FICH UNL

UNIVERSIDAD NACIONAL DEL LITORAL
Facultad de Ingeniería y Ciencias Hídricas

Optimización de un código para la identificación de parámetros en modelos capacitivos-resistivos (CRM) de reservorios de petróleo

Proyecto Final de Carrera

Alumno: Zilli, Matías Emanuel.
Director: Storti, Mario Alberto.

Año: 2017

Contenido

Índice de Figuras

Índice de Tablas

Lista de acrónimos

ALM: Augmented Lagrangian Method.
BHP: Bottom Hole Pressure.
CRM: Capacitive Resistive Method.
CRMIP: Capacitive Resistive Method for Injector-Producer pair.
CRMP: Capacitive Resistive Method for Producers.
CRMPK: Capacitive Resistive Method for Producers preseving “K”.
CRMT: Capacitive Resistive Method for single Tank.
KKT: Karush Kuhn Tucker.
LM: Lagrange Multiplier.
LS: Line Search.
MLR: Multiple Linear Regression.
NR: Newton Raphson.
STB/D: Stock Tank Barrel per Day.
STL: Standard Template Library for C++.

Lista de símbolos

τ_j : Constante de tiempo del productor j.

q_j : Caudal de producción en el productor j.

q_j^{obs} : Caudal observado en el productor j.

I_i : Caudal de inyección en el inyector i.

F_{ij} : Constante de transmisibilidad entre el productor j y el inyector i.

K_{ij} : Constante de transmisibilidad entre el productor j y el inyector i.

N_i : Número de inyectores.

N_s : Número de etapas.

N_r : Número de restricciones.

N_F : Número de parámetros F.

N_p : Número de productores.

N_{step} : Numero de pasos de tiempo u observaciones.

N_{dof} : Número total de parámetros a identificar.

G_j : Tiempo característico del productor j.

s : Número de etapa.

λ : Multiplicador de Lagrange.

μ : Coeficiente de penalización.

g_j : Restricciones.

Palabras claves

CRMP, capacitivo-resistivo, petróleo, CRMPK, ALM, CIMEC.

Resumen

Actualmente en el campo de extracción de petróleo existe una tendencia al uso de algoritmos computacionales para simular situaciones particulares y estimar ciertos parámetros. Hoy día los simuladores basados en el método CRM(Capacitive Resistive Method)[6], utilizados en proyectos de recuperación secundaria de petróleo, resuelven el problema de identificación de parámetros sin tener en cuenta las relaciones existentes entre las diferentes etapas (configuraciones de inyectores-productores). Es decir, si la configuración de productores alrededor de un inyector cambia de una etapa a otra, los parámetros de la nueva etapa se asumen completamente diferentes a los de la etapa anterior. Sin embargo, en la práctica, las conductancias hidráulicas de una nueva etapa serán prácticamente las mismas que las de la etapa anterior, por lo tanto existen ciertas relaciones que se preservan. Estas últimas pueden ser aprovechadas de manera tal de reducir la cantidad de parámetros a identificar, lo cual da ciertas ventajas.

El presente trabajo propone implementar modificaciones a la librería IDENT[1] desarrollada por el CIMEC[16], que implementa el método CRM, de forma que este contemple las relaciones entre etapas y a la vez, que mejore la los tiempos de ejecución.

1. Introducción

Hoy día, en lo que refiere a extracción de petróleo a través de técnicas convencionales, el procedimiento basado en pozos inyectores y productores es uno de los más utilizados. Este método consiste en una serie de perforaciones en las que en algunas se inyecta agua a alta presión (inyectores), y en otras se espera un caudal saliente de petróleo (productores). En la Fig. 1 se muestra una imagen de satélite perteneciente a un reservorio ubicado en la zona del Golfo San Jorge, cerca de Pico Truncado, Santa Cruz, Argentina; en dicha imagen se pueden observar los productores e inyectores los cuales aparecen como puntos de color marrón claro. Cabe destacar que esta configuración puede cambiar en el tiempo, ya sea porque se agregan pozos productores o inyectores, o bien porque se cierran. Cada intervalo de tiempo en el cual la configuración se mantiene inalterada se llama “etapa”.

Para llevar a cabo el procedimiento de extracción, se comienza realizando las perforaciones pertinentes y luego se procede a obtener un conjunto de parámetros subterráneos, como son conductividad hidráulica entre cada par productor-inyector y tiempo característico de cada pozo productor, el cual nos da una idea del lapso que tendrá que transcurrir desde que se comienza a inyectar agua hasta que finalmente emerge petróleo.

Para obtener los mencionados parámetros es posible realizar estudios geológicos aunque actualmente existe una creciente tendencia al uso de algoritmos computacionales para estimar estos parámetros a partir de los mismos datos de producción e inyección. Uno de los modelos más populares es el modelo CRM [6] el cual será descrito en detalle en el Cap. 1. Los CRM permiten cuantificar la comunicación entre pozos productores e inyectores proveyendo importante información, basándose solamente en datos de caudal de producción y de inyección de los pozos involucrados.

Actualmente los algoritmos de identificación de parámetros basados en CRM, utilizados en proyectos de recuperación secundaria de petróleo, resuelven el problema de identificación de los parámetros sin tener en cuenta las relaciones existentes entre las diferentes etapas (configuraciones). Es decir, si la configuración de productores alrededor de un inyector cambia de una etapa a otra, los parámetros de la nueva etapa se asumen completamente diferentes a los de la etapa anterior aunque, en la práctica, las conductancias hidráulicas de una nueva etapa serán prácticamente las mismas que las de la etapa anterior, por lo tanto existen ciertas relaciones que se preservan y pueden ser aprovechadas para reducir la cantidad de parámetros a identificar y además tener mayor cantidad de información por parámetro.

En el presente trabajo se propone implementar un cambio al algoritmo basado en el modelo CRM, el cual está implementado en la librería IDENT[1] de forma que este contemple las relaciones contenidas en los datos de entrada y a la vez, que sea capaz de mejorar su rendimiento computacional en términos de tiempo. Este nuevo modelo recibirá el nombre de CRMPK.

Este proyecto abordará la implementación del modelo CRMPK en lenguaje de programación C++ y la inclusión del mismo en la librería IDENT, además se realizarán optimizaciones a la librería IDENT actualmente implementada.

Finalmente el modelo se pondrá a prueba con diferentes casos reales de pozos con la finalidad de obtener resultados de tiempo de ejecución y precisión los cuales se contrastarán con los resultados obtenidos con el método CRMP actualmente implementado en IDENT.

Figura 1: Captura de satélite ampliada de un reservorio mostrando productores e inyectores

2. El modelo CRMP

2.1. Origen

El estudio de las propiedades de un reservorio es de gran importancia al planear, monitorear y optimizar la producción de petróleo. El uso de simulaciones numéricas es una excelente técnica utilizada en el estudio de reservorios pero demanda un gran número de datos de entrada, entre éstos se requiere información de varios meses para llevar a cabo una simulación. Un método de simulación numérica de reservorios es aquel basado en el modelo CRM [6].

Este modelo puede observarse desde el punto de vista de la electrónica en el cual una señal de entrada (inyecciones) es convertida en una señal de salida (producción), de manera similar a como el potencial eléctrico es convertido a voltaje o corriente en un circuito capacitor-resistor. De aquí el nombre de modelo capacitivo-resistivo.

El modelo capacitivo-resistivo ha sido desarrollado basándose en la hipótesis de que las propiedades del reservorio pueden ser inferidas a partir de los datos de producción de petróleo e inyección de agua.

Comparado con los métodos tradicionales de simulación de reservorios, el CRM ofrece una rápida evaluación del comportamiento entre inyectores y productores debido a que este modelo sólo requiere datos de inyección y producción.

Varios investigadores han estado estudiando el tema desde la primera propuesta originada del trabajo de Albertoni (2002) y Albertoni & Lake (2003). Estos últimos propusieron un método estadístico que hacía uso de regresiones lineales múltiples (MLR) para obtener los coeficientes de conectividad entre inyectores y productores. El método predecía los caudales de producción de petróleo de un pozo utilizando un modelo lineal que asumía una combinación lineal de caudales de inyección de cada inyector del campo, el cual también, tenía en cuenta la disipación dentro del reservorio al utilizar lo que los autores denominaron filtros de difusividad.

Yousef et al.(2005) mejoró el modelo, cuantificando de forma numérica la comunicación entre pozos inyectores y productores basándose en los caudales de inyección y producción. Además él tuvo en cuenta los efectos capacitivos (compresibilidad) y resistivos (transmisibilidad).

Para tener en cuenta los efectos antes mencionados se utilizan dos parámetros, en primer lugar, la constante de transmisibilidad (coeficiente de flujo) que indica la fracción de inyección que contribuye a la producción, es decir cuantifica la conectividad entre cada par inyector productor, y en segundo lugar la constante de tiempo de un productor que caracteriza el retardo entre inyecciones y producciones. Esta propuesta, tiene la propiedad de poseer filtros intrínsecos los cuales tienen en cuenta la atenuación y el retraso en la respuesta entre pares productor inyector.

Además, se han propuesto distintos tipos de soluciones para las ecuaciones diferenciales del modelo de Yousef, una de estas es según el volumen de control estudiado (CRMT: Capacitive Resistive Method for single Tank, CRMP: Capacitive Resistive Method for Producers, CRMIP: Capacitive

Resistive Method for Injector-Producer pair) (Sayarpour, 2008), otra de ellas se basa en no tener en cuenta la presión del fondo del pozo (BHP) (Liang, 2007).

Weber (2009) aplico el modelo CRMP en distintos casos de reservorios de gran escala en donde se tuvo en cuenta por primera vez que a lo largo de la vida del reservorio, los pozos productores e inyectores se abren, se cierran o se reconvierten (de productor a inyector o viceversa), entonces un reservorio no está ligado a una única configuración de inyectores-productores, si no que éste puede pasar por muchas configuraciones a lo largo de su vida, cada una de éstas es llamada “etapa”.

2.2. La variante CRMP

El modelo CRMP[6] propone construir el volumen de control con el volumen de drenaje alrededor de un productor dado, esto se ilustra en la siguiente figura.

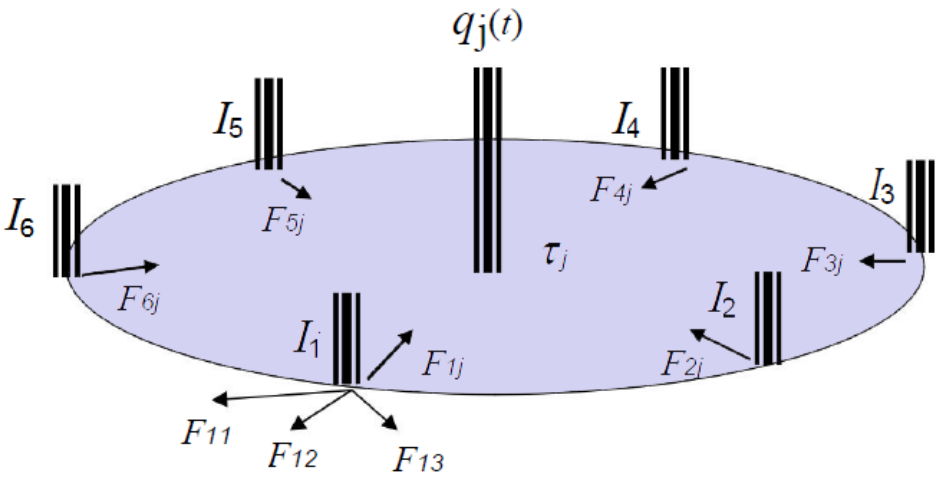

Figura 2: Configuración de inyectores-productores.

El modelo se caracteriza por la siguiente ecuación de continuidad para el volumen de producción total de un dado pozo productor j a partir de N_i pozos inyectores asociados al mismo.

$$\tau_j \dot{q}_j + q_j = \sum_{i=1}^{N_i} F_{ij} I_i \qquad (1)$$

donde q_j es el caudal de producción en el productor j debido a la inyección en los N_i inyectores, τ_j su constante de tiempo, I_i es el caudal de inyección en el inyector i, F_{ij} representa la constante de transmisibilidad entre el productor j y el inyector i. Se asume que BHP=0.

Para llevar a cabo la discretización temporal de la ecuación anterior, se consideran N_{step} observaciones, los instantes de tiempo $t_n = n\Delta t$, $n = 0,..,N_{step}$, y

por simplicidad un sólo productor y un inyector. Se denotarán los valores discretos $q(t^n)$ como el valor del caudal q en t^n, mientras que q^n es el valor promedio en el intervalo $T^n = [t^{n-1}, t^n]$. Por otra parte se asume que $I(t)$ es constante en el intervalo T^n.

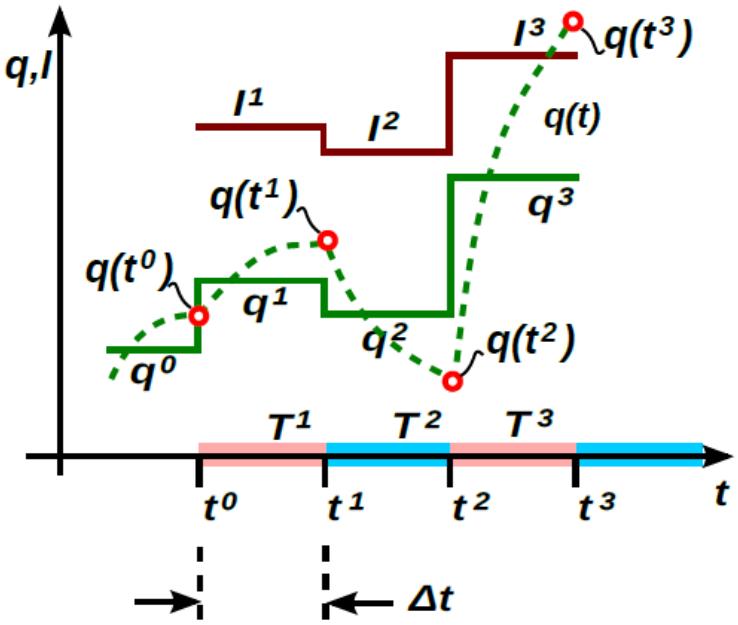

Figura 3: Aproximación de los caudales como funciones constantes a trozos.

Entonces si se resuelve la ecuación 1 en el intervalo $T^{n+1} = [t^n, t^{n+1}]$ se tiene

$$q(t) = FI^{n+1} + (q(t^n) - FI^{n+1})e^{-(t-t^n)/\tau}, \text{ para } t \in T^{n+1} \quad (2)$$

Por lo tanto

$$q(t^{n+1}) = FI^{n+1} + (q(t^n) - FI^{n+1})e^{-\Delta t/\tau} \quad (3)$$

donde $G = e^{-\Delta t/\tau}$ y F representa la constante de transmisibilidad (coeficiente de flujo) entre el productor y el inyector.

Si se aproxima el caudal promedio q^{n+1} como $q^{n+1} \approx \frac{q(t^{n+1}) + q(t^n)}{2}$, se obtiene la siguiente ecuación recursiva en el tiempo (discreto) para los caudales promedio a partir de sumar la ecuación 3 para $q(t^n)$ y $q(t^{n+1})$, esto es

$$q^{n+1} = Gq^n + (1-G)FI^{n+1} \quad (4)$$

Para identificar los parámetros de este modelo (coeficiente de flujo F y constante de tiempo τ), en primer lugar se construye una función objetivo Φ que consiste en la diferencia entre los caudales observados y los estimados a partir de la ecuación 4. Finalmente los parámetros son identificados a partir de una regresión lineal multivariable que minimiza dicha función objetivo la cual se muestra a continuación

$$\begin{aligned}\Phi &= \sum_{k=1}^{N_t}\sum_{j=1}^{N_p}(q_j^{k,obs} - q_j^k)^2 \\ &= \sum_{k=1}^{N_t}\sum_{j=1}^{N_p}(q_j^{k,obs} - (G_j q_j^{k-1} + (1-G_j)\sum_{i=1}^{N_i} F_{ij} I_i^k))^2\end{aligned} \tag{5}$$

donde $q_j^{k,obs}$, es el caudal de producción del productor j observado en el instante k, I_i^{k+1} es el caudal de inyección del inyector i observado en el instante $k+1$, q_j^k el valor total de producción calculado por el modelo, $G_j = e^{-\Delta t/\tau_j}$ es el tiempo característico del productor j, N_p el número de productores, N_t el número total de observaciones y N_i el número de inyectores.
Ésta última ecuación está sujeta a tres tipos de restricciones, ellas son

$$\begin{aligned}&\sum_{j=1}^{N_p} F_{ij} \le 1 && \forall i \\ &F_{ij} \ge 0 && \forall i \\ &\tau_j \ge 0, G_j \le 1 && \forall j\end{aligned} \tag{6}$$

La primera restricción corresponde a una ecuación de balance, la cual permite contemplar el caso en el cual parte del agua inyectada se pierda, es decir el volumen total de líquido inyectado será mayor al volumen total de producción. La segunda y tercer restricción aseguran que el agua inyectada no afecte la producción del reservorio, es decir un caudal no puede ser negativo y una constante de tiempo tampoco puede serlo ya que si no implicaría que los caudales crecen exponencialmente en el tiempo.

2.2.1. Identificación multietapa

Es de interés identificar los parámetros F y τ para los casos en que un productor se activa o desactiva. Al activarse un productor j puede ser que el coeficiente de interacción F_{ij} del productor con un inyector i cambie o no. Por lo tanto se dividen los pasos de tiempo N_t en ns etapas. Dentro de cada etapa se asume que los parámetros son constantes, y cada productor no cambia de estado (pasar de estar activo a inactivo o viceversa). En una dada etapa s un cierto par productor/inyector $(j;i)$ puede estar activo o no, con un dado

coeficiente de interacción $F_{s,ij}$. En la etapa siguiente $s+1$ este par puede estar activo o no, y si está activo su coeficiente puede ser el mismo ($F_{s+1,ij} = F_{s,ij}$) o haber cambiado ($F_{s+1,ij} \neq F_{s,ij}$).

Por ejemplo en el caso de la Fig. 4 se consideran dos etapas N_s = 2. En la primera (s = 1) hay dos productores y un inyector con sus correspondientes coeficientes de interacción F_1, F_2. En la segunda etapa (s = 2) se agrega un productor P_3 y un inyector I_2. El nuevo productor interactúa con el nuevo inyector y con el I_1, pero no con el I_2. Notar que el coeficiente de interacción I_1/P_1 cambia de una etapa a la otra (pasa a ser F_3), mientras que el I_1/P_2 no cambia. Notar también que no todos los pares interactúan. Cabe destacar que por conveniencia (debido a la notación utilizada por IDENT), a lo largo del documento se utilizará una notación con un sólo subíndice entero para los coeficientes de flujo, para este caso la correspondencia es $F_{1,11} = F_1$, $F_{1,12} = F_2$, $F_{2,11} = F_3$, $F_{2,12} = F_2$, $F_{2,13} = F_4$, $F_{2,23} = F_5$.

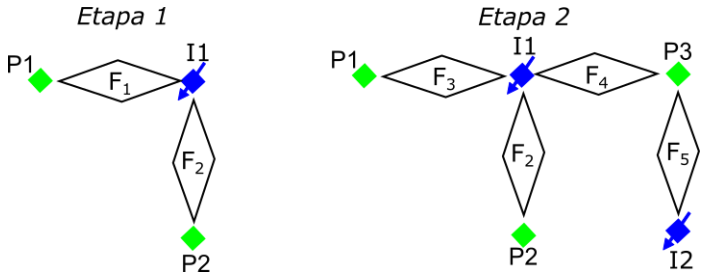

Figura 4: Configuración de campo con dos etapas. En la segunda etapa s = 2 se agrega un productor P_3 y un inyector I_2.

Debido a ésta configuración elegida se tienen como incógnitas del modelo 8 parámetros, a saber los 3 tiempos característicos de los productores $\tau_1,..,\tau_3$ y los 5 coeficientes de flujo $F_1,..,F_5$. La etapa (s = 1) dura para los primeros n_1 pasos de tiempo, es decir en el rango $n \in [1, n_1]$. La etapa s = 2 vale en el rango $n \in [n_1+1, n_2]$.

Cabe destacar que cada paso de tiempo (observación) tiene una duración de un mes por convención y los valores de inyección y producción están expresados en unidades de STB/D, es decir Stock Tank Barrel per Day (barril de tanque de almacenamiento por día).

2.3. Resolución del sistema de ecuaciones

Para resolver el problema de identificación de parámetros del modelo CRMP se deben resolver las ecuaciones de forma tal de satisfacer las restricciones propias de este modelo, esto es, encontrar el mínimo de un funcional con restricciones. Entonces si $\mathrm{X} \in \Re^N$ es el vector de parámetros a identificar y $\Phi(\mathrm{X})$ es el funcional a minimizar, X debe satisfacer $\Phi(\mathrm{X}) \leq \Phi(\mathrm{Y})$, $\forall \mathrm{Y} \in \Re^N$ tal que se cumpla $g_j(\mathrm{Y}) \leq 0$, $j = 1,..,N_R$, donde g_j son las

restricciones y N_R es el número de restricciones. Es de destacar que si $g_j(\mathrm{Y}) \leq 0$ entonces las N_R restricciones se cumplen.

Existen varios algoritmos para resolver problemas de optimización con restricciones. Como introducción se presentarán dos de estos, los cuales son la base del método ALM (Augmented Lagrangian Method) [7] que es el que se utiliza en este trabajo.

2.3.1. Método de penalización

El método de penalización [7] reemplaza un problema de optimización (búsqueda de mínimo) con restricciones por una serie de problemas de optimización sin restricción cuya solución idealmente converge a la solución del problema original (con restricciones). Los problemas sin restricciones son construidos a partir de incorporar un término llamado función de penalización a la función a minimizar, el cual consiste de un coeficiente de penalización multiplicado por una "medida" de violación de las restricciones. Esta medida de violación de restricción es cero cuando las restricciones se cumplen y es distinto de cero cuando las restricciones son violadas.
Una función de penalización muy utilizada es la función de penalización cuadrática la cual consiste básicamente en el cuadrado de la suma de las restricciones.

El coeficiente de penalización incide en la magnitud de la penalización, esto es, haciendo este coeficiente grande, se penaliza a la función más severamente, forzándola a permanecer dentro de la región en la cual se cumplen las restricciones. En teoría se debería elegir un coeficiente suficientemente elevado, pero esto trae problemas de mal condicionamiento, de forma que existe un compromiso entre el valor de éste y la convergencia del sistema.

El funcional de este método para el caso de restricciones unilaterales viene dado por

$$\Phi_P(\mathrm{X}) = \Phi(\mathrm{X}) + \mu \sum_j (g_j(\mathrm{X})^+)^2 \tag{7}$$

donde $\Phi(\mathrm{X})$ es el funcional a minimizar, μ es el coeficiente de penalización y g_j las restricciones. Además $(\mathrm{X})^+$ representa la parte positiva de X, es decir

$$(\mathrm{X})^+ \begin{cases} \mathrm{X} & \textbf{si } \mathrm{X} > \mathbf{0} \\ \mathbf{0} & \textbf{si } \mathrm{X} \leq \mathbf{0} \end{cases} \tag{8}$$

La ventaja de este método es que es fácil de implementar. Para μ finito las restricciones no se cumplen de forma exacta. Para hacerlo exacto se debería hacer tender μ a infinito, pero como se mencionó esto trae problemas de condicionamiento.

2.3.2. Método de multiplicadores de Lagrange

El método de multiplicadores de Lagrange [7] es otra de las estrategias para encontrar el mínimo de un funcional sujeto a restricciones g_j. Este método introduce una nueva variable λ llamada multiplicador de Lagrange y su funcional viene dado por

$$\Phi_{LM}(\mathrm{X},\lambda) = \Phi(\mathrm{X}) + \sum_j \lambda_j g_j(\mathrm{X})^+ \tag{9}$$

Este funcional debe ser minimizado con respecto a X y a los multiplicadores λ.
Si $\Phi(\mathrm{X}_0)$ es un mínimo de $\Phi(\mathrm{X})$ para el problema de optimización original, entonces existen constantes λ_j tal que (X_0,λ) son mínimos en un punto estacionario de la función de Lagrange (un punto estacionario es aquél donde las derivadas parciales de $\Phi_{LM}(\mathrm{X},\lambda)$ son cero). Sin embargo no todos los puntos estacionarios llevan a la solución del problema original, sino que además se deben satisfacer las siguientes condiciones de Karush-Kuhn-Tucker (KKT) [7] para restricciones unilaterales, tales condiciones son necesarias y si el funcional es cóncavo y las restricciones g_j son convexas, también son suficientes.

$$\begin{aligned} & g_j(\mathrm{X}) \le 0 \\ & \lambda_j \ge 0 \\ & g_j \lambda_j = 0 \end{aligned} \tag{10}$$

Una dada restricción g_j se dice que está activa si $g_j(\mathrm{X}) = 0$ y $\lambda_j \ge 0$. Si se conocieran de antemano cuáles son las restricciones activas, entonces el problema se reduce a un sistema de ecuaciones no-lineales en X y los multiplicadores activos. Sin embargo, si el número de restricciones es alto entonces probar con todas las combinaciones de conjuntos de restricciones activas es un problema combinatorio y se vuelve impráctico.

2.3.3. Método de Langragianos aumentados

Por último, se describe un método conocido como método de lagrangianos aumentados [7]. Éste está relacionado con los dos métodos vistos anteriormente (método de penalización y método de multiplicadores de Lagrange), de forma tal que se itera una solución con penalización, y en cada paso la penalización utilizada se utiliza para calcular una aproximación al multiplicador de Lagrange correspondiente.

A diferencia del método de penalización, este reduce la posibilidad de mal condicionamiento introduciendo explícitamente una aproximación al multiplicador de Lagrange dentro de la función a ser minimizada, la cual es conocida como lagrangiano aumentado y tiene la propiedad de ser suave.
Supongamos que se conoce una aproximación a los multiplicadores λ_j y sea

$$\phi_j(\mathrm{X}) = \begin{cases} \frac{\mu}{2}(g_j(\mathrm{X})^+)^2 & \text{si } \lambda_j = 0 \\ \frac{\mu}{2}(g_j(\mathrm{X}))^2 & \text{si } \lambda_j > 0 \end{cases} \tag{11}$$

Entonces al igual que en el método de penalización, se transforma el problema original con restricciones en uno de minimización sin restricciones el cual tiene el siguiente funcional

$$\Phi_{ALM}(\mathrm{X}) = \Phi(\mathrm{X}) + \sum_j \lambda_j g_j(\mathrm{X})^+ + \sum_j \phi_j(\mathrm{X}) \tag{12}$$

Aquí μ es un parámetro algorítmico que debe elegirse. La minimización de este último funcional se puede realizar mediante el método de Newton-Raphson (NR) [13]. Una vez obtenido el mínimo X^k los multiplicadores se actualizan de la siguiente manera

$$\lambda_j^{k+1} = (\lambda_j^k + \mu \sum_j g_j(\mathrm{X}))^+ \tag{13}$$

En condiciones apropiadas el algoritmo converge, es decir $(\mathrm{X},\lambda)^k \to (\mathrm{X},\lambda)^*$, que es la solución del problema de optimización planteado con multiplicadores de Lagrange.

La convergencia de este método puede ser asegurada sin incrementar μ indefinidamente, pero si μ es muy pequeño la penalización es débil y la solución de cada iteración del algoritmo es rápida y bien condicionada, pero $(\mathrm{X},\lambda)^k$ converge lentamente a $(\mathrm{X},\lambda)^*$. Por el contrario, si μ es grande la solución en cada iteración requiere mucho esfuerzo (dado que hay que resolver un problema nolineal mal condicionado), pero $(\mathrm{X},\lambda)^k$ converge a $(\mathrm{X},\lambda)^*$ en pocas iteraciones.

2.3.4. Implementación y pruebas

Con el objetivo de profundizar en la comprensión del método ALM aplicado al problema de identificación CRMP, se llevó a cabo el desarrollo de un algoritmo en Octave[17] que resuelve un caso simple de identificación de parámetros.

Se ha resuelto el caso de la siguiente figura considerando un esquema atemporal (τ_j = 0 o equivalentemente G_j = 0) de dos etapas el cual posee un productor y un inyector en la etapa 1 y tres productores y dos inyectores en la etapa 2:

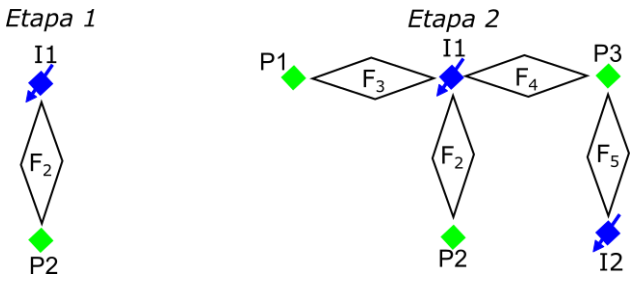

Figura 5: Configuración de inyectores-productores del caso a probar.

El sistema de ecuaciones proveniente de la configuración anterior viene dado por:

Etapa 1:

$$q_2^k = F_2 I_1^k \qquad k = 1 \tag{14}$$

Etapa 2:

$$\begin{aligned} q_1^k &= F_3 I_1^k \\ q_2^k &= F_2 I_1^k \qquad k = 2,3 \\ q_3^k &= F_4 I_1^k + F_5 I_2^k \end{aligned} \tag{15}$$

Aquí, el superíndice k denota observación k.
Este último sistema de ecuaciones está sujeto a las siguientes restricciones en consonancia con la ecuación 6:

$$\begin{aligned} &F_2 \le 1 \\ &F_2 + F_3 + F_4 \le 1 \\ &F_5 \le 1 \\ \\ &F_j \ge 0 \end{aligned} \tag{16}$$

Para identificar los parámetros se construye el siguiente funcional a partir de las ecuaciones 14 y 15 tal como se definió en la ecuación 5. Dicho funcional consiste en la diferencia entre los cuadrados de los caudales observados $q_j^{k,obs}$ y los q_j^k que se obtienen a partir de los parámetros F_j estimados. El número total de parámetros a identificar es 4, a saber F_j con $j \in [2,5]$.

$$\Phi(F_i) = \sum_{k=1}^{1} (q_2^{k,obs} - F_2 I_1^k)^2 + \sum_{k=2}^{3} (q_1^{k,obs} - F_3 I_1^k)^2 + (q_2^{k,obs} - F_2 I_1^k)^2 + (q_3^{k,obs} - F_4 I_1^k + F_5 I_2^k)^2 \tag{17}$$

Para llevar a cabo la identificación de forma correcta es necesario tener al menos tantas ecuaciones como parámetros a identificar, de manera que el sistema de ecuaciones a resolver sea compatible determinado. Debido a esto

último se utilizó una observación para la etapa 1 ($k=1$, dando como resultado una ecuación) y dos observaciones en la etapa 2 ($k=2,3$, dando como resultado 6 ecuaciones), con lo cual tenemos un sistema de ecuaciones con 4 incógnitas y 7 ecuaciones el cual es compatible determinado.

Los valores correspondientes a caudales de inyección para cada observación se generaron de forma aleatoria y se definen a priori los siguientes valores de F_j, $F_2=0.4$, $F_3=0.5$, $F_4=0.1$, $F_5=0.1$, con los cuales se obtienen los valores de los caudales de producción q_j para cada observación. A partir de los caudales de producción antes obtenidos se perturban estos introduciéndoles un ruido aleatorio uniforme de 10% para simular un problema real los cuales son afectados por distintos fenómenos físicos.

Luego con estos últimos valores de inyección y producción se efectúa la identificación; es de esperar que no se obtengan a los parámetros impuestos F_j, sino que obtendrá a una aproximación a estos debido a que se ha introducido ruido.

Resolviendo el sistema sin restricciones los resultados obtenidos fueron los siguientes: $F_2=0.4891, F_3=0.5900, F_4=0.1565, F_5=0.1330$, estos últimos no cumplen con las restricciones impuestas ya que $F_2+F_3+F_4>1$.

Ahora resolviendo el sistema con restricciones mediante método de lagrangianos aumentados los resultados obtenidos fueron los siguientes: $F_2=0.4481, F_3=0.5262, F_4=0.0257, F_5=0.2284$. Con esto se concluye que como era de esperar, no se ha arribó a los valores de F_j impuestos debido a la presencia de ruido en los caudales de producción, pero con estos valores aproximados obtenidos se han cumplido las restricciones impuestas (16) tal como lo establecen las ecuaciones 6.

2.4. Implementación de IDENT

2.4.1. Descripción del algoritmo

La librería IDENT[1] encuentra una solución particular a las ecuaciones diferenciales del modelo CRMP[6] y fue desarrollada por el CIMEC para el simulador Sahara®, descripto en el Cap. 5 de este trabajo.

$$\tau_j \dot{q}_j + q_j = \sum_{i=1}^{N_i} F_{ij} I_i \tag{18}$$

Para hallar la solución de éstas ecuaciones (descritas anteriormente en la Sec. 2.2) es necesario identificar los F_{ij} y τ_j sujetos a sus respectivas restricciones, para dados I_i y q_j. Ésta última tarea es la principal que realiza el código y la que más esfuerzo computacional conlleva.

La solución en forma discreta a las ecuaciones diferenciales anteriores viene dada por

$$q_j^{n+1} = G_j q_j^n + (1-G_j)\sum_{i=1}^{N_i} F_{ij} I_i^{n+1}, n = 1,..,M-1 \quad (19)$$

$$G_j = e^{-\Delta t/\tau_j} \quad (20)$$

donde M es el número de pasos de tiempo Δt.

Para llevar a cabo la identificación de los mencionados parámetros F_{ij} y τ_j, se parte de la anterior ecuación construyendo un funcional el cual consiste en la diferencia entre los caudales observados $\mathbf{q}^{n+1,obs}$ y los $\mathbf{q}^{n+1,est}$ obtenidos a partir de los parámetros estimados.

$$\Phi = \sum_{n=1}^{M} \left\| \mathbf{q}^{n+1,obs} - \mathbf{q}^{n+1,est} \right\|^2 = \sum_{n=1}^{M} \left\| \mathbf{q}^{n+1} - \mathbf{G}\mathbf{q}^{n} - \mathbf{B}\mathbf{I}^{n+1} \right\|^2 \quad (21)$$

donde

$$\mathbf{I} = [I_1, I_2, .., I_j], \mathbf{I} \in \Re^{N_i} \quad (22)$$

$$\mathbf{q} = [q_1, q_2, .., q_j], \ \mathbf{q} \in \Re^{N_p} \quad (23)$$

$$\mathbf{G} = [G_1, G_2, .., G_j], \ \mathbf{G} \in \Re^{N_p} \quad (24)$$

$$\mathbf{B} = [(1-G_1)F_1, (1-G_2)F_2, .., (1-G_j)F_j], \ \mathbf{B} \in \Re^{N_F} \quad (25)$$

Aquí N_F es el número de parámetros F, por lo tanto el número total de parámetros a identificar es $N_{dof} = N_p + N_F$.

Luego, si se tiene una historia de $\mathbf{I}$ y $\mathbf{q}$, es decir una serie de vectores $\{\mathbf{q}^n, \mathbf{I}^n\}$ para n=1,..,M-1, entonces es posible identificar los vectores $\mathbf{G}$ y $\mathbf{B}$ buscando el mínimo del funcional Φ, es decir, minimizando la diferencia entre los caudales reales observados $\mathbf{q}^{obs}$ y los caudales estimado a partir de los parámetros $\mathbf{q}^{est}$ (también se lo puede pensar como una regresión); para esto se define el vector de incógnitas como

$$\mathbf{X} = \begin{bmatrix} \mathbf{G} \\ \mathbf{B} \end{bmatrix} \quad (26)$$

y se re expresa el funcional cuadrático anterior en la forma

$$\Phi(\mathbf{X}) = \frac{1}{2}\mathbf{X}^{\mathbf{T}}\mathbf{H}\mathbf{X} - \mathbf{R}^{\mathbf{T}}\mathbf{X} + c \quad (27)$$

donde

$$H_{ij} = \frac{\partial^2 \Phi}{\partial X_j \partial X_k} \quad (28)$$

$$R_i = \left.\frac{\partial \Phi}{\partial X_i}\right|_{\mathbf{X}=0} \tag{29}$$

$$c = \Phi(\mathbf{X} = 0) \tag{30}$$

Nota: $\mathbf{H}$ es una matriz semidefinida positiva y en caso de que el problema esté bien planteado (es decir los parámetros sean identificables) es definida positiva [7].

Una vez definido el funcional es necesario definir las restricciones que debe satisfacer la solución encontrada, para así de ésta manera lograr que la solución satisfaga la física del problema. A continuación se define el vector de restricciones unilaterales $\mathbf{g} \in \Re^{N_r}$, con $N_r = N_p + N_i + N_{dof}$

$$\mathbf{g} = \begin{bmatrix} \mathbf{g}_{PT} \\ \mathbf{g}_{FS} \\ \mathbf{g}_{PF} \end{bmatrix} \tag{31}$$

donde

- $\mathbf{g}_{PT} \in \Re^{N_p}$ son las restricciones sobre la no negatividad de los τ, es decir $\tau \geq 0$, esto es equivalente a

$$g_{PTj} = G_j - 1 \leq 0 \text{ para } j = 1,..,N_p \tag{32}$$

- $\mathbf{g}_{FS} \in \Re^{N_i}$ son las restricciones sobre la suma de los F_{ij}

$$g_{FSi} = \sum_{j=1}^{N_p} f_{ij} - 1 \leq 0 \text{ para } i = 1,..,N_i \tag{33}$$

- $\mathbf{g}_{PF} \in \Re^{N_{dof}}$ son las restricciones sobre la no negatividad de los F_q

$$g_{PFq} = -F_q \leq 0 \text{ para } q = 1,..,N_{dof} \tag{34}$$

Finalmente, es posible resolver el problema sujeto a las restricciones unilaterales anteriormente descriptas mediante el siguiente algoritmo basado en el método de lagrangianos Aumentados (ALM) descrito en la Sec. 2.3.3.

1. Datos:

   1.a Caudales de producción e inyección, q_j^n, I_i^n.

   1.b Definición de las etapas n_s, s=1,.., N_s

   1.c Parámetros del ALM: factor de penalización μ, tolerancia ε_{ALM}, número de iteraciones n_{itALM}.

   1.d Parámetros del NR: tolerancia ε_{NR}, número de iteraciones n_{itNR}.

2. Calcular las matrices $\mathbf{H}$ y $\mathbf{R}$ en base a los caudales de acuerdo a lo descrito en la Sec. anterior.
3. Inicializar $\mathbf{X}^0$ resolviendo el problema sin restricciones $\mathbf{X}^0 = \mathbf{H}^{-1}\mathbf{R}$.
4. Inicializar los multiplicadores de Lagrange a 0: $\lambda^0 = 0$.
5. $for\ \ j = 1,..,n_{itALM}$

   5.a Inicializar el vector de iteración de NR $\mathbf{X}^{k0} = \mathbf{X}^0$

   5.b $for\ \ r = 1,..,n_{itNR}$

      5.b.1 Formar el gradiente del funcional de ALM

$$\nabla\Phi^k = \mathbf{H}\mathbf{X}^{kr} - \mathbf{R} + \sum_{l=1}^{N_r}(\lambda_l^k + \mu g_l(\mathbf{X}^{kr})^+)\nabla g_l(\mathbf{X}^{kr})$$

      5.b.2 $if\ (\left\|\nabla\Phi^k\right\| < \varepsilon_{NR})\ \ break$

      5.b.3 Formar el hessiano del funcional de ALM

$$\mathbf{H}_{ALM}^k = \nabla\nabla\Phi^k \approx \mathbf{H}\mathbf{X}^{kr} + \mu\sum_{l=1}^{N_r}\nabla g_l(\mathbf{X}^{kr})g_l(\mathbf{X}^{kr})^+(\nabla g_l(\mathbf{X}^{kr}))^T$$

      5.b.4 Resolver el sistema

$$X^{k,r+1} = X^{kr} - (\mathbf{H}_{ALM}^k)^{-1}\nabla\Phi^k$$

   5.c Actualizar los multiplicadores de Langrange

$$\lambda_l^{k+1} = \lambda_l^k + \mu g_l(\mathbf{X}^{kr})^+$$

   5.d $if\ (\left\|g^+ g\right\| + \left\|\lambda^- \lambda\right\| + \left\|g\lambda\right\| < \varepsilon_{ALM})\ \ break$ (condiciones KKT[7])

La optimización del funcional $\Phi(\mathbf{X})$ (búsqueda de su mínimo) con sus restricciones mediante el algoritmo descrito anteriormente es difícil de converger, especialmente cuando realizamos la optimización sobre los coeficientes temporales τ_j. Esto es así debido a que una pequeña variación en los τ_j implica un gran cambio en la respuesta del pozo luego de un tiempo "t", esto se da especialmente cuando los τ_j son grandes y una pequeña variación en estos repercute luego de un tiempo "t" grande. De hecho si los τ_j son conocidos, la optimización sobre los F_j es mucho mas fácil de converger. En consecuencia a lo mencionado anteriormente, se ha ideado una estrategia para facilitar la convergencia del proceso de optimización. Por un lado, la estrategia consiste en detectar los bloques (productor por productor) del funcional base cuyo hessiano es singular. Si se detectan bloques singulares se activan términos en el funcional base que corresponden a promover a ecuaciones las restricciones para los inyectores involucrados. Los pesos μ_f de estos términos son arbitrarios, pero preferentemente se eligen pequeños, de tal forma que sean lo menos intrusivos posibles. El nuevo funcional base $\Phi(\mathbf{X})$ queda entonces de la siguiente manera, donde $\Phi(\mathbf{X})_{orig}$ es el funcional base sin perturbar:

$$\Phi(\mathbf{X}) = \Phi(\mathbf{X})_{orig} + \mu_f\sum_{l=1}^{N_r} g_l(\mathbf{X})$$

Por otro lado, la estrategia está basada en la separación del problema en dos niveles: un lazo interno de optimización sobre τ_j y un lazo externo de optimización sobre los F_j.

Se debe recordar que el vector de incógnitas contiene tanto los τ_j (G_j) como los F's y viene dado por

$$\mathbf{X} = \begin{bmatrix} \mathbf{G} \\ \mathbf{B} \end{bmatrix} \tag{35}$$

entonces el funcional puede ser considerado como una función de estos.

$$\Phi(\mathbf{X}) = \Phi(\mathbf{G}, \mathbf{B}) \tag{36}$$

Sea $\Phi_G(\mathbf{G})$ el mínimo sobre $\mathbf{B}$ para un dado $\mathbf{G}$ constante, entonces $\Phi_G(\mathbf{G}) = \min_B \Phi(\mathbf{G}, \mathbf{B})$ con restricciones FS y PF.

Si $\mathbf{X}$ es el estado óptimo para $\Phi(\mathbf{X})$ entonces este puede ser encontrado como

1. Encontrar $\mathbf{G}^* = \arg\min \Phi_G(\mathbf{G})$ con restricciones PT.
2. Encontrar $\mathbf{B}^* = \arg\min_B \Phi(\mathbf{G}^*, \mathbf{B})$ con restricciones PT y PF.
3. $\mathbf{X} = [\mathbf{G}, \mathbf{B}]$.

Notar que cuando se realiza la optimización sobre $\mathbf{G}$ hay dos lazos anidados donde mientras se evalúa $\Phi_G(\mathbf{G})$ hay una optimización interna sobre $\mathbf{B}$.

Como se mencionó, la convergencia del algoritmo va empeorando a medida que los τ_j se hacen grandes, entonces para contrarrestar este efecto se restringen los τ_j a ser menores que un dado valor de umbral.

Adicionalmente, valores de τ_j muy grandes a menudo son síntomas de falta de datos de entrada o incluso mala calidad de estos.

Algoritmo de optimización implementado en IDENT:

1. Inicializar $\mathbf{G^0}, \mathbf{B^0}$ encontrando el mínimo del funcional $\Phi(\mathbf{G}, \mathbf{B})$ sin restricciones.
2. Hacer $\mathbf{G}^1$ sujeto a un máximo valor tal que $\mathbf{G}_j^1 = \min(\mathbf{G}_j^0, \mathbf{G}_{\max})$.
3. Parámetros: paso h, perturbación ε_G.
4. Sea $k = 1$. Optimizar Φ_G:
   - 4.a Calcular el gradiente $Z = \nabla\Phi_G(\mathbf{G}^k)$.
   - 4.b Si $\|Z\| < \varepsilon$ break.
   - 4.c Avanzar en la dirección del gradiente: $\mathbf{G}^{k+1} = \mathbf{G}^k - hZ$.
   - 4.d Limitar $\mathbf{G}_j^{k+1}$ a un máximo valor $\mathbf{G}_{\max}$: $\mathbf{G}_j^{k+1} \leftarrow \min(\mathbf{G}_j^{k+1}, \mathbf{G}_{\max})$
   - 4.e $k = k + 1$.
   - 4.f Ir a 4.a.

Notas:

- Los gradientes en 4.a son calculados de forma analítica.
- El calculo de Φ_G implica una optimización del funcional con respecto a la componente $\mathbf{B}$ y es realizado con el algoritmo de lagrangianos aumentados (ALM) descrito anteriormente en la Sec. 2.3.3.

2.4.2. Descripción de la implementación

IDENT[1] está desarrollada utilizando lenguaje de programación C++, ésta posee 34 archivos, de los cuales 8 son archivos de cabecera (hpp) y 26 son archivos fuente (cpp). En total cuenta con aproximadamente 14000 líneas de código, 54 clases, 114 funciones, 12 objetos globales y 115 macros de preprocesador.

A continuación se muestra la estructura de las clases más relevantes incluyendo aquellas en las cuales se deben hacer modificaciones para implementar el método CRMPK descrito más adelante en el Cap. 2. En éstas se pueden observar los diferentes atributos y métodos que poseen.

La clase “chrono_t”, cuya interfaz se muestra en Fig. 6, implementa un cronómetro incluyendo las funcionalidades específicas para su administración, éste es de especial utilidad para tomar tiempos a lo largo del código a la hora de hacer análisis de optimizaciones.

La clase “identificar_t” se observa en la Fig. 7, y es la clase principal ya que contiene la información más importante del algoritmo. Esta almacena entre otras cosas, la configuración espacial del reservorio, los datos de inyección/producción, los τ_j, los F_j, el hessiano del funcional $\mathbf{H}_{ALM}$ y el vector de incógnitas $\mathbf{X}$ y además posee los métodos para calcular el funcional base y las restricciones.

La clase “auglm_t” se presenta en la Fig. 8 y contiene toda la información referente al método de lagrangianos aumentados. Entre otras cosas almacena el vector de restricciones $\mathbf{g}$, los multiplicadores λ, la cantidad total de restricciones y el parámetro de penalización μ.

La clase “ident_opts_t” se muestra en la Fig. 9, y está diseñada con el fin de contener todos los parámetros del algoritmo, en ésta se almacenan y administran los parámetros para la ejecución del algoritmo, por ende es una de las primeras clases a instanciar en el código.

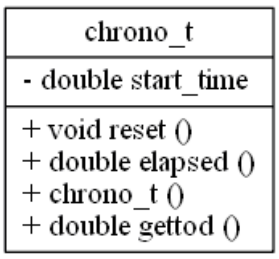

Figura 6: Clase “chrono_t”.

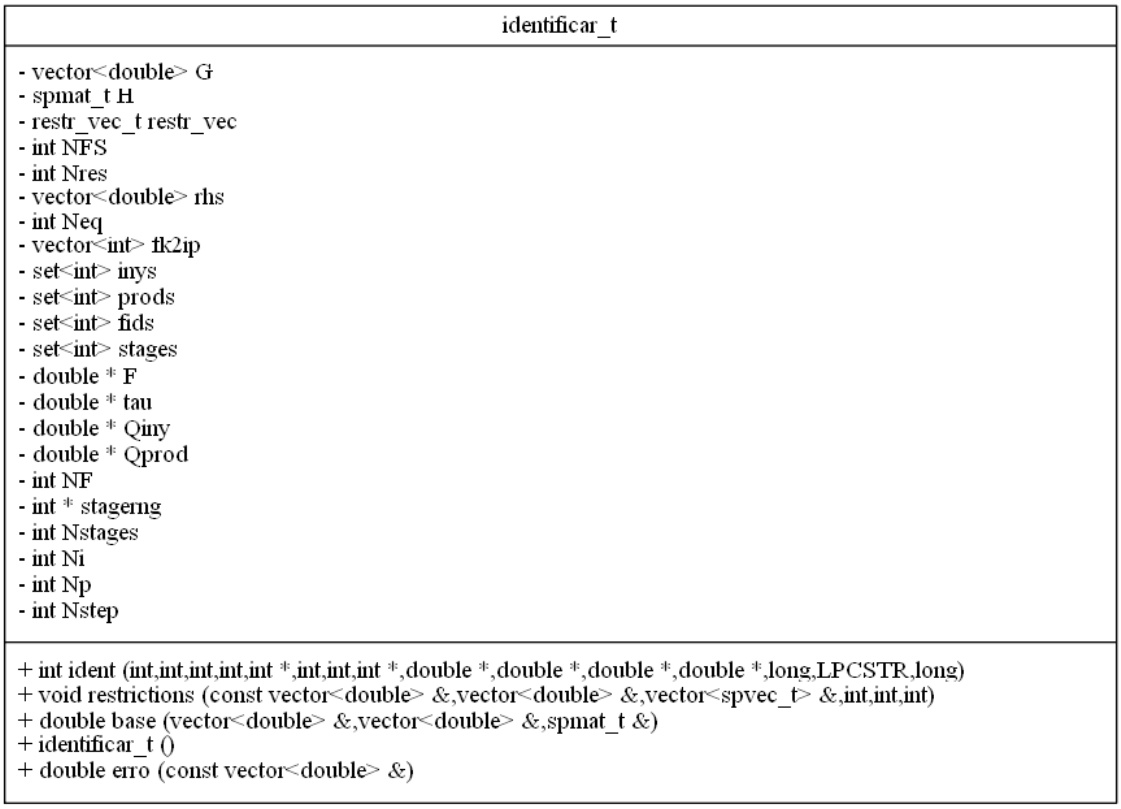

Figura 7: Clase "identificar_t".

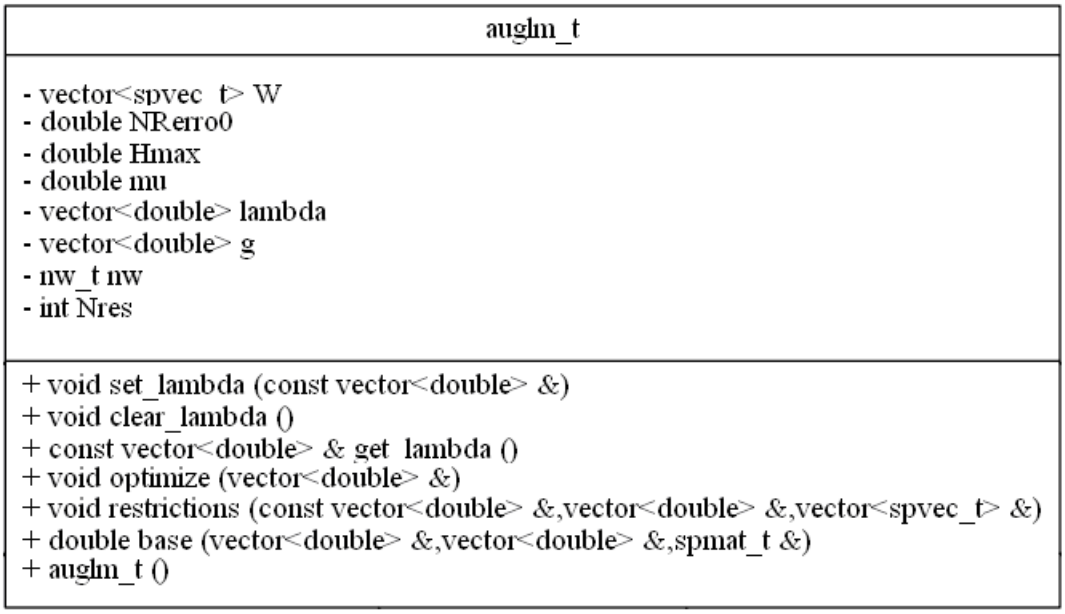

Figura 8: Clase "auglm_t".

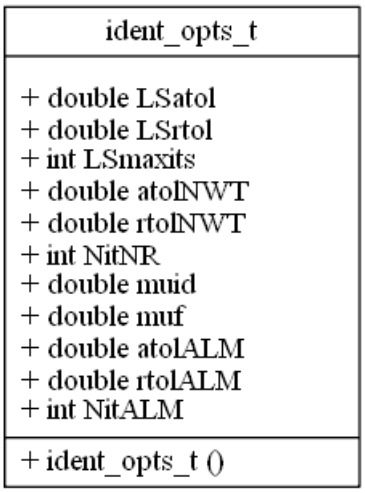

Figura 9: Clase "ident_opts_t".

3. El modelo CRMPK

3.1. Introducción

A partir del surgimiento de los CRM[8] se han propuesto distintos tipos de variantes de este para resolver el problema de identificación de parámetros en reservorios de petróleo, una de ellas es el modelo CRMP descrito en el Cap. anterior.

En este trabajo se presenta un nuevo modelo el cual será llamado CRMPK, donde la "K" final hace referencia al término "conductancia", la cual es representada con el signo Kappa (K) del alfabeto griego. El mencionado modelo propone una solución basada en el modelo CRMP[6] a la cual se le introduce un mecanismo para tener en cuenta las relaciones entre coeficientes de flujo a lo largo de las distintas etapas (diferentes configuraciones de productores-inyectores presentes en un caso).

3.2 Descripción del modelo

Este modelo puede observarse desde el punto de vista de la electrónica en el cual si se piensa que la conexión entre inyector y productor es a través de una resistencia, entonces los coeficientes de flujo F son la relación entre cada resistencia (en realidad su inversa $K = R^{-1}$, es decir la conductancia) y la suma de todas las otras.

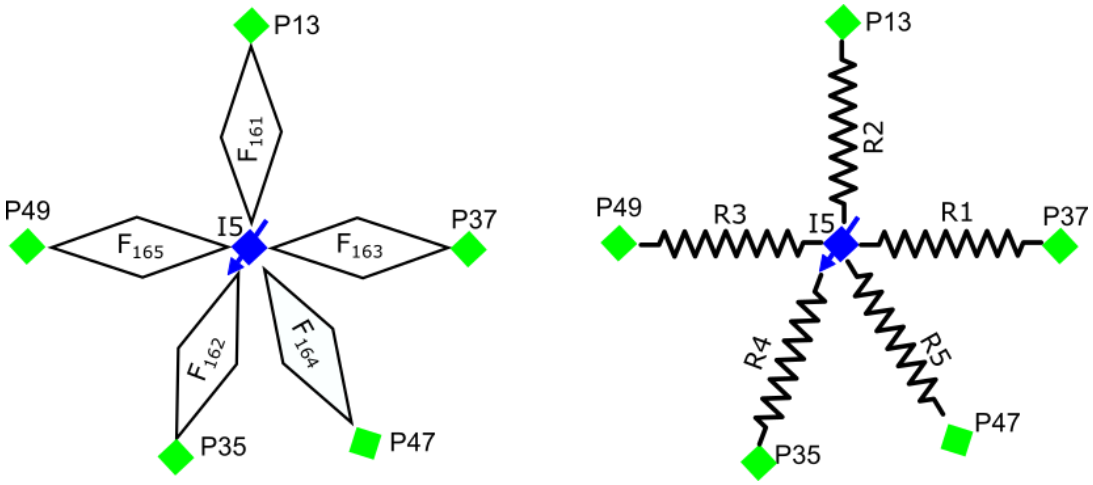

Figura 10: Configuración mostrando los coeficientes de flujo (izquierda) y las resistencias (derecha).

Si se pone atención en la configuración de la Fig. 10, la cual ha sido extraída de un caso real llamado Diana, se puede ver la representación como circuito a la derecha y como flujos a la izquierda. Entonces si se inyecta una cierta cantidad de fluido en el inyector I5, el flujo hacia el productor P49 es

$$
\begin{aligned}
Q_{I5\to P49} &= \frac{R_3^{-1}}{R_1^{-1}+R_2^{-1}+R_3^{-1}+R_4^{-1}+R_5^{-1}} I_5 \\
&= \frac{K_3}{K_1+K_2+K_3+K_4+K_5} I_5 \\
&= \frac{K_3}{SK} I_5
\end{aligned}
\tag{37}
$$

Donde

$$
SK = K_1 + K_2 + K_3 + K_4 + K_5 \tag{38}
$$

Es decir que el coeficiente F correspondiente viene dado por

$$
F_{165} = \frac{K_3}{SK} \tag{39}
$$

Ahora si se supone que existe otra etapa llamada etapa 2, con la misma configuración mostrada en la Fig. 10 con la excepción que desaparece el productor P13, entonces simplemente lo que ocurre es que desaparece la resistencia R_2 correspondiente a dicho productor. A continuación se muestra la configuración antes explicada.

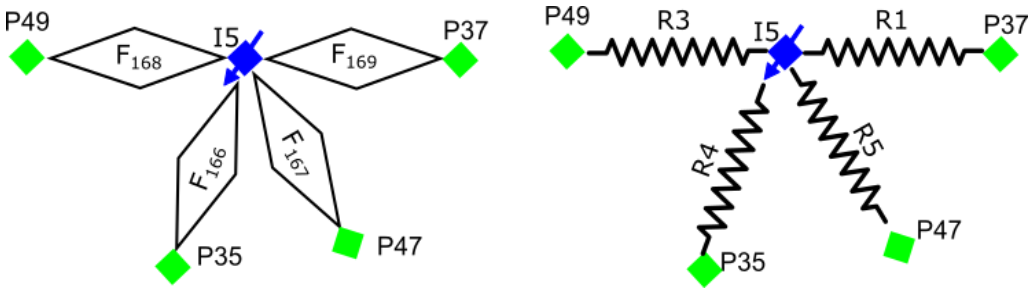

Figura 11: Configuración de la etapa 2 mostrando los coeficientes de flujo (izquierda) y las resistencias (derecha).

Entonces el coeficiente de flujo F para el productor P49 cambia a

$$
\begin{aligned}
F_{168} &= \frac{R_3^{-1}}{R_1^{-1}+R_3^{-1}+R_4^{-1}+R_5^{-1}} \\
&= \frac{K_3}{K_1+K_3+K_4+K_5} \\
&= \frac{K_3}{SK-K_2}
\end{aligned}
\tag{40}
$$

De esto último se puede observar que existe una relación de coeficientes de flujo entre ambas etapas, esto es

$$F_{165} = \frac{K_3}{SK}, F_{168} = \frac{K_3}{SK - K_2}, F_{162} = \frac{K_4}{SK}, F_{166} = \frac{K_4}{SK - K_2} \tag{41}$$

$$\frac{F_{165}}{F_{168}} = \frac{SK - K_2}{SK} \tag{42}$$

$$\frac{F_{162}}{F_{166}} = \frac{SK - K_2}{SK} \tag{43}$$

$$\frac{F_{165}}{F_{168}} = \frac{F_{162}}{F_{166}} \tag{44}$$

$$F_{165}F_{166} - F_{162}F_{168} = 0 \tag{45}$$

Por lo tanto es posible llevar a cabo la identificación de los F imponiendo ecuaciones como la anterior (45) en forma de restricciones al problema a resolver.

Este modelo, en comparación con CRMP, tiene la ventaja de que reduce la cantidad de parámetros a identificar, con lo cual es posible llevar a cabo la identificación con menor cantidad de datos haciendo provechoso su uso principalmente en aquellos casos donde se tienen pocos datos por etapa. Otra de las ventajas por sobre CRMP es que en este modelo para una dada cantidad de datos hay más datos por parámetro, debido a que como se dijo previamente hay menos parámetros a identificar, con lo cual es posible obtener resultados más precisos.

3.3. Extracción de restricciones

Para realizar la extracción de restricciones de forma automatizada es necesario desarrollar un algoritmo el cual a partir de un archivo con una configuración de productores-inyectores dada, permita hallar todas las relaciones entre coeficientes de flujo a lo largo de cualquier par de etapas. Para esto se debe analizar la estructura y formato de los datos de entrada con el fin de diseñar la estructura de datos adecuada para contenerlos. Una vez con los datos crudos en su correspondiente estructura se deben implementar estructuras de datos intermedias necesarias para proporcionar un acceso eficiente a la información de interés. Finalmente con todos los datos ya almacenados en las estructuras intermedias, es necesario analizar estos con el objetivo de extraer las relaciones pertinentes entre coeficientes de flujo.

3.3.1. Estructuras de datos

Análisis y estructura de los datos de entrada

Para realizar un análisis de los datos de entrada, en primer lugar se determinó el tipo de archivo que contiene tales datos, dando como resultado que estos vienen en formato de texto plano por lo que es posible abrirlos y visualizarlos con cualquier editor de texto disponible. Respecto a la forma en que se organizan y estructuran los datos dentro de estos archivos, se puede observar que cada línea de texto consiste de una cuadrupla que representa, en primer lugar el número de etapa, en segundo, el índice del coeficiente de flujo, luego el número de inyector y por último el número de productor. Entonces cada línea contiene toda la información referente a un dado coeficiente de flujo, esto es, la etapa a la que pertenece y además el inyector y el productor que conecta.

Para almacenar los datos de entrada se utiliza una matriz donde cada fila de ésta consiste en una fila del archivo. Ésta matriz se implementó como un vector de vectores de STL[14], es decir vector<vector<int>>. A continuación en la Fig. 12 y 13 se muestra un ejemplo simple ilustrando lo anteriormente mencionado. Dicho ejemplo será utilizado a lo largo de toda la sección, y consta de tan sólo un inyector, cuatro productores y tres etapas.

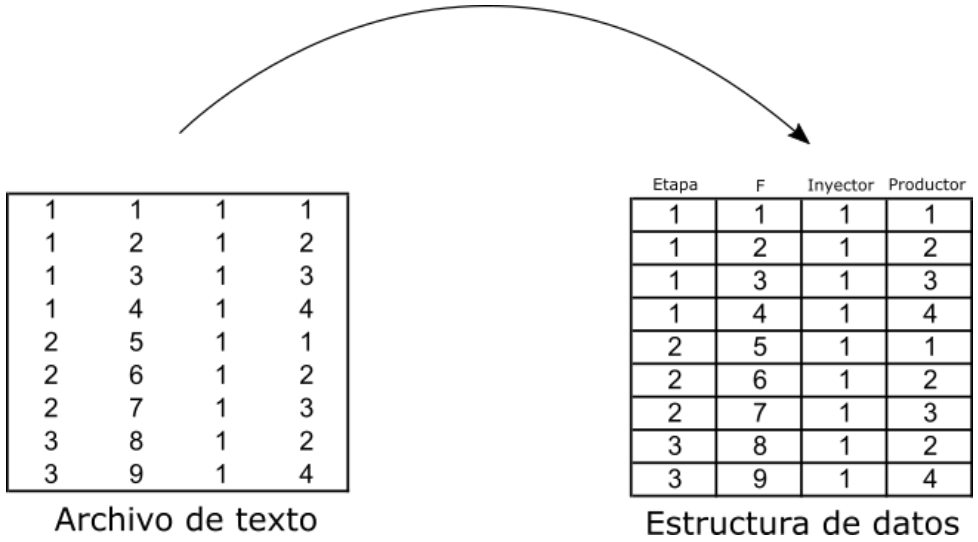

Figura 12: Archivo de texto de entrada y estructura de datos de entrada del caso de ejemplo.

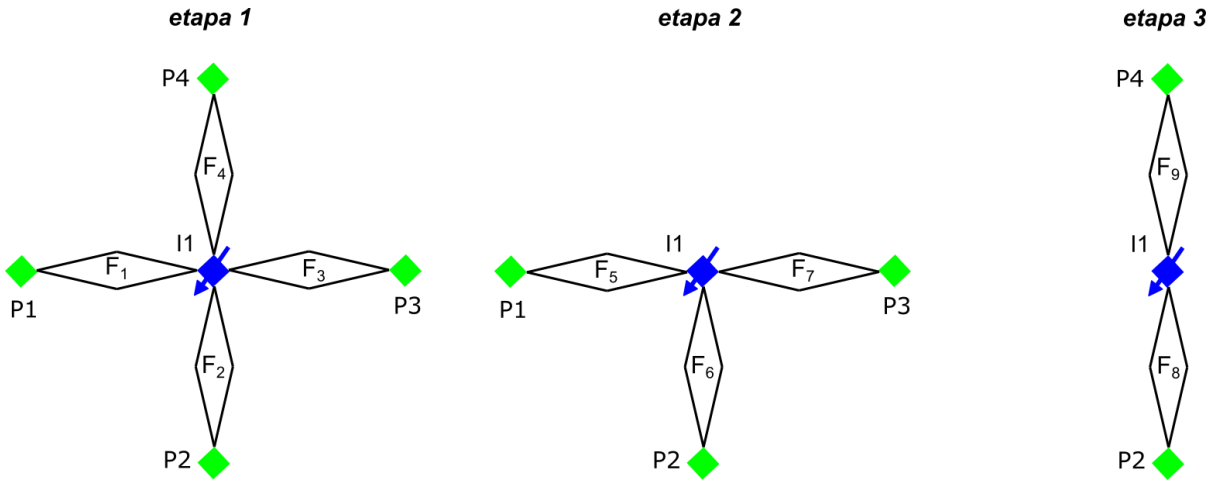

Figura 13: Configuración del caso de ejemplo.

Estructuras de datos intermedias

Para la manipulación y acceso eficiente a los datos de interés se desarrollaron e implementaron dos estructuras de datos intermedias.

En primer lugar se diseñó una estructura de datos intermedia 1 para poder obtener de forma fácil y eficiente un coeficiente de flujo dada su etapa, productor e inyector. Para esto se implementó una estructura tipo map<vector<int>,int>, es decir un mapa asociativo que vincula vectores con enteros, el cual a partir de un vector que contiene número de etapa, productor e inyector arroja el número de coeficiente de flujo asociado. Ésta estructura es conveniente en ésta situación en particular ya que el acceso a un conjunto de N datos tiene un tiempo de ejecución O(log N) permitiendo acceder de forma eficiente a estos y además de forma rápida debido a que sólo a partir de un vector se obtiene el resultado sin realizar ningún tipo de búsqueda explícita. A continuación se muestra una figura, la cual describe gráficamente el proceso antes explicado.

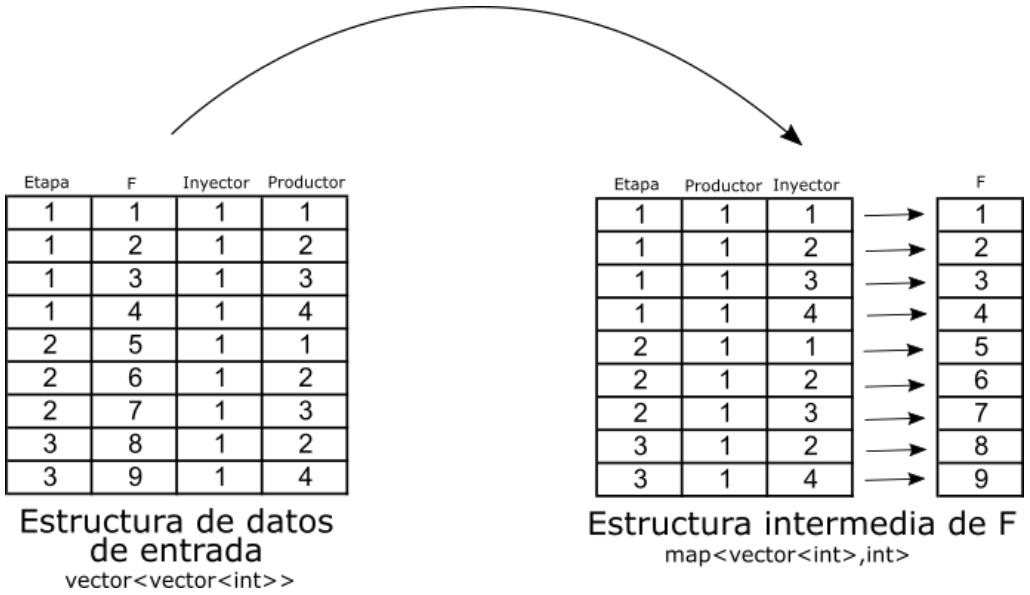

Figura 14: Estructura de datos de entrada y estructura de datos intermedia que permite el acceso eficiente a los coeficientes de flujo.

En segundo lugar se ha diseñado una estructura de datos intermedia 2 para almacenar información referida a cada inyector. Siendo de interés conocer por cada inyector, las etapas en las que está presente, y por cada una de éstas, a que productores se conecta, la estructura que se diseñó para este caso es del tipo vector<map<int,set<int>>>. En ésta estructura cada entrada del vector contiene información de un dado inyector donde por cada uno de estos se tiene un mapa asociativo el cual vincula cada etapa en la cual este inyector está presente con un conjunto de productores asociados a dicha etapa. Con este diseño se cuenta con la información referente a cada inyector totalmente separada uno de otro permitiendo hacer un análisis individual por cada uno de ellos teniendo en cuenta sólo su entorno, es decir, los productores con los que se conecta a lo largo de las etapas. A continuación en la Fig. 15 se muestra el proceso anteriormente explicado, es de notar que aquí sólo existe

un mapa asociativo dado que hay sólo un inyector, en caso que existieran más, existirían varios mapas similares al que se muestra, todos ellos contenidos en un vector, donde el n-ésimo elemento del vector correspondería al mapa del n-ésimo inyector.

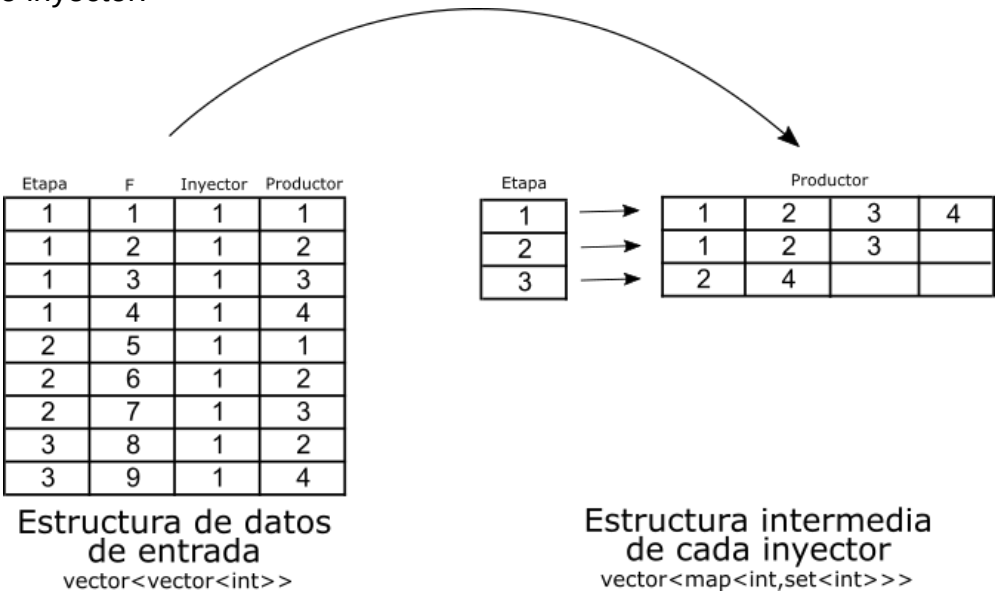

Figura 15: Estructura de datos de entrada y estructura de datos intermedia que permite condensar toda la información referente a un inyector en una estructura.

Estructura de datos de salida

Para almacenar los datos de salida del algoritmo, es decir las relaciones entre pares de coeficientes de flujo se ha diseñado una estructura tipo matriz, la cual se implementó como un vector de vectores, esto es vector<vector<int>>.

Con el objetivo de contener los pares de coeficientes de flujo de forma eficiente y ordenada, se almacenan los índices de estos en lugar de su valor, además estos se han dispuesto en cuatruplas, donde cada una de éstas ocupa una fila en la estructura de datos y contiene dos pares de índices de coeficientes de flujo pertenecientes a dos etapas distintas.

Dentro de cada cuatrupla los índices de coeficientes de flujo se organizan de la siguiente manera, el primer y tercer elemento hace referencia al par de coeficientes de flujo perteneciente a una dada etapa, el segundo y último elemento corresponde al par de coeficientes de flujo perteneciente a la etapa que se relaciona con la anteriormente mencionada. A continuación en la Fig. 16 se puede ver lo antes descripto a partir del ejemplo utilizado en ésta sección; allí se muestra parte de dicho ejemplo donde se observa que existe una relación entre F1, F5, F2 y F6, dicha relación según el modelo CRMPK es $\frac{F_1}{F_5} = \frac{F_2}{F_6}$ la cual será almacenada en la estructura de datos de salida como la cuatrupla [1 5 2 6]. Cabe destacar que si existe relación entre más de dos etapas, esta será representada por más de dos cuatruplas en la estructura de datos de salida.

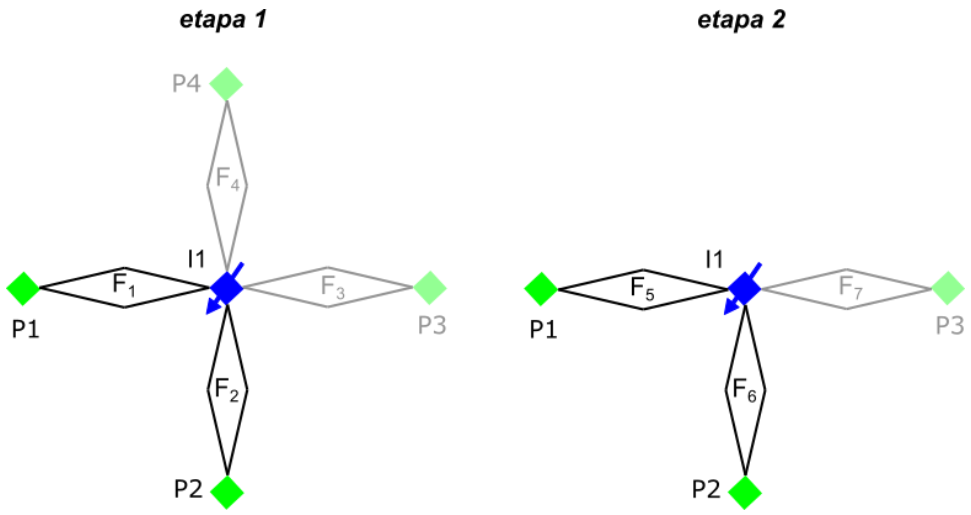

Figura 16: Parte de la configuración del ejemplo usado en ésta sección.

3.3.2 Obtención de relaciones

Para llevar a cabo la obtención de relaciones se desarrolló un algoritmo el cual consta de tres fases: Carga del archivo de entrada, Carga de datos en las estructuras intermedias y Búsqueda y almacenamiento de relaciones; a continuación se describe cada una de ellas.

La primera etapa del algoritmo consiste en abrir el archivo de texto de entrada el cual contiene toda la información del caso a analizar para luego ir recorriéndolo por filas e ir cargando cada una de éstas en la estructura de datos de entrada (vector<vector<int>>), con esto tendremos toda la información del archivo de entrada en el vector de vectores de C++. Este proceso se ilustra en la Fig. 12.

La segunda etapa radica en cargar la información en las estructuras de datos intermedias a partir de la información presente en la estructura de datos de entrada la cual fue procesada en la fase anterior.

Para cargar la estructura de datos intermedia 1 descrita en la Sec. 3.3.1, se recorre la estructura de datos de entrada fila a fila, donde para cada una de éstas se crea un vector con la etapa, inyector y productor para luego asignarlo como clave del mapa asociativo el cual asociará el mencionado vector con el número de coeficiente de flujo correspondiente a la fila en análisis tal como se puede ver en la Fig. 14.

Respecto a la estructura de datos intermedia 2 (vector<map<int,set<int>>>), el análisis de como se cargan los datos en ésta es un poco más complejo. En este caso se trabaja inyector por inyector, es decir, se recorre toda la estructura de datos de entrada filtrando la información para quedar sólo con los datos referentes a un dado inyector, luego ésta información se carga en uno de los mapas del vector de mapas asociativos de la siguiente manera: para el inyector en cuestión, por cada una de las etapas en la que está presente se inserta cada una éstas como clave del mapa asociativo, luego cada una de dichas etapas se vinculará con un conjunto de números de productores a los que se encuentra conectado el inyector en cuestión en dicha etapa. En la Fig. 15, se muestra este proceso de forma gráfica.

La tercera y última etapa corresponde a procesar y utilizar toda la información anteriormente guardada en las distintas estructuras de datos con el fin de extraer las relaciones entre coeficientes de flujo. Para extraer dichas relaciones se trabaja principalmente con la estructura de datos intermedia 2 (vector<map<int,set<int>>>). Para esto por cada inyector y su correspondiente mapa asociativo se hace lo siguiente: en primer lugar se halla la intersección entre todos los conjuntos de productores los cuales pertenecen a las distintas etapas, esto es simplemente una intersección todos contra todos. Luego por cada una de las intersecciones obtenidas se buscan los coeficientes de flujo correspondientes y de allí tendremos las relaciones entre los coeficientes de flujo entre las distintas etapas.

Con el objetivo de hacer más amena la comprensión de este proceso, se trabajará con el ejemplo que se ha venido utilizando en ésta sección.

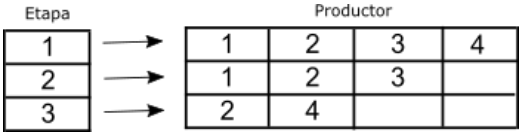

Figura 17: Estructura de datos intermedia con la información referente al inyector 1.

De las Figs. 13 y 17, se puede ver que el inyector 1 (único presente en este ejemplo) en la primera etapa se conecta con los productores 1, 2, 3 y 4, en la segunda, con 1, 2 y 3, y en la tercer etapa con los productores 2 y 4.

Para poder conocer que coeficientes de flujo se relacionan entre las distintas etapas se calcula la intersección entre todos los pares de etapas presentes en la estructura como se explicó anteriormente, en este caso sería etapa 1 ∩ etapa 2, etapa 1 ∩ etapa 3 y etapa 2 ∩ etapa 3, en la Fig. 18 se muestra esto gráficamente además de los resultados de dichas intersecciones.

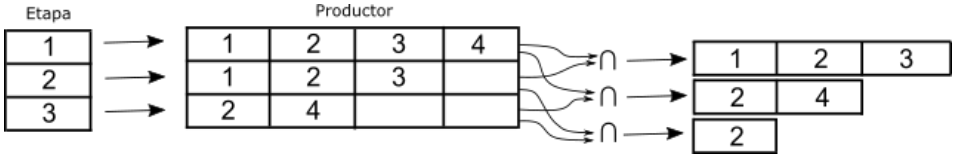

Figura 18: Estructura de datos intermedia con la información referente al inyector 1.

De la Fig. 18 mostrada arriba se puede ver que entre las etapas 1 y 2 existe relación entre 3 pares de coeficientes de flujo, los cuales involucran los productores 1,2 y 3; entre las etapas 1 y 3 existe relación entre dos pares de coeficientes de flujo los cuales involucran los productores 2 y 4, y por último entre las etapas 2 y 3 existe relación entre un sólo par de coeficientes de flujo los cuales están asociados sólo al productor 2.

Lo anteriormente dicho también se podría ver desde el siguiente punto de vista; de la primer intersección el inyector 1 está conectado a los productores 1,2 y 3 tanto en la etapa 1 como la 2 (por lo tanto existen 3 coeficientes de flujo en la etapa 1 que se relacionan con 3 coeficientes de flujo de la etapa 2), de la segunda se ve que el inyector 1 se conecta con los productores 2 y 4 en la etapa 1 como la 3, y por último, de la tercer intersección el productor 1 se conecta sólo con el productor 2 en la etapa 2 y 3.

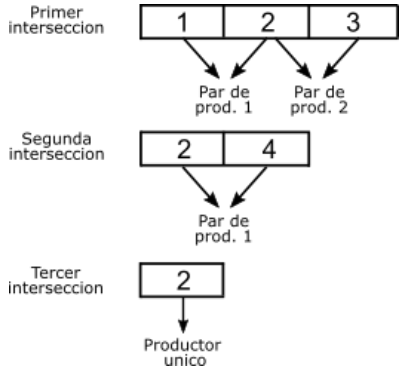

Figura 19: Pares de productores por cada una de las intersecciones.

El próximo paso es hallar los pares de coeficientes de flujo que se relacionan según el modelo CRMPK. Para esto se recorren cada una de las intersecciones de a pares consecutivos de productores, esto es, el primer elemento (productor) con el segundo, el segundo con el tercero, y así sucesivamente con el fin de hallar las relaciones entre pares de coeficientes de flujo presentes en las distintas etapas. Tal proceso se ilustra en la Fig. 19. Luego por cada uno de los pares de productores recorridos anteriormente es necesario hallar los pares de coeficientes de flujo antes mencionados, para esto a partir del número de inyector, etapa y productor podemos hacer las respectivas consultas en la estructura de datos intermedia diseñada para este fin; en este caso las consultas por cada intersección arrojan lo siguiente:

+ Primera intersección:

Pares de productores	Productor	Etapa	Inyector	Coef. de flujo (F)
Par 1	1	1	1	1
	1	2	1	5
	2	1	1	2
	2	2	1	6
Par 2	2	1	1	2
	2	2	1	6
	3	1	1	3
	3	2	1	7

Tabla 1: Resultados de la primera intersección (fila 1 y 2).

Entonces según el modelo CRMPK las relaciones para cada par de productores en este caso son:

- Relación del par 1: $\frac{F_1}{F_5} = \frac{F_2}{F_6}$
- Relación del par 2: $\frac{F_2}{F_6} = \frac{F_3}{F_7}$

+ Segunda intersección:

Pares de productores	Productor	Etapa	Inyector	Coef. de flujo (F)
Par 1	2	1	1	2
	2	3	1	8
	4	1	1	4
	4	3	1	9

Tabla 2: Resultados de la segunda intersección (fila 1 y 3).

Para ésta segunda intersección la relación viene dada por:

- Relación del par 1: $\frac{F_2}{F_8} = \frac{F_4}{F_9}$

+ Tercera intersección:

Productor	Etapa	Inyector	Coef. de flujo (F)
2	2	1	6
2	3	1	8

Tabla 3: Resultados de la tercera intersección (fila 2 y 3).

Para el caso de ésta última intersección es de notar que no existen pares de productores relacionados (sólo existe uno) entre la etapa 2 y 3, por lo tanto no es posible establecer relación alguna.

Para finalizar, las relaciones se almacenan en la estructura de datos de salida de la manera explicada en 3.3.1. Una vez con todas las relaciones presentes en dicha estructura, éstas se guardan en un archivo de texto de salida, de forma que cada fila presente en la estructura de datos ocupa una línea en el archivo de texto. A continuación en la Fig. 20 se muestra como ejemplo el resultado del caso bajo análisis.

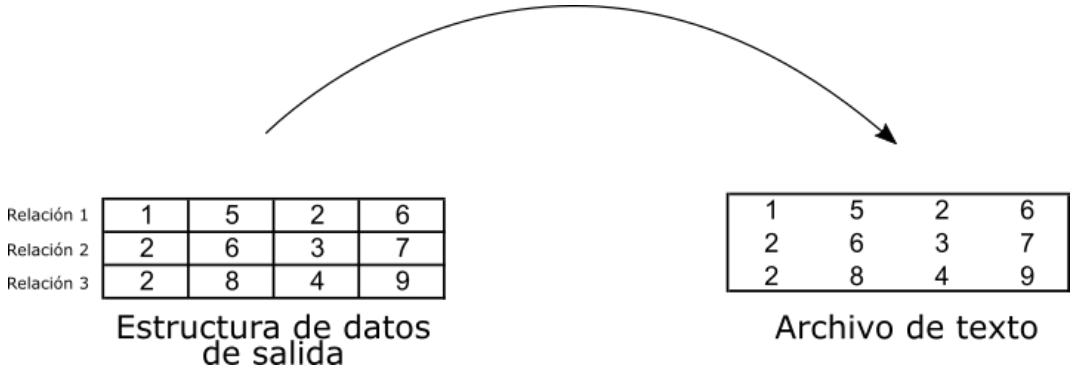

Figura 20: Estructura de datos de salida y archivo de texto de salida del caso de ejemplo.

3.3.3 Casos de prueba

Con el objetivo de validar el algoritmo desarrollado se propuso correr este con casos de diferentes dimensiones y complejidad. Para comenzar se puede observar paso a paso todo el proceso de obtención de las relaciones a través del ejemplo utilizado en la sección anterior. Para tal caso se puede ver la configuración en la Fig. 13 y el resultado final en la Fig. 20.

Luego se pondrá a prueba el algoritmo con dos casos más, por un lado un caso generado manualmente, de complejidad media, que posee dos etapas, 5 inyectores y dos productores. Finalmente, se utilizará un caso de alta complejidad de la vida real llamado caso Diana, el cual posee 77 etapas, 24 inyectores y 80 productores.

Para comenzar se parte del caso de complejidad media que se muestra debajo.

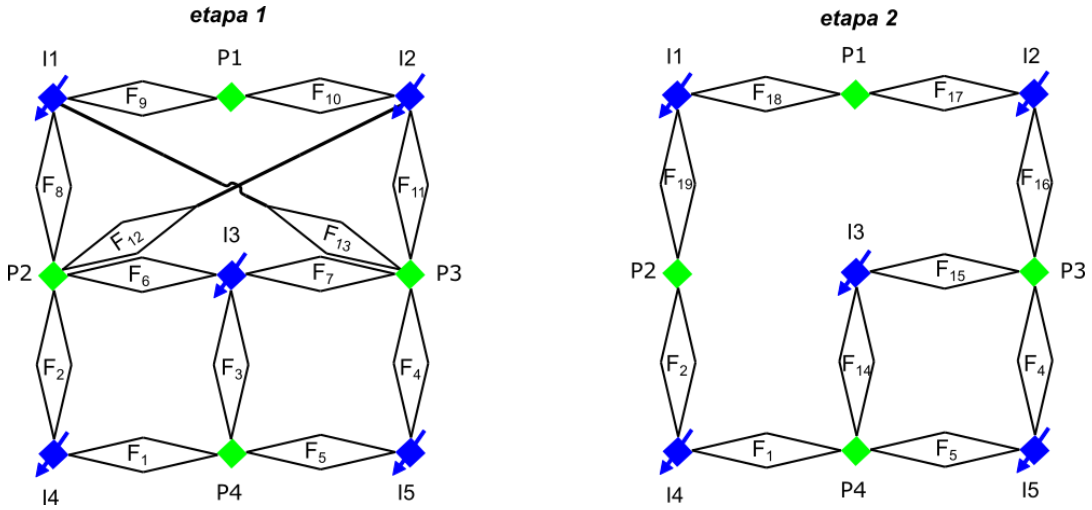

Figura 21: Configuración del caso de prueba de complejidad media.

El archivo de entrada para este caso es:

1 1 4 4
1 2 4 2
1 3 3 4
1 4 5 3
1 5 5 4
1 6 3 2
1 7 3 3
1 8 1 2
1 9 1 1
1 10 2 1
1 11 2 3
1 12 2 2
1 13 1 3
2 1 4 4
2 2 4 2
2 4 5 3
2 5 5 4

2 14 3 4
2 15 3 3
2 16 2 3
2 17 2 1
2 18 1 1
2 19 1 2

luego haciendo un análisis del caso se puede observar que según el modelo CRMPK las relaciones son:

- $\frac{F_9}{F_{18}} = \frac{F_8}{F_{19}}$
- $\frac{F_{10}}{F_{17}} = \frac{F_{11}}{F_{16}}$
- $\frac{F_7}{F_{15}} = \frac{F_3}{F_{14}}$

Una vez ejecutado el algoritmo desarrollado en la sección anterior y abriendo el archivo de salida, se puede apreciar que el resultado es:

9 18 8 19
10 17 11 16
7 15 3 14

lo cual refleja las relaciones deducidas anteriormente confirmando el correcto funcionamiento del algoritmo.

Para el último caso no se ha incluido figura debido a su gran tamaño (24 etapas) ni tampoco se incluye el archivo de entrada ya que posee 5989 filas. En cambio para llevar a cabo la verificación de este caso, una vez corrido el algoritmo se analizó el archivo de salida línea a línea verificando cada una de las relaciones lo cual finalmente ha dado un resultado correcto.

3.4 Implementación de CRMPK en IDENT

Para implementar el método CRMPK en la librería IDENT[1] se comenzó realizando un análisis integral y comprensión del mismo con el objetivo de familiarizarse con sus métodos, clases y librerías. De este análisis se desprende como resultado que se ha usado la herramienta PETSc-FEM[15] (también desarrollada en el CIMEC) y además que el código implementa las principales clases descriptas en la Sec. 2.4.2.

El método CRMPK se implementa imponiendo restricciones al funcional base utilizando el método de penalización descrito en la Sec. 2.3.1, esto permite la posibilidad de escoger el coeficiente de penalización μ_{CRMPK} de manera tal que, haciendo tender este coeficiente a infinito, penalizamos la función más severamente forzando ésta a permanecer dentro de la región en la cual se cumplen las restricciones, o eligiendo un coeficiente pequeño dando de ésta manera flexibilidad al problema a adaptarse a la física real. Las restricciones a imponer al funcional base se aplican utilizando la función de

penalización cuadrática la cual consiste básicamente en el cuadrado de la suma de las restricciones y éstas son del tipo bilaterales, es decir de igualdad. Dichas restricciones se definen en el vector $\mathbf{h} \in \Re^{N_{FRES}}$ donde N_{FRES} es el número de restricciones CRMPK.

Con todo esto el nuevo funcional base, su gradiente y su hessiano vienen dados por

$$\Phi_{CRMPK}(\mathbf{X}) = \Phi(\mathbf{X}) + \frac{1}{2}\mu_{CRMPK}\sum_{j=1}^{N_{FRES}}(h_j(\mathbf{X}))^2 \tag{46}$$

$$\nabla\Phi_{CRMPK}(\mathbf{X}) = \nabla\Phi(\mathbf{X}) + \mu_{CRMPK}\sum_{j=1}^{N_{FRES}}\nabla h_j(\mathbf{X})h_j(\mathbf{X}) \tag{47}$$

$$\nabla^2\Phi_{CRMPK}(\mathbf{X}) = \nabla^2\Phi(\mathbf{X}) + \mu_{CRMPK}\sum_{j=1}^{N_{FRES}}\nabla h_j(\mathbf{X})\nabla h_j(\mathbf{X})^T + h_j(\mathbf{X})\nabla^2 h_j(\mathbf{X}) \tag{48}$$

donde $\Phi(\mathbf{X})$ es el funcional del método CRMP implementado en IDENT.

Para incluir lo antes descrito en IDENT, ésta se ha modificado en ciertas partes sobre todo haciendo agregado de porciones de código. Para comenzar se definió un nuevo atributo llamado “mucrmpk” de tipo double (doble) en la clase “ident_opts_t” con el fin de almacenar el coeficiente de penalización μ_{CRMPK}.

Seguidamente, se crearon tres atributos en la clase “identificar_t”. Uno de estos almacena la estructura de datos de salida definida en 3.3.1 y se le asignó el nombre “Frelations”; como su nombre lo indica, en este se almacenan las relaciones entre los coeficientes de flujo que son utilizadas para construir las restricciones para CRMPK. El segundo atributo es del tipo int (entero), es llamado “Nfres” y contiene la cantidad total de restricciones CRMPK. El último de estos es llamado “deriv_relations”, es del tipo vector<map<int,int>>, en el se almacenan las derivadas de las restricciones CRMPK que son utilizadas para ensamblar el gradiente del funcional $\nabla\Phi_{CRMPK}(\mathbf{X})$, en esta ultima estructura la n-ésima fila del vector almacena el gradiente de la n-ésima restricción mediante un map<int,int> en el cual la clave indica respecto a que F se deriva la restricción y la imagen es el índice de F resultado de tal derivada. A continuación se muestra una figura en aras de aclarar lo antes mencionado, en tal se puede ver a la derecha el primer mapa asociativo de “deriv_relations” el cual corresponde al gradiente de la restricción 1.

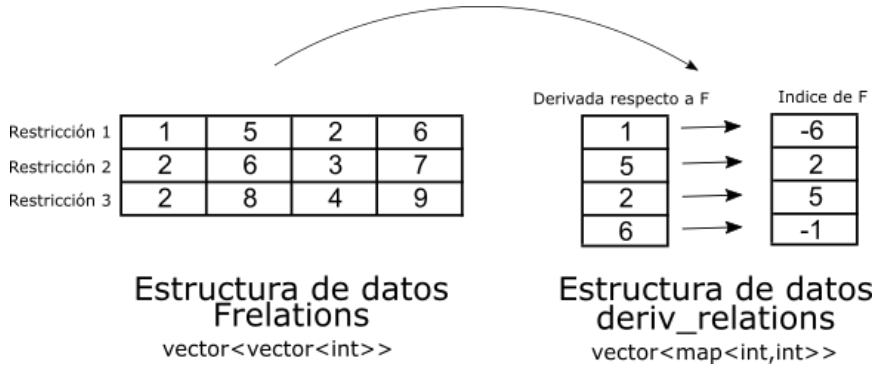

Figura 22: Caso de ejemplo de estructura de datos Frelations y deriv_relations.

Hasta aquí se han definido los atributos necesarios para almacenar la información referente al método CRMPK, a continuación se detallan las funciones que se han añadido al código con el fin de cargar los datos necesarios a los mencionados atributos. En primer lugar se agregó la función "extract_relations_itf(finfo, Nfinfo)", la cual recibe la configuración espacial del yacimiento "finfo", el número de coeficientes de flujo "Nfinfo" y retorna las relaciones entre coeficientes de flujo que son almacenadas en el atributo "Frelations" mencionado previamente. Todo el proceso llevado a cabo en dicha función fue descrito en la Sec. 3.3.

Luego se añadió la función "restrictions_crmpk_derivs(Frelations, deriv_relations)" la cual es encargada de dado un conjunto de relaciones entre coeficientes de flujo ("Frelations" en este caso), encontrar el gradiente de las restricciones CRMPK y retornarlo almacenado en "deriv_relations". Para llevar a cabo este procedimiento se comienza recorriendo la estructura "Frelations" y por cada restricción en ella (almacenada en una fila) se aplica la siguiente regla: la derivada respecto al primer elemento es el último con signo negativo, la derivada respecto al segundo elemento es el tercero, la derivada respecto al tercer elemento es el segundo y la derivada respecto al ultimo elemento es el primer elemento con signo negativo. Para ver esto con el ejemplo de la Fig. 22 se toma la restricción 1, con lo cual su gradiente se almacenará en el primer elemento de "deriv_relations", esta corresponde a la restricción $F_2F_5 - F_1F_6 = 0$. La derivada de ella respecto a F_1 es $-F_6$ y se inserta en el correspondiente mapa asociativo un elemento con clave 1 y valor -6, la derivada respecto a F_5 es F_2 y se inserta en el mapa asociativo un elemento con clave 5 y valor 2.

Ambas funciones se han agregado en el comienzo del código posibilitando de esta manera que se cuente con la información referente a CRMPK (relaciones y gradiente de restricciones) desde un principio.

Para agregar los términos correspondientes al funcional base (y su gradiente y hessiano) se ha modificado la función "base" perteneciente a la clase "identificar_t" la cual tiene la tarea de calcular el funcional base, su gradiente y su hessiano. En dicha función el atributo "phi" contiene el valor del funcional base, el vector "res" almacena el gradiente del funcional base y la matriz "H" contiene el hessiano del funcional base. Entonces debemos sumar al funcional "phi", al gradiente de este "res" y a su hessiano "H" la contribución debido a las restricciones CRMPK, para esto por cada restricción hacemos lo siguiente:

- Sumar a "phi" el valor de la restricción al cuadrado evaluado en "X" multiplicado por el coeficiente de penalización "mucrmpk" (línea 6) tal como se definió anteriormente.
- Sumar a "res" el término "mucrmpk" multiplicado por la restricción evaluada en "X" y por el gradiente de ésta evaluado en "X" (línea 8) tal como se definió anteriormente.
- Sumar a "H" el gradiente de la restricción evaluado en "X" por su transpuesta (línea 10 a 23) tal como se definió anteriormente.

Debajo se muestra el fragmento de código que se ha agregado a la función “base” para incluir las restricciones CRMPK tal como se describe arriba.

```
double identificar_t::base(vector<double> &x,vector<double>
&res,spmat_t &Hmodif)
...
1  for (int jj=0; jj<Nfres; jj++)
2  {
3      map<int,double> Wcol2;
4      evaluate_restr_crmpk(Frelations,X,jj,c);
5      double c;
6      phi += 0.5*mucrmpk*square2(c);
7      evaluate_derivs_crmpk(deriv_relations,X,Wcol2,Np,jj);
8      axpy(res,mucrmpk*c,Wcol2); //res=res+mucrmpk*c*Wcol2
9      map<int,double>::iterator q1 = Wcol2.begin(), q2;
10     while (q1!=Wcol2.end())
11     {
12         q2 = Wcol2.begin();
13         int j1 = q1->first;
14         double w1 = q1->second;
15         while (q2!=Wcol2.end())
16         {
17             int j2 = q2->first;
18             double w2 = q2->second;
19             H.addcoef(j1,j2,mucrmpk*w1*w2);
20             q2++;
21         }
22         q1++;
23     }
24 }
```

4. Mejora del desempeño del código

4.1. Introducción

Para llevar a cabo la optimización de la librería IDENT[1] se comenzará por tomar tiempos en distintas secciones de código para luego hallar y optimizar los cuellos de botella. Para esto se utilizan dos casos, el primero de ellos es llamado Diana-z172, el cual posee 6 inyectores, 19 productores, 53 coeficientes de flujo y 85 etapas. El segundo caso ha sido generado especialmente para este propósito mediante una función generadora de casos, fue llamado Synthetic-400 y posee 400 inyectores, 441 productores, 1600 coeficientes de flujo y una etapa. La razón principal por la cual es deseable tener casos grandes es reducir la incidencia de las inicializaciones en el tiempo total para así poder analizar la incidencia del tiempo de cálculo. Además otra razón es evaluar el desempeño en problemas de gran magnitud, los cuales podrían presentarse a futuro.

4.2. Descripción de las mejoras

La optimización del desempeño en primer lugar comenzó por tomar tiempos en diferentes partes del código, esto es secciones del algoritmo de lagrangianos aumentados[7] y algoritmo Newton-Raphson (NR)[13].

El análisis anterior dio como resultado la existencia de un alto consumo de tiempo en el algoritmo NR, por lo que se ha realizado un análisis fino de esta función, el cual concluyó en que el mayor consumo de tiempo residía en hacer cero todos los elementos de una matriz llamada “H”. La sección de código que realiza la puesta a cero de dicha matriz se muestra a continuación:

```
1   for (int j=0; j<Np; j++)
2   {
3       Gmax =dmax(Gmax,x[j]);
4       Gmin =dmin(Gmin,x[j]);
5       if (Gfixed[j])
6       {
7           res[j] = GRBcoef*(x[j]-G[j]);
8           for (int k=0; k<Neq; k++)
9               {
10                  H.setcoef(j,k,0.0);
11                  H.setcoef(k,j,0.0);
12              }
13          H.setcoef(j,j,GRBcoef);
14      }
15  }
```

La matriz que se pone a cero (“H”) almacena el hessiano del funcional y está implementada en forma sparse mediante la clase “sparse_t” que utiliza un map<pair<int,int>,double> (definido como map_t) para almacenar los coeficientes.

El problema reside en que para poner a cero dicha matriz se recorren todos los elementos del bloque [0,Np)x[0,Neq) (línea 10) y su transpuesto es

decir [0,Neq)x[0,Np) (línea 11), lo cual es muy ineficiente ya que internamente por cada uno de los elementos se realiza una búsqueda en un árbol binario (en el map<...> mencionado arriba) y en caso que exista tal elemento se elimina.

Para optimizar este proceso de puesta a cero se ha definido una función "make_zero_block(int i1, int i2, int j1, int j2)" perteneciente a la clase "spmat_t" la cual hace cero un bloque comprendido por las filas i1 a i2 y las columnas j1 a j2, es decir el bloque [0,Neq)x[0,Np). Entonces, para poner a cero dicho bloque de una manera eficiente sin realizar una búsqueda en un árbol binario se itera directamente sobre los elementos de la matriz (representados en el código como "coefs") verificando que estos estén en el rango dado por i1,i2,j1 y j2; en caso que estén dentro de dicho rango se borra el elemento, si no se avanza el iterador y se chequea el siguiente elemento. Es de destacar que debido a que se recorre el map<...> con iteradores, sólo borramos los elementos distintos de cero, sin necesidad de recorrer toda la matriz como se hacía en el método anterior.
A continuación se muestra la función descrita.

```
1  void spmat_t::make_zero_block(int i1,int i2,int j1,int j2){
2   map<ij_t,double>::iterator q = coefs.begin();
3   while (q!=coefs.end())
4   {
5       ij_t el=q->first;
6       if ((el.first>=i1 and el.first<i2) and (el.second>=j1 and
el.second<j2))  coefs.erase(q++);
7       else q++;
8   }
9  }
```

Para utilizar la función make_zero_block() se deben hacer dos llamadas a esta ya que como se mencionó hay que poner a cero el bloque [0,Neq)x[0,Np) y su transpuesto (bloque [0,Np)x[0,Neq)). Entonces implementando estas llamadas y suprimiendo el método original de puesta a cero, el nuevo algoritmo optimizado queda de la siguiente manera.

```
1  H.make_zero_block_it(0,Np,0,Neq);
2  H.make_zero_block_it(0,Neq,0,Np);
3  for (int j=0; j<Np; j++)
4  {
5   Gmax =dmax(Gmax,x[j]);
6   Gmin =dmin(Gmin,x[j]);
7   if (Gfixed[j])
8       {
9        res[j] = GRBcoef*(x[j]-G[j]);
10       H.setcoef(j,j,GRBcoef);
11      }
12 }
```

Para finalizar, se han tomado tiempos en ambos casos nuevamente. Para el caso Diana, completar con ceros la mencionada matriz tomaba 5.3s sobre un total de 10.8s que tomaba el algoritmo completo; mientras que con la nueva implementación toma 0.4s y 5.5s respectivamente, logrando así una reducción del tiempo a la mitad. Respecto al caso synthetic-400, hacer cero la matriz tomaba 6263.2s de un total de 6827.4s que tomaba el algoritmo completo, con la nueva implementación completar con ceros la matriz toma

22.9s y 594.9s el algoritmo completo consiguiendo así una notable mejora en rendimiento de 11X, es decir unas 11 veces mas rápido. Con esto podemos concluir que a medida que el problema se vuelve grande también aumenta el rendimiento obtenido con esta optimización.

A continuación se presenta una tabla mostrando un resumen de los tiempos (expresados en segundos).

Caso	Tiempo puesta a cero original	Tiempo total original	Tiempo puesta a cero optimizado	Tiempo total optimizado
Diana	5.3	10.8	0.4	5.5
Synthetic-400	6263.2	6827.4	22.9	594.9

Tabla 4: Resultados comparación de tiempos de ejecución.

5. Pruebas con el método CRMPK

5.1. Introducción

Para llevar adelante las pruebas del método CRMPK desarrollado en este trabajo se harán pruebas tanto en el algoritmo desarrollado en Octave[17] como en la librería IDENT[1].

En primer lugar se evaluará el método en Octave[17] a partir de dos casos. El primero de ellos es llamado caso mínimo ya que es muy pequeño y está compuesto sólo de 2 inyectores y 3 productores con un total de 10 coeficientes de flujo. Dicho caso fue creado manualmente incluyendo el menor número posible de inyectores y productores posibilitando así realizar pruebas sin adentrarse en configuraciones de productores-inyectores las cuales podrían traer problemas adicionales. La configuración espacial de inyectores y productores exhibiendo las conectividades hidráulicas y coeficientes de flujo para este caso, se muestran debajo en las Figs. 23 y 24 respectivamente.

El segundo caso posee 5 inyectores y 4 productores con un total de 30 coeficientes de flujo y está basado en un caso real llamado Synfield 5x4 desarrollado por Albertoni(2003) a partir del reservorio Chihuido de la Sierra Negra ubicado en la provincia de Neuquén. Con este caso de prueba se pretende probar el método sometiéndolo a un caso de la vida real con el fin de evaluar el desempeño en configuraciones más complejas.

El objetivo de las pruebas en Octave es realizar una evaluación bajo diferentes situaciones como pueden ser diferente cantidad de observaciones en cada etapa y distintos parámetros algorítmicos. Para llevar a cabo esto último se utilizarán los casos de prueba antes descritos aplicando a estos tanto el método CRMP[6] como CRMPK para determinar los parámetros algorítmicos óptimos y finalmente realizar comparaciones entre ambos métodos mencionados.

En segundo lugar y a partir de los parámetros determinados anteriormente, se realizarán pruebas a la librería IDENT tanto con el método CRMP como con CRMPK utilizando casos de gran magnitud proveídos por la empresa Interfaces S.A., con el objetivo de realizar comparaciones de desempeño entre ambos métodos.

5.2. Pruebas en Octave

5.2.1. Caso mínimo

En este primer caso de prueba se considera la configuración que se muestra en la Fig. 24, la cual consta de 2 inyectores y 3 productores. Hay dos etapas, en la primera el productor P3 no está activo. Entonces hay un total de 10 coeficientes independientes (F), esto es, 4 en la primera etapa y 6 en la segunda.

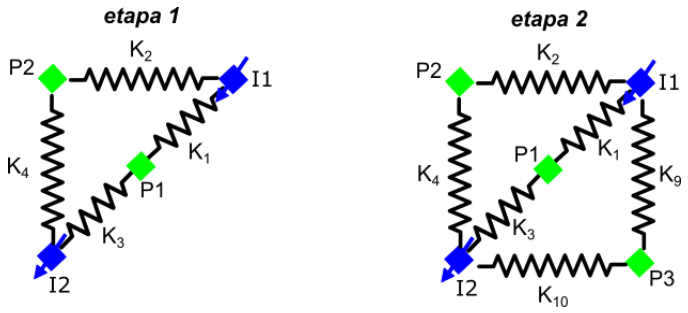

Figura 23: Configuración del caso mínimo mostrando las conectividades hidráulicas K.

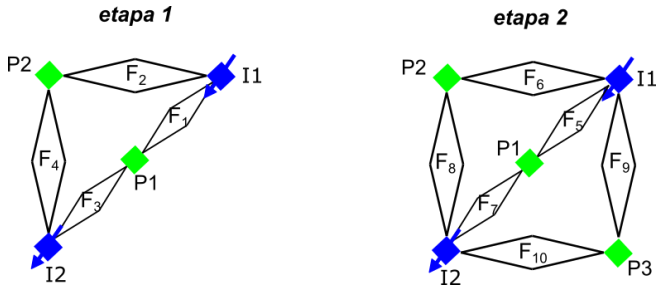

Figura 24: Configuración del caso mínimo mostrando los coeficientes de flujo F.

Las ecuaciones son (se asume un esquema atemporal, es decir $\tau_j = 0$)

$$\begin{aligned} q_1^k &= F_1 I_1^k + F_3 I_2^k \\ q_2^k &= F_2 I_1^k + F_4 I_2^k \end{aligned} \qquad k = 1 \tag{49}$$

para la etapa 1 y

$$\begin{aligned} q_1^k &= F_5 I_1^k + F_7 I_2^k \\ q_2^k &= F_6 I_1^k + F_8 I_2^k \\ q_3^k &= F_9 I_1^k + F_{10} I_2^k \end{aligned} \qquad k = 2,3 \tag{50}$$

para la etapa 2, el superíndice k denota observación k.
Si hay una observación en la etapa 1 y dos en la etapa 2, se tienen 2 ecuaciones pertenecientes a la etapa 1 y 6 ecuaciones a la etapa 2. Por lo tanto el sistema está indeterminado ya que hay 10 incógnitas y sólo 8 ecuaciones.

Escribiendo a los F en función de K para hallar las relaciones CRMPK, se obtienen las siguientes ecuaciones

$$F_1 = \frac{K_1}{K_1 + K_2} \;;\; F_5 = \frac{K_1}{K_1 + K_2 + K_9} \tag{51}$$

$$F_2 = \frac{K_2}{K_1 + K_2} \;;\; F_6 = \frac{K_2}{K_1 + K_2 + K_9} \tag{52}$$

$$F_3 = \frac{K_3}{K_3 + K_4} \quad ; \quad F_7 = \frac{K_3}{K_3 + K_4 + K_{10}} \tag{53}$$

$$F_4 = \frac{K_4}{K_3 + K_4} \quad ; \quad F_8 = \frac{K_4}{K_3 + K_4 + K_{10}} \tag{54}$$

de las cuales es posible deducir las siguientes relaciones

$$\frac{F_1}{F_5} = \frac{K_1 + K_2 + K_9}{K_1 + K_2} \tag{55}$$

$$\frac{F_2}{F_6} = \frac{K_1 + K_2 + K_9}{K_1 + K_2} \tag{56}$$

$$\frac{F_3}{F_7} = \frac{K_3 + K_4 + K_{10}}{K_3 + K_4} \tag{57}$$

$$\frac{F_4}{F_8} = \frac{K_3 + K_4 + K_{10}}{K_3 + K_4} \tag{58}$$

De estas últimas se puede ver que los F satisfacen las siguientes relaciones entre coeficientes de flujo

$$\frac{F_1}{F_5} = \frac{F_2}{F_6} \tag{59}$$

$$\frac{F_3}{F_7} = \frac{F_4}{F_8} \tag{60}$$

A partir de estas se obtienen dos ecuaciones adicionales las cuales pueden ser escritas como restricciones bilaterales sobre los F

$$F_1 F_6 - F_2 F_5 = 0 \tag{61}$$

$$F_3 F_8 - F_4 F_7 = 0 \tag{62}$$

con lo cual hay 10 ecuaciones con 10 incógnitas y el sistema es determinado.

Se ha implementado un script en Octave[17] para este problema, utilizando los siguientes valores para los F y generando valores aleatorios de caudales de inyección I.

F = [0.4; 0.6; 0.6; 0.4; 0.2; 0.3; 0.3; 0.2; 0.5; 0.5];

5.2.2 Determinación del parámetro de penalización (μ).

El término de penalización actúa de manera tal que fuerza a la solución a estar en la zona en la cual se cumplen las restricciones impuestas al problema.

Este parámetro debe ser elegido adecuadamente ya que si μ es muy pequeño la penalización es débil y la solución de cada iteración NR[13] es rápida y bien condicionada, pero X^k converge lentamente a la solución X^*. Por el contrario, si μ es grande la solución de cada iteración NR requiere mucho esfuerzo, pero X^k tiende a X^* en pocas iteraciones. Además si μ es grande, en ciertas ocasiones el NR puede no converger ya que el sistema se vuelve demasiado mal condicionado.

Para ver este comportamiento reflejado en el caso mínimo, se ha corrido un ejemplo con distintos valores de μ. A continuación las Figs. 25, 26, y 27 muestran la convergencia para valores de μ=10, μ=100 y μ=1000 respectivamente. En estas se muestra la convergencia de los dos lazos anidados (ALM y NR) donde el eje de abscisas representa las iteraciones mientras que el de ordenadas el logaritmo en base 10 del error absoluto (e). En verde se representa la convergencia del NR y en azul la del ALM[7].

Por ejemplo, para el caso de la Fig. 25 y μ=10 los primeros seis círculos rojos corresponden a la convergencia del NR para la primera iteración del ALM. Se observa que en la primera iteración del ALM se necesitan 6 iteraciones de NR para converger, mientras que a partir de la cuarta iteración de ALM se necesitan sólo dos iteraciones de NR.

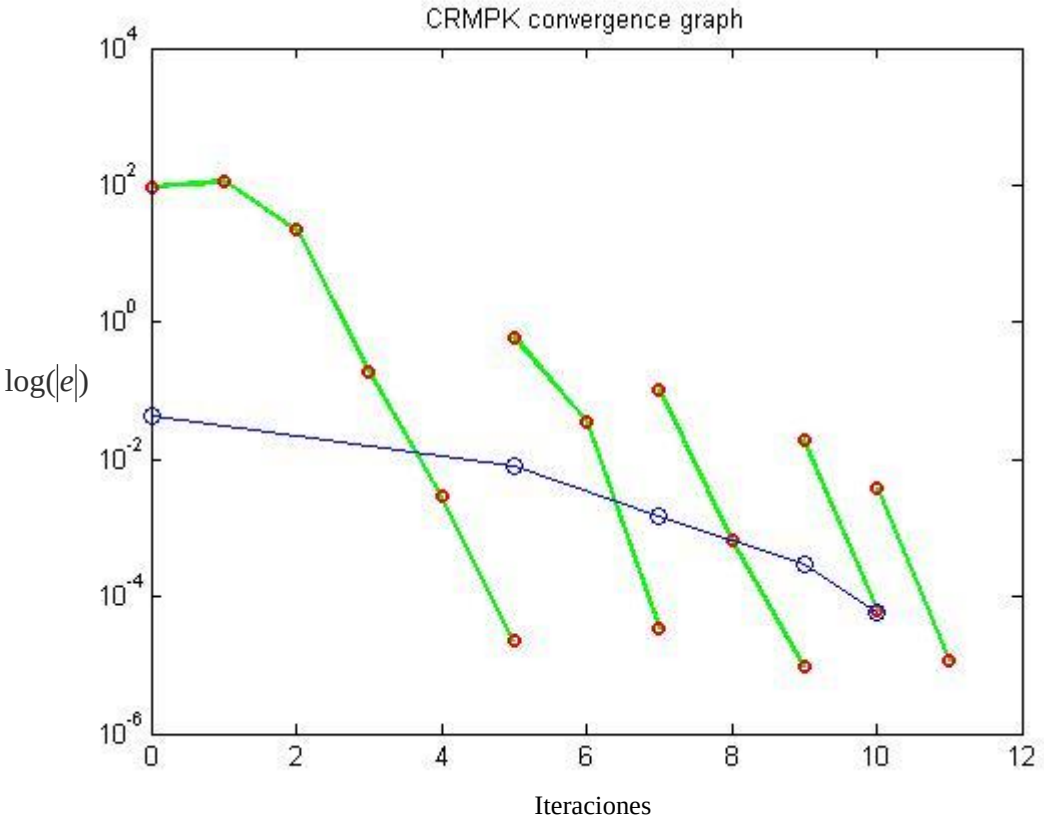

Figura 25: Convergencia del ALM y NR para =10. En verde muestra la convergencia de NR mientras que en azul la de ALM.

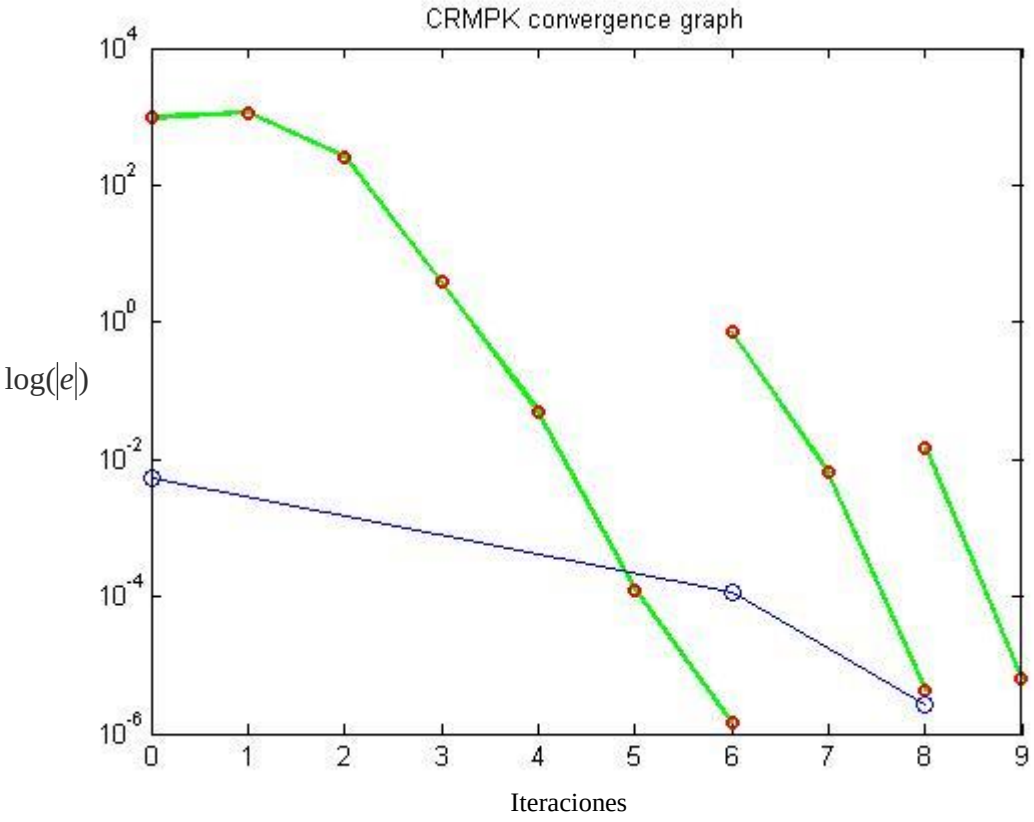

Figura 26: Convergencia del ALM y NR para =100. En verde se muestra la convergencia de NR mientras que en azul la de ALM.

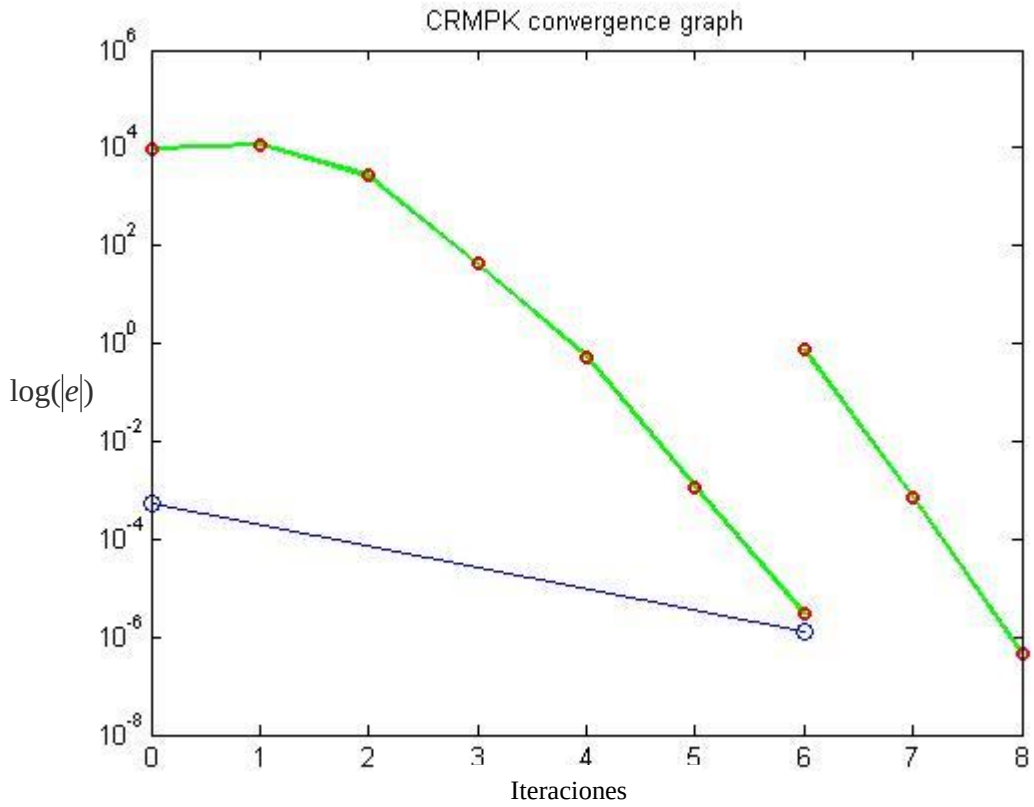

Figura 27: Convergencia del ALM y NR para =1000. En verde se muestra la convergencia de NR mientras que en azul la de ALM.

A partir de las gráficas 26, 27 y 28 se puede apreciar que en caso de usar μ=10, el ALM requirió 5 iteraciones, en caso de μ=100 requirió 3 iteraciones y finalmente para el caso de μ=1000 sólo 2 iteraciones, pero inversamente el número de iteraciones de NR se ha incrementado a medida que el número de iteraciones de ALM decrecía. Con esto podemos ver que nuestro caso cumple con la teoría de ALM y se comporta según lo esperado.

Para intentar obtener el mejor valor posible del parámetro μ para este caso, se han realizado pruebas con diferentes valores de μ. Las pruebas consistieron en generar un conjunto de 500 ejemplos a partir de caudales de inyección aleatorios e introduciendo un error en los caudales de producción del 10% contemplando dos observaciones tanto en la etapa 1 como la 2. Luego de correr la prueba se ha calculado el error relativo medio y tiempo medio incluyendo en este cálculo sólo información de los ejemplos que convergieron, además se cuentan los ejemplos que cumplen las restricciones del problema y los que no convergen, es decir los que no alcanzan la tolerancia de corte del método, la cual es 10^{-7}.
Los resultados se muestran en la siguiente tabla.

Caso	No converge	Tiempo medio (s)	Restricciones OK
μ =10	17	0.0062	484
μ =100	12	0.0035	488
μ =1000	12	0.0030	487

Tabla 5: Pruebas para diferentes valores de μ .

En ella se puede observar que el parámetro μ = 100 está cerca del óptimo ya que con este el mayor número de ejemplos cumplen las restricciones (488 ejemplos). Si se utiliza uno menor (por ejemplo para μ = 10) las restricciones no se cumplen en 484 ejemplos y si se utiliza uno mayor (por ejemplo μ = 1000) restricciones no se cumplen en 487 ejemplos. Además, respecto al tiempo medio de convergencia, es de notar que a medida que se incrementa μ el tiempo que necesita el algoritmo para resolver un problema baja. También es de notar que para los casos de μ = 100 y μ = 1000 sólo se aprecia una diferencia del 15%. Por último, teniendo en cuenta la convergencia, se observa que para μ = 100 y μ = 1000 un total de 12 ejemplos no han convergido, es decir no han alcanzado la tolerancia de corte del algoritmo, mientras que para μ = 10, un total de 17 ejemplos no convergieron.

Como conclusión al análisis anterior, se escogerá el parámetro μ = 100 debido a que con este valor la mayor cantidad de ejemplos cumplen las restricciones y la diferencia de tiempo algorítmica respecto a los demás casos no es considerable.

5.2.3 Utilización de Line Search en Newton-Raphson

El método Line Search (LS)[7] es una estrategia aplicada para ayudar a encontrar un mínimo local de una función objetivo $\Phi(X)$ de convergencia global. En la aplicación de este método, primero se halla la dirección de

descenso p^k, para luego encontrar cuan lejos (α) debemos movernos a lo largo de dicha dirección. El desafío en encontrar un buen valor de α es evitar que este sea demasiado grande o pequeño. Para esto debemos elegir un α tal que minimice $\Phi(X^k + p^k\alpha)$. Esto último lo podemos llevar a cabo de forma exacta es decir hallando $\nabla\Phi(X^k + p^k\alpha) = 0$, ó de forma inexacta buscando el mínimo con valores al azar de α tal que $\Phi(X^k + p^k\alpha) < \Phi(X^k)$.

En nuestro caso, la dirección de descenso p^k la calculamos con el método de Newton-Raphson[13], y el tamaño del paso α lo hallamos en forma exacta.

Para observar los beneficios de aplicar esta estrategia en la resolución de los sistemas de ecuaciones, se han realizado pruebas en el caso mínimo a partir del mismo conjunto de ejemplos utilizados en las pruebas anteriores para dos valores de μ con el fin de hacer una comparación contra los resultados presentados en la tabla 1. Los resultados se muestran en la siguiente tabla.

Caso	No converge	Tiempo medio (s)	Restricciones OK
μ =10	9	0.00230	496
μ =100	2	0.00232	500

Tabla 6: Pruebas para diferentes valores de μ .

De la tabla 6 se puede observar que tanto para el caso de μ = 10 como μ = 100 los resultados mejoran sustancialmente cuando se utiliza la estrategia LS. Para ambos casos probados se han obtenido mejores resultados tanto en tiempo como número de ejemplos con convergencias y cumplimiento de restricciones. Además se reafirma que la elección de μ = 100 sigue siendo la mejor, más aún en este caso, que el 100% de los ejemplos cumplen las restricciones y sólo 2 de ellos no alcanzan los requisitos de convergencia.

5.2.4 Comparación CRMP vs CRMPK

Para comparar el desempeño de CRMP[5] y CRMPK, se realizaron dos pruebas basándose en el caso mínimo y utilizando los parámetros μ y LS determinados anteriormente. Ambas pruebas consisten en generar y correr un conjunto de 1000 ejemplos con la diferencia que en la primera no se introduce error alguno mientras que en la segunda de ellas se introduce un error en los caudales de producción del 5%.
A continuación se muestra una tabla resumiendo los datos de ambas pruebas.

Prueba	Nº ejemplos	Observaciones etapa 1	Observaciones etapa 2	Error introducido
Nº 1	1000	1	2	0%
Nº 2	1000	1	2	5%

Tabla 7: Datos de prueba de comparación CRMP vs. CRMPK.

Luego de correr ambas pruebas se han confeccionado dos tablas incluyendo en ambas error relativo medio de identificación, tiempo medio de

convergencia, número de casos en que se han cumplido las restricciones y número de casos que no convergen. A continuación se muestran dichas tablas.

Método	No converge	Error relativo medio (%)	Tiempo medio (s)	Restricciones OK
CRMP	568	54.40	0.017	442
CRMPK	0	0.07	0.002	1000

Tabla 8: Resultados de prueba Nº 1.

Método	No converge	Error relativo medio (%)	Tiempo medio (s)	Restricciones OK
CRMP	652	56.1	0.009	461
CRMPK	3	18.0	0.002	1000

Tabla 9: Resultados de prueba Nº 2.

De las tablas anteriores se puede observar que tanto para la prueba Nº 1 como la Nº 2 los resultados mejoran sustancialmente cuando se utiliza CRMPK. Para ambas pruebas CRMPK logra cumplir las restricciones en la totalidad de los ejemplos además de reducir significativamente el error medio relativo de identificación y el tiempo de convergencia.

5.2.5 Caso con muchas observaciones

Si hay suficientes observaciones entonces la identificación se puede llevar a cabo tanto con CRMPK cómo con CRMP. Sin embargo, en general se obtienen mejores resultados con CRMPK ya que al haber menor cantidad de parámetros a identificar para la misma cantidad de datos entonces hay más datos por parámetro.

Para comprobar esto, a partir del caso mínimo se realizaron dos pruebas las cuales consistieron en generar un conjunto de 1000 ejemplos para cada prueba a partir de caudales de inyección aleatorios e introduciendo un error en los caudales de producción del 10% contemplando para la prueba 1, tres observaciones tanto en la etapa 1 como la 2 y para la prueba 2, diez observaciones en ambas etapas. A continuación se muestra una tabla sintetizando los datos de ambas pruebas.

Prueba	Nº ejemplos	Observaciones etapa 1	Observaciones etapa 2	Error introducido
Nº 1	1000	3	3	10%
Nº 2	1000	10	10	10%

Tabla 10: Datos de pruebas.

A partir de cada conjunto generado se ha realizado la identificación de los F tanto con CRMP como con CRMPK, tomando el error relativo $e_F = \frac{\left\| F - \overline{F} \right\|}{\left\| \overline{F} \right\|} 100$, donde $\overline{F}$ son los valores exactos usados para generar los caudales de producción.

En las Figs. 28 y 29 se muestran los resultados de las pruebas 1 y 2 respectivamente. En ellas podemos observar para cada uno de los ejemplos el error e_F de identificación tanto para CRMP (e_F^{CRMP}) como para CRMPK (e_F^{CRMPK}) representado por puntos azules.

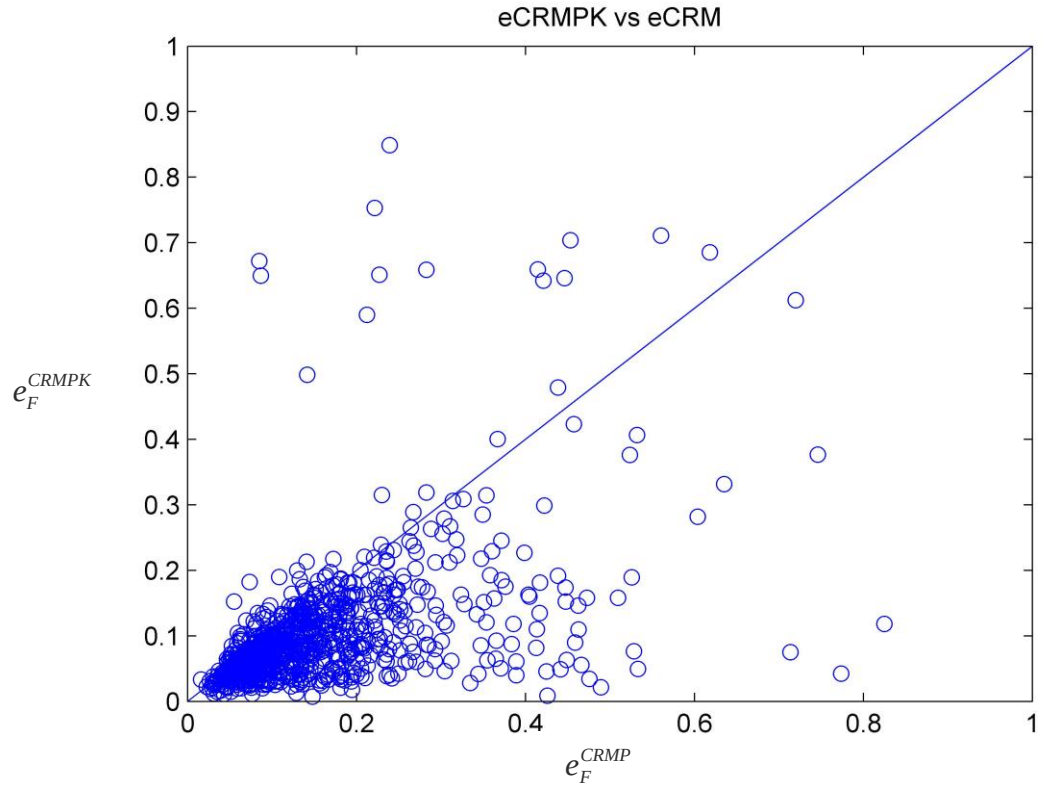

Figura 28: Errores de identificación CRMPK vs CRMP para los 1000 ejemplos correspondientes a la prueba 1.

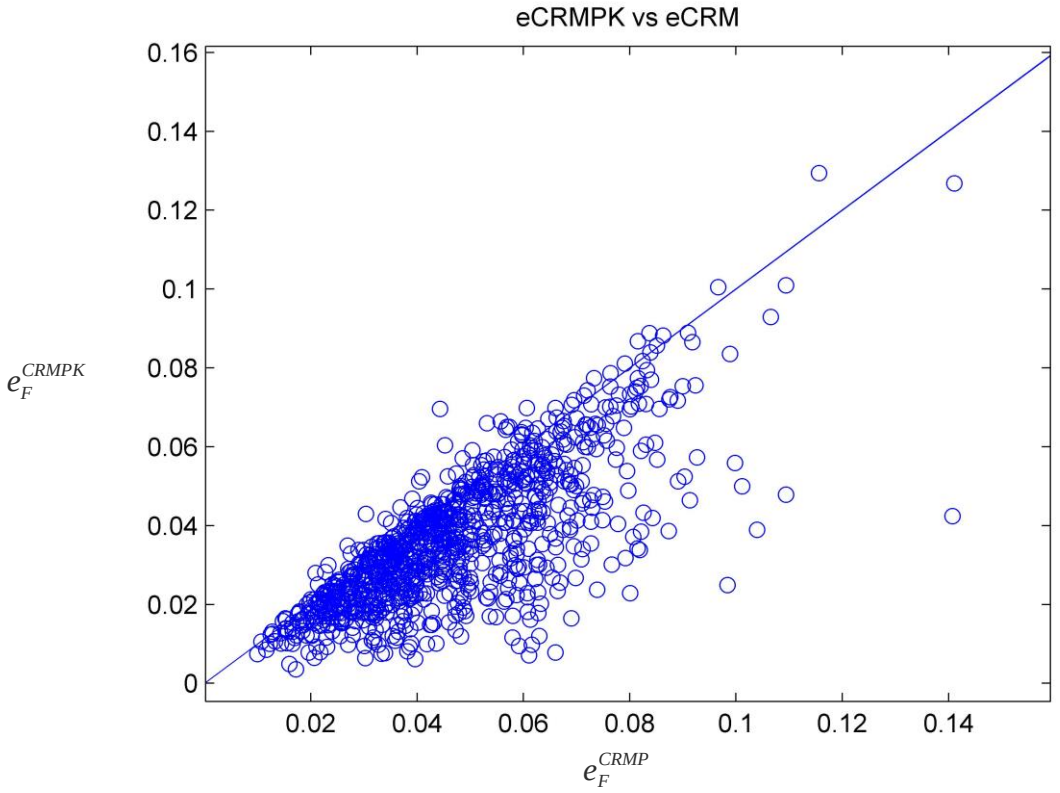

Figura 29: Errores de identificación CRMPK vs CRMP para los 1000 ejemplos correspondientes a la prueba 2.

Del resultado de la prueba 1 mostrado en la Fig. 28 se ve que en la mayoría de los ejemplos (81.4 %) el uso de CRMPK mejora la identificación. Adicionalmente se ha calculado el error relativo medio del total de los ejemplos tanto de CRMP como CRMPK, arrojando resultados de 0.14% y 0.09% respectivamente. Con esto se puede concluir que para esta prueba el método CRMPK presenta una clara y notable ventaja por sobre CRMP.

A partir de la Fig. 29 perteneciente al resultado de la prueba 2 observamos un comportamiento diferente a lo visto en la prueba 1, ya que se puede observar que la varianza en los errores no es alta, es decir todos los puntos están concentrados alrededor de 0.04, esto se debe a que la identificación se realizó con una alta cantidad de datos por parámetro dando un resultado muy preciso. El error relativo medio para CRMP en este caso fue 0.04% y para CRMPK 0.03%, además en el 85.2% de los ejemplos el uso de CRMPK mejora la identificación. De esto último podemos percibir que para este tipo de casos donde existe una gran cantidad de datos CRMPK presenta una gran mejora, si bien de todas formas los resultados con CRMP también son aceptables. Esto es debido a que ya se tiene suficiente información para realizar la identificación y por lo tanto las restricciones adicionales impuestas por CRMPK no hacen un aporte de información significativo.

5.2.6 Caso Synfield 5x4

En este caso se considera el ejemplo que se muestra en la Fig. 30, el cual posee 5 inyectores y 4 productores.

Hay dos etapas, en la primera todos los productores e inyectores se encuentran activos, en la segunda los productores P2 y P4 no están activos. Entonces tenemos 30 coeficientes independientes (F’s), 20 en la primera etapa y 10 en la segunda.

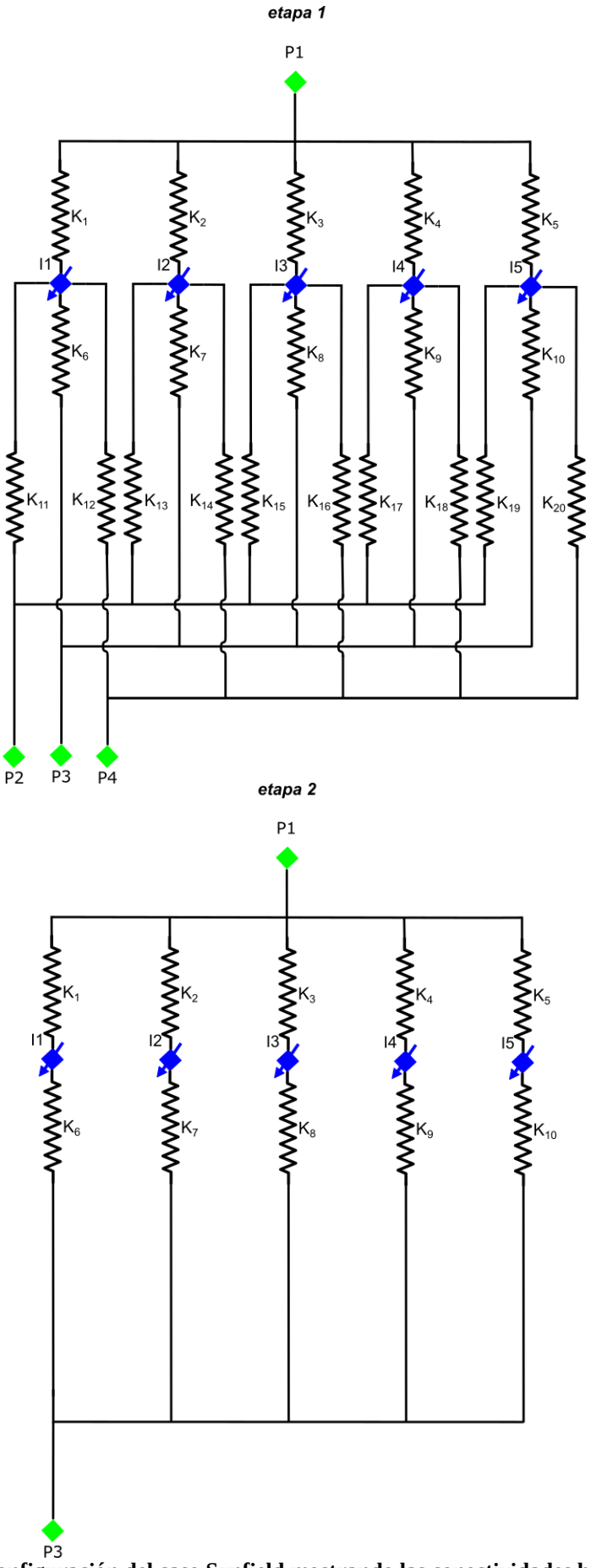

Figura 30: Configuración del caso Synfield mostrando las conectividades hidráulicas K.

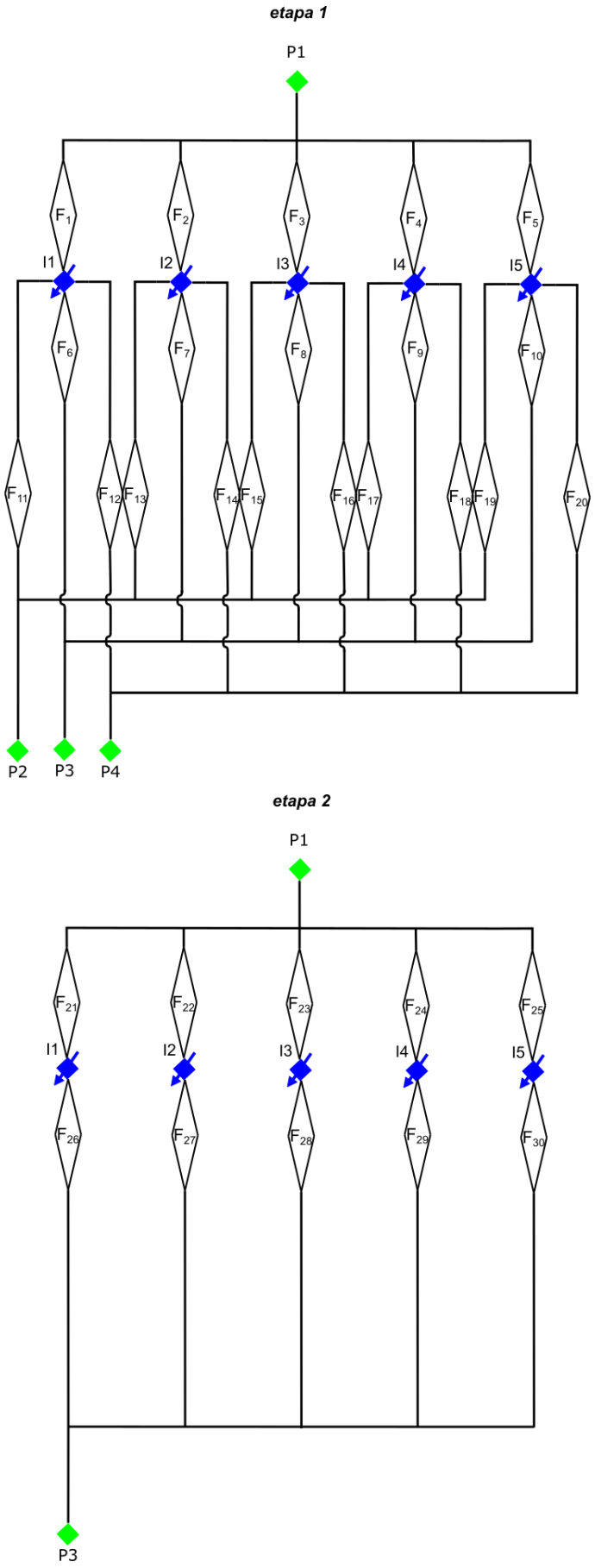

Figura 31: Configuración del caso Synfield mostrando los coeficientes de flujo F.

Asumiendo la ausencia del tiempo (τ_j = 0) las ecuaciones son

$$\begin{aligned} q_1^k &= F_1 I_1^k + F_2 I_2^k + F_3 I_3^k + F_4 I_4^k + F_5 I_5^k \\ q_2^k &= F_{11} I_1^k + F_{13} I_2^k + F_{15} I_3^k + F_{17} I_4^k + F_{19} I_5^k \\ q_3^k &= F_6 I_1^k + F_7 I_2^k + F_8 I_3^k + F_9 I_4^k + F_{10} I_5^k \\ q_4^k &= F_{12} I_1^k + F_{14} I_2^k + F_{16} I_3^k + F_{18} I_4^k + F_{20} I_5^k \end{aligned} \qquad k = [1,5] \qquad (63)$$

para la etapa 1 y

$$\begin{aligned} q_1^k &= F_{21} I_1^k + F_{22} I_2^k + F_{23} I_3^k + F_{24} I_4^k + F_{25} I_5^k \\ q_2^k &= F_{26} I_1^k + F_{27} I_2^k + F_{28} I_3^k + F_{29} I_4^k + F_{30} I_5^k \end{aligned} \qquad k = [6,8] \qquad (64)$$

para la etapa 2. Si tenemos cinco observaciones en la etapa 1 y tres observaciones en la etapa 2, tendremos 20 ecuaciones pertenecientes a la etapa 1 y 6 ecuaciones a la etapa 2. Por lo tanto el sistema está indeterminado ya que tenemos 30 incógnitas y un total de 26 ecuaciones.
Poniendo los F en función de K tenemos las siguientes ecuaciones

$$\begin{aligned} F_1 &= \frac{K_1}{K_1 + K_6 + K_{11} + K_{12}} \quad ; \quad F_{21} = \frac{K_1}{K_1 + K_6} \\ F_6 &= \frac{K_6}{K_1 + K_6 + K_{11} + K_{12}} \quad ; \quad F_{26} = \frac{K_6}{K_1 + K_6} \end{aligned} \qquad (65)$$

$$\begin{aligned} F_2 &= \frac{K_6}{K_2 + K_7 + K_{13} + K_{14}} \quad ; \quad F_{22} = \frac{K_2}{K_2 + K_7} \\ F_7 &= \frac{K_7}{K_2 + K_7 + K_{13} + K_{14}} \quad ; \quad F_{27} = \frac{K_7}{K_2 + K_7} \end{aligned} \qquad (66)$$

$$\begin{aligned} F_3 &= \frac{K_3}{K_3 + K_8 + K_{15} + K_{16}} \quad ; \quad F_{23} = \frac{K_3}{K_3 + K_8} \\ F_8 &= \frac{K_8}{K_3 + K_8 + K_{15} + K_{16}} \quad ; \quad F_{28} = \frac{K_8}{K_3 + K_8} \end{aligned} \qquad (67)$$

$$F_4 = \frac{K_4}{K_4 + K_9 + K_{17} + K_{18}} \quad ; \quad F_{24} = \frac{K_4}{K_4 + K_9} \qquad (68)$$

$$F_9 = \frac{K_9}{K_4 + K_9 + K_{17} + K_{18}} \quad ; \quad F_{29} = \frac{K_9}{K_4 + K_9}$$

$$\begin{aligned} F_5 &= \frac{K_5}{K_5 + K_{10} + K_{19} + K_{20}} \quad ; \quad F_{25} = \frac{K_5}{K_5 + K_{10}} \\ F_{10} &= \frac{K_{10}}{K_5 + K_{10} + K_{19} + K_{20}} \quad ; \quad F_{30} = \frac{K_{10}}{K_5 + K_{10}} \end{aligned} \tag{69}$$

de las cuales podemos deducir las siguientes relaciones

$$\begin{aligned} \frac{F_6}{F_{26}} &= \frac{K_1 + K_6}{K_1 + K_6 + K_{11} + K_{12}} \\ \frac{F_6}{F_{26}} &= \frac{K_1 + K_6}{K_1 + K_6 + K_{11} + K_{12}} \end{aligned} \tag{70}$$

$$\begin{aligned} \frac{F_2}{F_{22}} &= \frac{K_2 + K_7}{K_2 + K_7 + K_{13} + K_{14}} \\ \frac{F_7}{F_{27}} &= \frac{K_2 + K_7}{K_2 + K_7 + K_{13} + K_{14}} \end{aligned} \tag{71}$$

$$\begin{aligned} \frac{F_3}{F_{23}} &= \frac{K_3 + K_8}{K_3 + K_8 + K_{15} + K_{16}} \\ \frac{F_8}{F_{28}} &= \frac{K_3 + K_8}{K_3 + K_8 + K_{15} + K_{16}} \end{aligned} \tag{72}$$

$$\begin{aligned} \frac{F_4}{F_{24}} &= \frac{K_4 + K_9}{K_4 + K_9 + K_{17} + K_{18}} \\ \frac{F_9}{F_{29}} &= \frac{K_4 + K_9}{K_4 + K_9 + K_{17} + K_{18}} \end{aligned} \tag{73}$$

$$\begin{aligned} \frac{F_5}{F_{25}} &= \frac{K_5 + K_{10}}{K_5 + K_{10} + K_{19} + K_{20}} \\ \frac{F_{10}}{F_{30}} &= \frac{K_4 + K_9}{K_5 + K_{10} + K_{19} + K_{20}} \end{aligned} \tag{74}$$

De estas últimas podemos ver que los F satisfacen las siguientes relaciones entre coeficientes de flujo

$$\frac{F_1}{F_{21}} = \frac{F_6}{F_{26}} \tag{75}$$

$$\frac{F_2}{F_{22}} = \frac{F_7}{F_{27}} \tag{76}$$

$$\frac{F_3}{F_{23}} = \frac{F_8}{F_{28}} \tag{77}$$

$$\frac{F_4}{F_{24}} = \frac{F_9}{F_{29}} \tag{78}$$

$$\frac{F_5}{F_{25}} = \frac{F_{10}}{F_{30}} \tag{79}$$

las cuales se pueden reescribir como restricciones bilaterales sobre los F como

$$F_6 F_{21} - F_1 F_{26} = 0 \tag{80}$$

$$F_2 F_{27} - F_{22} F_7 = 0 \tag{81}$$

$$F_{23} F_8 - F_3 F_{28} = 0 \tag{82}$$

$$F_4 F_{29} - F_9 F_{24} = 0 \tag{83}$$

$$F_5 F_{30} - F_{10} F_{25} = 0 \tag{84}$$

con lo cual ahora se tienen 31 ecuaciones con 30 incógnitas y el sistema es determinado.

Se ha implementado un script en Octave[17] para este problema, utilizando los siguientes valores para los F's.

F = [0.3; 0.2; 0.3; 0.3; 0.1; 0.5; 0.1; 0.1; 0.1; 0.2; 0.1; 0.1; 0.2; 0.2; 0.3; 0.3; 0.3; 0.3; 0.2; 0.2; 0.3; 0.2; 0.3; 0.3; 0.1; 0.5; 0.1; 0.1; 0.1; 0.2]

Además se han usado los mismos parámetros determinados anteriormente, es decir, μ = 100, LS[7] y una tolerancia de corte de ALM[7] de 10^{-7}.

Para poner a prueba el método CRMPK en este caso, se generó un conjunto de 1000 ejemplos a partir de caudales de inyección aleatorios

introduciendo un error en los caudales de producción del 10% contemplando cinco observaciones en la etapa 1 y tres en la 2. Luego de correr la prueba se ha elaborado una tabla incluyendo el error relativo medio de identificación, tiempo medio de convergencia, número de casos en que se han cumplido las restricciones y número de casos que no convergen. A continuación se muestra dicha tabla.

Método	No converge	Error relativo medio (%)	Tiempo medio (s)	Restricciones OK
CRMP	119	67.1	0.01105	16
CRMPK	10	49.9	0.00343	408

Tabla 11: Resultados de prueba para caso Synfield 5x4.

De Tabla 11 se puede concluir que CRMPK reduce el error medio en un 18%, asimismo reduce el tiempo de ejecución del algoritmo en un orden de magnitud y logra un incremento del 39% en soluciones que cumplen las restricciones.

Además, se ha realizado un análisis de tiempo de convergencia con el objetivo de comparar el tiempo necesario tanto por CRMP como por CRMPK para resolver un problema. Para esto, a partir del conjunto de 1000 ejemplos generado anteriormente, se han tomado tiempos corriendo dichos ejemplos tanto con CRMP como con CRMPK. Luego se han graficado los resultados en un gráfico de barras donde el eje de abscisas representa el número de ejemplo, el eje de ordenadas representa el tiempo de ejecución en segundos. El tiempo de ejecución con CRMP se representa en rojo mientras que el de CRMPK en azul. En la Fig. 32 se muestran los resultados de los ejemplos que convergen en un gráfico de barras y en la Fig. 33 se muestran estos mismos pero en una gráfica logarítmica donde el eje de abscisas representa el tiempo de CRMP mientras que el de ordenadas tiempo de CRMPK.

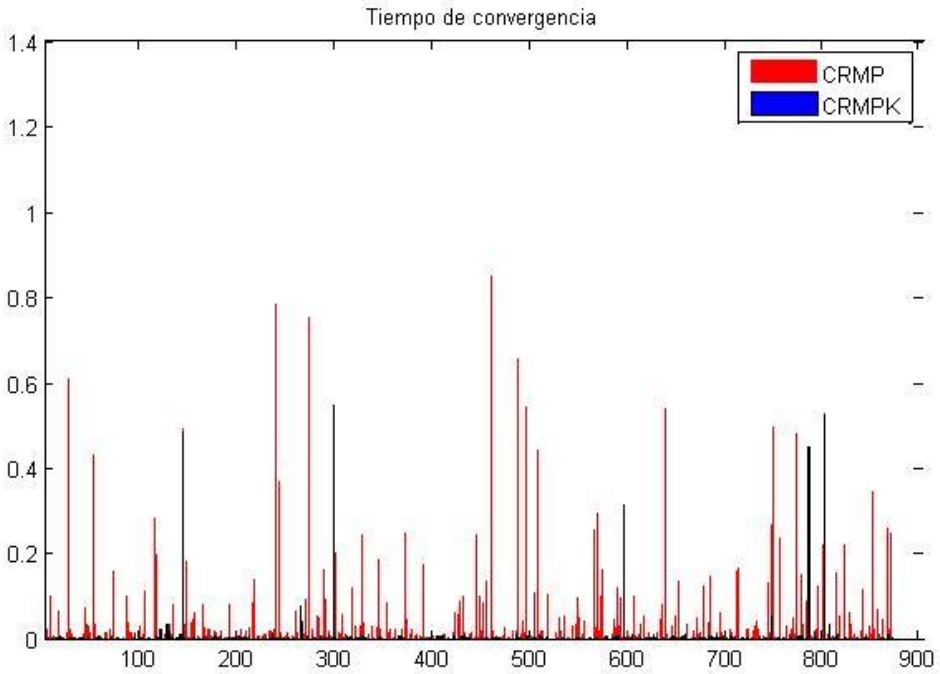

Figura 32: Tiempo de convergencia CRMPK vs CRMP.

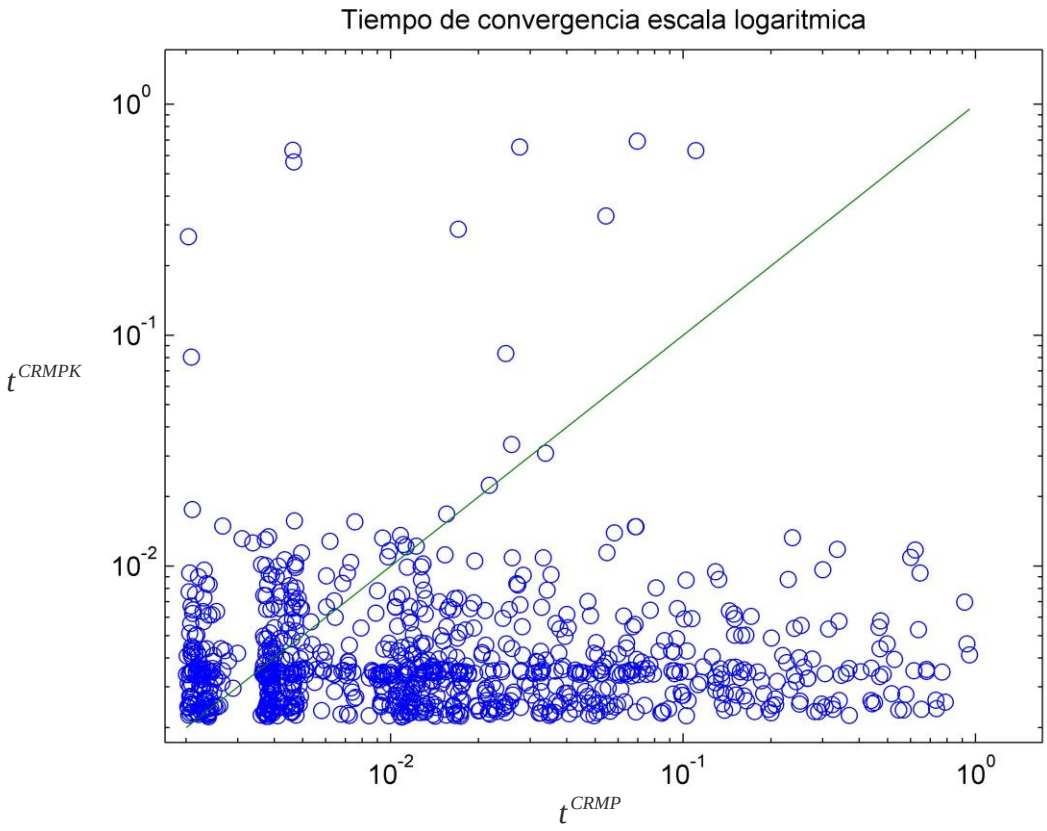

Figura 33: Tiempo de convergencia CRMPK vs CRMP.

De las gráficas anteriores se puede concluir que CRMPK reduce radicalmente el tiempo de convergencia para casi la totalidad de los casos, además se ha calculado el tiempo medio de convergencia (teniendo en cuenta todos los ejemplos) tanto con CRMP como con CRMPK, lo cual arrojó resultados de 0.0034s. y 0.0110s. respectivamente. De estos últimos se desprende que CRMPK reduce en un 69% el tiempo de convergencia en promedio.

Por otro lado, como se mencionó anteriormente, si hay suficientes observaciones entonces la identificación se puede llevar a cabo tanto con CRMPK cómo con CRMP pero, en general se obtienen mejores resultados con CRMPK debido a que hay menor cantidad de parámetros a identificar para la misma cantidad de datos y por lo tanto hay más datos por parámetro. Para verificar esto último, se generó otro conjunto de 1000 ejemplos a partir de caudales de inyección aleatorios introduciendo un error en los caudales de producción del 10% con seis observaciones en la etapa 1 y cinco en la 2.

En la siguiente figura se muestran los resultados de ésta prueba. Allí se puede observar para cada uno de los ejemplos, el error de identificación tanto para CRMP e_F^{CRMP} como para CRMPK e_F^{CRMPK} representado por un punto azul

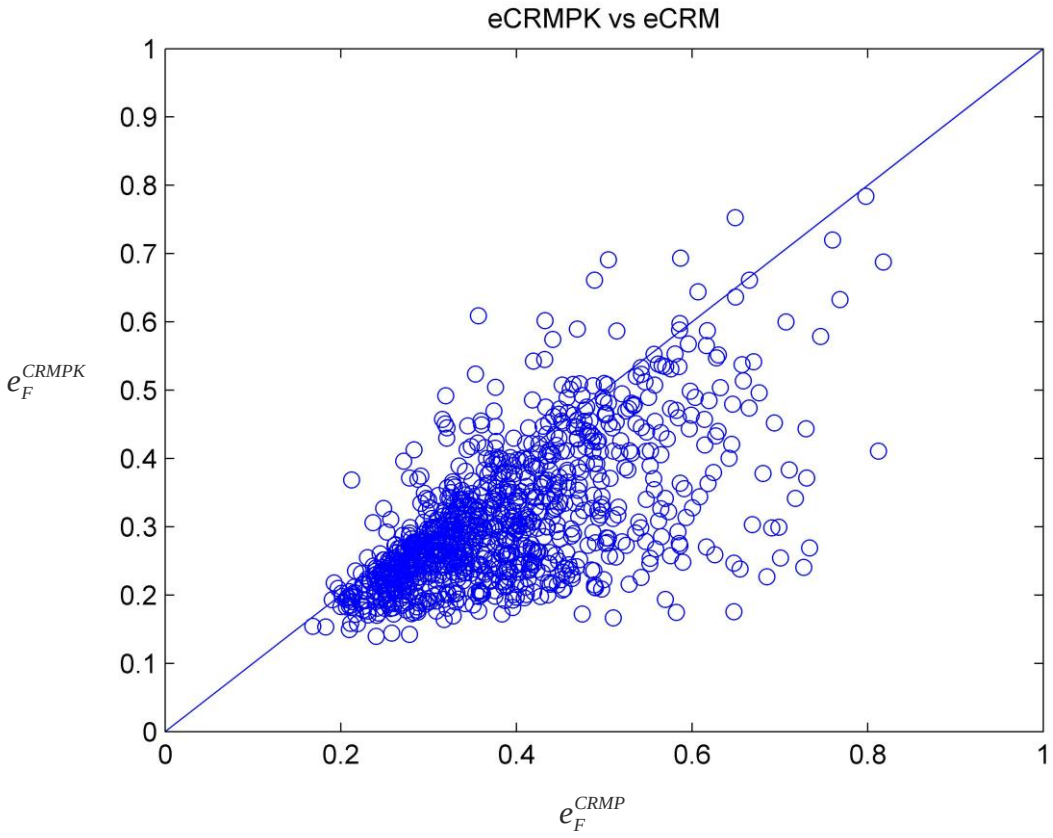

Figura 34: Errores de identificación CRMPK vs CRMP.

De ésta última figura se puede ver que en la mayoría de los ejemplos (87.4 %) el uso de CRMPK mejora la identificación. Con esto se concluye que en ésta prueba el método CRMPK también presenta una clara y notable ventaja por sobre CRMP.

5.3. Pruebas en la librería IDENT

5.3.1. Caso mínimo

Para este caso de prueba se considera el ejemplo que se muestra en las Figs. 23 y 24, el cual consta de 2 inyectores, 3 productores y 10 coeficientes de flujo. Este caso ya se ha utilizado en las pruebas realizadas con Octave[17], pero a diferencia de aquellas, aquí se utiliza un esquema temporal ya que la librería IDENT[1] tiene en cuenta el paso del tiempo y por ende calcula el tiempo característico de cada productor τ_j.

En ésta prueba se utilizan los siguientes valores para los F

$$F = [0.4; 0.6; 0.6; 0.4; 0.2; 0.3; 0.3; 0.2; 0.5; 0.5];$$

y los tiempos característicos de cada productor mostrados a continuación

$$\tau = [0.001; 0.001; 0.001].$$

Además, los valores de los caudales de inyección son generados aleatoriamente y se usan los mismos parámetros que se utilizaron en las pruebas realizadas en Octave[17], esto es, μ = 100, LS y una tolerancia de corte de ALM[7] de 10^{-7}.

Al igual que en las pruebas en Octave[17], se generó un conjunto de 1000 ejemplos a partir de caudales de inyección aleatorios pero a diferencia de aquellas no se introdujo error en los caudales de producción. Es de destacar que en este caso el número de variables a identificar se ha incrementado ya que como se dijo, la librería IDENT tiene en cuenta el tiempo característico de cada productor τ_j. Por lo tanto aquí se tienen 3 variables más a identificar (τ_1, τ_2, τ_3) con lo cual son necesarias 3 ecuaciones adicionales. Entonces, contemplando una observación en la etapa 1 y tres en la 2 se tienen 11 ecuaciones, las cuales sumadas a las dos ecuaciones debido a las relaciones CRMPK da un total de 13 ecuaciones y 13 incógnitas.

Luego de correr la prueba se ha elaborado una tabla incluyendo en ésta el error relativo medio de identificación como así también el número de iteraciones medio, y además un gráfico el cual muestra el error de identificación tanto para CRMP[6] (eje x) como para CRMPK (eje y) representado por un punto azul. A continuación se muestra dicha tabla y gráfico.

Método	Error relativo medio (%)	Iteraciones medias
CRMP	15.0	19
CRMPK	2.3	15

Tabla 12: Resultados de prueba para caso mínimo en IDENT.

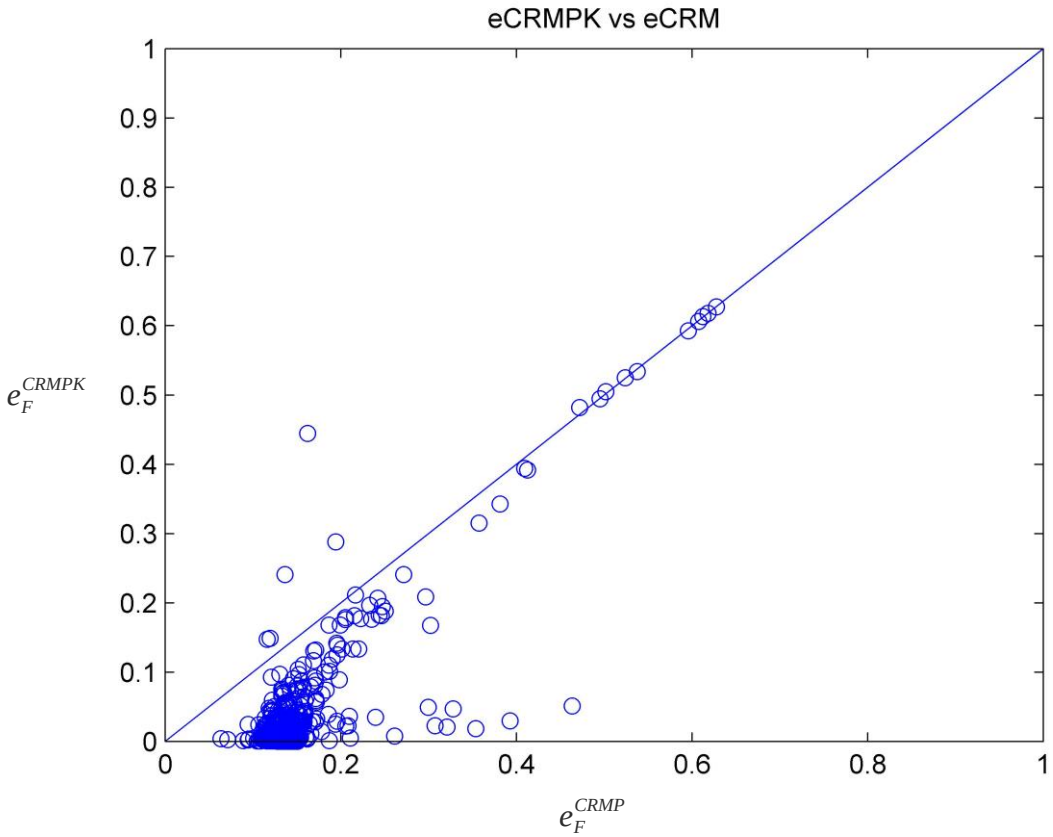

Figura 35: Errores de identificación CRMPK vs CRMP.

De aquí se concluye que CRMPK reduce el error medio en un factor de 6.4x y también reduce el número total de iteraciones del algoritmo en un 22%. Por otro lado se puede observar de la figura anterior que CRMPK da un resultado más preciso en un 99.2% de los ejemplos.

5.3.2. Caso Diana

En este caso de prueba se considera un caso real provisto por el CIMEC, llamado Diana. El mismo consta de 157 etapas, 178 inyectores, 80 productores y un total de 1904 coeficientes de flujo. Además sólo se cuenta con datos de inyección y producción.

En primer lugar se llevó a cabo una prueba con CRMP con el coeficiente $\mu_f = 10^{-6}$. Este coeficiente penaliza el sistema de forma que mejora el número de condición y hace posible su resolución, pero a la vez perturba el problema alejándolo del original, por lo cual es deseable mantenerlo lo más bajo posible.

La prueba con CRMP no ha convergido por iteraciones, es decir, superó el máximo de iteraciones de ALM impuesto en 200, entonces se corrió otra prueba incrementando μ_f a $\mu_f = 10^{-5}$, con lo cual ahora sí el problema convergió en 107 iteraciones dando un error medio absoluto en lo caudales de 14.73[STB/D] lo que corresponde a un 17% de los caudales observados medios (84.6[STB/D]).

Luego se llevaron a cabo dos pruebas con CRMPK sin variar μ_f contemplando distintos valores de μ_{CRMPK}. Para $\mu_{CRMPK} = 100$ el problema convergió en 77 iteraciones arrojando un error medio absoluto en los caudales de 14.76[STB/D]. Para $\mu_{CRMPK} = 1000$ el problema no converge debido a que el coeficiente de penalización alto mal condiciona demasiado el sistema.

A continuación se muestra una tabla sintetizando en ésta los datos de las pruebas realizadas.

Método	μ_f	μ_{CRMPK}	Error en los caudales	Iteraciones ALM
CRMP	10^{-6}	-	-	-
CRMP	10^{-5}	-	14.73	107
CRMPK	10^{-6}	100	14.76	77
CRMPK	10^{-6}	1000	-	-

Tabla 13: Resultados de prueba para caso Diana en IDENT.

De estas últimas pruebas se puede concluir que CRMPK converge en menos iteraciones que CRMP pero este último logra un resultado más preciso por muy poco. La ganancia en precisión con CRMP para este caso es insignificante, por lo tanto se podría sacrificar una mínima diferencia en precisión por una convergencia en menor número de iteraciones. Además CRMPK permite que el problema converja utilizando un valor de μ_f un orden de magnitud mas bajo que CRMP lo cual hace que la solución sea de mejor calidad ya que el problema se perturba en menor medida.

5.3.3. Caso Abcoa

Para ésta prueba se considera otro caso real provisto por el CIMEC, llamado Abcoa. El mismo consta de 123 etapas, 41 inyectores, 35 productores y un total de 752 coeficientes de flujo. Al igual que en la prueba anterior sólo se cuenta con datos de inyección y producción.

En primer lugar se llevó a cabo una prueba con CRMP, utilizando el valor $\mu_f = 10^{-6}$. La prueba no convergió por iteraciones, es decir, superó el máximo de iteraciones de ALM impuesto en 200, entonces se corrió otra prueba incrementando μ_f a $\mu_f = 10^{-5}$, con lo cual el problema convergió en 67 iteraciones arrojando un error medio absoluto en lo caudales de 2.32[STB/D] lo que corresponde a un 11% de los caudales observados medios (19.52[STB/D]).

Luego se llevaron a cabo tres pruebas con CRMPK sin variar μ_f contemplando distintos valores de μ_{CRMPK}. Para $\mu_{CRMPK} = 1$ el problema convergió en 49 iteraciones arrojando un error medio absoluto en los caudales de 2.27[STB/D], luego con $\mu_{CRMPK} = 10$ el problema convergió en tan sólo 33 iteraciones dando un error medio absoluto de 2.26[STB/D]. Por último se corrió una prueba con $\mu_{CRMPK} = 100$ la cual convergió en 198 iteraciones dando el mismo error medio absoluto que la prueba anterior (2.26[STB/D]).
A continuación se muestra una tabla sintetizando lo anteriormente mencionado.

Método	μ_f	μ_{CRMPK}	Error en los caudales	Iteraciones ALM
CRMP	10^{-6}	-	-	-
CRMP	10^{-5}	-	2.32	67
CRMPK	10^{-6}	1	2.27	49
CRMPK	10^{-6}	10	2.26	33
CRMPK	10^{-6}	100	2.26	198

Tabla 14: Resultados de prueba para caso Abcoa en IDENT.

De esta prueba, se puede concluir que CRMPK supera a CRMP en precisión ya que en todas las pruebas efectuadas con este primero se consiguen resultados con menor error. Además, para el caso con $\mu_{CRMPK} = 10$, se puede ver que CRMPK converge en aproximadamente la mitad de iteraciones que CRMP, lo cual representa una notable mejora en velocidad de convergencia.

6. Sahara

6.1. Introducción

Sahara®[2][3] es un sistema de análisis, visualización y seguimiento de reservorios de petróleo desarrollado por la empresa Interfases S.A., este sistema ha sido concebido para el manejo en forma integrada del gran volumen de información necesario para lograr la comprensión de los mecanismos físicos que gobiernan el comportamiento de un reservorio.

Mediante las herramientas de cálculo el usuario puede obtener rápidamente resultados útiles para el análisis y seguimiento del reservorio, además el software provee una gran variedad de gráficas en 2 y 3 dimensiones, las cuales brindan una extensa gama de opciones de visualización.

Sahara® es un software propietario desarrollado en Argentina y actualmente cuenta con usuarios en los siguientes países: Angola, Argentina, Canadá, Chile, Colombia, Ecuador, Egipto, EUA, Malasia, México, Nueva Zelanda, Perú, Tailandia y Venezuela.

6.2. Historia

El origen real se remonta hacia 1977, cuando la empresa Bridas necesitaba trabajar en un yacimiento en la Cuenca del Golfo San Jorge que tenía una geología muy compleja, y las simulaciones convencionales no daban resultado o demandaban mucho tiempo. En ese momento los ingenieros Jorge Valle, Antonio Paradiso y Alberto Gil desarrollaron un simulador el cual tuvo un gran éxito en el mencionado caso.

Con el tiempo, y sobre la base de aquélla, se creó una herramienta nueva, migrada a una tecnología más moderna, y ofrecieron servicios de consultoría de ingeniería, además del software.

Unos años después, se fundó una empresa sólo con la herramienta, que también creció y agregó funciones para dejar de ser sólo un simulador. Se llamó Sahara®, un nombre elegido por su carácter internacional, que no requiere traducción.

El hecho de ser una empresa pequeña, flexible (cuenta con 17 empleados, entre ingenieros, geólogos y especialistas en sistemas), con posibilidad de escuchar y responder a las demandas del usuario final, ha sido la clave del gran crecimiento de la misma.

6.3. El software Sahara

Sahara® es un sistema concebido para el manejo en forma integrada del gran volumen de información necesario para lograr la comprensión de los mecanismos físicos que gobiernan el comportamiento de un reservorio. Además, mediante las herramientas de cálculo el usuario puede obtener rápidamente resultados útiles para el análisis y seguimiento del reservorio, con varias opciones de visualización y ventanas que favorecen el trabajo interactivo que brinda un acceso integrado a la información.

Hoy Sahara® ya no es sólo un simulador; tiene un simulador como una de sus tantas herramientas, pero es una herramienta en sí, que permite utilizar mucha información (porque el negocio de extracción de petróleo maneja un gran volumen de información), de los reservorios ya que en estos se mide mucho, por lo cual visualizar todo eso es muy difícil. La prioridad del software es dar al usuario una interfaz amigable y con una alta potencia gráfica, esto es, mostrar la información recabada en los reservorios con facilidad y disponer de muchas opciones que el usuario pueda configurar. El resultado es una herramienta poderosa, pero muy sencilla de usar. La popularidad de Sahara® creció pese a que no ofrece soluciones ad-hoc ni módulos, sino que ofrece la herramienta "entera", con todas las utilidades disponibles, el usuario puede usar menos, pero la experiencia de la empresa demuestra que aunque lo compren para su uso en algo específico, luego prueban las utilidades que vienen adjuntas y las comienzan a utilizar también.

Entre las numerosas herramientas que se utilizan para la toma de decisiones en la industria del petróleo, ya sean numéricas o no numéricas, incluidas o no en plataformas más generales, el caso de Sahara®, sin duda, se inscribe como uno de los grandes éxitos argentinos. Pese a su bajo perfil, es conocida y utilizada por una gran parte de las empresas que trabajan en el país, pero también es utilizada en el exterior, donde la competencia es mucho mayor.

Como no se ciñe estrictamente a una geología en particular, Sahara® puede aplicarse a reservorios de todo el planeta. Es por esto que hoy día se utiliza en varios países. La difusión se debe tanto a la sucursal que tiene la empresa en Houston (Texas, Estados Unidos), ITF Software, como a su estrategia de difundirla en foros y conferencias internacionales, pero principalmente, se debe al buen resultado que les brindó a los profesionales de manera tal que los usuarios satisfechos por la ecuación (manejo simplificado de mucha información, más resultados satisfactorios) han hecho que luego se la "lleven" cuando son trasladados a otros destinos del mundo, y así se ha ido extendiendo. Hoy no es difícil encontrar la herramienta en los Estados Unidos o en Tailandia.

6.4. La librería IDENT en Sahara

La librería de identificación IDENT[1] desarrollada por el CIMEC ya se encuentra implementada en Sahara® como una utilidad con la cual los usuarios pueden hacer uso para llevar a cabo la identificación de parámetros en reservorios. Actualmente se está trabajando para que el método CRMPK desarrollado en este trabajo también sea implementado en Sahara®.

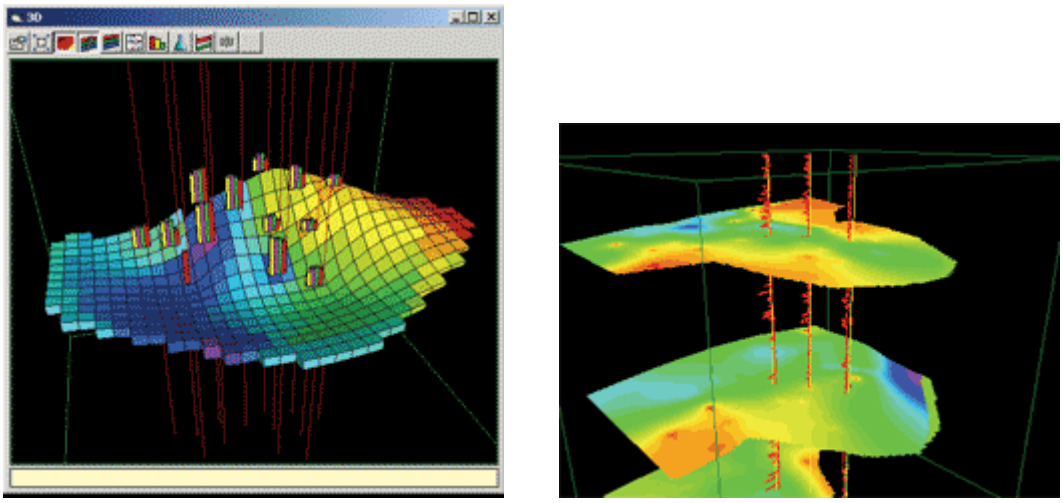

Figura 36: Captura de la interfaz gráfica 3D de Sahara.

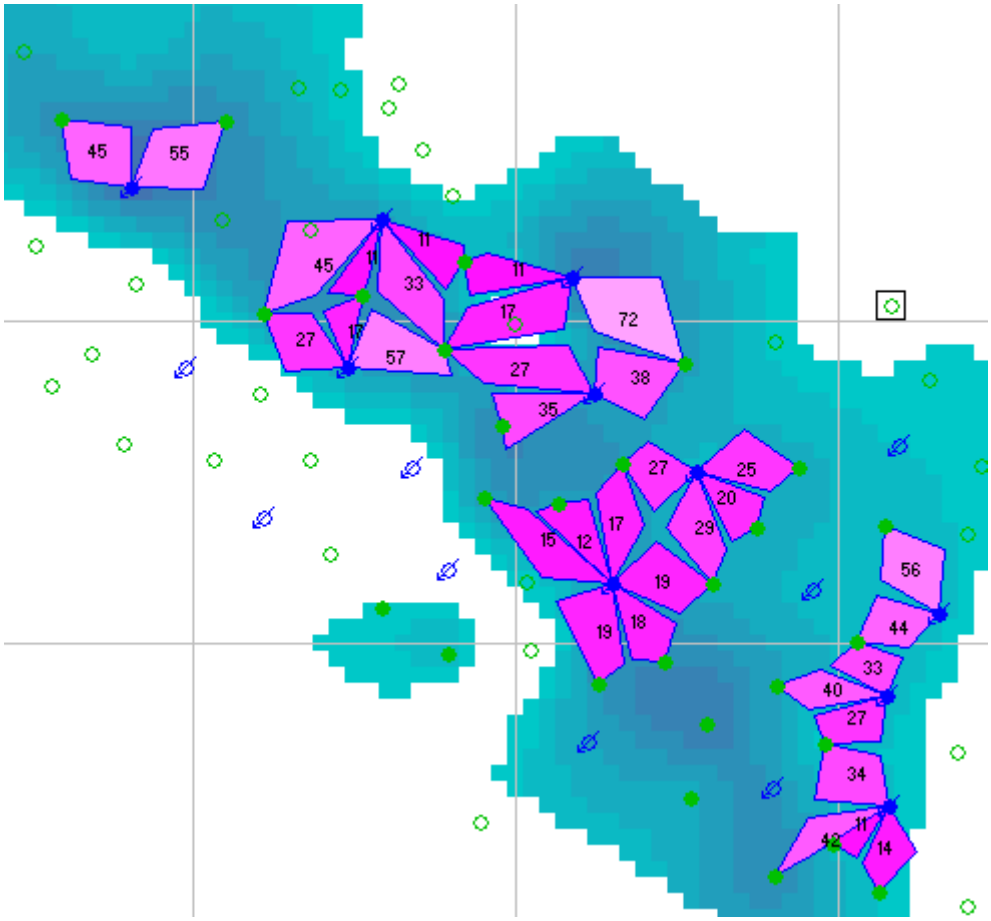

Figura 37: Captura de la interfaz gráfica de Sahara mostrando la configuración de un reservorio.

7. Conclusiones y trabajos a futuro

El proyecto llevado a cabo contempló las actividades necesarias para la optimización de un código de identificación de parámetros, con el objetivo de implementar mejoras tanto en su precisión como en su velocidad.

Para mejorar la eficiencia de la librería IDENT[1], aumentar la precisión y lograr un mayor aprovechamiento de la información de entrada, se desarrolló un método llamado CRMPK (Capacitive Resistive Method for Producers preseving “K”) el cual fue incluido en la librería IDENT. El proceso de desarrollo llevado a cabo contempló en primera medida el análisis del método CRMP[6] y la forma en que éste se implementa en IDENT. Debido a la existencia de un acuerdo de confidencialidad sobre la librería IDENT entre el CIMEC[16] y la empresa Interfaces S.A.[2][3], se organizaron reuniones con miembros del CIMEC para poder trabajar en el código de manera de mantener las restricciones de confidencialidad. Por otra parte, al momento de analizar el método CRMP se pudieron aplicar los conocimientos adquiridos durante la carrera, ayudando a comprender los diferentes mecanismos que implementa el método.

En segunda instancia se llevó a cabo la etapa de análisis del método ALM[7] y su implementación en Octave[17] siguiendo las pautas aprendidas en las materias relacionadas con éstas temáticas. Además se propusieron e implementaron diferentes casos de pruebas simples para corroborar el correcto funcionamiento del método. Todos los resultados fueron validados y aprobados por el director del proyecto.

En tercera instancia se desarrolló un algoritmo que permite identificar los cambios en las conectividades entre dos etapas. Para esto se analizaron la estructura y formato de los datos de entrada, luego se diseñaron las estructuras de datos para contener la información a procesar, y finalmente con todos los datos ya almacenados en las estructuras correspondientes, se implementó un algoritmo que permite extraer las relaciones entre los distintos coeficientes de flujo de un dado reservorio. Este algoritmo fue validado con casos de distintas dimensiones y complejidades para evaluar su precisión y robustez.

Luego de afianzar y consolidar el conocimiento y funcionamiento de los métodos CRMP y ALM, como así también de desarrollar el algoritmo de identificación de cambio en conectividades, se llevó a cabo la etapa de implementación del método CRMPK.

Al momento de comenzar con el desarrollo, se investigaron las alternativas con respecto a las tecnologías existentes. Previo a la elección del lenguaje de programación C++, se comenzó utilizando Octave, pero debido a que la implementación de la librería IDENT es en C++, se optó por migrar a éste último.

A medida que se avanzaba con la implementación del método, se estableció, junto con los desarrolladores de IDENT, que el modo de imponer las restricciones CRMPK, sería imponer estas como penalización al funcional a minimizar. Cabe destacar que debido a la gran cantidad de dependencias que posee la librería IDENT y la dificultad técnica de instalar estas, se debió trabajar de forma remota en el cluster del CONICET lo que ha sido un gran desafío ya que se debió aprender a trabajar sobre una arquitectura tipo cluster por consola a través de una conexión SSH.

Con el objetivo de lograr un incremento en el desempeño de la librería IDENT se implementaron distintas mejoras a la misma sin utilizar paralelización debido a la existencia de dependencias y bucles con un número indeterminado de iteraciones. Las mejoras implementadas en el algoritmo permitieron conseguir una notable mejora en el rendimiento de hasta 11X, es decir 11 veces más rápido. Además se observó que a medida que el problema se vuelve grande también aumenta el rendimiento obtenido con esta optimización lo cual es de gran importancia ya que permite que la optimización beneficie más a los casos que demandan gran cantidad de tiempo.

Finalmente, se desarrolló y efectuó el proceso de pruebas del método CRMPK desarrollado en este proyecto final de carrera. Por un lado se hicieron pruebas en Octave[17] bajo diferentes situaciones con distintas cantidades de observaciones en cada etapa y distintos parámetros algorítmicos. Estas últimas fueron de gran utilidad para determinar los parámetros del algoritmo y obtener un indicador de las ventajas de CRMPK. Por otro lado a partir de los parámetros determinados en las pruebas en Octave, se realizaron pruebas de la librería IDENT tanto con el método CRMPK como con CRMP utilizando casos provistos por la empresa Interfaces S.A. De estas últimas pruebas se pudo observar que el uso de CRMPK, en ciertos casos mejora tanto la precisión como la velocidad de convergencia haciendo a este de gran utilidad.

Se puede concluir que el proyecto desarrollado pudo cumplir con los objetivos propuestos. A continuación se presenta un conjunto de nuevas propuestas para mejorar diferentes aspectos del método CRMPK, como así también, nuevas funcionalidades

- Implementación de las restricciones como restricciones ALM: En este punto, se sabe que el método CRMPK mejora la precisión y velocidad de convergencia, por lo tanto es de interés poder lograr mayor automatización reduciendo la cantidad de parámetros a elección del usuario; una de las formas de lograr esto es incluyendo las restricciones CRMPK en el algoritmo ALM liberando al usuario de la elección del parámetro μ_{CRMPK}.
- Poner a prueba el método bajo mayor cantidad de casos de prueba: Los casos con los que se ha probado el algoritmo fueron provistos por la empresa Interfaces S.A., es de destacar que no es fácil obtener datos reales de reservorios y estos tienen un alto valor comercial. Es de interés evaluar el método CRMPK bajo mayor cantidad de casos de prueba para así obtener resultados de rendimiento bajo un gran abanico de casos.
- Implementación de CRMPK en Sahara®: Se está trabajando para que el método CRMPK desarrollado en este trabajo sea implementado en Sahara® lo cual abre una gran ventana de posibilidad de crecimiento del método a futuro. Es un gran halago que el método desarrollado en este trabajo forme parte de Sahara® y esté disponible para su uso en decenas de empresas al rededor del mundo.

Bibliografía

[1] *Mario Storti (CIMEC. CONICET-UNL).* "Identificación de parámetros en modelos CRMP/CRMIP". Argentina, 2016.

[2] *Sahara® (Interfaces S.A.*). <http://www.interfaces.com.ar>.

[3] *Maria sol Fraguio, Alejandro Lacivita y Jorge Valle (Interfaces S.A.) y Mario Storti (CIMEC.* CONICET-UNL). "Desarrollo e integración de Modelos Capacitivos Resistivos en un simulador analítico de mallas de inyección*".* Revista Petrotecnia, Octubre de 2015.

[4] *Albertoni, A.* "Inferring Interwell Connectivity Only From Well-Rate Fluctuations in Waterfloods". M. S. Thesis, The University of Texas at Austin, Austin, Texas. 2002.

[5] *Jong S. Kim y Larry W. Lake, Thomas F. Edgar.* "Integrated Capacitance-Resistance Model for Characterizing Waterflooded Reservoirs". Universidad de Texas, EEUU, Junio de 2012.

[6] *Weber, D. B*. "The Use of Capacitance-Resistance Models to Optimize Injection Allocation and Well Location in Water Floods". Universidad de Texas, EEUU, Agosto de 2009.

[7] *Jorge Nocedal y Stephen J. Wright*. "Numerical Optimization". Editorial Springer. 2006.

[8] *Nguyen, A. P.; Lasdon, L. S.; Lake, L W. and T. F. Edgar.* "Capacitance Resistive Model Application to Optimize Waterflood in a West Texas Field". SPE-146984-MS. 2011.

[9] *Yousef, A. A.; Gentil, P. H.; Jensen, J. L. and L. W. Lake* . "A Capacitance Model to Infer Interwell Connectivity from Production and Injection Rate Fluctuations". Paper SPE 95322 presented at the SPE Annual Technical Conference and Exhibition, San Antonio, Texas, 9-12 October 2005.

[10] *Liang, X., Weber, D.B., Edgar, T.F., Lake, L.W., Sayarpour, M. and Yousef, A.A*. "Optimization of Oil Production Based on a Capacitance Model of Production and Injection Rates". Paper SPE 107713, presented at the 2007 SPE Hydrocarbon Economics and Evaluation Symposium, Dallas, TX, 1-3 April 2007.

[11] *Albertoni, A. and L. W. Lake*, "Inferring Connectivity Only From Well-Rate Fluctuations in Waterfloods", SPE Reservoir Evaluation and Engineering Journal, 6 (1): 6-16. 2003.

[12] *Sayarpour, M.* "Development and Application of Capacitance-Resistive Models in Water/CO2 Floods", Ph. D. Dissertation. The University of Texas at Austin, Austin, Texas. 2008.

[13] *Burden, R.L., Faires, J.D.* "Numerical Analysis 7th Edition". Thompson Learning. 2002.

[14] *SGI*. "Standard Template Library Programmer's Guide". 1999. <http://www.sgi.com/tech/stl>.

[15] *Storti M., Nigro N., Paz R., Dalcin L.* "PETSc-FEM - A General Purpose, Parallel, Multi-Physics FEM Program". 2010. <http://www.cimec.org.ar/petscfem>.

[16] *CIMEC. Centro de Investigación de Métodos Computacionales*. Predio CONICET Santa Fe "Dr. Alberto Cassano", 3000 Santa Fe, Argentina. <http://www.cimec.org.ar/ >

[17] *John W. Eaton J., David Bateman, Søren Hauberg, Rik Wehbring*. “GNU Octave version 4.2.0”. manual: a high-level interactive language for numerical computations. 2016<http://www.gnu.org/software/octave/doc/interpreter/>

Anexo