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)

17 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) [**?**], P.J. Blanco et al. (2010) [**?**] y J. S. Leiva et al. (2011) [**?**].

1.2. Abordaje del modelado

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 [**?**]). 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 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. Se tiene que:

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

[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:

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

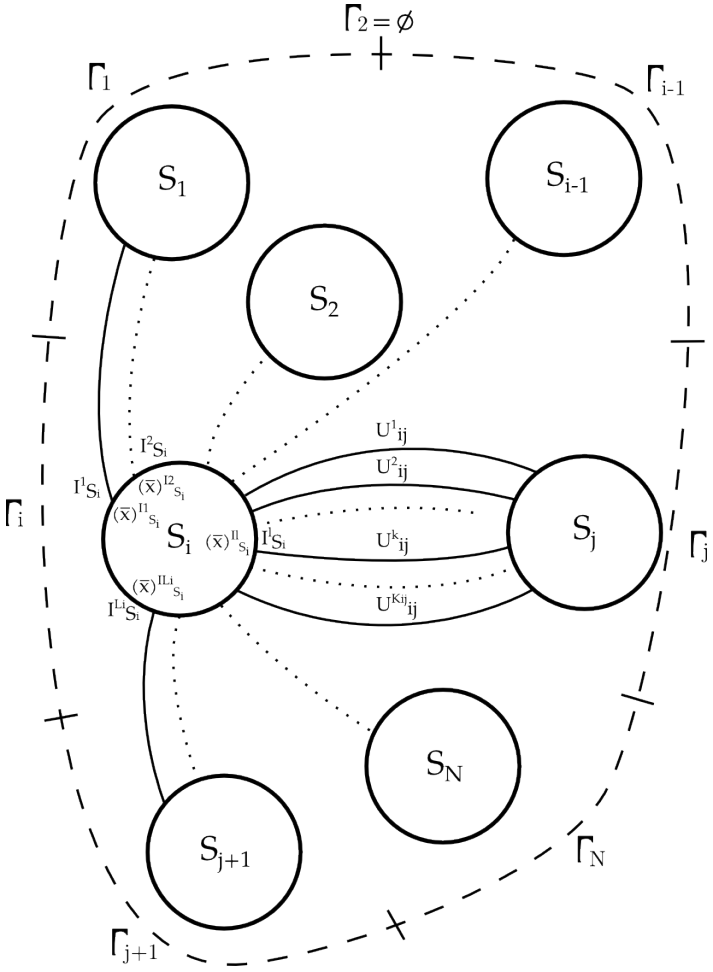

Figura 1.1: Esquema de subsistemas de estudio relacionados mediante condiciones de borde dinámicas en interfaces de acoplamiento.

donde ($\alpha_i(\Gamma_i)$ representa las condiciones de borde impuestas sobre la curva Γ_i. Estas relaciones, básicamente, mapean condiciones de borde de un tipo, que son impuestas como datos, en condiciones de borde de otro tipo. Notar que algunas de las dependencias pueden anularse dependiendo del modelo de estudio utilizado en cada subsistema. 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.

Sistema de ecuaciones a resolver

La estrategia de acoplamiento propuesta es una estrategia iterativa. En primera instancia se propone un valor *guess* para todas las incógnitas $(x_{m,guess})^{I_l}_{S_i}$, considerando

las relaciones de continuidad 1.1. Luego se seleccionan una serie de ecuaciones 1.2 y se resuelven mediante la ejecución de códigos especíicos, obteniendo los valores $(x_{m,calc})_{S_i}^{I_l}$. Las ecuaciones modelos seleccionadas junto con las ecuaciones de continuidad deben definir un problema bien planteado[2]. Finalmente, se computan las diferencias entre los valores *guess* y los valores calculados, obteniendo los residuos $(R_m)_i^l$:

$$(R_m)_i^l = (x_{m,guess})_{S_i}^{I_l} - (x_{m,calc})_{S_i}^{I_l} \tag{1.3}$$

donde m es el índice de incógnita, l de la intefaz e i del subsistema. La convergencia fuerte de los subsistemas acoplados requiere que estos residuos sean nulos, y por lo tanto el método iterativo utilizado debe buscar el siguiente resultado:

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

donde $\bar{R}$ es el vector de residuos de las ecuaciones.

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 su sencillez de implementación, ya que es un método explícito y por tanto, los valores calculados por algunos subsistemas 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 $[c, L]$. Se va a resolver la

[2] 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.

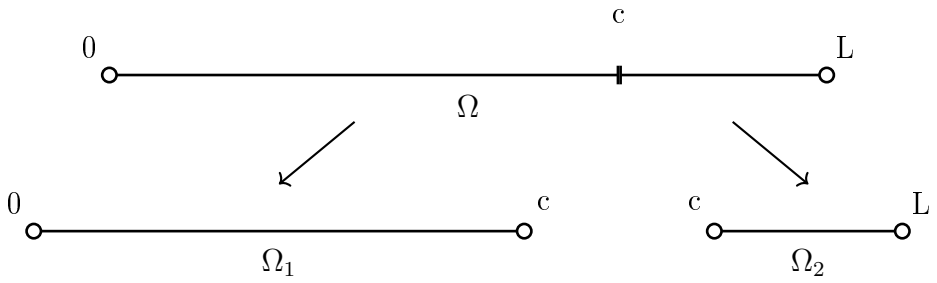

Figura 1.2: Descomposición disjunta de dominios en el cálculo del campo de temperatura a lo largo de una barra unidimensional.

ecuación 1.5 en cada uno de ellos, pero la condición de borde *Dirichlet* ahora solo aplica sobre el borde original del dominio. Para que cada problema quede bien planteado es necesario imponer una condición de borde extra en el punto de descomposición. En el método *Dirichlet-to-Neumann*, es necesario decidir qué subsistema va a ser resuelto en primera instancia. En este ejemplo se decide el subsistema de la izquierda. Ahora es necesario decidir qué tipo de condición de borde se le va a aplicar. 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. En el método *Dirichlet-to-Neumann*, este valor de flujo luego debe ser utilizado para ser impuesto 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. 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 (aquí solo existe una única variable de estado, la temperatura).

Otra desventaja de este método es que en general requiere demasiadas iteraciones para converger [**?**]. 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 [**?**]).

Otra forma de resolver el problema es aplicando métodos para encontrar raíces a funciones vectoriales no lineales (sistema de ecuaciones de residuos 1.4), 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 $\bar{x} = \bar{x}_{n+1}$ 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 $\bar{x}_n$, cuyo elemento J_{ij} debe evaluarse como $J_{ij} = \frac{\partial R_i}{\partial x_j}$. Suponiendo que $\bar{x}_{n+1}$ 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:

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

Para hallar la solución $\bar{x}_{n+1}$ simplemente hay que resolver el sistema:

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

que es el método de iterativo $Newton - Raphson$ con orden de convergencia cuadrática. 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 [**?**], y por lo tanto presentan una ventaja fuerte 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 estos métodos [**?**]. También existe otra variante del método, el método *Broyden ortonormal*, que asegura la convergencia en una cantidad finita de iteraciones para sistemas lineales [**?**], (la cantidad de iteraciones requerida 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 técnicas de gradientes descendentes.

[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.

Sin embargo, en cada paso de descenso requieren evaluaciones de residuos, y por tanto son altamente costosos.

Notar que los últimos sistemas presentados resuelven sistemas de ecuaciones de forma monolítica, y por lo tanto permiten incluir estrategias prohibidas al método *Dirichlet-to-Neumann*. El ejemplo presentado de cálculo de temperatura podría resolverse imponiendo cualquier combinación de condiciones de borde en las fronteras de acople de cada subdominio, como por ejemplo, fronteras *Dirichlet* para ambos.

Problemas de evolución

En problemas de evolución la estretegia es permitir que cada subsistema avance por separado acoplando sus resultados mediante las ecuaciones 1.4 solo cada ciertos pasos. Esto se permite porque cada subsistema podría requerir subpasos de evolución diferentes[4].

La estrategia de selección de variables que son datos o incógnitas en cada interfaz puede variar en la evolución, ya que pueden seleccionarse diferentes ecuaciones 1.2 en diferentes pasos de acople.

Notar que el método es incondicionalmente estable debido a que en cada paso de evolución 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

[4] Distintos subdominios pueden tener diferentes requisitos sobre el parámetro de evolución, dependiendo de la física implicada. Algunos, por ejemplo, podrían experimentar transitorios fluidodinámicos, en los que es de interés mantener por debajo de algún valor ciertos parámetros numéricos (como el número de *Courant*) proporcionales al paso de tiempo. Otros, en cambio, podrían estar siendo resueltos por algún programa computacional que consume elevados recursos temporales, por tanto sería de interés utilizar subpasos evolutivos mayores.

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 gobernado por fuerzas naturales, que se estudia subdividiéndolo en dos subsistemas, con dos interfaces de acople cada uno. El siguiente sistema de estudio es el vaciado del tanque reflector del reactor de investigación RA-10. Interesa analizar el tiempo 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 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."
— Sir Isaac Newton, 1643-1727

2.1. Paradigma maestro-esclavo

El modelo de comunicación utilizado en el trabajo recibe el nombre de *maestro-esclavo* [?]. 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 dado 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.

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 XYZ en XYZ. **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** [**?**], que será descripto en el Capítulo **??**.

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 procesos [**?**]. El paso de mensajes involucra la transferencia de datos desde un proceso que envía a otro 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 es posible para 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 de entrada del programa *esclavo* todos los parámetros necesarios para la ejecución del

[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*.

cálculo, ordena su ejecución, 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, ya que cada corrida del código *esclavo* solo vive para realizar un cálculo entre dos pasos de evolución acoplados.

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.

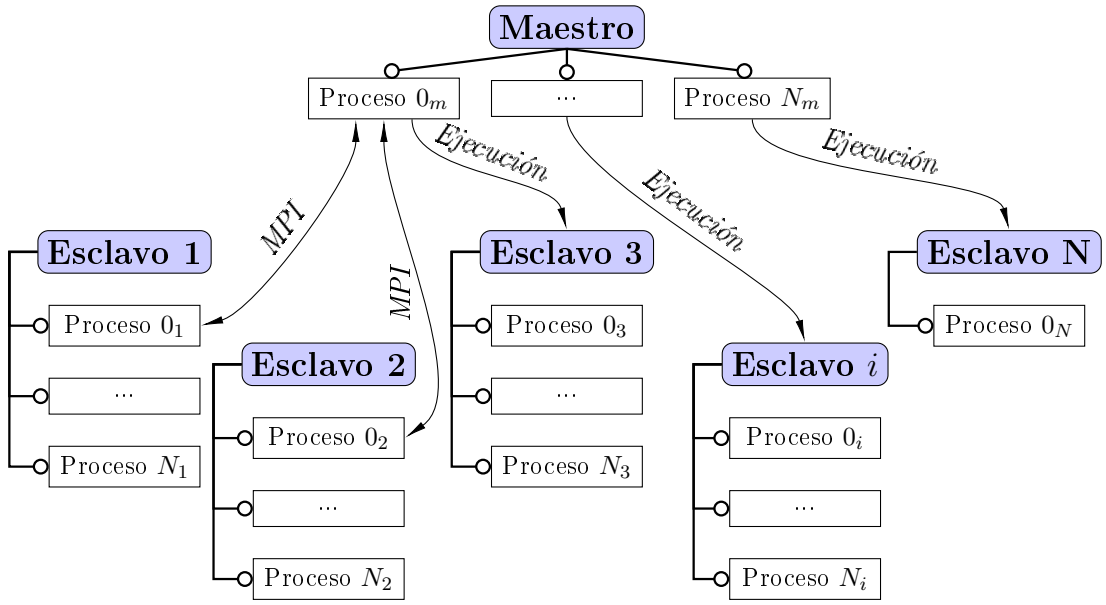

Figura 2.1: Esquema de comunicación entre los programas *esclavos* y el programa *maestro* implementado en el acoplamiento de códigos. Los *esclavos* se comunican solo con el *maestro* y no intercambian datos entre sí. Se utilizan dos modelos de comunicación diferentes. En el primero, cada código *esclavo* 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 cuatro 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.

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. Los problemas que no involucran evolución de variables pueden ser tratados como problemas con un solo paso de evolución. 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, como se explicó en el apartado 1.2 **Problemas de evolución**. 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 la 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 1.4. 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 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

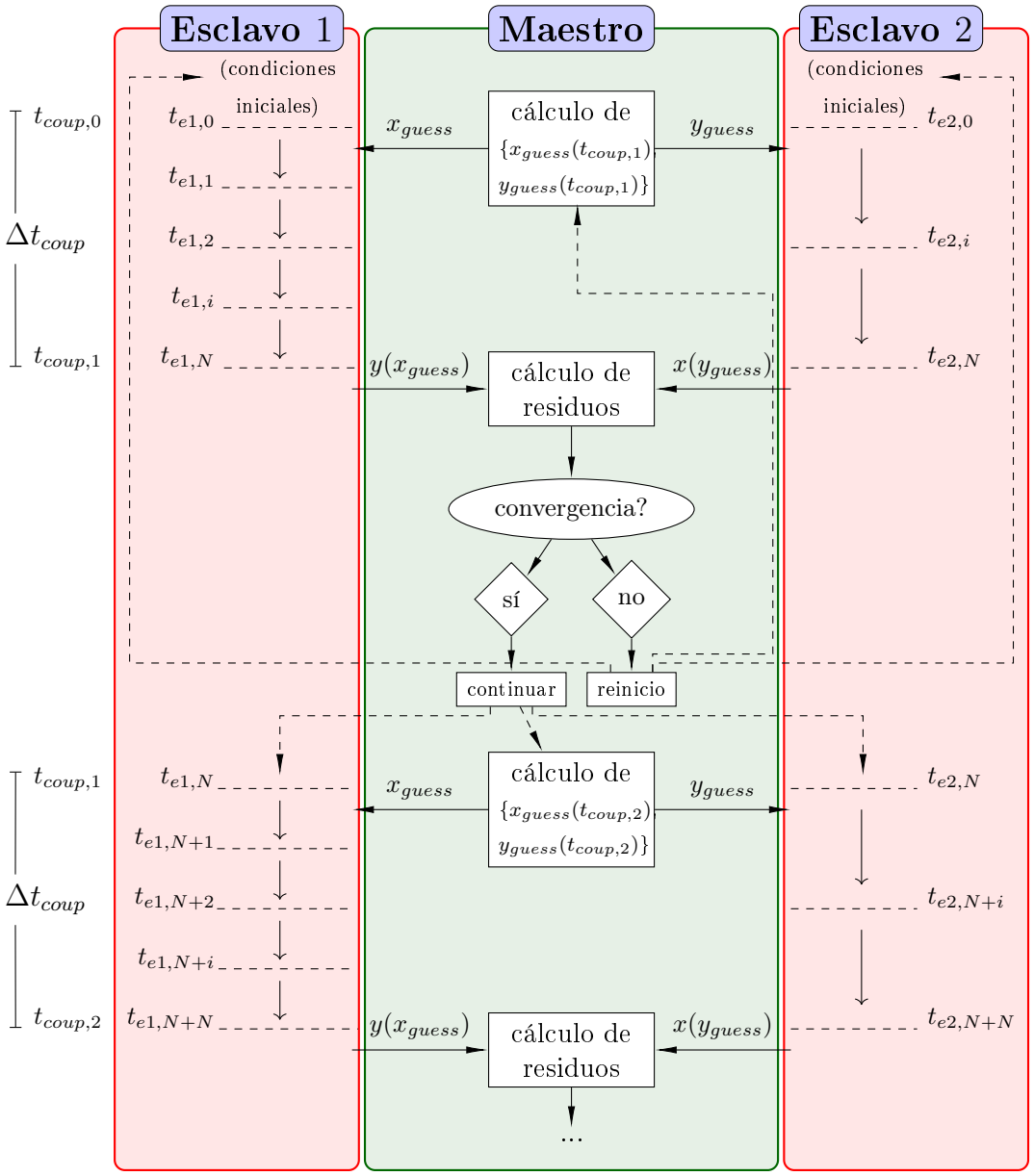

Figura 2.2: Esquema de acoplamento entre el programa *maestro* y los programas *esclavos*. En el ejemplo, cada código *esclavo* resuelve ecuaciones diferenciales en distintos subsitemas. Estos subsistemas están acoplados entre sí en alguna interfaz en las que las variables $\{x, y\}$ son incógnitas. 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. Es necesario resaltar que el esquema también aplica para el caso de comunicación entre programas mediante lectura y escritura de archivos, en el que de cada programa *esclavo* solo vive durante cada Δt_{coup}.

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.

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 [**?**]. En la Figura 3.1 puede apreciarse un diagrama del sistema.

En cada interfaz de acople existen incógnitas de velocidades, fuerzas, temperatura y flujo de calor. En el caso de las velocidades la estrategia implementada es definir una variable integral, el caudal volumétrico, que servirá como una de las variables de acoplamiento[1]. En el caso de las fuerzas se utilizan valores promediados para la fuerza

[1] Cuando se utilizan variables integrales o promediadas para el acoplamiento, es necesario definir una estrategia extra en el subdominio que la recibe. Como cada subproblema solo queda bien definido si la condición de borde está dada sobre todos los puntos del borde, estos valores deben distribuirse

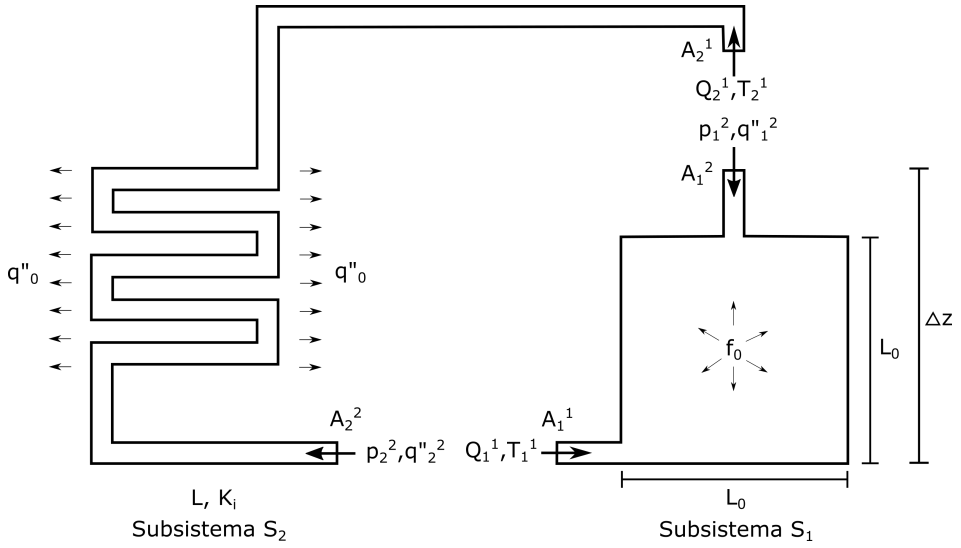

Figura 3.1: Esquema del sistema analizado. El subsistema de la izquierda es un intercambiador de calor y se estudia con un código cero-dimensional. El modelo de la derecha es una cavidad con una fuente de energía interna y se estudia con un código bi-dimensional. El sistema completo es abordado con una 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.

normal (presión). Las fuerzas tangenciales se consideran nulas bajo la hipótesis de que son despreciables en las interfaces de acople. Esta hipótesis es correcta cuando el flujo es paralelo, y por ello se selecciona como interfaz de acople aquella que se corresponda con el perfil de velocidades lo más plano posible, lejos de las curvas. Los valores de temperatura de acople también corresponden a valores promediados en la interfaz, y el flujo de calor corresponde al flujo integral a través de ella. En los subsistemas bi-dimensionales, los perfiles de velocidadades y temperaturas construidos a partir de las variables recibidas se consideran planos, bajo la hipótesis de flujo paralelo. Con esta estrategia, existen cuatro incógnitas en cada interfaz de cada subsistema. Considerando los dos subsistemas, existen en total dieciseis incógnitas. Por lo tanto el sistema queda definido por ocho ecuaciones de continuidad de campos de variables y otras 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:

considerando algún perfil. Por ejemplo, para el caso del subdominio que recibe un valor de caudal, y está modelado con ecuaciones bi-dimensionales, necesita definir un perfil de velocidades a lo largo de toda la sección de acople. La definición del perfil se basa en alguna hipótesis que la persona que está modelando considera adecuada conforme a la física del problema. Este paso debe analizarse con cuidado ya que los resultados del acoplamiento dependen de ello.

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

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 cuatro ecuaciones de residuos [2] del tipo $(R_m)_i^l = 0$:

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

y entre las dos interfaces del subsistema 2 se generan otras cuatro 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

[2] 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 $q"$.

$(R_{p,Q})_2^1 = 0$ 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 con flujo turbulento [**?**] y de transferencia de energía en un intercambiador de calor con flujo constante [**?**]:

$$\begin{cases} p_2^1 + \rho g \Delta z &= p_2^2 + \rho_2^1 \left(\frac{Q_1^1}{A_2^1}\right)^2 \left(\frac{f_D L}{D} + \sum_i K_i\right) \\ T_2^2 &= T_2^1 + 2\dfrac{q_0" L}{\frac{D}{2}\frac{Q_1^1}{A_2^1}\rho c_p} \end{cases} \tag{3.4}$$

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. Esta aproximación es correcta ya que las interfaces se seleccionaron lejos de fuentes y sumideros, donde los gradientes de temperatura son despreciables. Con este modelo, ninguna de las dos ecuaciones puede recibir valores de contorno *Dirichlet* en ambas interfaces, ya que los valores de caudal y temperatura en una interfaz determinan el valor en la otra. Por lo tanto, en la estrategia de acoplamiento, la primera ecuación debe tener, o bien ambos contornos con condiciones de tipo *Neumann*, o bien uno con condición de tipo *Neuman* y otro con condición de tipo *Dirichlet*. La segunda ecuación debe tener uno de los bordes con condición de tipo *Dirichlet* y otro con condición de tipo *Neumann*. Esta condición es necesaria a pesar de que el flujo de calor recibido no va a ser utilizado, basado en la hipótesis de que es despreciable. Si el flujo de calor fuera efectivamente apreciable, debería cambiarse el modelo en la ecuación planteada.

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 [**?**] y de transporte de energía [**?**]. Se utiliza la aproximación de *Boussinesq* considerando variaciones de densidad solo en el término de fuerza volumétrica: