RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL
POR: VICTOR ROCO C.
FECHA: 25 /08 /2008
PROF. GUÍA: Sr. FRANCISCO MARTINEZ C.

# NUEVOS MÉTODOS DE CALIBRACIÓN DE PROBLEMAS DE ENTROPÍA

El objetivo general del presente trabajo de título, es estudiar el rendimiento computacional de dos nuevos métodos de calibración de entropía ante situaciones reales de gran tamaño, estableciendo como comparación métodos clásicos de calibración. Se espera identificarlos por tiempo de ejecución ante distintas situaciones, variando el número de restricciones, homogeneidad de la matriz de las mismas y grado de convergencia.

Los modelos de máxima entropía son empleados para determinar la distribución de una variable frente a condiciones de información limitada. Su proceso de calibración es una tarea que requiere de un gran esfuerzo computacional, siendo relevante en modelos de gran tamaño. En el área de transporte se pueden apreciar en problemas de distribución espacial de viajes, asignación de viajes a la red vial y problemas de localización de hogares y firmas, entre otros.

Se implementaron seis métodos computacionales clásicos en conjunto con dos nuevos, y se aplicaron al modelo de distribución de viajes con múltiples categorías de usuarios. Estos métodos fueron depurados y sometidos a distintas pruebas con datos simulados, en donde se identificó su velocidad de convergencia en cada situación. Además, estos se probaron ante la calibración de una matriz de viajes correspondiente a la ciudad de Santiago.

Como resultado se obtuvo una caracterización de los métodos frente a las distintas situaciones; de los tres métodos más rápidos en todos los escenarios simulados, dos corresponden a los métodos nuevos. Se comprobó además que esta relación se mantiene para el caso de la matriz de viajes real.

Se concluye que uno de los métodos nuevos, que realiza un cambio de variables para acotar la búsqueda de la solución, se encuentra cercano en tiempos de ejecución con el método clásico más rápido. Siendo tiempos de ejecución de los demás métodos clásicos, varios ordenes de magnitud superiores. Además, este método se mantiene estable en la medida que la matriz de restricciones aumenta su homogeneidad, por lo que se recomienda su uso ante esta situación.

*Dedicado a mi familia, amigos y profesores.*

UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA CIVIL

# *NUEVOS MÉTODOS DE CALIBRACIÓN DE PROBLEMAS DE ENTROPÍA*

**MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL**

**VÍCTOR ANTONIO ROCO CASTILLO**

PROFESOR GUÍA:
FRANCISCO MARTÍNEZ CONCHA

MIEMBROS DE LA COMISIÓN:
PEDRO DONOSO SIERRA
MARCELA MUNIZAGA MUÑOZ

SANTIAGO DE CHILE
AGOSTO 2008

# Índice general

# Índice de figuras

# Índice de cuadros

# NOTACIÓN

La notación utilizada es la siguiente:

| | |
|---|---|
| i: | Zona de origen del viaje |
| j: | Zona de destino del viaje |
| N: | Total de zonas |
| n: | Clase de usuario |
| M: | Total de categorías de usuarios |
| $T_{nij}$: | Número de viajes con origen en la zona i y destino en la zona j para categoría n |
| T: | Número de viajes totales |
| $C_{0n}$: | Costo totales de viajes para categoría n |
| $c_{nij}$: | Costo de realizar un viaje de la zona i a la zona j, para el usuario de categoría n |
| $\alpha_i$: | Multiplicador de Lagrange asociado a la restricción de Orígen |
| $\theta_j$: | Multiplicador de Lagrange asociado a la restricción de Destino |
| $\beta_n$: | Multiplicador de Lagrange asociado a la restricción de Costo |
| $A_i$: | Factor de balance asociado al multiplicador de Lagrange $\alpha_i$ |
| $B_j$: | Factor de balance asociado al multiplicador de Lagrange $\theta_j$ |
| (k): | súper índice que indica las iteraciones globales |

Capítulo 1

# INTRODUCCIÓN

Los modelos de maximización de entropía (ME) son empleados para conocer la distribución de alguna variable observada bajo condiciones de información limitada. Su aplicación actual se puede observar en diversos ámbitos de la ingeniería, tales como la mecánica estadística, reconstitución de imágenes, teoría de colas, planificación de sistemas de usos de suelo y transporte, entre otros (Schofield, 2007, ver por ejemplo). En particular para el caso del transporte, los modelos ME se han usado, para representar la distribución espacial y la asignación de viajes a la red vial, entre otros casos.

Por otra parte, los modelos Logit Multinomial (MNL), introducidos al área del transporte por Domencich y McFadden (1975), son ampliamente empleados en el ámbito urbano para la representación y predicción del comportamiento de consumidores frente a la elección de alternativas discretas. El modelo MNL surge al asumir que cada consumidor maximiza su utilidad aleatoria, la cual sigue una distribución de probabilidad iid Gumbell, cuyos parámetros se estiman empleando el criterio de máxima verosimilitud. Su aplicación clásica en el área del transporte se puede observar en los modelos de elección discreta donde un usuario se ve enfrentado a la elección del medio de transporte que empleará para la realización de su viaje entre un conjunto discreto de alternativas.

Pese a tener una génesis diferente, los modelos MNL estimados por máxima verosimilitud y ME corresponden a problemas de optimización que presentan el mismo conjunto de condiciones de optimalidad, con lo cual son equivalentes. Esta equivalencia fue señalada por Anas (1983), quien demuestra en su trabajo que los modelos ME son equivalentes con los modelos MNL con estimadores máximo verosímiles. Varios autores han generalizado esta equivalencia posteriormente, demostrando que, para cualquier función de probabilidad, el modelo de máxima verosimilitud (MV) es el dual geométrico del modelo ME (Y. y A., 2006, ver por ejemplo ). Estos resultados muestran que la equivalencia entre los modelos ME y MV es más general de lo que Anas demostraba para el caso particular del modelo MNL.

Con gran frecuencia los modelos de ME son empleados en situaciones en que el problema de elección se ve enfrentado a un conjunto amplio de restricciones, ya sean en totales de viajes con origen o destinos en una zona (modelo de distribución de viajes clásico), como también en el caso de ajuste de distribuciones de viajes sujetas a restricciones de conteos de flujos Willumsen (1981), o para el caso de la producción de oferta inmobiliaria bajo regulaciones zonales Martinez y Henríquez (2007). Los modelos de elección discreta, donde el modelo MNL presenta su mayor número de aplicaciones, son usualmente considerados en conjuntos de menor tamaño que los modelos ME, con una consecuente disminución de parámetros a estimar en su proceso de calibración.

## 1.1. Motivación

La calibración de los modelos ME y MNL con estimadores MV, consiste en la obtención de los parámetros que ajustan el modelo de manera de representar de mejor manera los datos observados. Este proceso es fundamental previo a la utilización de los modelos para predicción y simulación de distintos escenarios, tareas comunes e importantes en la ingeniería.

El proceso de calibración es una tarea usualmente costosa para los modelos de máxima entropía con restricciones lineales, lo cual se reconoce por diversos autores por ejemplo Gonçalves y de Cursi (2001), Malouf (2002) o Wu (1997). Como es de esperarse, en la medida que los problemas aumentan de tamaño, aumentan también las dificultades de calibración. Para el caso de los modelos ME, el aumento del tamaño del problema implica la calibración de un mayor número de variables. Si se considera por ejemplo el modelo de distribución de viajes, para una ciudad de 500 zonas, se deberá realizar la calibración de al menos 500 variables si el modelo es simplemente acotado, 1000 variables si se considera el modelo doblemente acotado y estas cifras aumentan al considerar mayor número de usuarios, o mayor desagregación. Para el caso de Santiago, el modelo estratégico empleado considera más de 600 zonas y 13 categorías de usuarios distintas.

Obtener técnicas de calibración eficientes, se torna una tarea relevante no solo por liberar recursos computacionales, si no al permitir incorporar diversas mejoras a los modelos. Estas podrán ser debido a un mayor número de restricciones ( incorporando mayor información del comportamiento a los modelos ) o bien, debido a un proceso de calibración con información desagregada, pudiendo aumentar por ejemplo, el número de zonas de un modelo de distribución, o el número de categorías de usuarios a considerar en un modelo de localización.

Las técnicas de calibración de los modelos ME han ido de la mano con el desarrollo de los distintos modelos de máxima entropía, en donde es conveniente usar técnicas de calibración diferente según la naturaleza de las restricciones. Tal como se mencionó antes, los modelos ME y MNL calibrados por MV, son equivalentes y pueden por lo tanto, ser calibrados mediante diversas técnicas numéricas. Al coincidir en las condiciones que definen los óptimos, es posible emplear las técnicas de calibración de entropía para calibrar modelos MNL y viceversa. Existen diversos métodos para la calibración de modelos de entropía con restricciones lineales, en donde se destacan principalmente: el método de balance de Bregman, la Técnica de Reconstitución Multiplicativa Algebraica (MART), el método de Newton, el método de Hyman, (ver mayor detalle en Fang y Tsao, 1995), además de otros métodos heurísticos.

## 1.2. Objetivos

En este trabajo se estudiarán distintos algoritmos de calibración de modelos de entropía, estos se aplicarán a un modelo ME particular el cual presenta restricciones lineales de igualdad. Los algoritmos serán identificados por tiempo de ejecución frente a distintos análisis, principalmente en torno al aumento del número de restricciones y cambios en la homogeneidad de las mismas.

Además se propondrán dos nuevos algoritmos, los cuales serán contrastados con los métodos clásicos encontrados en la literatura. Finalmente los algoritmos serán probados frente a un problema de calibración real, donde se compararán los tiempos de ejecución de los mismos frente a la calibración de un modelo de distribución de viajes, empleando datos reales, para la ciudad de Santiago.

## 1.3. Alcances

En este trabajo se abarcan métodos clásicos de solución numérica de problemas, técnicas más recientes como optimización por algoritmos genéticos no son consideradas. Además se resolverá el problema de máxima entropía con restricciones lineales; no se considerarán especificaciones de entropía como por ejemplo restricciones cuadráticas o restricciones de desigualdad, las cuales pueden requerir algoritmos de calibración distintos a los presentados en este trabajo.

Se implementarán y realizarán pruebas de velocidad sobre los distintos algoritmos para calibrar el problema de entropía, contrastándolos con dos nuevos métodos propuestos; específicamente se prueban los algoritmos aplicados al problema de distribución de viajes con múltiples categorías de usuario. Las pruebas se realizan tanto para datos simulados, como para bases

de datos reales, de manera de identificarlos por costos computacionales únicamente (tiempo de ejecución). En este sentido se debe destacar que bajo el enfoque de calibración de parámetros se busca el cumplimiento de las condiciones de primer orden, por lo mismo no se espera obtener estadísticos de ajuste de la calibración que identifiquen la bondad del modelo, sino que únicamente se manejará una variable de ajuste en el cumplimiento de las condiciones de primer orden.

En el segundo capítulo se revisan los modelos de transporte que emplean el concepto de máxima entropía, además de los distintos métodos de calibración de parámetros en la literatura. En el tercer capítulo se formula el problema de máxima entropía sobre los cuales se trabajan y prueban los distintos algoritmos, además se plantean los algoritmos clásicos para la calibración y se formulan los nuevos. En el cuarto capítulo se describe la implementación de los algoritmos tratados y se describe la operación de los métodos programados para dar paso en el quinto capítulo a una revisión de las mejoras implementadas y sus efectos sobre el rendimiento de los algoritmos. En el sexto capítulo se prueban los tiempos de ejecución frente a distintas situaciones y se analizan sus resultados. Finalmente se prueban los algoritmos sobre matrices reales provenientes de la calibración de la Encuesta Origen-Destino realizada en la ciudad de Santiago el año 2002.

Capítulo 2

# REVISIÓN DE LOS MODELOS DE ENTROPÍA

En este capítulo se presenta una revisión de los modelos de entropía, para ello se separa el capítulo en cinco secciones. En primer lugar se formula matemáticamente el problema de máxima entropía general y el problema de entropía en probabilidad. En segundo lugar se formula su solución óptima. A continuación, se plantea la génesis de los modelos de máxima entropía desde un enfoque combinatorial y bajo el enfoque de la teoría de la información. En una cuarta sección se presenta el problema dual equivalente para el problema de máxima entropía. Luego, se presentará la equivalencia con los modelos máximo verosímiles MNL,y finalmente, se presenta una revisión de métodos empleados y variantes del problema. Se revisan también los métodos empleados para calibrar modelos de entropía en transporte, y su aplicación a los problemas planteados para transporte.

## 2.1. Definición del Problema General de Máxima Entropía

Consideremos el siguiente problema general de máxima entropía con restricciones lineales (MEG):

$$\max_{x_j} -\sum_{j}^{J} x_j(\ln(x_j) - 1) \tag{2.1}$$

s.a.:

$$\sum_{j=1}^{J} x_j = X \tag{2.2a}$$

$$\sum_{j=1}^{J} a_{ij}x_j = b_i \quad \text{para } i = 1, 2, \ldots I \tag{2.2b}$$

$$x_j \geq 0 \quad \text{para } j = 1, 2, \ldots J \tag{2.2c}$$

En este problema la primera restricción supone un conocimiento del total de la variable $x$ igual a $X$. La restricción (2.2b) representa un conjunto de restricciones lineales que agregan

información o certeza al problema. Estas restriciónes estan dadas por una matriz de dimensión I x J , donde J es el número de las variables $x_j$ a optimizar, y I es el número total de restricciones existentes.

El problema antes descrito puede ser formulado también bajo el enfoque de distribuciones de probabilidad, para ello consideremos la variable $p_i \in [0, 1]$ definida como sigue:

$$p_j = \frac{x_j}{\sum_j^J x_j} \tag{2.3}$$

Con esta variable podemos considerar el siguiente proble de máxima entropía en probabilidad (MEP)

$$\max_{p_j} \sum_j^J p_j(\ln(p_j) - 1) \tag{2.4}$$

s.a.:

$$\sum_{j=1}^J a_{ij}x_j \quad = \bar{b}_i \quad \text{para } i = 1, 2, \ldots I \tag{2.5a}$$

$$\textstyle\sum_i^I p_i \quad = 1 \tag{2.5b}$$

Como puede apreciarse el problema MEP es equivalente al problema MEG siempre que

$$\bar{b}_i = \frac{b_i}{\sum_j^J x_j} \tag{2.6}$$

## 2.2. Solución Optima del Problema General ME

La solución de este problema de optimización esta dado por la resolución de las condiciones de K.K.T. asociadas a la siguiente función lagrangeana:

$$L(x, w) = \sum_j^J x_j \ln(x_j) - \alpha(\sum_{i=1}^I x_i - X) + \sum_{i=1}^I w_i(\sum_{j=1}^J a_{ij}x_j - b_i) \tag{2.7}$$

donde $x_j \geq 0$. Además $w_i \in \Re$ son los multiplicadores de Lagrange asociados las restricciones lineales.

Las condiciones de optimalidad, o de K.K.T. estan dadas por las siguientes relaciones:

$$\frac{\partial L}{\partial x_j} \quad = 0 \quad \text{para } j = 1, 2, \ldots J \tag{2.8a}$$

$$\frac{\partial L}{\partial \alpha} \quad = 0 \tag{2.8b}$$

$$\frac{\partial L}{\partial w_i} \quad = 0 \quad \text{para } i = 1, 2, \ldots I \tag{2.8c}$$

Aplicadas al problema (2.1) nos entrega las siguientes relaciones:

$$x_j = \exp(\sum_{i=1}^{I} a_{ij} w_i - 1) \quad \text{para } j = 1, 2, \ldots J \tag{2.9a}$$

$$\sum_{i=1}^{I} x_i = X \tag{2.9b}$$

$$\sum_{j=1}^{J} a_{ij} x_j = b_i \quad \text{para } i = 1, 2, \ldots I \tag{2.9c}$$

$$x_j \geq 0 \quad \text{para } j = 1, 2, \ldots J \tag{2.9d}$$

En este punto podemos integrar las restriccion (2.9a) en la restricción (2.9c) y desligarnos de la variable a optimizar $x_j$, esto nos permitirá obtener sistemas de ecuaciones sólamente en la variable dual $w_i$. Las ecuaciones generadas son las siguientes:

$$h_i(w) \equiv \sum_{j=1}^{J} a_{ij} \exp(\sum_{k=1}^{I} a_{kj} w_k - 1) - b_i = 0 \text{ para i=1,2, } \ldots \text{I} \tag{2.10}$$

Estas ecuaciones $h_i$ corresponden a las condiciones de optimalidad expresadas en los multiplicadores de Lagrange y son fundamentales para el planteamiento del dual del problema de máxima entropía, que se verá en la sección siguiente.

## 2.3. Planteamiento del Problema Dual de Entropía

Dado el problema primal de entropía, definido en (2.1) y cuya formulación lleva a la función lagrangeana definida por (2.7), se puede enunciar un problema de optimización dual, cuyas condiciones de optimalidad coincidan con las ecuaciónes $h_i$ definidas en (2.10)
Para ello se define la siguiente función:

$$g(w) = \inf_{x \geq 0} L(x, w) \tag{2.11}$$

donde L es la función lagrangeana definida en (2.7) , al sustituir la restricción (2.9a) en (2.7) se podrá prescindir de las variables $x_j$ con lo cual se obtiene la siguiente ecuación:

$$g(w) = -\sum_{j=1}^{J} \exp(\sum_{k=1}^{I} a_{kj} w_k - 1) + \sum_{i=1}^{I} b_i w_i \tag{2.12}$$

De esta forma se puede definir el problema dual de la siguiente manera:

$$\min_{w \in \Re} d(w) = \sum_{j=1}^{J} \exp(\sum_{k=1}^{I} a_{kj} w_k - 1) + \sum_{i=1}^{I} b_i w_i \tag{2.13}$$

Dando orígen a un problema de optimización convexo e irrestricto.

## 2.4. Genesis del Problema de ME

En esta sección se presentará la derivación del problema ME, para ello en primer lugar se empleará el enfoque combinatorial q da orígen al uso de los modelos ME en transporte. En segundo lugar se presenta la deducción del problema ME bajo el enfoque de la teoría de la información.

### 2.4.1. Enfoque Combinatorial

Una primera forma de obtener el modelo de máxima entropía es mediante el enfoque de micro-estados equiprobables, a continuación se presenta la deducción tal como mostrara Ortúzar (1994).

En un sistema compuesto por una gran cantidad de componentes, separables y distinguibles, si se tiene una descripción completa de éste, se podrán especificar *estados micro* en que cada componente es único. Una descripción menos completa podrá definir *estados meso* en donde se podrán identificar muchos estados micro.

Si se considera el problema de la distribución de viajes para ejemplificar, un estado micro corresponderá a la identificación de un viajero con su viaje de orígen y destino. Sin embargo, si sólo nos interesa el total de viajes entre una zona i y una zona j $T_{ij}$, entonces se podría considerar esta descripción como un estado meso. Aún existe un nivel más de agregación, y corresponde al *estado macro* en donde conoceremos sólamente el total de viajes que se originan en una zona $O_i$ o el total de viajes que tienen como destino $D_j$ la zona j. Se debe destacar que muchos estados micro, producen un estado meso. A su vez, muchos estados meso producen un estado macro.

Bajo la hipótesis de que, salvo que exista información adicional, los estados micro serán igualmente probables se llega a la forma funcional de la entropía para encontrar los meso estados. Sólamente se imponen restricciones sobre los estados macro.

Se puede demostrar combinatorialmente, que el número de estados micro (o viajes $T_{ij}$) dado por $\gamma(M)$, asociados a un estado meso (matriz de viajes $M = \{T_{ij}\}_{ij}$) puede calcularse como sigue:

$$\gamma(M) = \frac{T!}{\prod_{ij} T_{ij}!} \tag{2.14}$$

En donde $T$ es el total de viajes, y las únicas matrices $\{T_{ij}\}_{ij}$ contadas son las que cumplen con las restricciones de los estados macro. Al tomar como válida la hipótesis de estados micro equiprobables, resulta evidente que si se desea obtener un estado meso, el más probable de ocurrir será aquel que presente mayor número de estados micro asociados. Por lo tanto

para encontrar el estado meso (o matriz de viaje en el ejemplo), se deberá maximizar $\gamma(M)$.

En lugar de maximizar directamente $\gamma(M)$, se puede maximizar de manera equivalente $ln(\gamma(M))$. Mediante la aproximación de Stirling ( válida para valores de T grande ) se tendrá:

$$lnT_{ij}! \approx T_{ij}lnT_{ij} - T_{ij} + \frac{1}{2}(ln2\pi + lnT_{ij}) \approx T_{ij}lnT_{ij} - T_{ij} \tag{2.15}$$

de donde:

$$ln(\gamma(M)) \approx TlnT - T - \sum_{ij} T_{ij}lnT_{ij} - T_{ij} \tag{2.16}$$

Debido a que $T$ es conocido, maximizar $\gamma(M)$ es equivalente a maximizar

$$-\sum_{ij} T_{ij}lnT_{ij} - T_{ij} \tag{2.17}$$

cantidad que es conocida como entropía. Se debe notar que para el problema planteado (2.1) en las restricciones Ax=b se puede considerar que una de las filas de la matriz A genera la restricción $\sum_i x_i = X$, que en este caso reproduce el conocimiento de el valor de $T$.

### 2.4.2. Enfoque de la teoría de la información

Otra manera de derivar el modelo de entropía se basa en el valor de la información. Para ello es necesario considerar un evento cuya probabilidad de ocurrencia P sea conocida a priori, y una función F que represente el valor de la información de un evento. Este valor representará una medida de la incertidumbre asociada al evento. Como es de esperarse, a medida que la probabilidad de ocurrencia del evento aumente, es decir, este se haga más probable, el valor de la información de P disminuirá. En el sentido contrario, mientras más improbable sea la ocurrencia del evento, mayor valor tendrá la información asociada. Por ejemplo se puede considerar el valor que tendrá el conocimiento de la ocurrencia de un terremoto, que intuitivamente será mayor que el valor del conocimiento de si lloverá un día de invierno. Por lo anterior, propiedades deseables de la función del valor de información serán:

$$F(P = 1) = 0 \tag{2.18}$$

$$\lim_{P\to\infty} F(P) = +\infty \tag{2.19}$$

Además se debería cumplir que si $P1$ y $P2$ son independientes:

$$F(P1 + P2) = F(P1) + F(P2) \tag{2.20}$$

Una función que cumple con estas propiedades es la función logaritmo, luego se puede definir la función F de la siguiente manera

$$F(x) = -\ln(x) \tag{2.21}$$

Si todos los estados posibles de un sistema son discretos y finitos, entonces se define la Entropía de este sistema como:

$$E(p_1, p_2, \ldots, p_n) = \sum_i p_i F(p_i) = -\sum_i p_i \ln(p_i) \tag{2.22}$$

Sujeta a

$$\sum_i p_i = 1 \tag{2.23}$$

ésta se interpreta como *el valor esperado de la medida de incertidumbre de sus estados*. Luego, el modelo de máxima entropía es un modelo que maximiza el valor esperado de la incertidumbre de una variable, sujeta a restricciones que entregan la medida de su certidumbre.

## 2.5. Equivalencia entre modelos MNL y ME

Un modelo de elección discreta clásico en transporte es el Logit Multinomial (MNL), este se deriva de funciones de utilidad aleatorias que siguen una distribución IID Gumbel. En este modelo, la probabilidad de escoger una alternativa i en un conjunto $A_q$ disponible para un individuo q, esta dada por la siguiente expresión:

$$P_i = \frac{\exp(\beta U_i)}{\sum_{j \in A_q} \exp(\beta U_j)} \tag{2.24}$$

La equivalencia de los modelos MNL y ME puede probarse al comparar las condiciones de primer orden obtenidas al calibrar mediante máxima verosimilitud (MV) el modelo logit, con las condiciones de primer orden del modelo ME.
Al reconocer que la función de utilidad $U_i$ es dependiente de parámetros $\theta$ y variables explicativas $x$, entonces se la puede llamar explicitamente:

$$U_i(\theta; x) = \sum_k^K \theta_k x_{ik} + \epsilon \tag{2.25}$$

En donde se ha empleado una especificación lineal para la función de utilidad, el término $\theta_k$ es la constante del atributo k-ésimo a calibrar, $x_{ik}$ son las variables explicativas del atributo k-ésimo y para la alternativa i-ésima y $\epsilon$ sigue una distribución IID Gumbel. De manera consecuente podemos encontrar que $P_i$ será $P_i(\theta; x)$

La calibración del modelo MNL mediante máxima verosimilitud se obtiene al encontrar los parámetros $\theta_k$ que maximizan la expresión:

$$\max_{\theta \in \Re} \sum_i \delta_i \ln P_i(\theta; x) \tag{2.26}$$

Donde $\delta_i$ puede interpretarse como el total de veces que la alternativa i fue elegida, o bién como una variable discreta con el valor 0 si la alternativa no fue escogida y 1 si fue elegida.

Se obtiene el siguiente lagrangeano asociado al problema de calibración por maxima verosimilitud, dado por:

$$\begin{aligned}\frac{\partial L_{MNL}}{\partial \theta_k} &= \sum_i \frac{1}{P_i(\theta;x)} \frac{\partial P_i(\theta;x)}{\partial \theta} \\ &= \sum_i \frac{1}{P_i(\theta;x)} (P_i(\theta;x)x_{ik} - P_i(\theta;x)\sum_i P_i(\theta;x)x_{ik}) \\ &= \sum_i P_i(\theta;x)x_{ik} - \sum_i \delta_i x_{ik} \end{aligned} \tag{2.27}$$

Luego se tiene que:

$$\sum_i P_i(\theta;x)x_{ik} - \sum_i \delta_i x_{ik} = 0 \tag{2.28}$$

Análogamente, podemos reformular el problema de máxima entropía en probabilidad (MEP) de la siguiente manera,

$$\text{máx} - \sum_i P_i \ln P_i \tag{2.29a}$$

s.a. :

$$\sum_i P_i = 1 \tag{2.29b}$$

$$\sum_i P_i x_{ik} = \sum_i \delta_i x_{ik} \tag{2.29c}$$

En esta formulación, las incógnitas son las distribuciones de probabilidad $P_i$, donde las probabilidades deben sumar uno y donde el valor esperado agregado de cada atributo k debe representar el valor observado, lo anterior indicado por las restricciones impuestas.

La solución analítica del problema esta dada por las condiciones de KKT sobre el lagrangeano:

$$L_{MEP} = \sum_i P_i \ln P_i - \alpha(\sum_i P_i - 1) - \sum_k \lambda_k (\sum_i P_i x_{ik} - \sum_i \delta_i x_{ik}) \tag{2.30}$$

Y las condiciones de KKT para este lagrangeano son las siguientes:

$$\frac{\partial L_{ME}}{\partial P_i} = 1 + \ln P_i - \alpha - \textstyle\sum_k \lambda_k x_{ik} = 0 \tag{2.31a}$$

$$\frac{\partial L_{ME}}{\partial \alpha} = \textstyle\sum_i P_i - 1 = 0 \tag{2.31b}$$

$$\frac{\partial L_{ME}}{\partial \lambda_k} = \textstyle\sum_i P_i x_{ik} - \sum_i \delta_i x_{ik} = 0 \tag{2.31c}$$

De ellas se deriva la siguiente relación

$$P_i = \frac{\exp(\sum_k \lambda_k x_{ik})}{\sum_j \exp(\sum_k \lambda_k x_{jk})} \tag{2.32}$$

Debido a que, considerando que la formulación de $P_i$ resultante para el modelo ME coincide a la formulada bajo el modelo MNL, y además, como el conjunto de condiciones de optimalidad (2.28) y (2.31c) son idénticos, se concluye la equivalencia entre los modelos.

## 2.6. Aplicaciones de Entropía en Transporte

Dentro de la planificación de transporte el modelo clásico para la planificación es el llamado modelo de cuatro etapas, el cual se constituye de cuatro submodelos: generación de viajes, distribución de viajes, partición modal y asignación de viajes. La aplicación de modelos de entropía cumplen un rol fundamental, principalmente debido a su empleo en los modelos de distribución. Sin embargo, los modelos de entropía presentan un potencial mayor, el cual se puede apreciar en modelos combinados de distribución de viaje y asignación, modelos de elección modal o en una etapa inicial de uso de suelo, generando un modelo de cinco etapas.

Los primeros modelos de distribución de viajes considerados por los planificadores de transporte, consideraban estimaciones de viajes origen-destino basadas en factores de crecimiento, cuya estructura extrapola una matriz de viaje base a una futura sin considerar cambios en los patrones de viajes. Posteriormente, se desarrollan los modelos gravitacionales, en donde se reconoce una relación analoga entre las fuerzas de atracción gravitacionales y los viajes entre zonas. Para ello consideran que la impedancia de realizar viajes es función decreciente con los costos de la realización de los mismos, y se emplean distinas formas funcionales para identificar los costos asociados.

Es debido a Wilson (1970) que los modelos de entropía son considerados formalmente en el área del transporte al proponer un modelo similar al gravitacional de distribución doblemente acotado, cuya formulación minimiza la entropía de un viaje sujeto a restricciónes tanto en orígenes como destinos.

La formulación de entropía de Wilson corresponde a la siguiente:

$$\max_T -\sum_i^I \sum_j^J T_{ij}(ln(T_{ij}) - 1) \tag{2.33}$$

s.a.

$$\sum_{j}^{J} T_{ij} = O_i \tag{2.34a}$$

$$\sum_{i}^{I} T_{nij} = D_j \tag{2.34b}$$

$$\sum_{i}^{I}\sum_{j}^{J} T_{ij} c_{ij} = C \tag{2.34c}$$

En este modelo, las primeras restricciones corresponden al total de viajes con orígen en una zona i, las segundas corresponden a al total de viajes que presentan como destino la zona j. La tercera restricción corresponde a una restricción lineal, que puede interpretarse como la minimización de un costo total por parte de los realizadores de los viajes $T_{ij}$, y cuyo total C corresponde al total de costos observados en el sistema.

Como se aprecia en (2.33), esta formulación corresponde a un problema de optimización mono-objetivo, formulaciones multi-objetivo del problema de entropía han sido planteadas por ejemplo por Islam y Roy (2006) y también por Leung (2007), donde se consideran como objetivos separados la maximización de la entropía y la minimización de costos. Se puede notar que esta formulación es equivalente con las formulaciones en donde la optimización es realizada sobre costos totales, tal como mostrara Erlander (1981).

A partir del concepto de entropía, comunmente asociado a mecánica estadística (ver Erlander, 1981) e introducido al transporte por Wilson, surgen formulaciones de problemas de asignación y distribución de viajes combinados. Por ejemplo Jörnsten (1981) propone un modelo de distribución y asignación conjunta señalando que para este tipo de problemas una alternativa a considerar en el modelo de distribución podrá ser formulación clásica de Wilson, ver (2.33), en la cual la entropía es la función objetivo y los costos son una de las restricciones. Otros trabajos de este tipo son los de Fisk (1988), Ho et al. (2006), Xu et al. (2008).

También para el caso de generación, distribucion y partición modal conjunta se pueden ver aplicaciones en los que el problema de máxima entropía se encuentra presente, por ejemplo Vrtic et al. (2007) desarrollaron un modelo para aplicar en Suiza, cuyo tamaño es de aproximadamente 3000 zonas, donde se destaca la necesidad de un método de calibración rápido.

En otras áreas del transporte se puede citar a Ham et al. (2005), quienes presentan e

implementan un modelo de envió de cargas interregional y multimodal, en el cual buscan predecir los envíos por zona y producto entre cada par de regiones y el flujo de envíos resultante de ellas. Para ello consideran que en ausencia de la información de los costos de envíos estos pueden ser modelados bajo funciones de entropía.

Se debe destacar también que la aplicación de los algoritmos de entropía va mas allá de su aplicación en el campo del transporte. Problemas de entropía son resueltos en ámbitos tan diversos como lo son la teoría de colas, reconstrucción de imágenes, planificación urbana, entre otras. Una revisión de estos modelos puede ser vista en Fang et al. (1999). Dentro de estos casos, el que surge con mayor relevancia para esta investigación es la aplicación de la entropía al procesamiento estadístico de lenguaje natural (ver por ejemplo Berger et al., 1996) debido al gran tamaño de restricciones y variables a encontrar y su relevancia en tiempos de ejecución.

## 2.7. Métodos de calibración de los modelos de entropía

El problema de maximización de la entropía es un problema de optimización no lineal, cuya calibración es una tarea que requiere el empleo de métodos numéricos distintos según sea la especificación del problema. A continuación se presenta una revisión de los métodos más comunes en la literatura para la calibración del problema de máxima entropía con restricciones lineales.

El métodomás señalado en la literatura, para resolver el problema cuando la fila i-esima de la matriz de restricciones es igual a la unidad, es conocido como: balanceo de parámetros, método de Furnes o DSP (ver Yun y Sen, 1994). Este método se basa en la forma funcional resultante de la solución del problema de optimización, que para el caso en que $a_{ij} = 1$ para algún j dado, la que puede derivarse de las ecuaciónes (2.9a) que toman la siguiente forma :

$$x_j = \exp(\sum_{i=1}^{r} w_i + \sum_{i=r}^{I} a_i j w_i - 1) \tag{2.35}$$

Por lo que al definir:

$$A_i = \frac{\exp(w_i)}{b_i} \text{ para los i tal que } a_{ij} = 1 \text{ para todo j} \tag{2.36}$$

y reemplazarlo en la ecuación (2.9c). Se obtiene:

$$x_j = \prod_{i}^{I} A_i b_i \exp(\sum_{i=r}^{I} a_i j w_i - 1) \tag{2.37}$$

Donde los parámetros $A_i$ se estiman iterativamente como sigue:
Sea $[A_i^0]$ un vector de factores de balance inicial, se actualizará la componente $A_i$ mediante

el siguiente esquema, para la iteración k-ésima:

$$A_i^{k+1} = \frac{1}{\sum_j^J \prod_{l \neq i} (A_l^k b_l) \exp(\sum_{i=r}^I a_{ij} w_i - 1)} \tag{2.38}$$

Aprovechando su estructura para realizar iteraciones de punto fijo en los parámetros. Las ventajas del empleo de este método son señaladas por autores como Gonçalves y de Cursi (2001).

**Método de Newton**

Erlander (1981) estudia la optimización de problemas lineales con restricciones de entropía y propone la utilización del método de Newton sobre el problema dual. Además, demuestra la equivalencia de este tipo de problemas de optimización con el problema planteado por Wilson (2.33), y señala que el método MART puede ser visto como un método aproximado del método de Newton-Kantorovich. Eriksson (1980) propone la utilización de un método que resuelve el problema (2.1) transformandolo en un sistema no-lineal de ecuaciones sobre el cual aplica el método del gradiente conjugado.

El método de Newton ( ver Apéndice A ) es aplicado para encontrar la raiz de una ecuación, ya sea escalar o vectorial. En este caso resulta útil considerar la aplicación sobre el sistema de ecuaciones definido por las funciones $[h_i(w)]$, cuya solución para un set de parámetros $[w_i^*]$ corresponde a la solución del problema de máxima entropía. Para la iteración del método de Newton es necesario conocer la matriz Jacobiana asociada a las ecuaciónes (2.10).

Esta matriz Jacobiana esta dada por:

$$J(w) \equiv (\frac{\partial h_i(w)}{\partial w_k}) \tag{2.39}$$

Donde :

$$\frac{\partial h_i(w)}{\partial w_k} = \sum_{j=1}^{J} a_{ij} x_j a_{ij} \text{ para } i = 1, \ldots I, \text{ y } k \leq J \tag{2.40}$$

Una aplicación del algoritmo de Newton se detalla a continuación:

Algoritmo de Newton

1. Sea un vector inicial de soluciones duales $[w_i^{(0)}] \in \Re$ y un nivel de tolerancia $\epsilon_w$ suficientemente pequeño. Luego se tendrá mediante la ecuación (2.9a):

$$x_j^{(0)} = \exp(\sum_{i=1}^{I} a_{ij} w_i^{(0)} - 1) \tag{2.41}$$

2. Encontrar la dirección de Newton dada por el vector $d^{(k)}$, que es solución de:

$$Ax^{(k)}Ad^{(k)} = -(Ax^{(k)} - b) \tag{2.42}$$

3. Actualizar el vector $[w_i^{(k)}]$ mediante $w_i^{k+1} = w_i^k + d_i^k$

4. Si el conjunto de ecuaciones de primer orden $h_i(w^{(k)})$ se cumple con una tolerancia $\epsilon_w$, entonces se procede a calcular las variables primales $x_i$ finales, si no se cumple entonces se actualiza el contador de iteraciones $k = k + 1$ y se regresa al punto 2.

**Método de Bregman**

Uno de los algoritmos empleados comunmente para resolver problemas de entropía fue planteado por Bregman para problemas de optimización no lineales. El método propuesto permite resolver problemas de optimización en forma general, y en partícular puede aplicarse al problema de máxima entropía. Diversos casos de este tipo de problemas se presentan en los casos de problemas de distribución y asignación conjunta, como por ejemplo los estudiados por Bar-Gera (2006) . Lamond y Stewart (1981) muestran que muchos de los métodos de balanceo descubiertos de manera independiente son de echo, casos especiales del método de balance de Bregman.

Para entender el método debemos considerar que de la ecuación (2.10) podemos encontrar un set de parámetros duales $w_i^*$ que mediante la ecuación (2.9a) nos entrega el conjunto $[x_j^*]$ que es solución del problema primal. En cada paso el objetivo del método de Bregman es minimizar la función dual $d(w_i)$, definida en (2.13), respecto a sólo una componente del vector de variables duales $[w_i]$. Explicitamente, el algoritmo de Bregman comienza con un vector de variables duales $[w_i^{(0)}]$ y en cada paso k-ésimo resuelve una de las ecuaciónes (2.10) con respecto a solo una variable $w_k^{(k)}$.

El algoritmo paso a paso del método de Bregman puede describirse, considerando una variable de actualización $\lambda$ para la componente a actualizar del vector $[w_i^{(k)}]$como sigue:

Algoritmo de Bregman

1. Sea un vector inicial de soluciones duales $[w_i^{(0)}] \in \Re$ y un nivel de tolerancia $\epsilon_w$ suficientemente pequeño. Luego se tendrá mediante la ecuación (2.9a):

$$x_j^{(0)} = \exp(\sum_{i=1}^{I} a_{ij} w_i^{(0)} - 1) \tag{2.43}$$

2. Fijar $i = k\text{mod}\ (NM+N+M)+1$, donde la función $k\text{mod}\ x$ recorre las k restricciones cíclicamente, esta función se define como :

$$k\text{mod}m = k - m\left[\frac{k}{m}\right] \tag{2.44}$$

Luego, para la restricción i-ésima se debe encontrar la solución $\lambda$, que actualiza la componente i-esima del vector $[w_i^{(0)}]$. El cálculo del parámetro $\lambda$ se obtiene al encontrar la raiz de una de las $J$ restricciones dadas por $h(w) = 0$, ecuación ((2.10)), para ello se define la ecuación$\Phi^{(i)}(\lambda)$ como sigue:

$$\begin{aligned}\Phi^{(k)}(\lambda) &= \sum_{j=1}^{J} a_{ij} \exp(\sum_{k=1}^{I} a_{kj}w_k + a_{ij}\lambda - 1) - b_i \\ &= \sum_{j=1}^{J} a_{ij}x_i^{(k)} \exp(a_{ij}\lambda) - b_i \end{aligned} \tag{2.45}$$

3. Actualizar la componente $i$-ésima del vector $w^k$ obteniendo $w_i^{k+1} = w_i^k + \lambda^k$

4. Si el conjunto de ecuaciones de primer orden $h_i(w^{(k)})$ se cumple con una tolerancia $\epsilon_w$, entonces se procede a calcular las variables primales $x_i$ finales, si no se cumple entonces se actualiza el contador de iteraciones $k = k + 1$ y se regresa al punto 2.

El algoritmo de Bregman converge a una solución única $x^*$, cuya demostración puede verse en Fang y Tsao (1995). Una manera de encontrar la raiz de cada ecuación $\Phi(\lambda)$ es mediante el método de Newton [1], como se esquematiza en la siguiente figura:

[1]Debe considerarse que esta no es la única manera de encontrar la raiz de la ecuación

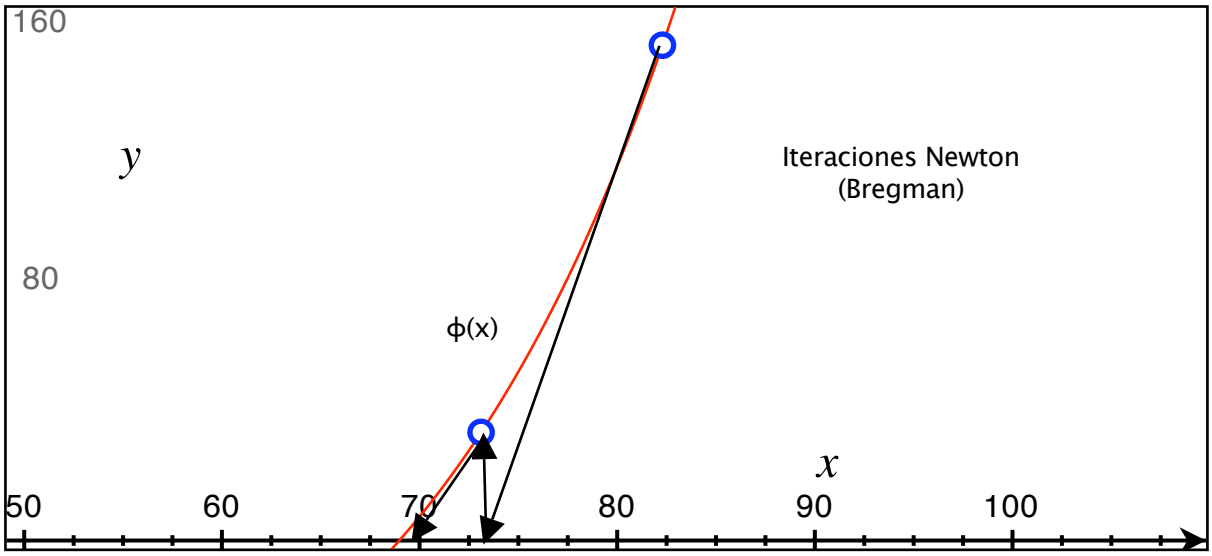


Figura 2.1: Ejemplos de Iteraciones de Newton para resolver $\Phi(\lambda) = 0$

En ella se puede apreciar como el método se aproxima, en cada iteracion (representada por una flecha), a la raiz de la función $\Phi(\lambda)$. La función $\Phi(\lambda)$ en la figura nos permite apreciar que para cada restricción i, existe una única raiz de $\Phi(\lambda)$.

**Método MART**

Otro método para calibrar problemas de entropía, es el llamado MART[2] desarrollado para el problema de reconstrucción de imágenes. Este método es estudiado por Lamond y Stewart (1981) para aplicaciones en transporte.

Pese a ser bastante similar al método propuesto por Bregman, pues también busca encontrar un set de parámetros $[w_i]$ óptimos, actualizando sólo una componente por iteración, presenta diferencias en la solución de cada variable dual. En lugar de resolver una ecuación $\Phi(\lambda)$, especifica una forma fija para el cambio en la variable. De esta manera se elimina la necesidad de resolver la búsqueda en una dimensión.

En cada iteración, la actualización de la variable i-ésima $w_i^{(k)}$ esta dada por:

$$\lambda \equiv \ln(\frac{b_i}{A_i x^{(k)}}) \tag{2.46}$$

Donde $A_i$ representa la iésima fila de la matriz $A$.

[2]Por sus siglas en inglés Multiplicative Algebraic Reconstruction Technique

Como puede verse gráficamente, el método MART aproxima en cada paso la solución de la función $\Phi(\lambda)$ definida en (2.45) de la siguiente manera:

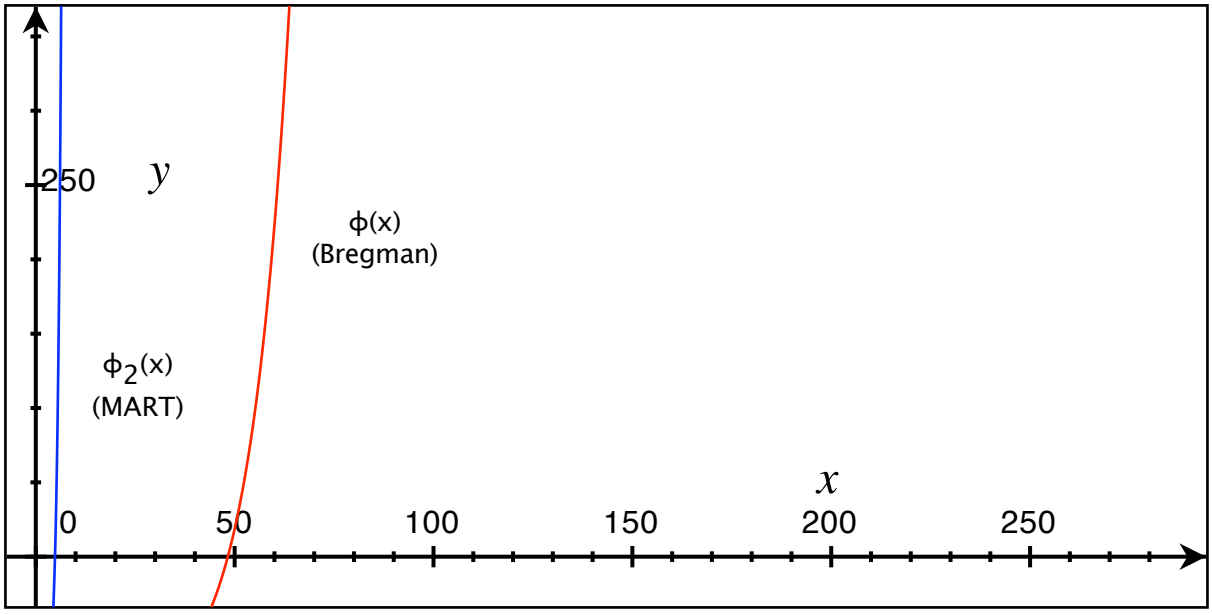


Figura 2.2: Ejemplos de Funciones de Bregman y aproximación MART

En la figura se puede apreciár la función $\Phi(x)$ que resuelve Bregman en cada iteración junto con la función $\Phi_2(x)$ donde:

$$\Phi_2(\lambda) = \sum_{j=1}^{J} a_{ij} x_i^{(k)} \exp(\lambda) - b_i \tag{2.47}$$

En ella se puede apreciar que la aproximación que realiza el método de MART coincidiría si $a_{ij} = 1$, sin embargo para los casos normales, donde $a_{ij} \neq 1$ el valor estimado no coincide con una aproximación de la raiz.

A continuación se detalla el algoritmo paso a paso:
Algoritmo MART

1. Sea un vector inicial de soluciones duales $[w_i^{(0)}] \in \Re$ y un nivel de tolerancia $\epsilon_w$ suficientemente pequeño. Luego se tendrá mediante la ecuación (2.9a):

$$x_j^{(0)} = \exp(\sum_{i=1}^{I} a_{ij} w_i^{(0)} - 1) \tag{2.48}$$

2. Fijar $i = (k\text{mod } nM+n+M)+1$, donde la función $k\text{mod } x$, definida en (2.44), recorre las k restricciones cíclicamente. Calcular lambda mediante la ecuación (2.46)

3. Actualizar la componente $i$-ésima del vector $w^k$ obteniendo $w_i^{k+1} = w_i^k + \lambda^k$

4. Si el conjunto de ecuaciones de primer orden $h_i(w^{(k)})$ se cumple con una tolerancia $\epsilon_w$, entonces se procede a calcular las variables primales $x_i$ finales, si no se cumple entonces se actualiza el contador de iteraciones $k = k + 1$ y se regresa al punto 2.

Se puede observar que el método MART sigue la misma estrategia del método de Bregman. Es más, si se considera la matriz A con todos sus elementos iguales a 1, el algoritmo de Bregman y MART generan la misma secuencia de puntos $[x_i^{(k)}]$. Este método presenta convergencia probada para las situaciones en que $0 \leq a_{ij} \leq 1$, según señala Fang y Tsao (1995). Además en la literatura se puede encontrar una variación heurística de este método propuesta por Wu (1997).

**Método de Hyman**

Otro enfoque para la resolución del problema de calibración fue señalado por Hyman (1969). Este método fue propuesto para la calibración del problema de distribución entrópico propuesto por Wilson (ver 2.33), en el cual se emplea una aproximación mediante el método de la secante para encontrar el valor de la variable dual presente en el problema y asociado a la restricción de costos del problema 2.33.

El método de la secante corresponde al siguiente:

Sean dos puntos $x_1$ y $x_2$ dos puntos iniciales en el dominio de una función $f(x)$, a la cual se le busca una raiz. El punto $x_{n+1}$ se estima mediante el uso de los puntos $x_n$ y $x_{n-1}$

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) \tag{2.49}$$

Se itera hasta encontrar $x*$ tal que $f(x^*) = 0$ bajo una tolerancia suficiente.

Sin embargo este método presenta una propiedad interesante en la formulación original, pues no realiza las iteraciones del método de la secante hasta obtener convergencia. Simplemente actualiza cada variable dual $w_i$ dando un paso en cada iteración, al igual que lo realiza el método MART.[3]

[3]Se debe considerar que la formulación original esta planteada específicamente para un problema gravitacional, que coincide en su formulación con un modelo ME como el propuesto por Wilson

De esta forma podemos considerar el método de Hyman como un algoritmo similar a MART para encontrar la raiz de $\Phi(\lambda)$, salvo que para la iteración k-ésima se resuelve un paso del método de la secante sobre $\Phi(\lambda)$, obteniendo una aproximación distinta a la del método MART ( sin llegar necesariamente a $\Phi(\lambda) = 0$).

Esto se puede ver de manera esquemática en la siguiente figura:

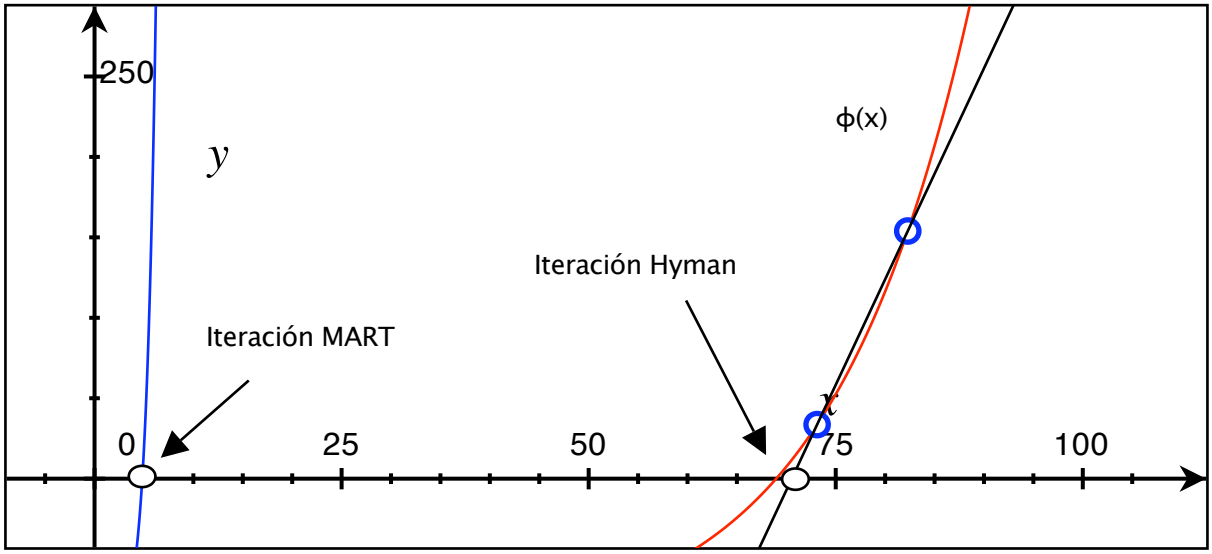


Figura 2.3: Comparación en Iteración del método de Hyman y MART

En la figura anterior, para un set de parámetros fijos, se muestra la diferencia entre el valor aproximado por el método MART y el valor aproximado por el método de Hyman. Además el empleo del método de la secante requiere de dos puntos iniciales (marcados con azul) para realizar la iteración, por lo que en cada iteración k+1 actualiza en base a puntos conocidos y sólo estima $f(x^{(k)})$.

**Otros métodos**

Existen además otros algoritmos como por ejemplo el propuesto por Fang y Tsao (1995), quienes desarrollan un método basado en búsqueda-curva (curved-search method), en el cual se destaca que todos los algoritmos antes explicado se mueven linealmente en cada iteración a lo largo de una curva, mientras que ellos proponen un método de tasa cuadrática de convergencia y lo comparan con el algoritmo de Bregman. Gonçalves y de Cursi (2001) también muestran un método de estimación de parámetros, en los cuales se reconoce que la mayor dificultad asociada al problema está dada por la restricción de costos.

Diversos métodos heurísticos pueden encontrarse en la literatura, por ejemplo Wu (1997) propone una variación heurística del método MART. Con el desarrollo de la técnica de optimización mediante algoritmos genéticos surgen también alternativas a la calibración como la presentada por Islam y Roy (2006) o métodos basados en conjuntos difusos como Leung (2007), quienes se enfocan en la resolución del problema de entropía multiobjetivo.

Capítulo 3

# DESCRIPCIÓN DEL PROBLEMA DE MÁXIMA ENTROPÍA A SIMULAR

En este capítulo se describe el modelo de maximización de la entropía sobre el cual se realizarán las distintas pruebas. En primer lugar se presenta el problema de distribución de viajes a resolver, que corresponde a una extensión del presentado por Wilson 2.33. Luego, se presentan los algoritmos empleados para la calibración del problema, aplicados específicamente al modelo descrito antes. Finalmente, se presentan dos nuevos algorimos de punto fijo para ser comparados con los anteriores.

## 3.1. Formulación del problema de máxima entropía

A continuación se presenta el problema de entropía a resolver, el cual corresponde a un problema de distribución de viajes doblemente acotado para múltiples usuarios.

Considérese el siguiente problema de maximización de la entropía, correspondiente al modelo de distribución de viajes para multiples categorías de usuarios (MEMU):

$$\max_{T} -\sum_{i}^{N}\sum_{j}^{N}\sum_{n}^{M} T_{nij}(ln(T_{nij}) - 1) \tag{3.1}$$

s.a.

$$\sum_{j=1}^{N} T_{nij} = O_{ni} \quad \forall i, n \tag{3.2a}$$

$$\sum_{n=1}^{M} \sum_{i}^{I} T_{nij} = D_j \quad \forall j = 1, \ldots, N \tag{3.2b}$$

$$\sum_{i=1}^{N} \sum_{j=1}^{N} T_{nij} c_{nij} = C_{n0} \quad \forall n = 1, \ldots, M \tag{3.2c}$$

Donde N es el total de zonas, y M es el total de categorías de usuarios.

Este es un problema cuya función objetivo es no-lineal con $N * M$ restricciones asociadas al origen (3.2a), $N$ restricciones asociadas a los destinos (3.2b) y $M$ restricciones de costos (3.2c)), todas restricciones lineales de igualdad.
El lagrangeano asociado al problema es el siguiente:

$$L = -\sum_{n,i,j} T_{nij}(ln(T_{nij}) - 1) + \sum_{i,n} \alpha_{ni}(\sum_{j} T_{nij} - O_{ni}) + \sum_{j} \theta_j(\sum_{i,n} T_{nij} - D_j) + \sum_{n} \beta_n(\sum_{i,j} T_{nij} c_{nij} - C_{n0})$$

Donde: $\alpha_{in}$ , $\theta_j$, $\beta_n$ son los multiplicadores de Lagrange asociados a las ecuaciones (3.2a),(3.2b),(3.2c).
Al resolver $\underset{T,\alpha,\theta,\beta}{\text{máx}} L$ se obtienen las condiciones de primer orden (CPO), las cuales estan dadas por:

$$-\ln(T_{nij}) + \alpha_{in} + \theta_j + \beta_n c_{nij} = 0 \quad \forall i, j, n \tag{3.3a}$$

$$\sum_{j}^{J} T_{nij} - O_{ni} = 0 \quad \forall i, n \tag{3.3b}$$

$$\sum_{i}^{I} \sum_{n}^{M} T_{nij} - D_j = 0 \quad \forall i \tag{3.3c}$$

$$\sum_{i}^{I} \sum_{j}^{J} T_{nij} c_{nij} - C_{n0} = 0 \quad \forall n \tag{3.3d}$$

Se obtiene el siguiente sistema de ecuaciones al despejar $T_{nij}$ de (3.3a) y sustituirla en las

ecuaciones (3.3b) -(3.3d):

$$\exp(\alpha_{in} + \theta_j + \beta_n c_{nij}) = T_{nij} \tag{3.4a}$$

$$\sum_j \exp(\alpha_{in} + \theta_j + \beta_n c_{nij}) = O_{ni} \tag{3.4b}$$

$$\sum_{i,n} \exp(\alpha_{in} + \theta_j + \beta_n c_{nij}) = D_j \tag{3.4c}$$

$$\sum_{i,j} \exp(\alpha_{in} + \theta_j + \beta_n c_{nij}) c_{nij} = C_{n0} \tag{3.4d}$$

Se definen los factores de balance $A_{in}$ y $B_j$, siguiendo la notación de Ortuzar y Willumsen (1994), como sigue:

$$\exp(\alpha_{in}) = A_{in} O_{ni} \quad \forall i, n \tag{3.5}$$

$$\exp(\theta_j) = B_j D_j \quad \forall j \tag{3.6}$$

Al remplazarlos en las ecuaciones (3.4b),(3.4c),(3.4d), se obtiene:

$$T_{nij} = A_{in} O_{ni} B_j D_j \exp(\beta_n c_{nij}) \tag{3.7}$$

$$A_{in} = \frac{1}{\sum_j^J B_j D_j \exp(\beta_n c_{nij})} \quad \forall i, n \tag{3.8a}$$

$$B_j = \frac{1}{\sum_{i,n}^{I,M} A_{in} O_{ni} \exp(\beta_n c_{nij})} \quad \forall j \tag{3.8b}$$

$$\sum_{i,j} A_{in} O_{ni} B_j D_j \exp(\beta_n c_{nij}) c_{nij} = C_{n0} \quad \forall n \tag{3.8c}$$

El sistema de ecuaciones determinado por las ecuaciones (3.7) a (3.8c) permite el número de viajes realizados por los usuarios de la categoría $n$, que viajan desde la zona $i$ hasta la zona $j$ , $[T_{nij}]$, que corresponden a los valores de máxima entropía. Además, podemos notar que las ecuaciones (3.8a) y (3.8b) son las que dan origen a las iteraciones tipo Furness o DSP, mientras que la ecuación (3.8c) es la señalada por diversos autores como la de dificil solución (ver por ejemplo Gonçalves y de Cursi, 2001).

## 3.2. Métodos de calibración clásicos

Dentro de los principales algoritmos para la solución del sistema (3.7) se encuentran el método de Newton-Raphson sobre el sistema de primer orden, el método de Newton-Raphson

sobre el las condiciones de primer orden del sistema equivalente máxima verosimilitud, el método de Bregman, el método MART y el método de Hyman. Estos métodos han sido estudiados en la literatura como soluciones para el problema de máxima entropía planteado y serán los considerados para realizar las pruebas de contraste con los algoritmos propuestos. A continuación se presentan la descripción de los algoritmos a comparar sobre el problema MEMU.

### 3.2.1. Método de Newton-Raphson sobre condiciones de primer orden

El método de Newton-Raphson es uno de los métodos más intuitivos para resolver problemas no lineales, su aplicación para resolver problemas de entropía resulta intuitiva al considerar el conjunto de restricciones derivadas al reemplazar la ecuación (3.7) en las restricciones (3.3b), (3.3c) y (3.3d). De esta manera se genera una ecuación $F(x)$ donde $x$ es un vector que contiene los parámetros a ser calibrados. Luego, el problema consistirá en encontrar una raíz de F(x) de modo de asegurar el cumplimiento de las condiciones de primer orden. Esta raiz $x^*$ corresponderá al vector de parámetros calibrados del problema.

Para la consideración del método de Newton-Raphson, se define la siguiente función $F(x)$:

$$\left.\begin{array}{rcll} F_i & = & \displaystyle\sum_{j}^{N} A_{in}O_{ni}BjD_j \exp(\beta_n c_{nij}) - O_{ni} & \text{para } i = 1 \ldots NM \\ F_{j+NM} & = & \displaystyle\sum_{n}^{M}\sum_{i}^{N} A_{in}O_{ni}B_jD_j \exp(\beta_n c_{nij}) - D_j & \text{para } j = 1 \ldots n \\ F_{n+N(M+1)} & = & \displaystyle\sum_{i}^{N}\sum_{j}^{N} c_{nij}A_{in}O_{ni}B_jD_j \exp(\beta_n c_{nij}) - C_{n0} & \text{para } n = 1 \ldots M \end{array}\right\} \quad (3.9)$$

El vector $x$, con los parámetros a estimar, se compone como sigue:

$$x = \left\{\begin{array}{rcll} x_i = & A_{in} & \text{para } i = 1 \ldots NM \\ x_{j+NM} = & B_j & \text{para } j = 1 \ldots N \\ x_{n+N(M+1)} = & \beta_n & \text{para } n = 1 \ldots M \end{array}\right. \quad (3.10)$$

A continuación se presenta el algoritmo considerado para el problema MEMU.

Algoritmo de Newton-Raphson para el problema de máxima entropía

- Sea $x^{(0)} \in \Re^{n(M+1)+n}$ un vector inicial para los parámetros definidos por (3.10).

- Se calcula de manera iterativa

$$x^{(n+1)} = x^{(n)} + J_F^{-1}(x^{(n)})F(x^{(n)}) \tag{3.11}$$

En donde $J^{-1}$ corresponde a la inversa de la matriz Jacobiana de la función F. Luego, se itera hasta cumplir con:

$$\left\|F(x^{(n)})\right\| < \epsilon \tag{3.12}$$

con $\epsilon > 0$ suficientemente pequeño

### 3.2.2. Método de Calibración Mediante Máxima Verosimilitud

De manera análoga al método de Newton-Raphson descrito en 3.2.1, que se aplica sobre las condiciones de primer orden derivadas directamente del problema de entropía (3.1), en este caso se resuelve Newton-Rhapson sobre las condiciones de primer orden del problema de máxima verosimilitud asociado. Este método ha sido estudiado por diversos autores, como por ejemplo Sen (1986) y Yun y Sen (1994).

El método busca encontrar los parámetros $\alpha_{in}, \theta_j, \beta_n$ que maximicen la verosimilitud de una muestra, para ello definimos la función de probabilidad $p_{nij}$ de la siguiente manera:

$$p_{nij}(\alpha, \theta, \beta) = \frac{\exp(\alpha_{in} + \theta_j + \beta_n c_{nij})}{\sum_{n,i,j} \exp(\alpha_{in} + \theta_j + \beta_n c_{nij})} \tag{3.13}$$

Podemos calibrar el problema de máxima verosimilitud empleando el método de Newton-Raphson sobre el sistema de ecuaciones de primer orden asociado al problema de máxima verosimilitud, para ello debemos considerar una función $\overline{F}$ definida como sigue:

$$\left.\begin{array}{lcll} \overline{F}_i & = & \sum_{j}^{N} p_{nij}(\alpha, \theta, \beta) - \frac{O_{ni}}{T_n^0} & \text{para } i = 1 \ldots NM \\ \overline{F}_{j+NM} & = & \sum_{n}^{M}\sum_{i}^{N} p_{nij}(\alpha, \theta, \beta) - \frac{D_j}{T_n^0} & \text{para } j = 1 \ldots N \\ \overline{F}_{n+N(M+1)} & = & \sum_{i}^{N}\sum_{j}^{N} c_{nij} p_{nij}(\alpha, \theta, \beta) - \frac{C_{n0}}{T_n^0} & \text{para } n = 1 \ldots M \end{array}\right\} \tag{3.14}$$

En donde $T_n^0 = \sum_{i,j} T_{nij}^0$ obteniéndose un vector de $M$ componentes. Luego, el método de Newton buscará una raiz de F. La solución será el vector de parámetros que satisfagan las condiciones de primer orden y por lo tanto, serán la solución óptima buscada.

El algoritmo a considerar es el siguiente:

Algoritmo de Newton-Raphson para el problema de máxima verosimilitud

- Sea $x_0 \in \Re^{N(M+1)+N}$ un vector inicial para los parámetros duales

- Se calcula

$$x_{n+1} = x_n + J_{\overline{F}}^{-1}(x_n)\overline{F}(x_n) \tag{3.15}$$

Donde $J_{\overline{F}}^{-1}$ corresponde a la matriz Jacobiana de la función $\overline{F}$. Se continua iterando hasta cumplir con:

$$\|F(x_{n+1})\| < \epsilon \tag{3.16}$$

para $\epsilon > 0$ suficientemente pequeño.

### 3.2.3. Método de Punto Fijo y Newton-Raphson (PF-N)

Este método surge de manera natural al considerar que las condiciones de primer orden (3.8a) y (3.8b) pueden resolverse mediante el método de punto fijo de Furness. Tal como señalara Gonçalves y de Cursi (2001), este método de punto fijo es ampliamente utilizado para resolver problemas de entropía sin la restricción lineal de costos, principalmente por su fácil implementación y robustéz.

Las ecuaciones empleadas para los puntos fijos son las descritas en (3.8a) y (3.8b), y la aplicación del método de Newton-Rhapson se realiza sobre la ecuación (3.8c), para lo cual se define la siguiente función sobre la variable dual $\beta_n$ y paramétrica en $A_{in}$ y$B_j$:

$$g_{A_{in},B_j}(\beta_n) = \sum_{i,j} A_{in}O_{ni}B_jD_j \exp(\beta_n c_{nij})c_{nij} - C_{n0} \tag{3.17}$$

Algoritmo de Punto Fijo y Newton (PF-N)

1. Sea $(B_j)^{(0)} \in \Re^{nM}$ un vector inicial con factores de balances y $\beta_n^{(0)}$ un vector incial de multiplicadores de Lagrange asociados a las restricciones de costo y $\epsilon$ positivo suficientemente pequeño. Se fija $k = 0$ contador de las iteraciones.

2. En base a la ecuación (3.8b) y al vector $[B_j]^{(k)}$ podemos encontrar un vector $[A_{in}]^{(k+1)}$ con la relación:

$$A_{in}^{(k+1)} = \frac{1}{\sum_{j}^{J} B_j^{(k)} D_j \exp(\beta_n^{(k)} c_{nij})}$$

3. Con el vector $[A_{in}]^{(k+1)}$ se calcula $[B_j]^{(k+1)}$ mediante la relación:

$$B_j^{(k+1)} = \frac{1}{\sum_{i}^{I} A_{in}^{(k+1)} O_{ni} \exp(\beta_n^{(k)} c_{nij})}$$

4. Con los valores de los factores de balance dados, se busca la $\beta_n^{(k+1)}$ raiz de la función $g_{A_{in},B_j}(\beta_n)$ definida en (3.17) mediante el método de Newton-Raphson, como sigue:

$$\beta_n^{(k+1)} = \beta_n^{(k)} + J^{-1}_{g_{A_{in}^{(k+1)},B_j^{(k+1)}}}(\beta_n^{(k)}) g_{A_{in}^{(k+1)},B_j^{(k+1)}}(\beta_n^{(k)}) \tag{3.18}$$

Donde $J^{-1}_{g_{A_{in}^{(k+1)},B_j^{(k+1)}}}$ corresponde a la matriz Jacobiana de la función $g_{A_{in}^{(k+1)},B_j^{(k+1)}}$. Se continua iterando hasta cumplir con:

$$\left\| g_{A_{in}^{(k+1)},B_j^{(k+1)}}(\beta_{n+1}) \right\| < \epsilon \tag{3.19}$$

### 3.2.4. Método de Bregman

El método de Bregman trabaja sobre el sistema de ecuaciones definido por las ecuaciones (3.4b),(3.4c) y (3.4d), las cuales corresponden a las variables duales derivadas de las condiciones de primer orden.

En este caso nuestro vector de parámetros $x_j$ corresponderá al total de viajes con origen $i$ y destino $j$ para el usuario $n$, es decir$T_{nij}$. Podemos notar que los parámetros de la matriz $a_{ij}$ corresponderán a los siguientes valores:

$$a_{ij} = \begin{cases} 1 & \text{para } i = 1 \dots NM \\ 1 & \text{para } j = 1 \dots N \\ c_{nij} & \text{para } n = 1 \dots M \end{cases} \tag{3.20}$$

Mientras que los valores de $b_i$ quedarán dados por:

$$b_i = \begin{cases} O_{in} & \text{para } i = 1 \dots NM \\ D_j & \text{para } j = 1 \dots N \\ C_{n0} & \text{para } n = 1 \dots M \end{cases} \tag{3.21}$$

La solución de la ecuaciones para $\lambda^{(k)}$ resulta directo para las primeras $N + NM$ restricciones, cuyo valor queda dado por:

$$\lambda^{(k)} = \begin{cases} \ln\left(\dfrac{O_{ni}}{\sum_j \exp(\theta_j^{(k)} + \beta_n^{(k)} c_{nij})}\right) & \text{para } i = 1 \dots NM \\ \ln\left(\dfrac{D_j}{\sum_{n,i} \exp(\alpha_{i,n}^{(k)} + \beta_n c_{nij})}\right) & \text{para } j = 1 \dots N \end{cases} \tag{3.22}$$

Se debe notar que las ecuaciones (3.22) coinciden con las ecuaciones iterativas de los factores de balance. Mientras que para el caso de la restricciones asociadas a los costos, se resuelven

mediante Newton escalar sobre la siguiente formulación:

$$F_{Bregman} = \sum_{i,j} \exp(\alpha_{i,n}^{(k)} + \theta_j^{(k)} + (\beta_n + \lambda^{(k)})c_{nij}) - C_{0n} \quad \text{para } n = 1 \dots M \tag{3.23}$$

Con los valores de $\lambda^{(k)}$ definidos para cada caso, el algoritmo de Bregman opera según se describe en el capítulo 2, sección 7. Para ello actualiza la componente i-ésima del vector de variables duales dado por:

$$x = [\alpha_{in}, \theta_j, \beta_n] \tag{3.24}$$

Algoritmo Bregman

1. Sea un vector inicial de soluciones duales $w^0 \in \Re^{NM+N+M}$ y un nivel de tolerancia $\epsilon_w$ suficientemente pequeño

2. Fijar $i = k \text{mod } (NM + N + M) + 1$ y encontrar la solución $\lambda^{(k)}$ según (3.22) para las restricciones de orígen y destino, $i = 1, \dots, NM + N$. Para las restricciónes de costos, $i = NM+N+1, \dots, NM+N+M$, se procede resolviendo mediante Newton la ecuación (3.23)

3. Actualizar la componente $i$-ésima del vector $w^k$ obteniendo $w_i^{(k+1)} = w_i^{(k)} + \lambda^{(k)}$

4. Si el conjunto de ecuaciones de primer orden (ecuaciónes 3.4d 3.4c 3.4b ) se cumple con norma euclidiana igual a una tolerancia $\epsilon_k$, entonces se procede a calcular los factores de balance, dados los multiplicadores de Lagrange finales según las ecuaciones (3.5), si no se cumple entonces se actualiza el contador de iteraciones $k = k + 1$ y se regresa al punto 2.

### 3.2.5. Método MART

El algoritmo MART implementado considera las mismas soluciones para las restricciones de los orígenes y los destinos que el método de Bregman, lo cual proviene de su formulación, debido a que para las restricciónes de orígen y destino $a_{ij} = 1$. Para las restricciones de costos se procede como sigue:

Para la actualización de una de las componentes de $\beta_n$ asociadas a una categoría $\widetilde{n}$

$$\lambda_n^{(k)} = \frac{C_{n0}}{\sum_{i,j} T_{nij}^{(k)} * C_{nij}} \tag{3.25}$$

En donde $T_{nij}^{(k)}$ y $C_{nij}$ corresponden a las submatrices asociadas a la categoría $n$. Con ello se tiene la siguiente estructura para la ejecución del algoritmo:

$$\lambda^{(k)} = \begin{cases} \ln\left(\dfrac{O_{ni}}{\sum_j \exp(\theta_j^{(k)} + \beta_n^{(k)} c_{nij})}\right) & \text{para } i = 1 \dots NM \\ \ln\left(\dfrac{D_j}{\sum_{n,i} \exp(\alpha_{i,n}^{(k)} + \beta_n c_{nij})}\right) & \text{para } j = 1 \dots N \\ \dfrac{C_{n0}}{\sum_{i,j} \exp(\alpha_{i,n}^{(k)} + \theta_j^{(k)} + \beta_n C_{nij}) C_{nij}} & \text{para } n = 1 \dots M \end{cases} \tag{3.26}$$

Algoritmo MART

1. Sea un vector inicial de soluciones duales $w^0 \in \Re^{nM+n+M}$ y un nivel de tolerancia $\epsilon_w$ suficientemente pequeño

2. Fijar $i = k \text{mod } (NM + N + M) + 1$, con ello i recorrerá cíclicamente todas las restricciones. Luego, encontrar la solución $\lambda^{(k)}$ según (3.26)

3. Actualizar la componente $i$-ésima del vector $w^k$ obteniendo $w_i^{(k+1)} = w_i^{(k)} + \lambda^{(k)}$

4. Si el conjunto de ecuaciones de primer orden (ecuaciónes 3.4d 3.4c 3.4b ) se cumple con norma euclidiana igual a una tolerancia $\epsilon_k$, entonces se procede a calcular los factores de balance, dados los multiplicadores de Lagrange finales según las ecuaciones (3.5), si no se cumple entonces se actualiza el contador de iteraciones $k = k + 1$ y se regresa al punto 2.

### 3.2.6. Método de Hyman

El método de Hyman corresponde a un método similaral método de Punto Fijo y Newton ( ver 3.2.3 ), pues al igual que el método anterior, realiza la estimación de los factores de balance mediante iteraciones de punto fijo. El problema que resulta encontrar el valor del vector $\beta_n$ no es resuelto empleando Newton, sino que resuelve aplicando el método de la secante un paso por cada iteración.

El algoritmo de hyman se plantea sobre la restricción de costos bajo la siguiente modificación: Sea $\bar{C_{0n}}$ los costos medios totales para la categoría n definidos por:

$$\bar{C}_{0n} = \frac{\sum_{ij} C_{0n} T_{ijn}}{\sum_{ij} T_{ijn}} \tag{3.27}$$

y además cada costo es transformado a un costo sobre el total de viajes para la categoría, de esta forma se tiene:

$$\bar{c_{ijn}} = \frac{c_{ijn}}{\sum_{ij} T_{ijn}} \tag{3.28}$$

Luego con el valor de $T_{ijn}^{(k)} = A_{in}^{(k)} O_{in} B_j^{(k)} D_j exp(\beta_n^{(k)})$ al finalizar la iteración k-ésima se podrá calcular:

$$C^{\bar{(k)}} = \frac{\sum_{ij} c_{ijn} T_{ijn}^{(k)}}{\sum_{ij} T_{ijn}^{(k)}} = \sum_{ij} c_{ijn}^{(\bar{k})} T_{ijn}^{(k)} \tag{3.29}$$

Con ello si se considera la restricción de costos, derivada de (3.8c), para una categoría de usuario n, se podrá definir la función $f(\beta_n)$, cuya componente i-ésima se define por:

$$f(\beta_n)_i = \sum_{i,j} A_{in} O_{ni} B_j D_j \exp(\beta_i c_{nij}) \bar{c_{ijn}} - \bar{C}_{0n} \tag{3.30}$$

Luego para cada valor de $f(\beta_n)$ se tendrá una aproximación del total de los costos medios, con ello podemos escribir:

$$C^{\bar{(k)}} = f(\beta_n^{(k)}) + \bar{C}_{0n} \tag{3.31}$$

Luego reemplazando $f$ en el paso genérico del método de la secante se tendrá:

$$\begin{aligned}
\beta_n^{(k+1)} &= \beta_n^{(k)} - \frac{\beta_n^{(k)} - \beta_n^{(k-1)}}{f(\beta_n^{(k)}) - f(\beta_n^{(k-1)})} f(\beta_n^{(k)}) && (3.32)\\
&= \beta_n^{(k)} - \frac{\beta_n^{(k)} - \beta_n^{(k-1)}}{\bar{C}_n^{(k)} - \bar{C}_n^{(k-1)}} (\bar{C}_n^{(k)} - \bar{C}_{0n}) && (3.33)\\
&= \frac{\beta_n^{(k)} \bar{C}_n^{(k)} - \beta_n^{(k)} \bar{C}_n^{(k-1)} - \beta_n^{(k)} \bar{C}_n^{(k)} + \beta_n^{(k-1)} \bar{C}_n^{(k)} + \beta_n^{(k)} \bar{C}_{0n} - \beta_n^{(k-1)} \bar{C}_{0n}}{\bar{C}_n^{(k)} - \bar{C}_n^{(k-1)}} && (3.34)\\
&= \frac{(\bar{C}_{0n} - \bar{C}_n^{(k-1)}) \beta_n^{(k)} - (\bar{C}_{0n} - \bar{C}_n^{(k)}) \beta_n^{(k-1)}}{\bar{C}_n^{(k)} - \bar{C}_n^{(k-1)}} && (3.35)
\end{aligned}$$

La expresión (3.35) corresponde a la actualización para cada componente del vector $\beta_n$ que realiza el método de Hyman.

La actualización de cada componente de $\beta_n$ puede verse en la siguiente figura:

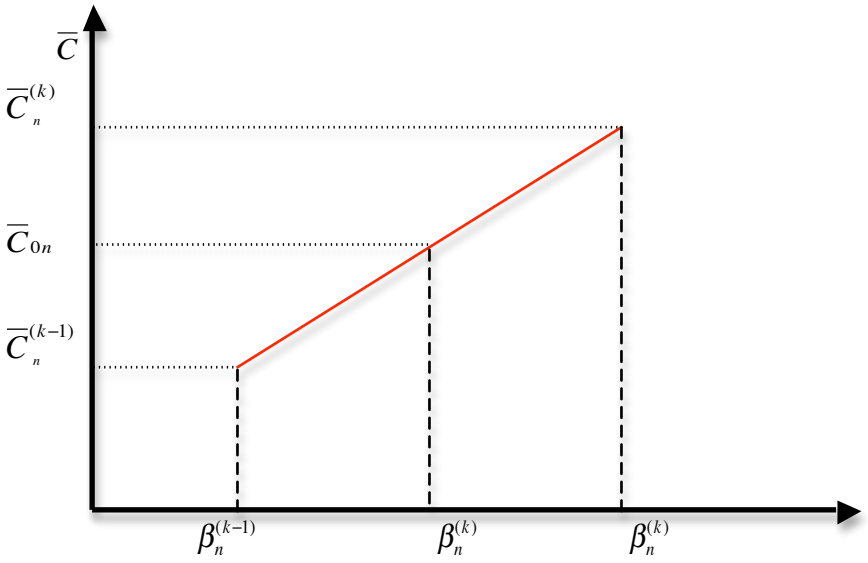


Figura 3.1: Representación Gráfica del método de Hyman

Gráficamente, corresponde a encontrar el valor de x en la recta que pasa por los puntos $(\beta^{(k-1)}, C^{(k-1)})$ y $(\beta^{(k)}, C^{(k)})$, tal que $y(x) = C_{0n}$

Algoritmo de Hyman

1. Sea $(B_j)^{(0)} \in \Re^{nM}$ un vector inicial con factores de balances y $\beta_n^{(0)}$ un vector incial de multiplicadores de Lagrange asociados a las restricciones de costo y $\epsilon$ positivo suficientemente pequeño. El valor inicial de $\beta_n^{(0)}$ sugerido es:

$$\beta_n^{(0)} = \frac{3}{2\bar{C}_n} \tag{3.36}$$

   Se fija $k = 0$ contador de las iteraciones.

2. En base a la ecuación (3.8b) y al vector $[B_j]^{(k)}$ podemos encontrar un vector $[A_{in}]^{(k+1)}$ con la relación:

$$A_{in}^{(k+1)} = \frac{1}{\sum_j^J B_j^{(k)} D_j \exp(\beta_n^{(k)} c_{nij})}$$

3. Con el vector $[A_{in}]^{(k+1)}$ se calcula $[B_j]^{(k+1)}$ mediante la relación:

$$B_j^{(k+1)} = \frac{1}{\sum_i^I A_{in}^{(k+1)} O_{ni} \exp(\beta_n^{(k)} c_{nij})}$$

4. Con los valores de los factores de balance dados se procede a encontrar un nuevo valor de $\beta_n$

   - si k=1, es decir, es la primera iteración. Calcular $\beta_n^{(k)}$ según:

   $$\beta_n^{(k)} = \frac{\bar{C_{0n}}\beta_n^{(0)}}{\bar{C}_n^{(1)}} \tag{3.37}$$

   - Si k>1, se procede a actualizar $\beta_n$ según (3.35)

5. Si el conjunto de ecuaciones de primer orden se cumple una tolerancia $\epsilon$ se termina. Si no se cumplen, entonces se actualiza el contador de iteraciones $k = k + 1$ y se regresa al punto 2.

## 3.3. Algoritmos Nuevos

En esta sección se presentan los dos nuevos algoritmos propuestos para ser constrastados con los mostrados en la sección anterior. Ambos métodos pertenecen a la familia de algoritmos de punto fijo, los cuales son implementados de la misma manera para los factores de balance asociados a las restricciones de Origen y Destino, es decir, la construcción de los parámetros $A_{in}$ y $B_j$ es realizada de manera iterativa, con el método conocido como equilibrio de matrices o balanceo de matrices para ambos algoritmos. Sin embargo difieren en la estimación de los parámetros $\beta_n$ asociados a las restricciones de Costo, mientras que el primer algoritmo corresponde a un método iterativo basado en un pivote, el segundo se basa en un cambio de variable del parámetro $\beta_n$ por una variable acotada y resuelve empleando Newton-Raphson

### 3.3.1. Método de Pivote

A continuación se presenta el primer algoritmo propuesto denominado de premultiplicación. Se presenta una deducción del algoritmo y posteriormente se formulan las distintas etapas del mismo.

#### Deducción del método de pivote

Dado el sistema de ecuaciones que conforman el problema de maximización de la entropía, se realizará la deducción para el caso de un individuo $n$. En particular para la deducción del algoritmo de modo de simplificar la notación con lo cual el subindice $n$ será omitido. Este método se extiende a la formulación con múltiples clases de usuarios.
Se puede escribir la restricción de costos del problema de máxima entropía como sigue:

$$\sum_{ij} C_{ij}T_{ij}(\beta) - C_0 = 0 \tag{3.38}$$

Dada la forma genérica de solución del problema $T_{ij} = A_i O_i B_j D_j e^{\beta C_{ij}}$, al reemplazar en (3.38), se tendrá:

$$\sum_{ij} C_{ij} A_i O_i B_j D_j e^{\beta C_{ij}} - C_0 = 0 \tag{3.39}$$

considerando que los costos $C_{ij}$, para cada categoría de usuarios, pueden ser escritos como

$$C_{ij} = \widetilde{C} + \Delta C_{ij} \tag{3.40}$$

donde $\widetilde{C} \in \Re$ es un valor cualquiera. Reemplazando en (3.39) se tendrá la siguiente ecuación:

$$\sum_{ij} C_{ij} A_i O_i B_j D_j e^{\beta \widetilde{C}} e^{\beta \Delta C_{ij}} - C_0 = 0 \tag{3.41}$$

Desde la cual se puede despejar $\beta$:

$$e^{\beta \widetilde{C}} \sum_{ij} C_{ij} A_i O_i B_j D_j e^{\beta \Delta C_{ij}} - C_0 = 0 \tag{3.42}$$

De esta forma, se obtiene la siguiente expresión:

$$\beta = \frac{1}{\widetilde{C}} \ln\left(\frac{C_0}{\sum_{ij} C_{ij} A_i O_i B_j D_j e^{\beta \Delta C_{ij}}}\right) \tag{3.43}$$

Con ello se obtiene una ecuación de punto fijo en beta, la cual es además paramétrica en la elección de la variable $\widetilde{C}$ Para el caso general, con múltiples categorías de usuario, el método se extiende para cada componente $n$-ésima del vector $\beta_n$ como sigue:

$$[\beta_n]_n = \frac{1}{[\widetilde{C_n}]_n} \ln\left(\frac{[C_{n0}]_n}{\sum_{ij} C_{nij} A_{in} O_{in} B_j D_j e^{[\beta_n]_n \Delta C_{nij}}}\right) \tag{3.44}$$

En donde $[X]_i$ representa la componente i del vector $X$.

Si para la elección de $\widetilde{C_n}$ , para cada clase de usuario $C^n_{max} = \text{máx}_{ij}\, C_{nij}$ y $C^n_{min} = \text{mín}_{ij}\, C_{nij}$ podemos parametrizar el algoritmo en $\lambda$ mediante la siguiente combinación convexa:

$$\widetilde{C_n} = \lambda C^n_{max} + (1 - \lambda) C^n_{min} \tag{3.45}$$

A pesar de no tener que elegir un valor para $\lambda$ en el intervalo $[0, 1]$ (pues $\widetilde{C_n} \in \Re$) no está necesariamente acotado , se elige la parametrización entre costos máximos y mínimos para trabajar con el algoritmo.

**Contractancia**

La solución de la función (3.43) puede ser encontrada mediante la utilización del teorema del punto fijo de Banach, el cual establece que una función $f(x) : \Omega \in \Re \rightarrow \Re$, contractante en $\Omega$, cerrado, presentará un único punto fijo $\overline{x} \in \Omega$ tal que:

$$\overline{x} = f(\overline{x})$$

Se probará que la función definida por:

$$f(\beta) = \frac{1}{\widetilde{C}} \ln(\frac{C_0}{\sum_{ij} C_{ij} A_i O_i B_j D_j e^{\beta \Delta C_{ij}}}) \tag{3.46}$$

es contractante, es decir que cumple con la siguiente condición:

$$|f(\beta_i) - f(\beta_j)| \leq K \, |\beta_i - \beta_j|$$

y además $0 \leq K \leq 1$.

Debido a que la función es continua, se puede probar que la función es contractante al probar que la norma de su derivada sea menor o igual a uno, en este caso, para la derivada de $f$, se tendrá la siguiente expresión:

$$\begin{aligned} \frac{d}{d\beta} f(\beta) &= \frac{1}{\widetilde{C}} \frac{\sum_{ij} A_i O_i B_j D_j e^{\beta \Delta C_{ij}}}{C_0} \frac{C_0}{(\sum_{ij} A_i O_i B_j D_j e^{\beta \Delta C_{ij}})^2} \sum_{ij} A_i O_i B_j D_j e^{\beta \Delta C_{ij}} \Delta C_{ij} \\ &= \sum_{ij} \left( \frac{A_i O_i B_j D_j e^{\beta \Delta C_{ij}}}{\sum_{ij} A_i O_i B_j D_j e^{\beta \Delta C_{ij}}} \right) \frac{\Delta C_{ij}}{\widetilde{C}} \end{aligned} \tag{3.47}$$

Luego se probará que : $\left| \frac{d}{d\beta} f(\beta) \right| \leq 1$

Prueba de contractancia

Analizaremos $\left| \frac{d}{d\beta} f(\beta) \right|$

$$\begin{aligned} \left| \frac{d}{d\beta} f(\beta) \right| &= \left| \sum_{ij} \left( \frac{A_i O_i B_j D_j e^{\beta \Delta C_{ij}}}{\sum_{ij} A_i O_i B_j D_j e^{\beta \Delta C_{ij}}} \right) \frac{\Delta C_{ij}}{\widetilde{C}} \right| \\ &< \sum_{ij} \left| \left( \frac{A_i O_i B_j D_j e^{\beta \Delta C_{ij}}}{\sum_{ij} A_i O_i B_j D_j e^{\beta \Delta C_{ij}}} \right) \frac{\Delta C_{ij}}{\widetilde{C}} \right| \\ &= \sum_{ij} \left( \frac{A_i O_i B_j D_j e^{\beta \Delta C_{ij}}}{\sum_{ij} A_i O_i B_j D_j e^{\beta \Delta C_{ij}}} \right) \left| \frac{\Delta C_{ij}}{\widetilde{C}} \right| \end{aligned} \tag{3.48}$$

Una condición suficiente para $\left|\frac{d}{d\beta}f(\beta)\right| \leq 1$ es que:

$$\left|\frac{\Delta C_{ij}}{\widetilde{C}}\right| < 1 \tag{3.49}$$

es decir

$$\begin{aligned} --1 < \quad & \tfrac{\Delta C_{ij}}{\widetilde{C}} \quad < 1 && \text{(3.50a)} \\ -\widetilde{C} < \quad & \Delta C_{ij} \quad < \widetilde{C} && \text{(3.50b)} \\ -\widetilde{C} < \quad & \widetilde{C} - C_{ij} \quad < \widetilde{C} && \text{(3.50c)} \\ 0 < \quad & C_{ij} \quad < 2\widetilde{C} && \text{(3.50d)} \end{aligned}$$

Por lo que si $\widetilde{C} > \frac{C_{ij}}{2} \quad \forall i, j$, o de manera equivalente

$$\widetilde{C} > \frac{\text{máx}_{ij}\, C_{ij}}{2} \tag{3.51}$$

entonces $f$ será contractante. Por lo anterior, para el resto del trabajo se escogerá arbitrariamente $\widetilde{C_n} = \text{máx}_{ij}\, C_{nij}$ para cada categoría $n = 1, \ldots, M$ Se define el algoritmo de Pivote como sigue:

Algoritmo de Pivote

1. Sea $A_{in}^{(0)} \in \Re^{nM}$ vector inicial con factores de balances factibles y $\epsilon$ positivo suficientemente pequeño. Se fija $k = 1$ contador global de las iteraciones.

2. En base a la ecuación (3.8b) y al vector $[A_{in}]^{(k)}$ podemos encontrar un vector $[B_j]^{(k+1)}$ con la relación:
$$B_j^{(k+1)} = \frac{1}{\sum_i^N A_{in}^{(k)} O_{ni} \exp(\beta_n^{(k)} c_{nij})}$$

3. Con el vector $[B_j]^{(k+1)}$ se calcula $[A_{in}]^{(k+1)}$ mediante la relación:
$$A_{in}^{(k+1)} = \frac{1}{\sum_j^N B_j^{(k+1)} D_j \exp(\beta_n^{(k)} c_{nij})}$$

4. Con los valores de los factores de balance estimados, se busca beta como sigue:
$$\beta_n^{(k+1)} = \frac{1}{\widetilde{C_n}} \ln\left(\frac{C_{0n}}{\sum_{ij} A_{nij}^{(k+1)} O_{ni} B_j^{k+1} D_j \exp(\beta_n^{(k)}(c_{nij} - C_{max})}\right) \tag{3.52}$$

5. Se verifica convergencia: Si el cumplimiento de las condiciones de primer orden, dado por el cumplimiento de la función $F(x)$ definida en (3.9) es con un nivel de tolerancia $\epsilon$ se termina el algoritmo, si no se aumenta el contador de iteraciones $k = k + 1$ y se regresa al punto número 1.

Se debe destacar que este método, al igual que el método de Hyman, sólo realiza una iteración del punto fijo. Es decir, no itera hasta la convergencia en cada iteración global k.

### 3.3.2. Método de Calibración Basado en Acotamiento de Solución (Búsqueda Acotada)

Este método, similar al método de Punto Fijo y Newton-Raphson, realiza la calibración de los factores de balance en base a un punto fijo y presenta una variación para el cálculo del la restricción asociada a los costos. Esta variación corresponde a un cambio de variable, que permite pasar de la estimación de un parámetro $\beta_n \in \Re$ a un parámetro $\theta_n \in [0, 1]$.

El método se fundamenta en la siguiente proposición, que caracteriza la solución de $\beta_n$

**Proposición Nº1**

Sea $C_{nMIN} = \min_{ij} \{c_{nij}\}$ y $C_{nMAX} = \max_{ij} \{c_{nij}\}$:

Si $\sum_{ijn} A_{in}O_{in}B_jD_jc_{nij} < C_{0n}$ entonces $\beta_n > 0$ y

$$\frac{1}{C_{MAX}} \ln \frac{C_{0n}}{\sum_{ij} A_{in}O_{in}B_jD_jc_{nij}} < \beta_n < \frac{1}{C_{MIN}} \ln \frac{C_{0n}}{\sum_{ij} A_{in}O_{in}B_jD_jc_{nij}} \tag{3.53}$$

Si $\sum_{ijn} A_{in}O_{in}B_jD_jc_{nij} > C_{0n}$ entonces $\beta_n < 0$ y

$$\frac{1}{C_{nMIN}} \ln \frac{C_{0n}}{\sum_{ij} A_{in}O_{in}B_jD_jc_{nij}} < \beta_n < \frac{1}{C_{nMAX}} \ln \frac{C_{0n}}{\sum_{ij} A_{in}O_{in}B_jD_jc_{nij}} \tag{3.54}$$

Luego podemos escribir

$$\beta_n = \frac{1}{c(\theta_n)} \ln \frac{C_{0n}}{\sum_{ij} A_{in}O_{in}B_jD_jc_{nij}} \tag{3.55}$$

donde $c(\theta_n) = \theta_n C_{nMIN} + (1 - \theta_n)C_{nMAX}$ con $\theta_n \in [0, 1]$ .

Demostración

Supongase que $\beta_n > 0$, entonces:

Se tendrá que:

$$\beta_n C_{nMIN} \leq \beta_n c_{nij} \quad \forall i, j \tag{3.56}$$

por lo tanto, como la función exponencial es creciente:

$$\exp(\beta_n C_{nMIN}) \leq \exp(\beta_n c_{nij}) \quad \forall i,j \tag{3.57}$$

Como $A_{in}O_{in}B_jD_j > 0$ pues son cada uno de ellos positivos, y además $c_{nij} > 0 \quad \forall i,j,n$

$$\begin{aligned} A_{in}O_{in}B_jD_j \exp(\beta_n C_{nMIN}) &\leq A_{in}O_{in}B_jD_j \exp(\beta_n c_{nij}) \quad \forall i,j,n & (3.58)\\ \sum_{ij} A_{in}O_{in}B_jD_j \exp(\beta_n C_{nMIN}) &\leq \sum_{ij} A_{in}O_{in}B_jD_j \exp(\beta_n c_{nij}) = C_{0n} & (3.59)\\ \sum_{ij} A_{in}O_{in}B_jD_j \exp(\beta_n C_{nMIN}) &\leq C_{0n} & (3.60) \end{aligned}$$

Luego :

$$\beta_n \leq \frac{1}{C_{nMIN}} \ln\left(\frac{C_{0n}}{\sum_{ij} A_{in}O_{in}B_jD_j}\right) \tag{3.61}$$

Por otra parte

$$\beta_n c_{nij} \leq \beta_n C_{nMAX} \quad \forall i,j \tag{3.62}$$

por lo tanto, como la función exponencial es creciente:

$$\exp(\beta_n c_{nij}) \leq \exp(\beta_n C_{nMAX}) \quad \forall i,j \tag{3.63}$$

Como $A_{in}O_{in}B_jD_j > 0$ pues son cada uno de ellos positivos, y además $c_{nij} > 0 \forall i,j,n$

$$\begin{aligned} A_{in}O_{in}B_jD_j \exp(\beta_n c_{nij}) &\leq A_{in}O_{in}B_jD_j \exp(\beta_n C_{nMAX}) \quad \forall i,j,n & (3.64)\\ C_{0n} = \sum_{ij} A_{in}O_{in}B_jD_j \exp(\beta_n c_{nij}) &\leq \sum_{ij} A_{in}O_{in}B_jD_j \exp(\beta_n C_{nMAX}) & (3.65)\\ C_{0n} &\leq \sum_{ij} A_{in}O_{in}B_jD_j \exp(\beta_n C_{nMAX}) & (3.66) \end{aligned}$$

Luego :

$$\frac{1}{C_{nMAX}} \ln\left(\frac{C_{0n}}{\sum_{ij} A_{in}O_{in}B_jD_j}\right) \leq \beta_n \tag{3.67}$$

Con lo que se tiene:

$$\frac{1}{C_{nMAX}} \ln\left(\frac{C_{0n}}{\sum_{ij} A_{in}O_{in}B_jD_j}\right) \leq \beta_n \leq \frac{1}{C_{nMIN}} \ln\left(\frac{C_{0n}}{\sum_{ij} A_{in}O_{in}B_jD_j}\right) \tag{3.68}$$

Si se supone $\beta_n < 0$ se tendrán los casos:

$$\beta_n c_{nij} \leq \beta_n C_{nMIN} \tag{3.69}$$

$$\beta_n C_{nMAX} \leq \beta_n c_{nij} \tag{3.70}$$

y procediendo de la misma manera anterior se tiene:

$$\frac{1}{C_{nMIN}} \ln\left(\frac{C_{0n}}{\sum_{ij} A_{in}O_{in}B_jD_j}\right) \leq \beta_n \leq \frac{1}{C_{nMAX}} \ln\left(\frac{C_{0n}}{\sum_{ij} A_{in}O_{in}B_jD_j}\right) \tag{3.71}$$

Luego podemos definir : donde $c(\theta_n) = \theta_n C_{nMIN} + (1-\theta_n)C_{nMAX}$ con $\theta_n \in [0,1]$, y se cumple que:

$$\beta_n = \frac{1}{c(\theta_n)} \ln \frac{C_{0n}}{\sum_{ij} A_{in}O_{in}B_jD_jc_{nij}} \tag{3.72}$$

A continuación se presenta el algoritmo, el cual realiza iteraciones sobre los factores de balance una vez por cada factor. Luego realiza la busqueda del valor de $\theta_n$ óptimo empleando sólamente un paso del algoritmo de Newton.

Algoritmo de Búsqueda Acotada

Se define la componente i-ésima de la función $R_{A_{in},B_j}(\theta_n)$ como sigue:

$$R_{A_{in},B_j}(\theta_n) = \sum_{ij} A_{in}O_{in}B_jD_j \left(\frac{C_{0n}}{\sum_{ij} A_{in}O_{in}B_jD_jc_{nij}}\right)^{\frac{c_{ijn}}{c(\theta)}} c_{nij} - C_0 \tag{3.73}$$

La cual proviene de realizar el cambio de variables (3.55) sobre la restricción de costos.

1. Sea $A_{in}^{(0)} \in \Re^{nM}$ vector inicial con factores de balances factibles y $\epsilon$ positivo suficientemente pequeño. Se fija $k = 1$ contador global de las iteraciones.

2. En base a la ecuación (3.8b) y al vector $[A_{in}]^{(k)}$ podemos encontrar un vector $[B_j]^{(k+1)}$ con la relación:a

$$B_j^{(k+1)} = \frac{1}{\sum_{i}^{I} A_{in}^{(k)} O_{ni} \exp(\beta_n^{(k)} c_{nij})} \quad \forall j = 1, \ldots, N$$

3. Con el vector $B_j{}^{(k+1)}$ se calcula $[A_{in}]^{(k+1)}$ mediante la relación:

$$A_{in}^{(k+1)} = \frac{1}{\sum_{j}^{J} B_j^{(k+1)} D_j \exp(\beta_n^{(k)} c_{nij})} \quad \forall i = 1, \ldots, M$$

4. Con los valores de los factores de balance dados, se busca $\theta \in [0,1]$ tal que la función $R_{A_{in},B_j}(\theta_n) = 0$ definida en (3.73) mediante el método de Newton-Raphson, dando sólamente un paso en cada iteración k.

5. Con el valor de $\theta_n$ podemos calcular $\beta_n$ mediante la relación:

$$\beta_n = \frac{1}{C(\theta_n)} \frac{C_{0n}}{\sum_{i,j} A_{in} O_{in} B_j D_j c_{nij}} \tag{3.74}$$

6. Se verifica convergencia: Si el cumplimiento de las condiciones de primer orden, dado por el cumplimiento de la función $F(x)$ definida en (3.9) es con un nivel de tolerancia $\epsilon$ se termina el algoritmo, si no $k = k + 1$ y se regresa al punto número 1.

Capítulo 4

# IMPLEMENTACIÓN COMPUTACIONAL DE ALGORITMOS

Con el objeto de comparar de los algoritmos descritos, los seis clásicos y dos nuevos, se realiza la implementación de los mismos sobre el software comercial Gauss, el cual es popular entre los programas econométricos y se destaca por su alta velocidad y eficiencia en el manejo de matrices. Se recibió una implementación básica de los algoritmos, salvo el método de Hyman, en donde cada una de las funciones se encontraba codificada en archivos separados. Sobre esta implementación se realizan calibraciones de prueba para obtener principalmente, los tiempos de ejecución de cada algoritmo. Se buscaron ineficiencias dentro de los métodos programados que ralentizaran la ejecución de los códigos, principalmente debido a que el software Gauss ejecuta instrucciones matriciales en menor tiempo que en bloques de instrucciones. En una segunda etapa los algoritmos fueron revisados principalmente para la corrección de ineficiencias detectadas y se realizará un mejoramiento completo de los algoritmos.

## 4.1. Descripción de los métodos

Para la ejecución de los algoritmos se realizó una implementación exploratoria, en esta versión se programaron los algoritmos a comparar con el objetivo de testear en una primera instancia el correcto desempeño de ellos. La estructura de esta programación se realizó en base a procedimientos maestros, correspondientes a los algoritmos a comparar, y procedimientos auxiliares, que se emplean en distintas ocasiones dentro del código principal. A continuación se presentan los algoritmos principales, se describe su funcionamiento, estructura, variables de entrada y de salida. En una segunda sección se presentan los programas auxiliares.

### 4.1.1. Programas Principales

A continuación se describen los procedimientos correspondientes a los algoritmos a comparar, se presentan también diagramas de flujos para cada algoritmo. Estos procedimientos son capaces de acceder a los valores de $[O_{in}]$, $[D_j]$ la matriz de costos para toda categoría de usuarios $[c_{nij}]$ y los parámetros $C_{0n}$, correspondientes al lado derecho de las restricciones de costos.
Se debe notar que los procedimientos para calcular las condiciones de primer orden, pese a ser siempre el mismo conjunto de restricciones, se encuentran programados en distintos métodos. Esto se implementó para poder evitar cálculos repetidos en cada procedimiento, por ello cada función que estima el error en el cumplimiento de las condiciones de primer orden será diferente según sea el procedimiento que la llame. De esta manera, cada procedimiento recibirá variables con parte del cálculo necesario para continuar sin repetir instrucciones realizadas antes.

**Newton Rhapson Sobre Condiciones de Primer Orden**

El procedimiento que realiza la calibración aplicando el método de Newton Rhapson sobre las condiciones de primer orden, es llamado NewtonCPO. Este recibe como parámetro externo la tolerancia al cumplimiento de las ecuaciones de primer orden y el vector de parámetros iniciales para las iteraciónes.

El procedimiento consiste básicamente en la búsqueda de los parámetros para los que $F(x) = 0$, donde F es la función definida en (3.9) . El método se basa en la aplicación de la función implementada en GAUSS llamada *eqSolve*, la cual resuelve sistemas no lineales de ecuaciones mediante la aplicación del método de Newton. El procedimiento *eqSolve* requiere una función auxiliar para encontrar la raiz, en este caso corresponde al procedimiento *errorcpo*, el cual recibe un vector de parámetros, y entrega la evaluación de la función F. Además el método toma el tiempo del método antes de comenzar las iteraciones y al terminarlas, de modo de obtener con precisión de centésimas de segundos el tiempo de ejecución. A continuación se describen los principales parámetros de entrada y salida del método.

1. Variables de entrada NewtonCPO:

   - Vector de dimensión $NM+N+M$: Vector de parámetros iniciales para el método, según 3.10
   - Parámetro global, escalar "$\_Tol$": Tolerancia para el término de las iteraciones (variable de la función *eqSolve*)

2. Variables de salida:

   - Vector de dimensiones $NM + N + M$ correspondiente al vector 3.10.
   - Escalar: Tiempo de ejecución del procedimiento en centésimas de segundo.

La función *errorcpo* es entregada como variable al procedimiento *eqSolve* quien se encarga de realizar las iteraciones del algoritmo Newton-Rhapson hasta cumplir con la siguiente condición:

$$\|F(x)\| \leq \_\_Tol$$

Como se puede apreciar, dicha condición es equivalente a la empleada en los otros métodos y corresponde al cumplimiento de las condiciones de primer orden bajo la condición que la norma euclidiana de la función (3.9) sea menor que la tolerancia entregada. La estimación de la matriz Jacobiana es realizada por el procedimiento eqSolve, esta también puede ser entregada como una variable al procedimiento. Se programó la matriz Jacobiana analítica, sin embargo su uso fué desestimado debido a que con ella el procedimiento presentaba una velocidad de convergencia inferior.

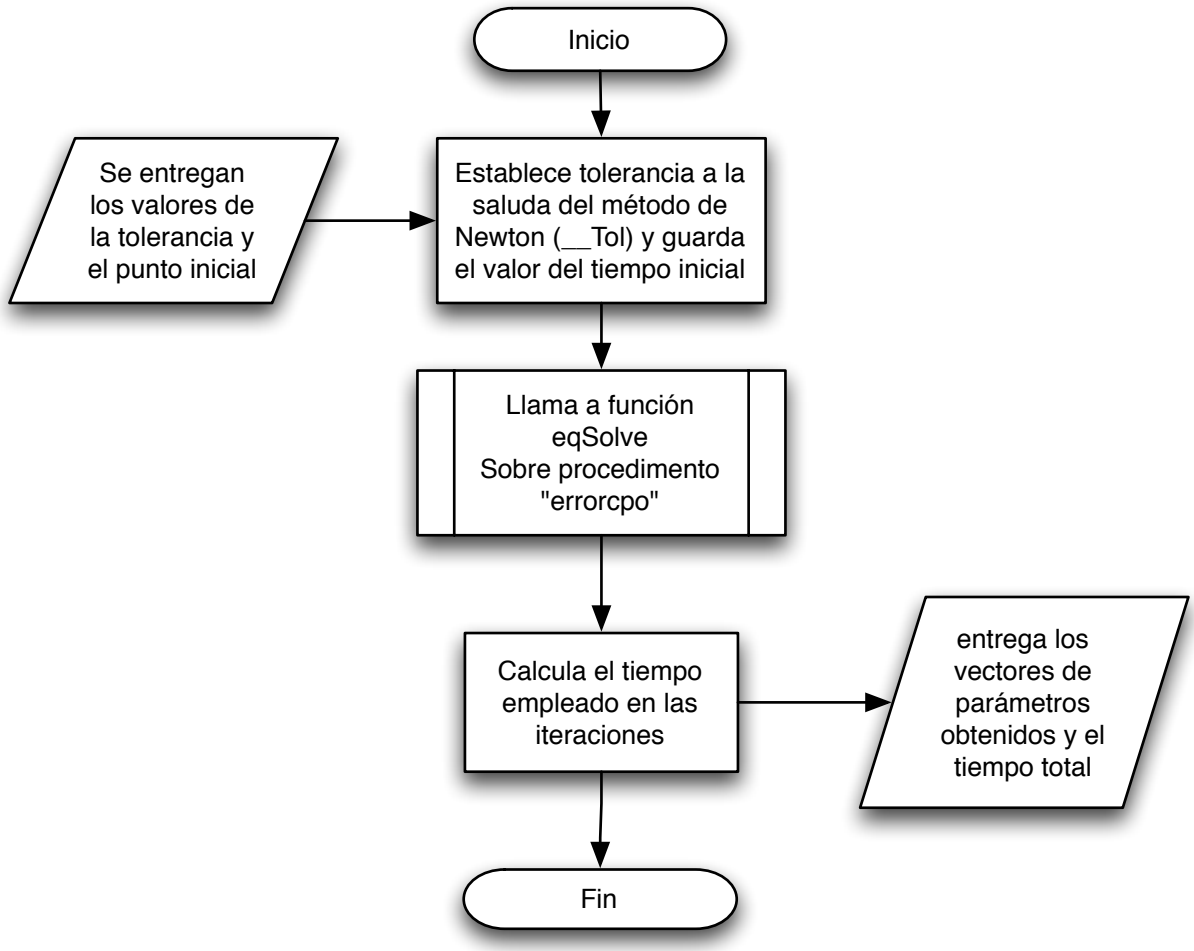


Figura 4.1: Operación del método Newton Rhapson Sobre Condiciones de Primer Orden

### Newton Rhapson sobre Máxima Verosimilitud

Este método emplea la función programada llamada *Fvero*, el cual es un procedimiento que representa la función 3.14. Su funcionamiento se basa, al igual que en el caso de Newton Rhapson sobre CPO, en la aplicación de la función *eqSolve* al procedimiento que entrega las condiciones de primer orden asociadas al problema de máxima verosimilitud. Este procedimiento es análogo al presentado antes, salvo que la raíz buscada ahora corresponde al optimo del sistema de CPO del problema de máxima verosimilitud. El procedimiento de la matriz Jacobiana del problema se encuentra implementado bajo el procedimiento *gradienteFvero* y ouede ser entregado de manera optativa.

1. Variables de entrada:

   - Vector de dimensión $NM+N+M$: Vector de parámetros iniciales para el método, según 3.10

- Parámetro global, escalar "*__Tol*": Tolerancia para el término de las iteraciones (parámetro de la función *eqSolve*)

2. Variables de salida:

   - Vector de dimensiones $NM + N + M$ correspondiente al vector 3.10.
   - Escalar: Tiempo de ejecución del método asociado al procedimiento.

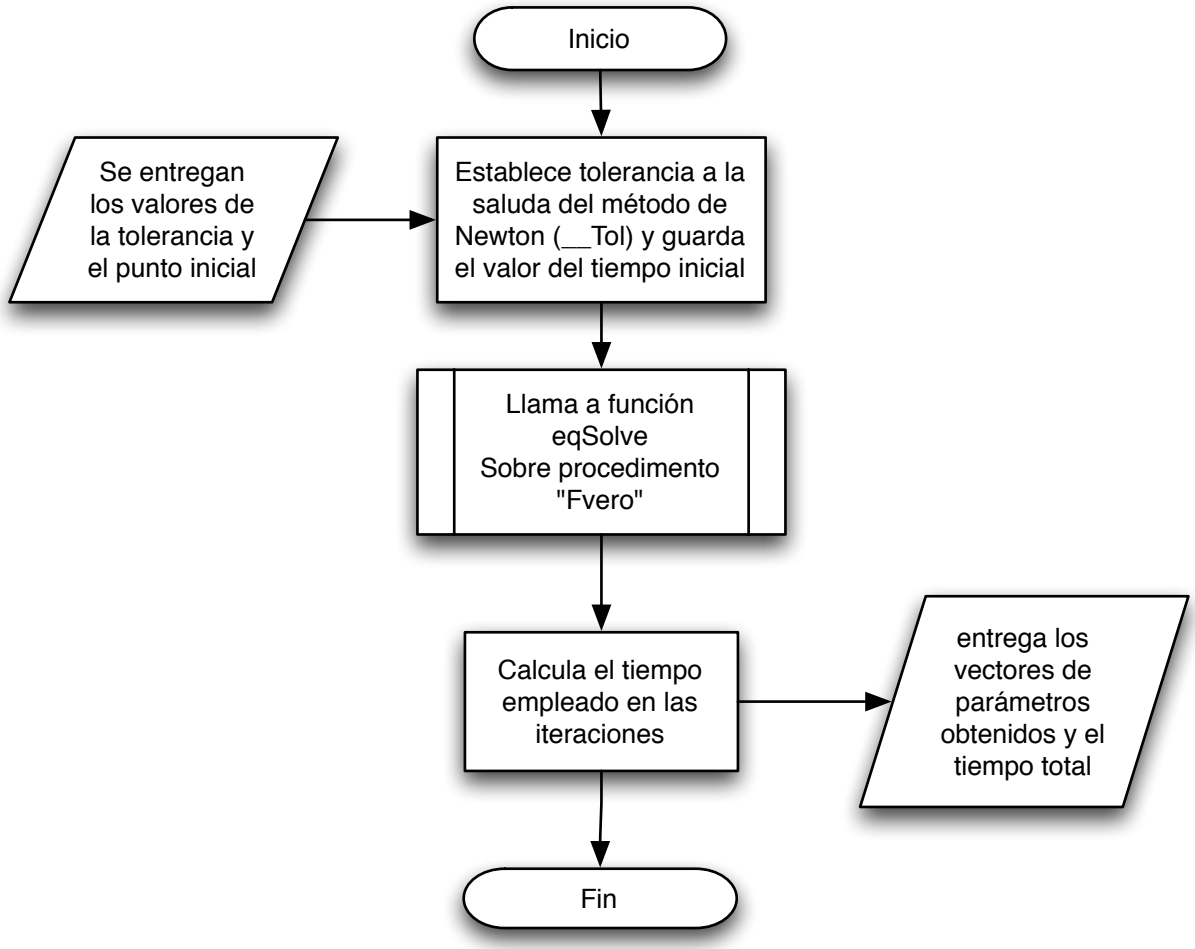


Figura 4.2: Operación del método Newton Rhapson sobre Máxima Verosimilitud

### Método de Punto Fijo sobre Factores de Balance y Newton Rhapson

Esta función denominada *PFN* en la librería programada, realiza el procedimiento descrito en 3.2.3, el cual consiste en iteraciones de punto fijo para los factores de balance, y resolver empleando Newton-Rhapson vectorial las condiciones de primer orden asociadas a las restricciones de costos. Este método emplea la función de GAUSS *eqSolve* para encontrar

los parámetros $\beta_n$, que corresponden a los multiplicadores de Lagrange para las restricciones de costos.

El método recibe las siguientes variables de entrada:

1. Variables de entrada:
   - Vector de dimensión $M$ : parámetros $\beta_n$ : $\beta_n^0$ iniciales
   - Vector de dimensión $N$ : factores de balance $[B_j]$ iniciales
   - Real: Tolerancia para finalizar las iteraciones, asociada a la norma euclidiana del error en el cumplimiento de las ecuaciones de primer orden (ver ecuación (3.14)).

   Además el método emplea las siguientes variables globales:
   - Matriz de dimensión $NxMN$: Costos para cada categoría de usuarios, almacenados en la variable $\_C_{nij}$
   - Vector de dimensión $NM$: Viajes totales de origen por zona y categoría de usuario $O_{in}$
   - Vector de dimensión $N$: Viajes totales de destino por zona $D_j$

   Finalmente el procedimiento programado entrega las siguentes variables de salida:

2. Variables de salida:
   - Vector de factores de balance $A_{in}$ (vector de dimensión $NM$)
   - Vector de factores de balance $B_j$ (vector de dimensión $N$)
   - Vector de parámetros $\beta_n$ (vector de dimensión $M$)
   - Contador de número de ejecuciones (variable entera)
   - Tiempo en centésimas de segundo empleado (variable real)

El procedimiento básicamente realiza los siguientes pasos:

- Con los parámetrosdel paso anterior, o valores iniciales si es la primera iteración, actualiza el valor de $A_{in}$ mediante el uso de la ecuación (3.8a)
- Con los parámetros anteriores y $A_{in}$ actualizado, actualiza el valor de $B_j$, según la ecuación 3.8b

- Luego el programa realiza el llamado al procedimiento .^{eq}Solve", el cual resuelve la restricción de costos 3.8c. Para ello se emplea el Jacobiano analítico programado en el procedimiento *gradienteCosto*.
- Se realiza el cálculo del vector $\beta_n$ actualizado, el cual es calculado hasta la convergencia del método de Newton.
- Finalmente, si se cumple el conjunto de condiciones de primer orden, el procedimiento entrega las soluciones óptimas. De no cumplirse, regresa al primer punto.

Su funcionamiento se esquematiza en el siguiente flujograma:

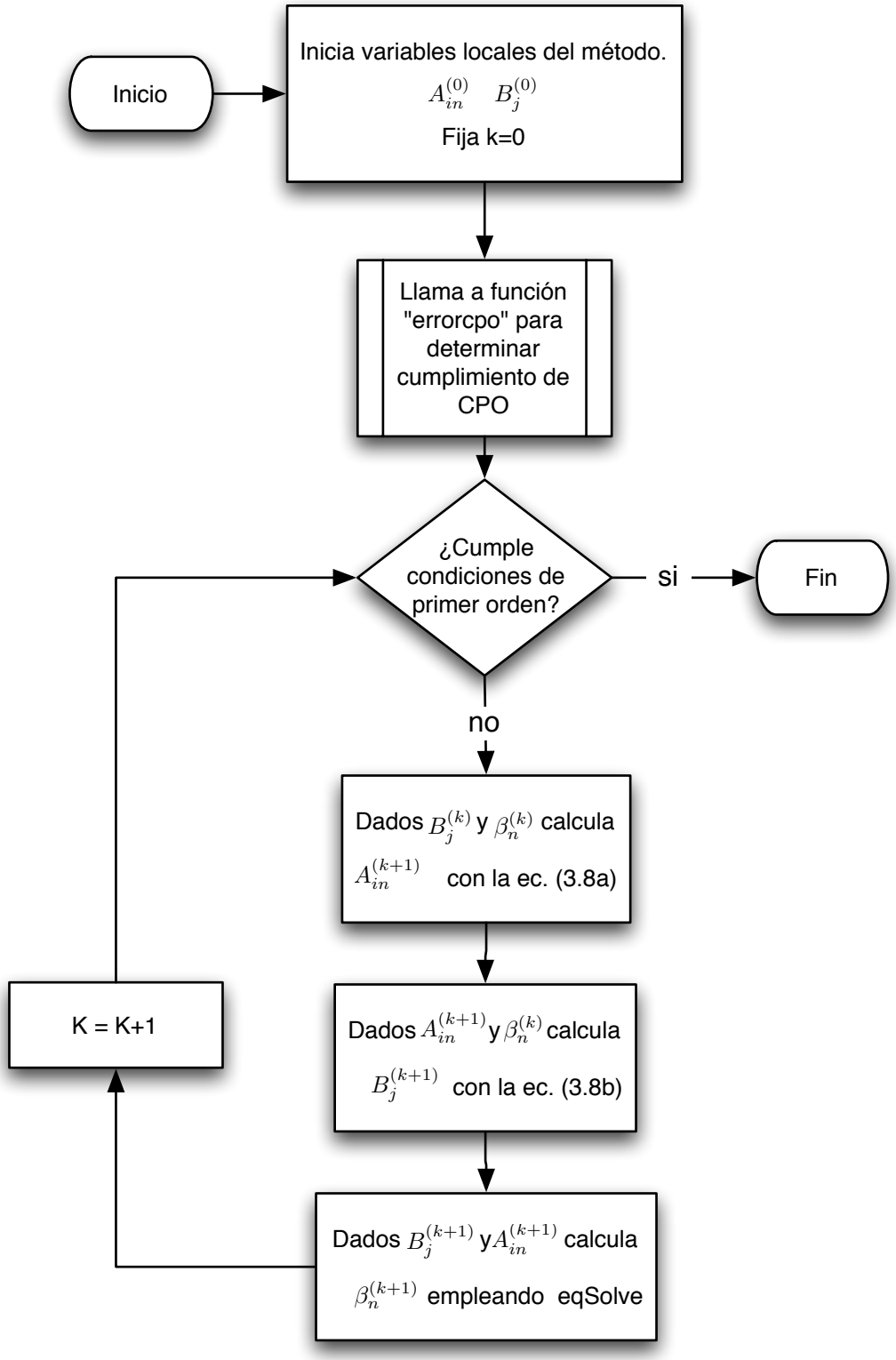


Figura 4.3: Operación del procedimiento *PFN*

**Método de Bregman**

El procedimiento de Bregman, cuyo nombre en la librería implementada es *Bregman*, corresponde al método presentado en 2.7. En él, se resuelve el problema de máxima entropía encontrando en cada iteración un factor *lambda* que actualiza en cada iteración uno de los multiplicadores de Lagrange. EL valor de *lambda* se encuentra resolviendo el sistema de ecuaciones (3.22) o resolviendo la ecuación (3.23) en *lambda* . Su implementación se describe a continuación.
En primer lugar el método de Bregman recibe las siguientes variables de entrada:

1. Variables de entrada:

   - Vector dimensión $NM + N + M$ con valores iniciales para variables duales $[\alpha_{in}]$, $[\theta_j]$ y $[\beta_n]$
   - Escalar con la tolerancia de cumplimiento para las condiciones de primer orden.

   Al igual que todos los procedimientos programados, *Bregman* accede a las variables globales dadas por:

   - Matriz de dimensión $NxMN$: Costos para cada categoría de usuarios, almacenados en la variable "$\_c_{nij}$"
   - Vector de dimensión $NM$: Viajes totales de origen por zona y categoría de usuario $O_{in}$
   - Vector de dimensión $N$: Viajes totales de destino por zona $D_j$

   Las variables de salida del procedimiento programado son las siguientes:

2. Variables de salida:

   - Vector de factores de balance $A_{in}$ (vector de dimensión $NM$)
   - Vector de factores de balance $B_j$ (vector de dimensión $N$)
   - Vector de parámetros $\beta_n$ (vector de dimensión $M$)
   - Contador de número de iteraciones realizadas (variable entera)
   - Tiempo en centésimas de segundo empleado (variable real)

El procedimiento además emplea la función de GAUSS *eqSolve* para la resolución del parámetro $\lambda$ según la ecuación (3.23).

Para el caso del procedimiento *Bregman*, la función encargada de verificar las condiciones de primer orden entrega no sólo el error, correspondiente a la diferencia entre la mano derecha

e izquierda de las condiciones de primer orden. Sino que también entrega la matriz de viajes estimada con esos parámetros. Esto se realizó para evitar operaciones duplicadas.

Para el cálculo de las ecuaciones de costos se emplea el procedimiento de Gauss *eqSolve*, el cual opera sobre el procedimiento *fnobjbreg*, que corresponde a la implementación de la función (3.23). Además *eqSolve* recibe un gradiente analítico para cada ecuación escalar, el cual se implementó en el procedimiento *fnobjbreg*.

El procedimiento *Bregman* realiza los siguientes pasos:

- Dado el vector inicial de concidiones de primer orden y la tolerancia al cumplimiento de la misma, se verifican el cumplimiento de las condiciones de primer orden y se inicializa una matriz de viajes $T$. Se inicializa un contador de iteraciones globales $k$
- Dado $k$, se actualiza el contador $i$ que recorre cíclicamente las restricciones. Éste corresponde a una implementación de la función $i = k\text{mod } (NM + N + M) + 1$
  - Si $i$ corresponde a una restricción de orígen o destino, se calcula $\lambda$ mediante (3.22), según corresponda.
  - Si $i$ corresponde a una restricción de costos se llama al procedimiento *eqSolve* entregándole el procedimiento del gradiente de la función, con ello se calcula el valor de *lambda* resolviendo (3.23).
- Con el valor de $\lambda$ se actualiza la componente i-ésima del vector de parámetros de primer orden.
- Si se cumplen las condiciones de primer orden, se transforman de las variables duales $[\alpha_{in}]$, $[\theta_j]$ a los factores de balance, y se termina. Si no, con la matriz actualizada $T$ se actualiza el contador *k=k+1* y se regresa

A continuación se presenta un esquema de la operación del procedimiento programado.

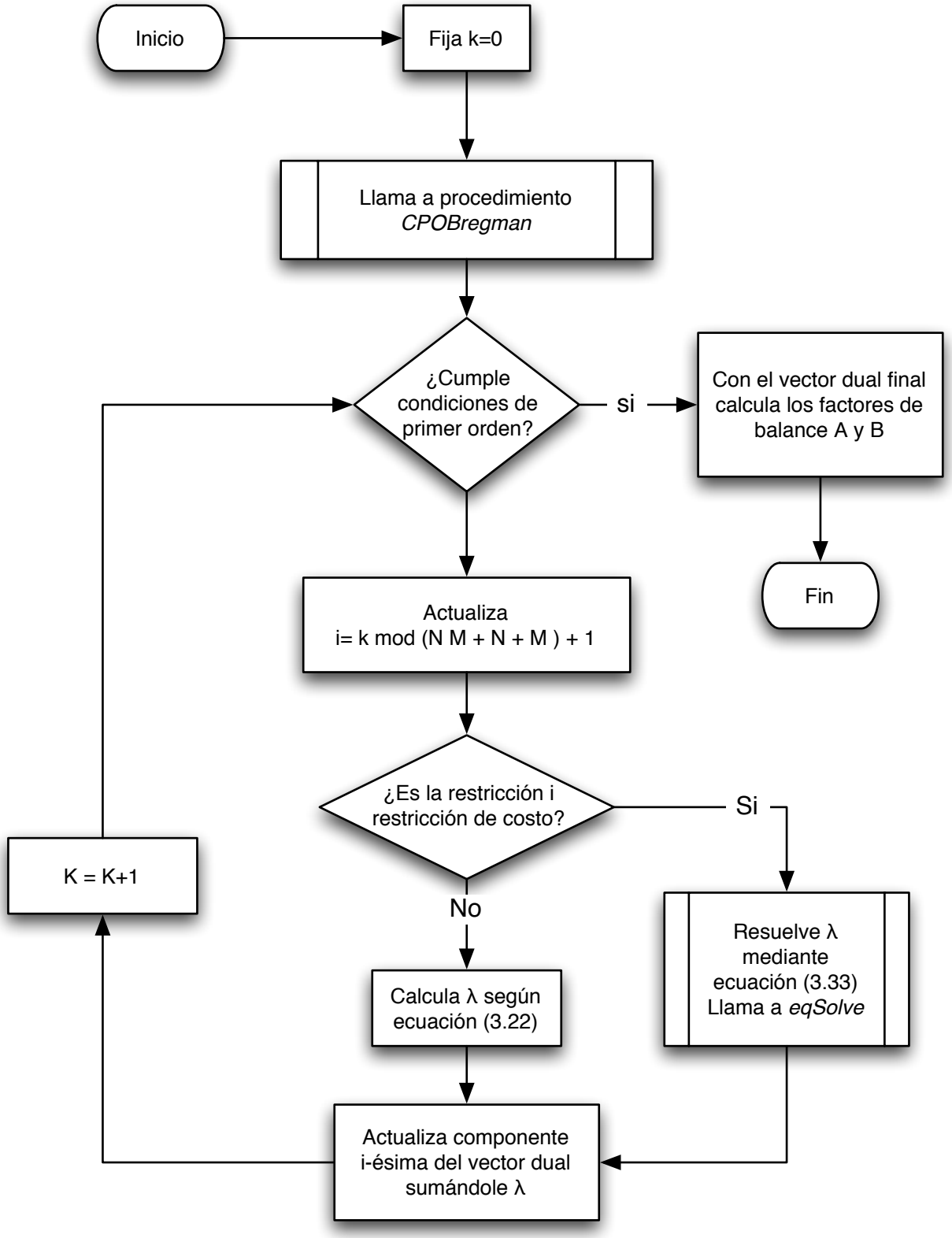


Figura 4.4: Operación del procedimiento *Bregman*

**Método MART**

El procedimiento MART, cuyo nombre en la librería implementada es *MART*, corresponde al método presentado en 3.2.5. En él, al igual que el método de Bregman, se resuelve el problema de máxima entropía encontrando en cada iteración un factor $\lambda$ que actualiza uno de los multiplicadores de Lagrange.

En este caso, el valor de $\lambda$ se encuentra resolviendo el sistema de ecuaciones (3.26) . Su implementación se describe a continuación.

1. Variables de entrada:
   - Vector dimensión $NM + N + M$ con valores iniciales para variables duales
   - Escalar con la tolerancia de cumplimiento para las condiciones de primer orden.
2. Variables de salida:
   - Vector de factores de balance $A_{in}$ (vector de dimensión $NM$)
   - Vector de factores de balance $B_j$ (vector de dimensión $N$)
   - Vector de parámetros $\beta_n$ (vector de dimensión $M$)
   - Contador de número de operaciones realizadas (variable entera)
   - Tiempo en centésimas de segundo empleado (variable real)

El procedimiento *MART* realiza los siguientes pasos:

- Dado el vector inicial de concidiones de primer orden y la tolerancia al cumplimiento de la misma, se verifican el cumplimiento de las condiciones de primer orden y se inicializa una matriz de viajes $T$. Se inicializa un contador de iteraciones globales $k$
- Dado $k$, se actualiza el contador $i$ que recorre cíclicamente las restricciones. Éste corresponde a una implementación de la función $i = k \text{mod } (NM + N + M) + 1$
- Se calcula $\lambda$ mediante(3.26), según corresponda.
- Con el valor de $\lambda$ se actualiza la componente i-ésima del vector de parámetros de primer orden.
- Si se cumplen las condiciones de primer orden, se transforman de las variables duales $[\alpha_{in}]$, $[\theta_j]$ a los factores de balance, y se termina. Si no, con la matriz actualizada $T$ se actualiza el contador *k=k+1* y se regresa

El procedimiento programado procede según se señala en la siguiente figura.

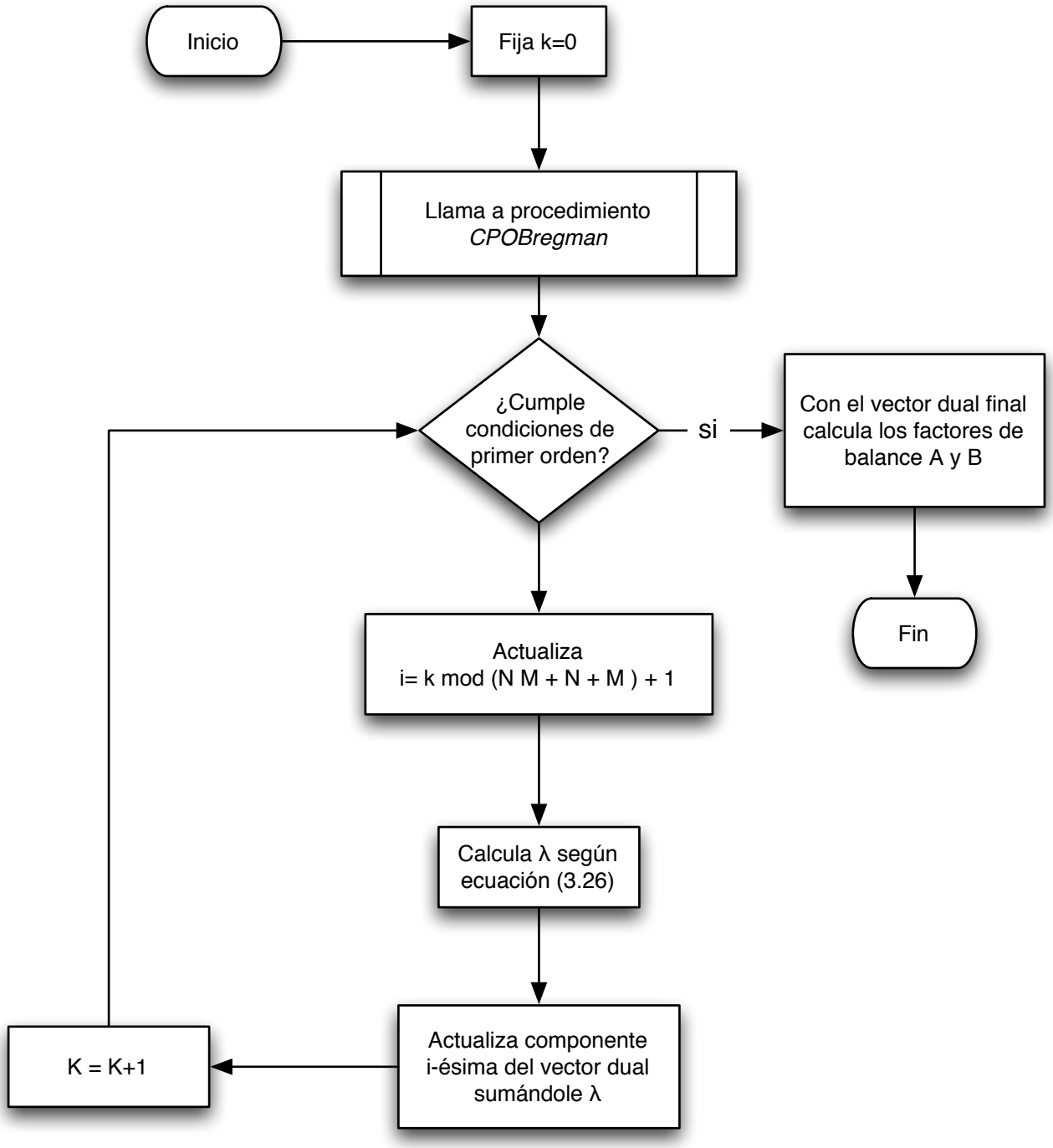


Figura 4.5: Operación del método *MART*

**Método de Hyman**

El método de Hyman implementado bajo el nombre de *Hyman* en la librería, corresponde al descrito en la sección 3.2.6. Su procedimiento es similar al método de Newton sobre las CPO, pues también realiza iteraciones sobre los factores de balance y luego calcula el valor de $\beta_n$, usando para ello el método de la secante. La gran diferencia con el método de Newton, es que el método de Hyman sólo realiza un paso en cada iteración.

A continuación se definen sus principales parámetros de entrada y salida:

1. Variables de entrada:

   - Vector de dimensión $M$ : parámetros $\beta_n$ : $\beta_n^0$ iniciales.
   - Vector de dimensión $N$ : factores de balance $[B_j]$ iniciales.
   - Real: Tolerancia para finalizar las iteraciones, asociada a la norma euclidiana del error en el cumplimiento de las ecuaciones de primer orden (ver ecuación (3.14)).
   - Vector de dimensión $M$: costos medios para cada categoría de usuarios, calculados mediantes la expresión (3.29)

   Además el método emplea las siguientes variables globales:

   - Matriz de dimensión $NxMN$: Costos para cada categoría de usuarios, almacenados en la variable $\_c_{nij}$
   - Vector de dimensión $NM$: Viajes totales de origen por zona y categoría de usuario $O_{in}$
   - Vector de dimensión $N$: Viajes totales de destino por zona $D_j$

   Finalmente el procedimiento programado entrega las siguentes variables de salida:

2. Variables de salida:

   - Vector de factores de balance $A_{in}$ (vector de dimensión $NM$)
   - Vector de factores de balance $B_j$ (vector de dimensión $N$)
   - Vector de parámetros $\beta_n$ (vector de dimensión $M$)
   - Contador de número de ejecuciones (variable entera)
   - Tiempo en centésimas de segundo empleado (variable real)

El procedimiento básicamente realiza los siguientes pasos:

- Con los parámetros del paso anterior, o valores iniciales si es la primera iteración, actualiza el valor de $A_{in}$ mediante el uso de la ecuación (3.8a)

- Con los parámetros anteriores y $A_{in}$ actualizado, actualiza el valor de $B_j$, según la ecuación 3.8b
- Luego el programa actualiza la variable $\beta_n$ realizando los siguientes pasos:
  - Si es la iteración inicial, actualiza $\beta_n$ mediante la relación (3.37).
  - Si no es la iteración inicial, da un paso genérico del método de la secante, según la ecuación (3.35)
- Finalmente, si se cumple el conjunto de condiciones de primer orden, el procedimiento entrega las soluciones óptimas. De no cumplirse, regresa al primer punto.

Finalmente el procedimiento se esquematiza en la siguiente figura:

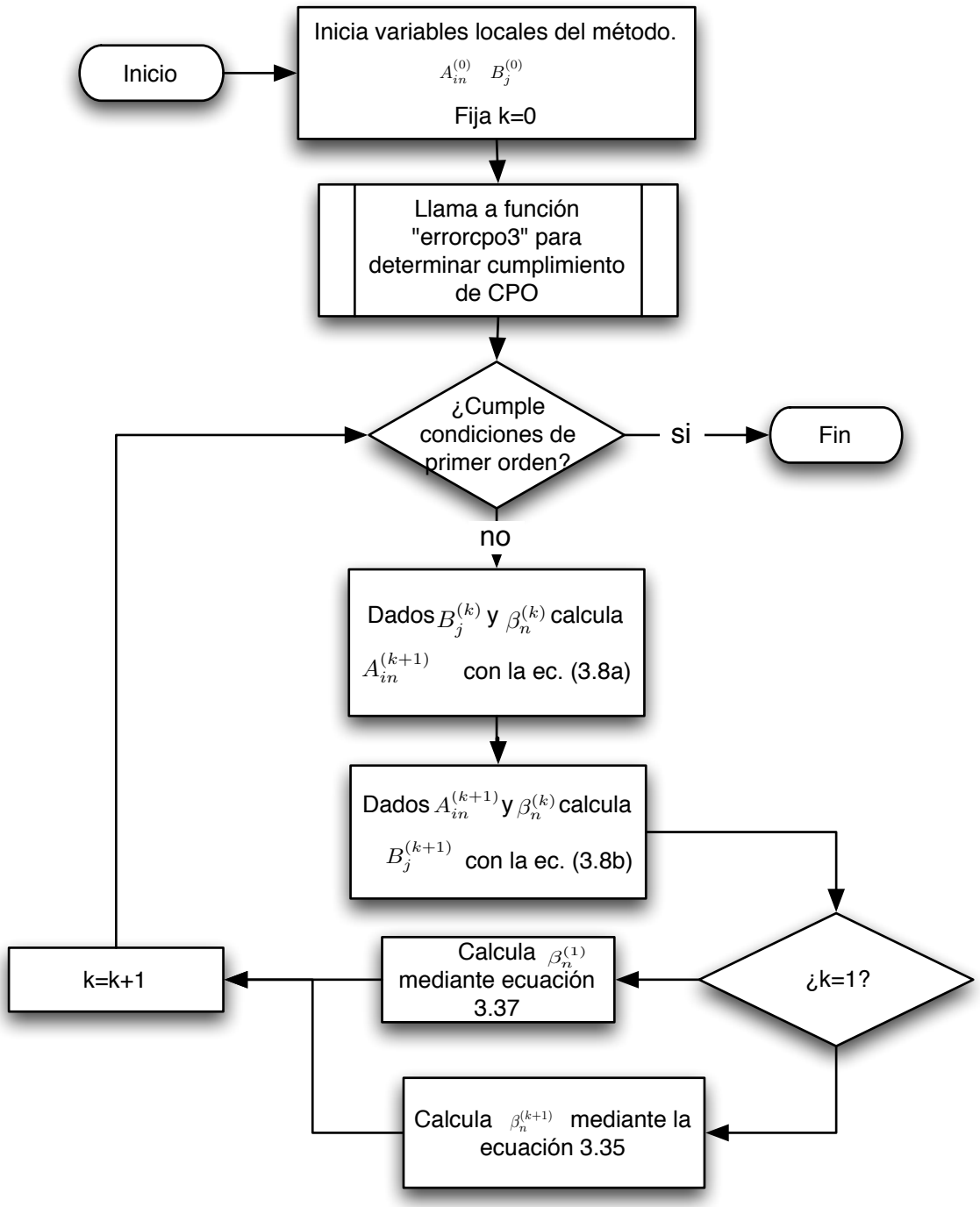


Figura 4.6: Operación del método *HYMAN*

### Método de Pivote

El método corresponde al primer método nuevo. Procede según se señala en 3.3.1. El procedimiento asociado a este método se encuentra bajo el nombre de *Pivote* en la librería programada. Su implementación se describe a continuación.

A continuación se describen las variables de entrada y salida que requiere y entrega el procedimiento programado:

1. Variables de entrada:
   - Vector dimensión $NM + N + M$ con valores iniciales para variables duales.
   - Escalar con la tolerancia de cumplimiento para las condiciones de primer orden.
   - Parámetro escalar correspondiente al ponderador entre costos mínimos y máximos.
2. Variables de salida:
   - Vector de factores de balance $A_{in}$ (vector de dimensión $nM$).
   - Vector de factores de balance $B_j$ (vector de dimensión $n$).
   - Vector de parámetros $\beta_n$ (vector de dimensión $M$).
   - Contador de número de iteraciones (variable entera).
   - Tiempo en centésimas de segundo empleado (variable real).

El procedimiento realiza los siguientes pasos:

- Con los parámetros del paso anterior, o valores iniciales si es la primera iteración, actualiza el valor de $A_{in}$ mediante el uso de la ecuación (3.8a)
- Con los parámetros anteriores y $A_{in}$ actualizado, actualiza el valor de $B_j$, según la ecuación 3.8b
- Luego el programa actualiza la variable $\beta_n$ siguiendo lo definido en (3.52).
- Finalmente, si se cumple el conjunto de condiciones de primer orden, el procedimiento entrega las soluciones óptimas. De no cumplirse, regresa al primer punto.

  Este procedimiento se puede ver esquematizado en la siguiente figura:

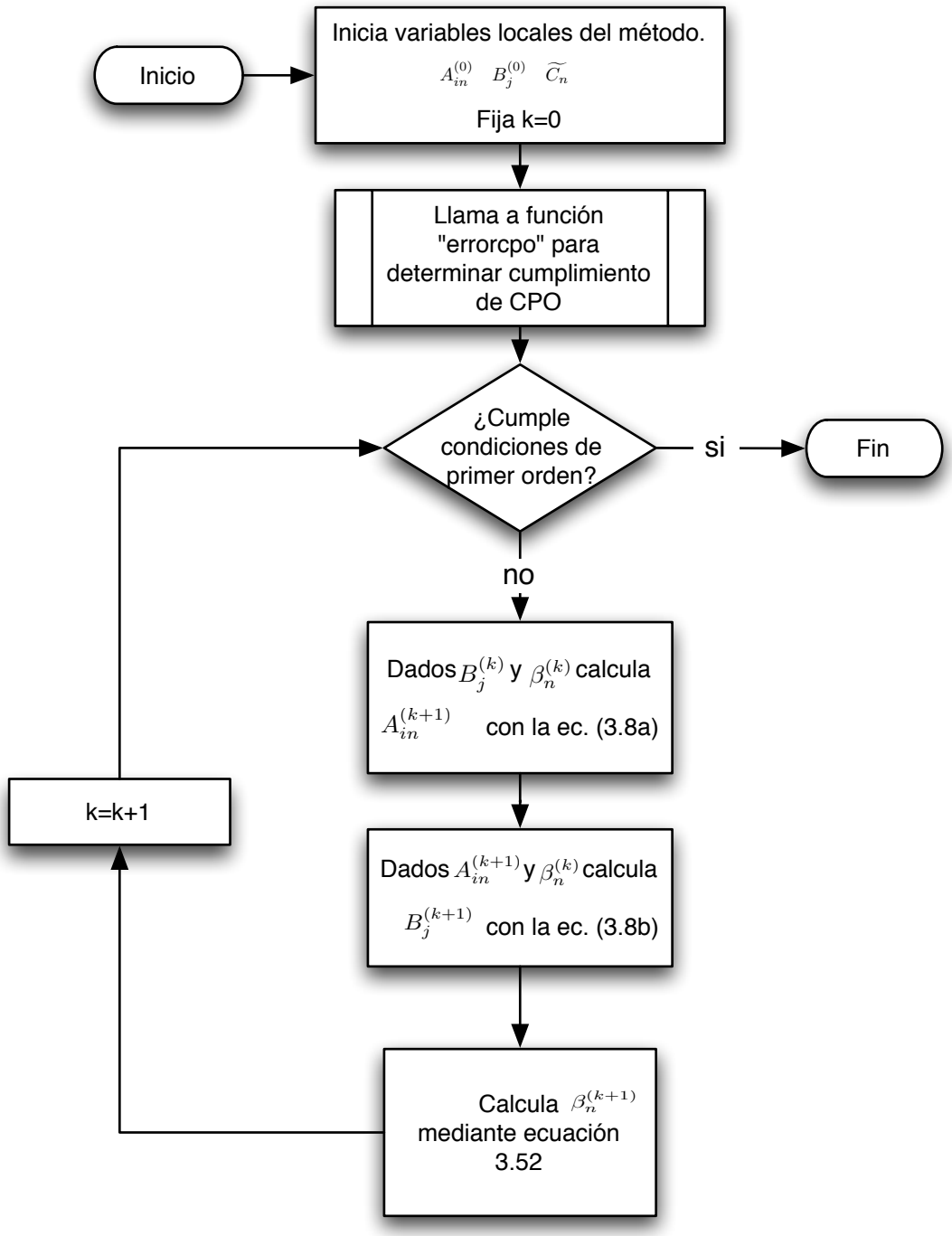


Figura 4.7: Operación del procedimiento *Pivote*

### Método de Búsqueda Acotada

El método corresponde al segundo propuesto para este análisis, y su codificación se realizó bajo el nombre *MBA* en la librería programada. El algoritmo procede según se señala en 3.3.2.

Las variables de entrada y salida del procedimiento *MBA* se presentan a continuación:

1. Variables de entrada:
   - Vector dimensión $NM + N + M$ con valores iniciales para variables duales
   - Escalar con la tolerancia de cumplimiento para las condiciones de primer orden.
   - Parámetro escalar correspondiente a $\lambda$ en la ecuación (3.45)
2. Variables de salida:
   - Vector de factores de balance $A_{in}$ (vector de dimensión $nM$)
   - Vector de factores de balance $B_j$ (vector de dimensión $n$)
   - Vector de parámetros $\beta_n$ (vector de dimensión $M$)
   - Contador de número de operaciones realizadas (variable entera)
   - Tiempo en centésimas de segundo empleado (variable real)

El procedimiento básicamente realiza los siguientes pasos:

1. Dados $A_{in}^{(0)} \in \Re^{nM}$ vector inicial con factores de balances factibles y $\epsilon$ positivo suficientemente pequeño. Se fija $k = 1$ contador global de las iteraciones.
2. En base a la ecuación (3.8b) y al vector $[A_{in}]^{(k)}$ podemos encontrar un vector $[B_j]^{(k+1)}$
3. Con el vector $B_j{}^{(k+1)}$ se calcula $[A_{in}]^{(k+1)}$ mediante la relación (3.8a)
4. Con los valores de los factores de balance dados, se busca $\theta \in [0, 1]$ tal que la función $R_{A_{in},B_j}(\theta_n) = 0$ definida en (3.73) mediante el método de Newton-Raphson, realizando sólamente un paso.
5. Con el valor de $\theta_n$ podemos calcular $\beta_n$ mediante la relación (3.74)
6. Se verifica convergencia: Si el cumplimiento de las condiciones de primer orden, dado por el cumplimiento de la función $F(x)$ definida en (3.9) es con un nivel de tolerancia $\epsilon$ se termina el algoritmo, si no $k = k + 1$ y se regresa al segundo punto.

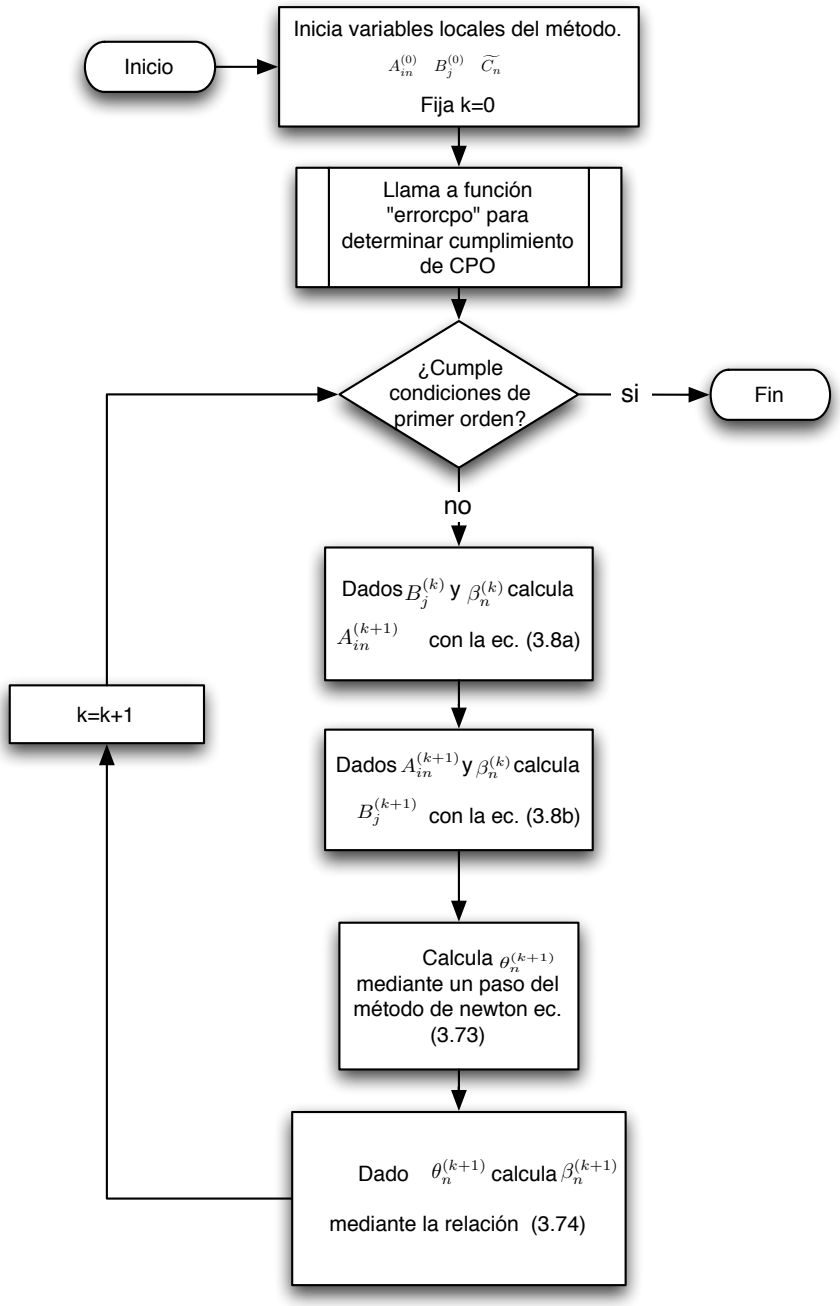


Figura 4.8: Operación del procedimiento *MBA*

### 4.1.2. Procedimientos auxiliares

1. Función *ErrorCPO*: Esta función es de una importancia vital pues no sólo forma parte principal del método de Newton sobre las condiciones de primer orden, sino que también es empleada por todos los métodos para verificar la convergencia de los algoritmos. Recibe vector de parámetros $\{A_{in}, B_j, \beta_n\}$ y empleando las variables globales $O_{in}$, $D_j$ y $c_{nij}$ entrega la función $F$ correspondiente al cumplimiento de las condiciones de primer

orden definida en (3.9). Este método es empleado por los procedimientos: *Pivote*, *MBA*, *NewtonCPO*

2. Función *ErrorCPO2*: Este procedimiento es idéntico al anterior, salvo que además entrega una matriz $T$ con cálculos intermedios para evitar cálculos repetidos en el procedimiento *PFN*.

3. Función *ErrorCPO3*: Este procedimiento es idéntico al anterior, salvo que además entrega una matriz $t$ con cálculos intermedios para evitar cálculos repetidos en el procedimiento *Hyman*.

4. Función *F1*: Este procedimiento es idéntico a *ErrorCPO*, sólo que recibe como parámetros de entrada las variables duales y entrega la matriz de viajes estimada por los parámetros recibidos. El método es empleado por los procedimientos *Bregman* y *MART*

5. Funcion *errorct*: Esta función realiza el cálculo del error asociado a las restricciones de costo, es decir, las últimas $M$ componentes de la función definida en (3.9). Es empleada en el cálculo de los parámetros $\beta_n$ por el método *PFN*.

6. Función *NormaEuclideana*: Procedimiento que recibe un vector de cualquier dimensión y calcula el valor de su norma euclidiana. Empleado sobre todas las funciones anteriores, para comparar el error en el cumplimiento de las CPO con la tolerancia entregada al método.

7. Función *fnobjbreg*: Procedimiento que recibe un vector de parámetros duales, y con él calcula la el valor de la ecuación $F_{Bregman}$ definido en la ecuación (3.23). Se emplea para resuelta mediante Newton-Rhapson.

8. Función *gfnobjbreg*: Procedimiento que recibe el vector de parámetros $\beta_n$ y con él calcula el gradiente de la $F_{Bregman}$ definido en la ecuación (3.23).

9. Función *agranda*: Procedimiento que recibe vector o matriz $X$ y escalar $n$ y entrega la matriz concatenada $n$ veces, es decir entrega $[X|X|X...|X]$

Capítulo 5

# MEJORAMIENTO Y EDICIÓN DE LOS ALGORITMOS

Para la programación de los algoritmos se recibió un código base programado como primera aproximación para los algoritmos. Éste fue modificado y refinado para la realización de las pruebas, de esta manera se eliminan ineficiencias que alteren los resultados. En este capítulo se describen y discuten las mejoras realizadas al código, tanto en estructuras de programación como en modificaciones a los algoritmos para mejorar la velocidad de convergencia, principal parámetro a comparar entre los diversos métodos.

En primer lugar, se describen los métodos empleados para el análisis del código, posteriormente se analizan cambios profundos en los algoritmos implementados y finalmente se presentan los resultados de dichas mejoras para algunos casos de prueba, las cuales se extrapolan a problemas de tamaños similares.

## 5.1. Métodos Empleados

A continuación se describen los métodos y herramientas empleadas para la corrección del código.

- Revisión de partes específicas del código:
  Las principales secciones de código revisadas están relacionadas con la búsqueda de eficiencia en los códigos, reduciendo al máximo posible los ciclos y sustituyendo los mismos, de ser factible, por operaciones matriciales. Así también se realizó la búsqueda de operaciones duplicadas, dando preferencia a mantener variables en memoria sobre realizar cálculos un número reiterado de veces. Este es el motivo por el cual existen distintos procedimientos para calcular el cumplimiento de las condiciones de primer orden, cada una de ellas es distinta pues entrega además del valor del error una variable distinta que permite ahorrar cálculos al programa principal que la emplea.

  Se analizó dentro de los distintos programas empleados el manejo de las variables de entrada y salida al realizar llamadas de procedimientos locales por procedimientos glo-

bales, se presta gran atención a este punto debido a que las operaciones matriciales programadas de manera ineficiente pueden llevar a grandes diferencias entre métodos que que realizan operaciones línea a línea (Bregman, MART) y matriciales (Newton sobre CPO, Newton sobre Máx. Verosimilitud, Hyman).

- Consultas a Ingenieros en Computación:
  Se contó con la ayuda de un especialista en el software GAUSS, con lo que permitió convertir secciones especificas del código que permiten ahorrar esfuerzo computacional, tanto en el manejo de la memoria como en el tiempo empleado por cada función. En especial se privilegió el mantener variables locales, como por ejemplo el vector de parámetros a calibrar, en memoria sobre su escritura en disco. Esto debido a que sobre pruebas de tamaños reales y bajo un adecuado manejo de variables, los problemas tratados no alcanzan a copar totalmente la memoria disponible para los programas .Además un segundo argumento a favor de privilegiar el rendimiento (o tiempo de retorno de la aplicacion) por sobre el tamaño en memoria principal, es debido a la existencia de sistemas operativos que pueden ejecutar instrucciones en 64-bits, aumentando la actual barrera de 32-bits (4 GB de RAM máximo) a un máximo de 17.2 billiones de GB, con lo que los problemas de paginación serán evitables.

- Medición del tiempo de ejecución del algoritmo por línea:
  Se realizó una búsqueda exhaustiva de ineficiencias en el algoritmo para lo cual se implementaron variables de control que permiten conocer el número de veces que fue llamada una línea y el tiempo total utilizado por la misma. Este método de trabajo permite reconocer en que puntos específicos cada procedimiento emplea mayor tiempo. Además, se puede establecer diferencias entre las velocidades por secciones de código lo que permite emplear las codificaciones más rápidas en distintos algoritmos.

- Pruebas en variaciones del código:
  Se probó la implementación de los algoritmos variando las estructuras de datos usadas, por ejemplo se probaron las siguientes modificaciones:

  - Se implementaron los algoritmos bajo la estructura de *arrays* en la cual se manejaron matrices de más de 2 dimensiones. Esto se realizó para comparar las velocidades de los códigos frente a distintas estructuras de almacenamiento de las variables.
  - Se implementaron códigos alternativos para la resolución de entropía mediante maximización de la entropía, utilizando el módulo comercial de GAUSS para la

máxima verosimilitud *MAXLIK* . El cual corrió principalmente mediante el método de optimización BHHH, el cual es un tipo de método de penalización, que se basa en la equivalencia del producto cruzado de las primeras derivadas con la matriz Hessiana, es decir, las segundas derivadas. Esta prueba se realizó para comparar si la implementación del método de Newton sobre las condiciones de primer orden del problema de máxima verosimilitud equivalente eran más rápidas o lentas que la aplicación del método *MAXLIK*.

## 5.2. Cambios Implementados

Podemos destacar grandes transformaciones de la programación de los algoritmos recibidos entre las cuales se destacan las siguientes:

1. Cambio de estructuras de programación:
   Se realizó un cambio general en la estructura de los programas, estos pasaron de ser funciones aisladas que realizan llamadas entre si (con la inevitable duplicación de variables en memoria) a métodos presentes dentro de una librería, la cual maneja variables globales comunes a todos los métodos evitando el traspaso de información de variables almacenadas al disco duro, operación que resulta ineficiente al comparar las tasas de lectura y escritura de un disco duro contra el almacenamiento de las variables en memoria RAM (Random Acces Memory)[1]. Por ejemplo se puede notar que en la codificación inicial, cada vez que el método para resolver ecuaciones mediante Newton ( *eqSolve* ) era llamado, se guardaban en disco algunas variables que luego eran leídas por el procedimiento que entrega la función a resolver.
   Otra mejora inherente al cambio de la estructura de los programas es su utilización, la cual mejora ostensiblemente al no tener que cargar función por función a la librería de procedimientos del software GAUSS. Bajo el formato de librería basta con incluirla e inicializar las variables globales de las mismas para poder obtener un acceso inmediato a todos los procedimientos programados.

2. Eliminación de variables y operaciones duplicadas:
   Debido a la implementación de herramientas que permiten conocer los tiempos de ejecución y número de llamados por línea de código, fue posible identificar instrucciones que se encontraban duplicadas en distintas funciones. Se introduce como cambio el traspaso de información entre procedimientos, dejando los cálculos más pesados sin duplicaciones.

En torno a las pruebas realizadas se obtuvieron las siguientes conclusiones:

[1] Información obtenida en conversación con Ingeniero en Computación

- Para el método de Punto Fijo y Newton Rhapson (ver sección 3.2.3) se cambió la estructura de programación recibida en la programación original, principalmente se observó una diferencia significativa entre el obtener convergencia de los factores de balance y luego realizar el cálculo del parámetro $\beta_n$ en comparación con la búsqueda secuencial de factores presentada en 4.3, la que fue finalmente implementada. Con ello se pasó del esquema de convergencia en $A_{in}$ y $B_j$ para pasar a $beta_n$, a un esquema donde se realizan iteraciones secuenciales $A_{in}$, $B_j$ y$beta_n$ en cada iteración.

- El mayor tiempo computacional en los distintos algoritmos es debido a la construcción de la matriz de viajes, la cual es realizada para la verificación de las matrices de primer orden. Para resolver este problema, se estima la matriz de viajes sólo una vez al termino de cada iteración de manera de obtener de ella el cumplimiento de las condiciones de primer orden y al mismo tiempo el dato de entrada para los cálculos siguientes.

- El método *MAXLIK* implementado sobre el método del gradiente conjugado no presenta mejoras significativas en tiempo de ejecución en relación al uso del método de Newton-Rhapson, sin embargo aporta información más detallada en pantalla sobre las iteraciones y al finalizar entrega test de significancias, los cuales son obtenidos con las matrices Jacobianas del problema.

- La programación en estructuras de datos tipo *ARRAY* presenta grandes ventajas en la visualización del código, debido a que almacenando en este típo de estructuras el tamaño del código se reduce notablemente. Sin embargo, no se aprecia una disminución significativa del tiempo de ejecución y se pierde compatibilidad con versiones anteriores de GAUSS, por esto se decidió conservar la estructura definida en un comienzo.

## 5.3. Resultados de las mejoras

Previo a la descripción de los resultados en las mejorasde cada método se introduce una notación para cada método.

Cuadro 5.1: Notación para Algoritmos

| Método | Notación |
|---|---|
| Búsqueda Acotada | PF($\alpha, \theta$)N($\beta$-Acotado) |
| Pivote | PF($\alpha, \theta, \beta$) |
| Punto Fijo y Newton | PF($\alpha, \theta$) N($\beta$) |
| Bregman | BREGMAN |
| MART | MART |
| Hyman | PF($\alpha, \theta$)S($\beta$) |
| Newton sobre CPO | N($A, B, \beta$) |
| Newton sobre Máx. Verosimilitud | N($\alpha, \theta, \beta$) |

Se realizó una comparación en los tiempos de ejecución, el principal punto a optimizar dentro de los algoritmos, entre los algoritmos iniciales y los resultados finales obteniendo los siguientes resultados:

Cuadro 5.2: Resultados de la Mejora de Código

| Algoritmo | Porcentaje de Mejora Promedio |
|---|---|
| PF($\alpha, \theta$)N($\beta$-Acotado) | 26 % |
| PF($\alpha, \theta, \beta$ | 87 % |
| PF($\alpha, \theta$) N($\beta$) | 89 % |
| BREGMAN | 39 % |
| MART | 73 % |

Las principales mejoras en los métodos se deben a la incorporación de las mejoras señaladas en 5.2 Para los métodos de Newton sobre las condiciones de primer orden, y máxima verosimilitud no se realizan cambios debido a que su codificación principalmente realiza llamadas a la función *eqSolve*. El método de Hyman fue programado con posterioridad y ya incorpora las mejoras detectadas para los otros algoritmos. Por ello los tiempos de ejecución permanecen invariantes.

## 5.4. Comentarios sobre paralelización de los algoritmos

En términos generales el paralelismo en computación se entiende como la realización de actividades simultáneas, esta forma de programación se ve fuertemente reforzada con la aparición de procesadores de uso casero que implementan más de un núcleo para un mismo

procesador. Como es de esperarse los procesos computacionales realizados en procesos paralelos serán más rápidos que los métodos tradicionales, en los que cada instrucción siguiente debe esperar a completar la instrucción anterior.

La principal caracterstica para la identificación de procesos o partes del código que puedan ser paralelizadas es su independencia con el resto del código al mismo nivel. En este sentido cabe destacar que los problemas descritos en esta investigación no son por su naturaleza independientes en cada proceso, sin embargo pueden abordarse de manera paralela al considerar la separación del código en porciones y resolver en procesos independientes. Por ejemplo el método de Newton-Rhapson, o las iteraciones de factores de balance podrán ser estimadas en de manera paralela. Supongamos que el vector de variables a actualizar $B_j$ de dimensión N es dividido en 2 vectores de igual dimensión $B1_j$ y $B2_j$. Con ello se podrá actualizar en un proceso independiente y simultáneo el vector $B1$ y el vector $B2$. Lo cual significará una reducción del tiempo de ejecución.

Capítulo 6

# PRUEBAS DE METODOS CON DATOS SIMULADOS

En este capítulo se describen las pruebas de los algoritmos sobre datos simulados y se entregan los resultados correspondientes. Las pruebas son realizadas para comparar numéricamente los tiempos de ejecución de los algoritmos propuestos en comparación con los algoritmos descritos en los capítulos anteriores.

Las pruebas se realizan para estudiar los tiempos de ejecución, principal variable a analizar en esta investigación. Por ello, en primer lugar se seleccionan los algoritmos más rápidos corriendo sobre un escenario pequeño.

En segundo lugar, se estudiarán las variaciones en los tiempos de ejecución de los algoritmos más rápidos ante distintos escenarios, escogidos en un tamaño suficiente para no causar problemas de paginación a los programas y representativos de una zonificación para una cuidad mediana. Los escenarios reproducen variaciones en el número de parámetros a estimar, se consideran distintos números de restricciones, distintos tamaños de la matriz (variable $N$) y distintas categorías de usuarios (variable $M$). Además se consideran distintas varianzas en las matrices de costos y finalmente se estudian escenarios con distintas tolerancias al cumplimiento de las condiciones de primer orden.

Un resumen de ello se puede apreciar en cuadro siguiente:

Cuadro 6.1: Resumen de escenarios a probar

| Escenario | N. Zonas $n$ | Categorías $M$ | Var. en $C_{ijn}$ | Tolerancia de CPO |
|---|---|---|---|---|
| Número de zonas | variable | 15 | 1 | $10^{-5}$ |
| Categorías de usuarios | 300 | variables | 1 | $10^{-5}$ |
| Varianza de Matriz de costo | 300 | 15 | variable | $10^{-5}$ |
| Tolerancia de CPO | 300 | 15 | 1 | variable |

A continuación, se describe la generación de los datos empleados para las pruebas, la selección del grupo de algoritmos más rápidos y luego se describen las pruebas realizadas.

Finalmente se presentan los resultados de las mismas y se plantea un análisis de lo obtenido.

## 6.1. Generación de Datos Simulados

Para efectos de las pruebas sobre datos simulados se empleó una rutina consistente en generar, dado un número de zonas: $N$ y un número de usuarios $M$, las matrices de viajes y matrices costos asociados a un problema de distribución de viajes doblemente acotado correspondiente a . La rutina procede como sigue:

- Para cada categoría se inicializan las variables correspondientes a los multiplicadores de Lagrange, con valores aleatorios que siguen una distribución uniforme entre 0 y 1.
- Dados los multiplicadores de Lagrange, se genera la matriz de viajes para la categora $n$-sima mediante la ecuación (3.4a).
- Dada la matriz de viaje, se suman los subtotales para categoría de usuario y se obtienen los $O_{in}$ y $D_j$ respectivos.
- Para cada categoría de usuario se genera una matriz de costos de tamaño $nxn$, cuyos elementos son generados de manera aleatoria, mediante una distribución normal con media 0 y desviación 1, mediante el uso de la función aleatoria uniforme del software GAUSS.
- Con esta la matriz de viaje y los costos, se suman los subtotales para categoría de usuario obteniendo los $C_{0n}$ respectivos.

Descripción del hardware empleado
Para la realización de las pruebas y su medición de tiempo de ejecución se empleará un computador con las siguientes características:

Cuadro 6.2: Hardware empleado

| CPU | Intel Pentium D 945 |
|---|---|
| Velocidad de Procesador | 3.40 GHz |
| Memoria Caché | 2048 KBytes |
| Memoria RAM | 3072 MBytes |
| Sistema Operativo | Windows XP Profesional Versión 2002,SP2 |

## 6.2. Pruebas preliminares

Se realizaron pruebas preliminares a los algoritmos, ellas fueron ejecutadas en distintos computadores con hardware y disponibilidad de memoria RAM diferente. Las pruebas fueron

realizadas bajo la codificación final de los algoritmos, es decir, incluyendo todas las mejoras y optimizaciones posibles. Los resultados de las pruebas mostró que los algoritmos siempre mantienen una relación similar, su orden y velocidad de ejecución es similar en todos los casos.

Un ejemplo de lo anterior se observa con los siguientes datos:

Cuadro 6.3: Datos Prueba Preliminar

| | |
|---|---|
| Número de Zonas | 100 |
| Número de categorías de usuarios | 5 |
| Tolerancia de salida | $10^{-5}$ |
| Varianza de Matrices de Costo | 1 |

La calibración se realiza en el computador descrito en el cuadro 6.2 y entrega los siguientes resultados.

Cuadro 6.4: Resultados Preliminares (Ejemplo)

| Método | Tiempo de ejecución [s] |
|---|---|
| PF($\alpha, \theta$)N($\beta$-Acotado) | 0,81 |
| PF($\alpha, \theta, \beta$) | 1,31 |
| PF($\alpha, \theta$) N($\beta$) | 1,46 |
| BREGMAN | 131,34 |
| MART | 473,05 |
| PF($\alpha, \theta$)S($\beta$) | 0,19 |
| N($A, B, \beta$) | 58,78 |
| N($\alpha, \theta, \beta$) | 38,34 |

De los resultados obtenidos en todas las simulaciones realizadas, se pueden diferenciar a grandes rasgos dos grupos de algoritmos de punto fijo, el primer grupo corresponderá a los algoritmos más rápidos y en un segundo grupo los más lentos. Esto se puede apreciar también en el cuadro 6.2.

Los grupos generados son los siguientes:

1. Grupo N°1: Algoritmos más rápidos.
   - Método de Hyman (PF($\alpha, \theta$)S($\beta$))
   - Método de Búsqueda Acotada (PF($\alpha, \theta$)N($\beta$-Acotado))
   - Método de Pivote $\beta$ (PF($\alpha, \theta, \beta$)).

- Método de Punto Fijo y Newton Vectorial (PF($\alpha, \theta$) N($\beta$))

2. Grupo N°2: Algoritmos más lentos.

   - Método de Bregman (BREGMAN)
   - Método MART (MART)
   - Método de Newton sobre CPO (N($A, B, \beta$))
   - Método de Newton sobre Sistema de máxima verosimilitud (N($\alpha, \theta, \beta$))

Debido a la intención de realizar pruebas sobre los algoritmos para enfrentar problemas de gran tamaño, se realizarán las pruebas descritas en el cuadro 6.1 a los algoritmos pertenecientes al primer grupo.

A continuación se presentan los resultados de las distintas pruebas, separadas en tres categorías que corresponden a distintos escenarios en que los algoritmos puedan presentar variaciones en sus tiempos de ejecución.

## 6.3. Pruebas con Variaciones en Número de Restricciones

Para el análisis sobre las variaciones en el número de restricciones, debemos considerar sus distintas naturalezas. Mientras que las restricciones asociadas al tamaño de la matriz (restricciones de origen y destino) poseen una solución analítica que permite despejar directamente los parámetros en función de los anteriores, las restricciones de costos, asociadas al número de categoría de usuarios a simular, son por su solución analítica difíciles de resolver.

Si se considera $M$ categorías de usuarios y una zonificación original de $N$ zonas de Origen y Destino, el problema original presentará la siguiente estructuras de restricciones:

$$\sum_{j}^{J} T_{nij} = O_{ni} \quad \forall i, n \tag{6.1}$$

$$\sum_{n}^{N} \sum_{i}^{I} T_{nij} = D_j \quad \forall j \tag{6.2}$$

$$\sum_{i}^{I} \sum_{j}^{J} T_{nij} c_{nij} = C_{n0} \quad \forall n \tag{6.3}$$

Como se puede apreciar existen $N * M$ restricciones asociadas a las restricciones de origen, $N$ restricciones asociadas a los destinos y $M$ restricciones asociadas a los costos por categoría de usuario.

- Al aumentar en una zona el total aparecerán $M$ restricciones del tipo (6.1), una restricción del tipo (6.2) y ninguna restricción del tipo (6.3).
- Al aumentar en uno el número de usuarios distintos considerados se tendrán $N$ nuevas restricciones del tipo (6.1) y una restricción del tipo (6.3).

### 6.3.1. Pruebas con Variaciones en Número de Zonas

Las pruebas se realizaron en el equipo señalado en la tabla 6.2 . Se analizan distintos escenarios que están definidos por variaciones en los tamaños de las zonas, las cuales iran aumentando Los parámetros restantes se mantendrán en los valores señalados en el cuadro 6.1.

Cuadro 6.5: Tiempo de ejecución en segundos, M=15

| N | Nº Restricciones | PF$(\alpha,\theta)$N($\beta$-Acotado) | PF$(\alpha,\theta,\beta)$ | PF$(\alpha,\theta)$ N($\beta$) | PF$(\alpha,\theta)$S($\beta$) |
|---|---|---|---|---|---|
| 200 | 3215 | 16,14 | 20,59 | 22,66 | 4,24 |
| 250 | 4015 | 26,27 | 22,609 | 22,094 | 7,01 |
| 300 | 4815 | 45,44 | 45,203 | 44,453 | 11,38 |
| 350 | 5615 | 65,40 | 91,14 | 99,84 | 15,95 |
| 400 | 6415 | 94,39 | 117,59 | 127,1 | 21,95 |
| 450 | 7215 | 117,31 | 136,2 | 129,45 | 29,47 |
| 500 | 8015 | 153,75 | 150,64 | 121,45 | 38,52 |
| 600 | 8815 | 177,03 | 245,97 | 205,5 | 44,26 |
| 700 | 9615 | 221,77 | 280,3 | 274,84 | 58,36 |

Se observa que los métodos pertenecientes al grupo más rápido no presentan grandes diferencias en los tiempos de ejecución. Además el método de Hyman se muestra siempre del orden de 4 veces más rápido que el método que lo sigue en tiempos de ejecución, en este caso el método de Búsqueda Acotada.

En segundo lugar se observa que el comportamiento de los algoritmos propuestos, pivote y búsqueda acotada, tiende a ser superior al algoritmo que resuelve el métodomediante punto fijo en factores de balance y newton en $\beta$ PF$(\alpha,\theta)$N($\beta$-Acotado). Aún así, son más lentos que Hyman, que emplea el método de la secante a un paso. La mayor diferencia entre los métodos de este grupo radica en el cálculo del vector $[\beta_n]$, pues el cálculo de los factores de balance es realizado de la misma manera.

El aumento en el número de zonas (N) genera el auménto en el cálculo de los factores de balance, y debido a que estos métodos resuelven los factores de balance de la misma manera, los resultados obtenidos en el cuadro 6.5 son representativos de esta situación.

Se observa además que la tendencia al aumento no es lineal con el aumento en el número de zonas, y a medida q estas aumentan, el ahorro en tiempo es proporcional para los métodos mas rápidos.

### 6.3.2. Pruebas con Variaciones en Número de Usuarios

Al igual que para el caso señalado antes, se realizarán pruebas sobre las variaciones del número de usuarios $M$. Este caso corresponde al más interesante, pues las restricciones que presentan mayor dificultad para su calibración estan asociadas directamente con el aumento de $M$.

Los resultados obtenidos son los siguientes:

Cuadro 6.6: Tiempos de ejecución en segundos para prueba variación de M , $N = 300$

| M | Nº Restricciones | PF($\alpha, \theta$)N($\beta$-Acotado) | PF($\alpha, \theta, \beta$) | PF($\alpha, \theta$) N($\beta$) | PF($\alpha, \theta$)S($\beta$) |
|---|---|---|---|---|---|
| 5 | 1805 | 13,61 | 11,81 | 12,09 | 3,58 |
| 10 | 3310 | 30,36 | 36,75 | 38,37 | 7,80 |
| 15 | 4815 | 44,59 | 58,67 | 51,75 | 11,11 |
| 20 | 6320 | 64,95 | 77,68 | 81,26 | 16,32 |
| 25 | 7825 | 82,61 | 149,61 | 138,42 | 22,03 |
| 30 | 9330 | 106,82 | 146,41 | 129,90 | 26,58 |
| 35 | 10835 | 126,24 | 130,38 | 130,79 | 31,40 |
| 40 | 12340 | 137,44 | 148,11 | 152,95 | 34,72 |
| 45 | 13845 | 175,56 | 214,7 | 218,91 | 44,06 |

Como se puede apreciar el aumento es similar para los distintos métodos , salvo el método de Hyman que presenta un tiempo mucho menor. Tanto el método de Hyman como el de búsqueda acotada presentan tendencias de aumento similares, manteniendo las diferencias de tiempo de ejecución. Los tramos donde el aumento del número de usuarios no incrementa el valor del tiempo de ejecución (entre M=30 y M=40) se puede atribuir a las distintas configuraciones de matrices empleadas.

Con el objeto de probar la efectividad del método ante la solución de problemas con restricciones lineales similares a las de costo, se realizan pruebas con un número pequeño de zonas y una gran cantidad de usuarios como se describe en la siguiente tabla. De esta manera se busca acentuar el efecto de las distintas estratégias para abordar la calibración de la restricción de costos.

Cuadro 6.7: Tiempos de Ejecución, Escenarios de aumento de usuarios nº2, $N = 10$

| M | Nº Restricciones | PF($\alpha, \theta$)N($\beta$-Acotado) | PF($\alpha, \theta, \beta$) | PF($\alpha, \theta$) N($\beta$) | PF($\alpha, \theta$)S($\beta$) |
|---|---|---|---|---|---|
| 250 | 2760 | 1,33 | 1,30 | 2,33 | 0,26 |
| 500 | 5510 | 4,83 | 3,20 | 11,70 | 0.59 |
| 750 | 8260 | 5,31 | 7,06 | 31,02 | 0,89 |
| 1500 | 16510 | 8,89 | 15,87 | 63,90 | 1,63 |

Se puede apreciar que el método de Hyman presenta menor tiempo de ejecución en comparación con los otros métodos analizados, el segundo mejor método corresponde al método de búsqueda acotada, cuyo aumento en el tiempo es similar al método de Hyman. Además, se observa que las ventajas se incrementan fuertemente al aumentar el número de restricciones de costos asociadas a cada nueva categoría de usuarios.

## 6.4. Pruebas con Variaciones en Grado de Convergencia

Las pruebas sobre variaciones en el grado de convergencia de los procedimientos, se lleva a cabo al tener parámetros que cumplan las restricciones de primer orden distintas tolerancias $\varepsilon$. Estas pruebas permiten estudiar en que manera la tolerancia al cumplimiento de las condiciones de primer orden condiciona el término del algoritmo y el tiempo de ejecución del mismo.

Cuadro 6.8: Escenarios de Variación en cumplimiento de CPO, $N = 300$, $M = 15$

| Tolerancia | PF($\alpha, \theta$)N($\beta$-Acotado) | PF($\alpha, \theta, \beta$) | PF($\alpha, \theta$) N($\beta$) | PF($\alpha, \theta$)S($\beta$) |
|---|---|---|---|---|
| $10^{-9}$ | 68,61 | 87,45 | 189,03 | 93,46 |
| $10^{-8}$ | 65,14 | 106,70 | 171,50 | 13,23 |
| $10^{-7}$ | 57,2 | 72,19 | 99,57 | 12,41 |
| $10^{-6}$ | 55,07 | 92,73 | 122,39 | 12,38 |
| $10^{-5}$ | 50,11 | 85,92 | 93,47 | 11,49 |
| $10^{-4}$ | 45,77 | 32,75 | 36,43 | 11,47 |
| $10^{-3}$ | 39,61 | 36,05 | 41,31 | 10,66 |
| $10^{-2}$ | 34,97 | 33,38 | 49,01 | 9,83 |
| $10^{-1}$ | 30,03 | 24,14 | 36,59 | 9,79 |

Se aprecia que la sensibilidad de los métodos es relativamente pequeña para el método de Búsqueda Acotada, mientras que el método de Hyman pierde su ventaja al someterse al cumplimiento de restricciones muy pequeñas. Lo anterior puede entenderse debido a que este método emplea una aproximación para encontrar la raiz de la restricción. Esta aproximación

se acerca a la solución a una tasa menor que la tasa del método de Newton, sin embargo requiere menos esfuerzo computacional al calcular la evaluación de la función sólo una vez por iteración, mientras que el método de Newton requiere estimar tanto la función en el punto, como la inversa de la matriz Jacobiana.

## 6.5. Pruebas con Distintas Varianzas en la Matriz de Costos

En este caso, se estudian distintos escenarios caracterizados por distintas matrices de costos, cuya varianza es alterada mediante la amplificación de las matrices.
Los resultados obtenidos son los siguientes:

Cuadro 6.9: Tiempos de ejecución en segundos, ante distintas varianzas en la matriz de costos

| Tolerancia | PF($\alpha, \theta$)N($\beta$-Acotado) | PF($\alpha, \theta, \beta$) | PF($\alpha, \theta$) N($\beta$) | PF($\alpha, \theta$)S($\beta$) |
|---|---|---|---|---|
| 1 | 49,36 | 48,89 | 44,609 | 11,54 |
| 2 | 47,49 | 124,01 | 63,24 | 12,50 |
| 4 | 46,203 | 238,52 | 95,8 | 16,58 |
| 6 | 41,31 | 310,78 | 106,12 | 30,21 |
| 8 | 32,93 | 415,01 | 91,84 | 76,23 |
| 10 | 34,84 | 509,55 | 88,21 | 120,49 |

Se observa que el incremento en la varianza de los costos lleva a un aumento sostenido en el tiempo de ejecución del algoritmo de Pivote y Hyman. Sin embargo los métodos de búsqueda acotada y Punto Fijo- Newton se mantienen estables. Se aprecia que en situaciones de mayor varianza, el método de búsqueda acotada es superior en rendimiento al método de Hyman.

Capítulo 7

# PRUEBA DE METODOS CON DATOS REALES

## 7.1. Descripción de la base de datos

Para la calibración del modelo de entropía se utilizarán los valores provenientes de la calibración en el modelo ESTRAUS de los datos provenientes de la encuesta Origen-Destino realizada en Santiago en el año 2001 (EOD 2001). La utilización de datos reales tiene el objeto de contar con estructuras de costos y viajes que sean representativas de la estructura real de la ciudad de Santiago. Además se destaca que se emplea una corrida que no incluye proyectos no-implementados, es decir, la situación a calibrar es representativa de la estructura de viajes en la ciudad para el año 2001.

La encuesta Origen-Destino es una herramienta encargada por el Ministerio de Planificación y Cooperación (MIDEPLAN), asesorado por la Secretaría Ejecutiva de la Comisión de Planificación de Inversiones en Infraestructura de Transporte (SECTRA), cuyo objetivo es conocer las características de las personas que habitan el hogar y los viajes que realizan un día predeterminado ("día de viaje").

La información que se empleará para trabajar corresponde a las provenientes de las modelaciones ESTRAUS, con lo cual se trabajarán con 13 distintos usuarios, provenientes del cruce de categorías económicas El modelo ESTRAUS es capaz de resolver, al año 2001, asignación y partición modal de manera conjuntas en el sentido en que existe consistencia entre los niveles de servicio y flujos. ESTRAUS considera, en su modelo de asignación y distribución conjunta, un modelo formulado como inecuación variacional, resuelto mediante diagonalización, donde en cada etapa se resuelve un problema de optimización en el que aparecen términos de entropía sujetos a restricciones lineales simples. Las matrices entregadas por Sectra corresponden a las matrices del modelo estratégico ESTRAUS generadas en base a la encuesta Origen -Destino 2001, y se encuentran desagregadas como sigue:

- Por periodo: Punta Mañana y Fuera de Punta

- Por propósito: Estudio 1 y 2, Trabajo y Otros.
- Por categoría de usuario, desde la 01, hasta la 13.
- Por modo: Auto Acompañante, Auto Chofer, Auto Acompañante-Metro, Bus, Bus-Metro, Taxi, Taxi Colectivo y Caminata

Estas matrices, debido al alto grado de desagregación, contienen entre un 3 % y un 5 % de información no nula

### 7.1.1. Descripción de las pruebas

Para la realización de las pruebas se escogió el propósito que presenta una menor cantidad de celdas no nulas para un periodo, que corresponde a las matrices con propósito*trabajo* en horario punta mañana ($AM$). En la preparación de la prueba se procedió como sigue:

- Se transforman las matrices en formato ESTRAUS empleando el programa utilitario *conviertematrices* el cual arroja un archivo de extensión .dat, en texto simple.
- Mediante la utilización del utilitario de GAUSS $ATOG$ se transformaron cada una de las matrices (todos los modos y todas las categorías de usuario) en archivos binarios de GAUSS
- Para el caso de las matrices de costos se empleó el valor 300 en las celdas que no presentan información, mientras que para el caso de las matrices de viajes se empleó el valor 0.
- Una vez convertidas las matrices se sumó celda a celda para cada categoría de usuario los modos disponibles
- Las 13 matrices de costos y 13 matrices de viajes son concatenadas según lo requieren los métodos programados generando una única matriz de costos y guardadas con formato *.fmt*, el que corresponde al formato de matriz de GAUSS.
- Se suman las columnas de las matrices de viajes para generar los elementos del vector $D_j$ y las filas de las mismas para generar las restricciones $O_{in}$
- Finalmente mediante la multiplicación elemento a elemento de las matrices de costo y viajes se obtienen los parámetros $C_{0n}$

Los algoritmos programados no han sido preparados para trabajar con matrices con poca información. Por ello los algoritmos trabajarán con matrices con información en todas sus celdas, tal como se señala en el tercer punto.

### 7.1.2. Resultados

Se realizó la calibración de los parámetros del modelo de entropía empleando los algoritmos descritos y se obtuvieron los resultados que se presentan en el siguiente cuadro:

Cuadro 7.1: Resultados calibración matrices reales

| Algoritmo | Tiempo de Calibración |
|---|---|
| PF$(\alpha, \theta)$S$(\beta)$ | 34, 23 [seg] |
| PF$(\alpha, \theta)$N($\beta$-Acotado) | 46,57 [seg] |
| PF$(\alpha, \theta, \beta)$ | 2,92 [min] |
| PF$(\alpha, \theta)$ N$(\beta)$ | 19,53 [min] |
| Bregman | 1,03 [hr] |
| MART | $\geq 3[hr]$ |
| N$(A, B, \beta)$ | $\geq 3[hr]$ |
| N$(\alpha, \theta, \beta)$ | $\geq 3[hr]$ |

Como se puede apreciar del cuadro anterior, el método de Hyman (PF$(\alpha, \theta)$S$(\beta)$ ) y Búsqueda Acotada (PF$(\alpha, \theta)$N($\beta$-Acotado) ) presentan una gran ventaja sobre los métodos alternativos. Además, la ventaja observada entre estos métodos se reduce para esta calibración, lo que es atribuible a la varianza introducida con los costos en valor 300. Se puede observar además que del grupo más lento, el algoritmo de Bregman es finalmente el más rápido alcanzando la convergencia en un tiempo cercano a una hora.
Las iteraciones fueron detenidas luego de tres horas, pues debido a la implementación de los algoritmos, el único criterio de parada es el cumplimiento de las condiciones de primer orden, por lo mismo los algoritmos pueden mantenerse iterando un tiempo indeterminado hasta cumplir con esta condición.

Capítulo 8

# Conclusiones

Se han planteado dos nuevos algoritmos pertenecientes a la familia de punto fijo para resolver el problema de ajuste de los parámetros de los modelos de entropía. Un primer algoritmo, emplea un pivote para generar un problema de punto fijo para la estimación de las variables. La convergencia en la búsqueda en los parámetros esta asegurada mediante la verificación de la contractancia del punto fijo planteado. Un segundo algoritmo es obtenido al generar un cambio de variables en la solución del problema, pasando de la búsqueda del parámetro en el conjunto de los reales a una búsqueda acotada en el intervalo $[0, 1]$. Este algoritmo empíricamente conserva la velocidad de convergencia del método de Newton pero requiere de un menor tiempo de ejecución por instrucción. Ambos algoritmos presentan una gran velocidad y eficiencia en la calibración de parámetros de entropía en comparación con otros algoritmos encontrados en la literatura de transporte.

Se observa que los métodos más rápidos son aquellos cuya estrategia es realizar iteraciones para factores de balance seguidas de dar un solo paso en la búsqueda del parámetro $\beta_n$. Es decir, el esquema propuesto por Bregman para la solución del problema, aplicada bajo distintos métodos, secante en caso de Hyman y Newton en $\theta$ para el método de búsqueda acotada, mostró ser lo más rápido.

## 8.1. Sobre la Implementación

Se obtuvo una codificación eficiente para todos los algoritmos de calibración, eliminando las posibles ventajas ficticias entre los métodos producto de ineficiencias del código. La codificación empleada permitirá la posterior implementación del código en otros modelos.

Los programas implementados no consideran específicamente el caso de matrices con poca información por lo que los métodos planteados trabajaron con matrices cuyos valores fueron completados de manera de evitar errores como por ejemplo, divisiones o amplificaciones por 0.

Los métodos presentados, son estrategias para abordar la solución del problema de calibración que pueden ser implementados sobre métodos especiales para matrices con poca

información.

Una de las principales conclusiones obtenidas en la etapa de depuración del código y pruebas de estrategias de calibración es la conveniencia de la realización de iteraciones tipo Furness para la calibración del problema de entropía. Además, se puede destacar que el esquema de actualización más eficiente en términos del tiempo de ejecución y simplicidad de código es mediante la actualización secuencial de los factores de balance, es decir, en la etapa (k) se deberán encontrar $A^{(k+1)}$ dado $B^{(k)}$ y$\beta^{(k)}$, seguido de la búsqueda de $B^{(k+1)}$ dado $A^{(k+1)}$ y $\beta^{(k)}$, finalmente $\beta^{(k+1)}$ dado $A^{(k+1)}$ y $B^{(k+1)}$. La configuración en donde los parámetros $B$ y $A$ son calculados hasta una cierta convergencia, pese a no requerir de un gran número de iteraciones no presenta beneficios en la convergencia general de la búsqueda, incluso en pruebas realizadas alcanza varios ordenes de magnitud en tiempo de ejecución sobre la versión secuencial. Otro nivel más es realizar el cálculo del parámetro $\beta$ mediante un paso por iteración global. Esto es implementado por los dos métodos más rápidos, los que entregaron para todos los casos los menores tiempos de ejecución.

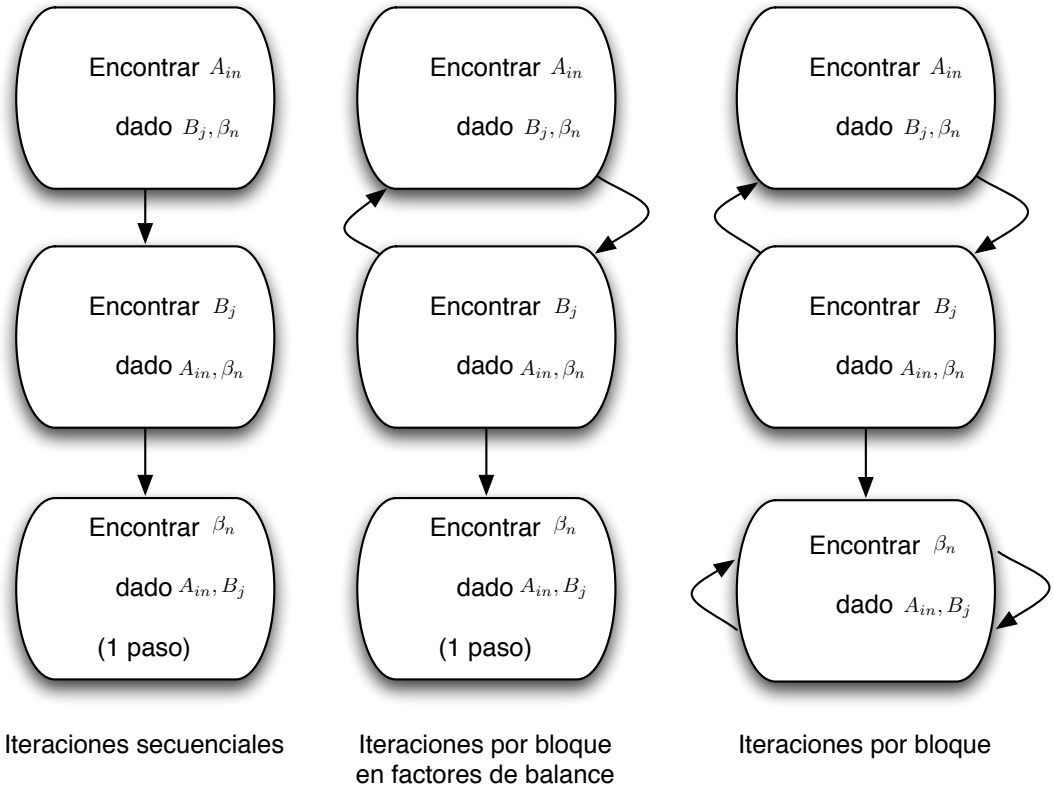


Figura 8.1: Estrategias para una iteración global

Así también el método que realiza iteraciones empleando el método de la secante (método

de Hyman), realiza en cada iteración solamente un paso. Estrategia que le permite avanzar rápidamente al punto de solución. En la práctica se observó que este método requiere de un número limitado de iteraciones para condiciones normales , del orden de 10 a 20 según el tamaño del problema. Sin embargo, al aumentar la exigencia en precisión al método, este aumentó su tiempo de ejecución así como también el número de iteraciones. Para los casos en que la varianza es alta, el método de Hyman empeoró su tiempo de ejecución, llegando a ser peor que el método que lo sigue en todas las pruebas (método de búsqueda acotada).

El método de la secante es una buena aproximación para la solución en la variable $\beta$, sin embargo en pruebas exploratorias al realizar iteraciones de secante sobre la variable acotada $\theta$, no se obtuvieron mejores resultados. Sin embargo, estos análisis quedan abiertos para un trabajo futuro.

Una segunda conclusión importante se puede obtener al considerar la implementación realizada para calibrar el modelo de entropía mediante el uso de máxima verosimilitud. Se observó que en general el método de GAUSS *MAXLIK* genera tiempos de ejecución superiores en comparación con los métodos que resuelven directamente las condiciones de primer orden del problema asociado.

## 8.2. Sobre los resultados Obtenidos

De las pruebas realizadas se obtuvo que entre los dos algoritmos propuestos el más eficiente resulta ser el método de búsqueda acotada, el cual ante escenarios de gran tamaño muestra ventajas apreciables para la obtención de los parámetros de calibración. Esta ventaja del método radica en la velocidad con que es resuelta cada iteración, debido principalmente a no realizar iteraciones locales para el parámetro $\theta$.

A pesar de sólo realizar un paso del método de Newton-Rhapson, el método de variable acotada en $\theta$ requiere de un número de iteraciones similar al método de Newton hasta la convergencia en variable $\beta$, el cual es muy superior al requerido por la aplicación del método de la secante (Hyman) a un paso. Se observa que el método de Pivote, pese a realizar mayor cantidad de iteraciones que el método de Punto Fijo y Newton, termina en menor tiempo, debido al bajo costo computacional que requiere cada iteración.

A pesar de las tendencias presentadas en los distintos resultados debe señalarse que no se debe considerar una extrapolación de los resultados. Limitantes como la cantidad de memoria disponible y configuraciones internas de las matrices que puedan resultar en cambios tanto

en los tiempos de ejecución resultantes, como en las tendencias reflejadas por los algoritmos. Por ejemplo, esto ocurre en el caso en que el tamaño de la matriz es tal que las dificultades de calibración sean por el agotamiento de la memoria disponible y el computador comience el proceso de paginación o escritura en disco, ralentizando el proceso.

La calibración empleando datos reales fue considerada con el objetivo de obtener un indicador real del rendimiento de los métodos, sin embargo debe establecerse que para el caso específico en que las matrices contengan poca información, los algoritmos empleados pierden competitividad contra algoritmos dedicados a resolver solamente aquellas zonas donde se tenga información.

## 8.3. Extensiones y trabajo futuro

Como extensión del trabajo se debe señalar que la técnica empleada para acotar el problema de la exponencial puede ser extendida también al problema con restricciones de desigualdad, los cuales se pueden observar en modelos de localización entre otros.
Otra arista importante es la aplicación de este método a problemas de entropía de asignación conjunta, en donde el problema de entropía o las variables a calibrar puedan ser resueltas mediante la aplicación de la técnica de búsqueda acotada.

Además se podrán estudiar los efectos tales como la codificación para múltiples núcleos, permitiendo realizar de manera simultánea secciones de código independientes, agilizando los procesos que ahora son estimados de manera secuencial. En los casos en que las matrices superen 4GB de memoria, se deberá estudiar la codificación en plataforma 64-bits en lugar de las plataformas actuales de 32-bits. Permitiendo trabajar con archivos de hasta 17,2 billones de GB según la cantidad de memoria disponible en el computador.

Finalmente se obtuvieron dos métodos nuevos, de los cuales la resolución mediante acotamiento de la exponencial mostró ser el mejor. El rendimiento de estos nuevos métodos supera ampliamente a métodos clásicos para la resolución de los problemas y es equiparable con el método más rápido encontrado. La estabilidad del método de acotamiento de la exponencial, o búsqueda acotada, principalmente ante cambios en la varianza de la matriz de costos y su capacidad de entregar resultados con gran precisión en tiempos reducidos lo sitúan como un método altamente viable para su utilización en problemas de gran tamaño.

# Bibliografía


Anas, A. (1983). Discrete choice theory, information theory and the multinomial logit and gravity models. *Transportation Research Part B: Methodological*, 17(1):13–23.

Bar-Gera, H. (2006). Primal method for determining the most likely route flows in large road networks. *Transportation Science*, 40(3):269–286.

Berger, A. L., Pietra, S. A. D., y Pietra, V. J. D. (1996). A maximum entropy approach to natural lenguage processing. *Computational Linguistics*, 22(1).

Domencich, T. A., McFadden, D. (1975). Urban travel demand: A behavioral analysis. *North Holland, Amsterdam.*

Eriksson, J. (1980). A note on solution of large sparse maximum entropy problems with linear equality constraints. *Mathematical Programming*, 18(1):146–154.

Erlander, S. (1981). Entropy in linear programs. *Mathematical Programming*, 21(1):137–151.

Fang, S.-C., Rajasekera, J., Tsao, H.-S. J. (1999). *ENTROPY OPTIMIZATION AND MATHEMATICAL PROGRAMMING.* Kluwer Academic Publishers.

Fang, S.-C., Tsao, H.-S. J. (1995). Linealry-constrained entropy maximization problem with quadratic cost and its applications to transportation planning problems. *Transportation Science*, 29(4):354–365.

Fisk, C. S. (1988). On combining maximum entropy trip matrix estimation with user optimal assignment. *Transportation Research Part B: Methodological*, 22(1):69–73.

Gonçalves, M., de Cursi, J. (2001). Parameter estimation in a trip distribution model by random perturbation of a descent method. *Transportation Research Part B*, 35(2):137–161.

Ham, H., Kim, T., Boyce, D. (2005). Implementation and estimation of a combined model of interregional, multimodal commodity shipments and transportation network flows. *Transportation Research Part B: Methodological*, 39(1):65–79.

Ho, H., Wong, S., Loo, B. (2006). Combined distribution and assignment model for a continuum traffic equilibrium problem with multiple user classes. *Transportation Research Part B: Methodological*, 40(8):633–650.

Hyman, G. M. (1969). The calibration of trip distribution models. *Environment and Planning*, 1:105–112.

Islam, S., Roy, T. (2006). A new fuzzy multi-objective programming: Entropy based geometric programming and its application of transportation problems. *European Journal of Operational Research*, 173(2):387–404.

Jörnsten, K. O. (1981). An algorithm for the combined distribution and assignment problem. *Transportation Research Part B: Methodological*, 15(1):21–33.

Lamond, B., Stewart, N. F. (1981). Bregman's balancing method. *Transportation Research Part B: Methodological*, 15(4):239–248.

Leung, S. (2007). A non-linear goal programming model and solution method for the multiobjective trip distribution problem in transportation engineering. *Optimization and Engineering*, 8(3):277–298.

Malouf, R. (2002). A comparison of algorithms for maximum entropy parameter estimation. *International Conference On Computational Linguistics*, pages 1–7.

Martinez, F. Henríquez, R. (2007). The rb & sm model: A random bidding and supply land use model. *Transportation Research Part B*, 42(6):632–651.

Ortúzar, J. (1994). *Modelos de demanda de transporte*. Ediciones Universidad Católica, primera edición edition.

Ortuzar, J., Willumsen, L. (1994). *Modelling transport*. Wiley New York.

Schofield, E. (2007). Fitting maximum-entropy models on large sample spaces. *PhD Disertation, Derpartment of Computing, Imperial College, London*.

Sen, A. (1986). Maximum likelihood estimation of gravity model parameters. *Journal of Regional Science*, 26(3):461–474.

Tsekeris, T.(2004) Hybrid meta-heuristic algorithm for the simultaneous optimization of the o-d trip matrix estimation. *Computer–Aided Civil and Infrastructure Engineering*, 19:421–435(15).

Vrtic, M., Frohlich, P., Schussler, N., Axhausen, K., Lohse, D., Schiller, C., and Teichert, H. (2007). Two-dimensionally constrained disaggregate trip generation, distribution and mode choice model: Theory and application for a swiss national model. *Transportation Research Part A: Policy and Practice*, 41(9):857–873.

Willumsen, L. (1981). Simplified transport models based on traffic counts. *Transportation*, 10(3):257–258.

Wilson, A. (1970). The use of the concept of entropy in system modelling. *Operational Research Quarterly*.

Wu, J. (1997). A real-time origin-destination matrix updating algorithm for on-line applications. *Transportation Research Part B: Methodological*, 31(5):381–396.

Xu, M., Chen, A., Gao, Z. (2008). An improved origin-based algorithm for solving the combined distribution and assignment problem. *European Journal of Operational Research*, 188(2):354–369.

Yun, S., Sen, A. (2006). Unifying divergence minimization and statistical inference via convex duality. *Learning Theory, Springer, Berlin/Heidelberg*, pages 139–153.

Yun, S., Sen, A. (1994). Computation of maximum likelihood estimates of gravity model parameters. *Journal of Regional Science*, 34(2):199–216.

Apéndice A

# Método de Newton

La idea de este método es la siguiente: se comienza con un valor razonablemente cercano al cero (denominado punto de arranque), entonces se reemplaza la función por la recta tangente en ese valor, se iguala a cero y se despeja (fácilmente, por ser una ecuación lineal). Este cero será, generalmente, una aproximación mejor a la raíz de la función. Luego, se aplican tantas iteraciones como se deseen.

Supóngase $f : [a, b] -> R$ función derivable definida en el intervalo real [a, b]. Empezamos con un valor inicial $x_0$ y definimos para cada número natural n

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Donde $f'$ denota la derivada de $f$.

Una forma de obtener el algoritmo es desarrollando la función $f(x)$ en serie de Taylor, para un entorno del punto $x_n$:

$$f(x) = f(x_n) + f'(x_n)(x - x_n) + (x - x_n)^2 \frac{f''(x_n)}{2!}. + ...$$

Entonces, si se trunca el desarrollo a partir del término de grado 2, y evaluamos en $x_{n+1}$:

$$f(x_{n+1}) = f(x_n) + f'(x_n)(x_{n+1}?x_n)$$

Si además se acepta que xn + 1 tiende a la raíz, se ha de cumplir que $f(x_{n+1}) = 0$, luego, sustituyendo en la expresión anterior, obtenemos el algoritmo.

Finalmente, hay que indicar que el método de Newton-Raphson puede interpretarse como un método de iteración de punto fijo. Así, dada la ecuación $f(x) = 0$, se puede considerar el siguiente método de iteración de punto fijo:

$$g(x) = x + h(x)f(x)$$

Se escoge $h(x)$ de manera que $g'(r) = 0$ (r es la raíz buscada). Dado que g'(r) es:

$$g'(r) = 1 + h'(r)f(r) + h(r)f'(r) = 1 + h(r)f'(r)$$

Entonces:

$$h(r) = \frac{-1}{f'(r)}.$$

Como $h(x)$ no tiene que ser única, se escoge de la forma más sencilla:

$$h(x) = \frac{-1}{f'(x)}.$$

Por tanto, imponiendo subíndices:

$$g(x_n) = x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Expresión que coincide con la del algoritmo de Newton-Raphson

Apéndice B

# Códigos Empleados

## B.1. Código Herramientas.src

```
#include herramientas.ext

/************************
Procedimientos Auxiliares
************************/

proc (1) =agranda(x,n);
local y;
y=x;
for i(1,n-1,1);
    y=y|x;
endfor;
retp(y);
endp;

/************************
procedimiento que calcula
vector de error de las CPO
************************/

proc errorcpo(x);
local Q,f1,f2,f3,i,j;

i=1;
j=0;
Q=zeros(_n*_M,_n);

// matriz de viajes funcional.

for i(1,_M,1);
    j=(i-1)*_n;
    Q[(j+1):(j+_n),.]=(x[(j+1):(j+_n),1].*_Oin[(j+1):(j+_n),1]).*
    exp(x[_n*_M+_n+i,1]*_cijn[(j+1):(j+_n),.]).*
    (x[(_n*_M+1):(_n*_M+_n),1].*_Dj)';
endfor;
f1=sumc(Q')-_Oin;
f2=sumc(Q)-_Dj;
f3=sumacosto(_cijn.*Q)-_COn;
```

```
retp (f1|f2|f3);
endp;

proc (2)= errorcpo2(x);
local Q,f1,f2,f3,i,j,t;
i=1;
j=0;
Q=zeros(_n*_M,_n);
    t=zeros(_n*_M,_n);
// matriz de viajes funcional.
    for i(1,_M,1);
    j=(i-1)*_n;
        t[(j+1):(j+_n),.]=exp(x[_n*_M+_n+i,1]*_cijn[(j+1):(j+_n),.]);
    Q[(j+1):(j+_n),.]=(x[(j+1):(j+_n),1].*_Oin[(j+1):(j+_n),1]).*t[(j+1):(j+_n),.].*(x[(_n*_M+1):(_n*_M+_n),1].*_Dj)';
endfor;
f1=sumc(Q')-_Oin;
f2=sumc(Q)-_Dj;
f3=sumacosto(_cijn.*Q)-_COn;
retp (f1|f2|f3,t);
endp;

proc (2)= errorcpo3(A,B,betan);
local Q,f1,f2,f3,i,j,t;
i=1;
j=0;
Q=zeros(_n*_M,_n);
    t=zeros(_n*_M,_n);
// matriz de viajes funcional.
    for i(1,_M,1);
    j=(i-1)*_n;
        t[(j+1):(j+_n),.]=exp(betan[i,1]*_cijn[(j+1):(j+_n),.]);
    Q[(j+1):(j+_n),.]=(A[(j+1):(j+_n),1].*_Oin[(j+1):(j+_n),1]).*t[(j+1):(j+_n),.].*(B.*_Dj)';
endfor;
f1=sumc(Q')-_Oin;
f2=sumc(Q)-_Dj;
f3=sumacosto(_cijn.*Q)-_COn;
retp (f1|f2|f3,t);
endp;

/************************
procedimiento que calcula
error en costos totales
************************/

proc errorct(x);
local f,Q,i,j;
Q=zeros(_n*_M,_n);
// matriz de viajes funcional.
for i(1,_M,1);
     j=(i-1)*_n;
    Q[(j+1):(j+_n),.]=(_Ain[(j+1):(j+_n),1].*_Oin[(j+1):(j+_n),1]).*
    exp(x[i,1]*_cijn[(j+1):(j+_n),.]).*(_Bj.*_Dj)';
endfor;
f=sumacosto(_cijn.*Q)-_COn;
```

```
retp (f);
endp;

/************************
procedimiento que calcula
la variaza de una matriz
************************/

proc (1) = var(X);
local y,i,j,z;
y=zeros(rows(X)*cols(X),1);
for i(1,cols(X),1);
    j=(i-1)*rows(X);
    y[(j+1):(j+rows(X)),.]=X[.,i];
endfor;
z=vcx(y);
retp(z);
endp;

/************************
procedimiento que calcula
la suma de costos totales por categoría
************************/


proc (1) =sumacosto(X);
local i,y,z,j;
    y=sumc(X');
    z=zeros(_M,1);
    for i(1,_M,1);
        j=(i-1)*_n;
        z[i,1]=sumc(y[(j+1):(j+_n),1]);
    endfor;
    retp (z);
endp;

/************************
procedimiento que calcula
la norma euclideana
************************/

proc (1) = NormaEuclideana(x);

local y;
//solo para vectores x.

y=sqrt(sumc(x.*x));

retp (y);
endp;

/************************
procedimiento que calcula
el gradiente de los costos.
************************/
```

```
proc gradienteCosto(x);
    local z,Q,W,betan,Ain,Bj,s,K,i,j,T;
    {s,K}=eye(_M);
    Q=zeros(_n*_M,_n);
    for i(1,_M,1);
        j=(i-1)*_n;
        Q[(j+1):(j+_n),.]=(_Ain[(j+1):(j+_n),1].*_Oin[(j+1):(j+_n),1]).*
        exp(x[i,1]*_cijn[(j+1):(j+_n),.]).*(_Bj.*_Dj)'.*_cijn[(j+1):(j+_n),.];
    endfor;

    z=sumacosto(Q);
    W=K.*z;

    retp (W);
endp;

/************************
procedimiento que calcula
LAS CPO DE BREGMAN
************************/

proc (2) = F1(x);
local Q,f1,f2,f3,i,j;
i=1;
j=0;
Q=zeros(_n*_M,_n);

// matriz de viajes funcional.
for i(1,_M,1);
j=(i-1)*_n;
Q[(j+1):(j+_n),.]=exp(x[(j+1):(j+_n),1]).*exp(x[_n*_M+_n+i,1]*_cijn[(j+1):(j+_n),.]).*
exp(x[(_n*_M+1):(_n*_M+_n),1])';
endfor;

/* calculo de cada grupo de funciones */
f1=sumc(Q')-_Oin;
f2=sumc(Q)-_Dj;
f3=sumacosto(_cijn.*Q)-_COn;

    retp ((f1|f2|f3),Q);
endp;



/*************************
funcion objetivo de bregman
*************************/

proc fnobjbreg(x);
local cijr,Tr,i;
cijr=_W[.,1:_n];   // traspaso los datos guardados costos por usuario
Tr=_W[.,(_n+1):(2*_n)]; // traspaso los datos guardados de viajes p.u.
i=_W[1,(2*_n+1)]; // vector con indices
```

```
retp (sumc(sumc(Tr.*exp(x*cijr)))-_COn[i,.]);
endp;

/*************************
gradiente de funcion objetivo de bregman
*************************/

proc gfnobjbreg(x);
local cijr,Tr,i;
cijr=_W[.,1:_n];   // traspaso los datos guardados costos por usuario
Tr=_W[.,(_n+1):(2*_n)]; // traspaso los datos guardados de viajes p.u.
i=_W[1,(2*_n+1)]; // vector con indices
retp (sumc(sumc(Tr.*exp(x*cijr).*cijr)));
endp;


/*************************
Crear vector Canonico
*************************/
proc (1) = vectorCanonico(i,n);

local y;

y=zeros(n,1);
y[i,.]=1;

retp (y);
endp;
/*************************
Funcion de Verosimilitud
*************************/

proc Fvero(x);

local Q,f1,f2,f3,i,j,sumQ,_TOn;

i=1;
j=0;
Q=zeros(_n*_M,_n);
sumQ=0;
    _TOn=sumc(_Oin');
// matriz de viajes fu_ncio_nal.

for i(1,_M,1);
    j=(i-1)*_n;
    Q[(j+1):(j+_n),.]=(x[(j+1):(j+_n),1].*_Oin[(j+1):(j+_n),1]).*exp(x[_n*_M+_n+i,1]*
    _cijn[(j+1):(j+_n),.]).*(x[(_n*_M+1):(_n*_M+_n),1].*_Dj)';
endfor;

sumQ=sumc(sumc(Q));
Q=Q./sumQ;

f1=sumc(Q')-_Oin./sumc(_TOn);
f2=sumc(Q)-_Dj./sumc(_TOn);
f3=sumacosto(_cijn.*Q)-_COn./sumc(_TOn);
```

```
retp (f1|f2|f3);
endp;

/*************************
Gradiente Funcion de Verosimilitud
*************************/
proc gradienteFvero(x);

local si,sco,sc,sf,sn,A,B,C,D,E,F,Faux,G,H,L,pijn,pijnc,pijncc,alfain,thetaj,betan,k,j,s,I_n,I_M,I_n_M;

k=1;
j=0;
pijn=zeros(_n*_M,_n);
pijnc=zeros(_n*_M,_n);
pijncc=zeros(_n*_M,_n);

A=zeros(_n*_M,_n*_M);
B=zeros(_n*_M,_n);
C=zeros(_n*_M,_M);
D=zeros(_n,_n*_M);
E=zeros(_n,_n);
F=zeros(_n,_M);
Faux=zeros(_n*_M,_n);
G=zeros(_M,_n*_M);
H=zeros(_M,_n);
L=zeros(_M,_M);

alfain=x[1:_n*_M,1];
thetaj=x[(_n*_M+1):(_n*_M+_n),1];
betan=x[(_n*_M+_n+1):(_n*_M+_n+_M),1];

for k(1,_M,1);
    j=(k-1)*_n;
    pijn[(j+1):(j+_n),.]=exp(alfain[(j+1):(j+_n),.]).*exp(betan[k,1]*_cijn[(j+1):(j+_n),.]).*exp(thetaj)';
    pijnc[(j+1):(j+_n),.]=pijn.*_cijn[(j+1):(j+_n),.];
    pijncc[(j+1):(j+_n),.]=pijnc.*_cijn[(j+1):(j+_n),.];
endfor;

{s,I_n}=eye(_n);
{s,I_M}=eye(_M);
{s,I_n_M}=eye(_n*_M);

sc=sumc(pijn);      //suma sobre i para cada _n pijn.
sf=sumc(pijn');     //suma sobre j para cada i y _n pijn.
sn=sumacosto(pijn); //suma sobre i,j para cada _n pijn.
sco=sumc(pijnc');   //suma sobre j para cada i,_n pijn x _cijn_n.
// construccion del gradiente por bloques matriciales.
//_Matriz A.
A = I_n_M.*sf-sf*sf';
//_Matriz B.
B = pijn-sf*sc';
//_Matriz C.
for k(1,_M,1);
    j=(k-1)*_n;
```

```
    C[(j+1):(j+_n),k] = sco[(j+1):(j+_n),.];
endfor;
C = C - sf*sn';
//Matriz D.
D = pijn' - sc*sf';
//Matriz E.
E = I_n.*sc-sc*sc';
//Matriz F.
for k(1,_M,1);
    j=(k-1)*_n;
    Faux[(j+1):(j+_n),.]=pijn[(j+1):(j+_n),.]*sn[k,.];
endfor;
si=sumc(Faux');
for k(1,_M,1);
    j=(k-1)*_n;
    F[.,k] = sf[(j+1):(j+_n),.]-si[(j+1):(j+_n),.];
endfor;
//_Matriz G.
for k(1,_M,1);
    j=(k-1)*_n;
    G[k,(j+1):(j+_n)] = sco[(j+1):(j+_n),.]';
endfor;
G = G - sn*sf';
//Matriz H.
H = F';
//Matriz L.
// construccion de la matriz gradiente.
retp( (A~B~C)|(D~E~F)|(G~H~L) );
endp;

/*****************************************************************************/
/* Procedimiento que calcula la utilidad lineal para cada componente de Tijn                */
/*****************************************************************************/

proc vlin(alfain,thetaj,betasn);
    local Vsalida;
    Vsalida=vec(_cijn').*(dummybr(seqa(1,1,_m*_n*_n),_n*_n*seqa(1,1,_m))*betasn);
    Vsalida=Vsalida+((dummybr(seqa(1,1,_m*_n*_n),_m*_n*seqa(1,1,_n*_m)))*alfain);
    Vsalida=Vsalida+ agranda(thetaj,_M*_n);

    retp(reshape(Vsalida,_m,_n*_n));
endp;

/*****************************************************************************/
/* Procedimiento que calcula las probabilidades de elección según modelo MNL         */
/*****************************************************************************/

proc PrLogit(alfain,thetaj,betan);
    local vt;
    vt=vlin(alfain,thetaj,betan);
    retp( exp(vt)./sumc(exp(vt)'));
endp;

proc loglik(x,T);
local alfain,thetaj,betan,i,j,T1,P;
```

```
    alfain=x[1:_n*_M,.];
thetaj=x[(_n*_M+1):(_n*_M+_n),.];
betan=x[(_n*_M+_n+1):(_n*_M+_n+_M),.];
T=reshape(T,_m,_n*_n);
P=PrLogit(alfain,thetaj,betan);
retp(sumc((T.*ln(P))'));
endp;

/************************
Procedimientos PRINCIPALES
************************/




/*********************************************

Algoritmo Nuevo 1 : Método de Búsqueda Acotada

*********************************************/
proc  (7)= PIVOTE(r, eps);

local x,t,tc,Q,F1,Ain,Bj,cpromn,cmaxn,lambdan,lambdanux,cminn,i,j,l,betan,betanux,tiempo,convergencia;
   tiempo=hsec;
betan=zeros(_M,1);
Ain=ones(_n*_M,1);
Bj=ones(_n,1);
t=zeros(_n*_M,_n);
tc=zeros(_n,_n);
Q=zeros(_n,_n);
F1=zeros(_n,_n);
cmaxn=zeros(_M,1);
cminn=zeros(_M,1);
cpromn=zeros(_M,1);
lambdan=r;
l=1;
    // busqueda del máximo por categoría de matriz de costo y mínimo por categoría.
for i(1,_M,1);
    j=(i-1)*_n;
    cmaxn[i,.]=maxc(maxc(_cijn[(j+1):(j+_n),.]));
    cminn[i,.]=minc(minc(_cijn[(j+1):(j+_n),.]));
endfor;
    // calculo de la combinación convexa entre c min y c max con theta inicial
cpromn=cminn.*lambdan +(1-lambdan).*cmaxn;
for i(1,_M,1);
    j=(i-1)*_n;
    tc=Ain[(j+1):(j+_n),.].*_Oin[(j+1):(j+_n),.].*_cijn[(j+1):(j+_n),.].*(Bj.*_Dj)';
    betan[i,.]=(1/cpromn[i,.])*ln(_COn[i,.]/sumc(sumc(tc)));
endfor;
x=Ain|Bj|betan;
    convergencia=NormaEuclideana(errorcpo(x));
do until ( NormaEuclideana(errorcpo(x))<= eps );
for i(1,_M,1); // para cada categoria calcula t, para luego usarlo en (56)
    j=(i-1)*_n;
    t[(j+1):(j+_n),.]=exp(betan[i,1]*_cijn[(j+1):(j+_n),.]);
```

```
endfor;

Ain=1./sumc(t'.*Bj.*_Dj);     // Calcula el valor nuevo de Ain
Bj=1./sumc((Ain.*_Oin).*t);     // Calcula el valor nuevo de Bj

lambdanux=lambdan; // guarda valor anterior de theta
betanux=betan;     // guarda valor anterior de beta

for i(1,_M,1);
    j=(i-1)*_n;
    tc=Ain[(j+1):(j+_n),.].*_Oin[(j+1):(j+_n),.].*_cijn[(j+1):(j+_n),.].*(Bj.*_Dj)'; // calculo del denom del log
    Q=tc.*exp(_cijn[(j+1):(j+_n),.].*(1/cpromn[i,.])*ln(_COn[i,.]/sumc(sumc(tc)))); // parte de R
      F1=Q.*_cijn[(j+1):(j+_n),.];                                                  // parte de  R'
             // actualiza theta nuevo
    lambdan[i,.]=lambdanux[i,.]-( sumc(sumc(Q))-_COn[i,.] )/( ln(_COn[i,.]/sumc(sumc(tc)))*
    ( (cmaxn[i,.]-cminn[i,.])/(cpromn[i,.]^2) )*sumc(sumc(F1)) );
    // con la nueva combinacion calcula los nuevos cprom y los nuevos betas.
            cpromn[i,.]=cminn[i,.]*lambdan[i,.]+(1-lambdan[i,.])*cmaxn[i,.];
    betan[i,.]=(1/cpromn[i,.])*ln(_COn[i,.]/sumc(sumc(tc)));
endfor;
        // calcula nuevo x para comprobar el cumplimiento de condiciones de primer orden
x=Ain|Bj|betan;
l=l+1;
    convergencia=convergencia|NormaEuclideana(errorcpo(x));
endo;
tiempo=hsec-tiempo;
retp (Ain,Bj,betan,lambdan,l,tiempo,convergencia);
endp;




/***********************************************

Algoritmo nuevo 2: Pivote

***********************************************/

proc (6) = AlgBeta(B0,r,eps);
local x,Ain1,Bj1,t,tc,Ain,Bj,cmaxn,cminn,cpromn,i,j,l,betan,betanux,tiempo,convergencia;
tiempo=hsec;
betan=ones(_M,1);
Ain=ones(_n*_M,1);
Ain1=ones(_n*_M,1);
Bj1=ones(_n,1);
t=ones(_n*_M,_n);
tc=zeros(_n,_n);
Bj=B0;
cmaxn=zeros(_M,1);
cminn=zeros(_M,1);

for i(1,_M,1);
    j=(i-1)*_n;
    cmaxn[i,.]=maxc(maxc(_Cijn[(j+1):(j+_n),.]));
    cminn[i,.]=minc(minc(_Cijn[(j+1):(j+_n),.]));
```

```
endfor;

cpromn=r.*cminn+(1-r).*cmaxn;

l=1;

x=Ain|Bj|betan;
convergencia=NormaEuclideana(errorcpo(x));
do until ( NormaEuclideana(errorcpo(x))<= eps );
    convergencia=convergencia|NormaEuclideana(errorcpo(x));
for i(1,_M,1);
    j=(i-1)*_n;
    t[(j+1):(j+_n),.]=exp(betan[i,1]*_Cijn[(j+1):(j+_n),.]);
endfor;
Ain1=Ain;                /* Guarda el valor anterior de a */
Ain=1./sumc(t'.*Bj.*_Dj);     /* Calcula el valor nuevo de a */
Bj1=Bj;                        /* Guarda el valor anterior de b */
Bj=1./sumc((Ain.*_Oin).*t);     /* Calcula el valor nuevo de b */
betanux=betan;
for i(1,_M,1);
    j=(i-1)*_n;
    tc=Ain[(j+1):(j+_n),.].*_Oin[(j+1):(j+_n),.].*exp( betan[i,.]*
    (_Cijn[(j+1):(j+_n),.]-cpromn[i,.]) ).*(Bj.*_Dj)'.*_Cijn[(j+1):(j+_n),.];
    betan[i,.]=(1/cpromn[i,.])*ln( _COn[i,1]/sumc(sumc(tc)) );
endfor;
x=Ain|Bj|betan;
l=l+1;
endo;
tiempo=hsec-tiempo;
retp (Ain,Bj,betan,l,tiempo,convergencia);
endp;

/*************************
    Método de Bregman
*************************/
proc (6) = Bregman(x0,eps);

local Ain,Bj,alfain,thetaj,betan,l,r,tiempo,i,k,j,T,x,Tr,cijr,lambdak,ret,err,convbreg;
k=0;
convbreg=0;
T=zeros(_n*_M,_n);  //matriz extendida de viajes
Tr=zeros(_n,_n);  //matriz de nxn para un tipo de usuario
cijr=zeros(_n,_n);  //cijr
l=0;
i=0;
j=0;

alfain=x0[1:_n*_M,.];  // extrae los multiplicadores de lagrange asociados a origen
thetaj=x0[(_n*_M+1):(_n*_M+_n),.];  // extrae los multiplicadores asocuaios a destino
betan=x0[(_n*_M+_n+1):(_n*_M+_n+_M),.];  // multiplicadores asociados a costos

lambdak=0;

x=x0;  // guarda el valor inicial del set de multiplicadores
tiempo=hsec;   // registra el tiempo inicial
```

```
    {err,T}=F1(x);
    convbreg=NormaEuclideana(err);
do until (NormaEuclideana(err)<=eps);
//ciclo mientras la evaluacion de F1 de los multiplicadores sea menor que eps
//cuyo contador global es k
l=recserrc(k,(_n*_M+_n+_M))+1;
if (l<=_n*_M);
lambdak=ln( _Oin[l,.]/sumc(T[l,.]') );
elseif (l>_n*_M) and (l<=(_n*_M+_n));
j=l-_n*_M;
lambdak=ln( _Dj[j,.]/sumc(T[.,j]) );
else;
r=l-(_n*_M+_n);
j=(r-1)*_n;
cijr=_cijn[(j+1):(j+_n),.];
Tr=T[(j+1):(j+_n),.].*cijr;

    _W=cijr~Tr~(ones(_n,1)*r);
            _eqs_IterInfo=0;
           _eqs_JacobianProc=&gfnobjbreg;
            __output=0;
            {lambdak,ret}=eqSolve(&fnobjbreg,0);

endif;

x=x+vectorCanonico(l,(_M*_n+_n+_M))*lambdak;
k=k+1;
{err,T}=F1(x);
        convbreg=convbreg|NormaEuclideana(err);

endo;

alfain=x[1:_n*_M,.];
thetaj=x[(_n*_M+1):(_n*_M+_n),.];
betan=x[(_n*_M+_n+1):(_n*_M+_n+_M),.];

tiempo=hsec-tiempo;
/* retorna variables*/

Ain=exp(alfain)./_Oin;
Bj=exp(thetaj)./_Dj;

retp (Ain,Bj,betan,k,tiempo,convbreg);
endp;


/************************************
METODO MART
************************************/


proc (6) = MART(x0,eps);

local Ain,Bj,alfain,thetaj,betan,l,r,tiempo,i,k,j,T,x,Tr,cijr,lambdak,err,converg;
```

```
    converg=0;
k=0;
T=zeros(_n*_M,_n);  //matriz extendida de viajes
Tr=zeros(_n,_n);    //matriz de nxn para un tipo de usuario
cijr=zeros(_n,_n);  //cijr matriz de costo para categoria de usuario

l=0;
i=0;
j=0;

alfain=x0[1:_n*_M,.];
thetaj=x0[(_n*_M+1):(_n*_M+_n),.];
betan=x0[(_n*_M+_n+1):(_n*_M+_n+_M),.];

lambdak=0;

x=x0;
    tiempo=hsec;
    {err,T}=F1(x);


//Ejecucion del Algoritmo.

do until (NormaEuclideana(err)<=eps);

l=recserrc(k,(_n*_M+_n+_M))+1;
/*************************************/
/*       Restricciones Origen.       */
/*************************************/
if (l<=_n*_M);
            lambdak=ln( _Oin[l,.]/sumc(T[l,.]') );
elseif (l>_n*_M) and (l<=(_n*_M+_n));
    j=l-_n*_M;
    lambdak=ln( _Dj[j,.]/sumc(T[.,j]) );
else;
    r=l-(_n*_M+_n);
    j=(r-1)*_n;
    Tr=T[(j+1):(j+_n),.].*_cijn[(j+1):(j+_n),.];
    lambdak=ln( _COn[r,.] / sumc(sumc(Tr)) );
endif;

x=x+vectorCanonico(l,(_M*_n+_n+_M))*lambdak;

k=k+1;
        {err,T}=F1(x);

endo;


alfain=x[1:_n*_M,.];
thetaj=x[(_n*_M+1):(_n*_M+_n),.];
betan=x[(_n*_M+_n+1):(_n*_M+_n+_M),.];

tiempo=hsec-tiempo;
Ain=exp(alfain)./_Oin;
```

```
Bj=exp(thetaj)./_Dj;


retp (Ain,Bj,betan,k,tiempo,converg);
endp;
/*************************************************************************************
 METODO PUNTO FIJO Y NEWTON COMPLETO
*************************************************************************************/
proc (5) = AlgEnt2(B0,betan,eps);
local x,t,err,Ain,Bj,Ain1,Bj1,j,r,tiempo,convergencia,q;
tiempo=hsec;        /* toma tiempo inicial */
Ain=ones(_n*_M,1);
Ain1=ones(_n*_M,1);
Bj1=ones(_n,1);
Bj=B0;
/***********************************************
La notacion es Ain y Bj son las variables nuevas
***********************************************/
x=Ain|Bj|betan;
t=ones(_n*_M,_n);
{err,t}=errorcpo2(x);
convergencia=err;
do until( NormaEuclideana(err)<= eps );
   /* Calculo de coeficientes A y B para Entropia, con beta dado */
Ain1=Ain; /* Guarda el valor anterior de a */
Ain=1./sumc(t'.*Bj.*_Dj);   /* Calcula el valor nuevo de a */
Bj1=Bj;                 /* Guarda el valor anterior de b */
Bj=1./sumc((Ain.*_Oin).*t); /* Calcula el valor nuevo de b */
_Ain=Ain;                 //guardo variables globales para procedimiento
_Bj=Bj;                     //guarda variable global.

_eqs_JacobianProc=&gradienteCosto;
__output=0;

{betan,q}=eqSolve(&errorct,betan); // calcula nuevo betan, dado los parametros nuevos
x=Ain|Bj|betan;     // guarda variable para chequear condiciones de primer orden
{err,t}=errorcpo2(x);  // llama a procedimiento para verificar condiciones de primer orden.
convergencia=convergencia|err;      // guarda evolución
endo;

tiempo=hsec-tiempo;
retp (Ain,Bj,betan,convergencia,tiempo);
endp;

/*************************************************************************************
Algoritmo de Hyman
*************************************************************************************/
proc (5) = Hyman(betan,B0,eps,Cmedio);
local x,t,Ain,Bj,C1,C2,i,j,beta1,beta2,contador,tiempo,err,iter;
tiempo=hsec;        /* toma tiempo inicial */
contador=0;     /* numero total de iteraciones */
i=0;
j=0;

Ain=ones(_n*_M,1);
```

```
Bj=ones(_n,1);
t=ones(_n*_M,_n);     // matriz de parametros para viajes, inicialmente nula

_Bj=B0;             // Se establece Bj como el B0 (externo)
C2=Cmedio; //inicializo Primer Costo total es el Inicial

{err,t}=errorcpo3(Ain,Bj,betan);
iter=err;  // guardo valore de cumplimiento de CPO inicial.

do until( NormaEuclideana(err)<= eps );
_Ain=1./sumc(t'.*_Bj.*_Dj);  // Calcula a0, dado t, Bj y _Dj '
Bj=_Bj;  // guarda valor anterior de bj
_Bj=1./sumc((_Ain.*_Oin).*t);  // Calcula el valor nuevo de b
Ain=_Ain;  // guarda valor anterior de ain
_Ain=1./sumc(t'.*_Bj.*_Dj);     // '
    C1=C2;  // guarda C anterior
for i(1,_M,1); // calculamos el nuevo costo medio C2 para todas las categorias i.
        j=(i-1)*_n;
C2[i,.]=sumc(sumc(_cijn[(j+1):(j+_n),.].*((_Ain[(j+1):(j+_n),1].*_Oin[(j+1):(j+_n),1]).*
exp(_cijn[(j+1):(j+_n),.].*betan[i,1]).*(_Bj.*_Dj)'))/sumc(sumc((_Ain[(j+1):(j+_n),1].*
_Oin[(j+1):(j+_n),1]).*exp(_cijn[(j+1):(j+_n),.].*betan[i,1]).*(_Bj.*_Dj)')); //'

    endfor;
/*************************************************************/
/*  A este nivel, tenemos a y b dado beta, actualizamos beta   */
/*************************************************************/
    if contador < 1; // C* esta guardado en C1 inicial, C2 es calculado con Ai,Bj dado.
        beta1=betan;  // guardo primer valor de beta en beta1.
betan=betan.*Cmedio./C2;  //actualizo valor de beta
else;
    beta2=betan;  // betan actualizada (k+1)
    betan=((Cmedio-C1).*betan-(Cmedio-C2).*beta1)./(C2-C1); // beta2 es k se lo paso a beta1 para el prox paso.
    beta1=beta2;
    endif;
    {err,t}=errorcpo3(Ain,Bj,betan);
iter=iter|err; //guardo el valor del cumplimiento de las cpo para referencia
contador=contador+1;
endo;
tiempo=hsec-tiempo;
retp (_Ain,_Bj,betan,contador,tiempo,iter);
endp;
```

## B.2. Código Herramientas.dec

```
declare matrix _Ain;

declare matrix _Bj;

declare matrix _beta;

declare matrix _Oin;

declare matrix _Dj;

declare matrix _cijn;
```

```
declare matrix _COn;

declare matrix _n;

declare matrix _M;

declare matrix _W=0;
```

## B.3. Código Herramientas.ext

```
external matrix _Ain;

external matrix _Bj;

external matrix _beta;

external matrix _Oin;

external matrix _Dj;

external matrix _cijn;

external matrix _COn;

external matrix _n;

external matrix _M;

external matrix _W;
```