestro está diseñado de tal forma que los códigos acoplados resuelvan ecuaciones del tipo 1.2 para incógnitas en interfaces con condiciones de borde de tipo *Dirichlet* o de tipo *Neumann*.

Con la idea de extender estas capacidades al acoplamiento de programas cuyos códigos fuente no estuvieran disponibles, así como a programas cuyos cálculos no dependieran exclusivamente de variables seteadas como condiciones de borde, sino de otros parámetros generales del sistema, se desarrolló un código más genérico de acoplamiento, el código **Newton** [10], que será descripto en el Capítulo 4.

2.3. Modelos de comunicación

La configuración *maestro-esclavo* requiere la ejecución de múltiples programas independientes. Al mismo tiempo, cada código podría estar corriendo en forma paralelizada[2]. Debido a esta complejidad es necesario planificar la estrategia de comunicación considerando la distribución del cálculo en múltiples procesadores, con el objetivo de no perder generalidad en la herramienta desarrollada. Los modos de comunicación implementados son los siguientes:

- paso de mensajes: este modo es implementado para comunicar procesos de programas en los cuales es posible modificar los códigos fuente;

- lectura y escritura de archivos de entrada y salida: este modo es implementado para comunicar procesos de programas que serán tratados como cajas negras.

Si bien este tipo de comunicaciones remotas son mucho más lentas que las comunicaciones locales, en general su latencia es despreciable frente al tiempo de cálculo de los procesos *esclavos*.

[2] En general, cada programa *esclavo* es un programa que ya ha sido utilizado y validado para algún tipo de cálculo. Si el código está paralelizado, en base a estudios de *speedup* se podría tener cierta experiencia en su modo de ejecución óptimo para alguna tarea dada. Esta ejecución podría requerir múltiples nodos en un clúster, por ejemplo.

Paso de mensajes

En sistemas de memoria distribuida, el paso de mensajes es un método de programación utilizado para realizar el intercambio de datos entre los procesadores [11]. En estos sistemas, los datos son enviados de un procesador a otro utilizando mensajes. El paso de mensajes involucra la transferencia de datos desde un proceso que envía a un proceso que recibe. El proceso que envía, necesita conocer la localización, el tamaño y el tipo de los datos, así como el proceso destino.

Estas funcionalidades se implementaron siguiendo el protocolo MPI. MPI es una especificación de paso de mensajes aceptada como estándar por todos los fabricantes de computadores. El objetivo principal de MPI es proporcionar un estándar para escribir programas (lenguajes *C*, *C++*, *Fortran*) con paso de mensajes. De esta forma, se pretende mejorar la portabilidad, el rendimiento, la funcionalidad y la disponibilidad de las aplicaciones.

Se utilizaron dos implementaciones alternativas. En la primera, cada código *esclavo*, así como el código *maestro*, son ejecutados de manera independiente en uno (SISD por sus siglas en inglés, Single Instruction, Single Data) o múltiples procesos (SIMD por sus siglas en inglés, Single Instruction, Multiple Data). El código *maestro* publica una serie de puertos a los cuales cada código *esclavo* puede conectarse[3]. Una vez aceptadas las conexiones, los programas pueden intercambiar mensajes siguiendo una lógica preestablecida. Cuando ya no es necesario que dos programas continúen comunicándose, se cierran las conexiones.

En la otra implementación, todos los programas son ejecutados al mismo tiempo (MIMD por sus siglas en inglés, Multiple Instruction, Multiple Data). En este tipo de ejecuciones todos los procesos cuentan con un único comunicador original, *MPI_COMM_WORLD*, y por ello es necesario crear nuevos grupos de procesos y de comunicadores. Una vez establecidos los comunicadores, los programas pueden intercambiar mensajes siguiendo la misma lógica preestablecida en el modelo previo. Si bien esta implementación no permite la posibilidad de realizar nuevas conexiones una vez que los programas han sido ejecutados, son mucho más seguras, ya que no dependen del éxito de encontrar los puertos requeridos para las conexiones.

Lectura y escritura de archivos de entrada y salida

La comunicación mediante lectura y escritura de archivos se implementó para demostrar la capacidad de acoplar códigos cuyos códigos fuente no son capaces de ser modificados. La idea principal es ejecutar corridas simples del código *esclavo* administradas desde el código *maestro*. Para ello el código *maestro* escribe en el archivo

[3] Para que los programas puedan encontrar los puertos publicados, es necesario que todos ellos pertenezcan a un mismo servicio de comunicación generado por el *ompi-server*.

de entrada del programa *esclavo* todos los parámetros necesarios para la ejecución del cálculo, ordena su ejecución, y espera a que este termine y luego realiza una búsqueda de los valores de las variables de interés en archivos de salida. En problemas de evolución, el código *maestro* debe notificar en el archivo de entrada el parámetro de evolución, así como otros valores de variables de estado del paso previo.

La ejecución de programas *esclavos* se implementó de dos formas alternativas. La primera forma es mediante el uso de la función *system* de la librería de *c*. Esta función deja al proceso que la ejecuta en pausa hasta que el programa *esclavo* finaliza, cuando ella retorna algún mensaje de error o de éxito. La ventaja de esto es que no debe implementarse alguna función extra para conocer cuándo leer los archivos de salida del programa *esclavo*. Sin embargo, no es posible disparar múltiples procesos de un programa mediante la función *system* desde un proceso que actualemente utiliza *MPI*. Ésta prohibición es necesaria para controlar el disparo de procesos. Para este tipo de ejecuciones, existe la función MPI_Comm_Spawn de *MPI*, que se implementó como forma alternativa de ejecución de programas *esclavos*. Esta función permite especificar la cantidad de procesos de ejecución del programa a disparar. El problema es que la función devuelve el control al programa *maestro* de forma instantánea, sin esperar a que el código disparado finalice, por lo que, en principio, debe implementarse alguna función extra para saber cuándo es posible leer el archivo de salida.

Es necesario notar que este modelo de comunicación no es tan eficiente como el de intercambio de mensajes, ya que la lectura y escritura de archivos consume mayores recursos de tiempo, por lo que, siempre que fuera posible, es recomendable implementar el otro modelo. Además, requiere la programación de rutinas extras específicas dedicadas a la escritura de archivos de entrada y lectura de archivos de salida de distintos códigos *esclavos*.

Estructura de comunicación implementada

Se desarrollaron funciones híbridas para códigos maestros con la finalidad de cubrir todas las formas de comunicación descriptas en la sección previa. En el caso de comunicación por intercambio de mensajes, la estrategia definida establece comunicaciones siempre entre un único proceso del código *maestro*, el proceso *raíz*, y un único proceso de cada código *esclavo* (sus propios procesos *raíces*[4]). En el caso de lectura y escritura de archivos y ejecución de programas *esclavos*, la estrategia definida paraleliza las responsabilidades entre todos los procesos lanzados del código *maestro*. El esquema 2.1 resume la estrategia de comunicación.

[4] Es responsabilidad del código *esclavo* la comunicación de los datos recibidos por el proceso *raíz* a los demás procesos.

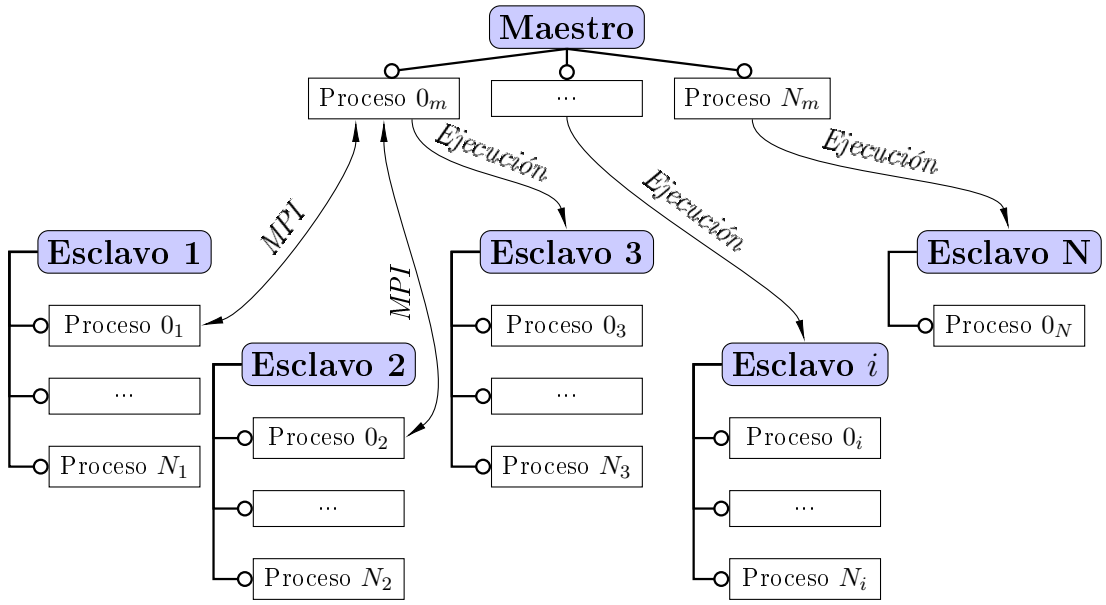


Figura 2.1: Esquema de comunicación entre los programas *esclavos* y el programa *maestro* implementado en el acoplamiento de códigos. Los *esclavos* se comunican solo con el *maestro* y no intercambian datos entre sí. Se utilizan dos modelos de comunicación diferentes. En el primero, cada código *esclavo* es comunicado con el código *maestro* a través de intercambio de mensajes por *MPI*. Como podrían correr en modo serial o paralelo, solo sus procesos *raíces* establecen la comunicación con el proceso *raíz* del programa *maestro*. En el segundo modelo de comunicación, los códigos *esclavos* son directamente *ejecutados* por el programa *maestro* en uno o varios procesos. En este modelo, la comunicación se establece solo mediante lectura y escritura de archivos.

2.4. Arquitectura de acoplamiento montada en códigos *esclavos* comunicados por paso de mensajes

En general, los códigos *esclavos* son programas de cálculo particulares que no han sido diseñados para mantenerse acoplados a otros códigos. En esta sección se demuestra cómo mediante unas mínimas modificaciones en sus rutinas es posible implementar un acoplamiento eficiente. Las acciones de acoplamiento deben ser llamadas en 4 instancias diferentes:

1. al principio del programa;
2. al principio de cada paso de evolución;
3. al finalizar cada paso de evolución;
4. al finalizar el programa.

Los problemas que no involucran evolución de variables pueden ser tratados como problemas con un solo paso de evolución. El programador podría definir una nueva variable *booleana* que a modo de bandera indique cuándo se está realizando un cálculo acoplado para ingresar o no en las instancias nombradas. A continuación se describen las instancias de acoplamiento.

Acoplamiento en instancia 1: al principio del programa

En esta instancia es necesario establecer la comunicación *MPI* entre el proceso *raíz* del código *esclavo* y el código maestro. Si ambos programas han sido ejecutados en el esquema *MIMD* los pasos a realizar son los siguientes:

- creación de grupo global de procesos;
- creación de subgrupo local de procesos;
- creación de un comunicador dentro del subgrupo previo, necesario para el paso de mensajes dentro del programa;
- creación de un grupo entre el proceso *raíz* del programa esclavo y el proceso *raíz* del programa *maestro*;
- creación de un comunicador en el grupo previo, necesario para el paso de mensajes de acople.

Si en cambio, el programa ha sido ejecutado en forma independiente, los pasos a realizar son los siguientes:

- búsqueda del puerto publicado por el el proceso *raíz* del programa *maestro*;
- conexión del proceso *raíz* del programa esclavo a este puerto y creación del comunicador.

Una vez implementada la comunicación, el código *esclavo* puede recibir datos generales (como parámetros de evolución iniciales, cantidad de pasos de evolución, cantidad de incógnitas en interfaces de acople, cantidad de interfaces de acople, etc.), y chequear la consistencia con los datos propios del programa. Si es necesario, los datos locales pueden ser cambiados notificando al usuario.

Acoplamiento en instancia 2: al principio de cada paso de evolución

La estrategia de acoplamiento se define entre el parámetro de evolución inicial $t_{coup,0}$ y el parámetro final $t_{coup,N}$, con $N+1$ pasos de acoplamiento cada $\Delta t_{coup} = \frac{t_{coup,N} - t_{coup,0}}{N}$. Si bien el código *esclavo* debe intercambiar mensajes en cada uno de estos pasos, es posible que además utilice subpasos de evolución Δt_{local} locales[5]. En estos casos, se implementa una estrategia de interpolación de los valores de las variables en las interfaces de acoplamiento entre los pasos de evolución acoplados.

Al principio de cada paso acoplado de cálculo el código *esclavo* recibe valores supuestos para las incógnitas que se toman como dato en las interfaces de acople, en función de la estretegia implementada (ver sección 1.2). El programa resuelve el paso Δt_{coup} en base a ellos.

Acoplamiento en instancia 3: al finalizar cada paso de evolución

Una vez resuelto cada Δt_{coup}, el programa envía al programa *maestro* los valores de las variables que se han definido como incógnitas en las interfaces de acoplamiento. Tras este envío, el programa queda en espera de órden para continuar. Mientras, el programa *maestro* recepciona los valores de las incógnitas calculados por los demás códigos *esclavos*. Con estos valores resuelve las ecuaciones de residuos. Si el módulo del residuo cae por debajo de cierta tolerancia prefijada el código *maestro* acepta los resultados y envía a sus *esclavos* la órden de continuar con el cálculo. En caso contrario, puede enviarles la órden de volver a calcular el mismo paso de acoplamiento, o incluso de abortar el cálculo. La Figura 2.2 esquematiza lo comentado para un caso sencillo

[5] Distintos programas pueden tener diferentes requisitos sobre el parámetro de evolución, dependiendo de la física que resuelven. Algunos, por ejemplo, podrían estar resolviendo transitorios fluidodinámicos, en los que es de interés mantener por debajo de algún valor ciertos parámetros (como el número de *Courant*) dependientes del paso de tiempo. Otros, en cambio, pueden no tener este requisito. La idea es que cada programa *esclavo* devuelva al programa *maestro* el mejor resultado posible al cabo de Δt_{coup}.

en que se acoplan dos programas *esclavos* al programa *maestro*. En este ejemplo, las variables x, y son las incógnitas en las interfaces de acople. El programa **Esclavo 1** resuelve cada paso de evolución acoplado en función de un valor x_{guess} recibido desde el código *maestro*, y le devuelve el valor y calculado a partir de él. El programa **Esclavo 2** calcula, en cambio, x en función de y_{guess}.

Acoplamiento en instancia 4: al finalizar el programa

Antes de finalizar el programa, es necesario cerrar las conexiones, liberar los grupos y los comunicadores establecidos.

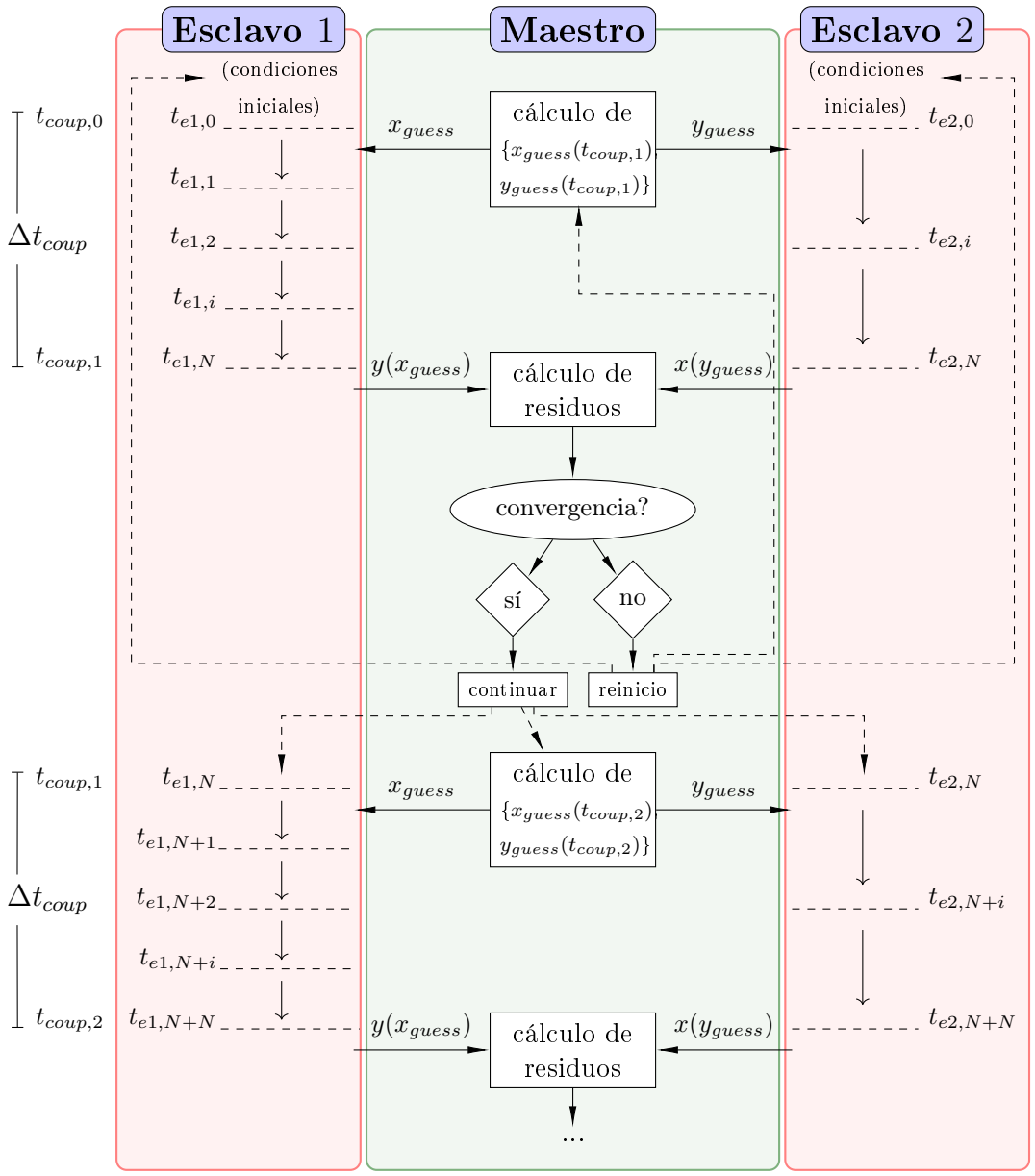


Figura 2.2: Esquema de acoplamento entre el programa *maestro* y los programas *esclavos*. En el ejemplo, cada código *esclavo* resuelve ecuaciones diferenciales en distintos subsitemas. Estos subsistemas están acoplados entre sí en alguna interfaz en las que las variables $\{x, y\}$ son incógnitas. Los cálculos se acoplan cada Δt_{coup}, pero cada programa utiliza subpasos de cálculos locales. El código **Esclavo 1** inicia recibiendo como *guess* para el tiempo $t_{coup,1}$ la variable x_{guess}. El valor de x_{guess} utilizado en los pasos intermedios de cálculo es simplemente una interpolación entre la condición incial y el valor recibido. El código **Esclavo 2** recibe alternativamente como *guess* para el tiempo $t_{coup,1}$ la variable y_{guess}. Al finalizar Δt_{coup} ambos programas devuelven al programa **Maestro** las variables conjugadas calculadas. **Maestro** computa los residuos entre los valores *guess* previamente propuestos y los valores recibidos. Si el residuo no supera cierta tolerancia prefijada, envía la órden de reinicio a cada programa *esclavo*, para volver a calcular el mismo paso de acoplamiento, tras lo cual enviará nuevos valores *guesses* propuestos. En caso contrario, cuando los resultados convergen, envía la órden de continuación, y ambos *esclavos* prosiguen con el cálculo. Notar que en problemas sin evolución, el proceso es similar, pero todos los programas calculan un único paso temporal ficticio.

Capítulo 3

Ejemplos de aplicación

"The City's central computer told you? R2-D2, you know better than to trust a strange computer. "
— C-3PO, from Star Wars

3.1. Movimiento por fuerza boyante en un circuito cerrado

Presentación del problema

Como primer ejemplo se presenta un sistema que se estudia analizándolo en dos subsistemas separados, definiendo dos interfaces de acople, y en cada una de ellas dos pares de variables dinámicas. El primer subsistema modela un fluido en un tanque de paredes adiabáticas y con fuente interna de energía. El segundo subsistema representa un circuito en el que el fluido transfiere energía en un intercambiador de calor para bajar su temperatura. Ambos se comunican mediante dos conexiones, una ubicada en la parte inferior y la otra en la parte superior, definiendo un circuito cerrado en el que el flujo queda completamente dominado por convección natural. El sistema completo modela el movimiento de un fluido en régimen de convección natural a través de una fuente fría de neutrones alojada próxima al núcleo de un reactor de investigación [12]. En la Figura 3.1 puede apreciarse un diagrama del sistema.

En cada interfaz de acople existen incógnitas de caudal, presión, temperatura y flujo de calor. Por lo tanto el sistema queda definido por ocho ecuaciones de continuidad de campos de variables y ocho ecuaciones de residuos que relacionan las incógnitas de forma similar a la que se presentó en la sección 1.2.

Las ecuaciones de continuidad en las interfaces implican que:

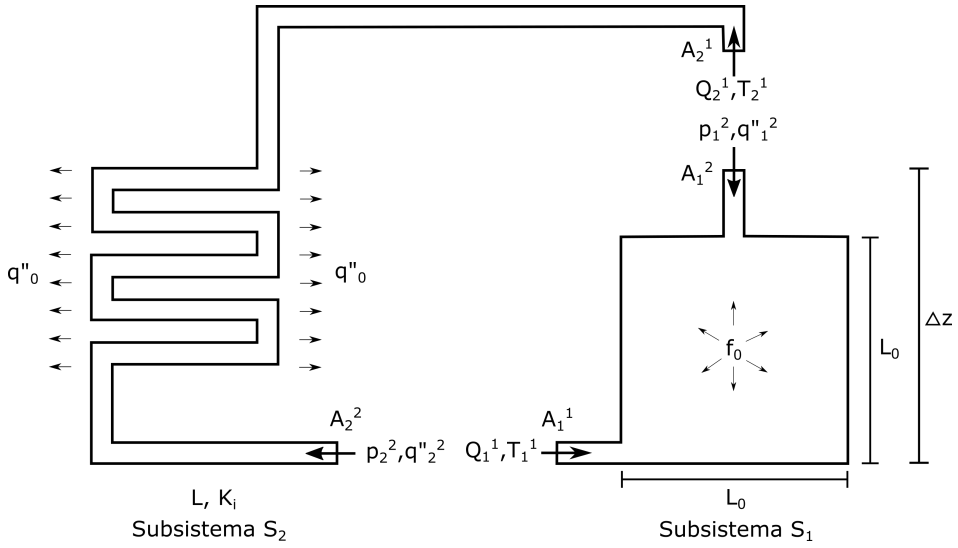


Figura 3.1: Esquema del sistema analizado. El subsistema de la izquierda es un intercambiador de calor y se estudia con un código cero-dimensional. El modelo de la derecha es una cavidad con una fuente de energía interna y se estudia con un código bi-dimensional. El sistema completo es abordado con una estrategia de acoplamiento mediante condiciones de borde dinámicas. En el esquema se ejemplifica una de las elecciones posibles para las variables que son datos en cada subsistema.

$$\begin{cases} Q_1^1 &= Q_2^2 \\ Q_1^2 &= Q_2^1 \\ p_1^1 &= p_2^2 \\ p_1^2 &= p_2^1 \\ T_2^1 &= T_2^2 \\ T_2^2 &= T_2^1 \\ q"_2^1 &= -q"_2^2 \\ q"_2^2 &= -q"_2^1 \end{cases} \tag{3.1}$$

donde Q es caudal, P es presión, T es temperatura y $q"$ es flujo de calor. Notar que el subíndice en cada variable refiere a la numeración global del subsistema, y el supraíndice indica el número de interfaz local, como se convino previamente en el Capítulo 1. Al evaluar los residuos en cada interfaz, se genera una ecuación no lineal por cada incógnita en cada interfaz. Para que las ecuaciones queden bien planteadas se selecciona solo una de las relaciones para el par presión-caudal y solo una para el par temperatura-flujo de calor en cada interfaz. Así entonces, entre las dos interfaces del subsistema 1 se generan 4 ecuaciones de residuos [1] del tipo $(R_m)_i^l = 0$:

[1] Cada ecuación de residuo relaciona las incógnitas según el modelo aplicado. En $R_{p,Q}$ se considera dependencia entre el caudal Q, la presión p y la temperatura T, y en $R_{T,q"}$ se considera dependencia entre el caudal Q, la temperatura T y el flujo de calor $q"$.

$$\begin{cases} (R_{p,Q})_1^1\,(Q_1^1, p_1^1, T_1^1, Q_1^2, p_1^2, T_1^2) &= 0 \\ (R_{T,q"})_1^1\,(Q_1^1, T_1^1, q"_1^1, Q_1^2, T_1^2, q"_1^2) &= 0 \\ (R_{p,Q})_1^2\,(Q_1^1, p_1^1, T_1^1, Q_1^2, p_1^2, T_1^2) &= 0 \\ (R_{T,q"})_1^2\,(Q_1^1, T_1^1, q"_1^1, Q_1^2, T_1^2, q"_1^2) &= 0 \end{cases} \tag{3.2}$$

y entre las dos interfaces del subsistema 2 se generan otras 4 ecuaciones de residuos:

$$\begin{cases} (R_{p,Q})_2^1\,(Q_2^1, p_2^1, T_2^1, Q_2^2, p_2^2, T_2^2) &= 0 \\ (R_{T,q"})_2^1\,(Q_2^1, T_2^1, q"_2^1, Q_2^2, T_2^2, q"_2^2) &= 0 \\ (R_{p,Q})_2^2\,(Q_2^1, p_2^1, T_2^1, Q_2^2, p_2^2, T_2^2) &= 0 \\ (R_{T,q"})_2^2\,(Q_2^1, T_2^1, q"_2^1, Q_2^2, T_2^2, q"_2^2) &= 0 \end{cases} \tag{3.3}$$

Notar que según la estrategia de acoplamiento seleccionada, algunas de las dependencias pueden anularse. En la Figura 3.1 se presenta una estrategia en la que las condiciones de borde dinámicas son de tipo de tipo Dirichlet para la interfaz inferior de la cavidad y la interfaz superior del intercambiador de calor, y de tipo de tipo Neumann para las restantes. Como el circuito es cerrado es necesario proveer un valor de referencia para la presión. En la formulación desarrollada se fija un valor de presión aritrario en la interfaz superior del intercambiador de calor, por lo que la ecuación $(R_{p,Q})_2^1 = 0$ queda descartada, y es sustituida por la siguiente:

$$p_2^1 = 0.$$

Subsistemas de estudio

Los parámetros del modelo del intercambiador de calor son los siguientes: flujo de calor por unidad de superficie $q_0" = -2 \cdot 10^5 W/m^2$, longitud de cañerías $L = 30\ m$, sumatoria de coeficientes de pérdida de carga concentrada $\sum K_i = 1{,}72$, rugosidad de cañerías $\epsilon = 0{,}5 \cdot 10^{-3}\ m$. Las áreas de las interfaces de acople son $A_2^1 = A_2^2 = 0{,}03\ m^2$. La altura total Δz de este subsistema es equivalente a la de la cavidad bidimensional. La evolución de las variables $\{p, Q, T, q"\}$ en el subsistema se calcula mediante un código cero-dimensional que resuelve ecuaciones de pérdida de carga en una red hidráulica [13] y de transferencia de energía en un intercambiador de calor con flujo constante [14]:

$$\begin{cases} p_2^1 + \rho g \Delta z &= p_2^2 + \rho \Delta u \\ \Delta u &= \frac{1}{2}\left(\frac{Q_1^1}{A_2^1}\right)^2 \left(\frac{f_D L}{D} + \sum_i K_i\right) \\ T_2^2 &= T_2^1 + 2\dfrac{q_0" L}{\frac{D}{2}\frac{Q_1^1}{A_2^1}\rho c_p} \end{cases} \tag{3.4}$$

donde f_D es el factor de Darcy de pérdida de carga distribuida y D es el diámetro de la tubería. En este modelo se supone que el flujo de calor es nulo en la dirección axial en cada interfaz de acople.

La cavidad bidimensional se modela con $L_0 = 0{,}3\ m$, y $A_1^1 = A_1^2 = 0{,}03\ m^2$. El fluido de trabajo es agua ($\rho_0 = 10^3\ Kg/m^3$, $\mu = 6 \cdot 10^{-4}\ Kg/ms$, $c_p = 4184\ J/KgK$, $k = 0{,}64\ W/mK$, $\beta = 0{,}44 \cdot 10^{-3} K^{-1}$) con fuente interna $f_0 = 10^6\ W/m^3$. La evolución de las variables $\{p, Q, T, q"\}$ en este subsistema se calcula resolviendo las ecuaciones de Navier-Stokes [15] y de transporte de energía [14]. Se utiliza la aproximación de *Boussinesq* considerando variaciones de densidad solo en el término de fuerza volumétrica:

$$\begin{cases} \dfrac{\partial \bar{u}}{\partial t} + (\bar{u} \cdot \nabla)\bar{u} + \dfrac{\nabla p}{\rho_0} - \nabla \cdot \left[(\nu + \nu_T) \left(\nabla \bar{u} + \nabla \bar{u}^T \right) \right] & & \\ \qquad - \left(1 - \beta(T - T_{ref})\right) \bar{g} & = & 0 \\ \nabla \cdot \bar{u} & = & 0 \\ \dfrac{\partial T}{\partial t} + (\bar{u} \cdot \nabla) T = 0 - \dfrac{k}{\rho_0 c_p} \Delta T & = & \dfrac{f_0}{\rho_0 c_p} \end{cases} \tag{3.5}$$

donde ρ_0 es la densidad del fluido a la temperatura de referencia T_{ref}.

Las paredes imponen condiciones de no deslizamiento para las ecuaciones de *Navier-Stokes* y de flujo de energía nulo para la ecuación de energía. Las ecuaciones (3.5) se resuelven mediante una formulación de elementos finitos, con elementos lineales para aproximar los campos de presiones, velocidades y temperaturas, estabilizando con los métodos *SUPG* [16] y *PSPG* [17].

La malla de cálculo se genera con **Gmsh** [18] y se discretiza el dominio en 43874 elementos triangulares con un tamaño medio de arista de $\Delta x \approx 0{,}005m$.

Como se mencionó previamente, no existe necesidad de que ambos códigos utilicen el mismo paso temporal de cálculo. Sin embargo en ambas simulaciones se utiliza $\Delta t = 0{,}01s$, debido a que ninguna requiere una mayor discretización temporal.

Resultados del cálculo

Las condiciones iniciales del sistema son estáticas y sin gradientes de temperatura. A medida que evoluciona el fluido comienza a incrementar su temperatura en la cavidad y a circular por fuerza boyante. El régimen del fluido depende del número adimensional de Richardson *Ri*, [19], que representa la relación entre las fuerzas boyantes y las fuerzas inerciales. Con los parámetros del subsistema bidimensional el *Ri* del fluido queda definido en $Ri = 28{,}34$. Como este valor es alto, el fluido se estratifica en capas de diferentes temperaturas. Las líneas de corrientes serpentean entre la entrada y la salida, manteniendo corrientes paralelas horizontales. En las Figuras 3.4 y 3.7 puede observarse la evolución de las líneas de corriente y del campo de temperatura en la

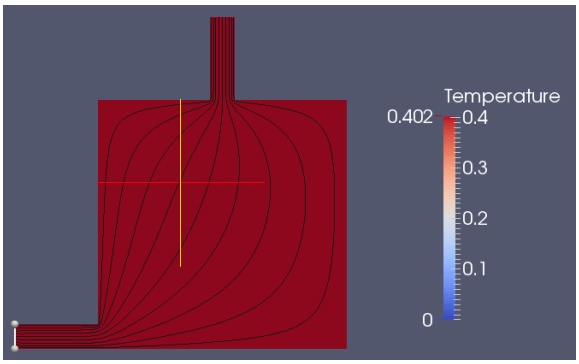


Figura 3.2: t=0 s

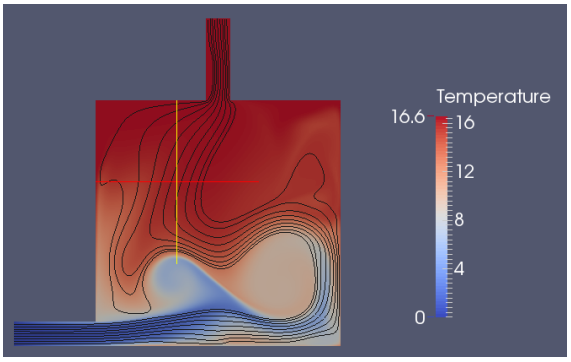


Figura 3.3: t=40 s

Figura 3.4: Evolución del fluido dentro de la cavidad bidimensional con fuente interna. El número de Richardson del fluido $Ri = 28{,}34$. Pueden apreciarse las líneas de corriente que se establecen al comienzo de la simulación.

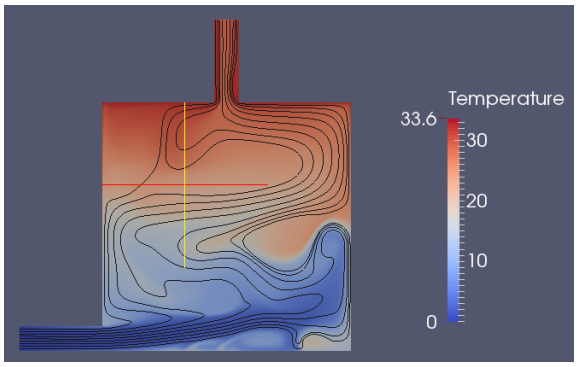


Figura 3.5: t=80 s

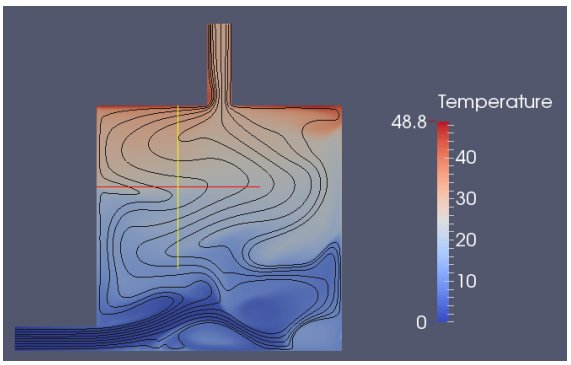


Figura 3.6: t=250 s

Figura 3.7: Evolución del fluido dentro de la cavidad bidimensional con fuente interna. El número de Richardson del fluido $Ri = 28{,}34$. Pueden apreciarse las líneas de corriente serpenteantes y la estratificación del fluido alcanzando un estado estacionario.

cavidad bidimensional.

Análisis de métodos de resolución del sistema de ecuaciones de residuos

Se exploran diferentes métodos numéricos para resolver el sistema de ecuaciones de residuos presentado en 3.2 y 3.3. En la Figura 3.8 puede apreciarse la cantidad de evaluaciones requeridas por cada método para disminuir los residuos debajo de cierta tolerancia prefijada, para cada paso temporal. El método de *Newton* calcula la matriz jacobiana en cada iteración. Este cálculo se realiza con diferencias finitas a primer órden y por lo tanto requiere 1 evaluación de los residuos en el punto inicial, y 8 evaluaciones extras para el cálculo de cada diferencia finita. En total son 9 evaluaciones extras. Puede observarse que la cantidad de iteraciones del método para converger es en promedio una sola, ya que en general utiliza 10 evaluaciones en cada paso temporal.

Los métodos *quasi-Newton* inicializan la matriz jacobiana sólo en el primer paso

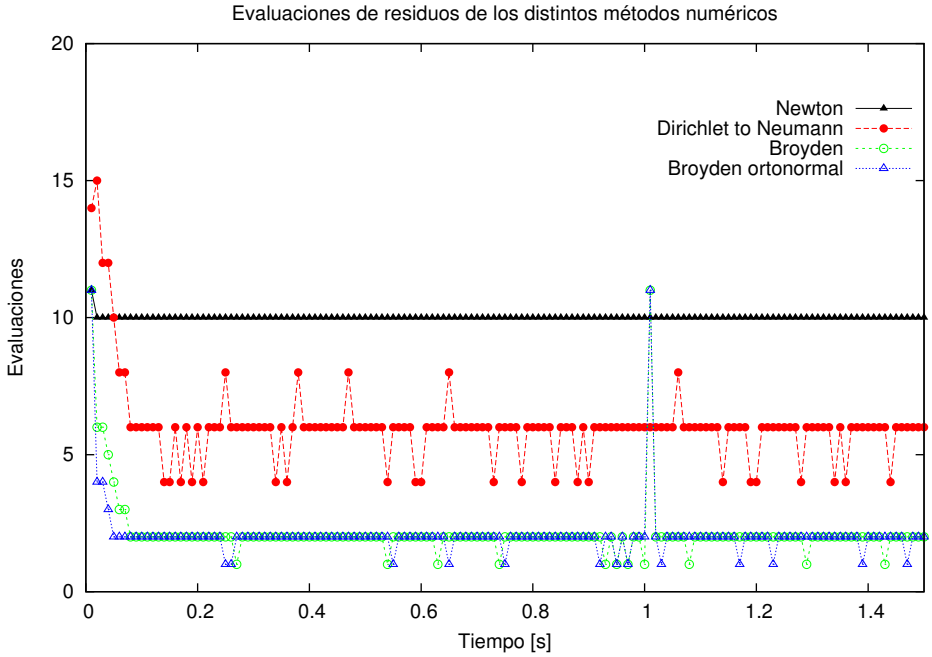


Figura 3.8: Evaluaciones de residuos requeridas por diversos métodos numéricos para resolver los sistemas de ecuaciones planteados en el problema doblemente acoplado descripto de la fuente fría de neutrones.

temporal, y luego utilizan aproximacionas económicas de la misma. Cada cierta cantidad de pasos temporales pueden reinicializar la matriz también mediante diferencias finitas. En los modelos realizados se utiliza reinicialización cada 100 pasos temporales, y por lo tanto la primera reinicialización se efectúa en el paso 101. En promedio estos métodos requieren dos iteraciones por cada paso temporal, además de las 9 llamadas extras a códigos en cada paso de reinicialización. Los métodos *Broyden* y *Broyden ortonormal* tinen comportamiento similar y demuestran ser más eficientes que el método clásico. El método *Dirichlet-to-Neumann* es el que mayor cantidad de iteraciones necesita por cada paso temporal, excediendo el doble de los pasos requeridos por los métodos *quasi-Newton*.

3.2. Análisis del segundo sistema de parada del reactor RA-10

Presentación del problema

La estrategia de análisis presentada puede aplicarse al estudio del vaciado del tanque reflector del reactor RA-10 como Segundo Sistema de Parada (SSP). El drenado del material reflector (agua pesada) disminuye drásticamente la reactividad, apagando el reactor. Es de interés verificar si el diseño cumple con el criterio de éxito, a saber, completar el 55 % del vaciado en un tiempo inferior a los 15 segundos, ante una falla

simple del sistema (falla de apertura de cualquiera de las válvulas).

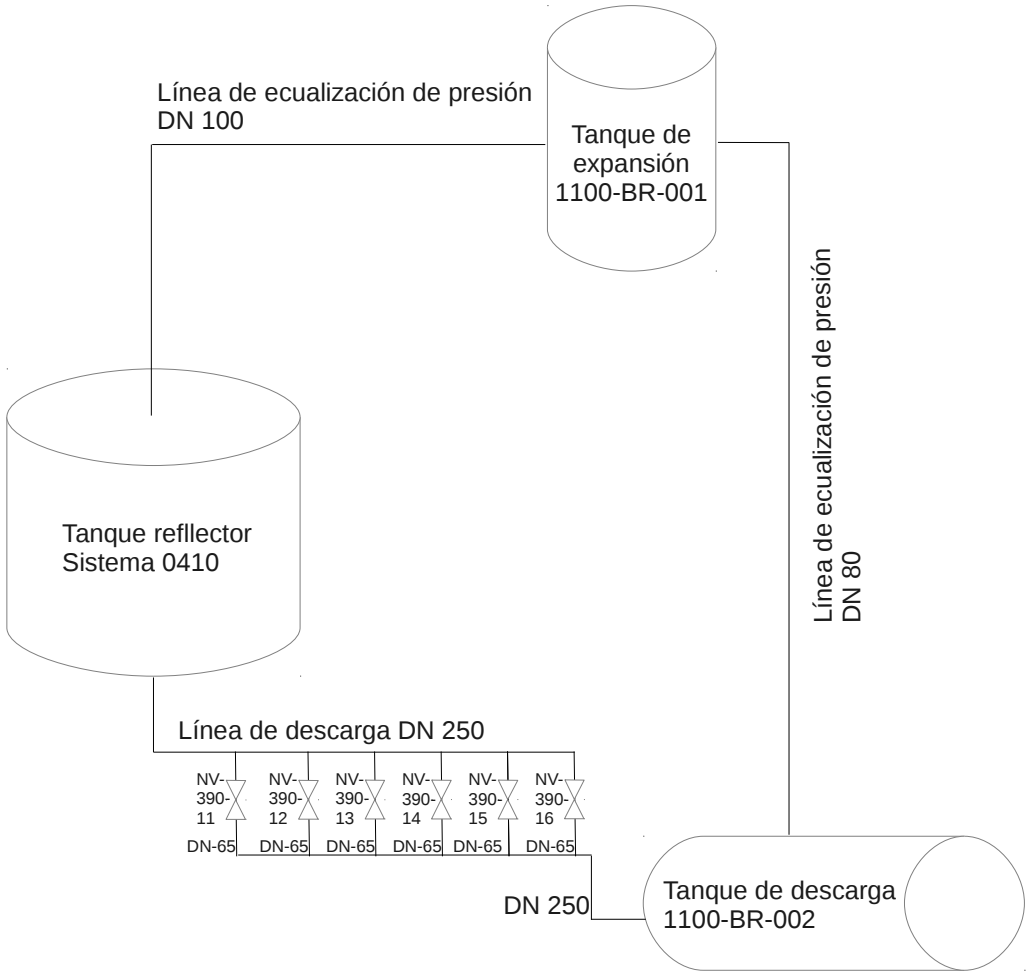


Figura 3.9: Esquema del segundo sistema de parada del reactor RA10.

La Figura 3.9 esquematiza el SSP. En el mismo pueden destacarse tres grandes subsistemas: el tanque del reflector, la red hidráulica de descarga y la red hidráulica de ecualización de presiones. En operación normal del reactor las válvulas que pueden observarse en la red hidráulica permanecen cerradas, y el agua pesada rellena las cañerías y el tanque de reflector. El resto del sistema es rellenado con gas Helio, excepto una porción del tanque de expansión que también permanece rellena con líquido. Cuando es accionado el SSP se abren las válvulas y el líquido comienza a drenar hacia el tanque de descarga, acelerado por la fuerza gravitatoria. Asimismo, el Helio circula en el mismo sentido en el resto del sistema, rellenando el volumen desplazado de líquido.

La simulación del problema completo demandaría elevados recursos computacionales debido a los requerimientos de malla. Por ello se propone desarrollar un modelo multiescala del sistema, desacoplándolo en subsistemas que pueden estudiarse por separado, aplicando la estrategia presentada en este trabajo para acoplarlos mediante condiciones de borde apropiadas. El SSP del RA10 se divide en tres subsistemas:

- Subsistema del tanque del reflector,

- Subsistema de la red hidráulica de descarga,
- Subsistema de la red hidráulica de ecualización de presiones.

Como validación de las herramientas para el cálculo de la evolución de la descarga, se estudia un problema muy similar en el que se conocen datos experimentales del tiempo de vaciado, y sirve a modo de problema modelo para contrastar los resultados obtenidos. Este problema es el vaciado del tanque del reflector del mockup del reactor OPAL, montado por INVAP en San Carlos de Bariloche, [20].

Debido a que el mockup del OPAL está abierto a la atmósfera, la línea de ecualización de presiones no está presente y por lo tanto no es considerada en este estudio.

Subsistemas de estudio

En un estudio previo [21] se analizó el detalle fluídico tridimensional en el tanque reflector durante la descarga, modelando con ecuaciones cero-dimensionales la pérdida de carga en la red hidráulica y acoplando los subsistemas de forma débil. Es de interés ahora estudiar con mayor detalle la distribución de caudales a través del arreglo de válvulas, ya que si bien existen factores de pérdida de carga concentrada tabulados para diferentes geometrías, no existe un valor tabulado para el arreglo de codos y tuberías presente en la configuración de válvulas del SSP. La pérdida de carga se construyó a partir de un modelo sencillo de tés y codos.

Se proponen entonces dos subsistemas de estudio: el primero incluye el tanque del reflector acoplado a una porción de la red hidráulica en la descarga, y el segundo modela el arreglo de válvulas. Ambos están conectados a través de una sección de la tubería, en la cual quedan acoplados los valores de caudal y presión. El subsistema tanque del reflector tiene como incógnitas la presión p_1^1 y el caudal Q_1^1 en la interfaz de acople I_1^1. Asimismo, el subsistema arreglo de válvulas tiene como incógnitas p_2^1 y Q_2^1 en I_{2_1}. Las ecuaciones de continuidad implican que:

$$\left\{ \begin{array}{rcl} p_1^1 & = & p_2^1 \\ Q_1^1 & = & Q_2^1 \end{array} \right. \tag{3.6}$$

Se utiliza la siguiente estrategia: condiciones de borde de tipo *Neumann* en la interfaz de acople para el subsistema tanque del reflector, y condiciones de borde de tipo *Dirichlet* para el subsistema arreglo de válvulas. Las ecuaciones de residuos quedan entonces:

$$\left\{ \begin{array}{rcl} (R_{p,Q})_1^1 \, (p_1^1) & = & 0 \\ (R_{p,Q})_2^1 \, (Q_2^1) & = & 0 \end{array} \right. \tag{3.7}$$

Parámetro	Valor
A_T	$5.30\ m^2$
A_D	$0.05\ m^2$
Δh_{red}	$h + 4.98\ m$
l_D	$11.98\ m$
p_{atm}	$92000\ Pa$
ρ	$998\ Kg/m^3$
D	$0.254\ m$
$\sum_i K_i$	1.13

Tabla 3.1: Parámetros del subsistema del tanque del reflector con acople de sección de red hidráulica

El primer subsistema se analiza con ecuaciones cero-dimensionales, realizando balances de masa y energía. La evolución de la altura h de la superficie libre en el tanque del reflector queda modelada a través de la siguiente ecuación [22]:

$$\ddot{h}h + \frac{\dot{h}^2}{2}\left(1 - \left(\frac{A_T}{A_D}\right)^2\right) + g\Delta h_{red} + \ddot{h}l_D = \frac{p_{atm} - p_1^1}{\rho} + \Delta\hat{u} \tag{3.8}$$

donde p_1^1 es la presión en la interfaz de acople, que se recibe como dato de contorno, A_T es la área transversal del tanque del reflector, A_D es la sección transversal de la línea de descarga, Δh_{red} es la altura total de la columna de líquido en el subsistema, l_D es la longitud total de cañerías en el subsistema, p_{atm} es la presión sobre la superficie libre, y ρ es la densidad del agua. Δu representa la pérdida de carga por unidad de masa y puede modelarse como:

$$\Delta u = \frac{1}{2}{v_D}^2\left(\frac{f_D l_D}{D} + \sum_i K_i\right) \tag{3.9}$$

donde v_D es la velocidad del fluido en la línea de descarga, (que puede escribirse en términos de $\dot{h}$), $\frac{f_D * l_D}{D}$ es el factor de pérdida de carga distribuida en las tuberías, (en función del factor de Darcy f_D, la longitud de tuberías l_D y el diámetro de las mismas D) y $\sum_i K_i$ es la sumatoria de factores de pérdida de carga concetrada.

La Tabla 3.1 reúne los parámetros del subsistema. Los datos geométricos pueden consultarse en las referencias [20]. El factor de pérdida de carga concentrada fue calculado en función de estos datos geométricos [13], e incluye la contracción abrupta en la unión entre el tanque y la red hidráulica, y tres codos de 90º presentes en ella, previos al arreglo de válvulas.

Una vez resuelta la ecuación (3.8) para un dado valor de tiempo, el caudal de descarga Q_1^1 puede calcularse simplemente como:

$$Q_1^1 = -\dot{h}A_D \tag{3.10}$$

El subsistema arreglo de válvulas es modelado con una malla tridimensional de elementos tetraédricos realizada en **Salomé** [23]. El caudal ingresa a través del extremo superior y se reparte entre los múltiples caños que comunican los colectores. En operación normal del reactor cada uno de ellos está bloqueado mediante una válvula esférica, y del otro lado las cañerías están rellenas de gas, pero durante el accionamiento del sistema de parada las mismas se abren dejando pasar libremente al fluido. Las válvulas esféricas instaladas no presentan pérdidas de carga concentrada y por lo tanto no son modeladas. Como es de interés el análisis ante falla simple del sistema, se supone que una de las válvulas no abre y por ello ese caño tampoco se modela. Como otra simplificación del problema se supone que inicialmente el agua rellena todas las cañerías en forma estática. Más adelante se estudia la validez de éstas aproximaciones. Los datos dimensionales de las cañerías pueden consultarse en las referencias [20].

Debido a que el régimen del fluido es turbulento durante la mayor parte de la descarga, y una simulación DNS demandaría elevados recursos computacionales, se utiliza el modelo de turbulencia de tipo RANS $\kappa - \epsilon$ para modelar la fricción interna del fluido. Las ecuaciones se estabilizan mediante un método de control de coeficientes [24]. El sistema de ecuaciones resultantes en el segundo subsistema es:

$$\begin{cases} \dfrac{\partial \bar{U}}{\partial t} + (\bar{U}\cdot\nabla)\bar{U} + \dfrac{\nabla P^*}{\rho} - \nabla\cdot\left[(\nu+\nu_T)\left(\nabla\bar{U}+\nabla U^T\right)\right] - \bar{f} &= 0 \\ \nabla\cdot\bar{U} &= 0 \\ \nu_T - c_\mu\dfrac{\kappa^2}{\epsilon} &= 0 \\ \dfrac{\partial \kappa}{\partial t} + (\bar{U}\cdot\nabla)\kappa - \dfrac{c_\mu}{2}\kappa^2\epsilon\left|\nabla\bar{U}+\nabla\bar{U}^T\right|^2 - \nabla\cdot\left(c_\mu\dfrac{\kappa^2}{\epsilon}\nabla\kappa\right) + \epsilon &= 0 \\ \dfrac{\partial \epsilon}{\partial t} + (\bar{U}\cdot\nabla)\epsilon - \dfrac{c_1}{2}\kappa\left|\nabla\bar{U}+\nabla\bar{U}^T\right|^2 - \nabla\cdot\left(c_\epsilon\dfrac{\kappa^2}{\epsilon}\nabla\epsilon\right) + c_2\dfrac{\epsilon}{\kappa} &= 0 \end{cases} \tag{3.11}$$

donde $\bar{f}$ es una fuerza volumétrica, κ es la energía cinética turbulenta, ϵ es la disipación viscosa de energía turbulenta, ν_T es la viscosidad turbulenta y P^* es la presión efectiva del sistema, que se calcula como $P^* = P + \dfrac{2}{3}\kappa$. Las variables mayúsculas refieren a valores medios estadísticos. Los parámetros de las ecuaciones de transporte de κ y ϵ toman los siguientes valores: $c_\mu = 0{,}09$, $c_1 = 0{,}126$, $c_2 = 1{,}92$ y $c_\epsilon = 0{,}07$ [25].

Para evitar la resolución de la capa límite en las paredes de las tuberías se implementa un modelo de pared, en el que se reemplaza la misma por una tracción tangencial equivalente a la que realizaría la misma sobre la corriente externa [26]. Este modelo impone condiciones de tipo *Dirichlet* para κ y ϵ en la frontera en que se impone la ley de pared. Las condiciones de borde al ingreso de la cañería dependen del valor Q_2^1 impuesto, que define las velocidades del fluido. En base a estas velocidades se calcula un valor para la intensidad turbulenta I_T, y con ella se aproximan los valores de κ y ϵ

en la interfaz. En la descarga de la cañería se impone una fuerza normal que depende de la presión atmosférica.

El sistema de ecuaciones (3.11) es resuelto en pasos fraccionados [27] mediante una formulación de elementos finitos con elementos lineales, estabilizada mediante *SUPG* [16] y *PSPG* [17]. En el primer paso fraccionado se resuelve el transporte de κ, en el segundo paso se resuelve el transporte de ϵ, y en el último paso se resuelven en forma monolítica las ecuaciones de *Navier-Stokes*.

Una vez resueltas las ecuaciones es posible calcular el valor de la presión p_2^1 en la interfaz I_{2_1}:

$$p_2^1 = P^*_{I_2^1} - \frac{2}{3}\kappa_{I_2^1} \tag{3.12}$$

Resultados del cálculo

Se realizan cálculos utilizando mallas del modelo tri-dimensional con diferente refinamiento para estudiar la convergencia de los resultados. La primera es una malla con $\Delta x = 0{,}01m$ y 1145659 de elementos. La segunda es malla tiene $\Delta x = 0{,}008m$ y 1806202 elementos. La tercera es la malla más fina y tiene $\Delta x = 0{,}005m$ y 2951259 elementos. Se utiliza $\Delta t = 0{,}01s$ en los cálculos con las dos primeras mallas y $\Delta t = 0{,}005s$ en los cálculos con la última malla. Las ecuaciones de residuos se resuelven estudiando diferentes métodos numericos.

En la Figura 3.10 se observa la evolución de la altura de la superficie libre del líquido en el tanque obtenida en las dos simulaciones. Comparativamente se reportan también los valores obtenidos acoplando el modelo cero-dimensional del tanque a un modelo cero-dimensional que modela la pérdida de carga en el arreglo de válvulas, y los resultados del cálculo en el que se modela el transporte de gas en las cañerías (se comenta en el siguiente apartado). También se muestran los valores experimentales reportados en la referencia [20].

Puede notarse cómo la evolución simulada modela la dinámica hallada experimentalmente. Durante los primeros segundos los resultados del cálculo imitan al experimento. Luego comienzan a separarse. Sobre el final ésto puede explicarse considerando la geometría del tanque del reflector. En el tanque existe un cajón que envuelve la entrada a la red hidráulica y no permite el vaciado más allá de los 60 cm, por lo que el nivel de líquido, que es medido fuera de este cajón, tiende asintóticamente a este valor. Esta dinámica no es considerada en el modelo cero-dimensional del tanque.

Análisis de sensibilidad de resultados ante válvula en falla

Los cálculos previos se realizaron suponiendo que falla la válvula de la última conexión entre los colectores. Es de interés conocer si existe variación en los tiempos de

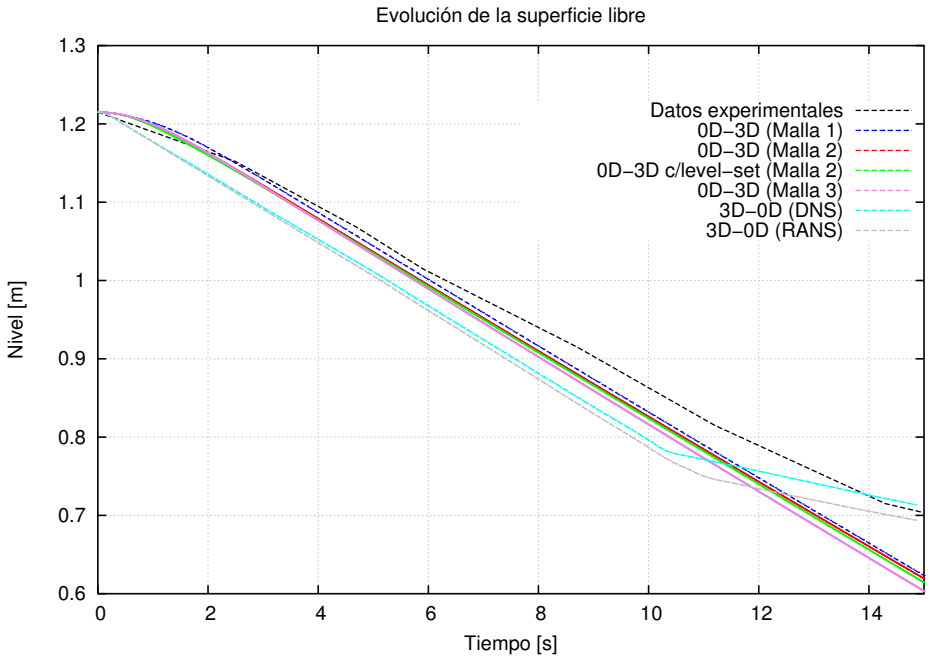


Figura 3.10: Evolución del nivel de líquido en el mockup del tanque del reflector ante accionamiento del SSP.

descarga si la válvula que falla es alguna otra. En la Figura 3.11 se compara la evolución de la superficie libre ante fallas en la primera, la tercera y la sexta válvula. Como puede observarse no se notan diferencias considerables en la evolución de la descarga. La pérdida de carga total del arreglo de válvulas apenas llega a ser sensible a la válvula que falla.

Transporte de superficie libre en las tuberías

Como se comentó, en los cálculos realizados previamente no se consideró el gas de relleno en las tuberías durante los primeros instantes del drenado. Es de interés estudiar su influencia. Se utiliza la técnica de level-set para transportar la superficie libre [28]. Para ello se añade un paso fraccionado extra al sistema de ecuaciones (3.11):

$$\left\{ \frac{\partial \phi}{\partial t} + (\bar{u} \cdot \nabla)\phi = 0 \right. \tag{3.13}$$

donde ϕ es el campo que representa la distancia con signo de cada punto a la superficie libre. Las porciones del sistema con líquido tienen ϕ positivo y las porciones con gas tienen ϕ negativo. ϕ tiene valor nulo en la superficie libre. La ecuación (3.13) requiere un valor de contorno allí donde $\bar{u} \cdot \bar{n} < 0$, y por lo tanto debe proveerse el valor del campo a la entrada de la tubería. Esta ecuación también es resuelta mediante una formulación de elementos finitos con elementos lineales y estabilización *SUPG*. Se utiliza, además, un enriquecimiento del espacio de presiones en los elementos de la interfaz [29]. El campo

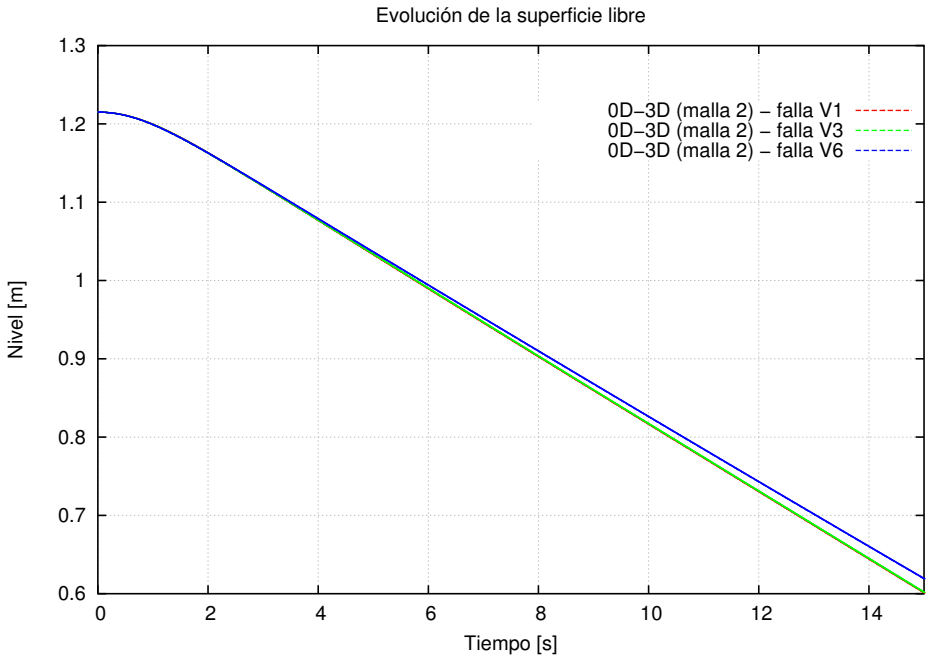


Figura 3.11: Comparación de la evolución del nivel de líquido en el mockup del tanque del reflector ante accionamiento del SSP considerando falla simple en diferentes válvulas.

del level set es reinicializado mediante cálculos geométricos cada 10 pasos temporales.

En la Figura 3.10 se compara la evolución obtenida de la superficie libre con los resultados anteriores, y en la Figura 3.12 se compara la evolución durante el transitorio inicial. Puede observarse que al modelar el transporte del gas la descarga se acelera durante el primer instante, debido a la menor pérdida de carga. Sin embargo, este efecto no tiene mayor peso. La evolución posterior es similar a la obtenida sin el modelado de la superficie libre, y por lo tanto la aproximación realizada inicialmente es conservativa, ya que considera una mayor pérdida de carga.

En la Figura 3.13 se observa la evolución de la superficie libre durante los primeros instantes de tiempo.

p vs q

all eval

d2n eval

extrapol

3.3. Resolución de grandes redes hidráulicas

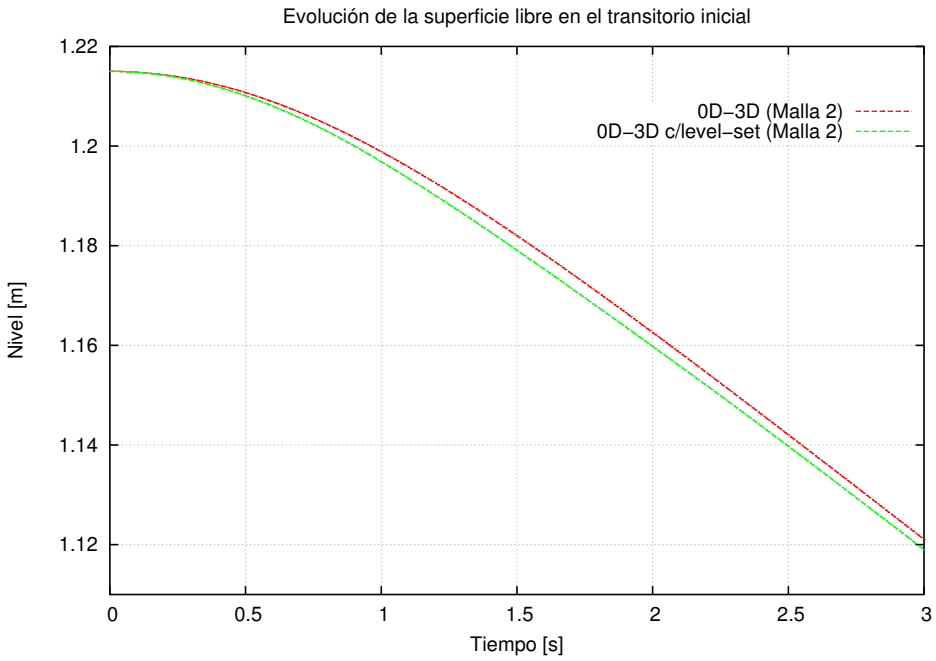
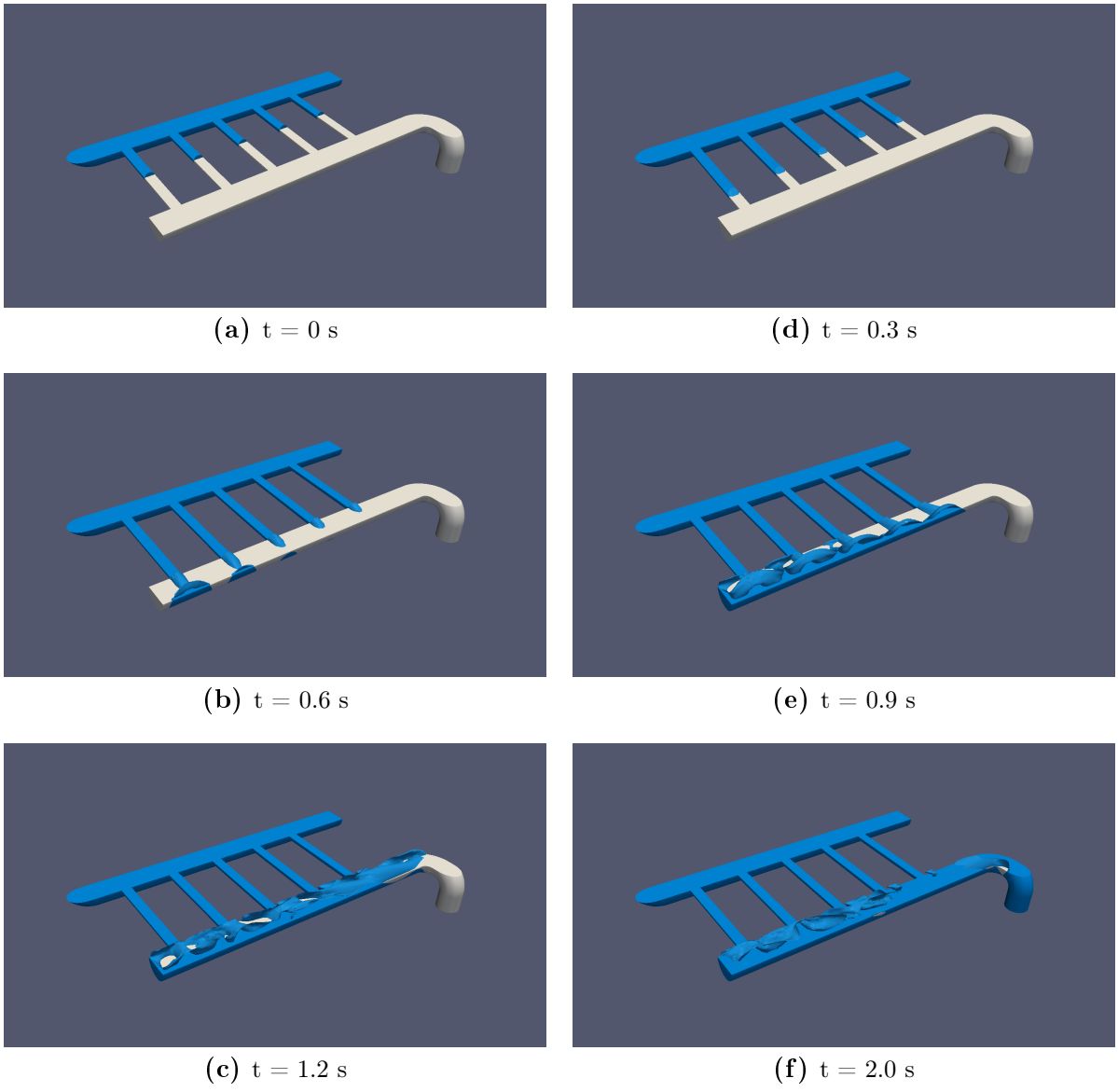


Figura 3.12: Evolución del nivel de líquido en el mockup del tanque del reflector ante accionamiento del SSP durante el transitorio inicial. Se comparan la solución obtenida despreciando el gas en la cañería y la obtenida con transporte de superficie libre mediante la técnica de *level-set*.



(a) t = 0 s **(d)** t = 0.3 s

(b) t = 0.6 s **(e)** t = 0.9 s

(c) t = 1.2 s **(f)** t = 2.0 s

Figura 3.13: Transitorio inicial de la descarga del tanque a través del arreglo de válvulas, con falla simple en la última válvula (no se modela). El corte horizontal en la geometría permite observar el detalle de la evolución de la superficie libre. El líquido (azul) se encuentra inicialemente en condición estática rellenando las cañerías hasta la posición de las válvulas. Al otro lado el gas (blanco) rellena el resto de la red hidráulica.

Capítulo 4

Extensión al problema neutrónico-termohidráulico

“”

— alguien, alguna vez

4.1. Descripción del código Newton

con toda la experiencia ganada se desarrolló el código newton, gral libre, open source, effiecietn error handling, etc.

features: formas de conexión mappers, alphas betas gammas y deltas,

Con la idea de extender estas capacidades al acoplamiento de programas cuyos códigos fuente no estuvieran disponibles, así como a programas cuyos cálculos no dependieran exclusivamente de variables seteadas como condiciones de borde, sino de otros parámetros generales del sistema, se desarrolló un código más genérico de acoplamiento, el código **Newton** [10], publicado en **Github** [30] bajo Licencia Pública General (GPL por sus siglas en inglés, *Public General License*) del Proyecto GNU colaborativo de software libre. **Newton** es descripto en el Capítulo **??**.

4.2. Acople neutrónico-termohidráulico

breve descripción de los códigos utilizados

acople relap fermi

acople fermi cr

acople relap puma

esquema de variables intercambiadas y mapeos

Capítulo 5

Conclusiones

"The trouble with the world is that the stupid are cocksure and the intelligent are full of doubt."

— Bertrand Russell, 1872-1970

Bibliografía

[1] Leiva, J. S., Buscaglia, G. C. Estretegias de acoplamiento entre códigos 0d/1d and códigos cfd 3d. *Mecánica Computacional*, **XXV**, 53–82, 2006. 1

[2] P.J. Blanco, J. L., Feijóo, R., Buscaglia, G. Black-box decomposition approach for computational hemodynamics: One-dimensional models. *Computer Methods in Applied Mechanics and Engineering*, **200**, 1389–1405, 2010. 1

[3] Leiva, J. S., Blanco, P. J., Buscaglia, G. C. Partitioned analysis for dimensionally-heterogeneous hydraulic networks. *Multiscale Model. Simul.*, **9**, 872–903, 2011. 1

[4] Quarteroni, A. Introduction to domain decomposition methods. *6th Summer School in Analysis and Applied Mathematics*, 2011. 2

[5] Caccia, F. A., Dari, E. A. Acoplamiento multiescala en cálculos fluidodinámicos. *Mecánica Computacional*, **XXXIV**, 1955–1972, 2016. 5

[6] Jorge S. Leiva, P. J. B., Buscaglia, G. C. Iterative strong coupling of dimensionally heterogeneous models. *International Journal for Numerical Methods in Engineering*, **81**, 1558–1580, 2009. 5

[7] Dennis, J., Shnabel, R. Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Society for industrial and applied mathematics, 1996. 6

[8] Broyden, C. A class of methods for solving nonlinear simultaneous equations. *American Mathematical Society*, **19**, 577–593, 1965. 6

[9] Hendrickson, B., Devine, K. Dynamic load balancing in computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, págs. 485–500, 2000. 9

[10] Newton: a multiphysics coupling master code. *Repositorio GitHub:* https://github.com/fedecaccia/newton, 2017. 10, 35

[11] Calviño, B. O. Estrategias de descomposición en dominios para entornos grid. *Tesis Doctoral, Universidad Politécnica de Cataluña*, 2007. 11

[12] Buscaglia, G. C., Dari, E. A., Martín, E. J., Arnica, D. L., Bonetto, F. J. Finite element modeling of liquid deuterium flow and heat transfer in a cold-neutron source. *International Journal of Computational Fluid Dynamics*, **18**, 355–365, 2004. 19

[13] Iedelchik, I. Handbook of Local Resistance and Friction. 1960. 21, 27

[14] Kays, W., Crawford, M. Convective Heat and Mass Transfer, 3rd Edition. 1993. 21, 22

[15] Gunzburger, M. D. Finite Element Methods for Viscous Incompressible FLows. 1989. 22

[16] Codina, R. Comparison of some finite element methods for solving the diffusion-convection-reaction equation. *Computer Methods in Applied Mechanics and Engineering*, **156**, 185–210, 1998. 22, 29

[17] Hughes, T., Franca, L., Balestra, M. A new finite element formulation for computational fluid dynamics: V. circumventing the babuška-brezzi condition: a stable petrov-galerkin formulation of the stokes problem accomodating equal-order interpolations. *Computer Methods in Applied Mechanics and Engineering*, **59**, 85–99, 1986. 22, 29

[18] Geuzaine, C., Remacle, J. F. Gmsh reference manual, 2016. 22

[19] Buscaglia, G. C., Dari, E. A. Numerical investigation of flow through a cavity with internal heat generation. *Numerical Heat Transfer*, págs. 525–541, 2003. 22

[20] Invap. Segundo sistema de parada - validación del cfd para drenaje del tanque reflector, 2013. 26, 27, 28, 29

[21] L.M. Rechiman, F. C. A. C., M.I. Cantero, Dari, E. Three-dimensional hydrodynamic modeling of the second shutdown system of an experimental nuclear reactor. *Nuclear Engineering and Design*, **319**, 163, 2017. 26

[22] Bird, R., Stewart, W., Lightfoot, E. Transport Phenomena, 2nd Edition. 2002. 27

[23] CEA/DEN, E. R., CASCADE, O. Salome version 7.8.0: Public release announcement, 2016. 28

[24] Lew, A. J., Buscaglia, G. C., Carrica, P. M. A note on the numerical treatment of the k-epsilon turbulence model. *International Journal of Computational Fluid Dynamics*, 2001. 28

[25] Durbin, P., Reif, B. P. Statistic Theory and Modeling for Turbulent Flows. 2011. 28

[26] Mohammadi, B., Pironneau, O. Analysis of the K-Epsilon TURBULENCE MODEL. 1994. 28

[27] Lew, A. J. El método de elementos finitos en entornos computacionales de alta performance. *Trabajo Especial Carrera de Ingeniería Nuclear, Instituto Balseiro*, págs. 57–64, 1998. 29

[28] Osher, S., Fedkiw, R. Level Set Methods and Dynamic Implicit Surfaces. 2003. 30

[29] Ausas, R. F., Buscaglia, G. C., Idelsohn, S. R. A new enrichment space for the treatment of discontinuous pressures in multi-fluid flows. *International Journal for Numerical Methods in Fluids*, 2011. 30

[30] GitHub, Inc. Plataforma de desarrollo colaborativo. *https://github.com*, 2008. 35

Publicaciones asociadas

- Informes técnicos en Comisión Nacional de Energía Atómica:
  - Rechiman, L.; Cantero, M.; Dari, E.; Caccia, F.; Chacoma, A. 2015. “Análisis hidrodinámico del Segundo Sistema de Parada del reactor RA10”. Reporte técnico CNEA IN-ATN40MC- Mayo de 2015, Bariloche, Argentina.
- Publicaciones en revistas internacionales:
  - Rechiman, L.; Cantero, M. ; Caccia, F.; Chacoma, A. and Dari, E. 2017. “Three-dimensional hydrodynamic modeling of the second shutdown system of an experimental nuclear reactor” ("Modelado hidrodinámico tridimensional del segundo sistema de parada de un reactor nuclear experimental"), Nuclear Engineering and Design, vol. 319, pp 163-175.
- Presentaciones en congresos con publicación en actas:
  - Caccia, F. and Dari. E. “Acoplamiento multiescala en cálculos fluidodinámicos”. Mecánica Computacional, vol XXXIV, págs 1955-1972.
  - Rechiman, L.; Chacoma, A.; Caccia, F.; Dari E. and Cantero, M. "Validation of a multiscale model of the second shutdown system of an experimental nuclear reactor“ ("Validación del modelo multiescala del segundo sistema de parada del reactor nuclear experimental RA-10”). Mecánica Computacional, vol XXXIV, págs 2199-2215.