UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

Una metodología para encontrar puntos críticos
de la función de Gibbs

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Matemático

PRESENTA:

Juan Andrés Ramírez Segundo

TUTOR

M. en C. César Carreón Otañez

Ciudad Universitaria, CD. MX., 2022

Agradecimientos

Me llena de recuerdos todo lo que he vivido para poder llegar a este momento, por lo que dedico algunas palabras para agradecer a aquellas personas que conocí y me apoyaron a lo largo del camino.

A mi mamá, porque sin ti no sería la persona que soy en este momento, gracias por cada uno de los esfuerzos que has hecho por mí, el camino no ha sido nada fácil, sin embargo, siempre has estado ahí para apoyarme, cuidarme, darme una palabra de aliento y enseñarme a nunca rendirme, al final esto es más tuyo que mio. ¡Gracias¡

A Celia, Eric y Jony porque son mi familia y siempre han estado ahí para apoyarme cuando más lo necesito, gracias por los momentos de felicidad que hemos compartido.

A mi tutor el M. en C. César Carreón por ser mi guía en esta última etapa de mi carrera en la Facultad, aún más le agradezco por el apoyo, paciencia y tiempo para la realización de este proyecto. Mis respetos y admiración para usted.

Agradezco a mis sinodales el Dr. Pedro González, el Dr. Mico Djurdjevic, la M. en C. María Velasco y la M. en C. Brenda Zavala, por sus valiosos comentarios en el proceso de revisión de este trabajo.

A la Dra. Leticia Campos por el apoyo, confianza, la oportunidad de desarrollarme profesionalmente y principalmente por el aprendizaje en este tiempo. Además un agradecimiento al equipo, Francisco, Eduardo, Puga, David y Lore por los momentos compartidos y las enseñanzas que hemos tenido.

A las grandes personas que he conocido en este camino Karen, Marisol, Alma, Jona, Itzel, Lalo, Daniel, Rodrigo por las anécdotas y consejos, hay mucho que recordar y por lo cual sonreir.

A la Facultad de Ciencias y a la UNAM por todas las experiencias y aprendizajes que me han permitido desarrollarme como ser humano y profesionista.

Índice general

Resumen

En diversas operaciones industriales como lo es la petrolera existen diversos comportamientos de fases de los fluidos y mezclas que surgen de la extracción del crudo. Conocer si la fase de los fluidos es estable o no, es de gran importancia para conocer la forma en que sean tratados, transportados y procesados. La información se utiliza para evaluar las reservas de un yacimiento para diseñar y construir plataformas adecuadas para el manejo optimo de los fluidos.

Dada la composición de una mezcla para un fluido que se somete a una temperatura y presión para su manejo es necesario que se tenga un equilibrio de fases, es decir, saber si se encuentran en un equilibro Líquido - Vapor, Líquido - Líquido o si encuentran en fase gaseosa o aceite. Michelsen propone resolver dicho problema determinando si una fase dada es termodinámicamente estable o no, encontrando el mínimo de la Energía de Gibbs de dicha mezcla, dado el mínimo podremos saber si la fase es estable o no.

Se proponen diversos métodos de optimización para encontrar el mínimo de la función objetivo, en el capítulo 2 se describen diversos métodos de optimización que se dividen en métodos directos como la sección áurea, interpolación parabólica que solo utilizan las evaluaciones de la función objetivo y los métodos indirectos como el método de Newton y secante que utilizan las condiciones necesarias y suficientes para encontrar el mínimo de la función objetivo. El capítulo 3 se desarrollan los métodos de optimización en $\mathbb{R}^n$ que son el método del gradiente y gradiente conjugado, además del método de Newton como son de Quasi-Newton que buscan encontrar una aproximación numérica de la matriz Hessiana.

Los tiempos computacionales para encontrar los puntos críticos pueden ser largos si el punto inicial no se encuentran en una vecindad cercada al mínimo global, por lo que en el capítulo 4 se describe una propuesta para encontrar nuevos puntos iniciales que permitan hallar en tiempos más cortos un puto crítico. En el capítulo se describen los resultados que se obtuvieron utilizando una propuesta para encontrar los mínimos de la energía de Gibbs para distintas mezclas. Por último en el capítulo 5 se pueden observar las rutinas de cada uno de los métodos descritos en el trabajo, dichos códigos fueron compilados en el Software numérico Scilab.

Capítulo 1

Energía de Gibbs

El petróleo es una de las principales fuentes de energía que es utilizada para la producción de diversos hidrocarburos, como lo son el gas natural, gas licuado de petróleo, diversas gasolinas, combustóleo, entre otros. Son utilizados para el transporte, la industria, así como también para la generación de energía eléctrica. Cerca del 88 % de la energía que se consume en México proviene de dicho energético. Por lo que es de gran importancia el hacer una buena planeación para la exploración y extracción del mismo.

Los fluidos en una reserva de petróleo son mezclas naturales de gas natural y de crudo que, existen bajo temperaturas y presiones elevadas, las composiciones del líquido del yacimiento incluyen cientos o miles de hidrocarburos además de nitrógeno, CO_2 y sulfuro de hidrógeno, sus propiedades físicas dependen principalmente de su composición, de las condiciones de temperatura y presión en las que se encuentren.

Mientras se extraen el petróleo y gas, la presión del yacimiento disminuye y las mezclas de hidrocarburos restantes cambian en su composición, también sus propiedades volumétricas y comportamientos de fase, es decir, se pueden encontrar en un estado líquido, gaseoso o vapor. El comportamiento de fase de uno o dos componentes pueden ser útiles para describir los efectos de la presión, temperatura y su composición.

Dichos fluidos se pueden dividir en cinco categorías: gas seco, gas húmedo, condesado de gas, aceite volátil y aceite negro. La distribución de los componentes en un fluido de reserva se determina conociendo que tan cerca están de su punto crítico.

Para que se pueda lograr un equilibrio se necesita que no se produzca una interferencia de energía entre cada una de las fases, es decir, las temperaturas y presiones deben ser las mismas.

En distintas operaciones industriales como la extracción, la absorción o destilación del petróleo se debe conocer el comportamiento de fases para los fluidos o mezclas, es necesario conocer la variación de energía que ocurre en cada intercambio de fases.

La ciencia que se encarga del estudio de las transformaciones de la energía es la Termodinámica, en ella se estudia el intercambio de energía en sus diversas formas, las propiedades de la meteria y el uso racional de la energía. La segunda ley de la termodinámica expresa que, en los procesos espontáneos se pasa de estados de menor entropía a estados de mayor entropía, por lo que no puede haber una transferencia espontánea de calor de frío a caliente.

Se entiende por Entropía al grado de dispersión de la energía que, definiremos como S. Las unidades de la entropía son $\frac{joule}{kelvin}$, es decir, la entropía se entiende como la variación que experimenta un sistema cuando absorbe el calor de un joule a la temperatura de un grado kelvin. Como ejemplo se tiene que el estado gaseoso que es un sistema desordenado, en el que la dispersión de la energía llega a un extremo, debido a que las moléculas en el gas se desplazan a altas velocidades y en distintas direcciones, por lo que es el estado que más dispersión energética tiene.

Por otra parte, se entiende a la Entalpía como la cantidad de calor que un sistema termodinámico libera o absorbe del entorno que lo rodea cuando se encuentra a presión constante, es decir, la entalpía es la suma de la energía interna de la materia y el producto de su volumen por la presión, por lo que de finiremos a la entalpía H y la unidad de medida es el Joule (J) y la fórmula para obtenerla es la siguiente:

$$H = U + pV$$

Donde definimos a U como la energía interna, p como la presión y V como el volumen.

En el caso de la extracción de crudo y de gas natural, se tiene un sistema que se encuentra a una temperatura y presión constante en el cambio de sus fases, es decir, ocurre un cambio de energía libre de un sistema que va de un estado inicial a un estado final, por lo que ha dicho fenómeno se le conoce como la energía libre de Gibbs.

Sin embargo, otra propiedad en la termodinámica que se puede utilizar para predecir si un proceso se produce de forma espontánea, es la energía libre de Helmholtz. La diferencia con la energía libre de Gibbs, es que la energía libre de Helmholtz ocurre en condiciones de volumen y temperatura constante y el proceso de extracción de crudo es diferente ya que se puede tener un volumen distinto en cada una las fases.

La energía de Gibbs se define como:

$$G = H - TS,$$

donde H es la entalpia, T la temperatura y S es la entropía.

Una forma de resolver el problema del equilibrio de fases es mediante un análisis de estabilidad en el que se considera una mezcla de N_c componentes a una temperatura,

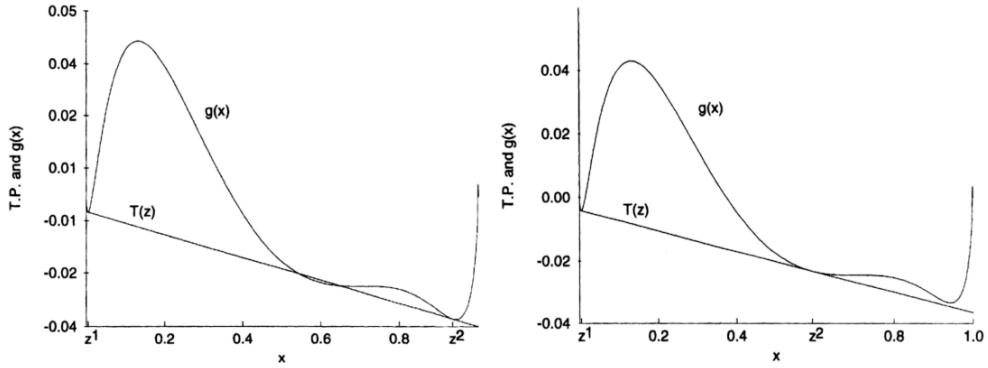
presión y composición $(z_1, z_1, ..., z_{N_c})$. Asumiendo que la mezcla se divide en dos fases y que el número de moles de la nueva fase tiene una cantidad infinitesimal η, con una fracción de mol $(y_1, y_2, ..., y_{N_c})$.

Denotamos a G_0 como la energía de Gibbs para la mezcla, denotando a G_1 y G_2 como las energías de Gibbs del segundo estado del sistema. Además se considera que el cambio de energía de Gibbs es la diferencia entre el estado inicial y el estado final del sistema, es decir:

$$\Delta G = (G_1 + G_2) - G_0.$$

Michelsen propone resolver el análisis de estabilidad mediante el plano tangente de la energía de Gibbs, el cual requiere de encontrar un mínimo global. A partir de ubicar dicho mínimo se evalúa en la energía de Gibbs si el resultado es un número negativo, la nueva fase genera una perturbación hacía el estado de equilibrio más estable. Por lo que el cambio de fase será estable si al evaluar el mínimo en la Energía de Gibbs es un número positivo.

Sea una mezcla de 2 componentes, como el que se ve en la *figura 1.1* donde $g(x)$ es la curva de la energía libre molar de Gibbs y sea x una fracción molar de uno de los componentes de la mezcla.



(a) Plano tangente por debajo de un punto de la superficie de Gibbs

(b) Plano tangente por debajo de la superficie de Gibbs

Figura 1.1: `Plano tangente de Gibbs`

En la *figura 1.1*, la fracción molar del segundo componente está dado por $1-x$. Sea una solución de dos fases que corresponde al mínimo local de la función de la energía libre de Gibbs, tiene la fracción molar del componente uno en la fase indicada por z^1 y la fase dos por z^2, que está asociado por los potenciales químicos $\mu^0(z^1)$ y $\mu^0(z^2)$ respectivamente.

Dicha solución local satisface la condición necesaria del primer orden de potenciales químicos iguales $\mu^0(z^1) = \mu^0(z^2)$, si la tangente de la superficie se traza sobre z^1 y z^2, dicha recta corresponde al plano tangente de Gibbs y la denotamos como $T(z)$, por lo que es asociada a una solución local, gráficamente se muestra que dicha línea se encuentra por encima de la superficie de Gibbs en $y = .6$ lo que indica una inestabilidad termodinámica.

Por lo que se debe buscar una solución global para z^1 y z^2 distinta a la anterior. En la *figura 1.1 (b)* se puede observar que la tangente de Gibbs se muestra por de bajo de la superficie de Gibbs. Por lo que si existe un plano tangente que se encuentre por debajo de la superficie de Gibbs, entonces la solución de equilibrio correspondiente al mínimo global de la energía libre de Gibbs está dada por las composiciones en los puntos de tangencia.

La propuesta es que a partir del plano tangente que en la *figura 1.1* se puede observar por $F(y)$, la cual debe ser positivo sobre todo el intervalo para que sea estable y $F(y)$ será positiva. Entonces se tiene que:

$$g(y) = 1 + \sum_{i=1}^{N_c} y_i(\ln(y_i) + \ln(\phi_i) - h_i - 1) \geq 0.$$

La ecuación $g(y)$ es el criterio de estabilidad, por lo que se le denominará como la función objetivo. A partir de aquí se busca encontrar el mínimo y evaluarlo para conocer si la mezcla es estable o no.

Las variables y_i cumplen que $y_i \geq 0$ y que $\sum_{i=1}^{N_c} y_i$, dicha variable es la energía molar de Gibbs. Por otra parte ϕ_i es la constante de fugacidad y h_i es la fracción de mol de la mezcla.

Capítulo 2

Optimización en una dimensión

Un problema de optimización se puede expresar matemáticamente como la búsqueda de determinar un argumento para el cual una función dada tiene un valor extremo (mínimo o máximo) en un dominio.

Sea una función $f : \mathbb{R}^n \longmapsto \mathbb{R}$ y un vector de n variables $x = (x_1, x_2, x_3, ..., x_n)$. Se le denominará a f como la función objetivo, donde se busca que x sea el mínimo de dicha función.

Definiremos un punto mínimo y sus diferentes características así como también, lo que es un punto crítico y un punto silla.

DEFINICIÓN. *Sea $f : \mathbb{R}^n \longmapsto \mathbb{R}$ y sean $x^*, x \in \mathbb{R}^n$, x^* será un mínimo local si para cualquier x cerca de x^* se cumple que $f(x^*) \leq f(x)$.*

DEFINICIÓN. *Sea $f : \mathbb{R}^n \longmapsto \mathbb{R}$ y $x^*, x \in \mathbb{R}^n$ x^* será un mínimo global si para cualquier x en el dominio se cumple que $f(x^*) \leq f(x)$.*

Como se muestra gráficamente en la *figura 2.1* pueden existir varios mínimos locales a partir de la vecindad donde se busque el mínimo.

Sea $U \subseteq \mathbb{R}^n$, donde U es un conjunto definido por un sistema de ecuaciones y desigualdades llamadas restricciones, éstas pueden ser lineales o no lineales, si $U = \mathbb{R}^n$ el problema es no restringido. Los vectores $x \in U$ que satisfacen las restricciones se llaman puntos factibles.

DEFINICIÓN. *Sea $f : U \longmapsto \mathbb{R}$, x^* será un punto crítico o estacionario, si cumple con que el gradiente de la función es 0, es decir, $\nabla f(x^*) = 0$.*

DEFINICIÓN. *Sea una función $f : U \longmapsto \mathbb{R}$, f será una función suave si todas sus derivadas parciales de cualquier orden existen.*

A lo largo de los métodos descritos en el siguiente capítulo, se trabajarán con funciones suaves que tengan primera y segunda derivada continua denotada como

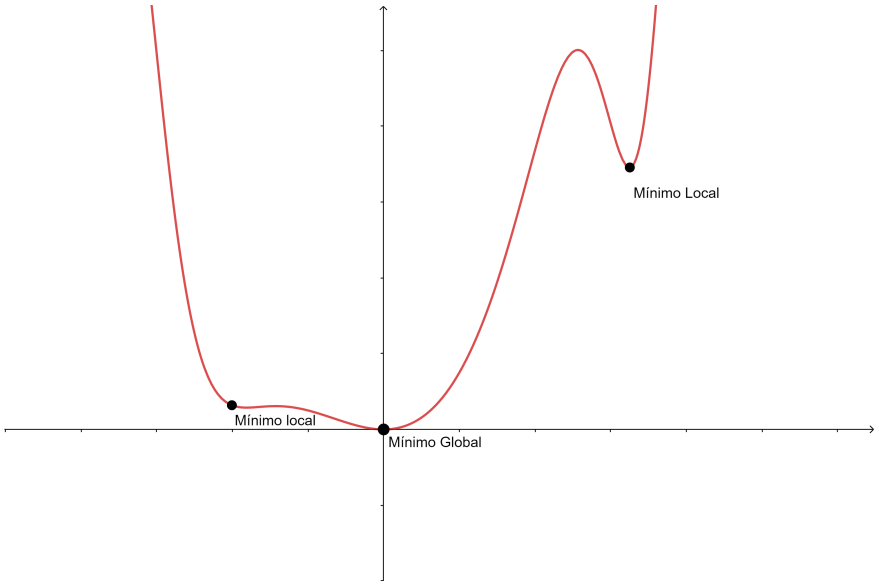


Figura 2.1: `Mínimos locales y Mínimo Global`

$f(x) \in C^2$, es decir, necesitaremos información del gradiente y del Hessiano de la función.

Tendremos que si $f : \mathbb{R}^n \longmapsto \mathbb{R}$ es diferenciable, el vector evaluado en la función $\bigtriangledown f : \mathbb{R}^n \longmapsto \mathbb{R}$, se le denominará como el gradiente de f y se define como:

$$\bigtriangledown f(x) = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n} \right]^t.$$

Por otra parte si tenemos una función $f : \mathbb{R}^n \longmapsto \mathbb{R}$ dos veces diferenciable, la matriz evaluada en la función $H(f(x)) : \mathbb{R}^n \longmapsto \mathbb{M}_{nxn}(\mathbb{R})$, se le denominará como la matriz Hessiana de f. La matriz Hessiana es la matriz Jacobiana del gradiente de la función $\bigtriangledown f(x)$. Más aún si la segunda derivada parcial de f es continua entonces:

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i},$$

es decir, el Hessiano de f es una matriz simétrica.

$$Hf(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}$$

Si x^* es un punto crítico entonces se cumple que el $\bigtriangledown f(x^*) = 0$. Si $H(f(x^*))$ es definido positivo en x^*, entonces x^* es un mínimo local de f. Por otra parte si x^* es un punto crítico y el $H(f(x^*)) = 0$, entonces x^* es un punto silla de f.

Como hemos visto el gradiente y el Hessiano de la función $f(x)$ satisfacen ciertas condiciones para el punto mínimo x^*.

Teorema 1. *Condición Necesaria*
Si f es dos veces continuamente diferenciable y x^* es un mínimo local de f. Entonces:

$$\bigtriangledown f(x^*) = 0,$$

y $\bigtriangledown^2 f(x^*)$ es definida positiva.

Demostración. Sea $u \in \mathbb{R}^n$ usando Teorema de Taylor para una t suficientemente pequeña y usando que f es dos veces diferenciable, entonces

$$f(x^* + tu) = f(x^*) + t \bigtriangledown f(x^*)^T u + \frac{t^2}{2} u^T \bigtriangledown^2 f(x^*) u + \circ(t^2).$$

Por hipótesis x^* es un mínimo local entonces:

$$f(x^*) \leq f(x),$$

para cualquier x cercana a x^*, en nuestro caso:

$$f(x^*) \leq f(x^* + tu),$$

$$\Rightarrow f(x^*) \leq f(x^* + tu) = f(x^*) + t \bigtriangledown f(x^*)^T u + \frac{t^2}{2} u^T \bigtriangledown^2 f(x^*) u + o(t^2),$$

restando $f(x^*)$

$$0 \leq f(x^* + tu) - f(x^*) = t \bigtriangledown f(x^*)^T u + \frac{t^2}{2} u^T \bigtriangledown^2 f(x^*) u + o(t^2),$$

dividiendo entre t

$$0 \leq \bigtriangledown f(x^*)^T u + \frac{t}{2} u^T \bigtriangledown^2 f(x^*) u + o(t),$$

para toda t suficientemente pequeña y toda $u \in \mathbb{R}^n$. Si se establece que $t = 0$ y $u = -\bigtriangledown f(x^*)$

$$0 \leq (\bigtriangledown f(x^*)^T)(-\bigtriangledown f(x^*)) + \frac{0}{2} - \bigtriangledown f(x^*)^T(\bigtriangledown^2 f(x^*))(-\bigtriangledown f(x^*)) + o(t),$$

por lo que

$$\Rightarrow \|\bigtriangledown f(x^*)\|^2 = 0,$$

por definición de norma $\bigtriangledown f(x^*) = 0$.

Retomando

$$0 \leq f(x^* + u) - f(x^*) = t \bigtriangledown f(x^*)^T u + \frac{t^2}{2} u^T \bigtriangledown^2 f(x^*) u + o(t^2),$$

como $\bigtriangledown f(x^*) = 0$, dividiendo entre t^2 y tomando t suficientemente pequeña

$$\Rightarrow 0 \leq \frac{1}{2} u^T \bigtriangledown^2 f(x^*) u,$$

$$\therefore 0 \leq u^T \bigtriangledown^2 f(x^*) u.$$

□

La condición de que el $\bigtriangledown f(x^*) = 0$ es solamente la condición necesaria de primer orden y el punto que satisface dicha condición es un punto estacionario o un punto crítico.

Un punto estacionario no necesariamente es un punto mínimo, se necesita que la segunda derivada sea estrictamente positiva para que se tenga un mínimo.

Teorema 2. *Condición Suficiente.*
Sea f dos veces continuamente diferenciable en una vecindad de x^*. Si $\bigtriangledown f(x^*) = 0$ y $u^t \bigtriangledown^2 f(x^*) u \geq 0$. Entonces x^* es un mínimo local de f.

Demostración. Sea $u \in \mathbb{R}^n, u \neq 0$ usando el Teorema de Taylor para t suficientemente pequeña

$$f(x^* + u) = f(x^*) + t \bigtriangledown f(x^*)^T u + \frac{t^2}{2} u^T \bigtriangledown^2 f(x^*) u + \circ(t^2).$$

Por hipótesis tenemos que $\bigtriangledown f(x^*) = 0$,

$$\Rightarrow f(x^* + u) = f(x^*) + \frac{t^2}{2} u^T \bigtriangledown^2 f(x^*) u + o(t^2).$$

Se toma como λ el valor propio más pequeño de $\bigtriangledown^2 f(x^*)$, entonces podemos expresarlo como $\bigtriangledown^2 f(x^*) u = \lambda u$.

$$\Rightarrow f(x^* + u) = f(x^*) + \frac{t^2}{2} u^T \lambda u + o(t^2),$$

tenemos que $\frac{t^2}{2} u^T \lambda u \geq \frac{\lambda}{2} \|tu\|^2$, ya que $\|tu\| = \sqrt{t \sum_{i=1}^{n} u_i^2}$.

$$\Rightarrow f(x^* + u) - f(x^*) \geq \frac{\lambda}{2} \|tu\|^2 + o(t^2) > 0.$$

Además tomando t suficientemente pequeña se cumple que:

$$\therefore f(x^* + u) \geq f(x^*).$$

□

El primer acercamiento que se desarrollará es la optimización para funciones en una dimensión, es decir, $f : \mathbb{R} \longrightarrow \mathbb{R}$, el objetivo es encontrar si existen los mínimos de la función por lo cual se exploran distintos métodos que exigen ciertas condiciones sobre la función objetivo.

2.1. Optimización en Una dimensión

Una función $f : \mathbb{R} \longrightarrow \mathbb{R}$ es unimodal en un intervalo $[a,b]$ si existe un único valor $x^* \in [a,b]$, donde $f(x^*)$ es un mínimo de f en $[a,b]$ y para cualquier $x_1, x_2 \in [a,b]$ con $x_1 < x_2$ ocurre que:

$$x_2 < x^*$$

entonces

$$f(x_1) > f(x_2)$$

y

$$x_1 > x^*$$

entonces

$$f(x_1) < f(x_2),$$

es decir, $f(x)$ es estrictamente decreciente si para cualquier $x \in [a,b]$ se cumple que $x \leq x^*$ y $f(x)$ es estrictamente creciente si $x \geq x^*$.

Observamos que si se cumple

$$x_2 < x^*$$

y por hipótesis

$$x_1 < x_2 \Rightarrow x_1 < x_2 < x^*,$$

de las condiciones requeridas en la *figura 2.2 (a)* se muestra un ejemplo en el que se cumple:

$$f(x_1) > f(x_2).$$

Por otra parte si se cumple que

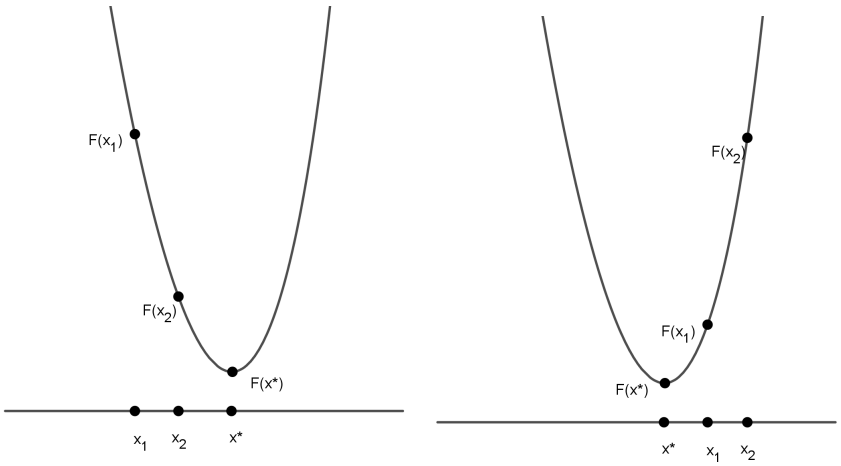
$$x_1 > x^*,$$

y por hipótesis

$$x_1 < x_2 \Rightarrow x_2 > x_1 > x^*,$$

además en la *figura 2.2 (b)* podemos observar que

$$f(x_1) < f(x_2).$$



(a) $x_2 < x*$ (b) $x_1 > x*$

Figura 2.2: `Función Unimodal`

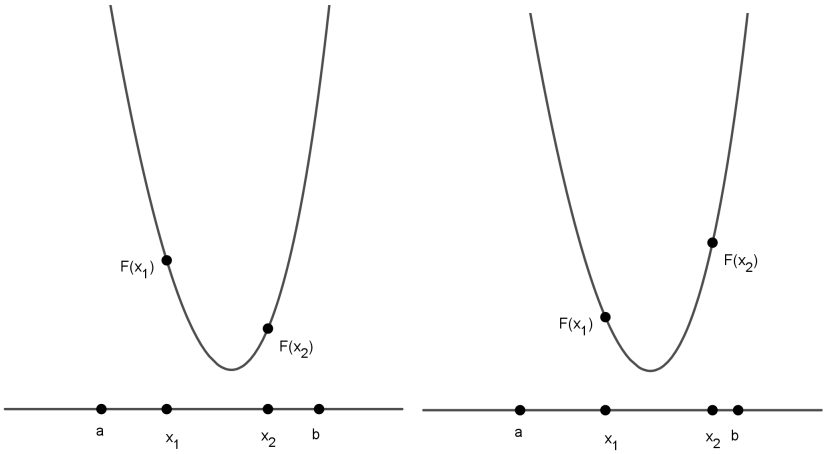
Una función unimodal permite asegurar la existencia de un mínimo, además un intervalo que contenga una solución para que a su vez descarta regiones del intervalo de acuerdo a las evaluaciones de la función.

2.1.1. Sección Áurea

El método de la sección áurea permite encontrar un mínimo de una función $f : \mathbb{R} \longrightarrow \mathbb{R}$ a partir de un intervalo dado encontrando dos nuevos puntos, después evaluarlos en la función y utilizando las propiedades de la unimodalidad poder descartar una región del intervalo para así poder encontrar el mínimo.

Supongamos que f es una función unimodal en el intervalo $[a,b]$ y sean $x_1, x_2 \in [a,b]$ con $x_1 < x_2$, a continuación se evalúa f en x_1 y x_2 se comparan los valores obtenidos y utilizando la definición de unimodalidad podemos descartar el subintervalo $[a,x_1)$ o $(x_2,b]$, en particular si $f(x_1) < f(x_2)$ el mínimo no se encuentra en el intervalo $(x_2,b]$ y si $f(x_1) > f(x_2)$ el mínimo no se encuentra en el intervalo $[a,x_1]$. Véase la *figura 2.3* .

Si $f(x_1) > f(x_2)$ el mínimo no se encuentra en el intervalo $[a,x_1)$, entonces el mínimo lo encontramos en el intervalo $[x_1,b]$, como en la *figura 2.3 (a)*. En otro caso si $f(x_1) < f(x_2)$ el mínimo no se encuentra en el intervalo $(x_2.b]$ entonces el mínimo se encuentra en el intervalo $[a,x_2]$, como en la *figura 2.3 (b)*.



(a) mínimo en (x_1,b) (b) mínimo en (a,x_2)

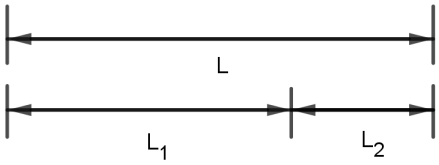
Figura 2.3: `Sección Áurea`

Se van obteniendo subintervalos más pequeños a partir de encontrar los nuevos puntos x_1 o x_2, se hace la evaluación en la función f, se comparan los valores, se descartan los intervalos y se repite el proceso hasta converger al mínimo.

La forma de encontrar los puntos x_1 y x_2 se basa en la regla de la razón áurea. Se toma un intervalo de longitud L que a su vez se divide en dos intervalos de longitudes distintas L_1 y L_2 sin pérdida de generalidad, supongamos que $L_1 > L_2$ y que deben cumplir las siguiente proporción: la razón de la longitud del sub-intervalo más pequeño (L_2) al más grande (L_1) es igual a la razón entre la longitud del subintervalo más grande (L_1) y el intervalo a dividir (L), es decir,

$$\frac{L_2}{L_1} = \frac{L_1}{L},$$

Ahora como $L = L_1 + L_2$,



(a) Intervalo de longitud L

Figura 2.4: `Sección Aurea`

$$\begin{aligned}
\Rightarrow \frac{L_2}{L_1} &= \frac{L_1}{L_1+L_2}, \\
\frac{L_2(L_1+L_2)}{L_1^2} &= 1, \\
\frac{(L_2)(L_1)+L_2^2}{L_1^2} &= 1, \\
\frac{(L_2)(L_1)}{L_1^2} + \frac{L_2^2}{L_1 2} &= 1, \\
\frac{L_2}{L_1} + \frac{L_2^2}{L_1^2} &= 1,
\end{aligned}$$

Si $\gamma = \dfrac{L_2}{L_1}$,

$$\Rightarrow \gamma^2 + \gamma = 1.$$

Resolviendo el sistema de ecuaciones obtenemos que $\gamma = \dfrac{\sqrt{5}-1}{2}$, tendremos dos puntos dentro del intervalo γ y $(1-\gamma)$. Entonces los puntos x_1, x_2 serán:

$$x_1 = a + (1-\gamma)(b-a),$$

y

$$x_2 = a + \gamma(b-a).$$

Por lo que se muestra el algoritmo de Sección Áurea.

Algoritmo 1: Sección áurea

Input: Función para hallar el mínimo, valores de intervalo $[a,b]$.

$\gamma = \dfrac{\sqrt{(5)}-1}{2}$
while $(b-a) > tol$ **do**
  if $F_1 > F_2$ **then**
    $a = x_1$
    $x_1 = x_2$
    $F_1 = F_2$
    $x_2 = a + \gamma(b-a)$
    $F_2 = F(x_2)$
  else
    $b = x_2$
    $x_2 = x_1$
    $F_2 = F_1$
    $x_1 = a + (1-\gamma)(b-a)$
    $F_1 = F(x_1)$
return x_i

Output: Valor más próximo al mínimo

EJEMPLO 1. *Sea*

$$f(x) = e^x + x^2 - 3,$$

en el intervalo $[-2,1]$ *obtendremos las primeras dos iteraciones usando el algoritmo de la sección áurea para poder tener una aproximación del mínimo de la función.*

Tenemos que $a = -2, b = 1$ *y un aproximado de* $\gamma \approx 0.6180$

$$\Rightarrow x_1 = -2 + (1-\gamma)(1-(-2)) = -.85410 \text{ y}$$

$$x_2 = -2 + \gamma(1-(-2)) = -.14589,$$

$$f(x_1) = 4.1550247 \text{ y } f(x_2) = 3.8854737 \Rightarrow f(x_1) > f(x_2).$$

Entonces el mínimo no se encuentra en el intervalo $[a,x_1)$ *y ocurre el caso de la figura 2.3 (b), donde el mínimo se encuentra en el intervalo* $[x_1,b]$*, es decir, en* $[-.85410, 1]$. *Continuando con el algoritmo* $a = x_1$ *y* $x_1 = x_2 = -.14589$ *y encontramos el nuevo valor para* x_2

$$\Rightarrow x_2 = -.85410 + \gamma(1-(-.85410)) = .29179,$$

$$f(x_1) = -2.11446 \text{ y } f(x_2) = -1.576025 \Rightarrow f(x_1) < f(x_2).$$

Por lo tanto, el mínimo no se encuentra en el intervalo $(x_2,b]$*, se encuentra en el intervalo* $[a,x_2]$ *como en la figura 4 y continuando con el algoritmo* $b = x_2 = .2917$ *y obtendremos el nuevo valor de* x_1

$$x_1 = -.85412 + (1-\gamma)(.2917961 - (-.85412) = -.41691,$$

$$f(x_1) = -2.16 \text{ y } f(x_2) = -2.11446 \Rightarrow f(x_1) < f(x_2).$$

2.1.2. Interpolación parabólica sucesiva

El método de Interpolación parabólica sucesiva se utilizará para encontrar el mínimo de una función $f : \mathbb{R} \longmapsto \mathbb{R}$ que, parte de tres puntos iniciales, donde a partir de la evaluación de dichos puntos en la función se aproxima a una función cuadrática y donde la sucesión de mínimos de la función cuadrática se acerca al mínimo de la función objetivo, si se tienen las condiciones necesarias.

Suponiendo que la función f es unimodal, el algoritmo comienza con 3 puntos en el dominio, sean $a, b, c \in \mathbb{R}$ con sus respectivas evaluaciones cumplen que $f(a) > f(b) < f(c)$. La función cuadrática por dichas evaluaciones se encuentra a través de la interpolación de Lagrange.

$$q(x) = f(a)\frac{(x-b)(x-c)}{(a-b)(a-c)} + f(b)\frac{(x-c)(x-a)}{(b-c)(b-a)} + f(c)\frac{(x-a)(x-b)}{c-a)(c-b)}.$$

Para encontrar el mínimo de $q(x)$ se debe satisfacer que, $q'(x) = 0$ donde x es el valor mínimo de la función cuadrática. Entonces se encontrara el mínimo de la función.

$$\Rightarrow q'(x) = f(a)\frac{(2xb-c)}{(a-b)(a-c)} + f(b)\frac{2x-a-c}{(b-c)(b-a)} + f(c)\frac{(2x-a-c)}{(c-a)(c-b)} = 0.$$

Si multiplicamos por $(a-b)(b-c)(c-a)$:

$$\Rightarrow f(a)(b-c)(2x-b-c) + f(b)(2x-c-a)(c-a) + f(c)(2x-b-a)(a-b) = 0,$$

$$2x(f(a)b - f(a)c + f(b)c - f(b)a + f(c)a - f(c)b) - f(a)b^2 + f(a)c^2 - f(b)c^2 + f(b)a^2 - f(c)a^2 + f(c)b^2) = 0.$$

$$x = \frac{1}{2}\frac{f(a)(b^2-c^2) + f(b)(c^2-a^2) + f(c)(a^2-b^2)}{f(a)(b-c) + f(b)(c-a) + f(c)(a-b)}.$$

El mínimo de $q(x)$ es x, que es el mínimo más cercano a la función f. Para la nueva iteración se remplazará el valor de a por c, c por b y b por el valor que se obtuvo de x. Se repite el proceso hasta converger al mínimo.

El algoritmo del método de la Interpolación Parabólica Sucesiva se muestra a continuación.

Algoritmo 2: Interpolación parabólica sucesiva

Input: f función unimodal, a,b,c tal que $f(a) > f(b)Xf(c)$

for $i = 4 \rightarrow 10$ **do**

$$x(i) = \frac{1}{2}\frac{f(a)(b^2-c^2)+f(b)(c^2-a^2)+f(c)(a^2-b^2)}{f(a)(b-c)+f(b)(c-a)+f(c)(a-b)}$$

$a = c$

$c = b$

$b = x_i$

$f_a = f_c$

$f_c = f_b$

$f_b = f(x_i)$

return x_i

Output: Valores $x(i)$ más cercanos al mínimo.

EJEMPLO 2. *Sea*

$$f(x) = e^x + x^2 - 3,$$

y sean los valores $a = -2.5$. $b = -1.2$ *y* $c = .7$, *con sus respectivas evaluaciones y que se cumple que* $f(a) = 9.33 > f(b) = 4.74 < f(c) = 5.5$, *después obtendremos el mínimo de la función cuadrática que se forma a partir de las evaluaciones, se desarrolla de la siguiente forma:*

$$x = \frac{1}{2}\frac{F(a)(b^2-c^2)+F(b)(c^2-a^2)+F(c)(a^2-b^2)}{F(a)(b-c)+F(b)(c-a)+f(c)(a-b)},$$

$$\Rightarrow x = \frac{1}{2}\frac{9.33(-1.2^2-.7^2)+4.74(.7^2-(-2.5^2))+5.5(-2.5^2-(-1.2^2))}{9.33(-1.2-.7)+4.74(.7-(-2.5))+6.72(-2.5-(-1.2))},$$

$$x = -.41328.$$

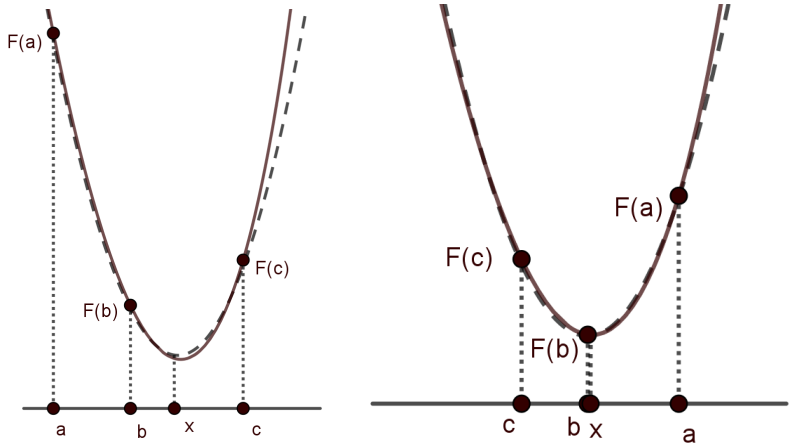
x es el mínimo de la función cuadrática formada por $f(a), f(b)$ *y donde* $f(b)$ *es el mínimo más cercano a la función* f. *La primera iteración para los puntos* a,b,c *y x se pueden observar en la* figura 2.5 (b).

Para continuar los nuevos puntos serán $a = c = 0.7$, $c = b = -1.2$ *y el valor b se remplazará por el mínimo encontrado, es decir,* $b = -0.45499$, *con sus respectivas evaluaciones* $f(a) = 6.72, f(b) = 3.84$ *y* $f(c) = 4.74$, *a partir de estas evaluaciones obtendremos el mínimo de la función cuadrática.*

$$\Rightarrow x = \frac{1}{2}\frac{6.72(-.45499^2-(-1.2^2))+3.84(-.7.2^2-0.7^2)+4.74(1^2-(-0.45499^2))}{6.72(-0.45499-(-1.2))+3.84(-1.2-0.7)+4.74(0.7-(-0.45499))},$$

$$x = 0.39351.$$

Entonces x es el mínimo más cercano a la función f como se ve en la figura 2.5 (b) *continuando con las iteraciones se encontrará el mínimo de la función f.*



(a) Iteración 1 (b) Iteración 2

Figura 2.5: `Iteraciones del método de Interpolación Parabólica Sucesiva.`

2.1.3. Método de Newton

El método de Newton para hallar mínimos de una función $f : \mathbb{R} \to \mathbb{R}$, asume que existen la primera y segunda derivada en un punto x, por lo que a partir de dicha información se aproxima a una función cuadrática y el mínimo de dicha función será el mínimo más próximo a la función f.

El algoritmo del método se basa en obtener una aproximación cuadrática mediante una Serie de Taylor.

$$f(x+h) \approx f(x) + f'(x)h + \frac{1}{2}f''(x)h^2.$$

derivando obtenemos el mínimo de la función cuadrática h,

$$\frac{\partial f(x+h)}{\partial x} = f'(x) + f''(x)h = 0,$$

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

El método comienza con un punto inicial x_0 y los siguientes puntos tomarán la forma

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}.$$

Por lo que finalmente se obtiene el algoritmo del Método de Newton de la siguiente forma:

Algoritmo 3: Newton

Input: La función para hallar su mínimo, punto inicial x_0.

for $i = 0 \to n$ **do**

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$$

return x_i

Output: Valores x_i más cercanos al mínimo.

EJEMPLO 3. *Sea*

$$f(x) = e^x + x^2 - 3.$$

Obtenemos la primera derivada de f, $f'(x) = e^x + 2x$ *y la segunda derivada* $f''(x) = e^x + 2$ *y comenzaremos en el punto* $x_0 = -1$.

Para la primera iteración se tiene que:

$$x_1 = x_0 - \frac{-f'(x_0)}{f''(x_0)} = -(-1) - \frac{-f'(-1)}{f''(-1)} = -0.31072,$$

para la siguiente itaración obtenemos que:

$$x_2 = x_1 - \frac{-f'(x_1)}{f''(x_1)} = -(-.31072) - \frac{-f'(-.31072)}{f''(-.31072)} = -0.35151.$$

2.1.4. Método de la secante

El método de Newton para minimizar una función $f : \mathbb{R} \to \mathbb{R}$, se debe calcular la segunda derivada de la función en distintos puntos.

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}.$$

Dada la dificultad que a veces conlleva calcular la segunda derivada, el método de la secante propone aproximar numéricamente la segunda derivada.

$$f''(x_i) = \frac{f'(x_i) - f'(x_{i-1})}{x_i - x_{i-1}}.$$

Utilizando dicha aproximación la sustituimos en el método de Newton:

$$x_{i+1} = x_i - \frac{f'(x_i) - f'(x_{i-1})}{x_i - x_{i-1}} f'(x_i).$$

Por lo que el método de la Secante necesita dos puntos iniciales y cada actualización se puede escribir de la siguiente forma:

$$x_{i+1} = \frac{x_{i-1}f'(x_i) - x_i f'(x_{i-1})}{f'(x_i) - f'(x_{i-1})}.$$

Por lo que se tiene el algoritmo de la secante.

Algoritmo 4: Método de la Secante

Input: La función f, x_0, x_1 puntos iniciales.

1 **for** $i = 0 \to n$ **do**

2 $\quad x_{i+1} = \dfrac{x_{i-1}f'(x_i) - x_i f'(x_{i-1})}{f'(x_i) - f'(x_{i-1})}$

3 **return** x_i

Output: Valores $x(i)$ más cercanos al mínimo.

EJEMPLO 4. *Sea la función:*

$$f(x) = e^x + x^2 - 3.$$

y sean los puntos iniciales $x_0 = -1$ *y* $x_1 = -.5$*,*

En el ejemplo del método de Newton obtuvimos que la primera derivada de la función es

$$f'(x) = \exp(x) + 2x.$$

Calculando la primera iteración se tiene que:

$$x_2 = \frac{x_0 f'(1) - x_1 f'(x_0)}{f'(x_1) - f'(x_0)} = \frac{-.4225}{1.2386} = -.3411.$$

Para la segunda iteración se tiene que:

$$x_3 = \frac{x_1 f'(x_2) - x_2 f'(x_1)}{f'(x_2) - f'(x_1)} = \frac{-.1485}{.422} = -.3519.$$

Capítulo 3

Optimización en varias variables

Una vez descrito los métodos para encontrar mínimos de funciones reales, se procederán a describir métodos que encuentran o aproximan los mínimos de funciones multievaluadas.

3.1. Método del Gradiente

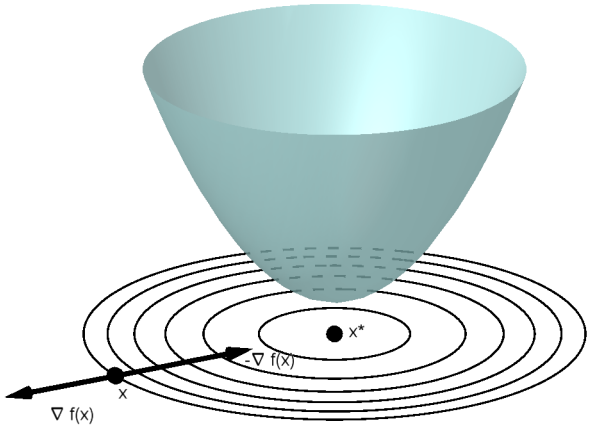
Uno de los primeros métodos para aproximar al mínimo de una función $f : \mathbb{R}^n \longmapsto \mathbb{R}$, es el método del gradiente que se basa principalmente en las evaluaciones de la función objetivo y en la información que se pueda obtener del gradiente.

Dada una función $f : \mathbb{R}^n \longmapsto \mathbb{R}$, se denomina como conjunto o curva de nivel a los puntos $x \in \mathbb{R}^n$ que satisfacen $f(x) = c$, donde c es una constante, de forma que si tenemos una función $f : \mathbb{R}^2 \longmapsto \mathbb{R}$ podemos observar sus curvas de nivel como en la *figura 3.1*.

Sea el gradiente de una función f en un punto x_0, lo detonamos como $\bigtriangledown f(x_0)$, si el vector es distinto de cero, entonces es un vector ortogonal tangente a una curva suave arbitraria que pasa por x_0 en el conjunto de nivel $f(x) = c$, por lo que la dirección de máximo ascenso de una función diferenciable en dicho punto es ortogonal al conjunto de nivel.

Por lo que la dirección del $\bigtriangledown f(x)$ es la del máximo ascenso en el punto x. Con lo que $-\bigtriangledown f(x)$ es la dirección del máximo descenso de la función f en el punto x, a partir de dicha dirección se hará la búsqueda del mínimo de la función.

Teniendo el vector $-\bigtriangledown f(x_0)$ se obtiene una dirección para encontrar un mínimo, como se puede visualizar en la *figura 3.1(a).*



(a) curvas de nivel

Figura 3.1: $f(x,y) : \mathbb{R}^2 \longrightarrow \mathbb{R}$

A partir del punto inicial x_0 se buscará un punto crítico de una función $f : \mathbb{R}^n \to \mathbb{R}$ además, utilizando un nuevo punto de la forma $x_0 - \alpha \bigtriangledown f(x_0)$ y el Teorema de Taylor se obtiene:

$$f(x_0 - \alpha \bigtriangledown f(x_0)) = f(x_0) - \alpha \| \bigtriangledown f(x_0) \|^2 + o(\alpha),$$

donde $\| \bigtriangledown f(x_0) \| \neq 0$ y teniendo un escalar α suficientemente pequeña y mayor a cero, tenemos que:

$$f(x_0 - \alpha \bigtriangledown f(x_0)) < f(x_0),$$

Por lo que se puede considerar que el nuevo punto $x_0 - \alpha \bigtriangledown f(x_0)$ es una buena forma de acercarse al mínimo de la función. Por lo que podemos encontrar los nuevos puntos x_{i+1} de forma iterativa partiendo de un punto x_0 incial de la siguiente manera:

$$x_{i+1} = x_i - \alpha_i \bigtriangledown f(x_i),$$

se tiene que cada α_i es una constante mayor a cero, por lo que el escalar debe cumplir que sea la distancia más pequeña que se realice sobre la línea de búsqueda, es decir, en cada iteración se buscara minimizar la función real $f(x_i - \alpha g_i)$ respecto a α, donde se

tiene que $g_i = \bigtriangledown f(x_i)$.

Para encontrar el valor de α utilizaremos el Hessiano de la función $H_{f(x_i)}$ y una aproximación mediante la serie de Taylor.

$$f(x_i - \alpha g_i) \approx f(x_i) - \alpha g_i^t g_i + \frac{1}{2}\alpha^2 g_i^t H_i g_i.$$

Obteniendo la derivada respecto a α e igualando a cero,

$$\begin{aligned} \frac{d}{d\alpha} f(x_i - \alpha g_i) &= 0, \\ \Rightarrow \frac{d(f(x_i) - \alpha g_i^t g_i + \frac{1}{2}\alpha^2 g_i^t H_i g_i)}{d\alpha} &= 0, \end{aligned}$$

derivando obtenemos que:

$$-g_i^t g_t + \alpha g_i^t H_i g_i = 0,$$

despejando α,

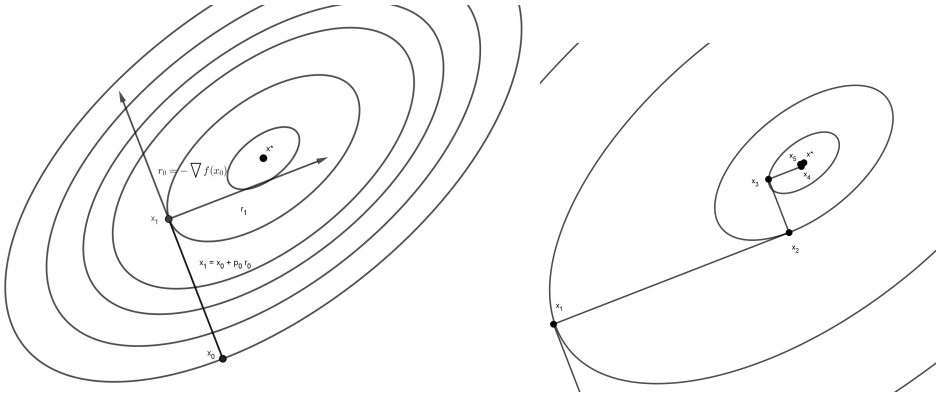
$$\alpha = \frac{g_i^t g_t}{g_i^t H_i g_i}.$$

Por lo que, α minimiza la función real $f(x_i - \alpha g_i)$ y podemos reescribir la forma de encontrar los nuevos puntos x_i de la siguiente manera:

$$x_{i+1} = x_i - \frac{g_i^t g_i}{g_i^t H_i g_i} g_i.$$

En la *figura 3.2 (a)* se pueden observar las primeras dos iteraciones del método del gradiente para una función $f(x,y) : \mathbb{R}^2 \to \mathbb{R}$, comenzando en un punto inicial x_0 y encontrando un punto x_1 en la dirección de $d_0 = - \bigtriangledown f(x_0)$, por otra parte en la figura $3.2b$) se pueden observar las siguientes iteraciones para acercarse al punto x^*.

Por lo tanto tenemos como algoritmo del método del Gradiente.



(a) Primeras dos iteraciones (b) Iteraciones para encontrar x^*

Figura 3.2: `Itereaciones método del Gradiente`

Algoritmo 5: Método del Gradiente

Input: Matriz H, punto incial x_0

$g = \bigtriangledown f(x)$
$tol = norma(g)$
$epsilon = .ooooo1$
while $tol > epsilon$ **do**
    $\alpha = \dfrac{g^t g}{g^t H g}$
    $x = x - \alpha g$
    $g = \bigtriangledown f(x)$
    $tol = norma(g)$
    $i = i + 1$
return x_i

Output: Valor más próximo al punto crítico

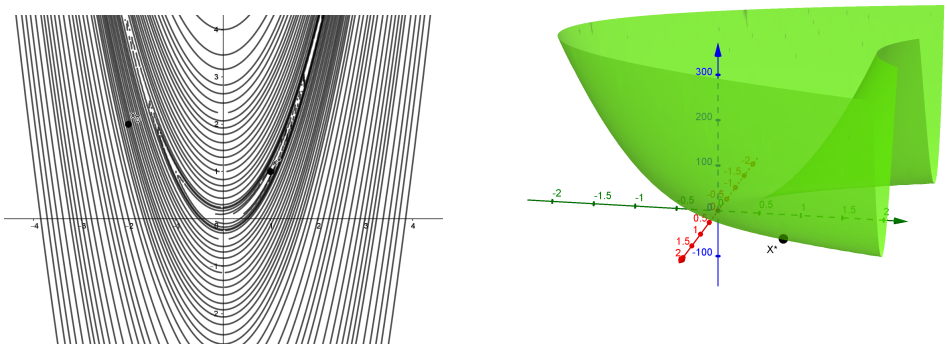
Se tomara como ejemplo la función de Rosenbrock,

$$f(x,y) = (1-x)^2 + 100(y-x^2)^2.$$

A partir de la función se tratará de llegar al mínimo por los diferentes métodos. Se observará en cada uno de los métodos las gráficas y sus curvas de nivel, como los distintos métodos van convergiendo al punto mínimo. Como se puede ver en la *figura 3.3* la función tiene un punto mínimo en $(1,1)$.

El gradiente y Hessiano de la función de Rosenbrock están dados por:

$$\bigtriangledown f(x,y) = (2(200x^3 - 200xy + x - 1), 200(y-x^2).$$



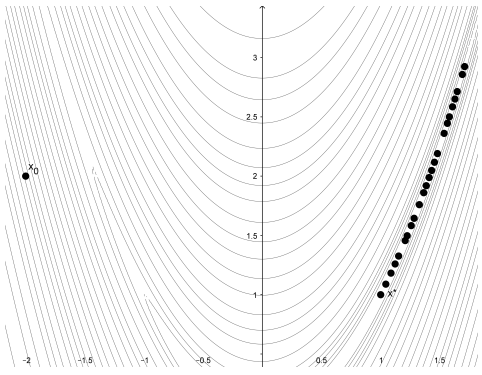
(a) Curvas de nivel de la función de Rosenbrock

(b) Gráfica de la función en R^3

Figura 3.3: `Función de Rosenbrock`

$$Hf(x,y) = \begin{bmatrix} 2(600x^2 - 200y + 1) & -400x \\ -400x & 200 \end{bmatrix}.$$

Dado el punto inicial $x_0 = (-2,2)$ observaremos en la *figura 3.4* algunos puntos para converger al mínimo en $(1,1)$ mediante el método del Gradiente.



(a) Convergencia al punto mínimo en $(-2,2)$

Figura 3.4: `Método del Gradiente`

3.2. Gradiente Conjugado

Con el método del Gradiente se obtuvo que la dirección de descenso se conforma de la siguiente manera:

$$\alpha_0 = \frac{g_0^t g_0}{g_0^t H g_0}.$$

Iniciando con α_0 como dirección de descenso y a partir de la segunda iteración del método del Gradiente se buscará una dirección de descenso distinta, dicha dirección será aquella que une el punto x_1 con el punto crítico x^*, por lo que la dirección se buscará sin conocer la ubicación del punto crítico y la denotaremos como g^*, se puede ver graficamente en la *figura 3.5.*

Utilizando la forma iterativa del método del Gradiente para encontrar el punto $x_1 = x_0 - \alpha_0 g_0$ y $g_0 = - \bigtriangledown f(x_0)$ como la primer dirección de descenso, se buscará una mejor dirección g_1^*.

La dirección de g_1^* será aquella que une x_1 con el punto crítico x^*.

$$g_1^* = x^* - x_1 = x^* - x_0 - \alpha_0 g_0.$$

Primero se demostrará que los vectores g_0 y g_1^* son conjugados.

PROPOSICIÓN 1. *Los vectores g_0 y g_1^* son conjugados respecto a la matriz H.*

Demostración. Se demostrará que $(g_0)^t H(g_1^*) = 0$.

$$\begin{aligned} g_0^t H(g_1^*) &= g_0^t H(x^* - x_1), \\ &= g_0^t H(x^* - x_0 - \alpha_0 g_0), \\ &= g_0^t (Hx^* - Hx_0 - H\alpha_0 g_0), \\ &= g_0^t (Hx^* - Hx_0 - \alpha_0 H g_0). \end{aligned}$$

Si tenemos que $Hx^* = b$ entonces se obtendría que $b - Hx_o^* = g_0$.

$$\begin{aligned} g_0^t (Hx^* - Hx_0 - \alpha H_0 g_0) &= g_0^t (g_o - \alpha_0 H g_0), \\ &= g_0^t g_o \alpha_0 g_0^t H g_0, \\ &= g_0^t g_o (\frac{g_0^t g_0}{g_0^t H g_0}) g_0^t H g_0, \\ &= g_0^t g_0 - g_0^t g_0 = 0. \end{aligned}$$

□

Se tiene que $g_1 = - \bigtriangledown f(x_1)$ como la dirección de descenso en el punto x_1, se buscará mejorar la dirección de descenso como combinación lineal de las direcciones g_1, g_0 y utilizando dos escalares β_1 y β_2,

$$d_1^* = \beta_1 g_1 + \beta_2 g_0.$$

No es necesario encontrar explícitamente el vector d_1^*, solo basta con encontrar la dirección ya que las direcciones son equivalentes.

$$d_1 = \frac{1}{\beta_1} d_1^* = \frac{1}{\beta_1}\beta_1 g_1 + \frac{1}{\beta_1}\beta_2 g_0 = g_1 + \frac{\beta_2}{\beta_1} g_0 = g_1 + \gamma_0 g_0.$$

Para obtener el parámetro γ_0 utilizamos que los vectores g_0 y d_1 son conjugados respecto a la matriz H, entonces:

$$\begin{aligned} g_0^t H d_1 &= 0, \\ \Rightarrow g_0^t H(g_1 + \gamma_0 g_0) &= g_0^t H g_1 + g_0^t H \gamma_0 g_0 = 0, \end{aligned}$$

despejando γ_0 :

$$\gamma_0 = -\frac{g_o^t H g_1}{g_0^t h g_0}.$$

Por lo tanto, se tiene la dirección que pasa por x_1 y x^* a partir de la primera iteración del método del Gradiente por lo que se tiene el método del Gradiente Conjugado por lo que se tiene para las dos primeras iteraciones lo siguiente:

$$\begin{aligned} g_0 = r_0 &= b - Ax_0, \\ a_0 &= \frac{g_o^t g_0}{g_o^t H g_0}, \\ x_1 &= x_0 + a_0 g_0, \\ r_1 &= b - Ax_1, \\ \gamma_0 &= -\frac{g_0^t H r_1}{g_0^t H g_0}, \\ g_1 &= r_1 + \alpha_0 r_0, \\ \alpha_1 &= \frac{r_1 g_1}{g_1^t H g_1}, \\ x_2 &= x_1 + \alpha_1 g_1. \end{aligned}$$

Finalmente se tiene de forma iterativa el siguiente algoritmo del Método del Gradiente Conjugado.

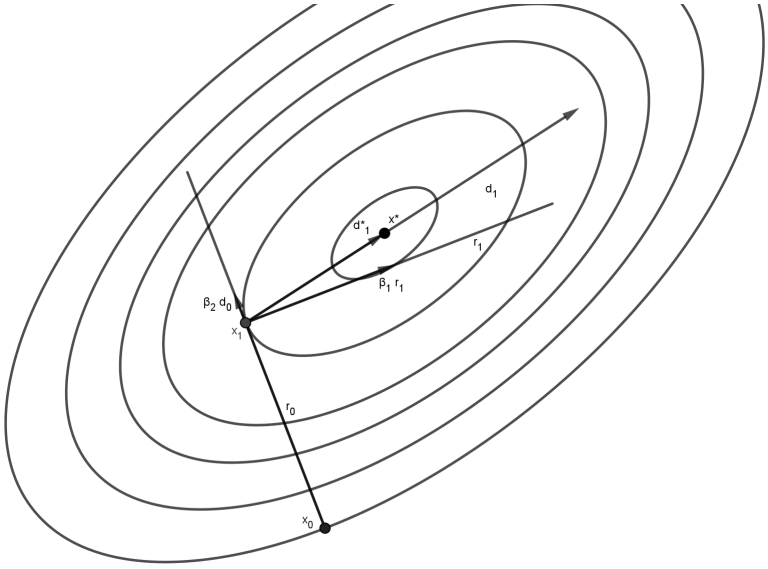
Algoritmo 6: Método del Gradiente Conjugado

Input: Matriz H, vector (b), punto incial x_0

$i = 0$
$r = b - Ax$
$d = r^t r \; tol = norma(r)$
$epsilon = .ooooo1$
while $tol > epsilon$ **do**
    $\alpha_i = \dfrac{(r_i^t)d_i}{d_i^t H d_i}$
    $x_{i+1} = x_i + a_l pha_i d_i$
    $r_{i+1} = b - Hx_{i+1}$
    $\gamma_i = -\dfrac{r_{i+1}^t H g_i}{g_i^t H g_i}$
    $d_{i+1} = r_{i+1} + \gamma_i d_i$
    $tol = norma(r_i)$
    $i_i + 1$
return x_i

Output: Valor más próximo al punto crítico

En la *figura 3.5* podemos observas las primeras iteraciones del método del Gradiente Conjugad.



(a) Curvas de nivel de una función cuadrática, primeras dos iteraciones

Figura 3.5: `Gradiente conjugado`

3.3. Método de Newton

Los métodos del Gradiente y Gradiente Conjugado son favorables cuando se parte de un adecuado punto inicial y una buena búsqueda de dirección, por lo que para mejorar dichos métodos se utilizan derivadas de orden superior que permiten la convergencia al punto crítico en un tiempo más rápido, como es el caso del método de Newton que usa las primeras y segundas derivadas de una función $f : \mathbb{R}^n \to \mathbb{R}$, es decir, utiliza la información del Gradiente y del Hessiano de la función para hallar un punto mínimo.

El método de Newton se basa en hacer una aproximación de la función objetivo con una función cuadrática utilizando el gradiente y Hessiano de la función mediante una serie de Taylor de la siguiente manera:

$$f(x) \approx \varphi(x) = f(x_0) + (x - x_0)^t \bigtriangledown f(x_0) + \frac{1}{2}(x - x_0)^t H_{f(x_0)}(x - x_0).$$

Al hacerse la aproximación por la serie de Taylor, el mínimo relativo de $f(x)$ se aproxima por el mínimo relativo de $\varphi(x)$. Si un nuevo punto x_1 es un mínimo relativo de $\varphi(x)$, entonces es un punto estacionario por lo que, si se satisfacen las condiciones necesarias de primer orden se tiene que $\bigtriangledown\varphi(x_1) = 0$.

$$\begin{aligned} \bigtriangledown\varphi(x) &= \bigtriangledown f(x_0) + \bigtriangledown^2 f(x_0)(x_1 - x_0) = 0, \\ &\Rightarrow x_1 = x_0 - \bigtriangledown f(x_0) H^{-1}_{f(x_0)}. \end{aligned}$$

Por lo que tenemos la siguiente forma para generar los nuevos puntos.

$$x_{i+1} = x_i - \bigtriangledown f(x_i) H^{-1}_{f(x_i)}.$$

Donde $-\bigtriangledown f(x_i) H^{-1}_{f(x_i)}$ indica la dirección de descenso más rápido. Por lo tanto, tenemos el algoritmo del Método de Newton.

Algoritmo 7: Método de Newton

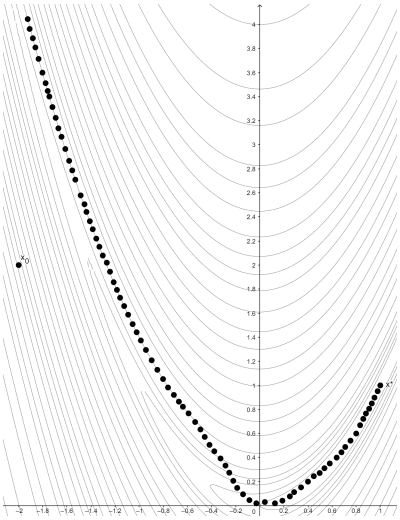
Input: función f, punto incial x, tol

```
r_0 =|| ▽f(x) ||
while || ▽f(x) ||> tol do
    H = ▽²f(x)
    s = x + ▽f(x)H^{-1}
    ▽f(x)
return x_i
```

Output: Valor más aproximado al punto crítico

Continuando con la función de Rosenbrock $f(x,y) = (1-x)^2 + 100(y-x^2)^2$ y partiendo del punto inicial $(-2,2)$, se observa que el costo del método es resolver el sistema para encontrar la matriz Hessiana, pero se realizan menos iteraciones que en los métodos anteriores. En la *figura 3.6* se observan los primeros pasos del método de

Newton y como se va acercando al punto mínimo.



(a) Convergencia al punto mínimo iniciando en $(-2,2)$

Figura 3.6: `Método de Newton`

3.4. Métodos Quasi-Newton

El método de Newton es de los métodos que convergen más rápido al punto crítico aunque existen algunas desventajas como calcular la inversa de la matriz Hessiana, así como también para encontrar un mínimo global, el punto inicial x_0 debe ser cercano a éste, además de necesitar que la matriz Hessiana sea definida positiva. Se han desarrollado distintos métodos basados en el método de Newton que permiten mejorar los tiempos de convergencia, el grupo de métodos son llamados métodos Quasi-Newton.

A partir de la siguiente actualización:

$$x_{i+1} = x_i - \bigtriangledown f(x_i) H^{-1}_{f(x_i)}.$$

El objetivo será evitar realizar el calculo de la inversa de la matriz Hessiana. Una forma de hacerlo es aproximar $\bigtriangledown^2 f(x_i)$ mediante una matriz H^* utilizando solamente las primeras derivadas parciales de la función. Entonces un nuevo punto x_{i+1} se cálcula utilizando una aproximación de la matriz Hessiana H^* y basándose en el método de Newton tenemos:

$$x_{i+1} = x_i \bigtriangledown f(x_i) H^*_i.$$

3.4.1. Método de Broyden

El primer método que se desarrollará es el método de Broyden que tiene como premisa en su segunda iteración sustituir a la matriz Hessiana por una matriz A_i que se obtiene de una matriz A_{i-1}, es decir, la matriz A_0 será la matriz Hessiana evaluada en el punto x_0, $A_0 = H_{x_0}$. Entonces tendremos de forma iterativa el siguiente sistema:

$$A_i(x_i - x_{i-1}) = f(x_i) - f(x_{i-1}).$$

Supongamos que x, x_i, x_{i-1} son puntos suficientemente cercanos a un punto crítico y $f(x)$ se puede aproximar mediante una serie de Taylor de la siguiente manera:

$$\begin{aligned} f(x) &\approx f(x_i) + H_{x_i}(x - x_i), \\ f(x) &\approx f(x_{i-1}) + H_{x_{i-1}}(x - x_{i-1})), \end{aligned}$$

Si restamos la primera ecuación menos la segunda:

$$f(x_i) + H_{x_i}(x - x_i) - f(x_{i-1}) - H_{x_{i-1}}(x - x_{i-1})) \approx 0,$$

El objetivo es aproximar H_{x_i} mediante una matriz A_{x_i}, entonces se debe minimizar dado cualquier vector $x \in \mathbb{R}^n$ con un vector de la forma:

$$\begin{aligned} f(x_i) + A_{x_i}(x - x_i) - f(x_{i-1}) - A_{x_{i-1}}(x - x_{i-1})) &= 0, \\ f(x_i) + A_{x_i}(x) - A_{x_i}(x_i) - f(x_{i-1}) - A_{x_{i-1}}(x) - A_{x_{i-1}}(x_{i-1}) + A_{x_i}(x_{i-1}) - A_{x_i}(x_{i-1}) &= 0, \\ f(x_i) - f(x_{i-1}) - A_{x_i}(x_i - x_{i-1}) + (A_{x_i} - A_{x_{i-1}})(x - x_{i-1}) &= 0. \end{aligned}$$

Teníamos que $A_{x_i}(x_i - x_{i-1}) = f(x_i) - f(x_{i-1})$. Entonces buscamos minimizar:

$$(A_{x_i} - A_{x_{i-1}})(x - x_{i-1}) = 0.$$

Por otra parte, tenemos que el vector $(x - x_{i-1})$ se puede ver como un vector proporcional a $x_i - x_{i-1}$ y como un vector ortogonal, es decir,

$$(x - x_{i-1}) = \alpha(x_i - x_{i-1}) + v,$$

donde $\alpha > 0$, se definirá como $u_i = (x_i - x_{i-1})$ y v es un vector ortogonal a u_i entonces $(v^t)(u_i) = 0$, tenemos que:

$$\begin{aligned}(A_{x_i} - A_{x_{i-1}})(x - x_{i-1}) &= (A_{x_i} - A_{x_{i-1}})(\alpha(u_i) + v),\\ &= \alpha(A_{x_i} - A_{x_{i-1}})(u_i) + (A_{x_i} - A_{x_{k-1}})v.\\ &= 0.\end{aligned}$$

Tenemos que $\alpha(A_{x_i} - A_{x_{i-1}})(u_i)$ es un valor constante, por lo que debe minimizarse $(A_{x_i} - A_{x_{i-1}})v = 0$, lo que ocurre sólo si todos los vectores fila de $(A_{x_i} - A_{x_{i-1}})$ son ortogonales a v, es decir, como $uv = 0$ todos los vectores fila de $(A_{x_i} - Ax_{i-1})$ deben ser proporcionales u_i, por lo tanto $A_{x_i} - A_{x_{i-1}} = b(u_i^t)$ donde b es un vector columna formado por los factores de proporcionalidad de cada fila, por lo que ahora encontraremos b.

Partimos de $A_{x_i}(u_k) = f(x_i) - f(x_{i-1})$ y restamos en cada lado de la ecuación $A_{x_{i-1}}(u_i)$:

$$\begin{aligned}A_{x_i}(u_i) - A_{i-1}(u_i) &= f(x_i) - f(x_{i-1}) - A_{i-1}(u_i),\\ (A_{x_i} - A_{x_{i-1}})(u_i) &= f(x_i) - f(x_{i-1}) - A_{i-1}(u_i),\\ b(u_i^t)(u_i) &= f(x_i) - f(x_{i-1}) - A_{i-1}(u_i),\\ \Rightarrow b &= \frac{f(x_i) - f(x_{i-1}) - A_{i-1}(u_i)}{(u_i^t)(u_i)}.\end{aligned}$$

Por lo que a partir de $A_{x_i} - A_{x_{i-1}} = b(u_i^t)$ y dado el valor de b tenemos que la matriz A_{x_i} se puede ver de la forma:

$$vA_{x_i} = A_{x_{i-1}} + \frac{f(x_i) - f(x_{i-1}) - A_{i-1}(u_i)}{(u_i^t)(u_i)}u_i^t.$$

El objetivo es encontrar A_i^{-1} y utilizando la fórmula de Sherman-Morrison.

Teorema 3. *Fórmula de Sherman-Morrison Sea A una matriz no singular A_{nxn} y sean vectores $u, v \in R^n$. $A + uv^t$ es invertible si y solo si $1 + v^t A^{-1} u \neq 0$, entonces*

$$(A + vu^t)^{-1} = A^{-1} - \frac{A^{-1} v u^t A^{-1}}{1 + u^t A^{-1} v}.$$

Definamos a $y_i = f(x_i) - f(x_{i-1})$ y $w_i = \dfrac{y_i - A_{x_{i-1}} u_i}{(u_i^t)(u_i)}$, entonces:

$$A_{x_i} = A_{x_{i-1}} + w_i(u_i^t).$$

Utilizando la fórmula Sherman-Morrison tenemos que:

$$\begin{aligned}A_{x_i}^{-1} &= A_{x_{i-1}}^{-1} - \frac{(A_{x_{i-1}}^{-1})(w_i)(u_i^t)(A_{x_{i-1}}^{-1})}{1 + (u_i^t)(A_{x_{i-1}}^{-1})(w_i)} =\\ &= A_{x_{i-1}}^{-1} - \frac{((u_i - A_{x_{i-1}}^{-1} y_i))(u_i^t)(A_{x_{i-1}}^{-1})}{(u_i^t)(A_{x_{i-1}}^{-1})(y_i)}.\end{aligned}$$

Ahora tenemos el algoritmo del método de Broyden.

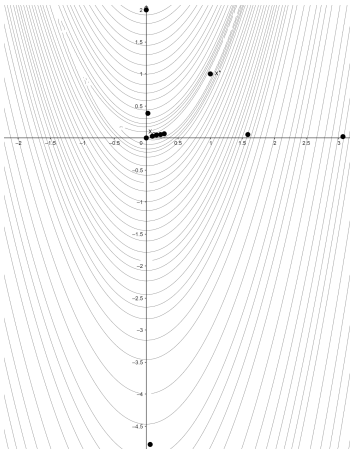
Algoritmo 8: Método de Broyden

Input: punto inicial x_0, f función a minimizar, tol

B = matriz identidad
$g = \bigtriangledown f(x_0)$
$tol = norma(g)$
$epsilon = .ooooo1$
while $tol > epsilon$ **do**
    $g = \bigtriangledown f(x_0)$
    $s = -B^{-1}g$
    $x_1 = x_0 + s$
    $y = \bigtriangledown f(x_1) - \bigtriangledown f(x_0)$
    $x = x1$
    if $\mid s's \mid > \varepsilon$ **then**
        $B = B + \frac{(y' - Bs)s'}{s's}$
return x_i

Output: Valor más próximo al mínimo

En la *figura 3.7* se observa gráficamente para la función de Rosenbrock como va convergiendo a partir del punto inicial $x_0 = (0,0)$ al punto mínimo de la función $f(x,y) = (1-x)^2 + 100(y-x^2)^2$.



(a) Función de Rosenbrock

Figura 3.7: `Convergencia del método de Broyden`

3.4.2. Método Davidon-Fletcher-Powell

El método de Broyden en la matriz Hessiana mantiene la simetria para H_{i+1} a partir de la matriz H_i, pero no se garantiza que H_{i+1} siga siendo definida positiva. Por lo que se puede probar una actualización de la siguiente forma:

$$H_i = h_{i-1} + \alpha uu^t + \beta vv^t.$$

Partiendo de la ecuación de la secante $H_i(f(x_i) - f(x_{i-1})) = (x_i - x_{i-1})$ sustituimos el valor de H_i y definamos a $y_i = f(x_i - f(x_{i-1}))$ y $s_i = (x_i - x_{i-1})$.

$$\begin{aligned}(h_{i-1} + \alpha uu^t + \beta vv^t)y_i &= s_i,\\ h_{i-1}y_i + \alpha uu^t y_i + \beta vv^t y_i &= s_i y_i,\\ \alpha(u^t y_i)u + \beta(v^t y_i)v &= s_i y_i - h_{i-1}y_i.\end{aligned}$$

Tomemos a $u = \delta_i$ y $v = H_i y_i$ como una posible solución del sistema y encontramos los valores de α y β que resuelven el sistema,

$$\begin{aligned}\alpha(s_i^t y_i)s_i + \beta((H_i y_i)^t y_i)H_i y_i &= S_i - H_i y_i,\\ \alpha(s_i^t y_i)s_i + \beta(y_i^t H_i y_i)H_i y_i &= S_i - H_i y_i,\end{aligned}$$

entonces, $\alpha = \dfrac{1}{\delta_i^t y_i}$ y $\beta - \dfrac{1}{y_i^t H_i y_i}$.

Sustituimos en la actualización de Davidon-Fletcher-Powell (DFP) $H_i = h_{i-1} + \alpha uu^t + \beta vv^t$ y se obtiene que:

$$H_i = H_{i-1} + \frac{\delta_i \delta_i^t}{\delta_i^t y_i} - \frac{H_i y_i y_i^t H_i}{y_i^t H_i y_i}.$$

Ahora demostraremos que la actualización DFP mantiene la matriz definida positiva.

Teorema 4. *Sea S_i una matriz simétrica y definida positiva, entonces la matriz S_{i+1} generada por la actualización de Davidon-Fletcher-Powell es también una matriz definida positiva.*

Demostración. Sea un vector $x \in \mathbb{R}^n$. Multiplicaremos por el vector x y su transpuesto x^t en la actualización de DFP.

$$x^t S_i x = x^t S_i x + \frac{x^t \delta \delta^t x}{\delta^t y_i} - \frac{x^t S_i y_i y_i^t S_i x}{y_i S_i y_i}.$$

Si escribimos a S_i como $US_iS^t = A$, donde U cumple que $U^tU = U^tU = I$ y A sea una matriz diagonal donde sus elementos en la diagonal son los eigenvalores de S_i, también usaremos la propiedad de que S_i es simétrica y definida positiva, entonces podemos escribir a la matriz S_i como:

$$\begin{aligned}\Rightarrow S_i &= UAU^t = UA^{1/2}A^{1/2}U^t,\\ &= UA^{1/2}U^tUA^{1/2}U^t,\\ &= S_i^{1/2}S_i^{1/2}.\end{aligned}$$

Si definimos $u = S_i^{1/2}$ y $v = S_i^{1/2}y_i$.

$$\begin{aligned}x^tS_ix &= x^tS_i^{1/2}S_i^{1/2}x + \frac{x^t\delta_i\delta_i^tx}{\delta_i^ty_k} - \frac{x^tS_i^{1/2}S_i^{1/2}y_iy_i^tS_i^{1/2}S_i^{1/2}x}{y_i^tS_i^{1/2}S_i^{1/2}y_i},\\ &= u^tu + + \frac{x^t\delta_i\delta_i^tx}{\delta_i^ty_i} - \frac{u^tvv^tu}{v^tv},\\ &= \frac{u^tuv^tv - u^tvv^tu}{v^tv} + \frac{x^t\delta_i\delta_i^tx}{\delta_i^ty_i}.\end{aligned}$$

Dado que en los problemas de minimización queremos encontrar el punto $x_{i+1} = x_i + \alpha d_i$ donde $d_i = -S_ig_i$ es una dirección de descenso, entonces se define a $\delta_i = \alpha_id_i = -\alpha_iS_ig_i$ donde α_i es el valor que minimiza a $f(x_i + \alpha d_i)$ en el punto $x = x_{i+1}$. Ahora utilizaremos algún método de minimización de funciones en una dimensión para obtener el valor mínimo de α por lo que derivando la función $f(x_i + \alpha d_i)$ respecto α obtengamos:

$$\frac{f(x_i + \alpha d_i)}{d\alpha} = g(x_i + \alpha_i d_i)^t d_i = 0.$$

Partiendo de la última igualdad, multiplicamos por α_i en cada parte de la ecuación:

$$\alpha_i g(x_i + \alpha_i d_i)^t d_i = g(x_i + \alpha_i d_i)^t \alpha_i d_i = g(x_i + \alpha_i d_i)^t \delta_i = \delta_i^t g(x_i + \alpha_i d_i) = 0.$$

Dado que y_i es la diferencia de los gradientes $y_i = \bigtriangledown f(x_{i+1}) - \bigtriangledown f(x_i)$ y usando la última igualdad $\delta_i^t g(x_i + \alpha_i d_i) = 0$.

$$\delta_i^t y_i = \delta_i^t(\bigtriangledown f(x_{i+1}) - \bigtriangledown f(x_i)) = \delta_i^t \bigtriangledown f(x_{i+1}) - \delta_i^t \bigtriangledown f(x_i) = -\delta_i^t \bigtriangledown f(x_i)$$

Si utilizamos que $\delta_i = \alpha_i d_i = -\alpha S_i g_i$,

$$\delta_i^t y_i = -\delta_i^t \bigtriangledown f(x_i) = -(-\alpha S_i g_i)^t \bigtriangledown f(x_i) = \alpha_i \bigtriangledown f(x_i)^t S_i \bigtriangledown f(x_i).$$

Usando $u^tu = \|u\|^2$, $v^tv = \|v\|^2$ y $u^tv = (\|u\|\|v\|cos(\theta))^2$ donde θ es el ángulo que existe entre las dos vectores y retomando que $x^tS_ix = \dfrac{u^tuv^tv - u^tvv^tu}{v^tv} + \dfrac{x^t\delta_i\delta_i^tx}{\delta_i^ty_i}$, obtenemos.

$$x^tS_ix = \frac{\|u\|^2\|v\|^2 - (\|u\|\|v\|cos(\theta))}{\|v\|^2} + \frac{(x^t\delta_i)^2}{\alpha_i \bigtriangledown f(x_i)^tS_i \bigtriangledown f(x_i)}.$$

El valor mínimo ocurre cuando $\theta = 0$ por lo que se tiene que:

$$x^t S_i x = \frac{(x^t \delta_i)^2}{\alpha_i \bigtriangledown f(x_i)^t S_i \bigtriangledown f(x_i)}.$$

por lo que tenemos que, u y v son vectores que van en la misma direccion, partiendo de $u = S_i^{1/2} x$,

$$\begin{aligned} u &= S_i^{1/2} x, \\ &= \beta v, \\ &= \beta S_i^{1/2} y_i, \\ &= S_i^{1/2} \beta y_i, \\ \Rightarrow x &= \beta y_i. \end{aligned}$$

Siempre que $\beta > 0$, tendríamos:

$$\begin{aligned} x^t S_i x &= \frac{(x^t \delta_i)^2}{\alpha_i \bigtriangledown f(x_i)^t S_i \bigtriangledown f(x_i)}, \\ &= \frac{((\beta y_i)^t \delta_i)^2}{\alpha_i \bigtriangledown f(x_i)^t S_i \bigtriangledown f(x_i)}, \\ &= \frac{\beta \alpha_i \bigtriangledown f(x_i)^t S_i \bigtriangledown f(x_i))^2}{\alpha_i \bigtriangledown f(x_i)^t S_i \bigtriangledown f(x_i)}, \\ &= \alpha_i \beta^2 \bigtriangledown f(x_i)^t S_i \bigtriangledown f(x_i). \end{aligned}$$

Para el caso en que $\theta \geqslant 0$,

$$x^t S_i x \geq \alpha_i \beta^2 \bigtriangledown f(x_i)^t S_i \bigtriangledown f(x_i).$$

Si tomamos x distinto al mínimo de f tendríamos el $\bigtriangledown f(x_k) \neq 0$ y como S_k es definida positiva entonces:

$$x^t S_{i+1} x \geq \alpha_i \beta^2 \bigtriangledown f(x_i)^t S_i \bigtriangledown f(x_i) > 0.$$

Por lo que $S_{i+1} > 0$, es decir, S_{i+1} es definido positivo.

□

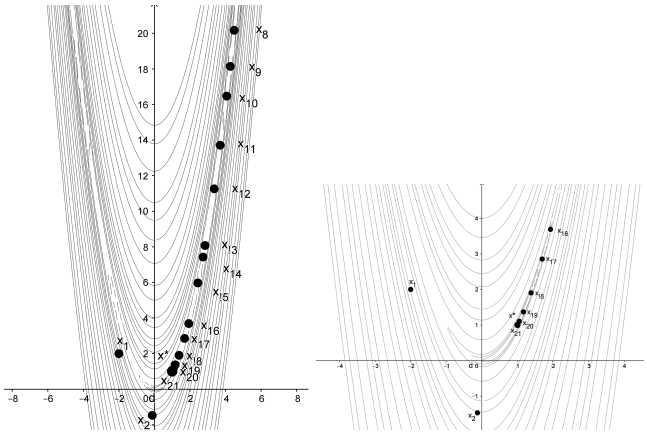
Por lo tanto, tenemos el siguiente algoritmo para la actualizacion de Davidon-Fletcher-Powell.

Algoritmo 9: Método Davidon-Fletcher-Powell

Input: punto inicial x_0, función f a minimizar, tol

```
B = matriz identidad
g = ▽f(x_0)
tol = norma(g)
epsilon = .ooooo1
while tol > epsilon do
    g = ▽f(x_0)
    s = −B^{-1} g
    x_1 = x_0 + s
    y = ▽f(x_1) − ▽f(x_0)
    x = x1
    if | s's | ¿epsilon then
        B = B + (ss')/(s's) − Byy'B/(y'By)
return x_i
```

Output: Valor más próximo al punto mínimo



(a) Primeras Iteraciones (b) Convergencia al mínimo

Figura 3.8: `Convergencia del método DFP`

3.4.3. Método Broyden-Fletcher-Goldfarb-Shanno

El método de Broyden-Fletcher-Goldfarb-Shanno (BFGS) mantiene a las actualizaciones de la matriz Hessiana que sea simétrica y definida positiva. El método se desarrollo independientemente entre los 4.

El método de BFGS se basa en la propuesta de Davidon-Fletcher-Powell. A partir

de la ecuación del método de la secante $B_{i+1}S_i = y_i$ consideraremos la definición de dualidad con $B_i \leftrightarrows H_i$ y $s_i \leftrightarrows y_i$, entonces tendremos una manera distinta de obtener la inversa del Hessiano que satisface la ecuación de la secante $H_{i+1}y_i = s_i$. Basandose en dicha dualidad obtenemos una nueva forma de aproximarnos a la matriz Hessiana.

$$B_{i+1} = B_i + \frac{y_i y_i^t}{y_i^t s_i} - \frac{B_i s_i s_i^t B_i}{s_i^t B_i s_i}.$$

Teniendo B_{i+1}, utilizaremos la fórmula de Sherman-Morrison en su segunda equivalencia para aproximar su inversa.

Definamos:

$$u_1 = v_1 = \frac{y_i}{(s_i^t y_i)^{1/2}}, u_2 = -v_2.$$

Por lo que se encontrarán $u_1 v_1^t$ y $u_2 v_2^t$,

$$u_1 v_1^t = (\frac{y_i}{(s_i^t y_i)^{1/2}})(\frac{y_i}{(s_i^t y_i)^{1/2}})^t = \frac{y_i y_i^t}{(s_i^t y_i)^{1/2}(y_i^t s_i)^{1/2}} \text{ y}$$
$$u_2 v_2^t = (\frac{B_i s_i}{(s_i^t B_i s_i)^{1/2}})(-\frac{B_i s_i}{(s_i^t B_i s_i)^{1/2}})^t = -\frac{B_i s_i s_i^t B_i}{s_i^t B_i s_i}.$$

Ahora encontraremos las entradas de la matriz C a partir de que $c_{i,j} = \delta_{i,j} + v_i^t A^{-1} u_j$ si $i = j \Rightarrow \delta = 1$, si $i \neq j \Rightarrow \delta = 0$.

Entonces tenemos que la entrada $c_{1,1}$,

$$\begin{aligned} c_{1,1} &= \delta_{1,1} + v_1^t B_i^{-1} u_1, \\ &= 1 + (\frac{y_i}{(s_i^t y_i)^{1/2}})^t (B_i^{1/2})(\frac{y_i}{(s_i^t i_k)^{1/2}}), \\ &= 1 + \frac{y_i^t H_i y_i}{S_i^t y_i}. \end{aligned}$$

para $c_{2,2}$:

$$\begin{aligned} c_{2,2} = \delta_{2,2} + v_2^t B_i^{-1} u_2 &= 1 + (-\frac{B_i s_i}{(s_i^t B_i s_i)^{1/2}})^t B_i^{-1} (\frac{B_i s_i}{(s_i^t B_k s_i)^{1/2}}), \\ &= 1 - \frac{s_i^t B_i B_i^{-1} B_i s_i}{s_i^t B_i s_i}, \\ &= 1 - 1 = 0. \end{aligned}$$

en la entrada $c_{1,2}$ se obtiene:

$$\begin{aligned} c_{1,2} &= \delta_{1,2} + v_1^t B_i^{-1} u_2 = 0 + (\frac{y_i}{(y_i^t s_i)^{1/2}})^t B_i^{-1} (\frac{B_i s_i}{(s_i^t B_i s_i)^{1/2}}), \\ &= \frac{y_i^t s_i}{(s_i^t y_i)^{1/2} (s_i^t B_i s_i)^{1/2}}, \\ &= \frac{(s_i^t y_i)^{1/2}}{(s_i^t B_i s_i)^{1/2}}. \end{aligned}$$

por último se obtiene para $x_{2,1}$ que:

$$\begin{aligned} c_{2,1} &= \delta_{2,1} + v_2^t B_i^{-1} u_1 \\ &= (-\frac{B_i s_i}{(s_i^t B_i s_i)^{1/2}}) B_i^{\frac{1}{2}} \frac{y_i}{(s_i^t y_i)^{1/2}} \\ &= -\frac{s_i^t y_i}{(s_i^t y_i s_i)^{1/2}} \\ &= -\frac{(s_i^t y_i)^{1/2}}{(s_i^t y_i . s_i)^{1/2}}. \end{aligned}$$

Por lo que la matriz C y su inversa se puede ver de la forma:

$$C = \begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & 0 \end{bmatrix} \Rightarrow C^{-1} = \frac{1}{\alpha^2} \begin{bmatrix} 0 & -\alpha \\ \alpha & \beta \end{bmatrix}.$$

Donde:

$$\alpha = c_{1,2} = c_{2,1} = \frac{(s_i^t y_i)^{1/2}}{(s_i^t y_i s_i)^{1/2}} \text{ y}$$
$$\beta = 1 + \frac{y_i^t H_i y_i}{S_i^t y_i}.$$

Ahora, si definimos $\tilde{U} = H_i U$ y $\tilde{V} = H_i V$ y utilizando la fórmula de Sherman-Morrison deducimos que:

$$H_{i+1} \leftarrow H_i - H_i U C^{-1} V^t H_i = H_i - \tilde{U} C^{-1} \tilde{V}^t.$$

donde:

$$\begin{aligned} \tilde{U} C^{-1} \tilde{V} &= [\tilde{u_1} \tilde{u_2}] \frac{1}{\alpha^2} \begin{bmatrix} 0 & -\alpha \\ \alpha & \beta \end{bmatrix} \begin{bmatrix} \tilde{v_1}^t \\ \tilde{v_2}^t \end{bmatrix}, \\ &= \frac{1}{\alpha^2} [\tilde{u_1} \tilde{u_2}] \begin{bmatrix} 0 & -\alpha \tilde{v_2}^t \\ \alpha \tilde{v_1}^t & \beta \tilde{v_2}^t \end{bmatrix}, \\ &= \frac{1}{\alpha^2} (\tilde{u_1} \alpha \tilde{v_2} + \tilde{u_2} \alpha \tilde{v_1}^t + \tilde{u_2} \beta \tilde{v_2}^t), \\ &= \frac{1}{\alpha} (\tilde{u_1} \tilde{u_2}^t + \tilde{u_2} \tilde{v_1}^t + \frac{\beta}{\alpha^2} \tilde{u_2} \tilde{v_2}^t). \end{aligned}$$

Multiplicando por H_i,

$$\tilde{U}C^{-1}\tilde{V} = \frac{1}{\alpha}(H_i\tilde{u_1}\tilde{u_2}^t H_i + H_i\tilde{u_2}\tilde{v_1}^t H_i + \frac{\beta}{\alpha^2}H_k\tilde{u_2}\tilde{v_2}^t)H_i.$$

Con lo anterior, sustituimos los valores de $\alpha\beta\tilde{u}\tilde{v}$ y obtenemos la actualización de Broyden-Fletcher-Goldfarb-Shanno.

$$H_{i+1} \leftarrow H_i - \frac{H_i y_i s_i^t + y_i^t s_i^t H_i}{s_i^t y_i} + \frac{s_i s_i^t}{s_i y_i}(1 + \frac{y_i^t H_i y_i}{s_i^t y_i}).$$

Por lo que finalmente el algoritmo del método de Broyden-Fletcher-Goldfarb-Shanno es de la siguiente manera:

Algoritmo 10: Método de Broyden-Fletcher-Goldfarb-Shanno

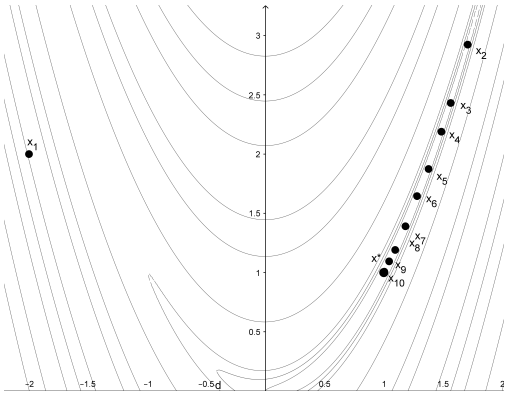
Input: punto inicial x_0, función f a minimizar, tol

B = matriz identidad
$g = \bigtriangledown f(x_0)$
$tol = norma(g)$
$epsilon = .ooooo1$
while $tol > epsilon$ **do**
    $g = \bigtriangledown f(x_0)$
    $s = -B^{-1}g$
    $x_1 = x_0 + s$
    $y = \bigtriangledown f(x_1) - \bigtriangledown f(x_0)$
    $x = x1$
    if $\mid s's \mid$ ¿$epsilon$ **then**
        $B = B + \frac{(yy')}{y's} - \frac{Bss'B}{s'Bs}$
return x_i

Output: Valor más próximo al mínimo

Por último observaremos que ocurre con el método de BFGS dado el mismo punto inicial $x_0 = (-2,2)$ y la función de Rosenbrock en la *figura 3.9.*

Finalmente se hace una comparación de los tiempos de convergencia para llegar al mínimo de la funcion de Rosenbrock teniendo como punto inicial $(-2,2)$. Lo que se puede observar en la tabla es que el método de BFGS es el que el tiene un tiempo de convergencia más rápido como se observa en el cuadro 3.1.



(a) Iteraciones

Figura 3.9: `Convergencia al mínimo por el método de BFGS`

Método	Tiempo (segundos)
Gradiente	42.123212
Gradiente Conjugado	27.1121312
Newton	6.8761231
Broyden	18.583363
DFP	8.1243245
BFGS	5.8925278

Cuadro 3.1: Tiempos de convergencia al mínimo en segundos

Capítulo 4

Puntos Críticos de la Energía de Gibbs

4.1. Metodología Propuesta

En los capítulos anteriores se describieron los métodos de optimización para buscar los puntos críticos de funciones en una variable y varias variables. Los métodos llegan a tener una desventaja si parten de un punto inicial que no se situé en una vecindad cercana al mínimo global, por lo que puede ocurrir que los métodos tarden demasiado tiempo computacionalmente para hallar un mínimo, o que los algoritmos encuentren un mínimo local y no uno global.

Dado el problema se propone buscar un punto inicial que se halle en una vecindad más cercana al mínimo global para así mejorar los tiempos de búsqueda.

Sea la función objetivo $f(x)$, tal que, $f(x) : \mathbb{R}^n \longmapsto \mathbb{R}$ y un punto inicial $x_0 \in \mathbb{R}^n$ con $x_o = (\lambda_1, \lambda_2, ..., \lambda_n)$. Donde el punto inicial x_0 se compone de n valores y cada $\lambda_i \in \mathbb{R}$.

Si se evalúa el punto inicial x_o en la función objetivo se tendrá una constante, es decir,

$$f(x_0) = f(\lambda_1, \lambda_2, ..., \lambda_n) = k,$$

con

$$k \in \mathbb{R}.$$

Por otra parte, si se tiene un nuevo punto $x_0^1 = (x_1, \lambda_2, ..., \lambda_n)$, si x_1 es una variable y $\lambda_2, ...\lambda_n$ siguen siendo las mismas constantes en $\mathbb{R}$ del punto x_0 quedándose fijas, al evaluarlo en la función objetivo se obtiene lo siguiente:

$$f^1(x_0^1) = f(x_1, \lambda_2, ..., \lambda_n).$$

La función $f^1(x_0^1)$ al evaluarse en el punto x_0^1 se tiene una función real, es decir, $f^1(x_0^1) : \mathbb{R} \longmapsto \mathbb{R}$ ya que al dejarse x_1 como variable y las otras $n-1$ valores como constantes se obtiene una función real.

A partir de la función real se busca el mínimo utilizando los métodos de optimización en una dimensión descritos en el capítulo 2, dicho mínimo si se llega a encontrar se le asignará como λ_1^*.

Continuando de la misma forma dejando como variable la entrada dos del punto x_0 y fijando las otras entradas como constantes se tendría el nuevo punto $x_0^2 = (\lambda_1, x_2, \lambda_3, ..., \lambda_n)$ al ser evaluado en la función $f(x)$ se obtendría nuevamente una nueva función real $f^2(x_0^2)$. A partir de la función se buscará el mínimo con los métodos de optimización en una dimensión y se le asignará como λ_2^*.

Si se continua con el mismo procedimiento se tendrían $n-$puntos que se conforman de la siguiente forma:

$$\begin{aligned}
x_0^1 &= (x_1, \lambda_2, ..., \lambda_n), \\
x_0^2 &= (\lambda_1, x_2, ..., \lambda_n), \\
&\vdots \\
x_0^j &= (\lambda_1, \lambda_2, ..., x_j, ..., \lambda_n), \\
&\vdots \\
x_0^n &= (\lambda_1, \lambda_2, ..., x_n),
\end{aligned}$$

cada uno de los puntos se evalúa en la función objetivo $f(X)$,

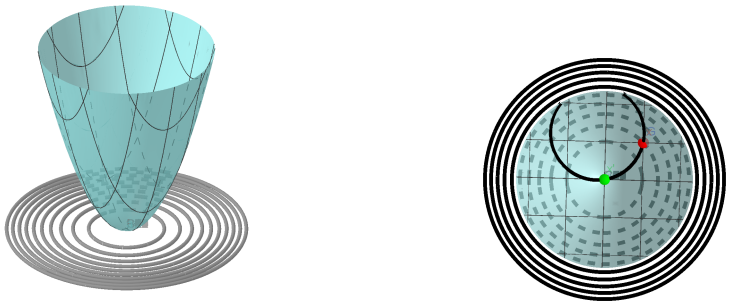
$$\begin{aligned}
f^1(x_0^1) &= f((x_1, \lambda_2, ..., \lambda_n)), \\
f^2(x_0^2) &= f((\lambda_1, x_2, ..., \lambda_n)), \\
&\vdots \\
f^j(x_0^j) &= f((\lambda_1, \lambda_2, ..., x_j, ..., \lambda_n)), \\
&\vdots \\
f^n(x_0^n) &= f((\lambda_1, \lambda_2, ..., x_n)).
\end{aligned}$$

Encontrando los mínimos de dichas funciones tenemos $\lambda_1^*, \lambda_2^*, ..., \lambda_j^*, ..., \lambda_n^*$, es decir, $n-$escalares. Por lo que se tiene el punto:

$$x_0^* = (\lambda_1^*, \lambda_2^*, ..., \lambda_j^*, ..., \lambda_n^*).$$

El punto x_0^* se conforma de los mínimos de las funciones reales, la colección de todos ellos, será el nuevo punto inicial.

En la *figura 4.1* se encuentra un paraboloide de una función $f : \mathbb{R}^2 \longmapsto \mathbb{R}$ y punto inicial $x_0 = (x, y)$ de color rojo. Dado que la gráfica de f se puede visualizar en $\mathbb{R}^3$, se muestra como ejemplo en la *figura 4.1 (a)* un paraboloide, donde se observan las curvas de nivel y el punto inicial x_0 en rojo, si se utiliza la propuesta de dejar una variable libre y las otras entradas fijas entonces se haría un corte sobre la función a partir del punto que se deja como variable y se tendría una función real donde se empieza a buscar el mínimo. Como se ve en la *figura 4.1 (b)*, se representa el punto inicial como un punto rojo y el verde es aquel que se encontró fijando cada una de las dos coordenadas de x_0 utilizando los métodos de optimización en una dimensión. A partir del nuevo punto inicial, que se encuentra en una vecindad más cercana al mínimo con lo cual logra una búsqueda más rápida.



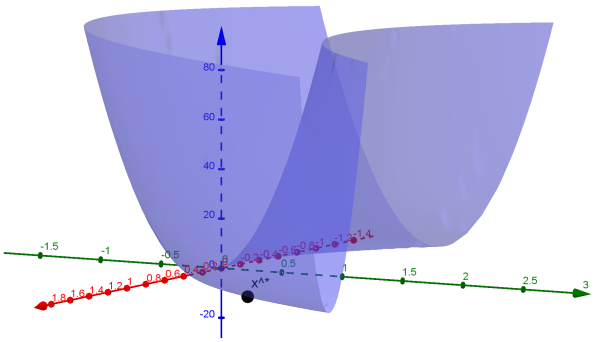
(a) Vista frontal (b) Vista superior

Figura 4.1: `Paraboloide`

4.2. Función de Rosenbrock

Para ejemplificar la metodología anterior se utilizará la función de Rosenbrock que se puede ver en la *figura 4.2* y en la *figura 4.3* se observan sus curvas de nivel.

$$f(x,y) = (1-x)^2 + 100(y-x^2)^2.$$



(a) $f(x,y) = (1-x)^2 + 100(y-x^2)^2$

Figura 4.2: `Función de Rosenbrock`

Como se vio en el capítulo 3 la función de Rosenbrock tiene un punto mínimo en $(1,1)$, en la *figura 4.2* se puede visualizar el mínimo como el punto x^* de color negro.

Dado un punto inicial $x_0 = (2,-50)$ y utilizando el método BFGS, programada se tiene que se logra encontrar el punto mínimo en un tiempo de 5.6062 segundos además, viendo las curvas de nivel en la *figura 4.3 (a)* se observa que a partir del punto inicial x_0 la convergencia al punto crítico es lenta y con un mayor número de iteraciones.

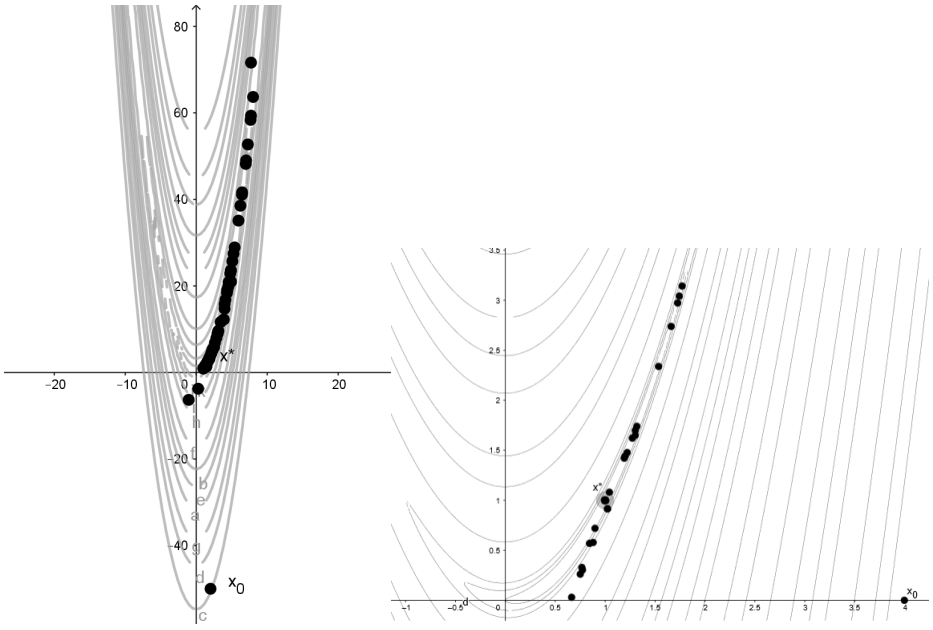
Dada dicha problemática se propone hallar un nuevo punto inicial que se encuentre en una vecindad más cercana al punto mínimo y poder mejorar el tiempo de convergencia.

Se tiene el punto inicial $x_0 = (2,-50)$ si se deja libre la segunda variable y se fija la primer entrada del punto x_0 se obtiene la siguiente función:

$$f_1(2,y) = (1-2)^2 + 100(y-(2)^2)^2 = -1 + 100(y-4)^2.$$

Por lo que se tiene $f_1(2,y) = 1 + 100(y-4)^2$. Encontrando el mínimo de la función utilizando el método de la secante se tiene que existe un punto mínimo en 3.9999, dicho punto será sustituido por la segunda entrada del punto inicial x_0, por lo que se propone que el nuevo punto inicial que definiremos como x_0^* estaría conformado por $(2,3.99999)$ siendo el punto una aproximación más cercana al mínimo.

Continuando de la misma forma dejando libre la variable x y fijando la segunda entrada del punto inicial x_0 se tiene la siguiente función:



(a) punto inicial $(2,-50)$ (b) punto inicial $(3.9, 0.0000995)$

Figura 4.3: `Curvas de nivel función de Rosenbrock`

$$f_2(x,-50) = (1-x)^2 + 100(-50-x^2)^2.$$

A partir de la función $f_2(x,-50)$, se encuentra el punto mínimo en .0000102, dicho valor será la primer entrada del nuevo punto inicial x_0^*. Por lo que finalmente se obtiene el nuevo punto inicial $x_0^* = (3.9, .0000102)$.

Los tiempos para hallar cada punto inicial de la funciones en $\mathbb{R}$ fue de 1.2343 y .9121 segundos para la primera y la segunda entrada respectivamente. Por lo tanto, se obtiene un tiempo total para encontrar el nuevo punto inicial x_0^* de 2.1464 segundos.

A partir de x_0^* y utilizando el método de BFGS, se tiene un tiempo de 1.7154 segundos para tener el punto mínimo $(1,1)$. Se puede observar en la *figura 4.3 (b)* las curvas de nivel donde se inician con el punto $(3.9, 0.0000995)$ y se ve un menor número de iteraciones que con en el punto inicial x_0^*. Por lo que el tiempo total para encontrar el punto mínimo es de 3.8618 segundos que queda conformado por el tiempo de búsqueda del nuevo punto inicial y el punto mínimo de la función de Rosenbrock.

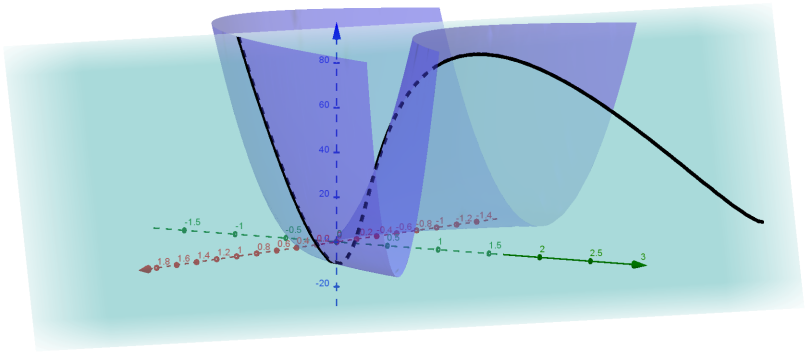
Se tiene una mejora de 1.7444 segundos con respecto al punto inicial x_0 y el tiempo total del nuevo punto inicial x_0^*, como se puede ver en el cuadro 4.1.

Gráficamente lo que se propone es que a partir de la función de Rosenbrock se ha-

Punto Inicial	Tiempo (segundos)
$x_0 = (2, -50)$	5.6062
$x_0^* = (3.9, .0000102)$	3.8618

Cuadro 4.1: Tiempos de convergencia al mínimo

ga un corte con un plano, la que se observa de color azul en la *figura 4.4* y a partir del corte se forma una función en $\mathbb{R}$ que se marca de color negro y que corta a la función de Rosenbrock, para que a partir de la función que se forma del corte, poder encontrar el mínimo y obtener un punto más cercano a la vecindad del mínimo global.



(a) función en $\mathbb{R}$

Figura 4.4: `Función de Rosenbrock`

4.3. Energía de Gibbs

Sea la función objetivo de la Energía de Gibbs:

$$g(y_i) = 1 + \sum_{i=1}^{N_c} y_i(\ln(y_i) - \ln(\phi) - h_i - 1).$$

Se utilizará dicha función para una mezcla de dos componentes, dado que se propone un estimado inicial Y_m para encontrar el mínimo. De la misma forma que se hizo con la función de Rosenbrock se propone dejar una variable de Y_m y dejar las otras constantes fijas para así obtener una función $f : \mathbb{R} \longmapsto \mathbb{R}$.

La mezcla que se utilizará se conforma por los compuestos del cuadro 4.2 y los datos de la Temperatura crítica y Presión crítica que se encuentran en el cuadro $A.2$ del apéndice.

Temperatura	Presión	$Zn2$	$Zc2$
270	76	0.18	0.82

Cuadro 4.2: Compuestos de la mezcla

Dado que, para cada mezcla se da un estimado inicial $Y_M = (x_0, y_0)$, con dicho punto se propone dejar la variable x de la función $f(x,y)$ y dejar fija a la constante $y = 0.3275$, entonces:

$$\begin{aligned} f_1(x, y_0) &= 1 + (x(\ln(x) - 0.52288) + y_0(\ln(y_0) + 0.0159)), \\ &= x(\ln(x) - 0.5228) + 0.66008. \end{aligned}$$

a partir de la función $f_1(x, y_0) = x(log(x) - 0.5228) + 0.66008$ y utilizando los métodos de optimización en $\mathbb{R}$, se tiene que el método de la secante encuentra un mínimo en $x = 0.1432$.

De la misma manera se deja la variable y además como constante a $x = 0.1243$,

$$\begin{aligned} f_2(x_0, y) &= 1 + (x(\ln(x) - 0.5228813) + y_0(\ln(y_0) + 0.0159503)), \\ &= y_0(\ln(y_0) + 0.0159503) + 0.8890609. \end{aligned}$$

para así obtener el mínimo $y^* = .2435$. Finalmente se encuentra un nuevo punto inicial Y_m^* conformada por los mínimos de las funciones anteriores, es decir, $Y_m^* = (x^*, y^*) = (0.1432, 0.2435)$.

Usando el método de BFGS, el tiempo para encontrar el punto crítico de la energía de Gibbs utilizando el punto Y_m es de 5.1231 y para el nuevo punto inicial Y_m^* se tiene un tiempo de 2.2454 segundos.

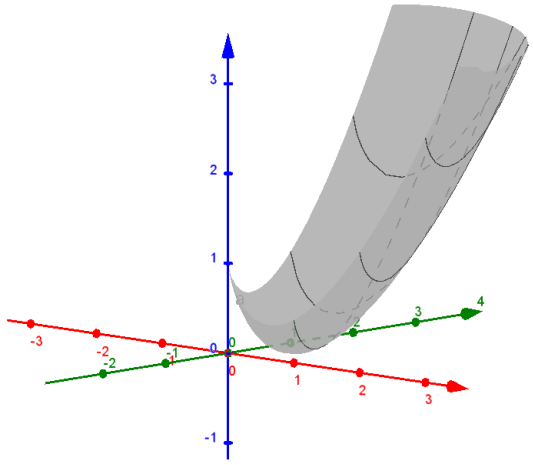
Sin embargo, el tiempo en segundos para tener el punto Y_m^* con el método de la secante en una dimensión es de 1.8792 segundos. Por lo que al sumar el tiempo anterior de 2.2454 segundos más el de 1.8792, se tiene un tiempo en segundos total de 4.1246 para encontrar el punto Y_m^*.

Con los resultados, el tiempo para hallar un punto crítico es menor si primero encontramos un punto inicial nuevo con los métodos de optimización en una dimensión y después utilizamos el método de BFGS, como se puede ver en el cuadro 4.3.

Punto Inicial	Tiempo (segundos)
Y_m	5.1231
Y_m^*	4.1246

Cuadro 4.3: Tiempos de convergencia al mínimo

Gráficamente la función de la energía de Gibbs se ve como en la *figura 4.5*. Donde la función solo tiene valores reales positivos.



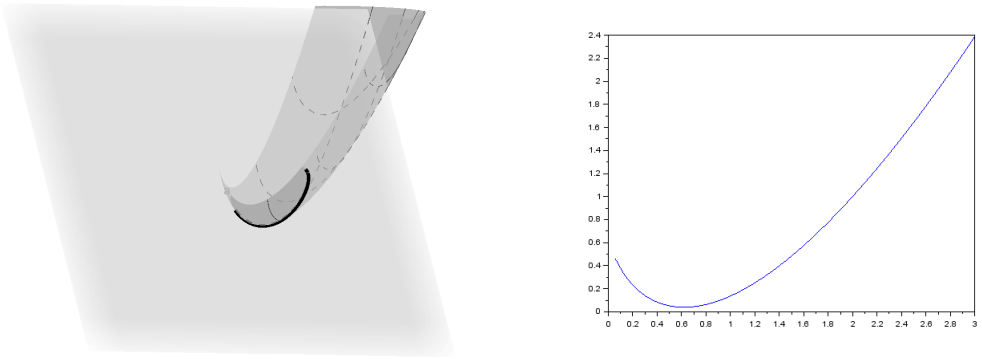
(a) $f(x,y)$

Figura 4.5: `Energia de Gibbs de dos componentes`

Geométricamente lo que se hizo es un corte con un plano sobre la función objetivo y a partir del corte se forma una función cuadrática como se ve en la *figura 4.6 (a)* donde se busca encontrar el mínimo de la función cuadrática como se ve en la *figura 4.6 (b)*, dicho mínimo será la nueva entrada del nuevo punto inicial.

4.4. Mezclas de la Energía de Gibbs

Para una mezcla de la energía de Gibbs de N_c- componentes se tiene una función objetivo $f(x) : \mathbb{R}^{N_c} \longmapsto \mathbb{R}$, es decir, dependiendo del número de componentes de la mezcla es la dimensión del dominio de la función.



(a) Corte de nivel (b) Función en R

Figura 4.6: `Energía de Gibbs de dos componentes`

Para cada mezcla se tiene un punto inicial Y_m para comenzar la búsqueda del punto crítico. Dada la propuesta antes descrita se espera encontrar un nuevo punto inicial Y_m^* para que se implemente el método de BFGS y así reducir los tiempos de convergencia al punto crítico de la función objetivo.

Se utilizan los datos de las mezclas que se encuentran en el apéndice en el cuadro $A.1$ donde se tienen 4 mezclas de la energía de Gibbs con 2 componentes, en el cuadro $A.3$ existen 4 mezclas de 3 componentes y en el cuadro $A.5$ se sitúa una mezcla de 7 componentes en el apéndice, para cada una de las mezclas se encuentra un punto crítico.

Mezcla	$Y_m = (x_o, y_o)$	$Y_m^* = (x_0^*, y_0^*)$
Mezcla 1	4.6825	2.0409
Mezcla 2	6.2801	4.9490
Mezcla 3	5.6659	2.9867
Mezcla 4	5.8128	2.0033
Mezcla 5	7.8890	2.0060
Mezcla 6	4.0397	1.9109
Mezcla 7	5.9584	2.9728
Mezcla 8	6.5221	2.0392

Cuadro 4.4: Comparación de los tiempos de convergencia

En el cuadro 4.4 se hace una comparación de los tiempos en converger al punto crítico. En la Segunda columna se ubica el tiempo en segundos de la convergencia dado el punto inicial Y_m y en la tercera columna se tiene el tiempo en segundos de la convergencia dado el nuevo punto inicial Y_m^*

Como se observa en el cuadro 4.4 los tiempos de convergencia utilizando el nuevo

punto inicial son más cortos que comenzando por el punto inicial Y_m.

Mezcla	Método en $\mathbb{R}$	Método en $\mathbb{R}^n$	Tiempo total
Mezcla 1	1.8635	2.0409	3.9045
Mezcla 2	1.0067	4.9490	6.9557
Mezcla 3	2.4072	2.9867	4.9949
Mezcla 4	2.0758	2.0033	4.0791
Mezcla 5	2.6487	2.0060	4.6547
Mezcla 6	1.9703	1.9109	3.8812
Mezcla 7	1.9463	2.9728	4.9191
Mezcla 8	0.4419	2.0392	3.4811

Cuadro 4.5: Tiempos total para encontrar el mínimo utilizando el nuevo punto inicial

Sin embargo, para tener el nuevo punto inicial se necesita utilizar los métodos en una dimensión por lo que en el cuadro 4.5, en la columna 2 se tiene el tiempo en segundos que se tarda hallar el nuevo punto inicial Y_m^* mediante los métodos en una dimensión, por otra parte en la columna 3 se encuentra el tiempo que se tarda en converger al mínimo por el método de BFGS a partir del nuevo punto inicial Y_m^*, dicho valor es el mismo al de la segunda columna del cuadro 4.4, Finalmente se hace la suma total entre el tiempo que se tarda en buscar el nuevo punto inicial y el tiempo en encontrar el punto mínimo con el método de BFGS en $\mathbb{R}^n$.

Mezcla	$Y_m = (x_o, y_o)$	$Y_m^* = (x_0^*, y_0^*)$
Mezcla 1	4.6825	3.9045
Mezcla 2	6.2801	6.9557
Mezcla 3	5.6659	4.9949
Mezcla 4	5.8128	4.0791
Mezcla 5	7.8890	4.6547
Mezcla 6	4.0397	3.8812
Mezcla 7	5.9584	4.9191
Mezcla 8	6.5221	3.4811

Cuadro 4.6: Tiempos totales en encontrar el mínimo

Finalmente, se muestra en el cuadro 4.6 los dos tiempos totales que se tardan en hallar el mínimo a partir del punto inicial Y_m y el nuevo punto encontrado Y_m^*.

Dados los dos distintos tipos de métodos de optimización en una dimensión descritos en el capítulo anterior como lo son los métodos directos y las evaluaciones. Al utilizarse uno de los métodos para mejorar el punto inicial, se hace una comparación para ver cual realiza un menor tiempo de convergencia al punto crítico de la función $f : \mathbb{R} \longmapsto \mathbb{R}$.

Mezcla	Secante	Newton	Sección Aurea
Mezcla 1	3.6825	2.9045	8.1233
Mezcla 2	4.2801	2.9598	10.6542
Mezcla 3	3.6659	3.4006	8.6531
Mezcla 4	3.8128	3.0792	12.2314
Mezcla 5	4.8890	3.6547	9.7312
Mezcla 6	4.0397	2.8812	8.1245
Mezcla 7	3.9584	2.9192	11.2575
Mezcla 8	3.5221	3.4812	9.8743

Cuadro 4.7: Métodos en R

El cuadro 4.7 se muestra la comparación de los tiempos en que se tarda computacionalmente los métodos de la Secante, Newton y la Sección Áurea en encontrar el punto mínimo. Para ello se utiliza la función de la energía de Gibss en una dimensión.

$$f(x) = x(\ln(x) + \ln(\phi) - h).$$

Se observa que el método de la Secante y Newton son los más rápidos, dichos métodos están programados de forma que, al calcular las derivadas sea de forma numérica. Por lo que se propone utilizar dichos métodos para buscar el nuevo punto inicial.

Por otro lado, se comparan los métodos de optimización en $\mathbb{R}^n$ para ver la diferencia en tiempos de convergencia para hallar el punto mínimo.

Mezcla	Gradiente	Gradiente Conjugado	Broyden	DFP	BFGS
Mezcla 1	8.1225	3.6985	5.9045	3.1214	2.9045
Mezcla 2	9.28231	3.6974	4.0198	4.1234	2.9598
Mezcla 3	10.1432	5.5465	4.4006	4.5892	3.4006
Mezcla 4	11.8128	4.8613	3.0792	5.7312	3.0792
Mezcla 5	6.8890	3.9873	6.6547	4.2153	3.6547
Mezcla 6	7.0397	4.6532	5.8812	4.1432	2.8812
Mezcla 7	6.9584	3.6398	5.9192	3.1243	2.9192
Mezcla 8	5.5221	3.4532	6.4812	4.2311	3.4565

Cuadro 4.8: Métodos en $\mathbb{R}^n$

En el cuadro 4.8 se observa que uno de los métodos que tiene menos tiempo de búsqueda es el de BFGS, entonces se utiliza el método para la búsqueda de los puntos mínimos en las funciones de $\mathbb{R}^n$.

Se utiliza una mezcla de 7 componentes con los parámetros que se muestran en el cuadro $A.7$.

Se utilizó el punto inicial Y_m y los métodos para encontrar puntos críticos de funciones en $\mathbb{R}^n$. No se pudo llegar a un punto mínimo global.

Gradiente	Gradiente Conjugado	Newton	Broyden	DFP	BFGS
65.8446	60.3063	N/D	6.2616	5.4473	4.8634

Cuadro 4.9: Tiempos de convergencia para la Mezcla de 7 componentes

Los tiempos de convergencia fueron los que se observan en el cuadro 4.9.

En el cuadro 4.9 se observa que los métodos del Gradiente y Gradiente Conjugado son los que tienen un mayor tiempo de convergencia al mínimo. Para el caso del método de Newton no se tiene un tiempo definido ya que el software no pudo calcular la matriz inversa del Hessiano de la función objetivo.

Dada la mezcla se tiene un punto inicial Y_m. Utilizando los métodos en una dimensión obtenemos nuevos puntos iniciales Y_m^*.

Los tiempos para hallar los nuevos puntos iniciales utilizando los 3 métodos de optimización son los que se muestran en el cuadro 4.10.

En el siguiente cuadro se muestra que el método de Newton es el de convergencia más rápido.

Método de Newton	Método de la secante	Método de la Sección Aurea
2.232	3.567	4.1523

Cuadro 4.10: Tiempo en segundos para encontrar un punto crítico por los métodos en una dimensión

Utilizando las 6 rutinas para buscar puntos críticos de funciones en $\mathbb{R}^n$ y teniendo los nuevos puntos iniciales obtenemos los tiempos de convergencia en el cuadro 4.11.

Gradiente	Gradiente Conjugado	Newton	Broyden	DFP	BFGS
18.8308	15.1241	N/D	4.9109	5.2878	4.2334

Cuadro 4.11: Tiempos de convergencia para la Mezcla de 7 componentes con los nuevos puntos iniciales Y_m

De la misma forma que en el recuadro 4.9, el método de Newton no converge a un punto crítico por no poder calcular la inversa de la matriz Hessiana y se observa que el método que tiene una convergencia más rápida al punto crítico es el de BFGS.

Por lo tanto con la propuesta de encontrar un nuevo punto inicial para comenzar los métodos de optimización en $\mathbb{R}^n$ se puede observar que se aceleran los tiempos

de convergencia a un punto crítico, lo cual es de suma importancia ya que hallar los puntos mínimos es parte de un proceso más complejo que tiene altos costos en tiempo y recursos computacionalmente.

Capítulo 5

Rutinas

Se desarrollaránn las rutinas programadas en el software Scilab en la versión 6.0.2. Las ventajas de utilizar del software son que es gratuito y de código abierto.

Las rutinas descritas en el Capítulo 2 fueron programadas en Scilab, además que se utilizarón las rutinas para obtener los valores de la mezcla de la energía de Gibbs que también son desarrolladas en el mismo software.

Para las rutinas de la energía de Gibbs se utilizarón mezclas de distintos componentes, sólo se escriben los códigos para la mezcla de 2 componentes, por otra parte para las mezclas de más componentes se utilizan la tablas que se describierón anteriormente.

5.1. Energía de Gibbs

Se muestran las rutinas de la energía de Gibbs para dos componentes y las rutinas que se desarrollan son las siguientes:

- Componentes para la mezcla de la energa de Gibbs de 2 componentes;
- Función Zeta;
- FNFF FNDERF FNF1;
- Fugacidad;
- h: Vector de dimensión N_c la suma de los vectores $\ln(z_i) + \ln(\phi_i)$;
- Constantes de componentes puros;
- Regla de mezclado Van Der Waals (clásica);
- Propiedades de mezcla;

- Puntos iniciales;
- Función de la energía de Gibbs.

```
//////////////////////////////////////////
///     Componentes para la mezcla      ///
///       de la energía de Gibbs        ///
///          en 2 variables             ///
//////////////////////////////////////////
// Propiedades de los componentes de la mezcla:
// Z = Fracción mol.
// TC = Temperatura crítica.
// PC = Presión crÌtica.
// W = factor acentrico.
// MI - PESO MOLECULAR (OPCIONAL).
// T : TEMPERATURA.
// P : PRESIÓN.

    IEQ = 3

//Sistema N2-C2

// NITRÓGENO - N2
    Z(1)    =   0.44
    TC(1)   = 126.2
    PC(1)   =   33.9
    W(1)    =   0.04
    Mi(1)   =   28.0

// ETANO - C2
    Z(2)    =   0.56
    TC(2)   = 305.4
    PC(2)   =   48.8
    W(2)    =   0.098
    Mi(2)   =   30.0


// Parámetros de interación binarios
    Ko = zeros(2,2)
    kb = zeros(2,2)

    Ko(1,2) = 0.08
    Ko(2,1) = 0.08

// Temperatura y Presión

    T = 270
    P = 76

// VALOR H DEPENDE DE T, P, Zi

    I_CONSTP = 0
    J = 1

////////////////////////////////////////////////////////
////                  FUNCIÓN ZETA                  ////
```

```
53 //////////////////////////////////////////////////////////
54
55
56 // SUBRUTINA QUE CALCULA LA RAIÍ DEL FACTOR DE COMPRESIBILIDAD Z.
57
58 // Variables de entrada:
59 // IEQ = 3.
60 // IFASE: escalar que indica el número de fase.
61 // A,B constantes mayores que cero.
62 //
63 // Programas requeridos:
64 // FNFF.
65 // FNFI.
66 // FNDERF.
67 //
68 // Variables de Salida:
69 // Z: Vector de Nc-componentes dependiendo el número de fases.
70 // BB0: Escalar.
71 // IEXT: Escalar.
72 // IEXTV: Escalar.
73 function [Z,BB0,IEXT,IEXTV] = ZETA(IEQ,IFASE,A,B)
74
75       if (IEQ == 1)
76 //     RKSM
77         U = 1
78         WW = 0
79         A0 = 4.933962452
80         RO0 = 0.25992105
81       else
82 //     PRSV
83         U = 2
84         WW = -1
85         A0 = 5.877359948
86         RO0 = 0.2530765866
87       end
88
89       IEXTV = 0 // AGREGADA \\
90
91       IEXT = 0
92 //    CALCULO DE LA SOLUCIÓN ANALÍTICA
93       ALFA = 1 - (U - 1)*B
94       BETA = A - U*B + (WW - U)*B^2
95       GAMA = B*(A + B*WW*(1 + B))
96       C = 3*BETA - ALFA^2
97       D = -ALFA^3+4.5*ALFA*BETA - 13.5*GAMA
98       Q =C^3+D^2
99
100      if (Q <= 0)
101        QC = -D/sqrt(-C^3)
102        TETA = acos(QC)
103
104            if (IFASE == 1)
105 //          LÍQUIDO
106            Z = (ALFA + 2*sqrt(-C)*cos((TETA + 6.283186)/3))/3
107
108                 if (Z <= B)
109                     Z = (ALFA + 2*sqrt(-C)*cos(TETA/3))/3
```

```
110                 end
111             else
112 //          VAPOR
113             Z = (ALFA + 2*sqrt(-C)*cos(TETA/3))/3
114             end
115
116       else
117         SDEL = sqrt(Q)
118         QMAS = -D + SDEL
119         QMEN = -D - SDEL
120
121     if (QMAS == 0)
122       SIGNO1 = 1
123     else
124       SIGNO1 = abs(QMAS)/(QMAS)
125     end
126
127         if (QMEN == 0)
128       SIGNO2 = 1
129     else
130       SIGNO2 = abs(QMEN)/(QMEN)
131     end
132
133       Z = (ALFA + SIGNO1*(abs(QMAS))^(1/3)+SIGNO2*(abs(QMEN))^(1/3))/3
134       end
135
136       RO = B/Z
137       FF = FNFF(RO,A,B,U,WW)
138       BB0 = 1
139 //    PRUEBA SI LA SOLUCIÓN ES APROPIADA
140       if (IFASE == 1)
141
142 //////////////////////////////////////////////
143 ////          FASE LÍQUIDA                ////
144 //////////////////////////////////////////////
145
146         IFASE2 = 0
147
148     if (IFASE2 == 1)&(IEXTL == 2) return end
149     if (IFASE2 == 3)&(IEXTW == 2) return end
150
151             if (RO > RO0)&(FF > 0.1)
152              BB0 = 1
153 //          SOLUCIÓN APROPIADA NO EXTRAPOLA
154              return
155
156             else
157 //           *************** INICIA EXTRAPOLACIÓN PARA EL LÍQUIDO
158              FF = FNFF(RO0,A,B,U,WW)
159              IEXT = 1
160                   if ((A/B) < A0)&(FF > 0.1)
161                    RO1 = RO0
162                    FF  = FNFF(RO1,A,B,U,WW)
163                    FF1 = FNF1(RO1,A,B,U,WW)
164                    FF2 = (RO1 - 0.7*RO0)*FF
165                    FF0 = FF1 - FF2*log(RO1 - 0.7*RO0)
166                    RO  = exp((B - FF0)/FF2) + 0.7*RO0
```

```
                    Z    = B/RO
                    BB0  = FNF1(RO,A,B,U,WW)/B
                    return
                   else
                    RO1 = 0.5
                    for i = 1:20
                       FF =FNFF(RO,A,B,U,WW)
                       ERROR = FF - 0.1
                         if (abs(ERROR) < 10^(-4))
                          FF   = FNFF(RO1,A,B,U,WW)
                          FF1  = FNF1(RO1,A,B,U,WW)
                          FF2  = (RO1 - 0.7*RO0)*FF
                          FF0  = FF1 - FF2*log(RO1 - 0.7*RO0)
                          RO   = exp((B - FF0)/FF2) + 0.7*RO0
                          Z    = B/RO
                          BB0  = FNF1(RO,A,B,U,WW)/B
                          return
                         else
                          DERF = FNDERF(RO1,A,B,U,WW)
                          RO1  = RO1 - (ERROR/DERF)

                               if (RO1 < RO0)
                                   RO1 = (RO1+ERROR/DERF+RO0)/2
                               end
                               if (RO1 > 1)
                                   RO1 = (RO1 + ERROR/DERF + 1)/2
                               end
                         end
                       end

                end
            end

        RO1 = RO0
        FF   = FNFF(RO1,A,B,U,WW)
        FF1  = FNF1(RO1,A,B,U,WW)
        FF2  = (RO1 - 0.7*RO0)*FF
        FF0  = FF1 - FF2*log(RO1 - 0.7*RO0)
        RO   = exp((B - FF0)/FF2) + 0.7*RO0
        Z    = B/RO
        BB0  = FNF1(RO,A,B,U,WW)/B
        return
//            TERMINA EXTRAPOLACIÓN PARA EL LÍQUIDO
  else

////////////////////////////////////////////////
////          PRUEBA PARA VAPOR              //////
////////////////////////////////////////////////

    if (IEXTV == 2) return end

            if ((A/B) < A0)
              BB0 = 1
//        NO EXTRAPOLA PARA VAPOR
              IEXT = 0
              return

```

```
224             else
225                 if (RO < RO0)&(FF > 0.1)
226                   BB0 = 1
227 //          NO EXTRAPOLA PARA VAPOR
228                   IEXT = 0
229                   return
230                 else
231 //          INICIA EXTRAPOLACION PARA VAPOR
232                   IEXT = 1
233                   RO1  = 0.1
234                 for i = 1:20
235                   FF = FNFF(RO1,A,B,U,WW)
236                   ERROR = FF - 0.1
237                     if (abs(ERROR) < 10^(-4))
238                       FF   = FNFF(RO1,A,B,U,WW)
239                       FF1  = FNF1(RO1,A,B,U,WW)
240                       FF2  = (FF*(RO0 - RO1)/2 - FF1)/(FF1*(RO0 - RO1)
    /2)^2
241                       FF3  = 2*FF2*RO1 + FF/(FF1*FF1)
242                       FF0  = 1/FF1 + FF3*RO1 - FF2*RO1^2
243                       RO   = (FF3 - sqrt(FF3^2 - 4*FF2*(FF0 - 1/B)))
    /(2*FF2)
244                       Z    = B/RO
245                       BB0  = 1
246 //          EXTRAPOLA PARA VAPOR
247                       return
248                     else
249                       DERF = FNDERF(RO1,A,B,U,WW)
250                       RO1  = RO1 - ERROR/DERF
251
252                         if (RO1 > RO0)
253                             RO1 = (RO1 + ERROR/DERF + RO0)/2
254                         end
255                         if (RO1 < 0)
256                             RO1 = (RO1 + ERROR/DERF)/2
257                         end
258                     end
259                 end
260                 end
261             end
262       end
263         RO1 = RO0
264         FF  = FNFF(RO1,A,B,U,WW)
265         FF1 = FNF1(RO1,A,B,U,WW)
266         FF2 = (FF*(RO0 - RO1)/2 - FF1)/(FF1*(RO0 - RO1)/2)^2
267         FF3 = 2*FF2*RO1 + FF/(FF1*FF1)
268         FF0 = 1/FF1 + FF3*RO1 - FF2*RO1^2
269         RO  = (FF3 - sqrt(FF3^2 - 4*FF2*(FF0 - 1/B)))/(2*FF2)
270         Z   = B/RO
271         BB0 = 1
272 //          TERMINA EXTRAPOLACIÓN  PARA VAPOR
273
274 endfunction
275
276
277
278 //////////////////////////////////////////
```

```
279 ///              Algoritmo                ///
280 ///         para obtener los              ///
281 ///           valores  de                 ///
282 ///         FNFF FNDERF FNF1              ///
283 //////////////////////////////////////////
284 //
285 // variables de entrada:
286 //
287 // RO: punto inicial
288 // NC: dimensión de la Función
289 //
290 // Variables de salida:
291 //
292 //FNFF
293 //FNDERF
294 //FNF
295 function F = FNFF(RO,A,B,U,WW)
296     F  = 1/(1-RO)^2 - (A/B)*RO*(2 + U*RO)/(1 + U*RO+WW*RO*RO)^2
297 endfunction
298 function F = FNDERF(RO1,A,B,U,WW)
299     F = 2/(1 - RO1)^3 - (A/B)*2*(1 - WW*(RO1^2)*(3 + U*RO1))/(1 + U*RO1
    +WW*RO1^2)^3
300 endfunction
301 function F = FNF1(RO1,A,B,U,WW)
302       F = RO1/(1 - RO1) - (A/B)*RO1*RO1/(1 + U*RO1 + WW*RO1^2)
303 endfunction
304
305 //////////////////////////////////////////
306 ///              Subrutina                ///
307 ///      para obtener la fugacidad        ///
308 ///        de la energía de Gibbs         ///
309 //////////////////////////////////////////
310 //
311 // Subrutina para elegir el coeficiente de fugacidad con menor
312 // energía de Gibbs
313 //
314 // Rutinas Necesarias
315 // PROP_MOD
316
317 // Variables de entrada
318 // NC: dimensión de la Función
319 // Z: Vector de dimensión NC con las fracciones de mol de los
320 //    compuesto de la mezcla
321 // H : Ln(Z) + Ln(FUG(Z))
322 // Y : Vector independiente de Nc-componentes
323 // IEQ: Constaente igual 3
324 // I_CONSTP: Constante igual a 0
325 // T : Constante de Temperatura
326 // P : Constante de Presión
327 // TC : Constante de la Temperatura crítica
328 // PC : Constante de la Presión crítica
329 // Ko : Matriz de Nc X Nc componentes de interacción binarios
330 //
331 // Variables de Salida
332 // F: Vector de Nc-Componentes que determina la fugacidad para
333 // la mezcla dada
334
```

```
335
336
337 function F = FUG(Z,Y,H,IEQ,I_CONSTP,T,TC,P,PC,W,Ko)
338     GIBV = 0
339     GIBL = 0
340
341     IFASE = 1              // Calculo de la EnergÌa Libre de
342                            // Gibbs para el lÌquido
343
344         [FL,FV] = PROP_MOD(Y,IEQ,IFASE,I_CONSTP,T,TC,P,PC,W,Ko)
345
346     for i = 1:NC
347         GIBL = GIBL + Y(i)*(log(Y(i)) + log(FL(i)) - H(i))
348     end
349     FLP=FL;
350     IFASE = 2              // Cálculo de la EnergÌa Libre
351                            // de Gibbs para el vapor
352
353          [FL,FV] = PROP_MOD(Y,IEQ,IFASE,I_CONSTP,T,TC,P,PC,W,Ko)
354
355     for i = 1:NC
356         GIBV = GIBV + Y(i)*(log(Y(i)) + log(FV(i)) - H(i))
357     end
358     FVP = FV;
359     if (GIBV < GIBL)         // Elección del coeficiente de fugacidad
360         for i = 1:NC
361             F(i) = FVP(i)
362         end
363     elseif (GIBL < GIBV)
364         for i = 1:NC
365             F(i) = FLP(i)
366         end
367     elseif (GIBL == GIBV)
368         for i = 1:NC
369             F(i) = FLP(i)
370         end
371     end
372
373 endfunction
374
375 //////////////////////////////////////////
376 ///          Subrutina para            ///
377 ///         el calculo de H            ///
378 //////////////////////////////////////////
379 //
380 // Variables de entrada
381 // NC: dimensión de la energía de Gibbs
382 // Z: Vector de dimensión NC con las fracciones de mol de los
383 //    compuesto de la mezcla
384 // IEQ: Constaente igual 3
385 // I_CONSTP: Constante igual a 0
386 // T : Constante de Temperatura
387 // P : Constante de Presión
388 // TC : Constante de la temperatura crítica
389 // PC : Constante de la presión crÌtica
390 // Ko : Matriz de Nc X Nc componentes de interacción binarios
391 //
```

```
// Variable de salida
// H: Vector de dimensión NC, constante de mezcla
//
// Rutinas Necesarias
// PROP_MOD
//


function H = VALUE_H(Z,IEQ,I_CONSTP,T,TC,P,PC,W,Ko)

// SI EL FACTOR DE COMPRESIBILIDAD TIENE MULTIPLES RAÍCES,
// SE ELIGE EL COEFICIENTE DE
// FUGACIDAD CON LA MENOR ENERGÍA DE GIBBS
    NC = length(Z)

    GIBL = 0
    GIBV = 0
    IGLOBAL = 0

    IFASE = 1              // Cálculo de la Energía Libre de Gibbs
                           // para el líquido


    [FL,FV] = PROP_MOD(Z,IEQ,IFASE,I_CONSTP,T,TC,P,PC,W,Ko)

    for i = 1:NC
        GIBL = GIBL + Z(i)*(log(Z(i)) + log(FL(i)))
    end

    for i = 1:NC
        FLP(i)=FL(i)
    end

    IFASE = 2                // Cálculo de la Energía Libre de
                             // Gibbs para el vapor

    [FL,FV] = PROP_MOD(Z,IEQ,IFASE,I_CONSTP,T,TC,P,PC,W,Ko)

    for i = 1:NC
        GIBV = GIBV + Z(i)*(log(Z(i)) + log(FV(i)))
    end

    for i = 1:NC
    FVP(i) = FV(i)
    end

    if (GIBV < GIBL)          // Eleción del coeficiente de
                              // fugacidad GIBV < GIBL
        for i = 1:NC
            F(i) = FVP(i)
        end
    elseif (GIBL < GIBV)
        for i = 1:NC
            F(i) = FLP(i)
        end
    else (GIBL == GIBV)
        for i = 1:NC
```

```
449             F(i) = FLP(i)
450         end
451     end
452     for i = 1:NC                    // Calculo de H
453         H(i) = log(Z(i)) + log(F(i))
454     end
455
456 endfunction
457
458
459 //////////////////////////////////////////
460 ///           Subrutina para             ///
461 ///     el cálculo de constantes         ///
462 ///       de componentes puros           ///
463 //////////////////////////////////////////
464 //
465 // SE ESTIMAN LOS PARÁMETROS DE COMPONENTES PUROS
466 // DE LA ECUACIÓN DE ESTADO UTILIZANDO PENG-ROBINSON
467 //
468 // Variables de entrada
469 // T : Constante de Temperatura
470 // P : Constante de Presión
471 // TC : Constante de la Temperatura crítica
472 // PC : Presión crÌtica
473 // W: Vector de Nc-Componentes
474 //
475 // Variable de salida
476 // A1,B1: Vectores de Nc-Componentes.
477 //
478
479 function [A1,B1] = CONSTP_MOD(T,TC,P,PC,W)
480
481       NC = length(W)
482
483          R1 = 0.378893
484          R2 = 1.4897153
485          R3 = -0.17131848
486          R4 = 0.0196554
487
488          for i = 1:NC
489            OMA(i) = 0.45723553
490            OMB(i) = 0.077796074
491            M = R1 + R2*W(i)+R3*W(i)^2+R4*W(i)^3
492            TR1 = T/TC(i)
493            if (T <= TC(i))
494              TR2 = sqrt(TR1)
495              ALFA(i) = (1 + M*(1 - TR2))^2
496            else
497              C = 1 + 0.5*M
498              TR2 = TR1^C
499              ALFA(i) = exp(2*(C - 1)/C*(1 - TR2))
500            end
501              A1(i) = OMA(i)*(P/PC(i))*ALFA(i)*(TC(i)/T)^2
502              B1(i) = OMB(i)*(P/PC(i))*(TC(i)/T)
503          end
504 endfunction
505
```

```
506
507 //////////////////////////////////////////
508 ///           Subrutina de la            ///
509 ///         regla de mezclado de         ///
510 ///       Van Der Waals (clásica)        ///
511 ///       de componentes puros           ///
512 //////////////////////////////////////////
513 //
514 // SE ESTIMAN LOS PARÁMETROS DE MEZCLA DE LA ECUACIÓN DE ESTADO
515 // UTILIZANDO REGLA DE MEZCLADO DE VAN DER WAALS
516 //
517 // Variables de entrada
518 // T : Constante de Temperatura
519 //
520 //
521
522 function [AS, BS, A, B]=CONSTM_MOD(X, A1, B1, T, Ko)
523
524     NC = length(X)
525     Kb = zeros(NC,NC)
526     A = 0
527     B = 0
528     DAMT = 0
529
530     for i = 1:NC
531         AS(i) = 0
532         B = B + X(i)*B1(i)
533         for j = 1:NC
534             RAIZ=sqrt(A1(i)*A1(j))
535
536             Ka = Ko(i,j)+Kb(i,j)*T/1000
537             dKadT = Kb(i,j)/1000
538
539             A = A + X(i)*X(j)*RAIZ*(1 - Ka)
540             AS(i) = AS(i) + X(j)*RAIZ*(1 - Ka)
541         end
542     end
543     for i = 1:NC
544         AS(i) = 2*AS(i)
545         BS(i) = B1(i)
546     end
547 endfunction
548
549 //////////////////////////////////////////
550 ///             Subrutina                ///
551 ///         con las propiedades          ///
552 ///            de la mezcla              ///
553 //////////////////////////////////////////
554 //
555 // Subrutina que cálcula el coeficiente de fugacidad
556 //
557 // Variables de entrada
558 // x: Composición de la mezcla
559 // IFASE : Define la fase a aprobar
560 // IFASE = 1 si es un liquido
561 // IFASE = 2 si es un vapor
562 // I_COMSTP :
```

```
563 // Y : VARIABLE IDEPENDIENTE NORMALIZADA
564 // T : Constante de Temperatura
565 // P : Constante de Presión
566 // TC : Temperatura crítica
567 // PC : Presión crÌtica
568 // Ko : Parametros de interacción binarios
569 //
570 // Rutinas Requeridas
571 // CONSTP_MOD
572 // ZETA
573 //
574
575
576 function [FL,FV] = PROP_MOD(X,IEQ,IFASE,I_CONSTP,T,TC,P,PC,W,Ko)
577      NC = length(X)
578
579       A = 0
580       B = 0
581       DAMT = 0
582       Z = 0
583       BB0 = 0
584       L = 0
585       R = 8.314
586
587       for i = 1:NC
588         AS(i) = 0
589       end
590
591       if (I_CONSTP == 0)
592
593         [A1,B1] = CONSTP_MOD(T,TC,P,PC,W)
594         I_CONSTP = 1
595         for i = 1:NC
596             A_CONST(i) = A1(i)
597             B_CONST(i) = B1(i)
598         end
599       end
600       for i = 1:NC
601           A1(i) = A_CONST(i)
602           B1(i) = B_CONST(i)
603       end
604
605         [AS,BS,A,B] = CONSTM_MOD(X,A1,B1,T,Ko)
606
607         [Z,BB0,IEXT,IEXTV] = ZETA(IEQ,IFASE,A,B)
608
609    if (IEQ > 1)
610         SQ2 = sqrt(2)
611         L = (A/(2*SQ2*B))*log((Z + B*(1 + SQ2))/(Z + B*(1 - SQ2)))
612
613       end
614       for i =1:NC
615          F = exp(BS(i)/B*(Z - 1) - log(Z - B) + L*(BS(i)/B - AS(i)/A))*
    BB0                          // los valores de BS son constantes no
    vectores
616         if (IFASE == 1)
617           FL(i) = F
```

```
618           FV = []
619         else
620           FV(i) = F
621           FL = []
622         end
623       end
624
625       if (IFASE == 1) //Líquido
626           ZL = Z
627           IEXTL = IEXT
628       else              //Vapor
629           ZV = Z
630           IEXTV = IEXTV
631       end
632
633 endfunction
634
635 //////////////////////////////////////////
636 ///              Subrutina               ///
637 ///           para los puntos            ///
638 ///              iniciales               ///
639 //////////////////////////////////////////
640 //
641 // Subrutina que calcula los 6 tipos de estimados iniciales
642 //
643 // Variables de entrada
644 // Z : Vector de dimensión Nc
645 // T : Constante de Temperatura
646 // P : Constante de Presión
647 // TC : Temperatura crítica
648 // PC : Presión crÌtica
649 // J : Número de variable Inicial
650 //
651 // Variable de salida
652 // YM Promedio
653
654 function [YM,Y] = initial_guess(Z,H,T,TC,P,PC,W,J)
655     NC = length(Z)
656     S1 = 0.0
657
658     for i = 1:NC
659         TR(i) = T/TC(i)
660         PR(i) = P/PC(i)
661         KWILSON(i) = (1/PR(i))*exp(5.37*(1 + W(i))*(1 - (1/TR(i))))
662     end
663
664     if (J == 1)
665         for i = 1:NC
666             YM(i) = Z(i)*KWILSON(i)
667         end
668     elseif (J == 2)                    //  como fase liquida
669         for i = 1:NC
670             YM(i) = Z(i)/KWILSON(i)
671         end
672     elseif (J == 3)                    //  como gas ideal
673         for i = 1:NC
674             YM(i) = exp(H(i))
```

```
675         end
676     elseif (J == 4)   // como media aritmética entre
677                       //fase de vapor y fase líquida
678         for i = 1:NC
679             YM(i) = 1/(2*(Z(i)*KWILSON(i) + Z(i)/KWILSON(i)))
680         end
681     elseif (J == 5)       //con primera inicialización propuesta por
682                           // Firoozabadi
683         for i = 1:NC
684             YM(i) = Z(i)*(KWILSON(i))**(1/3)
685         end
686     elseif (J == 6)       // Con segunda inicialización propiesta por
687                           // Firoozabadi
688         for i = 1:NC
689             YM(i) = Z(i)/(KWILSON(i))**(1/3)
690         end
691     else (J > 6)          // utilizando componentes clave
692         for i = 1:NC
693             YM(i) = 0.0001/(NC - 1)
694         end
695         YM(J - 6) = 0.9999
696     end
697
698     for i = 1:NC          // Cálculo de composición fase prueba
699         S1 = YM(i) + S1
700     end
701     for i = 1:NC
702         Y(i) = YM(i)/S1
703     end
704 endfunction
705
706
707 //////////////////////////////////////////
708 ///              Rutina                ///
709 ///          para la ecuación          ///
710 ///      de la energía de Gibbs        ///
711 //////////////////////////////////////////
712 //
713 // Rutina que obtiene la ecuación de la energía de Gibbs
714 //
715 // Variables de entrada
716 // Z :
717 // T : Constante de Temperatura
718 // P : Constante de Presión
719 // TC : Temperatura crítica
720 // PC : Presión crítica
721 //
722 // Variable de salida
723 // G : ENergía de Gibbs
724 // YM : Estimado incial
725 // H :
726 // F : Fugacidad
727 // NC : Número de componentes
728 //
729 // Rutinas necesaras
730 //
731 // Value_H
```

```
// intial_guess
// FUG
//


function [G,YM,H,F,NC] = gibss(Z,IEQ,I_CONSTP,T,TC,P,PC,W,Ko)
    NC = length(Z)

    H = VALUE_H(Z,IEQ,I_CONSTP,T,TC,P,PC,W,Ko)
        YM = Z
        S1 = 0
        for i = 1:NC
            S1   = Z(i) + S1
            Y(i) = Z(i)/S1
        end
    [YM,Y] = initial_guess(Z,H,T,TC,P,PC,W,J)
    F = FUG(Z,Y,H,IEQ,I_CONSTP,T,TC,P,PC,W,Ko)

        t=0;
    for i = 1:NC
        p=YM(i)*(log(YM(i)) + log(F(i)) - H(i) - 1) ;
        t=t+p;
        G=t+1;
    end

endfunction



[G,YM,H,F,NC] = gibss(Z,IEQ,I_CONSTP,T,TC,P,PC,W,Ko)
```

5.1.1. Rutina de la energía de Gibbs

La rutina obtiene el valor de la energía de Gibbs en el punto x.

Variables de entrada:
x: punto de la energía de Gibbs;
NC: número de componentes y dimensión de la función;
F: valor de la Fugacidad;
h: Vector de dimensión NC la suma de los vectores $\ln(z_i) + \ln(\phi_i)$.

```
//////////////////////////////////////////
///         Función de Gibbs           ///
///        de NC - componentes         ///
//////////////////////////////////////////
//
// La rutina obtiene el valor de la anergía de Gibbs en el punto x.
//
// variables de entrada:
// x: punto de la energía de Gibbs.
// NC: número de componentes y dimensión de la función.
// F: valor de la fugacidad.
//
// Variable de salida:
```

```
14 // y: Valor de la energía de Gibbs en el punto x.
15 //
16 function y = myfun(x)
17     t = 0; //
18     for i = 1:NC // la suma de las NC componentes
19     p =  x(i)*(log(x(i)) + log(F(i)) - H(i) -1);
20     t = t+p;
21     end
22     y = t+1;
23 endfunction
```

5.2. Optimización en una dimensión

Se muestran las rutinas utilizadas para encontrar los puntos críticos de la energía de Gibbs en una dimensión. Además de las rutinas de optimización.

5.2.1. Energía de Gibbs en una dimensión

Rutina que obtiene las constantes de la energía de Gibbs, para que sea en una dimensión.

Variables de entrada:

x: vector de NC componentes del punto inicial;
F: Valor de la fugacidad;
h: Vector de dimensión NC la suma de los vectores $\ln(z_i) + \ln(\phi_i)$.

5.2.2. Derivadas númericas para funciones en $\mathbb{R}$

Rutina que obtiene las derivadas de una función en $\mathbb{R}$.

derf: primera derivada de la energía de Gibbs en el punto x;
der2f: segunda derivada de la energía de Gibbs en el punto x.

Variable de entrada:
x: punto inicial.

Variables de salida:
y: valor de la derivada en el punto x.

5.2.3. Método de Newton en $\mathbb{R}$

Rutina que obtiene el punto crítico de la energía de Gibss en una dimensión.

Variable de entrada:
x: valor inicial.
Variable de salida:
x: punto crítico.

Rutinas necesarias:
derf: primera derivada de la energía de Gibbs;
der2f: segunda derivada de la energía de Gibbs.

5.2.4. Método de la Secante en $\mathbb{R}$

Rutina que obtiene el punto crítico de la energía de Gibss en una dimensión.

Variable de entrada:
x0: Primer valor inicial.
x1: Segundo valor inicial.
Variable de salida:
x: punto crítico.

Rutinas necesarias:
derf: primera derivada de la energía de Gibbs;

5.2.5. Método de la Secció Áurea

Rutina que obtiene el punto crítico de la energía de Gibss en una dimensión.

Variable de entrada:
(g,h) intervalo donde se busca un punto crítico
Variable de salida:
x: punto crítico.

Rutinas necesarias:
derf: primera derivada de la energía de Gibbs;
der2f: segunda derivada de la energía de Gibbs.

```
1  ////////////////////////////////////////
2  ///       Variables de la función      ///
3  ///             de Gibbs               ///
4  ///          en una dimensión          ///
5  ////////////////////////////////////////
6  //
7  // Rutina que obtiene las constantes de la energía de Gibbs.
8  //
9  // variables de entrada:
10 //
11 // x: vector de NC componentes del punto inicial.
```

```
// F: valor de la fugacidad.
//

function [a,b] = fun_1(x,j)
    s = 0;
    t = 0;
    a= log(F(j)) - H(j) -1;
      m = j-1;
       for i = 1:m
       p =  x(i)*(log(x(i)) + log(F(i)) - H(i) -1);
       t = t+p;
       end
       k = j+1;
       for i = k:NC
       o = x(i)*(log(x(i)) + log(F(i)) - H(i) -1);
       s = s + o;
       end
    b = s+t+1;
endfunction

//////////////////////////////////////////
///          Derivadas números          ///
///                  para               ///
///            funciones en R           ///
//////////////////////////////////////////
//
// Rutina que obtiene las derivadas de una función en R.
//
// derf: primera derivada de la energía de Gibbs en el punto x.
// der2f: segunda derivada de la energía de Gibbs en el punto x.
//
// variables de entrada:
// x: punto inicial.
//
// Variable de salida:
// y : valor de la derivada en el punto x.
function y=derf(x)
    h=.000001
    y=(f(x+h)-f(x))/h;
endfunction

function y=der2f(x)
    h=.000001
     y=(derf(x+h)-derf(x))/h;
endfunction

//////////////////////////////////////////
///          Rutina del método          ///
///             de Newton               ///
///          en una dimensión           ///
//////////////////////////////////////////
//
// Rutina que obtiene el punto crítico
// de la energía de Gibss en una dimensión.
//
// Variables de entrada:
// x: valor inicial.
```

```
69 //
70 // Variables de salida:
71 // x : punto crítico.
72 //
73 // Rutinas necesarias:
74 // derf: primera derivada de la energía de Gibbs.
75 // der2f: segunda derivada de la energía de Gibbs.
76
77 function x=Newton(d)
78     j = 1;
79     i = 2;
80     x(1)=d;
81     ea(j)=100;
82     while abs(ea(j))>= .0000001
83         x(i)=x(i-1)-(derf(x(i-1))/der2f(x(i-1)));
84         ea(j+1)=abs((x(i)-x(i-1))/x(i))*100;
85         j=j+1;
86         i=i+1;
87     end
88     x = x(i-1);
89 endfunction
90
91 ////////////////////////////////////////
92 ///              Método de           ///
93 ///               la secante         ///
94 ///           en una dimensión       ///
95 ////////////////////////////////////////
96 //
97 // Rutina que obtiene un punto crítico mediante
98 // el método de la Secante.
99 //
100 // variables de entrada:
101 // x0: Primer punto inicial.
102 // x1: Segundo punto inicial.
103 //
104 // Variables de Salida:
105 // x : punto crítico mediante el método de la Secante.
106 //
107 // Rutinas necesarias:
108 // derf : primera derivada numerica de la función.
109 //
110 function x = secante(x0,x1)
111     i = 2;
112     j = 1;
113     x(1) = x0;
114     x(2) = x1;
115     ea(j)=100;
116     while abs(ea(j))>= .0000001
117         x(i+1)=(x(i-1)*derf(x(i))-x(i)*derf(x(i-1)))/(derf(x(i))-derf(x
    (i-1)));
118         ea(j+1)=abs((x(i+1)-x(i))/x(i+1))*100;
119         j=j+1;
120         i=i+1;
121     end
122     x = x(i);
123 endfunction
124
```

```
125 //////////////////////////////////////////
126 ///           Método de la             ///
127 ///           Sección Áurea            ///
128 //////////////////////////////////////////
129 //
130 // Rutina que obtiene un punto crítico mediante
131 // el método de la Sección Áurea.
132 //
133 // variables de entrada:
134 // (g,h): Intervalo donde se busca un punto crítico.
135 //
136 // Variables de salida:
137 // z : punto crítico de la función.
138
139 function [z]=goldensection(g,h)
140
141     tol=.000001;
142     t = (sqrt(5)-1)/2;
143     x1 = g + (1 - t)*(h-g);
144 x2 = g + t*(h-g);
145 f1 = f(x1);
146 f2 = f(x2);
147
148 while ((h-g)>tol) do
149     if(f1>f2) then // si f1>f2 el minimo no estara en el intervalo [a,
    x1]
150         g = x1;// entonces el mínimo se encontrará en [x1,b]
151         x1 = x2;// x1 será el punto anterior x2 del intervalo anterior
152         f1 = f2;// No se necesita evaluar, solo f2 será el valor de f1
153         x2 = g + t*(h-g);//calculamos el nuevo x2 para el nuevo
    intervalo
154         f2= f(x2);//evaluamos el punto
155     else //por otro lado se encontrara en [a,x2]
156         h = x2; //el termino B será sustituido por x2 para el nuevo
    ciclo
157         x2 = x1;// x2 terminará siendo x1 del anterior intervalo
158         f2= f1;// de la misma forma al ser evaluado
159         x1= g +(1-t)*(h-g); // el nuevo punto del intervalo a
    determinar sera x1
160         f1 = f(x1); // se evalua x1 en la funcion
161     end
162 end
163 z = x2
164 endfunction
165
166 //////////////////////////////////////////
167 ///          Rutina para los           ///
168 ///           nuevos puntos            ///
169 ///             Iniciales              ///
170 //////////////////////////////////////////
171 //
172 // Rutina que obtiene los puntos iniciales a partir de los métodos de
173 // de Newton, la Secante y Sección Áurea.
174 //
175 // Variables de entrada:
176 //Los puntos iniciales para el método de Newton y de la Secante son
177 // .000000001 y .00000002.
```

```
178 //Para el método de la sección Áurea suponemos que el mínimo
179 // se encuentra en el intervalo (.1,.9).
180 //
181 // Variables de salida:
182 // x: Vector de Nc componentes por el método de Newton.
183 // y: Vector de Nc componentes por el método de la Secante.
184 // z: Vector de Nc componentes por el método de la Sección Áurea.
185 //
186
187 function [x,y,z]= inicial()
188 for j = 1:NC
189     [a(j),b(j)] = fun_1(YM,j);
190     a = a(j);
191     b = b(j);
192 function y = f(x)
193     y = x*(log(x) + a) + b;
194 endfunction
195    s = Newton(.000000001);              // Método de Newton
196    t = secante(.000000001,.000000002) // Método de la Secante
197    u = goldensection(.1,.9)             // Método de la sección Áurea.
198    x(j) = s;
199    y(j) = t;
200    z(j) = u;
201 end
202 endfunction
203
204 [X Y Z] = inicial()
```

5.3. Optimización en $\mathbb{R}^n$

Se desarrollan los métodos de optimización en $\mathbb{R}^n$.

5.3.1. Método del Gradiente

La primer rutina desarrolla el gradiente de función $f : \mathbb{R}^n \longmapsto \mathbb{R}$ y la evalúa en el punto x.

Variables de entrada:
x: Vector sobre el que se calcula el gradiente;
erro: escalar que indica el error admitido en la rutina.

Variable de salida:
x: vector que indica la dirección del punto máximo de la función.

La segunda rutina desarrolla el método del gradiente el cual busca un punto crítico de la función, utilizando la información del gradiente de dicha función.

Variable de entrada:
x: Vector de NC componentes del punto inicial.

Variable de salida
x: Punto crítico de la función.

Rutina necesaria:
grad: Calcula el gradiente en el punto x de la energía de Gibbs.

```
//////////////////////////////////////////
///              Gradiente              ///
///               de una                ///
///              función                 ///
//////////////////////////////////////////
//
// Rutina que obtiene el gradiente de
// forma numerica de la función en el punto x.
//
// variables de entrada:
// x: vector  sobre el que se calcula el gradiente
// erro: error admitido en la rutina.
//
// Variables de salida:
// x: vector que indica la direccón del punto máximo de la función.

function [g] = grad(x,erro)
delta =  erro/10
n =  size(x,1);
g = zeros(n,1);
e = zeros(n,1);

for i = 1:n
    e(i) = 1;
    g(i) = [myfun(x+delta*e) - myfun(x)]/delta;
    e(i) = 0;
end
endfunction

//////////////////////////////////////////
///            Método del              ///
///              Gradiente             ///
//////////////////////////////////////////
//
// Rutina que obtiene un punto crítico de la energía de Gibbs
// por el método del Gradiente.
//
// variables de entrada:
// x : vector de NC componentes, del punto inicial.
//
// Variables de salida:
// x : punto crítido de la función.
//
// Rutinas necesarias:
// grad :  calcula el gradiente en el punto x de la energía de Gibbs.

function [x] = gradiente(x)
     g = grad(x, .000001);
     while (norm (g) > .000001)
        g = grad(x, .000001) //gradiente de la función
```

```
51          [J H] = numderivative (myfun,x,1,[],"blockmat");//Hessiano de
    la función
52          a = (g'*g)/(g'*H*g) //escalar de dirección hacía el punto crí
    tico
53          x1 = x - a*g //nuevo punto
54          x = x1
55          disp(x)
56          end
57  endfunction
```

5.3.2. Método del Gradiente Conjugado

La primer rutina desarrolla el gradiente de función $f : \mathbb{R}^n \longmapsto \mathbb{R}$ y la evalúa en el punto x.

Variables de entrada:
x: Vector sobre el que se calcula el gradiente;
erro: escalar que indica el error admitido en la rutina.

Variable de salida:
x: vector que indica la dirección del punto máximo de la función.

La segunda rutina desarrolla el método del Gradiente Conjugado el cual busca un punto crítico de la función, utilizando la información del gradiente de dicha función y a partir de la segunda iteracin busca un escalar que encuentre de forma optima el punto crtico

Variable de entrada:
x: Vector de NC componentes del punto inicial.

Variable de salida
x: Punto crítico de la función.

Rutina necesaria:
grad: Calcula el gradiente en el punto x de la energía de Gibbs.

```
1  //////////////////////////////////////////
2  ///              Gradiente             ///
3  ///               de una               ///
4  ///              función               ///
5  //////////////////////////////////////////
6  //
7  // Rutina que obtiene el gradiente de
8  // forma numerica de la función en el punto x.
9  //
10 // variables de entrada:
11 // x: vector  sobre el que se calcula el gradiente
12 // erro: error admitido en la rutina.
13 //
```

```
// Variables de salida:
// x: vector que indica la direccón del punto máximo de la función.

function [g] = grad(x,erro)
delta =  erro/10
n =  size(x,1);
g = zeros(n,1);
e = zeros(n,1);

for i = 1:n
    e(i) = 1;
    g(i) = [myfun(x+delta*e) - myfun(x)]/delta;
    e(i) = 0;
end
endfunction


////////////////////////////////////////
///         Método del Gradiente       ///
///              Conjugado             ///
////////////////////////////////////////
//
// Rutina que obtiene un punto crítico de la energía de Gibbs
// por el método del Gradiente Conjugado.
//
// variables de entrada:
// x : vector de NC componentes, del punto inicial.
//
// Variables de salida:
// x : punto crítido de la función.
//
// Rutinas necesarias:
// grad :  calcula el gradiente en el punto x de la energía de Gibbs.

//NC = length(x);
erro = .000001;

function [x] = gradiente_conjugado(x)
     g = grad(x, .000001);
     while (norm (g) > .000001)
         g0 = grad(x,.000001)
        d0 = -grad(x, .000001)
        [J H] = numderivative (myfun,x,1,[],"blockmat");//Hessiano de
    la función
        a = -(g0'*d0)/(d0'*H*d0) //Escalar en dirección al punto crí
    tico
        x1 = x + a*d0 // nuevo punto inicial
        if norm(x1-x) == 0 //evaluación de la norma
            x = x1
        else
            g1 = grad(x1, .000001)
            p = (g1'*H*d0)/(d0'*H*d0) //escalar para la dirección del
    vector
            d1 = -g1 + p*d0 //Nueva dirección del vector
            x = x1
        end
        disp(x)
```

```
        end
endfunction
```

5.3.3. Método de Newton en $\mathbb{R}^n$

Rutina que obtiene un punto crítico de la energía de Gibbs por el método de Newton.

Variable de entrada:
x: Vector de NC componentes, del punto inicial.

Variables de salida:
x: Punto crítico de la energía de Gibbs.

```
////////////////////////////////////////
///              Gradiente             ///
///               de una               ///
///              función               ///
////////////////////////////////////////
//
// Rutina que obtiene el gradiente de
// forma numerica de la función en el punto x.
//
// variables de entrada:
// x: vector  sobre el que se calcula el gradiente
// erro: error admitido en la rutina.
//
// Variables de salida:
// x: vector que indica la direccón del punto máximo de la función.


function [g] = grad(x,erro)
delta =  erro/10
n =  size(x,1);
g = zeros(n,1);
e = zeros(n,1);

for i = 1:n
    e(i) = 1;
    g(i) = [myfun(x+delta*e) - myfun(x)]/delta;
    e(i) = 0;
end
endfunction

////////////////////////////////////////
///             Método de             ///
///               Newton              ///
////////////////////////////////////////
//
// Rutina que obtiene un punto crítico de la energía de Gibbs
// por el método de Newton.
//
// variables de entrada:
```

```
// x : vector de NC componentes, del punto inicial.
//
// Variables de salida:
// x : punto crítido de la función.
//
// Rutinas necesarias:
// grad :  calcula el gradiente en el punto x de la energía de Gibbs.

function [x] = metodonewton(x)
    g = grad(x,.000001);
    while norm (g) > .000001
        [J H] = numderivative (myfun,x,1,[],"blockmat");//Hessiano de
    la función
        g = grad(x,.000001); //gradiente de la función
        s = -inv(H)*g; //Nueva dirección
        x = x + s; //Nuevo punto
    end
endfunction
```

5.3.4. Método de Broyden

Rutina que obtiene un punto crítico de la energía de Gibbs.

Variables de entrada:
NC: Dimensión de la energía de Gibbs.
x: Vector de dimensión NC del punto inicial.

Variable de salida
x: Punto crítico de la energía de Gibbs.

Rutina necesarias
Gradiente: Rutina que obtiene el gradiente de la función objetivo.

```
////////////////////////////////////////
///            Método de             ///
///              Broyden             ///
////////////////////////////////////////
//
// Rutina que obtiene un punto crítico de la energía de Gibbs
// utilizando la propuesta de C. G. Broyden.
//
// Variables de entrada:
// NC: dimensión de la Energía de Gibbs.
// x: Vector de dimensión NC del punto inicial.
//
// Variables de salida:
// x: Punto critico de la energía de gibbs.
//
// Rutinas necesarias:
// Gradiente.
//

```

```
function [g] = grad(x,erro)
delta =  erro/10
n =  size(x,1);
g = zeros(n,1);
e = zeros(n,1);

for i = 1:n
    e(i) = 1;
    g(i) = [myfun(x+delta*e) - myfun(x)]/delta;
    e(i) = 0;
end
endfunction


////////////////////////////////////////
///            Método de             ///
///              Broyden             ///
////////////////////////////////////////
//
// Rutina que obtiene un punto crítico de la energía de Gibbs
// utilizando la propuesta de C. G. Broyden.
//
// Variables de entrada:
// NC: dimensión de la Energía de Gibbs.
// x: Vector de dimensión NC del punto inicial.
//
// Variables de Salida:
// x: Punto crítico de la energía de gibbs.
//
// Rutinas necesarias:
// Gradiente.
//

function [x] = broyden(x)
    B = eye(NC,NC);      // Matriz Diagonal de NCxNC
    g = grad(x,.000001);
    while norm (g) > .000001  //El gradiente en el punto x debe ser
                              // mayor al error
        g = grad(x,.000001);
        s = -inv(B)*g;        // Dirección de descenso
        x1 = x+s;
        y = grad(x1,.000001) -  grad(x,.000001);
        x = x1;
        if abs(s'*s) > .000001
           B = B + ((y-B*s)*s')/(s'*s);  //Acutalización del Hessiano
                                         //propuesta por Broyden
        end
    end
endfunction
```

5.3.5. Método de Davidon-Fletcher-Powell

Rutina que obtiene un punto crítico de la energía de Gibbs.

Variables de entrada:

NC: Dimensión de la energía de Gibbs;
x: Vector de dimensión NC del punto inicial.

Variable de salida:
x: Punto crítico de la energía de Gibbs.

Rutina necesarias
Gradiente: Rutina que obtiene el gradiente de la función objetivo.

```
//////////////////////////////////////////
///            Método de               ///
///         Davidon Fletcher           ///
///            and Powell              ///
//////////////////////////////////////////
//
// Rutina que obtiene un punto crítico de la energía de Gibbs
// utilizando la propuesta de William C. Davidon, Roger Fletcher y
// Michael J. D. Powell.
//
// Variables de entrada:
// NC: dimensión de la Energía de Gibbs.
// x: Vector de dimensión Nc del punto inicial.
//
// Variables de salida:
// x: Punto crítico de la energía de gibbs.
//
// Rutinas necesarias:
// Gradiente.
//

function [x] = DFP(x)
    B = eye(NC,NC);
    g = grad(x, .000001);
    while (norm (g) > .000001) //Mientras la norma del gradiente en
                               // x sea mayor al error se calcula el
        g = grad(x, .000001); // Gradiente evaluado en el punto x
        x1 = x - B*g;
        q = grad(x1, .000001) - grad(x, .000001);
        p = x1-x
        x = x1;
        B = B + (p*p') / (p'*q) - (B*q*q'*B)/(q'*B*q);
        // actualización del Hessiano propuesto por DFP
    end
endfunction
```

5.3.6. Método de Broyden Fletcher Goldfarb y Shanno

Rutina que obtiene un punto crítico de la energía de Gibbs.

Variables de entrada
NC: Dimensión de la energía de Gibbs.

x: Vector de dimensión NC del punto inicial.

Variable de salida
x: Punto crítico de la energía de Gibbs.

Rutina necesaria:
Gradiente: Rutina que obtiene el gradiente de la función objetivo.

```
////////////////////////////////////////
///            Método de             ///
///         Broyden Fletcher         ///
///       Goldfarb and shanno       ///
////////////////////////////////////////
//
//Rutina que obtiene un punto crítico de la energía de Gibbs mediante
// la propuesta de  Charles George Broyden, Roger Fletcher,
// Donald Goldfarb and David Shanno.
//
// Variables de entrada:
// NC: dimensión de la Energía de Gibbs.
// x: Vector de dimensión NC del punto inicial.
//
// Variables de salida:
// x: Punto crítico de la energía de Gibbs.
//
// Rutinas necesarias:
// Gradiente.
//

function [x] = BFGS(x)
   B = eye(NC,NC);       //Matriz Diagonal de NCxNC
    g = grad(x, .000001);    //Cálculo del Gradiente en el punto x
    while (norm (g) > .000001)    //El gradiente en el punto x debe ser
                                  // mayor al error
        g = grad(x, .000001)
        x1 = x - B*g              // dirección de descenso
        q = grad(x1, .000001) - grad(x, .000001)
        p = x1-x
        x = x1
        B=B+(1+(q'*B*q)/(p'*q))*((p*p')/(p'*q))-(p*q'*B+B*q*p')/(p'*q)
                     // Actualización del Hessiano propuesto por BFGS
    end
endfunction
```

Apéndice A

A.1. Factorización de Cholesky

Un método que nos permite resolver el sistema de ecuaciones $Hx = b$, es mediante la facorización de Cholesky que nos permite factorizar la matriz H siempre y cuando sea definida positiva.

$$H = LL^T.$$

Donde L es una matriz triangular inferior con elementos distintos de cero en su diagonal,.

$$LL^t = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \dots & l_{n1} \\ 0 & l_{22} & \dots & l_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & l_{nn} \end{bmatrix}$$

$$= \begin{bmatrix} l_{11}^2 & l_{21}l_{11} & \dots & l_{n1}l_{11} \\ l_{21}l_{11} & l_{21}^2 + l_{22}^2 & \dots & l_{n1}l_{21} + l_{n2}l_{22} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1}L11 & l_{n1}l_{21} + l_{n2}l_{22} & \dots & l_{n1}^2 + l_{n2}^2 + ... + l_{nn}^2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{21} & \dots & h_{n1} \\ h_{21} & h_{22} & \dots & l_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n1} & 0 & \dots & h_{nn} \end{bmatrix} = H.$$

Para obtener cada entrada de la matriz L se calcular columna por columna. Se obtendrán las primeras entradas.

Para obtener l_{11} observamos que:

$$\begin{gathered} h_{11} = l_{11}^2 \\ \Rightarrow \sqrt{h_{11}} = l_{11}, \end{gathered}$$

Obtenemos l_{21} donde:

$$\begin{gathered} h_{21} = l_{21}l_{11} \\ \rightarrow l_{21} = \frac{h_{11}}{l_{11}}, \end{gathered}$$

Para l_{22} tomamos:

$$l_{21}^2 + l_{22}^2 = h_{22},$$
$$\rightarrow l_{22} = \sqrt{h_{22} - l_{21}^2}.$$

Para encontrar los términos L_{jj} :

$$L_{jj} = \sqrt{H_{jj} - \sum_{k=1}^{j-1} L_{jk}^2}.$$

Por otro lado para los términos L_{ij} :

$$L_{i,j} = \frac{1}{L_{jj}}(H_{ij} - \sum_{k=1}^{j-1} L_{ik}L_{jk}).$$

Algoritmo 11: Factorización de cholesky

Input: Matriz H

1 n = tamao A

2 $L_{k,k} = \sqrt{Ak,k - U_{1:k-1,k}^2}$

3 $L_{k,j} = A_{k,j} - \dfrac{L_{1:k-1,k}^t L_{1:k-1,j}}{L_{k,k}}$

4 **return** x_i

Output: Matriz L triangular inferior

DEFINICIÓN. *Si $x_0, x_1, ..., x_n$ son $n+1$ números distintos y si f es una funciún cuyos valores estn dados en esos números, entonces existe un único polinomio $P(x)$ de grado a lo ms n, con la propiedad de que:*

$$f(x_k) = p(x_k) \text{ para cada } k = 0, 1, ..., n.$$

El polinomio está dado por:

$$P(x) = f(x_0)L_{n,0}(x) + \cdots + f(x_n)L_{n,n}(x) = \aleph = \sum_{k=0}^{n} f(x_n)L_{n,k}(x),$$

donde para cada $k = 0, 1, ..., n$.

$$L_{n,k}(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} = \prod_{\substack{i=0 \\ i\neq k}}^{n} \frac{(x-x_i)}{(x_k-x_i)}.$$

Teorema 5. *Sea f dos veces diferenciable continua en una vecindad, sea un punto $x^* \in R^n$. Entonces para $e \in R^n$ y la $\|e\|$ suficientemente pequeña se tiene que:*

$$f(x+e^*) = f(x^*) + \bigtriangledown f(x^*)^T e + e^T \bigtriangledown^2 f(x^*) \frac{e}{2} + \circ(\|e\|^2).$$

A.2. Mezclas de la Energía de Gibbs

Se mostrarǫn los datos de las mezclas utilizadas para las pruebas de los métodos de optimización.

Para una mezcla de dos componentes se utiliza una Temperatura = 270 y una presión = 76, el cuadro 5.1 muestra los datos de Z, en la columna J se indica con que punto inicial se inicio.

Mezcla	ZN2	ZC2	J
mezcla 1	0.44	0.56	1
mezcla 2	0.44	0.56	2
mezcla 3	0.3	0.7	1
mezcla 4	0.3	0.7	2

Cuadro A.1: Datos de la mezcla

En el cuadro 5.2 se muestran los valores de cada componentes como lo es la temperatura crítica (TC), la presión crítica, para la energía de Gibbs de dos componentes.

Componentes	TC	PC	W	Mi	k_{iC2}
N_2	126.2	33.9	0.04	28	0
C_2	305.4	48.8	0.098	30	0.08

Cuadro A.2: Valores de las Componentes

Para una mezcla de 3 componentes se utiliza las datos del cuadro 5.3, donde se muestra los datos de la componentes Z, donde se tiene una Temperatura = 270 grados Kelvin y una presión de 76 bar.

Mezcla	ZN2	ZC1	ZC2	J
mezcla 5	0.15	0.3	0.55	2
mezcla 6	0.15	0.3	0.55	3
mezcla 7	0.3	0.1	0.6	1
mezcla 8	0.3	0.1	0.6	2

Cuadro A.3: Datos de Z para una mezcla de 3 componentes

En el cuadro 5.3 se muestran los valores de cada componentes como lo es la temperatura crítica (TC), la presin crítica, para la energía de Gibbs de dos componentes.

Para una mezcla de 7 componentes se muestran en el cuadro $A.5$.

Componentes	TC	PC	W	Mi	k_{iC1}	k_{iC2}
N_2	126.2	33.9	0.04	28	0	0
C_1	190.6	46.0	0.008	16.0	0.038	0
C_2	305.4	48.8	0.098	30.0	0.08	.021

Cuadro A.4: Valores de las Componentes para una mezcla de 3

Componente	Z	Temperatura Crtica	Presin critica	w	Mw(g/mol)	$K_i co2$
CO_2	0	304.211	73.819	.225	44	
C_{5-7}	.2354	516.667	28.82	.2651	88.9	.115
C_{8-10}	.3295	590	23.743	.3644	125.7	.115
C_{11-14}	.1713	668.611	18.589	5.4987	174.4	.115
C_{15-20}	.1099	745.778	14.8	.6606	240.3	.115
C_{21-28}	.0574	812.67	11.954	.8771	336.1	.115
C_{122}	0.0965	914.889	8.523	1.2789	536.7	.115

Cuadro A.5: Mezcla de 7 componentes

Bibliografía

[1] Moore, J.W., Kotz, J.C., Stanitski, C.L., Joesten, M.D. y Wood, J.L. *El mundo de la química.* Conceptos y aplicaciones, Pearson Educación, México, 2000.

[2] Burden, Richard L, J. Douglas Faires. *Análisis Numérico*. International Thomson Editores S.A. Sétima Edición, 2002.

[3] Heath, Michael T. *Scientific Computing An Introductory Survey*. University of Ilinois at Urbana-Champaign. The McGraw-Hill Companies, 1997.

[4] Nocedal Jorge, Stephen J. Wright. *Numerical Optimization.* Springer, Springer-Verlag New York, 1999.

[5] Andreas Antoniou, Wu-Sheng-Lu. *Practiacal Optimization Algorithms and Engineering Applicatons*. Springer, Department of Electrical and Computer Engineering University of Victoria, Canada, 2007

[6] Kahaner, D. Moler, C. Nash, S. *Numerical Methods and Software.* Prentice Hall, Englewood Cliffs, New Jersey 07632, 1989.

[7] Kincaid, David & Cheney, Ward. *Numerical Analysis Mathematics of Scientific Computing*. The University of Texas Austin, Brooks/ColePublishing Company, 1991.

[8] Overton, Michael L. *Numerical Computing with IEEE Floating Point Arithmetic*. Courant Institute of Mathematical Sciences, New York University, New York, New York, 2001.