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)

27 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*)

MISD: Múltiples Instrucciones, Un Dato (*Multiple Instruction, Single 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*)

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

SISD: Una Instrucción, Un Dato (*Single Instruction, Single 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, FLUJO MULTIFASE, 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, MULTIPHASE FLOW, 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

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, $i = 1, ..., N$.
- $U_{i,j}^k$ es la unión k entre subsistemas i y j, $k = 1, ..., K_{i,j}$.
- $I_{S_i}^l$ 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.
- $(x_m)_{S_i}^{I_l}$ es el valor de la variable x_m en la interfaz l del subsistema i, $m = 1, ..., M_i$.
- $(\bar{x})_{S_i}^{I_l}$ 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 $U_{i,j}^k$ que relaciona los sistemas S_i y S_j mediante las interfaces $I_{S_i}^{l_1}$ e $I_{S_j}^{l_2}$ respectivamente, define una relación de continuidad[1] entre las incógnitas $(x_m)_{S_i}^{I_{l_1}}$ y $(x_m)_{S_j}^{I_{l_2}}$, de tal forma que:

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

Estas relaciones reducen a la mitad la cantidad de incógnitas. Las demás ecuaciones necesarias para despejar las incógnitas se encuentran a partir del modelo de estudio de cada subsistema. Sean $(F_m)_i^l$ las relaciones funcionales que calculan el valor de las incógnitas $(x_m)_{S_i}^{I_l}$ 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:

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

[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:

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

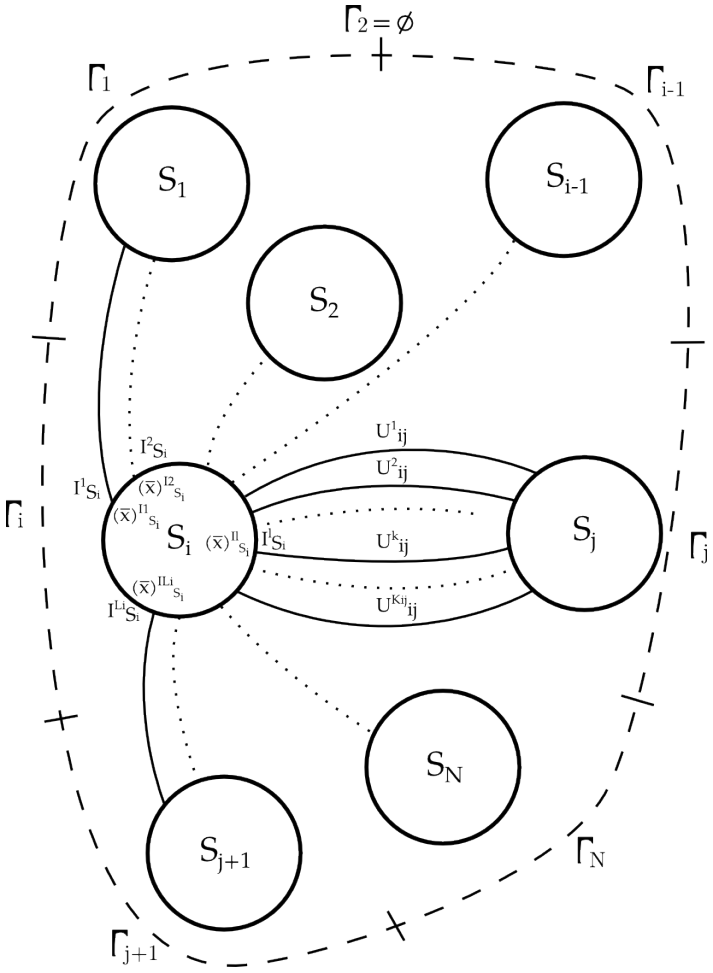


Figura 1.1: Esquema de subsistemas de estudio relacionados mediante condiciones de borde dinámicas en interfaces de acoplamiento.

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 1.2 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.

Estrategia de resolución

A continuación se propone un ejemplo didáctico para comprender las estrategias de acoplamiento posteriormente comentadas. Se desea resolver un problema de cálculo de

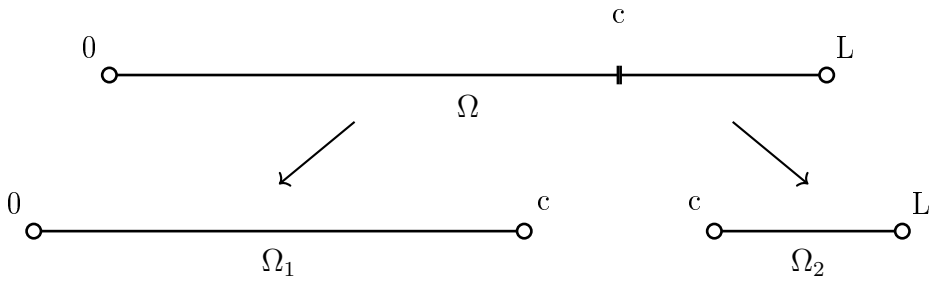


Figura 1.2: Descomposición disjunta de dominios en el cálculo del campo de temperatura a lo largo de una barra unidimensional.

campo de temperatura mediante el Método de Descomposición Disjunta de Dominios. El sistema global de análisis es una barra unidimensional de longitud L con condiciones de borde homogéneas, fuente interna de energía f y conductividad térmica k. El modelo matemático utilizado es el siguiente:

$$\begin{cases} -k\Delta u = f \\ u|_{\partial\Omega} = 0 \end{cases} \tag{1.3}$$

El dominio original $[0, L]$ es particionado en los subdominios $[0, c]$ y $[c, L]$ (ver Figura 1.2). Se va a resolver la ecuación 1.3 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 acople c de cada subsistema. Esta decisión depende del método empleado para resolver el acoplamiento.

La forma clásica de resolución es el método *Dirichlet-to-Neumann*. Este método es un método explícito de simple implementación, que resuelve el acoplamiento mediante iteraciones de tipo *Piccard*. En el método *Dirichlet-to-Neumann*, es necesario decidir qué subsistema va a ser resuelto en primera instancia. Si se decidiera, por ejemplo, comenzar con el subsistema de la izquierda, luego es necesario decidir qué tipo de condición de borde se le va a aplicar. Estas primeras decisiones son arbitrarias. Si se decide 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. Es decir, uno de los valores de las incógnitas en la interfaz de acople se calcula como función del valor de la otra incógnita, que fue supuesto como dato, a partir de la relación 1.2. En este caso la relación recibe el nombre de operador *Steklov-Poincaré* y se define como:

$$(q''_{calc})_{S_1} = \mathscr{N}_1\left((T_{guess})_{S1}\right) \tag{1.4}$$

donde $\mathscr{L}$ es el operador que mapea condiciones de borde de tipo *Dirichlet* en condiciones de borde de tipo *Neumann*, $(T_{guess})_{S1}$ es la temperatura que fue impuesta como

condición de borde en el primer subsistema y $(q''_{calc})_{S_1}$ es el flujo calórico calculado. En base a la relación de continuidad 1.1 este flujo de calor se impone en el segundo subsistema como condición de borde. Esta condición define una temperatura en la interfaz de acople, mediante el segundo operador *Steklov-Poincaré* definido:

$$(T_{calc})_{S_2} = \mathscr{D}_2\left((q''_{calc})_{S1}\right) \tag{1.5}$$

donde $\mathscr{D}$ es el operador que mapea condiciones de borde de tipo *Nemaumann* en condiciones de borde de tipo *Dirichlet*, y $(T_{calc})_{S_2}$ es la temperatura calculada. Si $(T_{calc})_{S_2}$ coincide con $(T_{guess})_{S1}$, los resultados están convergidos y entonces finaliza el cálculo En caso contrario, el proceso debe repetirse, pero ahora se impone T_2 al primer subsistema. El cálculo continúa así hasta que los valores $\{(q''_{calc})_{S_1}, (T_{calc})_{S_2}\}$ convergen en iteraciones contiguas. Si inicialmente se hubiera impuesto una condición de tipo *Neumann* en el primer subdominio, necesariamente al segundo subdominio debería habérsele impuesto una condición de tipo *Dirichlet*, ya que la variable calculada en la interfaz de acople por el primer subsistema hubiera sido una temperatura. 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). Esta característica es un poco restrictiva, ya que no permitiría, por ejemplo, que todos los subproblemas de análisis se resuelvan con condiciones de borde de tipo *Dirichlet*.

Sin embargo esta no es la única desventaja del método *Dirichlet-to-Neumann*. Otra desventaja 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]).

Existe una forma alternativa de resolver el acoplamiento, y es mediante una técnica implícita. Esta técnica es una metodología iterativa, y consiste en la evaluación de residuos sucesivos construidos por diferencia entre valores propuestos y valores calculados. En primera instancia se propone un valor *guess* para todas las incógnitas $(x_{m,guess})_{S_i}^{I_l}$. Estos valores deben respetar las relaciones de continuidad 1.1. En segunda instancia se decide arbitrariamente qué tipos de condiciones de borde va a recibir cada interfaz de cada subsistema, definiendo problemas parciales bien planteados. En el ejemplo analizado, sería posible definir condiciones de borde de tipo *Dirichlet* en ambos subsistemas. En función de esta decisión, se toman los valores correspondientes del vector $\bar{x}_{guess}$ y se establecen como datos para cada problema parcial. Una vez resuelto el problema en cada subdominio, se calcula el valor de las variables que no fueron tomadas como

dato en las interfaces de acople. En el ejemplo, si se había establecido un valor de temperatura como condición de borde de tipo *Dirichlet* en ambas interfaces, se calcula entonces el valor del flujo de calor en la misma interfaz para cada subsistema, mediante los operadores de *Steklov-Poincaré* $\mathscr{N}_1$ y $\mathscr{N}_2$:

$$\begin{cases} (q''_{calc})_{S_1} = \mathscr{N}_1\left((T_{guess})_{S1}\right) \\ (q''_{calc})_{S_2} = \mathscr{N}_2\left((T_{guess})_{S2}\right) \end{cases} \tag{1.6}$$

Al hacer esto se están resolviendo las ecuaciones 1.2, obteniéndose los valores $(x_{m,calc})^{I_l}_{S_i}$. Finalmente, se computan las diferencias entre los valores *guess* y los valores calculados, obteniendo los residuos $(r_m)^l_i$:

$$(r_m)^l_i = (x_{m,guess})^{I_l}_{S_i} - (x_{m,calc})^{I_l}_{S_i} \tag{1.7}$$

donde m es el índice de incógnita, l de la intefaz e i del subsistema. En el ejemplo, las ecuaciones de residuos quedarían:

$$\begin{cases} (r_{q''})^1_{S1} = (q''_{guess})_{S_1} - (q''_{calc})_{S_1} \\ (r_{q''})^1_{S2} = (q''_{guess})_{S_2} - (q''_{calc})_{S_2} \end{cases} \tag{1.8}$$

La convergencia fuerte de los subsistemas acoplados requiere que estos residuos sean nulos, y por lo tanto se busca el siguiente resultado:

$$\bar{r} = \bar{0} \tag{1.9}$$

donde $\bar{r}$ es el vector de residuos de las ecuaciones. Comúnmente en la primera iteración esto no sucede, y por lo tanto es necesario repetir el proceso sucesivas veces hasta obtener convergencia. En general los sistemas acoplados se modelan con sistemas de ecuaciones no lineales y por ello se investigan distintos métodos numéricos para la resolución de 1.9.

Métodos numéricos para la resolución de sistemas de ecuaciones de residuos

Existen diferentes métodos numéricos para hallar las raíces del sistema de ecuaciones 1.9. 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:

$$\bar{r}(\bar{x}_{n+1}) = \bar{r}(\bar{x}_n) + \nabla\bar{r}(\bar{x}_n)(\bar{x}_{n+1} - \bar{x}_n) \tag{1.10}$$

donde $\nabla\bar{r}(\bar{x}_n)$ es la matriz jacobiana del sistema $\mathbb{J}$ 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 1.10 y operando algebraicamente se llega a la siguiente expresión:

$$\bar{r}(\bar{x}_n) = -\mathbb{J}(\bar{x}_n)(\bar{x}_{n+1} - \bar{x}_n) \tag{1.11}$$

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

$$\bar{x}_{n+1} = \bar{x}_n - \mathbb{J}^{-1}(\bar{x}_n)\bar{r}(\bar{x}_n) \tag{1.12}$$

que es el método de iterativo *Newton-Raphson*. La principal ventaja de este método es que tiene orden de convergencia cuadrática, pero la desventaja 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 evaluaciones. Cada elemento $J_{ij} = \frac{\partial R_i}{\partial x_j}$ puede aproximarse mediante diferencias finitas a primer orden como:

$$J_{ij} \approx \frac{r_i(\bar{x}_j + \Delta\bar{x}_j) - r_i(\bar{x}_j)}{\|\Delta x_j\|} \tag{1.13}$$

donde $\Delta\bar{x}_j$ es un vector que contiene valores nulos en todos sus elementos excepto en el elemento de la posición j, cuyo valor es la diferencia incremental para esa variable j. 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_{unk} evaluaciones extras, donde N_{unk} es la cantidad total de incógnitas. Es decir, en total, se requieren $N_{unk} + 1$ evaluaciones. Si bien estas evaluaciones son independientes entre sí y por lo tanto altamente paralelizables, este método requeriría excesivos recursos de cálculo.

Una forma alternativa y elegante de resolver el sistema de ecuaciones planteado en 1.9 es mediante métodos *quasi-Newton*. La característica principal de estos métodos es que aproximan la matriz jacobiana sin necesidad de realizar evaluaciones extras, y por lo tanto tienen una ventaja fuerte frente al método de *Newton-Raphson*. Además, tienen convergencia superlineal [7], y por lo tanto también presentan una ventaja frente a los métodos mediante iteraciones de *Piccard*.

El método de *Broyden* es uno de estos métodos [8]. La matriz jacobiana $\mathbb{B}_n$ en la iteración n es aproximada mediante una matriz $\mathbb{B}_n$ que se construye solo en función del valor de los vectores incógnitas $\bar{x}_{n-1}$ y $\bar{x}_n$, de los vectores residuos $\bar{r}_{n-1}$ y $\bar{r}_n$, y de la matriz de la iteración previa $\mathbb{B}_{n-1}$:

$$\mathbb{B}_n = \mathbb{B}_{n-1} + \frac{\Delta\bar{r}_n - \mathbb{B}_{n-1}\Delta\bar{x}_n}{\|\Delta\bar{x}_n\|^2}\Delta\bar{x}_n^T \tag{1.14}$$

donde $\Delta\bar{r}_n = \bar{r}_n - \bar{r}_{n-1}$ y $\Delta\bar{x}_n = x_n - \bar{x}_{n-1}$. Existe una variante del método, el método *Broyden ortonormal*, que aproxima la matriz $\mathbb{B}_n$ en función de una base vectorial de residuos previos calculados. Este método asegura la convergencia en una cantidad finita de iteraciones para sistemas lineales [7], (la cantidad de iteraciones requerida es la dimensión de la matriz jacobiana).

Existe también un conjunto de métodos conocidos como métodos *Newton-Krylov* [9] para la resolución del sistema 1.9. Estos métodos no requieren el cálculo de matriz jacobiana ya que resuelven el sistema mediante técnicas de gradientes descendentes. Sin embargo, en cada paso de descenso requieren evaluaciones de residuos, y por tanto son altamente costosos.

Notar que los métodos presentados en el presente apartado resuelven el sistema de ecuaciones 1.9 de forma monolítica, y es por este motivo que permiten incluir estrategias para la selección de condiciones de borde prohibidas al método *Dirichlet-to-Neumann*. Podría pensarse que el método *Dirichlet-to-Neumann* resuelve un sistema de ecuaciones similar pero solo para ciertas estrategias de condiciones de borde y en forma segregada, actualizando los valores de las variables a medida que avanza en el cálculo. En este caso, los residuos se definirían como la diferencia entre el valor de las variables calculadas en dos iteraciones consecutivas. Para el ejemplo de cálculo de temperaturas analizado en este capítulo, si se comenzara resolviendo el subsistema izquiero, en la iteración n el sistema de ecuaciones a resolver quedaría:

$$\begin{cases} (r_{q''})^1_{S1} &= (q''_{calc,n})_{S_1} - (q''_{calc,n-1})_{S_1} &= \mathscr{N}\left((T_{calc,n-1})_{S_2}\right) - (q''_{calc,n-1})_{S_1} \\ (r_T)^1_{S2} &= (T_{calc,n})_{S_2} - (T_{calc,n-1})_{S_2} &= \mathscr{D}\left(\left(q''_{calc,n}\right)_{S_1}\right) - (T_{calc,n-1})_{S_2} \end{cases} \tag{1.15}$$

donde se ha supuesto que $(T_{guess,n})_{S_1} = (T_{calc,n-1})_{S_2}$, y que $\left(q''_{guess,n}\right)_{S_2} = \left(q''_{calc,n}\right)_{S_1}$, empleando las relaciones de continuidad 1.1.

Problemas de evolución

En problemas de evolución la estretegia es permitir que cada subsistema avance por separado acoplando sus resultados mediante las ecuaciones 1.9 solo cada ciertos pasos. Esto se permite porque cada subsistema podría requerir subpasos de evolución diferentes[2]. Notar que no es necesario mantener la misma estrategia de selección de condiciones de borde para cada interfaz a lo largo de la evolución. Las variables que son datos o incógnitas para cada subdominio pueden irse alternando, seleccionando

[2] 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 mediante métodos complejos que consumieran elevados recursos temporales, por tanto sería de interés utilizar subpasos evolutivos mayores.

diferentes ecuaciones 1.2 a resolver en diferentes pasos de acople. Debido a la convergencia fuerte de los valores de las variables en las interfaces de acople, el método es incondicionalmente estable a lo largo de la evolución.

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 describe la estrategia para la resolución de problemas acoplados mediante diferentes programas de cálculo. Se presenta la estructura de comunicación definida y se describen las implementaciones necesarias para su funcionamiento. En el Capítulo 3 se muestran algunas aplicaciones de la herramienta estudiada. 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 de un reactor de investigación. 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. El tercer sistema de análisis es un modelo de redes hidráulicas con múltiples componentes. Con este estudio se busca demostrar la eficiencia de la herramienta en acoples fluidodinámicos de mayor escala. Sobre el final del capítulo se extiende la herramienta a problemas multifísicos. Se analiza un ejemplo de aplicación al 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

La estrategia general para la resolución de problemas acoplados mediante el método de descomposición disjunta de dominios es comentada a continuación. Los problemas parciales pertenecientes a distintos subdominios se resuelven con diferentes códigos específicos. Cada uno de estos códigos recibe valores de condiciones de borde y en base a ellos se encarga de calcular el valor de las incógnitas en las interfaces de acople mediante las ecuaciones 1.2. Existe un único programa que acopla los resultados. Este programa se abstrae completamente de la física implicada en cada subsistema. Simplemente recibe los valores para las incógnitas calculados por los diferentes programas y con ellos resuelve el sistema de ecuaciones de residuos 1.7. En base a estos residuos propone[1] nuevos valores para las incógnitas en las interfaces de acople y los comunica nuevamente a los demás programas. Este tipo de comunicación utilizado recibe el nombre de *maestro-esclavo* [10], en el cual existe un programa *maestro* que tiene el control unidireccional sobre los demás programas, que actúan bajo el rol de *esclavos*. Además del envío y recepción de valores de variables, son funciones del código *maestro* el envío de órdenes a sus *esclavos* 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 diferentes procesos del código *maestro* en la resolución de algún problema. Debido a la complejidad en destinatarios de mensajes,

[1] Esta propuesta reside en el método de resolución de ecuaciones no lineales seleccionado, ver sección 1.2.

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 siguiente sección.

2.2. 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 en los cuales no es posible modificar los códigos fuente.

En ambos casos los programas esclavos se comunican solo con el programa maestro. Si bien estos tipo de comunicaciones remotas son más lentas que las comunicaciones locales, en general su latencia es despreciable frente al tiempo de cálculo de los procesos *esclavos*.

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 [11]. 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 del estándard Interfaz de Paso de Mensajes (MPI por sus siglas en inglés, *Message Passing Interface*). 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++`, *texttt*) con paso de mensajes. De esta forma,

[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.

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*, es ejecutado 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 con él siguiendo una lógica preestablecida. Cuando ya no es necesario que los programas continúen comunicándose, se cierran las conexiones.

En la otra implementación, todos los programas son ejecutados al mismo tiempo en uno (MISD por sus siglas en inglés, Multiple Instruction, Single Data) o varios procesos (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 ya pueden intercambiar mensajes con el código *maestro* siguiendo la misma lógica que se establecerá para ambos modelos. 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 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

[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*.

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 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.

2.3. Códigos maestros utilizados

En este trabajo se utilizaron dos códigos maestros que cumplen con la estrategia de acoplamiento descripta previamente. El primer código *maestro* utilizado es el código **Coupling** [1] [2] [3]. **Coupling** utiliza los modelos de comunicación por paso de mensaje entre programas ejecutados en modos SISD y SIMD. El código maestro está diseñado de tal forma que los códigos acoplados resuelvan las ecuaciones 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 en modos MISD

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

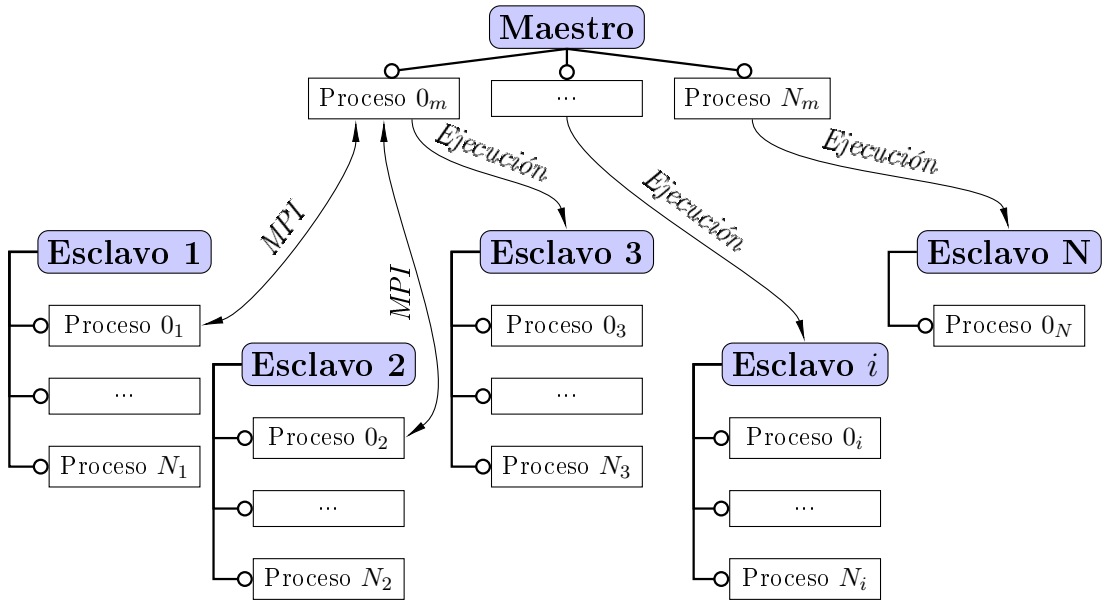


Figura 2.1: Esquema de comunicación entre los programas *esclavos* y el programa *maestro* implementado en el acoplamiento de códigos. Los *esclavos* se comunican solo con el *maestro* y no intercambian datos entre sí. Se utilizan dos modelos de comunicación diferentes. En el primero, cada código *esclavo*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

y MIMD, al acoplamiento de programas por lectura y escritura de archivos, 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** [12]. En los Apéndices A y B se describen las principales características de ambos códigos.

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 por paso de mensajes.

Las acciones de acoplamiento se reúnen en cuatro instancias diferentes:

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

En cada una de estas instancias el programa *esclavo* debe llamar a una función específica de acoplamiento. El programador podría definir una nueva variable *booleana* que a modo de bandera indique cuándo se está realizando un cálculo acoplado para realizar el llamado o evadirlo. 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 los esquemas SISD o SIMD 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.

Si, en cambio, los programas han sido ejecutados en los modos MISD o 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.

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 acoplado

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 $\Delta t_{coup} = \frac{t_{coup,N} - t_{coup,0}}{N}$. La solución para $t_{coup,0}$ es la condición inicial del problema, y es dato para todos los programas. En este paso se llama a la instancia 2, en la cual se recibe valores de condiciones de borde para el primer paso de acoplamiento, en $t_{coup,1}$. En base a estos valores, el programa *esclavo* resuelve las ecuaciones implicadas en el subdominio correspondiente. Es posible que además utilice subpasos de evolución Δt_{local} locales (como se explicó en el apartado 1.2 **Problemas de evolución**), por lo cual requerirá valores extras para las condiciones de borde en estos pasos. En estos casos, se implementa una estrategia de interpolación de valores entre la solución acoplada previa, y los valores recibidos para el siguiente paso acoplado. Este proceso se repite al principio de cada paso de acople.

Acoplamiento en instancia 3: al finalizar cada paso de evolución acoplado

Una vez resuelto cada Δt_{coup}, el programa *esclavo* envía al programa *maestro* los valores de las incógnitas calculadas 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 1.9. 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. La estrategia definida consiste en imponer al primer subdominio un valor *guess* para x y al segundo subdominio un valor *guess* para y. **Maestro** es el programa que se ocupa de proponer estos valores *guess* y de enviarlos a los *esclavos*. El programa **Esclavo 1** resuelve cada paso de evolución acoplado en función del valor x_{guess} recibido, y envía a **Maestro** el valor y calculado a partir de él. El programa **Esclavo 2** calcula x en función de y_{guess} y también envía el resultado a **Maestro**. **Maestro** entonces resuelve las ecuaciones de residuos, y decide si los resultados están convergidos o si es necesario continuar con las iteraciones.

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. Cada programa *esclavo* debe recibir una order para ejecutar estas acciones.

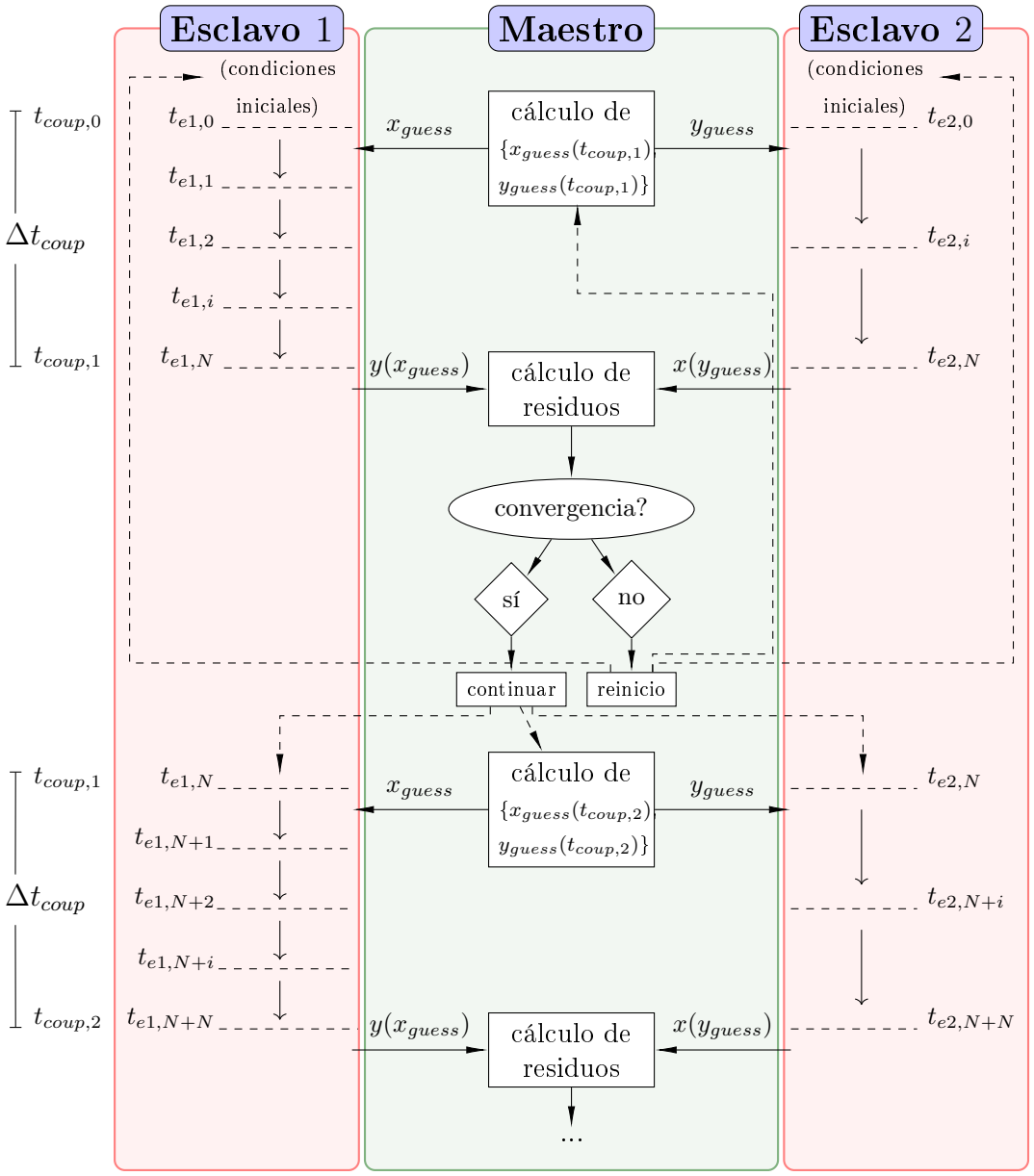


Figura 2.2: Esquema de acoplamento entre el programa *maestro* y los programas *esclavos*. En el ejemplo, cada código *esclavo* resuelve ecuaciones diferenciales en distintos subsitemas. Estos subsistemas están acoplados entre sí en alguna interfaz en las que las variables $\{x, y\}$ son incógnitas. 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}.

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 [13]. 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

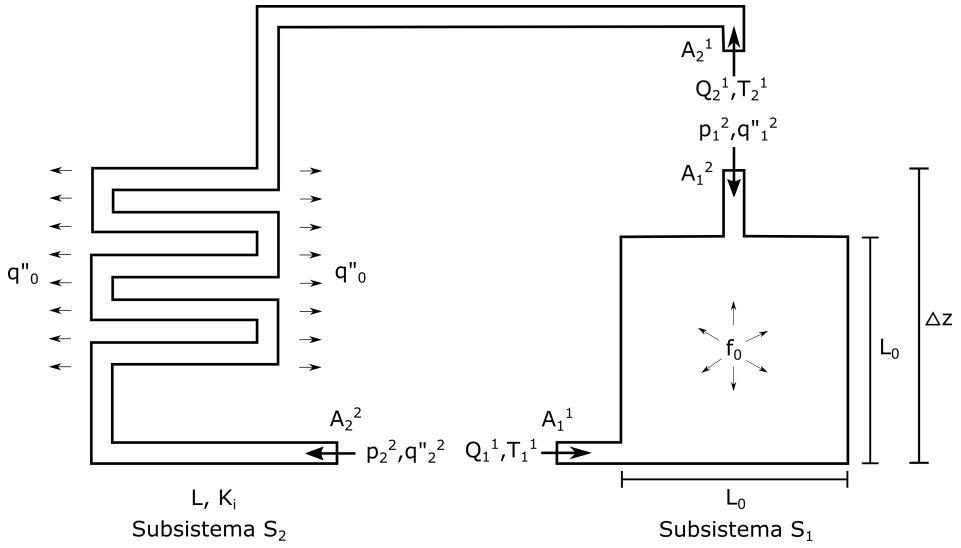


Figura 3.1: Modelo de la funete fría analizada. El subsistema de la izquierda es un intercambiador de calor y se estudia con un código cero-dimensional. El modelo de la derecha es una cavidad con una fuente de energía interna y se estudia con un código bi-dimensional. El sistema completo es abordado 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 [14] y de transferencia de energía en un intercambiador de calor con flujo constante [15]:

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

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 [16] y de transporte de energía [15]. 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 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* [17] y *PSPG* [18]. El método *SUPG* (*Streamline Upwind Petrov-Galerkin*) se utiliza para estabilizar problemas de transporte con alto número de *Peclet* (*Pe*). El *Pe* es un número adimensional que relaciona la velocidad de advección de un flujo y la velocidad de difusión, y está relacionado con el número de *Reynolds* (*Re*). En las ecuaciones de *Navier-Stokes*, el método estabiliza las evoluciones con alto *Re*, y consiste en la adición de una difusividad extra en la dirección de las líneas de corriente. El método *PSPG* (Pressure Stabilizing Petrov-Galerkin) se utiliza para evadir la condición *LBB*, que básicamente impone restricciones sobre los espacios de elementos utilizados en el problema de *Stokes*.

Las paredes imponen condiciones de no deslizamiento para las ecuaciones de *Navier-Stokes* y de flujo de energía nulo para la ecuación de energía. En las interfaces de acople, deben definirse una serie de condiciones de borde. En las ecuaciones de *Navier-Stokes*, en cada interfaz pueden setearse valores de velocidades normales, suponiendo velocidades tangenciales nulas (bajo la hipótesis de flujo paralelo), o valores de fuerzas normales (presión), suponiendo que las fuerzas tangenciales son nulas. En la ecuación de energía, cada borde necesita o bien un perfil de temperaturas o bien un perfil de flujo de calor. Los cálculos se realizaron implementando diferentes estrategias y verificando que los mismos convergieran.

La malla de cálculo se genera con **Gmsh** [19] 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 ambos modelos se utiliza $\Delta t = 0{,}01s$, debido a que ninguno requiere una mayor discretización temporal.

Los cálculos cero-dimensionales se realizan con un programa escrito para este propósito. Los cálculos bi-dimensionales se realizan con **Par-GPFEP** [20] [21]. Las modificaciones necesarias para implementar el acoplamiento de **Par-GPFEP** son comentadas en el Apéndice C.

Resultados del cálculo

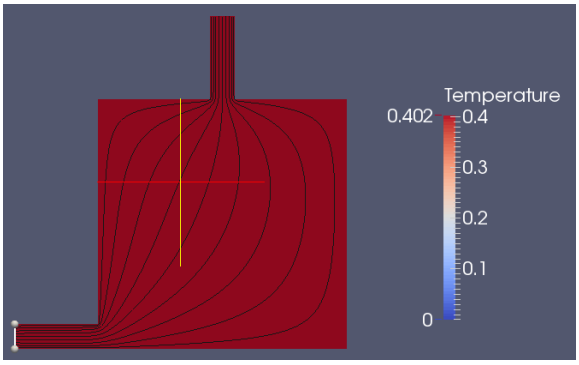


Figura 3.2: t=0 s

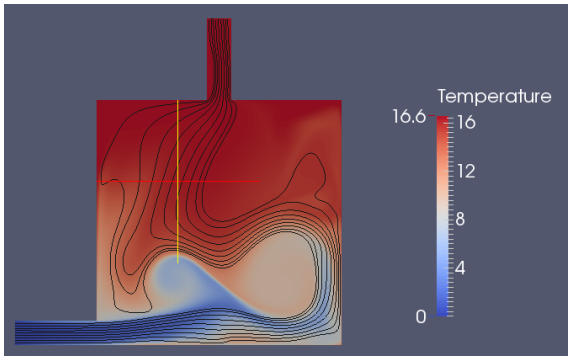


Figura 3.3: t=40 s

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.

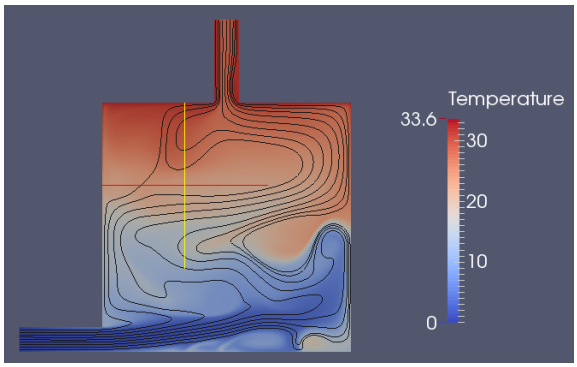


Figura 3.5: t=80 s

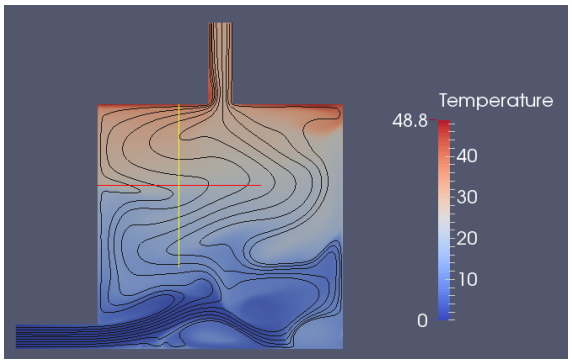


Figura 3.6: t=250 s

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.

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, [22], 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 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.

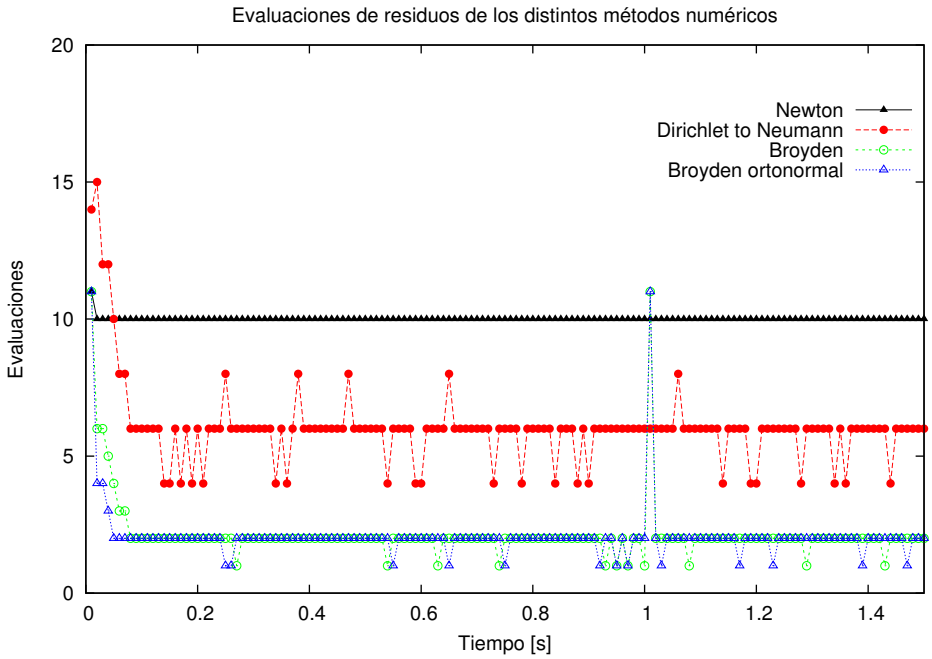


Figura 3.8: Evaluaciones de residuos requeridas por diversos métodos numéricos para resolver los sistemas de ecuaciones planteados en el problema doblemente acoplado descripto de la fuente fría de neutrones.

Los métodos *quasi-Newton* inicializan la matriz jacobiana sólo en el primer paso 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 de un reactor de investigación

Presentación del problema

El Departamento de Mecánica Computacional de CNEA tuvo a cargo el análisis del segundo sistema de parada (SSP) del reactor RA-10. El SSP consiste en el accionamiento del vaciado del tanque reflector. El drenado del material reflector (agua pesada) disminuye drásticamente la reactividad, apagando el reactor. La tarea consistió en 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). Este requerimiento pudo verificarse tras el análisis [23].

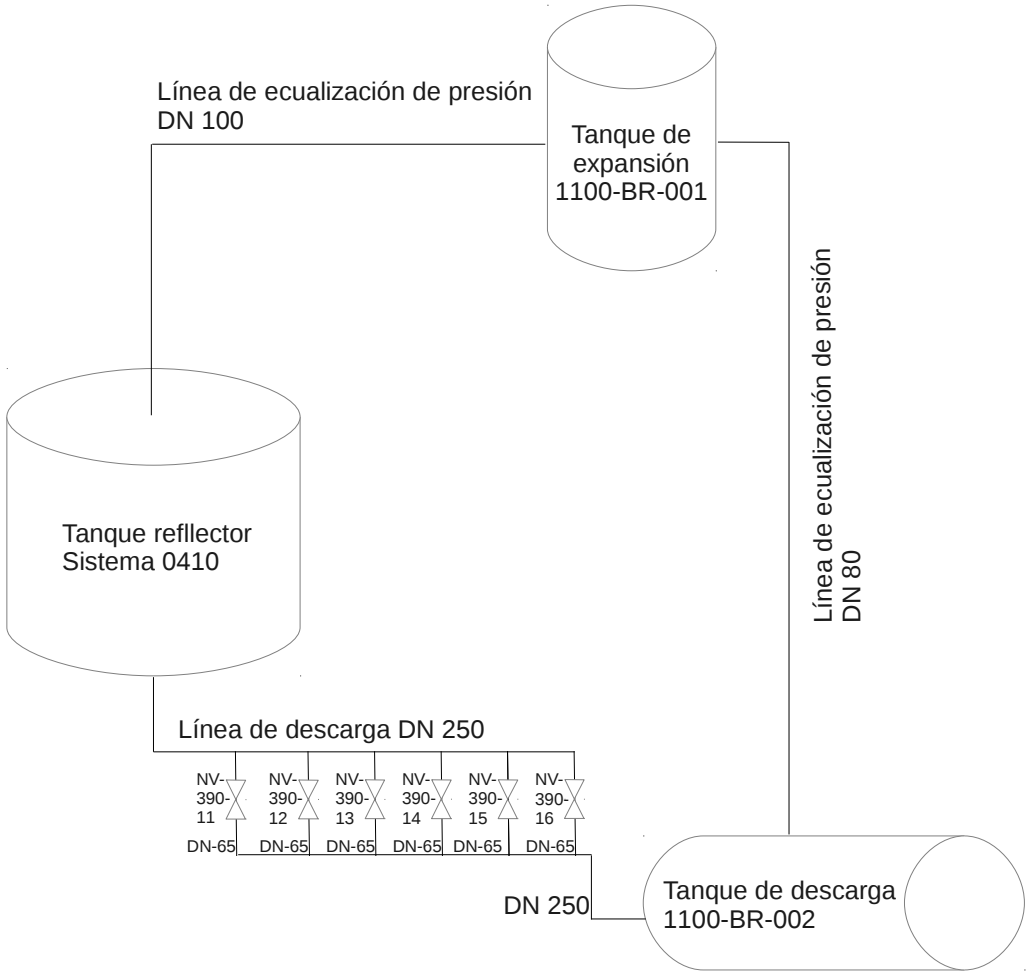


Figura 3.9: Esquema del segundo sistema de parada del reactor RA10.

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.

El análisis del problema completo hubiera demandado elevados recursos computacionales debido a los requerimientos de malla. Por ello se propuso desarrollar un modelo multiescala del sistema, desacoplándolo en subsistemas que pudieron estudiarse por separado con estrategias de acoplamiento mediante condiciones de borde apropiadas. El SSP del RA10 se dividió en tres subsistemas:

- Subsistema del tanque del reflector,
- Subsistema de la red hidráulica de descarga,
- Subsistema de la red hidráulica de ecualización de presiones.

En el trabajo presentado los subsistemas se acoplaron mediante una estrategia de acoplamiento débil [24] [25]. Durante el trabajo se realizaron tareas de validación de las herramientas de cálculo. Para ello se estudió un sistema similar para el que se conocían datos experimentales de tiempo de vaciado. Estos datos sirvieron para contrastar los resultados obtenidos con las herramientas de cálculo. El sistema analizado fue el tanque del reflector del *mockup* del reactor OPAL, montado por INVAP en San Carlos de Bariloche, [26].

Debido a que el mockup del OPAL está abierto a la atmósfera, no cuenta con línea de ecualización de presiones y por lo tanto no se consideró en el modelo.

En un estudio [27] 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. En el estudio aquí presentado se analiza con mayor detalle la distribución de caudales a través del arreglo de válvulas, modelando el comportamiento del resto del sistema con ecuaciones cero-dimensionales. El propósito de este estudio es investigar si existe algún efecto que podría no estar siendo considerado en el otro modelo.

Subsistemas de estudio

Se proponen 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 velocidades y fuerzas. La estretegia implementada es similar a la utilizada en 3.1 ya que se utilizan como variables de acoplamiento el caudal volumétrico y la presión promedio. Las fuerzas tagenciales se consideran nulas. A fines de cumplir con esta hipóteisis, la interfaz de acople se selecciona lejos de los codos. 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:

$$\begin{cases} p_1^1 &= p_2^1 \\ Q_1^1 &= Q_2^1 \end{cases} \tag{3.6}$$

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[3]. En base al caudal recibido en este subsistema se calcula un perfil de velocidades. Las ecuaciones de residuos quedan entonces:

$$\begin{cases} (R_{p,Q})_1^1 \, (p_1^1) &= 0 \\ (R_{p,Q})_2^1 \, (Q_2^1) &= 0 \end{cases} \tag{3.7}$$

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 [28]:

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

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

[3] Se podrían haber definido otras estrategias. La estrategia implementada permite resolver las ecuaciones en ambos subdominios de una forma cómoda. En el modelo tri-dimensional, por ejemplo, el valor de caudal recibido es útil para construir valores para las condiciones de borde del modelo turbulento utilizado.

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

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.

CB

La Tabla 3.1 reúne los parámetros del subsistema. Los datos geométricos pueden consultarse en las referencias [26]. El factor de pérdida de carga concentrada fue calculado en función de estos datos geométricos [14], 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** [29]. 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. En los primeros cálculos se supone que la válvula en falla es la ubicada en la última rama del arreglo. 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 [26].

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 un modelo de turbulencia de tipo RANS para modelar la fricción interna del fluido. El modelo utilizado es el modelo $\kappa - \epsilon$ *realizable*, en el que las ecuaciones se estabilizan mediante un método de control de coeficientes [30]. 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 $P^* = P + \frac{2}{3}\kappa$. 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 $c_\epsilon = 0{,}07$ [31].

En las ecuaciones de *Navier-Stokes*, cada borde necesita un perfil de velocidades normales o de fuerzas normales, y otro de velocidades tangenciales o de fuerzas tangenciales [16]. Las condiciones de borde al ingreso de la cañería dependen del valor Q_2^1 impuesto, a partir del cual se define un perfil de velocidades plano 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, despreciando las fuerzas tangenciales. Las ecuaciones de κ y ϵ no requieren condiciones contorno en esta interfaz. 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 [32]. Este modelo impone condiciones de tipo *Dirichlet* para κ y ϵ en la frontera en que se impone la ley de pared.

El sistema de ecuaciones (3.11) es resuelto en pasos fraccionados [33] mediante una formulación de elementos finitos con elementos lineales, estabilizada mediante *SUPG* [17] y *PSPG* [18]. 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 promedio

$< p_2^1 >$ en la interfaz I_{2_1}, a partir de los valores $< P^*_{I_2^1} >$ y $< \kappa_{I_2^1} >$ promediados en ella:

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

Los cálculos cero-dimensionales se realizan con un programa escrito para este propósito. Los cálculos tri-dimensionales se realizan con **Par-GPFEP**.

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. mediante el método de Broyden ortonormal, con reinicialización de la matriz jacobiana cada 100 pasos temporales. En la Figura 3.10 se reportan los resultados obtenidos para la evolución de los caudales y de las presiones en la interfaz de acople.

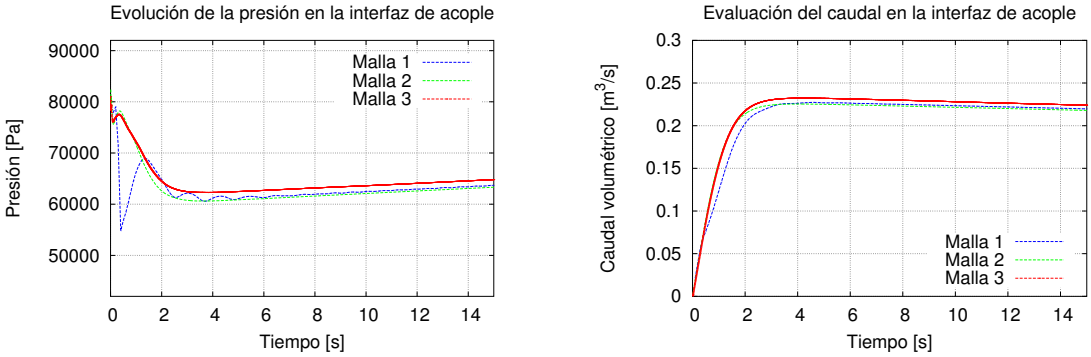


Figura 3.10: Evolución de la presión y del caudal volumétrico en la interfaz de acople entre los dos subsistemas. La presión atmosférica es de 92000 Pa.

En la Figura 3.11 se observa la evolución de la altura de la superficie libre del líquido en el tanque durante los primeros quince segundos obtenida en diferentes cálculos. La curva azul reporta los resultados obtenidos con la malla más gruesa, la curva roja los resultados obtenidos con la malla intermedia y la curva violeta los resultados obtenidos con la malla más fina. La curva verde muestra resultados de análisis estudiando la condición inicial de gas de relleno en las cañerías, que será comentada en la sección 3.2. Las curvas cyan y gris muestran resultados del cálculo del modelo tri-dimensional del tanque con acoplamiento débil al modelo cero dimensional de la red hidráulica [24],

obtenidas sin utilizar modelo de turbulencia, y utilizando el modelo *RANS* previamente comentado. Comparativamente se muestran también los valores experimentales reportados en la referencia [26].

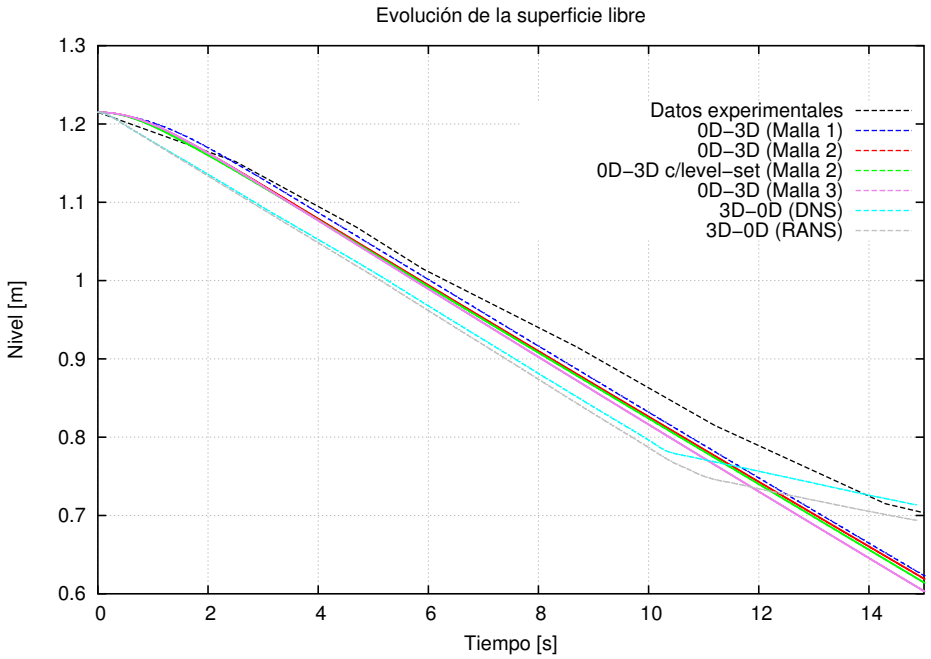


Figura 3.11: Evolución del nivel de líquido en el *mockup* del tanque del reflector del reactor OPAL ante accionamiento del SSP. La curva negra está construida con datos experimentales proporcionados por INVAP S.E. Las curvas azul, roja, verde y violeta reportan datos calculados mediante diferentes mallas para el modelo tri-dimensional del arreglo de válvulas, con acoplamiento fuerte al modelo cero-dimensional del resto del sistema. Las curvas cyan y gris muestran resultados del cálculo del modelo tri-dimensional del tanque con acoplamiento débil al modelo cero dimensional de la red hidráulica.

Los modelos computacionales predicen un comportamiento dinámico similar al reportado experimentalmente. Durante los primeros segundos de evolución existe una cierta inercia en la descarga que solo es captada por los modelos que describen el detalle en el arreglo de válvulas. Tras este transitorio inicial, todos los modelos predicen una pendiente de vaciado similar. Esta pendiente se corresponde con similares caudales de descarga entre los diferentes modelos, con lo que se verifica que la pérdida de carga total considerada en dos modelos independientes (modelo tri-dimensional del tanque acoplado, y modelo cero-dimensional del tanque acoplado) es similar. La curva experimental presenta ciertas ondulaciones que se deben al efecto que el oleaje en la superficie del líquido genera sobre el punto de medición. Estas variaciones son filtradas en el modelo tri-dimensional del tanque ya que la curva reporta una altura efectiva, calculada a partir del volumen restante de líquido en el tanque. Transcurridos los diez segundos de descarga, existe un quiebre en las curvas del modelo tri-dimensional del tanque. Este quiebre se corresponde al momento en el que las cañerías succionan tanto gas que es posible desacoplar el modelo cero-dimensional de pérdida de carga, basándose en la

hipótesis de que se establece una vena gaseosa entre el punto de succión y el orificio de descarga. Esta hipótesis es conservativa para el objetivo de estudio previsto, ya que si el acoplamiento de la red no fuera realmente despreciable, el tanque se vaciaría a mayor velocidad que la modelada. 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, lo que explica las diferencias entre las curvas en los últimos segundos.

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 descarga si la válvula que falla es alguna otra. En la Figura 3.12 se compara la evolución de la superficie libre ante fallas en la primera, la tercera y la sexta válvula.

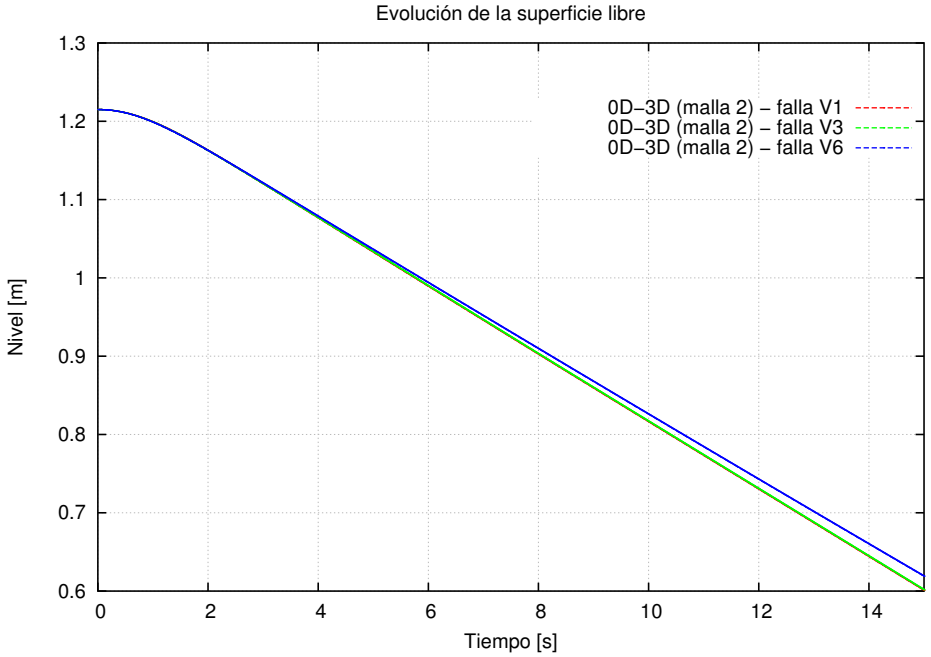


Figura 3.12: Evolución del nivel de líquido en el *mockup* del tanque del reflector del OPAL ante accionamiento del SSP considerando falla simple en diferentes válvulas.

Como puede observarse no es posible notar diferencias considerables en la evolución de la descarga. La pérdida de carga total del arreglo de válvulas es levemente 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 [34]. Para ello se añade un paso fraccionado extra al sistema de ecuaciones (3.11):

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

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 [35]. El campo del level set es reinicializado mediante cálculos geométricos cada 10 pasos temporales.

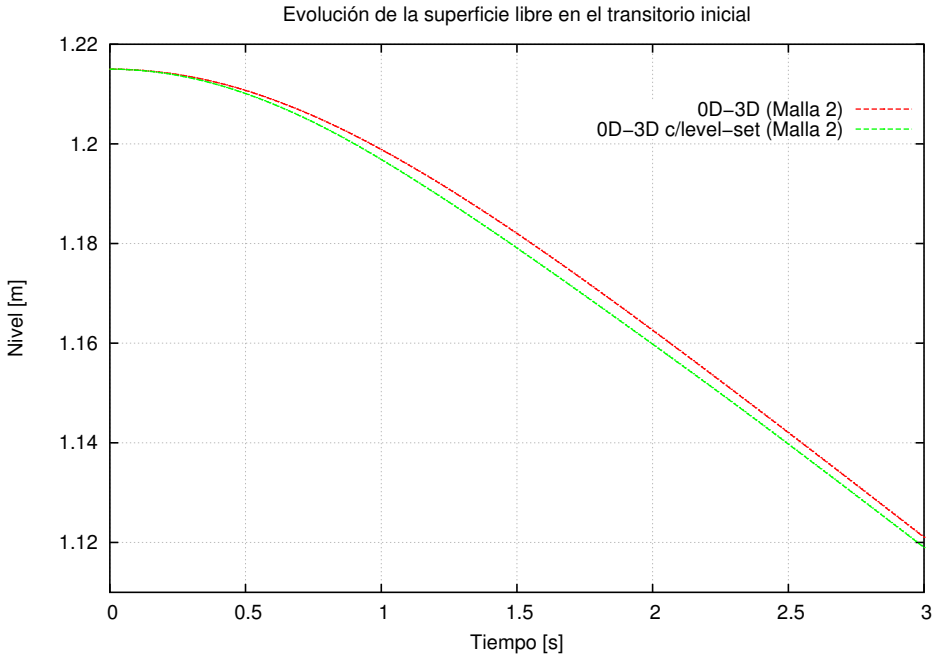
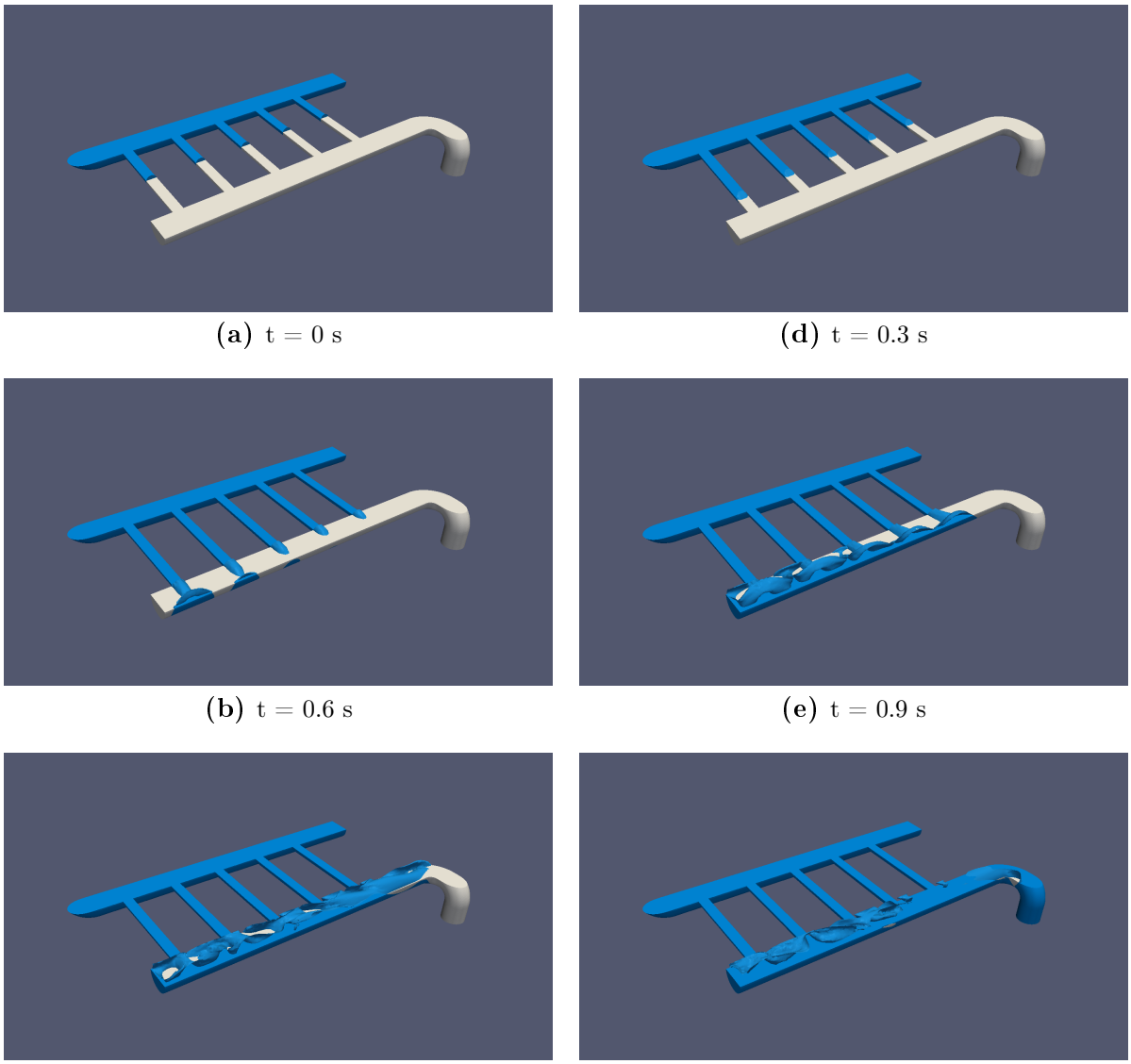


Figura 3.13: Evolución del nivel de líquido en el *mockup* del tanque del reflector del reactor OPAL ante accionamiento del SSP durante el transitorio inicial. Se comparan la solución obtenida despreciando el gas en la cañería y la obtenida con transporte de superficie libre mediante la técnica de *level-set*.

En la Figura 3.11 se compara la evolución obtenida de la superficie libre con los resultados anteriores, y en la Figura 3.13 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 3.14 se observa la evolución de la superficie libre durante los primeros instantes de tiempo.



(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 3.14: 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.

Conclusiones del análisis

La herramienta de análisis de acoplamiento fuerte de subsistemas permite incorporar el estudio de la inercia fluídica en la red hidráulica de descarga. Este estudio

concluye que los modelos que incluyen el fenómeno inercial del fluido en la red hidráulica predicen un retraso de la descarga en un máximo de un segundo respecto a los modelos que no lo incluyen. Como la inclusión del efecto inercial modela una dinámica similar a la reportada experimentalmente durante el transitorio inicial y, además, es conservativa en función del objetivo de estudio establecido, debería ser considerada en futuros análisis de seguridad.

Análisis de métodos de resolución del sistema de ecuaciones de residuos

A fines de comparar la efectividad de diferentes métodos numéricos se realizaron distintos cálculos utilizando la malla más gruesa. El parámetro de interés aquí no es la cantidad de iteraciones de cada método sino la cantidad de evaluaciones de funciones que cada uno requiere, ya que el tiempo de cálculo está directamente relacionado con ellas. La Figura 3.15 compara la cantidad de evaluación de funciones en función de paso temporal para diferentes métodos de resolución.

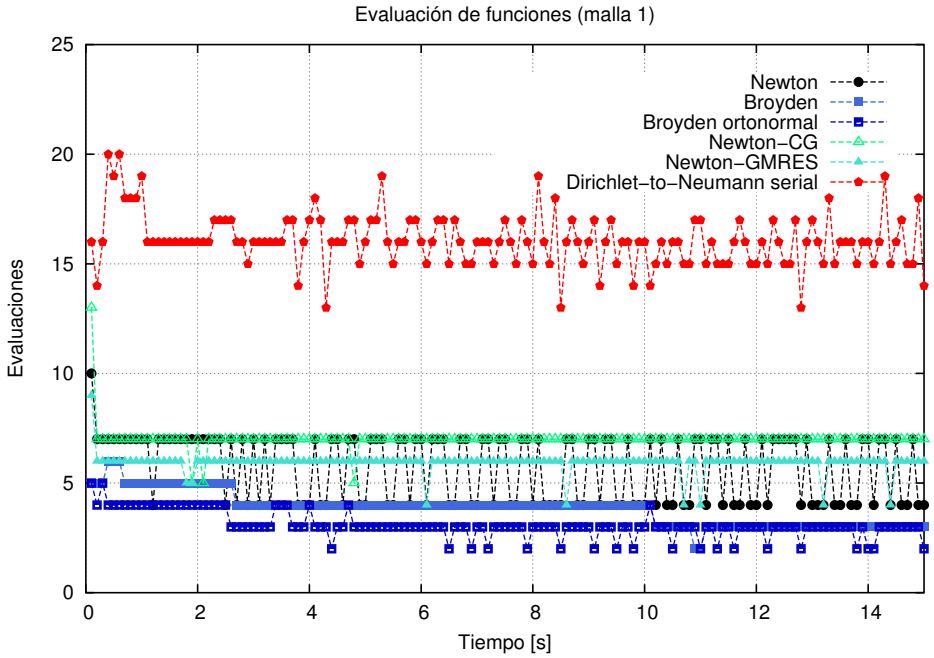


Figura 3.15: Evaluación de diferentes métodos numéricos en la resolución del sistema de ecuaciones de residuos resultante para el problema del vaciado del tanque reflector del *mockup* del reactor OPAL. El método explícito *Dirichlet-to-Neumann* requiere excesiva cantidad de evaluaciones en cada paso de tiempo, mientras que los métodos implícitos *quasi-Newton* son los más eficientes.

El método explícito *Dirichlet-to-Neumann* es el que mayor cantidad de evaluaciones consume, debido a que requiere una excesiva cantidad de iteraciones para converger. Los métodos de tipo *Newton-Krylov*: *Newton-GMRES* y *Newton-CG* (*Newton-Gradientes*

Conjugados) requieren baja cantidad de iteraciones, pero debido a la forma de resolución toman más evaluaciones que los métodos *quasi-Newton*: *Broyden* y *Broyden ortonormal*, los cuales convergen con baja cantidad de iteraciones y de evaluaciones asociadas. El método de *Newton-Raphson* toma tantas evaluaciones como los métodos *Newton-Krylov*, sin embargo, estas evaluaciones están asociadas a muy baja cantidad de iteraciones, ya que consume evaluaciones en la construcción de la matriz jacobiana.

En conclusión, al igual que en los resultados presentados en la sección 3.2, los métodos *Broyden* y *Broyden ortonormal* resultaron ser los más eficientes. A fines de acelerar aún más el cálculo, se estudió la forma de optimizarlos. Se ensayaron diferentes métodos para la propuesta de semillas del vector de incógnitas $\bar{x}_n$ y de la matriz $\mathbb{B}_n$ para cada paso temporal de resolución. Hasta ahora las semillas para el primer paso temporal eran el vector de ceros $\bar{x}_1 = \bar{0}$ y la matriz identidad $\mathbb{B}_1 = \mathbb{I}$, y las semillas para cualquier paso temporal próximo eran el vector $\bar{x}_{n-1}$ de la solución convergida en el paso previo, y la matriz $\mathbb{B}_{n-1}$ de la última iteración correspondiente a ese paso. Ahora el objetivo radica en intentar generar semillas que aceleren la convergencia.

Se propone utilizar un método de extrapolación, a partir de la información de los resultados que se van obteniendo en los sucesivos pasos. La semillas para $\bar{x}_n$ y para $\mathbb{B}_n$ podrían tener órdenes de extrapolación $k_{\bar{x}}$ y $k_{\mathbb{B}}$ diferentes. En el paso n, se van a utilizar los valores de los vectores $\bar{x}_i$, con $i \in \{n-1-k_{\bar{x}}, n-1\}$, y los valores de las matrices $\mathbb{B}_j$, con $j \in \{n-1-k_{\mathbb{B}}, n-1\}$. Estas extrapolaciones son válidas solo cuándo $n > k_{\bar{x}} + 1$ y $n > k_{\mathbb{B}+1}$ respectivamente.

La Figura 3.16 reporta la cantidad de evaluaciones de funciones requeridas para la convergencia en cada paso temporal, jugando con diferentes órdenes de extrapolación para $\bar{x}_n$ y $\mathbb{B}_n$. Las evaluaciones de funciones aquí están directamente relacionadas con las iteraciones para la convergencia, ya que el método de *Broyden* realiza una sola evaluación en cada iteración. Al comienzo del cálculo todos los esquemas numéricos requieren excesivas iteraciones para converger, y luego comienzan a converger con menor cantidad. El cálculo con órden nulo de extrapolación para ambas variables es el que más tarda en bajar la cantidad de iteraciones. Le siguen todos aquellos esquemas sin extrapolación para la matriz $\mathbb{B}_n$. Los esquemas con $k_{\mathbb{B}} = 3$ y $k_{\mathbb{B}} = 5$ son los que más rápidamente bajan la cantidad de iteraciones, por lo que se deduce que la extrapolación para la generación de semillas para $\mathbb{B}_n$ es altamente útil para arrancar el cálculo. En etapas avanzadas la matriz $\mathbb{B}_n$ se estabiliza y comienza a converger a resultados similares en los sucesivos pasos. Es decir, la tasa de cambio del vector solución $\bar{x}_n$ se vuelve aproximadamente constante (como puede observarse en la Figura 3.10). Ante una pequeño cambio, los esquemas de extrapolación para $\mathbb{B}_n$ amplifican esta perturbación y comienzan a generar malas semillas, por lo que comienzan a requerir mayor cantidad de iteraciones para converger. Este efecto puede observarse a partir de los $10s$ de cálculo. Por el contrario, los esquemas con bajo órden de extrapolación para $\mathbb{B}_n$ y

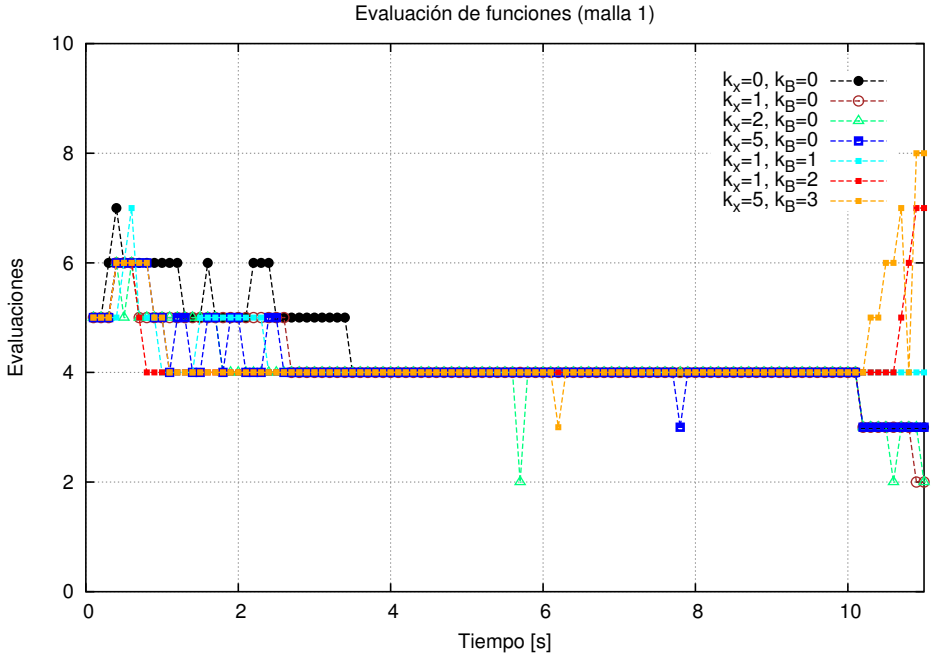


Figura 3.16: Eficiencia para diferentes esquemas de extrapolación en la generación de semillas para $\bar{x}_n$ y $\mathbb{B}_n$ en cada paso temporal. $k_{\bar{x}}$ indica el órden de extrapolación para $\bar{x}_n$ y $k_{\mathbb{B}}$ indica el órden de extrapolación para $\mathbb{B}_n$ utilizado en cada esquema. Los métodos con alto órden de extrapolación para $\mathbb{B}_n$ requieren menor cantidad de iteraciones para converger el cálculo en la primer etapa, pero a su vez requieren excesivas iteraciones ante alguna perturbación en los resultados. Los métodos con algún órden de extrapolación para $\bar{x}_n$ son más eficientes en estas instancias.

algún órden de extrapolación para $\bar{x}_n$ son más eficientes en esta etapa. Aquí podría pensarse que los esquemas de extrapolación son inestables ante perturbaciones en $\mathbb{B}_n$, pero estables para perturbaciones en $\bar{x}_n$.

En base a estos resultados, se deduce que el esquema de generación de semillas ideal requeriría órdenes de extrapolación $k_{\bar{x}}$ y $k_{\mathbb{B}}$ dependientes del tiempo, comenzando con alto $k_{\mathbb{B}}$ y bajo $k_{\bar{x}}$, y tendiendo a $k_{\mathbb{B}} = 0$ y alto $k_{\bar{x}}$ a medida que avanza el cálculo.

3.3. Resolución de redes hidráulicas de múltiples componentes

Presentación del problema

Con el interés de conocer el comportamiento de la metodología de resolución para problemas abordados mediante el Método de Descomposición Dijsunta de Dominios en sistemas con grandes cantidades de incógnitas, se propuso analizar redes hidráulicas de múltiples componentes interconectados. La idea es utilizar modelos sencillos para describir el comportamiento de cada componente particular para poder centrar el análisis solo en el estudio de convergencia.

Subsistemas de estudio

Se proponen sistemas de redes hidráulicas ramificadas divergentes. Debido a la metodología de abordaje propuesta en el trabajo, las interfaces deben seleccionarse de forma que cada una de ellas solo conecte dos subdominios contiguos. Por lo tanto, cada porción del sistema que comprende una ramificación es pensada como un subdominio diferente, de modo que cada subdominio contenga tres interfaces de acoplamiento. La Figura 3.17 esquematiza un modelo de estudio con 5 subsistemas acoplados. Cada subdominio es modelado con balances cero-dimensionales de conservación de masa y energía. Las ecuaciones resultantes para un subsistema genérico que no contiene bordes del dominio original son las siguientes:

$$\begin{cases} \frac{p_1}{\rho} + \frac{{v_1}^2}{2} &= \frac{p_2}{\rho} + \frac{{v_2}^2}{2} + gz_2 + \Delta u_{12} \\ \frac{p_1}{\rho} + \frac{{v_1}^2}{2} &= \frac{p_3}{\rho} + \frac{{v_3}^2}{2} + gz_3 + \Delta u_{13} \\ A_1 v_1 &= A_2 v_2 + A_3 v_3 \end{cases} \tag{3.14}$$

donde los subíndices 1, 2 y 3 refieren a diferentes extremos locales del contorno del subdominio, p_i, v_i, z_i, y A_i indican *presión*, *velocidad*, *altura* y *área* de la sección en el extremo i respectivamente, y Δu_{ij} refiere a la diferencia de energía del flujo entre los exremos i y j. El extremo 1 siempre corresponde al izquierdo de cada subdominio, y las otros se numeran en forma horaria creciente. Deben prestarse algunas consideraciones extras en las ecuaciones para los subsistemas que requieren condiciones $CB_{k,l}$ sobre extremos que pertenecían al borde original del sistema completo, donde k indica el subsistema y l el extremo local.

Los términos de presión estática $\frac{p_i}{\rho}$ y presión dinámica $\frac{{v_i}^2}{2}$ para el extremo i pueden agruparse en una única incógnita P_i para simplificar el cálculo:

$$P_i = \frac{p_i}{\rho} + \frac{{v_i}^2}{2} \tag{3.15}$$

El término Δu_{ij} puede aproximarse mediante una función de pérdida de carga como [36]:

$$\Delta u_{ij} = \frac{{v_i}^2}{2}\left(\frac{f_{D_i} L_i}{D_i} + \sum_t K_{i,t}\right) + \frac{{v_j}^2}{2}\left(\frac{f_{D_j} L_j}{D_j} + \sum_t K_{j,t}\right) \tag{3.16}$$

En esta ecuación, el primer término está modelando la pérdida de carga total entre el extremo i y el nodo de divergencia, y el segundo extremo modela la pérdida de carga total entre este nodo y el extremo j. Las variables D_i y L_i corresponden al *diámetro* y a la *longitud* de la cañería desde el extremo i hasta el nodo de divergencia, f_{D_i} corresponde al *factor de Darcy* del flujo en esa porción y $K_{i,t}$ corresponde al *factor de pérdida de carga concentrada* t de cualquier componente hidráulico presente lo largo de algún punto de esa porción de cañería. Bajo algunas modificaciones sería posible

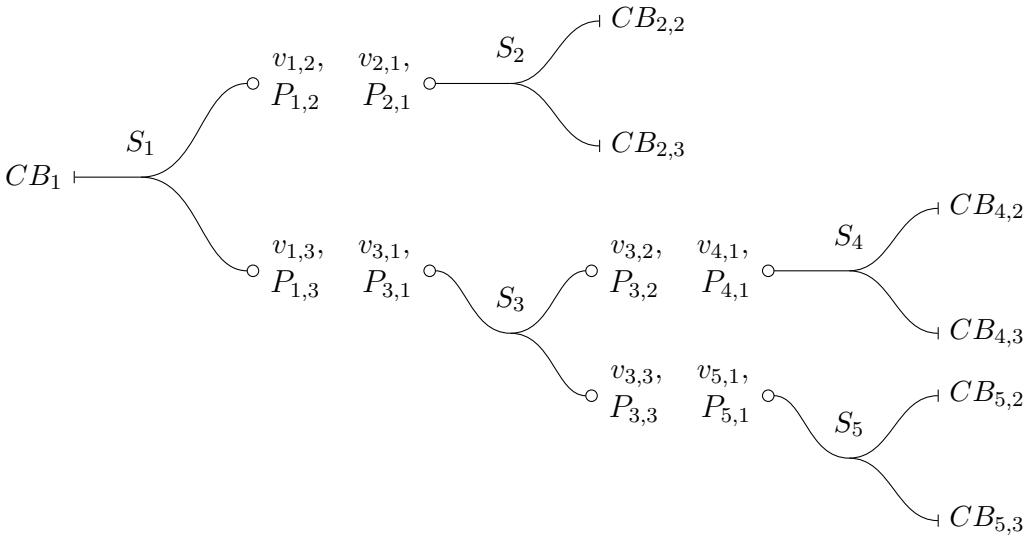


Figura 3.17: Descomposición disjunta de dominios en un modelo de red hidráulica con 16 incógnitas en las interfaces de acoplamiento. La incógnita $v_{i,j}$ refiere a la velocidad en el extremo j del subsistema i. La incógnita $P_{i,j}$ agrupa las presiones estática y dinámica en el extremo j del subsistema i. Las incógnitas pueden reducirse rápidamente a la mitad aplicando las relaciones de continuidad 1.1.

incorporar cambios en las secciones a lo largo de estas porciones, pero no se realizó por simplicidad.

Considerando que el flujo corre por la red hidráulica en régimen laminar, f_{D_i} puede modelarse como [36]:

$$f_{D_{i,lam}} = \frac{64}{Re_{D_i}} \tag{3.17}$$

donde $Re_{D_i} = \frac{\rho v_i D_i}{\mu_i}$, siendo μ la viscocidad dinámica del fluido. Bajo esta aproximación, la ecuación 3.18 queda lineal en v_i y en v_j para aquellos subsistemas en los que pudiera despreciarse la pérdida de carga concentrada:

$$\Delta u_{ij,lam} = \frac{v_i}{2}\left(\frac{64\mu L_i}{\rho {D_i}^2}\right) + \frac{v_j}{2}\left(\frac{64\mu L_j}{\rho {D_j}^2}\right) \tag{3.18}$$

Conforme al esquema de resolución descripto en la sección 1.2, es necesario definir una estrategia para las condiciones de borde en las interfaces de acoplamiento de cada subsistema. La estrategia propuesta es establecer condiciones de tipo *Dirichlet* sobre las interfaces ubicadas a la izquierda de cada subdominio (fijando v) y condiciones de tipo *Neumann* sobre las interfaces ubicadas a la derecha (fijando P).

Cada subdominio es modelado con funciones *esclavas* sencillas escritas en `Octave`. Los parámetros geométricos de cada subsistema se sortean aleatoriamente entre valores típicos. El fluido de trabajo es agua a temperatura y presión ambiente. Los códigos esclavos calculan el valor de las incógnitas en cada una de las interfaces de acoplamiento a partir de los datos recibidos como condiciones de borde. El sistema de ecuaciones de

residuos 1.9 resultante tras aplicar las ecuaciones de continuidad 1.1 entre subdominios, es resuelto por una función *maestra*.

Redes hidráulicas con regímenes de flujo laminar

En la Figura 3.18 (a) se pueden observar la cantidad de iteraciones requeridas por diferentes métodos para la convergencia de resultados en sistemas hidráulicos laminares sin pérdidas de carga concentrada, variando la cantidad de subsistemas acoplados. La cantidad de incógnitas en el eje x corresponde a una simplificación obtenida tras aplicar las ecuaciones de continuidad.

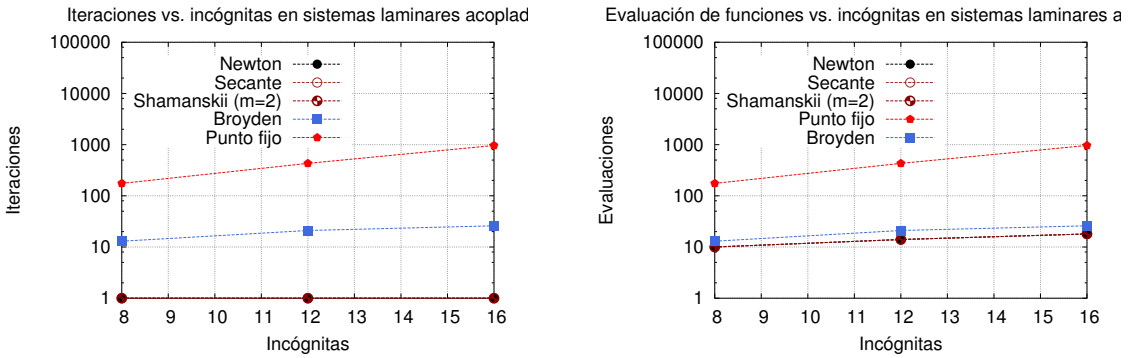


Figura 3.18: Comparación de diferentes métodos numéricos para la resolución de sistemas de redes hidráulicas: (a) iteraciones requeridas y (b) evaluaciones de funciones requeridas.

El método del punto fijo reportado es un método explícito. En este método se genera una semilla inicial $\bar{x}_0$ para las incógnitas y se las envía a los códigos *esclavos*. Los resultados que ellos devuelven conforman directamente el vector de iteración $\bar{x}_1$. El método del punto fijo podría pensarse como una combinación múltiple de métodos *Dirichlet-to-Neumann*, en el que los residuos se computan directamente entre vectores de iteración contiguos. En la figura se observa que este método requiere excesiva cantidad de iteraciones. Las mismas ascienden hasta 1000 para sistemas con 16 incógnitas reducidas, pero este valor puede variar dependiendo de las semillas iniciales y de los parámetros del sistema. El método de *Broyden* requiere decenas de iteraciones para cada sistema. El método de *Newton-Raphson*, el método de la *secante* (construye la matriz jacobiana solo en la primera iteración y luego la utiliza sin cambios en las siguientes iteraciones) y el método de *Shamanskii* (construye la matriz jacobiana cada m iteraciones) requieren solo una iteración, debido a que los sistemas de cálculo son lineales (las tres curvas se encuentran superpuestas). Sin embargo, el parámetro de comparación de interés en estos casos para medir la eficiencia de cada método es la cantidad total de evaluaciones que requiere. En la Figura 3.18 (b) se observa que la ventaja obtenida por los métodos que construyen la matriz jacobiana es despreciable frente al método de *Broyden*.

Debido a que los métodos *quasi-Newton* han presentado elevada confiabilidad en la resolución de sistemas acoplados a lo largo de todo el trabajo, se investigaron formulaciones alternativas al método de *Broyden*, y en la Figura 3.19 se reportan los resultados.

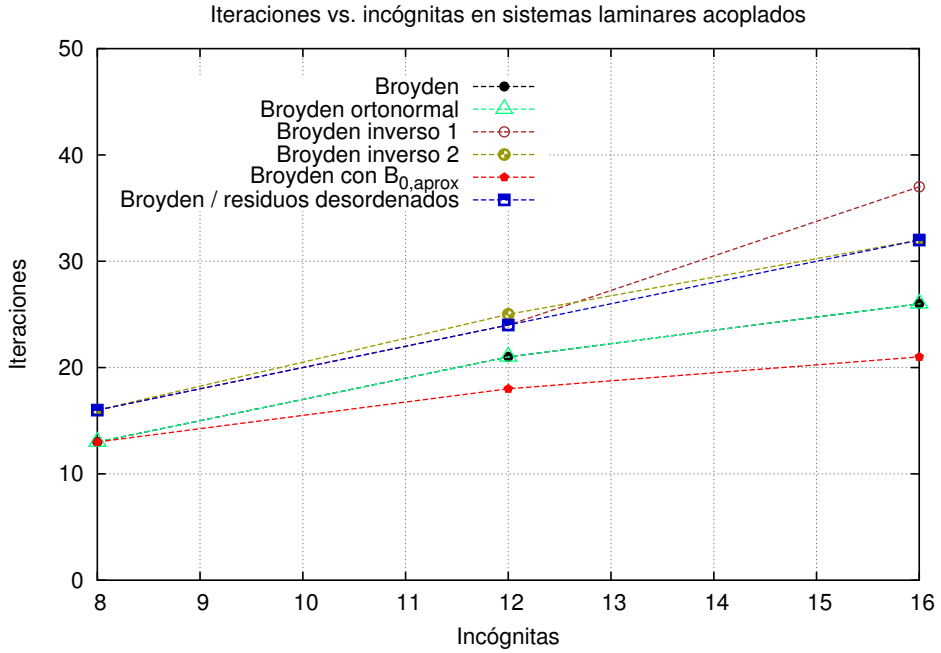


Figura 3.19: Comparación de diferentes esquemas *quasi-Newton* para la resolución de sistemas de redes hidráulicas con regímenes de flujo laminar.

Además del método *Broyden ortonormal*, que en este estudio se comporta con igual eficiencia que el método de *Broyden* (ambas curvas se solapan), existen formulaciones que aproximan directamente la inversa de la matriz jacobiana (*Broyden inverso 1* y *Broyden inverso 2*). Estos métodos se conocen en la bibliografía como *bad Broyden update* (*mala actualización de Broyden*) [37] y en la figura puede verse que tienen eficiencia inferior.

El método de *Broyden* se utilizó también mejorando la semilla inicial para la matriz $\mathbb{B}_n$. En todos los estudios reportados se venía utilizando la matriz identidad, pero aquí se reemplazó por una matriz con *unos* en los elementos que corresponden a posiciones llenas de la matriz jacobiana original, y *ceros* en el resto de los elementos. Este llenado es sencillo de implementar ya que simplemente depende de las relaciones de dependecia de las residuos y las incógnitas, que *a priori* son conocidas. Con esta implementación puede observarse en la curva de *Broyden con* $B_{0,aprox}$ que la convergencia mejora a medida que la cantidad de incógnitas aumenta.

La curva *Broyden / residuos desordenados* corresponde a un esquema en el que el residuo i se corresponde con alguna incógnita j distinta de i. Utilizar la matriz identidad como semilla para $\mathbb{B}_n$ es una mala propuesta en este caso, ya que para cada fila i, el *uno* debería ubicarse en la posición j. Esto genera mayor dificultad para la

convergencia. Por lo tanto puede comprenderse que la diferencia entre las curvas *azul* y *roja* (que representan los resultados para el mismo método de *Broyden*) depende exclusivamente de una elección inteligente en la forma de proponer la semilla para la matriz $\mathbb{B}_n$. Esta diferencia asciende a 10 iteraciones en un sistema con 16 incógnitas reducidas.

Redes hidráulicas con regímenes de flujo turbulento

Habiendo investigado sistemas de redes hidráulicas con regímenes de flujo laminar, el siguiente paso de estudio consiste en analizar sistemas de redes hidráulicas con regímenes de flujo turbulento. En estos modelos se utiliza directamente la ecuación 3.18 para representar la pérdida de carga entre dos extremos de un dado subdominio, por lo que el sistema de ecuaciones global ahora es un sistema de ecuaciones no lineales. Aquí los métodos estudiados en el apartado anterior se comportan de forma diferente. El método de *Newton-Raphson* ya no converge en una única iteración como lo hace en sistemas lineales. La Figura 3.20 detalla la cantidad de evaluaciones de funciones requerida por distintos métodos para la convergencia de los resultados en sistemas con diferente cantidad de incógnitas reducidas.

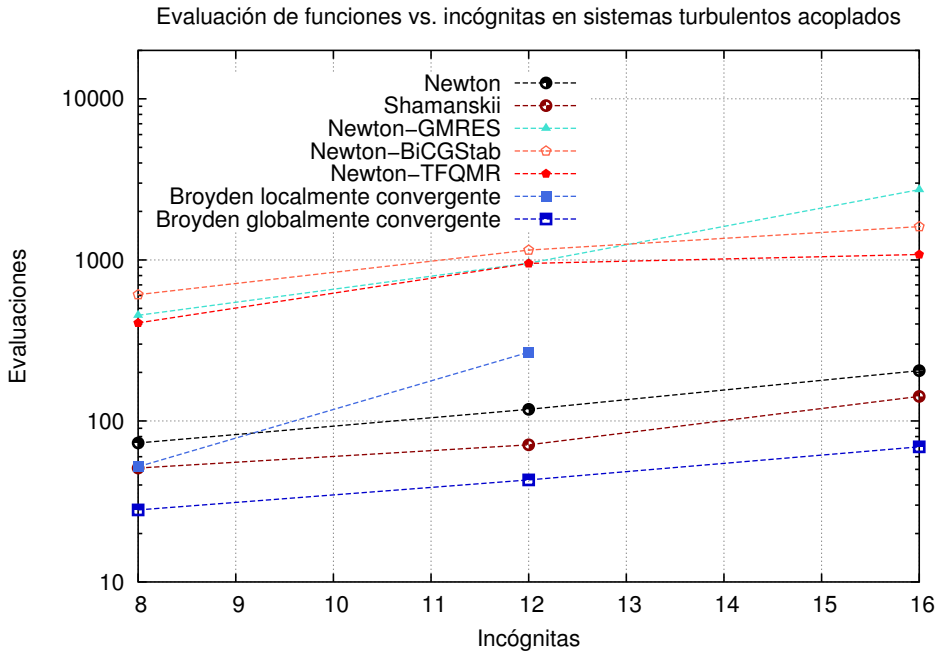


Figura 3.20: Comparación de diferentes métodos numéricos para la resolución de sistemas de redes hidráulicas con regímenes de flujo turbulento.

Los métodos *Newton-Krylov* toman demasiadas evaluaciones para converger. El método de *Broyden localemente convergente*, que no es otro sino el método de *Broyden* que hasta ahora se venía utilizando, solo se comporta bien a baja cantidad de incógnitas, y cuando la cantidad de incógnitas es elevada diverge. Los métodos de *Newton-Raphson*

y *Shamanskii* tienen buena convergencia. De estos dos el último presenta mayor ventaja ya que elude el cálculo de la matriz jacobiana en iteraciones contiguas. El método que mejor se comporta es el método de *Broyden globalmente convergente*, que realiza búsquedas del mínimo a lo largo de la dirección de cambio en cada iteración (empleando evaluaciones de funciones extras). Estas búsquedas son conocidas en la bibliografía como *line searching* [9].

3.4. Extensión al problema neutrónico-termohidráulico

Estrategia de acoplamiento extendida

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 4

Conclusiones

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

— Bertrand Russell, 1872-1970

Apéndice A

Descripción del código maestro Coupling

Coupling es un código escrito en *Fortran 90*, desarrollado por Jorge Leiva, Pablo Blanco y Gustavo Buscaglia. Fue diseñado como programa *maestro* en las funciones de acoplamiento explícito e implícito, estableciendo la comunicación con diversos programas *esclavos* mediante funciones de MPI. Su desempeño fue validado en diversos trabajos de acoplamiento [1] [2] [3] [6].

A.1. Modelado de problemas acoplados

El programa **Coupling** resuelve problemas que fueron formulados mediante el Método de Descomposición Disjunta de Dominios descripto en el Capítulo 1. El esquema de resolución fue diseñado de tal forma que se permite que las interfaces de acoplamiento cuenten con múltiples pares de incógnitas vectoriales en diferentes nodos. Las principales variables en la definición de un problema son las siguientes:

- Cantidad de subdominios acoplados (variable *Ncomp*).
- Cantidad de uniones globales (variable *Nbond*).
- Cantidad de superficies en cada subdominio (variable *Nsupc*).
- Matriz de conectividades entre las interfaces de diferentes subdominios (variable con numeraciones locales *Localsur* y globales *Incid*.
- Cantidad de pares de variables incógntias en cada interfaz (variable *Npair*). Se supone que cada incógnita de cada interfaz está acompañada de otra variable incógnita relacionada con su gradiente. Por ejemplo, *temperatura* y *flujo de calor*, o *velocidad* y *presión*.

- Cantidad de nodos con incógnitas en cada interfaz (variable *Nodpair*).
- Cantidad de componentes escalares en cada nodo (variable *Npairc*).
- Tipo de condición de borde para cada nodo (variable *indexbc*). Las opciones posibles son condiciones de borde *esenciales* (tipo *Dirichlet*) o *naturales* (tipo *Neumann*), y deben establecerse por separado para cada subdominio.

Para que el problema de acoplamiento quede bien definido, todos estos datos deben ser provistos en el archivo de configuración `mesh.cfg` utilizando palabras claves específicas. Además, debe proveerse un archivo de configuración extra, el arhivo `couple.cfg`. En este archivo se detalla el método de acoplamiento a utilizar en la resolución del problema, y sus diferentes parámetros específicos. También se detallan los parámetros necesarios en problemas de evolución.

En la sección A.3 se describe como ejemplo la formulación utilizada en una de las aplicaciones analizadas en el Capítulo 3.

A.2. Metodología de resolución

El archivo `src/couple_ddm.for` contiene la estructura principal del programa. El modelo de comunicación entre programas implementado es el de paso de mensajes entre programas ejecutados en modos SISD o SIMD. Conforme a la estrategia de acoplamiento descripta en el Capítulo 2, se siguen los siguientes pasos:

1. Lectura de archivos de datos `mesh.cfg` y `couple.cfg`.
2. Inicialización de las funciones de MPI y publicación de puertos de conexión.
3. Establecimiento de la comunicación con los diferentes programas esclavos.
4. Envío de datos generales del problema a los programas *esclavos* para chequeo de compatibilidad de datos.
5. Ingreso en el bucle de evolución.
   - *a*) Envío de valores *guess* para las condiciones de borde de cada subdominio a cada programa *esclavo*.
   - *b*) Recepción de valores calculados para las incógnitas de cada subdominio de cada programa *esclavo*.
   - *c*) Resolución de las ecuaciones de residuos 1.7.
   - *d*) Evaluación de la convergencia:

- si los resultados están convergidos el problema puede continuar con el siguiente paso de evolución;
- si los resultados no están convergidos, es necesario proponer nuevos valores *guess* y reiniciar el cálculo.

e) Envío de orden de avance o de reinicio a cada programa *esclavo*.

6. Finalización del bucle de evolución.

7. Cierre de las conexiones con los programas *esclavos* y finalización.

Los pasos descriptos requieren una arquitectura de acoplamiento específica montada en los códigos *esclavos*, la cual fue comentada en la sección 2.4.

A.3. Ejemplo de uso

A continuación se comenta la formulación del problema analizado en la sección **??**: *Análisis del segundo sistema de parada de un reactor de investigación*. Se definen dos subdominios: el primero es el subsistema tri-dimensional arreglo de válvulas y el segundo es el subsistema cero-dimensional del tanque del reactor y porción superior de la red hidráulica. Existe una única unión entre ambos subdominios, la *Unión 1*. Es necesario definir una numeración global de interfaces de acoplamiento. El subsistema tri-dimensional contiene la interfaz global *Interfaz 1* y el subsistema cero-dimensional contiene la interfaz global *Interfaz 2*. En cada interfaz existe un solo par de variables acopladas, el par *caudal-presión*. El *caudal* es impuesto como condición de borde *esencial* al *Subdominio 1* y la *presión* como condición de borde *natural* al *Subdominio 2*. La Figura A.1 resume las definiciones previas.

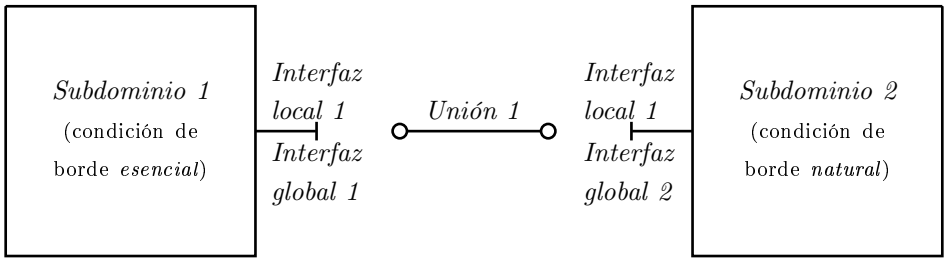


Figura A.1: Definición del problema de acoplamiento en el Segundo Sistema de Parada del reactor de investigación investigado en la sección **??**.

Las siguientes líneas conforman el archivo de configuración `mesh.cfg`:

```
! Nombre del problema
*TITLE
manifold mockup OPAL
```

```
! Número de subdominios
*NUMBER_OF_COMPONENTS
2     ! Subdominio 3D y subdominio 0D

! Cantidad de interfaces en cada subdominio
*NUMBER_OF_COMPONENT_INTERFACES
1     ! Subdominio (1): 1 interfaz
1     ! Subdominio (2): 1 interfaz

! Uniones en cada componente
*CONECTIVITY_BETWEEN_COMPONENT_INTERFACES
1 1   ! Subdominio (1); Interfaz local (1): Unión 1
2 1   ! Subdominio (2); Interfaz local (1): Unión 1

! Pares de variables en cada componente
*PAIR_VARIABLES_NUMBER_BY_COMPONENT_INTERFACE
1     ! Subdominio (1); interfaz local (1): 1 par
1     ! Subdominio (2); interfaz local (1): 1 par

! Componentes escalares en cada par de variables
*PAIR_VARIABLES_NUMBER_OF_COMPONENTS
1     ! Subdominio (1); interfaz (1); par (1): 1 componente escalar
1     ! Subdominio (2); interfaz (1); par (1): 1 componente escalar

! Tipo de condición de borde en cada par
*BOUNDARY_CONDITION_TYPE_BY_COMPONENT_PAIRS
1     ! Subdominio (1); interfaz (1); par (1): tipo Dirichlet (1)
-1    ! Subdominio (2); interfaz (1); par (1): tipo Neumann (-1)

! Cantidad de nodos en cada par de variables
*PAIR_VARIABLES_NUMBER_OF_NODES
1     ! Subdominio (1); interfaz (1); par (1): 1 nodo
1     ! Subdominio (2); interfaz (1); par (1): 1 nodo

! Finalización
*END
```

Las siguientes líneas en el archivo de configuración `couple.cfg` definen la metodo-

logía numérica de resolución:

```
! Parámetro de evolución inicial
*INITIAL_CONTINUATION_PARAMETER
0.0

! Parámetro de evolución final
*FINAL_CONTINUATION_PARAMETER
15.0

! Cantidad de pasos de evolución
*NUMBER_OF_CONTINUATION_STEPS
1500

! Maxima cantidad de iteraciones del método de acoplamiento en cada
paso
*MAXIMUM_NONLINEAR_ITERATIONS
800

! Método de acoplamiento
*NONLINEAR_SOLVER
2 ! Broyden

! Tolerancia absoluta en residuos de variables acopladas
*ABSOLUTE_TOLERANCE
1.0e-15

! Tolerancia relativa en residuos de variables acopladas
*TOLERANCE_NONLINEAR
1.0e-08

! Iteraciones lineales máximas en la resolución de sistemas
algebraicos definidos en métodos implícitos
*MAXIMUM_LINEAR_ITERATIONS
800

! Pasos entre las reinicializaciones de la matriz jacobiana
*JACOBIAN_INITIALIZATION
100
```

```
! Parámetro de perturbación en cálculo de la matriz jacobiana mediante
diferencias finitas
*JACOBIAN_PERTURBATION_PARAMETER
0.1

! Máxima cantidad de evaluaciones de funciones en cada paso
*MAXIMUM_FUNCTION_EVALUATIONS
10000

! Órden de extrapolación de variables incógnitas
*STATE_VARS_EXTRAPOLATION_ORDER
1

! Órden de extrapolación de matriz jacobiana
*JACOBIAN_EXTRAPOLATION_ORDER
0

! Finalización
*END
```

Apéndice B

Descripción del código maestro Newton

Newton es un código escrito en `C++` desarrollado durante la maestría, publicado en **Github** [38] bajo Licencia Pública General (GPL por sus siglas en inglés, *Public General License*) del Proyecto GNU colaborativo de software libre. Fue diseñado como programa *maestro* con un propósito general, de forma que permita resolver cualquier tipo de ecuaciones acopladas, parcialmente resueltas por códigos *esclavos*. Algunos ejemplos de los problemas que pueden resolverse con **Newton** son los siguientes:

- Cálculos formulados con el Método de Doscomposición Disjunta de Dominios: acoplamientos fluidodinámicos multiescala, con o sin transferencia de calor, como los presentados en las secciones 3.1, **??** y 3.3.

- Cálculos multifísicos: acoplamiento fluido-estructura, acoplamiento neutrónico-termohidráulico, como el presentado en la sección 3.4. También podrían pensarse modelos de núcleo completo (*full core models*), acoplando el comportamiento de los materiales a la dinámica neutrónica-termohidráulica y otros fenómenos de interés.

B.1. Principales características

Archivo de configuración

La ejecución de **Newton** requiere el archivo de configuración `newton.config` en el que debe detallarse la formulación del problema a resolver, la forma de comunicación con cada código *esclavo*, los mapeos utilizados, y métodos numéricos y demás parámetros asociados a la resolución del sistema 1.9. El formato de entrada fue pensado de forma tal que resulte simple e intuitivo al usuario. Las funciones implementadas en

la clase `Parser` se encargan de analizar las diferentes palabras claves ingresadas y sus atributos. Se desarrolló un manual de comandos de entrada [39].

Mapeos

En ciertas formulaciones las variables de acoplamiento pueden ser expresadas bajo transformaciones en las diferentes ecuaciones acopladas. Por ejemplo, en acoplamientos fluidodinámicos es común que en un subsistema utilice velocidades en la interfaz y que otro utilice un valor de caudal en la interfaz acoplada. El caudal se define como la integral del perfil de velocidades normal a la interfaz de acople, sobre toda esta sección. Esta integral puede pensarse como un mapeo *velocidad-caudal*. En otros tipos de problemas también es necesario realizar mapeos. En acoplamiento neutrónico-termohidráulico, por ejemplo, las variables de cálculo del programa termohidráulico podrían ser densidades y termperaturas, pero el programa neutrónico utiliza como variables secciones eficaces, que dependen de ellas. Además, las variables transformadas podrían ser condensadas o multiplicadas. Aquí podría pensarse un mapeo *termohidráulico-secciones eficaces*.

En general, los mapeos son transformaciones simples de las variables de acoplamiento, y pueden extraerse de la resolución de las ecuaciones parciales, definiendo ecuaciones extras que pueden ser resueltas por el programa *maestro*. Debido a esta generalización, es necesario definir cuál va a ser la variable acoplada utilizada para resolver el sistema de residuos 1.9. Diferentes mapeos pueden ser utilizados para diferentes códigos *esclavos*, ya sea para adecuar variables *guess* o variables calculadas. Su implementación requiere una mínima programación en funciones listas para ser utilizadas pertenecientes a la clase `Mapper`. Luego deben ser incluidos mediante palabras claves en el archivo de configuración `newton.config`.

Modelos de comunicación

Con la finalidad de poder establecer conexiones con programas *esclavos* cuyos códigos fuente no están disponibles, se incorporaron diferentes modos de comunicación, conforme a la estrategia descripta en el Capítulo 2:

- Comunicación por MPI en modos SISD, SIMD, MISD y MIMD con programas con acceso al código fuente.
- Comunicación por lectura y escritura de archivos con programas sin acceso al código fuente.

Todas estas funciones fueron implementadas en la clase `Communicator`. **Newton** puede ser ejecutado en múltiples procesos de modo que la conexión mediante manejo de archivos sea paralelizada. Las tareas se distribuyen automáticamente de forma eficiente conforme a la cantidad de procesos lanzdos por el usuario.

Métodos numéricos

Newton puede resolver acoplamientos utilizando diferentes métodos numéricos:

- Métodos explícitos:
  - método *Piccard* simple;
  - método *Piccard* combinado.
- Métodos implícitos:
  - método de *Newton-Raphson*;
  - método de la secante para sistemas de ecuaciones;
  - método de *Broyden*.

A futuro sería posible implementar otros métodos en la clase `Solver`. Los sistemas algebraicos resultantes en los métodos implícitos son resueltos mediante la biblioteca **PETSc** [40] altamente eficiente para este tipo de cálculos.

Manejo de errores

El manejo eficiente de errores fue pensado como una de las características principales durante el diseño del programa, debido a la complejidad de conexiones y consecuente diversidad en origen de fallas. Cualquier error generado comienza la siguiente cascada de eventos:

1. Devolución de un mensaje correspondiente en la salida estandard.
2. Comunicación del error a todos los procesos de **Newton**.
3. Envío de órden de aborto a los programas *esclavos* por parte del proceso *raiz*.
4. Finalización de las tareas.

Depuración

Se desarrolló un visualizador genérico en la clase `Debugger` para realizar seguimiento de la evolución de las variables pertenecientes a las diferentes clases implementadas. El usuario puede requerir su uso en clases específicas mediante palabras claves ingresadas en el archivo de configuración para la ejecución de **Newton**. `Debugger` va imprimiendo la salida requerida en archivos particulares para cada clase.

Validación

Las funciones implementadas y la metodología de cálculo fueron puestas a prueba exitosamente en la resolución de sistemas de ecuaciones lineales y no lineales acoplados. Los mismos pueden encontrarse dentro de la carpeta `examples` en el directorio principal de **Newton**. Se verificó el correcto funcionamiento de los diferentes tipos de comunicación, de las intancias de mapeos y de los distintos métodos numéricos implementados para la resolución del sistema e ecuaciones de residuos.

B.2. Modelado de problemas acoplados

Newton resuelve problemas que comprenden algún sistema de ecuaciones acopladas, en el que las ecuaciones son resueltas por separado por diferentes programas. Conforme a la estrategia de acoplamiento extendida comentada en la sección 3.4, es necesario definir qué variables van a ser datos y qué variables van a ser incógnitas para cada subsistema de análisis. Como existe la posibilidad de transformaciones previas o posteriores a la resolución del sistema de ecuaciones de residuos característico de cada problema, es necesario además definir en forma clara cuáles van a ser las variables transformadas y los mapeos. Para organizar esta formulación en forma clara `Newton` trabaja con cuatro instancias de variables para cada programa *esclavo*: las instancias α, β, γ y δ.

Alfas, betas, gammas y deltas

Existen cuatro instancias de variables para cada programa *esclavo*:

1. Instancia α: corresponde al valor crudo de las variables calculadas por el programa.

2. Instancia β: corresponde al valor transformado de las variables α. En el caso de que no se haya definido una transformación, se utiliza la transformación de identidad por defecto. Las variables β están listas para ser utilizadas en el cálculo de residuos.

3. Instancia γ: corresponde a los valores *guesses* generados por **Newton** para ese programa.

4. Instancia δ: corresponde a los valores γ transformados, utilizando también la transformación de identidad por defecto.

La Figura B.1 esquematiza lo expresado.

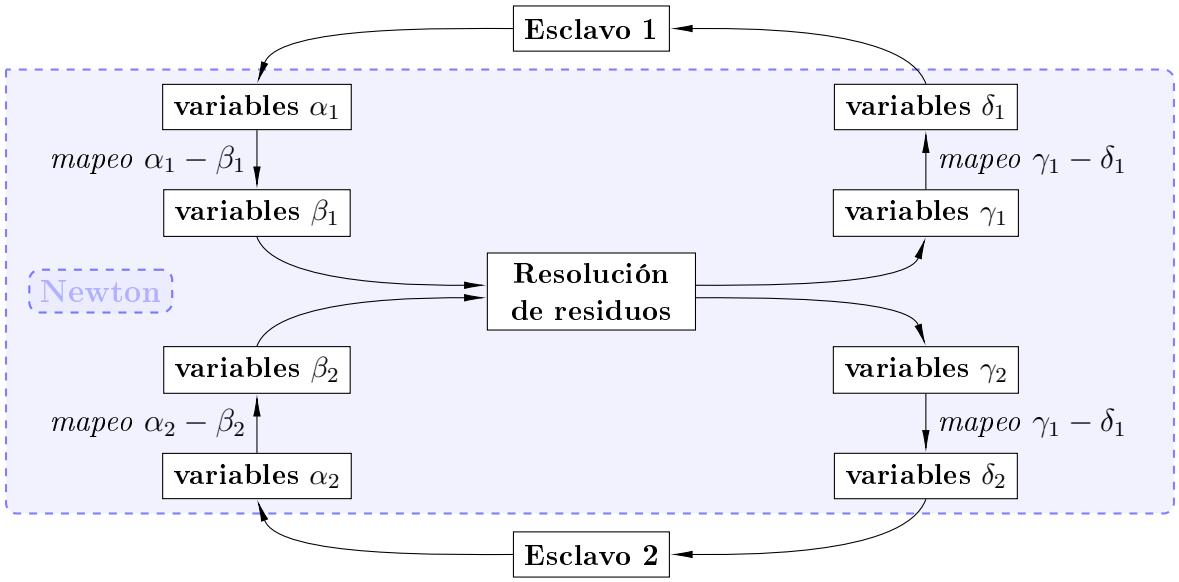


Figura B.1: Instancias de cálculo α, β, γ y δ de las variables relacionadas con cada programa *esclavo*. En este ejemplo los programas **Esclavo 1** y **Esclavo 2** se comunican con **Newton**.

Configuración del problema

Una vez definidas estas instancias para cada subsistema de análisis, debe completarse el archivo de configuración `newton.config`. En el mismo deben definirse *clientes*, que son los diferentes programas *esclavos*. Para cada *cliente* es necesario ingresar el nombre de las variables de instancia β depués de la palabra clave `CALCS`, y de instancia γ después de la palabra clave `GUESSES`. En el caso de que no sean requeridos mapeos no debe especificarse nada sobre las instancias α y δ. Si se requieren transformaciones, éstas deben ser ingresadas en otra instancia del archivo, bajo la palabra clave `MAPPER`. Allí se detalla el *cliente*, si corresponde a una transformación $\alpha - \beta$ o a una transformación $\gamma - \delta$, y la cantidad de variables implicadas en la transformación.

Para cada *cliente* también es necesario ingresar el tipo de comunicación con **Newton**, y otras opciones de conexión en el caso de manejo de archivos.

Finalmente, es necesario definir el método de acoplamiento a utilizar y sus parámetros específicos.

B.3. Metodología de resolución

El archivo `src/main.cpp` contiene la estructura principal del programa. En el mismo se crea un objeto de clase `Newton` y se ejecutan sus tres funciones principales:

1. Función de inicio: `Newton::initialize`.
2. Función de ejecución: `Newton::run`.
3. Función de finalización: `Newton::finish`.

A continuación se describen brevemente las tareas desarrolladas en cada función.

Función de inicio

La función `Newton::initialize` ejecuta todas las tareas necesarias para comenzar el cálculo:

1. Creación de objetos de uso interno.
2. Lectura del archivo de configuración.
3. Creación de comunicadores con procesos *raíces* de programas ejecutados en modos MISD o MIMD junto con **Newton**.
4. Publicación de puertos y establecimiento de conexiones con procesos *raíces* de programas que necesitan establecer comunicación por MPI.
5. Envío de datos generales del problema a los programas *esclavos* para chequeo de compatibilidad de datos.
6. Asignación de espacio en memoria a los arreglos de uso interno dependientes del problema.

Función de ejecución

La función `Newton::run` ejecuta todas las tareas necesarias para completar el cálculo:

1. Ingreso en el bucle de evolución.
   a) Comunicación de valores *guess* para las condiciones de borde de cada subdominio a cada programa *esclavo*:
      1) escritura de los valores en archivos de entrada y ejecución de los programa *esclavos* conectados por manejo de archivos;
      2) envío de valores por MPI a aquellos programas conectados por paso de mensajes.
   b) Recepción de valores calculados para las incógnitas de cada subdominio de cada programa *esclavo*:
      1) lectura de los valores en archivos de salida de los programa *esclavos* conectados por manejo de archivos;
      2) recepción de valores por MPI desde aquellos programas conectados por paso de mensajes.

c) Resolución de las ecuaciones de residuos 1.7.

d) Evaluación de la convergencia:

- si los resultados están convergidos el problema puede continuar con el siguiente paso de evolución;
- si los resultados no están convergidos, es necesario proponer nuevos valores *guess* y reiniciar el cálculo.

e) Envío de orden de avance o de reinicio a cada programa *esclavo* conectado por MPI.

2. Finalización delbucle de evolución.

Función de finalización

La función `Newton::finish` ejecuta todas las tareas necesarias para terminar el programa de forma segura:

1. Desconexión de programas comunicados por MPI.
2. Liberación de comunicadores.
3. Finalización de **PETSc** y de MPI.
4. Cierre de archivos de depuración.

B.4. Ejemplo de uso

A continuación se comenta la resolución de uno de los problemas utilizados para la validación de las funciones desarrolladas. Se quiere resolver el siguiente sistema de ecuaciones:

$$\begin{cases} 1w + 2x + 3y + 4z = 17 \\ 12w + 13x + 14y + 5z = 18 \\ 11w + 16x + 15y + 6z = 19 \\ 10w + 9x + 8y + 7z = 20 \end{cases} \tag{B.1}$$

Uno de los programas *esclavos* calcula (w, x, y) en función de algún valor z_{guess} poporcionado, y resuelve las primeras tres ecuaciones:

$$\begin{cases} w = (17 - 2x - 3y - 4z_{guess})/1 \\ x = (18 - 12w - 14y - 5z_{guess})/13 \\ y = (19 - 11w - 16x - 6z_{guess})/15 \end{cases} \tag{B.2}$$

El otro programa *esclavo* calcula z como función de los valores $(w_{guess}, x_{guess}, y_{guess})$ proporcionados, resolviendo la última ecuación de B.1:

$$z = (20 - 10w_{guess} - 9x_{guess} - 8y_{guess})/7 \tag{B.3}$$

La solución analítica para este problema es la siguiente:

$$\begin{cases} w = -0{,}70909 \\ x = -1{,}89091 \\ y = 2{,}36364 \\ z = 3{,}60000 \end{cases} \tag{B.4}$$

Para formular este problema comunicando los programas por MPI, mediante una ejecución en modo MISD de **Newton** y los dos programas *esclavos*, el archivo de configuración `newton.config` se completa como sigue:

```
 # Métodos numéricos

METHOD secant # Método selecionado
ABS_TOL 1e-15 MAX_ITER 10 # Parámetros de iteración
X_INI w 1 x 2 y 3 z 4 # Semilla inicial

 # Especificaciones para códigos esclavos

CLIENT linear1 # Apodo del primer programa esclavo
ROOT_GLOBAL_RANK 2 # Rango global del proceso raíz
CALCS w x y GUESSES z # Variables de cálculo y variables dato
CONNECTION mpi_comm # Tipo de conexión

CLIENT linear2 # Apodo del primer programa esclavo
ROOT_GLOBAL_RANK 3  # Rango global del proceso raíz
CALCS z GUESSES w x y # Variables de cálculo y variables dato
CONNECTION mpi_comm # Tipo de conexión
```

Notar que en este modo de ejecución **Newton** debe conocer cuál es la numeración global de los procesos raíces de los programas *esclavos* para poder establecer los comunicadores con ellos. Con el método de resolución seleccionado, **Newton** construye la matriz jacobiana del sistema y luego converge a la solución B.4 en una sola iteración, utilizando en total 6 evaluaciones de funciones.

Apéndice C

Acoplamiento de Par-GPFEP

Par-GPFEP (programa de elementos finitos de propósito general paralelizado, el nombre se debe a sus siglas en inglés, *Parallel General Purpose Finite Element Program*) es un código escrito en `Fortran 90` y en `C`, desarrollado inicialmente por G. Buscaglia, S. Felicelli, E. Dari, A. Lew y M. Raschi, y que ha ido teniendo contribuciones a partir de diferentes trabajos y tesis realizadas en el Departamento de Mecánica Computacional de la Comisión Nacional de Energía Atómica ([20], [21]).

En el presente trabajo se utilizaron y modificaron las rutinas relacionadas con el bucle de resolución general y el ensamblaje de matrices para las ecuaciones de *Navier-Stokes*, de modelos de turbulencia, flujo multifase y transporte de energía. Además, se implementaron las rutinas necesarias para realizar el acoplamiento mediante el modelo de comunicación por paso de mensajes a partir de la conexión a puertos publicados por el código *maestro*, de modo que pudiera comunicarse con **Coupling** y con **Newton** de forma exitosa. Éstos trabajos son comentados en la sección C.1. También se desarrolló un programa para la identificación de condiciones de borde sobre conjuntos de nodos específicos, el cual es comentado en la sección C.2.

C.1. Implementación de la arquitectura de acoplamiento en Par-GPFEP

La estructura principal de acoplamiento se compone de cuatro instancias genéricas. Estas cuatro instancias fueron descriptas en la sección 2.4 y se programaron en la rutina `pargpfep_control.for`. Cada una de estas intancias es llamada desde el bucle de resolución general en momentos adecuados. Pero para que este acoplamiento resultara exitoso, fue necesario además implementar una serie de funciones extra:

- Análisis de palabras clave del archivo de configuración `gpfep.cfg` específicas a la formulación del problema de acoplamiento (implementado en el archivo `cfgreadingCoup.for`).

- Seteo paralelizado de variables sobre grupos de superficie conforme a perfiles previstos:
  - para condiciones de borde *esenciales* (modificaciones en el archivo `bcfemco.for`);
  - para condiciones de borde *naturales* (modificaciones en el archivo `nssurf.for`);
- Cálculo integral paralelizado de variables sobre grupos de superficie (implementado en el archivo `sendValuesCoupling.for`).
- Cálculo de campos vectoriales gradientes a partir de campos escalares (modificaciones en el archivo `gpmain.F`).
- Condicionamiento sobre el avance evolutivo para respetar las órdenes de reinicio o continuación recibidas desde programa *maestro* (modificaciones en el archivo `gpmain.F`).
- Resguardo (*back-up*) de las soluciones calculadas debido a la posibilidad de reinicio en determinados pasos de cálculo (modificaciones en el archivo `gpmain.F`).

C.2. El programa genbco

Normalmente, las condiciones de borde de tipo *Dirichlet* se establecen con los programas **genbcc** y **gpboco**. Estos programas imponen para alguna variable dada, el mismo valor a todos los nodos que pertenecen a un grupo de superficie seleccionado. Durante el trabajo realizado, resultó necesario diferenciar los nodos sobre los que se aplicaba las condiciones de borde. Por ejemplo, era necesario sustraer los nodos de las aristas a los nodos de una cara, englobar varias aristas en un mismo grupo, englobar varias caras en un mismo grupo, etc. Con estos fines fue desarrollado el código **genbco**. **genbco** unifica las funciones previamente desempeñadas por los códigos **genbcc** y **gpboco**. Los archivos de entrada necesarios para su uso son el archivo de nodos de superficie y conectividades `.sur` específico de **Par-GPFEP** y el archivo de configuración `genbco.cfg`. En `genbco.cfg` se especifican los conjuntos de nodos sobre los que se desea imponer condiciones de borde de tipo *Dirichlet* para cada variable de cálculo. Esta especificación se realiza a partir de operaciones *booleanas* sobre los grupos originales de superficie. Las operaciones permitidas son:

- unión (U);
- intersección (I);
- sustracción (m).

Sobre cada uno de los conjuntos se especifican, además, los valores iniciales de contorno. El archivo de salida `gp.bco` contiene una lista con todos los nodos y sus condiciones de borde para cada variable y está listo para ser leído posteriormente por **Par-GPFEP**.

C.3. Ejemplo de uso de Par-GPFEP en forma acoplada

A continuación se comentan las consideraciones necesarias en **Par-GPFEP** para la resolución del problema *Análisis del segundo sistema de parada de un reactor de investigación* analizado en **??**.

En primer lugar, es necesario notificar en el archivo de configuración `gpfep.cfg` que las condiciones de borde de algunas interfaces están acopladas. El llamado a las funciones de acoplamiento se activa con la siguiente bandera:

```
! Switch para acople
*COUPLING_SWITCH
1
```

El método de comunicación utilizado es el de conexión a puertos previamente publicados por el programa *maestro*. En este método es necesario definir un código de identificación para cada programa *esclavo*:

```
! ID del código
*CODE_ID
1
```

Las siguientes tarjetas se utilizan para definir las interfaces y las variables de acople:

```
! Cantidad de interfaces de acople
*INTERFACES
1

! Área [m2] de cada interfaz
*INTERFACES_AREAS
0.05067

! Grupo de la malla correspondiente a cada interfaz
*INTERFACES_GROUPS
3
```

```
! Pares de variables en cada interfaz
*UNKNOWN_PAIRS_PER_INTERFACE
1

! Tipo de par intercambiado en cada interfaz:
! (Tipo 1: caudal-presión; Tipo 2: temperatura-flujo de calor)
*INTERFACES_PAIRS_TYPES
1
```

A continuación es necesario definir el tipo de perfil que va a ser utilizado para calcular los vectores de velocidades en cada interfaz con condición de borde de tipo *Dirichlet*.

```
! Tipo de perfil de velocidades para la componente x, componente y,
! componente z, en cada interfaz con seteo de caudal
! (Tipo 0: nulo; Tipo 1: uniforme; Tipo 2: parabólico 2D; Tipo 3:
! parabólico 3D )
*FLOW_PROFILES
1 ! Cantidad de perfiles
1 0 0 ! Tipo para cada componente escalar (x,y,z)
```

Si bien en el modelado del problema se utiliza un perfil de velocidades plano, en ciertas ocasiones puede requerirse el uso de perfiles parabólicos. Para calcular estos perfiles es necesario proveer algunas coordenadas de referencia. Si se hubiera utilizado un perfil parabólico tri-dimensional, y dado que la sección es circular, debe proveerse la posición del centro de la misma y el valor de su radio del siguiente modo:

```
! Coordenadas del punto de referencia de cada interfaz con seteo de
! caudal
*INTERFACES_REFERENCES
0.0 0.0 0.0

! Radio en cada interfaz con seteo de caudal
*INTERFACES_RADIUS
0.127
```

Previo al modelado de futuros problemas acoplados, debe verificarse la implementación de ciertas particularidades bajo la palabra clave `USUARIO` en los archivos comentados en la sección C.1. Por ejemplo, cuando la variable de acople en una interfaz dada es una fuerza, ésta puede recibirse con diferentes signos según la formulación del problema, por lo que es responsabilidad del usuario su correcto uso.

Bibliografía

[1] Leiva, J., Buscaglia, G. Estretegias de acoplamiento entre códigos 0d/1d and códigos cfd 3d. *Mecánica Computacional*, **XXV**, 53–82, 2006. 1, 14, 49

[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, 14, 49

[3] Leiva, J., Blanco, P., Buscaglia, G. C. Partitioned analysis for dimensionally-heterogeneous hydraulic networks. *Multiscale Model. Simul.*, **9**, 872–903, 2011. 1, 14, 49

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

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

[6] Leiva, J., Blanco, P. J., Buscaglia, G. Iterative strong coupling of dimensionally heterogeneous models. *International Journal for Numerical Methods in Engineering*, **81**, 1558–1580, 2009. 5, 49

[7] Dennis, J., Shnabel, R. Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Society for industrial and applied mathematics, 1996. 7, 8

[8] Broyden, C. A class of methods for solving nonlinear simultaneous equations. *American Mathematical Society*, **19**, 577–593, 1965. 7

[9] Kelley, C. Iterative Methods for Linear and Nonlinear Equations. siam, 1995. 8, 46

[10] Hendrickson, B., Devine, K. Dynamic load balancing in computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, págs. 485–500, 2000. 11

[11] Calviño, B. Estrategias de descomposición en dominios para entornos grid. *Tesis Doctoral, Universidad Politécnica de Cataluña*, 2007. 12

[12] Newton: a multiphysics coupling master code. *Repositorio GitHub: https://github.com/fedecaccia/newton*, 2017. 16

[13] Buscaglia, G., Dari, E., Martín, E., Arnica, D., Bonetto, F. 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. 21

[14] Iedelchik, I. Handbook of Local Resistance and Friction. 1960. 24, 31

[15] Kays, W., Crawford, M. Convective Heat and Mass Transfer, 3rd Edition. 1993. 24

[16] Gunzburger, M. Finite Element Methods for Viscous Incompressible FLows. 1989. 24, 32

[17] 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. 25, 32

[18] 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. 25, 32

[19] Geuzaine, C., Remacle, J. Gmsh reference manual, 2016. 25

[20] Buscaglia, G., Felicelli, S. Un sistema de generación de programas de elementos finitos. *Documentación: Versión 1.1*, 1995. 25, 63

[21] Buscaglia, G., Dari, E., Lew, A., Raschi, M. Un programa general de elementos finitos en paralelo. *Mecánica Computacional*, págs. 845–854, 1999. 25, 63

[22] Buscaglia, G., Dari, E. Numerical investigation of flow through a cavity with internal heat generation. *Numerical Heat Transfer*, págs. 525–541, 2003. 26

[23] Rechiman, L., Cantero, M., Dari, E., Caccia, F., Chacoma, A. Análisis hidrodinámico del segundo sistema de parada del ra-10. *Informe técnico CNEA IN-ATN40MC-03/2015*, 2015. 28

[24] Rechiman, L., Cantero, M., Caccia, F., Dari, E., Chacoma, A. Three-dimensional hydrodynamic modeling of the second shutdown system of an experimental nuclear reactor. *Nuclear Engineering and Design*, **319**, 163–175, 2017. 29, 33

[25] Rechiman, L., Cantero, M., Caccia, F., Dari, E. Validation of a multiscale model of the sencond shutdown system of an experimental nuclear reactor. *Mecánica Computacional*, **XXXIV**, 2199–2215, 2016. 29

[26] Invap. Segundo sistema de parada - validación del cfd para drenaje del tanque reflector, 2013. 29, 31, 34

[27] Rechiman, L., Cantero, M., Dari, E., Chacoma, A. Modelo multiescala del mockup del segundo sistema de parada del reactor ra-10. *Informe técnico CNEA IN-ATN40MC-02/2015*, 2015. 29

[28] Bird, R., Stewart, W., Lightfoot, E. Transport Phenomena, 2nd Edition. 2002. 30

[29] CEA/DEN, E. R., CASCADE, O. Salome version 7.8.0: Public release announcement, 2016. 31

[30] Lew, A., Buscaglia, G., Carrica, P. A note on the numerical treatment of the k-epsilon turbulence model. *International Journal of Computational Fluid Dynamics*, 2001. 32

[31] Durbin, P., Reif, B. P. Statistic Theory and Modeling for Turbulent Flows. 2011. 32

[32] Mohammadi, B., Pironneau, O. Analysis of the k-epsilon turbulence model. 1994. 32

[33] Lew, A. El método de elementos finitos en entornos computacionales de alta performance. *Trabajo Especial Carrera de Ingeniería Nuclear, Instituto Balseiro*, págs. 57–64, 1998. 32

[34] Osher, S., Fedkiw, R. Level Set Methods and Dynamic Implicit Surfaces. 2003. 36

[35] Ausas, R., Buscaglia, G., Idelsohn, S. A new enrichment space for the treatment of discontinuous pressures in multi-fluid flows. *International Journal for Numerical Methods in Fluids*, 2011. 36

[36] White, F. Fluid Mechanics, 4th edition. Mc Graw Hill Education, 1998. 41, 42

[37] Griewank, A. Broyden updating, the good and the bad!, 2000. 44

[38] GitHub, Inc. Plataforma de desarrollo colaborativo. *https://github.com*, 2008. 55

[39] Newton: a multiphysics coupling master code user's manual. *Repositorio GitHub: https://github.com/fedecaccia/newton/doc/newton-u-m.pdf*, 2017. 56

[40] Balay, S., Abhyankar, S., Adams, M. F., Brown, J., Brune, P., Buschelman, K., *et al.* PETSc Web page. http://www.mcs.anl.gov/petsc, 2016. URL http://www.mcs.anl.gov/petsc. 57

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-03/2015 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.