Modelos No lineales: Gauss-Newton y Newton-Raphson. Implementaciones en Maple, Mathematica, Gauss, Matlab y Excel

Jorge Mauricio Oviedo [1]

Resumen: El presente trabajo tiene por objetivo brindar una exposición clara y exhaustiva de los principales Métodos Numéricos en materia de Estimación de Modelos No lineales por mínimos cuadrados, especialmente los métodos de Gauss-Newton y Newto-Raphson. Para llevar a cabo dicha tarea se presenta una revisión teórica de tales tópicos brindándose además rutinas de programación en Matemática, Maple, Matlab, Gauss y Excel que automatizan la tarea de resolución de dichos problemas y permiten visualizar la evolución de los algoritmos. De esta manera, se logra cumplir el fin de fomentar el uso de tales métodos cuantitativos minimizando el esfuerzo de aprendizaje y resolución.

Palabras clave: Función, Ecuación algebraica, Máximo Global, Máximo Local, Derivada, Matriz Jacobiana

[1] joviedo@eco.unc.edu.ar

MODELOS NO LINEALES

La no linealidad es frecuente en los estudios de relaciones económicas pero las estimaciones de tales modelos por medio de del Método de Mínimos Cuadrados suele arrojar problemas de resolución en virtud de que el sistema de ecuaciones normales es por lo general no lineal en los parámetros.
Previamente conviene dividir a los modelos no lineales en intrínsecamente no lineales y los no intrínsecamente lineales. Estos últimos surgen cuando ante presencia de un modelo no lineal en los parámetros existe alguna transformación de los datos que permita transformarlo en lineal. Cuando dicha transformación no existe estamos ante la presencia de un modelo intrínsecamente no lineal. En estos últimos nos concentraremos en este ejercicio.
La no linealidad del sistema de ecuaciones normales que arrojan tales modelos no lineales suelen presentar problemas de resolución cuando no existe una manera algebraica de resolverlos. Ante tales circunstancias se hace necesario hallar métodos numéricos que faciliten la resolución de los mismos[2].
Para el ejercicio en cuestión, se describirán y aplicarán dos métodos de estimación basados en la minimización de la función de suma de cuadrados de los residuos: el algoritmo de **Gauss-Newton** y el algoritmo de **Newton-Raphson**

ALGORITMO DE GAUSS-NEWTON

Caso de un Parámetro:

Considerando el siguiente modelo no lineal de un único parámetro:

$$y_t = f(\mathbf{X}_t, \beta) + e_t$$

donde suponemos que e es ruido blanco con media "0" y varianza σ^2. Y el estimador mínimo cuadrado no lineal es aquel que minimiza la suma de los cuadrados de los residuos. El algoritmo de Gauss-Newton se basa en la minimización de la función de suma de cuadrados de los residuos. Será necesario, entonces, plantear esta función para ***el modelo en (1):***

$$S(\beta) = \sum e_t^{\,2} = \sum_{t=1}^{T} \left[y_t - f(x_t, \beta) \right]^2$$

La condición de primer orden para un mínimo de la función será:

$$\frac{\partial S(\beta)}{\partial \beta} = 2 \sum_{t=1}^{T} \left[y_t - f(\mathbf{x}_t, \beta) \right] \left(- \frac{df(\mathbf{x}_t, \beta)}{d\beta} \right)$$

[2] No solamente la inexistencia de solución algebraica exacta hace necesario recurrir a métodos numéricos sino que a veces la solución algebraica suele ser tan complicada de aplicar (como las fórmulas de Cardano y Tartaglia para los polinomios completos de grado tres y cuatro, caso del ejercicio) que el recurrir a un método numérico suele ser mas barato computacionalmente.

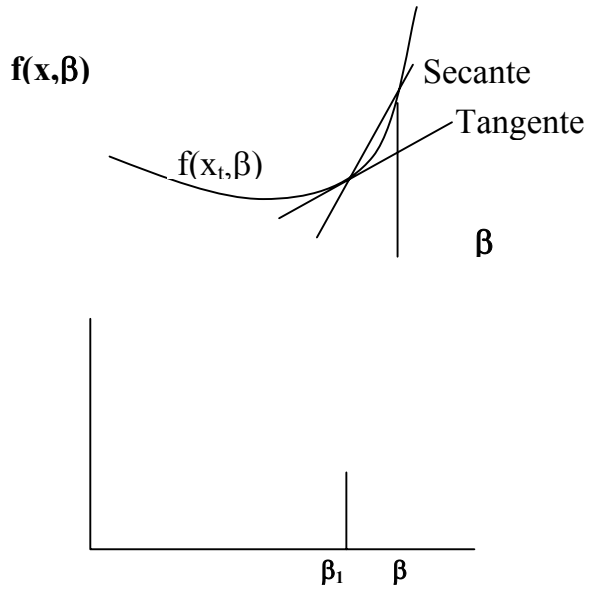
El problema de estimación es encontrar el valor de β que satisfaga la condición de primer orden anterior y que corresponda a un mínimo global de la función de suma de cuadrados. Para ello, se procederá a reemplazar $f(\mathbf{x}_t, \beta)$ por una aproximación por desarrollo en serie de Taylor de primer orden alrededor de un punto β_1:

$$\left.\frac{df(\mathbf{x}_t,\beta)}{d\beta}\right|_{\beta_1} \cong \frac{f(\mathbf{x}_t,\beta) - f(\mathbf{x}_t,\beta_1)}{(\beta - \beta_1)}$$

que puede rescribirse como:

$$\left.\frac{df(\mathbf{x}_t,\beta)}{d\beta}\right|_{\beta_1} \cong \frac{f(\mathbf{x}_t,\beta) - f(\mathbf{x}_t,\beta_1)}{(\beta - \beta_1)}$$

Donde el lado izquierdo de esta expresión es la pendiente de la tangente de la función $f(\mathbf{x}_t, \beta)$ en el punto β_1. Mientras que el lado derecho de la expresión es la línea secante de la función $f(\mathbf{x}_t, \beta)$ que pasa por los puntos β_1 y β (siendo β un valor generalizado del parámetro). Por lo tanto estamos aproximando el valor de la tangente de $f(\mathbf{x}_t, \beta)$ a partir de la secante que pasa por el punto β_1 y β (otro cualquiera) que pertenecen a la función. Mientras más cercanos sean estos puntos entre sí, mejor será la aproximación**.**



En el gráfico superior podemos ver como si fuéramos acercando el valor de β al valor de β_1, la aproximación mejoraría.

Si simplificáramos la notación como:

$$z_t(\beta_1) = \left.\frac{df(x_t,\beta)}{d\beta}\right|_{\beta_1} \Rightarrow \quad S(\beta) = \sum e_t^{\,2} = \sum_{t=1}^{T}\left[y_t - f(x_t,\beta_1) - z_t(\beta_1)(\beta - \beta_1)\right]^2$$

$$= \sum_{t=1}^{T}\left[\bar{y}_t(\beta_1) - z_t(\beta_1)\beta\right]^2$$

$$\text{donde } \bar{y}_t = y_t - f(x_t,\beta_1) - z_t(\beta_1)\beta_1$$

Para un valor determinado de β_1, tanto $\overline{y}_t(\beta_1)$ como $z_t(\beta_1)$, son observables.
La suma de cuadrados de los residuos en puede tomarse como la función a minimizar para encontrar el estimador de β por mínimos cuadrados (MCO) del siguiente modelo lineal:

$$\overline{y}_t(\beta_1) = z_t(\beta_1)\beta + e_t$$

Este modelo se lo suele denominar *pseudomodelo lineal*, el cual posee como estimador mínimos cuadrados a:

$$\beta_2 = \frac{\sum_{t=1}^{T}\left[\overline{y}_t(\beta_1) z_t(\beta_1)\right]}{\sum_{t=1}^{T}\left[z_t(\beta_1)\right]^2}$$

En notación matricial:

$$\beta_2 = \left[z(\beta_1)' z(\beta_1)\right]^{-1} z(\beta_1)' \overline{y}(\beta_1)$$

Por lo tanto, hasta ahora lo que hemos hecho consiste en partiendo de un valor inicial de β (β_1), y aproximando la función $f(x_t,\beta)$ a través de una serie de Taylor de primer orden alrededor de β_1, obtuvimos un segundo valor estimado para β (β_2) aplicando MCO al pseudomodelo lineal (el cual es una aproximación lineal del modelo general.
Repitiendo este proceso una vez más (partiendo de β_2 y aplicando MCO al pseudomodelo), obtenemos un segundo valor estimado de β:
Este proceso puede ser continuado para encontrar valores sucesivos de estimadores de β. Considerando el estimador (n+1):

$$\begin{aligned}
\beta_{n+1} &= \left[z(\beta_n)' z(\beta_n)\right]^{-1} z(\beta_n)' \overline{y}(\beta_n) \\
&= \left[z(\beta_n)' z(\beta_n)\right]^{-1} z(\beta_n)' \left[y - f(x,\beta_n) - z(\beta_n)\beta_n\right] \\
&= \beta_n + \left[z(\beta_n)' z(\beta_n)\right]^{-1} z(\beta_n)' \left[y - f(x,\beta_n)\right]
\end{aligned}$$

La condición de primer orden (CPO) para un mínimo, puede ser escrita de forma matricial como:

$$\mathbf{z}(\beta_n)'\left[\mathbf{y} - \mathbf{f}(x,\beta_n)\right] = 0$$

Si se cumple la CPO anterior, debe darse que $\beta_{n+1}=\beta_n$. Por lo tanto, comenzando con un valor inicial de $\beta=\beta_1$ y se aplica repetidamente el procedimiento descrito anteriormente (aplicar MCO al pseudomodelo) hasta que se da la convergencia entre los valores de los estimadores obtenidos, se habrá arribado a la solución de la condición de primer orden para la minimización de la suma de cuadrados del modelo no lineal dada en (1).

Una vez alcanzado el valor notable por medio del proceso iterativo anteriormente descrito, restará por determinarse si dicho valor corresponde realmente a un mínimo (o si el valor es un máximo) y si es así, si este mínimo es de carácter global o local. Para tratar de maximizar las posibilidades de que se trate de un mínimo absoluto y no tan solo de un mínimo local, una de las prácticas habituales consiste en utilizar el algoritmo para diferentes valores iniciales de β. Para los distintos valores iniciales, podemos obtener distintos mínimos de la función, el mínimo que se corresponde con la menor suma de cuadrados de los residuos será el estimador por mínimos cuadrados no lineales.

Por otra parte, el propio procedimiento no puede conducir en dirección a un máximo. Para demostrar ello, partiremos de la condición de primer orden para minimizar la suma de los cuadrados de los residuos, escrita en notación matricial, era:

$$\frac{\partial S(\beta)}{\partial \beta} = -2\mathbf{z}(\beta_n)'\left[\mathbf{y} - \mathbf{f}(x, \beta_n)\right] = 0$$

Y por lo tanto, la estimación n+1 de β se puede expresar como:

$$\beta_{n+1} = \beta_n + \frac{1}{2}\left[z(\beta_n)'z(\beta_n)\right]^{-1} \frac{\partial S(\beta)}{\partial \beta}\bigg|_{\beta_n}$$

Sabiendo que $[z(\beta)'z(\beta)]^{-1}$ será siempre positiva debido a que es una función cuadrática, si se comienza el procedimiento con un valor inicial de β situado a la derecha de un mínimo, la pendiente de la función a minimizar S(β) será positiva, por lo cual el algoritmo conducirá en la dirección correcta y obtendremos valores de β menores que β_n. Por lo tanto nos estaremos moviendo hacia un mínimo (absoluto o local) y asimismo, si se comienza con un valor inicial de β situado a la izquierda de un mínimo, la pendiente de $S(\beta)$ será negativa, por lo cual el cambio en β será positivo y nuevamente nos moveremos hacia un mínimo.

Un problema adicional que puede ocurrir es que los sucesivos cambios producidos en las estimaciones iterativas de β sean demasiados grandes y no pueda localizar el mínimo.

Puede ser conveniente introducir una variable de "longitud del paso" t_n (step length) de manera de disminuir la probabilidad de que el cambio en β sea demasiado grande y por ello se pase por alto el mínimo (es decir, se pase de un valor de β a la izquierda del β que minimiza la función, a otro a la derecha, o viceversa), lo cual puede conducir a que el número de iteraciones necesarias para arribar al valor crítico sea muy grande. Considerando esta posibilidad, el algoritmo se transforma en:

$$\beta_{n+1} = \beta_n + t_n\left[z(\beta_n)'z(\beta_n)\right]^{-1} z(\beta_n)'\left[y - f(x, \beta_n)\right]$$

El estimador obtenido por mínimos cuadrados no lineales posee las siguientes propiedades: es consistente y puede considerarse que tiene una distribución aproximadamente normal con media β y varianza $\sigma^2[z(\beta)'z(\beta)]^{-1}$. Es de destacar que σ^2 puede ser estimada por:

$$\hat{\sigma}^2 = \frac{S(\beta)}{T-1}$$

El Algoritmo de Gauss-Newton para K Parámetros

A continuación haremos extensivo los resultados de la sección anterior, pero para el caso de un modelo generalizado no lineal donde:

$$y_t = f(\mathbf{x_t}, \boldsymbol{\beta}) + e_t$$

En que $\boldsymbol{\beta}$ es un vector de *K* componentes (y por lo tanto hay que estimar K parámetros) y e se supone que son ruido blanco (de media "0" y varianza σ^2). Expresándolo de forma matricial:

$$\mathbf{y} = f(\mathbf{X}, \boldsymbol{\beta}) + \mathbf{e} \qquad ; E[\mathbf{e}] = \mathbf{0} \quad E[\mathbf{ee'}] = \sigma^2 \mathbf{I}$$

Nuevamente minimizamos la suma de cuadrado de los residuos para encontrar los estimadores:

$$S(\boldsymbol{\beta}) = \mathbf{e'e} = [\mathbf{y} - \mathbf{f}(X, \beta)]'[\mathbf{y} - \mathbf{f}(X, \beta)]$$

Donde las K condiciones de primer orden para encontrar el mínimo de esta función, representadas matricialmente son:

$$\frac{\partial S(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -2 \frac{\partial f(\mathbf{X}, \boldsymbol{\beta})'}{\partial \boldsymbol{\beta}} [y - f(\mathbf{X}, \boldsymbol{\beta})] = \mathbf{0}$$

donde $\frac{\partial f(X, \beta)'}{\partial \beta}$ es una matriz KxT de derivadas

Por lo que la condición de primer orden anterior puede ser escrita como:

$$z(\boldsymbol{\beta})[y_t - \mathbf{f}(x_t, \boldsymbol{\beta})] = \mathbf{0}$$

Nuevamente, aproximamos el problema a través de un desarrollo de Taylor de primer orden alrededor del punto inicial β_1. En ello consiste el algoritmo de Gauss-Newton, donde la aproximación de la observación t está dado por:

$$f(x_t, \boldsymbol{\beta}) \cong f(x_t, \boldsymbol{\beta}_1) + [\left.\frac{df(x_t, \beta)}{d\beta}\right|_{\beta_1} \cdots \left.\frac{df(x_t, \beta)}{d\beta}\right|_{\beta_k}](\boldsymbol{\beta} - \boldsymbol{\beta}_1)$$

Donde tendríamos T aproximaciones. Escribiendo las de forma matricial:

$$f(X,\boldsymbol{\beta}) \cong f(X,\boldsymbol{\beta_1}) + Z(\boldsymbol{\beta_1})(\boldsymbol{\beta} - \boldsymbol{\beta_1})$$

Sustituyendo esta aproximación en obtenemos:

$$\mathbf{y} \cong f(X,\boldsymbol{\beta_1}) + Z(\boldsymbol{\beta_1})(\boldsymbol{\beta} - \boldsymbol{\beta_1}) + \mathbf{e}$$

Nuevamente podemos derivar un pseudomodelo lineal, de la forma:

$$\overline{y}(\beta_1) = Z(\beta_1)\beta + e$$
$$\text{donde} \qquad \overline{y}(\beta_1) = y - f(X,\beta_1) + Z(\beta_1)\beta_1$$

Estimando (4') por mínimos cuadrados ordinarios, ya que el modelo es lineal, se obtiene un segundo valor de β:

$$\boldsymbol{\beta_2} = \left[Z(\boldsymbol{\beta_1})'Z(\boldsymbol{\beta_1})\right]^{-1} Z(\boldsymbol{\beta_1})'\overline{y}(\boldsymbol{\beta_1})$$

Continuando este proceso de manera iterativa, aplicando el algoritmo sucesivamente, llegaría a la estimación n+1 dada por:

$$\begin{aligned}\boldsymbol{\beta_{n+1}} &= \left[Z(\boldsymbol{\beta_n})'Z(\boldsymbol{\beta_n})\right]^{-1} Z(\boldsymbol{\beta_n})'\overline{y}(\boldsymbol{\beta_n}) \\ &= \left[Z(\boldsymbol{\beta_n})'Z(\boldsymbol{\beta_n})\right]^{-1} Z(\boldsymbol{\beta_n})'\left[y - f(X,\boldsymbol{\beta_n}) - Z(\boldsymbol{\beta_n})\boldsymbol{\beta_n}\right] \\ &= \boldsymbol{\beta_n} + \left[Z(\boldsymbol{\beta_n})'Z(\boldsymbol{\beta_n})\right]^{-1} Z(\boldsymbol{\beta_n})'\left[y - f(X,\boldsymbol{\beta_n})\right]\end{aligned}$$

Se habrá obtenido un mínimo de la función cuando se logre una convergencia en las estimaciones sucesivas, hasta el punto donde $\beta_n = \beta_{n+1}$. Al cumplirse dicha condición de igualdad, la CPO es satisfecha.
Las consideraciones que se hicieron para el caso de un único parámetro referentes a si el valor encontrado constituye un mínimo y no un máximo, y de ser así, si este es de carácter local o global, se mantienen para el caso de más de un parámetro: el algoritmo por sí mismo asegura que nunca se avanzará en dirección a un máximo y es conveniente partir de distintos vectores iniciales con el objeto de comparar los valores estimados y así determinar cuál es el mínimo global de la función. Además, nuevamente, las probabilidades de pasar por alto el mínimo pueden ser reducidas introduciendo una variable de longitud de paso.
Los estimadores conseguidos por este método son insesgados y tienen una distribución aproximadamente normal y la matriz de varianzas y covarianzas de los estimadores puede ser estimada consistentemente por $\sigma^{\wedge 2}[Z(\beta)'Z(\beta)]^{-1}$, donde:

$$\hat{\sigma}^2 = \frac{S(\beta)}{T - K}$$

Por lo tanto, estamos en condiciones de realizar test de hipótesis y estimación de intervalos de confianza.

Para establecer la normalidad asintótica es conveniente exigirle a la función f(X,β) que sea continua en ambos argumentos y al menos diferenciable dos veces con respecto a β. También debe ser posible invertir [Z(β)'Z(β)] (esta es una condición suficiente), o por lo menos (condición necesaria y suficiente) que (1/T)[Z(β)'Z(β)] sea no singular cuando T tiende a infinito.
Pero estas son propiedades asintóticas. No es posible derivar propiedades generales válidas para muestras pequeñas para modelos no lineales, debido a que hay diferentes especificaciones no lineales.

Algoritmo de Newton-Raphson:

El caso de un parámetro:

Partiendo nuevamente de un modelo general (en notación matricial) con un único parámetro:

$$\mathbf{y} = f(X, \boldsymbol{\beta}) + \mathbf{e}$$
$$\text{suponiendo}: \quad E[\mathbf{e}] = \mathbf{0} \quad E[\mathbf{e}\mathbf{e}'] = \sigma^2 \mathbf{I}$$

Donde β es un escalar. Se debe minimizar la suma de cuadrados de los residuos:

$$S(\boldsymbol{\beta}) = \left[y - f(X, \boldsymbol{\beta}) \right]' \left[y - f(X, \boldsymbol{\beta}) \right]$$

En este caso en vez de aproximar la función f (X,β) por un desarrollo en serie de Taylor de primer orden, se aproximará la función S(β) por un desarrollo en serie de Taylor de segundo orden alrededor de un punto inicial de β (β_1):

$$S(\beta) \cong S(\beta_1) + \left. \frac{dS(\beta)}{d\beta} \right|_{\beta_1} (\beta - \beta_1) + \frac{1}{2} \left. \frac{d^2 S(\beta)}{d\beta^2} \right|_{\beta_1} (\beta - \beta_1)^2$$

Para resolver el problema de minimización, derivamos con respecto a β la expresión anterior (el polinomio de Taylor de segundo grado) e igualamos a cero, obtenemos la CPO para encontrar un mínimo a la suma de los cuadrados de los residuos.

$$\frac{\partial S(\beta)}{\partial \beta} \cong \left. \frac{dS(\beta)}{d\beta} \right|_{\beta_1} + h(\beta_1)(\beta - \beta_1) = 0$$

Resolviendo para β, obtenemos el estimador que minimiza la aproximación propuesta para la suma de los cuadrados, el cual es un segundo estimador de β, denominado β_2:

$$\beta_2 = \beta_1 - h^{-1}(\beta_1) \left. \frac{dS(\beta)}{d\beta} \right|_{\beta_1}$$

Continuando el proceso en iteraciones sucesivas, obtenemos la iteración n-ésima del algoritmo que estará dada por:

$$\beta_{n+1} = \beta_n - h^{-1}(\beta_n) \left. \frac{dS(\beta)}{d\beta} \right|_{\beta_n}$$

Al igual que en el caso del algoritmo de Gauss-Newton, el algoritmo de Newton-Raphson continuará hasta que se logre convergencia entre las estimaciones, donde dos estimaciones de iteraciones sucesivas β_n y β_{n+1} sean iguales. Si esto sucede, entonces se cumple la CPO de la minimización de $S(\beta)$.
Surgen en este caso los mismos interrogantes que se discutieron cuando se presentó el algoritmo de Gauss-Newton sobre si el valor encontrado es un mínimo y, de ser así, sobre si éste es local o global. Debido a que en el entorno de un mínimo el término $h(\beta_n)$ será positivo, el procedimiento conducirá a un mínimo si el valor inicial de β está lo suficientemente cerca del mínimo. Sin embargo, si β_1 no se encuentra cercano a un mínimo sino a un máximo (por ej., a la derecha de un máximo), la derivada de la función será negativa y $h(\beta_1)$ será también negativo, por lo cual el procedimiento conducirá en dirección a un máximo. La cuestión es que los puntos notables pueden ser tanto máximos como mínimos (e incluso pueden darse situaciones extrañas, como puntos de silla en casos en que la derivada segunda también sea 0).
Esta característica del algoritmo de Newton-Raphson hace necesario ser cuidadoso a la hora de aplicar el método y no tomar un único valor del estimador como definitivo, por lo que es muy recomendable realizar la estimación para varios valores iniciales de β_1. Además, en el caso en que realmente se haya arribado a un mínimo, esto contribuye a determinar si el mismo es local o global.
Por último, como en el procedimiento de Gauss-Newton, existe la posibilidad de que se pase por alto el valor del mínimo, por lo cual se recomienda, como se hizo antes, la utilización de una variable de longitud de paso (t_n) para disminuir la probabilidad de que esto ocurra. La forma más general del algoritmo queda entonces establecida como:

$$\beta_{n+1} = \beta_n - t_n h^{-1}(\beta_n) \left. \frac{dS(\beta)}{d\beta} \right|_{\beta_n}$$

Para cada iteración t_n es especificado de forma tal que $S(\beta_{n+1}) < S(\beta_n)$. La variable se calcula a través de un proceso de prueba y error.

El algoritmo de newton-raphson para k parámetros:

A continuación haremos extensivo los resultados de la sección anterior, pero para el caso de un modelo generalizado no lineal donde:

$$\mathbf{y} = \mathbf{f}(\mathbf{X}, \boldsymbol{\beta}) + \mathbf{e} \qquad ; E[\mathbf{e}] = \mathbf{0} \qquad ; E[\mathbf{e}\mathbf{e}'] = \sigma^2 \mathbf{I}$$

Donde β es un vector de K componentes (y por lo tanto hay que estimar K parámetros) y e se supone que son ruido blanco. El algoritmo encuentra el valor de β que minimiza $S(\hat{a})=e'e$, y la n-ésima iteración, generalizando los resultados obtenidos en el apartado anterior, es:

$$\boldsymbol{\beta}_{\mathbf{n+1}} = \boldsymbol{\beta}_{\mathbf{n}} - t_n \mathbf{H}_n^{-1} \left. \frac{dS(\boldsymbol{\beta})}{d\boldsymbol{\beta}} \right|_{\boldsymbol{\beta}_{\mathbf{n}}}$$

Aclarando que:

$$\left. \frac{dS(\boldsymbol{\beta})}{d\boldsymbol{\beta}} \right|_{\boldsymbol{\beta}_{\mathbf{n}}} = \left[\frac{dS(\boldsymbol{\beta})}{d\beta_1}, \frac{dS(\boldsymbol{\beta})}{d\beta_2}, \mathrm{L}\ \frac{dS(\boldsymbol{\beta})}{d\beta_k} \right]_{\boldsymbol{\beta}_{\mathbf{n}}}$$

es el vector gradiente valuado en β_n

$$H_n = \left. \frac{\partial S(\beta)}{\partial \beta \partial \beta'} \right|_{\beta_n} = \left(\begin{array}{ccc} \frac{\partial^2 S(\boldsymbol{\beta})}{\partial \beta_1^2} & \mathrm{K} & \frac{\partial^2 S(\boldsymbol{\beta})}{\partial \beta_1 \partial \beta_k} \\ \mathrm{M} & \mathrm{O} & \mathrm{M} \\ \frac{\partial^2 S(\boldsymbol{\beta})}{\partial \beta_k \partial \beta_1} & \mathrm{L} & \frac{\partial^2 S(\boldsymbol{\beta})}{\partial \beta_k^2} \end{array} \right)_{\beta_n}$$

es la matriz Hessiana de (KxK) valuada en el punto β_n

Las mismas consideraciones que se hicieron para el modelo con un parámetro sobre si el valor es un máximo o un mínimo y, en este último caso, si es global o local, son válidas en esta oportunidad también, por lo que no se volverá a repetirlas.

La varianza del vector β puede ser estimada por:

$$\hat{\Sigma}(\beta) = 2\hat{\sigma} \left[\frac{\partial^2 S(\beta)}{\partial \beta \partial \beta'} \right]^{-1} \Bigg|_{\beta} \qquad \text{donde: } \hat{\sigma} = \frac{S(\beta)}{T-K}$$

<u>Relaciones entre ambos Algoritmos</u>

Si se observan las secciones anteriores se puede ver que las iteraciones de ambos procedimientos responden a la forma general:

$$\boldsymbol{\beta}_{\mathbf{n+1}} = \boldsymbol{\beta}_{\mathbf{n}} - t_n \mathbf{P}_{\mathbf{n}} \left. \frac{dS(\boldsymbol{\beta})}{d\boldsymbol{\beta}} \right|_{\boldsymbol{\beta}_{\mathbf{n}}}$$

Ambos algoritmos son solo dos de un amplio conjunto posible que responde a esta forma general.
La matriz P_n es una matriz definida positiva denominada *matriz de dirección*. Los algoritmos alternativos se diferencian en cómo definen la matriz de dirección.
Para el caso del algoritmo Gauss-Newton, se da que:

$$\mathbf{P}_{\mathbf{n}} = \frac{1}{2} \left[\mathbf{Z}(\beta_n)' \mathbf{Z}(\beta_n) \right]^{-1}$$

Y para el caso del algoritmo Newton-Raphson, se da que:

$$\mathbf{P_n} = \frac{1}{2}\left[\mathbf{Z}(\beta_n)'\mathbf{Z}(\beta_n)\right]^{-1}$$

Mientras también se incluye una variable t_n que es un número positivo denominado step lenght o longitud de paso (el cual ya fue explicado cuando correspondía).
Finalmente, entre los dos métodos estudiados, el algoritmo de Newton-Raphson es el más eficiente en el sentido que converge más rápidamente al mínimo que Gauss-Newton. Sin embargo, debe tenerse en cuenta que este último asegura que siempre el valor del estimador corresponderá a un mínimo de la función S(β), mientras que el algoritmo de Newton-Raphson puede llegar a un máximo si el punto inicial β_1 se encuentra lo suficientemente cerca de un máximo. Por último, la varianza del estimador obtenido por este último método será un poco mayor que la del estimador por Gauss-Newton.

Aplicaciones de ambos métodos

En orden de brindar un ejemplo de aplicación se propone el siguiente ejercicio:

Estimar por Mínimos cuadrados ordinarios el siguiente modelo en base a los datos que se brindan mas abajo:

$$y_t = \beta\, x_{t1} + \beta^2\, x_{t2} + e_t$$

El cual posee un único parámetro (β) y posee una variable dependiente (y) y dos variables independientes (x_1 y x_2).

y	x_1	x_2
3,284	0,286	0,645
3,149	0,973	0,585
2,877	0,384	0,310
-0,467	0,276	0,058
1,211	0,973	0,455
1,389	0,543	0,779
1,145	0,957	0,259
2,321	0,948	0,202
0,998	0,543	0,028
0,379	0,797	0,099
1,106	0,936	0,142
0,428	0,889	0,296
0,011	0,006	0,175
1,179	0,828	0,180
1,858	0,399	0,842
0,388	0,617	0,039
0,651	0,939	0,103
0,593	0,784	0,620
0,046	0,072	0,158

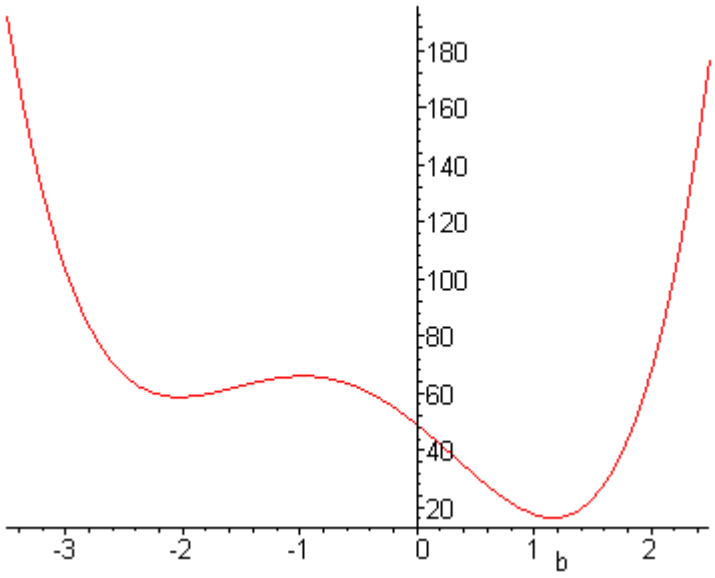
1,152	0,889	0,704

El objetivo del ejercicio es obtener el valor de b que minimice la suma de los cuadrados de los desvíos (es decir el estimador mínimo cuadrático no lineal). Nótese que este caso la función suma de los cuadrados de los desviós es un polinomio de grado 3 y en consecuencia para hallar los candidatos a extremos relativos se debe resolver una condición de primer orden que resulta ser un polinomio de grado tres. Algebraicamente existen formulas exactas para obtener estas raíces por medio de las famosas expresiones de Cardano y Tartaglia. A continuación se ofrece la representación gráfica de la función "Suma de los cuadrados de los residuos" para el rango de valores de b comprendidos entre -2 y 3.5. Se expone también la grafica de su derivada y las expresiones exactas de los puntos en donde se encuentran los valores extremos junto a expresiones numéricas con 70 dígitos de precisión.

```
with(linalg):
Da:=[[3284, 0286, 0645],[
3149, 0973, 0585],[
2877, 0384, 0310],[
-0467, 0276, 0058],[
1211, 0973, 0455],[
1389, 0543, 0779],[
1145, 0957, 0259],[
2321, 0948, 0202],[
0998, 0543, 0028],[
0379, 0797, 0099],[
1106, 0936, 0142],[
0428, 0889, 0296],[
0011, 0006, 0175],[
1179, 0828, 0180],[
1858, 0399, 0842],[
0388, 0617, 0039],[
0651, 0939, 0103],[
0593, 0784, 0620],[
0046, 0072, 0158],[
1152, 0889, 0704]]:
y1:=[seq(Da[i,1],i=1..nops(Da))]:
d1:=[seq([Da[i,2],Da[i,3]],i=1..nops(Da))]:
y:=1/1000*y1:
d:=1/1000*d1:
g:=b*x1+b^2*x2:
dd:=[x1,x2]:f:=sum((y[i]-eval(b*x1+b^2*x2,[seq(dd[j]=d[i][j],j=1..nops(dd))]))^2,
i=1..nops(y)):
plot(f,b=-3.5..2.5);

EnvExplicit := true:
exact:=solve(diff(f,b)=0,b);
aprox_digit:=evalf(r,70);
plot(diff(f,b),b=-3.5..2.5);
```

Grafica 1: Suma de los Cuadrados de los residuos



$$-\frac{45350274078347}{12}\frac{1}{\%1^{\left(\frac{1}{3}\right)}}-\frac{4414097}{7173858}+\frac{1}{2}I\sqrt{3}\left(\frac{1}{8577373100694}\%1^{\left(\frac{1}{3}\right)}-\frac{45350274078347}{6}\frac{1}{\%1^{\left(\frac{1}{3}\right)}}\right),$$

$$-\frac{1}{17154746201388}\%1^{\left(\frac{1}{3}\right)}-\frac{45350274078347}{12}\frac{1}{\%1^{\left(\frac{1}{3}\right)}}-\frac{4414097}{7173858}$$

$$-\frac{1}{2}I\sqrt{3}\left(\frac{1}{8577373100694}\%1^{\left(\frac{1}{3}\right)}-\frac{45350274078347}{6}\frac{1}{\%1^{\left(\frac{1}{3}\right)}}\right)$$

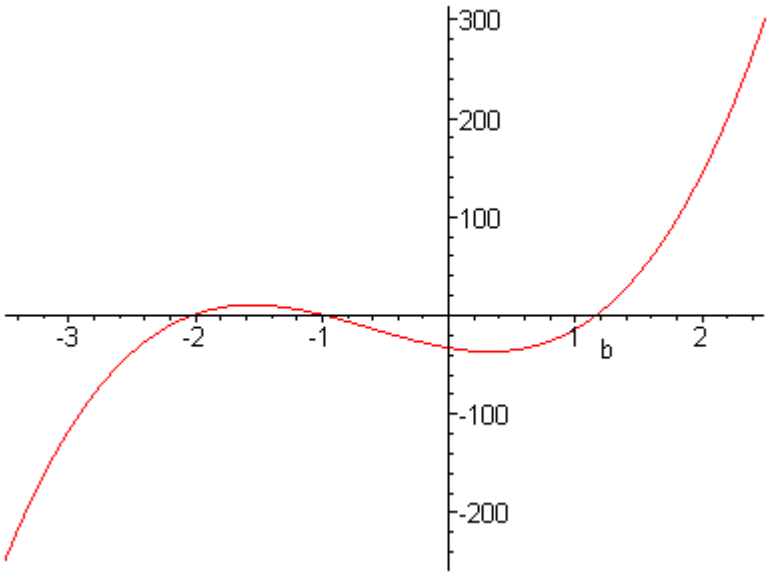
$$\%1 := 28720966885519651923950559484590678496 4$$
$$+1429562183449\,I\sqrt{92970783970401560024011502584017988535169999193082731}$$

$$aprox\_digit := 1.16120662846748055053431827571132328581494927450363689202483835410587 3 + .2\,10^{-69}\,I,$$

$$-2.02949440709429704327932117732109727970979622142981272147456707594501 9 + .2\,10^{-69}\,I,$$

$$-.97762149362250876571425291304393116726111683155954273369997399042530 15 - .2\,10^{-69}\,I$$

Grafica 2: Derivada primera de S(β)



De la observación directa del gráfico surgen algunas cuestiones a tener en cuenta: en primer lugar, el mínimo global parece darse para un valor de b entre 1 y 1.5. Por otra parte, la función tiene además un máximo y un mínimo global, lo cual ayudará para ilustrar las dificultades de los dos métodos utilizados en reconocer mínimos globales, locales y máximos**.** Otros puntos notables extraños que puedan provocar dificultades a la hora de encontrar mínimos y llevar el algoritmo en la dirección equivocada, no serán tenidos en cuenta ya que no se presentan en la función elegida y solo se mencionará dicha posibilidad al final de este ejercicio. Si bien la suma de cuadrados posee otros puntos notables (dos puntos de inflexión), éstos no son extraños y no provocan "malos comportamientos" en los algoritmos.

A continuación se resolverá el ejercicio planteado mediante la utilización de diversos softwares en los cuales se implementaron los Algoritmos anteriormente mencionados.

GAUSS LIGHT 4.0

Algoritmo Gauss Newton: se reproducen a continuación las programaciones utilizadas:

```
       load dat[20,3]=datos2.txt;
       cls;
       y=dat[.,1];
       x1=dat[.,2];
       x2=dat[.,3];
       proc(1)=fnb(beta);
           retp(x1*beta+x2*(beta^2));
       endp;

       /* procedimiento para estimar los parámetros por MCNL con el
algoritmo de Gauss-Newton */
       proc(2)=GaussNewton(b0,&fi,y);
           local fi:proc;
           local crit, iter, k, s, u, sse, b, covb, ss1, ss2, u1, u2,
sighat, z, b1, b2, bn, h;
           crit=1;
           iter=1;
           k=rows(b0);
           ?; ?;
           " valor inicial:" b0';
           ?;
           "        Nro Iter           b               S(b)    ";
           ?;
           do until crit<0.0000001 or iter>25;
               z=gradp(&fi,b0);
               u=y-fi(b0);
               sse=u'u;
               h=invpd(z'z)*z'*(y-fi(b0));
               iter b0' sse ;
               b=b0+h;
               iter=iter+1;
               crit=abs(h);
               b0=b;
           endo;
           ?;
           sighat=sse/(rows(y)-k);
           covb=sighat*invpd(z'z);
           ?;?;"Resultados Finales:"; ?;
           "Coeficientes :" b0;
           "Desviación estándar:" sqrt(diag(covb));
           "sighat :" sighat;
           retp(b0,z);

       endp;

       ?;"=======================================================";
         "Estimación de beta, a partir de cuatro valores iniciales";
       ?;"=======================================================";
       b0=4;
       {bn,z}=GaussNewton(b0,&fnb,y);
       ?;"_____________________________________________________________";
       b0=-3;
```

```
{bn,z}=GaussNewton(b0,&fnb,y);
?;"_____________________________________________________________";

b0=-1.05;
{bn,z}=GaussNewton(b0,&fnb,y);
?;"_____________________________________________________________";

b0=-0.9;
{bn,z}=GaussNewton(b0,&fnb,y);
?;"_____________________________________________________________";
```

==========

Estimación de beta, a partir de cuatro valores iniciales

==========

valor inicial: 1.0000000

Nro Iter	b	S(b)
1.0000000	1.0000000	17.475884
2.0000000	1.1633106	16.308187
3.0000000	1.1612626	16.307970
4.0000000	1.1612081	16.307970
5.0000000	1.1612067	16.307970

Resultados Finales:

Coeficientes : 1.1612066
Desviación estándar: 0.13066609
sighat : 0.85831421

---

valor inicial: -2.0000000

Nro Iter	b	S(b)
1.0000000	-2.0000000	58.543856
2.0000000	-2.0210412	58.525137
3.0000000	-2.0270802	58.523569
4.0000000	-2.0288056	58.523440
5.0000000	-2.0292979	58.523430

6.0000000	-2.0294384	58.523429
7.0000000	-2.0294784	58.523429
8.0000000	-2.0294898	58.523429
9.0000000	-2.0294931	58.523429
10.000000	-2.0294940	58.523429
11.000000	-2.0294943	58.523429

Resultados Finales:

Coeficientes : -2.0294944
Desviación estándar: 0.30239316
sighat : 3.0801805

---

valor inicial: 0.10000000

Nro Iter	b	S(b)
1.0000000	0.10000000	45.529003
2.0000000	1.5167318	23.698761
3.0000000	1.1953980	16.366227
4.0000000	1.1623622	16.308035
5.0000000	1.1612371	16.307970
6.0000000	1.1612074	16.307970
7.0000000	1.1612066	16.307970

Resultados Finales:

Coeficientes : 1.1612066
Desviación estándar: 0.13066609
sighat : 0.85831421

---

valor inicial: -0.90000000

Nro Iter	b	S(b)
1.0000000	-0.90000000	65.840064

```
     2.0000000     -0.68930728      64.498203
     3.0000000      0.32268818      37.475993
     4.0000000      1.3672489       18.615783
     5.0000000      1.1754547       16.317983
     6.0000000      1.1616250       16.307979
     7.0000000      1.1612176       16.307970
     8.0000000      1.1612069       16.307970
     9.0000000      1.1612066       16.307970
```

Resultados Finales:

Coeficientes : 1.1612066
Desviación estándar: 0.13066609
sighat : 0.85831421

Con respecto a los resultados obtenidos, ellos son los mismos que los reproducidos en el libro de Judge en la página 507. Por lo tanto, el objetivo del problema planteado se verifica.
El programa también calcula la desviación estándar del estimador y el valor estimado (sighat) de:

$$\hat{\sigma}^2 = \frac{S(\beta)}{T-K}$$

Algoritmo Newton Raphson

```
/* Programa para estimar parámetros por mínimos cuadrados no lineales
por el método de Newton-Raphson */

load dat[20,3]=datos2.txt;
cls;
cls;
y=dat[.,1];
x1=dat[.,2];
x2=dat[.,3];
proc(1)=fnb(beta);
    retp((y-beta*x1-x2*(beta^2))'*(y-beta*x1-x2*(beta^2)));
endp;
```

```
/*Procedimiento para estimar parámetros por mínimos cuadrados no
lineales utilizando el método de Newton-Raphson con derivadas
analíticas */
proc(2)=Newton(b0,&fi,y);
    local fi:proc;
    local  crit, iter, k, s, u, sse, b, covb, ss1, ss2, u1, u2,
sighat, b01, b02, df, d2f, b1, b2, bn,h;
    crit=1;
    iter=1;
    k=rows(b0);
    ?; ?; ?;
    " valor inicial:" b0'; ?;
    "       Nro Iter             b               S(b)     "; ?;
    do until crit<1e-7 or iter>25;
        df=gradp(&fi,b0);
        h=hessp(&fi,b0);
        sse=fi(b0);
        iter b0 sse;
        b=b0-inv(h)*df;
        iter=iter+1;
        crit=abs(b-b0);
        b0=b;
    endo;
    ?;
    sighat=sse/(rows(y)-k);
    covb=sighat*invpd(df'df);
    "Resultados Finales:"; ?;
    "Coeficientes :" b0;
    "Desviación estándar:" sqrt(diag(covb));
    "sighat :" sighat;
    retp(b0,df);

endp;
"Estimación de beta, a partir de cuatro valores iniciales";
b0=1;
{bn,df}=Newton(b0,&fnb,y);
b0=-2;
{bn,df}=Newton(b0,&fnb,y);
b0=0.1;
{bn,df}=Newton(b0,&fnb,y);
b0=-0.9;
{bn,df}=Newton(b0,&fnb,y);
```

Estimación de beta, a partir de cuatro valores iniciales

valor inicial: 1.0000000

Nro Iter	b	S(b)
1.0000000	1.0000000	17.475884
2.0000000	1.1863055	16.339215

3.0000000	1.1616846	16.307981
4.0000000	1.1612068	16.307970
5.0000000	1.1612066	16.307970

Resultados Finales:

Coeficientes : 1.1612066
Desviación estándar: 0.093626478
sighat : 0.85831421

valor inicial: -2.0000000

Nro Iter	b	S(b)
1.0000000	-2.0000000	58.543856
2.0000000	-2.0306642	58.523462
3.0000000	-2.0294962	58.523429
4.0000000	-2.0294944	58.523429

Resultados Finales:

Coeficientes : -2.0294944
Desviación estándar: 0.25290953
sighat : 3.0801805

valor inicial: 0.10000000

Nro Iter	b	S(b)
1.0000000	0.10000000	45.529003
2.0000000	-2.0962680	58.636893
3.0000000	-2.0344532	58.524023
4.0000000	-2.0295253	58.523429
5.0000000	-2.0294944	58.523429

Resultados Finales:

Coeficientes : -2.0294944
Desviación estándar: 0.25290953
sighat : 3.0801805

valor inicial: -0.90000000

Nro Iter	b	S(b)
1.0000000	-0.90000000	65.840064
2.0000000	-0.97528541	65.939519
3.0000000	-0.97761903	65.939607
4.0000000	-0.97762146	65.939607

Resultados Finales:

Coeficientes : -0.97762146
Desviación estándar: .
sighat : 3.4705057

Comparación entre los Algoritmos:

Para apreciar con mayor nitidez las ventajas y desventajas de cada algoritmo se procede a construir la siguiente tabla similar a la que se plantea en el texto de donde se tomó el ejercicio:

	Gauss-Newton	**Newton-Raphson**
β_0= *1*		
Iteraciones	5	5
Coeficientes	1.1612066	1.1612066
$\sigma(\beta)$	0.13066609	0.13066610
$\sigma^2(e)$	0.85831421	0.85831421
β_0= **-2**		
Iteraciones	11	4
Coeficientes	-2.0294944	-2.0294944
$\sigma(\beta)$	0.30239316	0.30239314
$\sigma^2(e)$	3.0801805	3.0801805
β_0= **0.1**		
Iteraciones	7	5
Coeficientes	1.1612066	-2.0294944
$\sigma(\beta)$	0.13066609	0.30239314
$\sigma^2(e)$	0.85831421	3.0801805
β_0= **-0.9**		
Iteraciones	9	1
Coeficientes	1.1612066	-0.90000000
$\sigma(\beta)$	0.13066609	0.75045196
$\sigma^2(e)$	0.85831421	3.4652665

Comparando los resultados de los dos procedimientos, se ve que siempre el algoritmo de Newton-Raphson logra la convergencia en menor número de iteraciones que el de Gauss-Newton; sin embargo, en el primer caso, el estimador de β posee una varianza mayor o igual al de Gauss-Newton, por lo que se puede concluir que Gauss-Newton es más eficiente, y el estimador de la varianza del término de perturbaciones (e) también es siempre menor.

Otra ventaja de Gauss-Newton es que alcanza el mínimo global en tres de los cuatro casos planteados, mientras que el primero lo hace sólo en un caso, cuando el valor de inicio está lo suficientemente próximo al mínimo. Esto conduce a concluir que el algoritmo de Gauss-Newton es más confiable que el de Newton-Raphson para encontrar el valor del estimador que minimiza la suma de cuadrados de los residuos de un modelo no lineal.

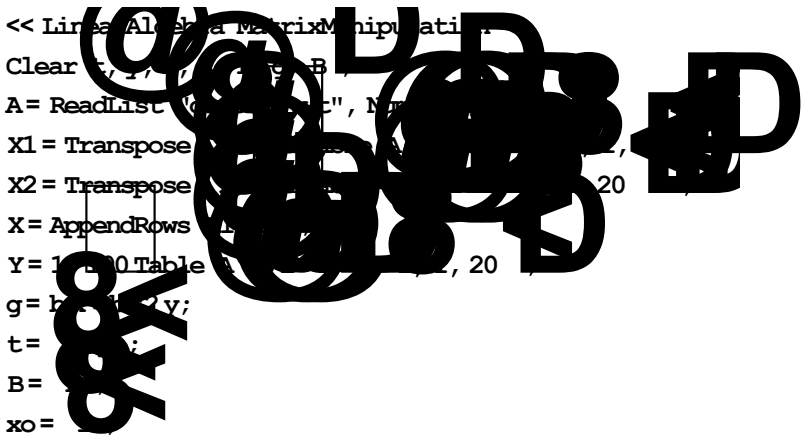
Otra ventaja del Gauss-Newton se puede observar claramente en el último caso planteado, cuando β_0=0.9. En dicho caso, Gauss-Newton nos lleva efectivamente hacia el mínimo de la función, pero Newton-Raphson no ya que no puede distinguir entre un máximo y un mínimo.

MATHEMATICA 4.0

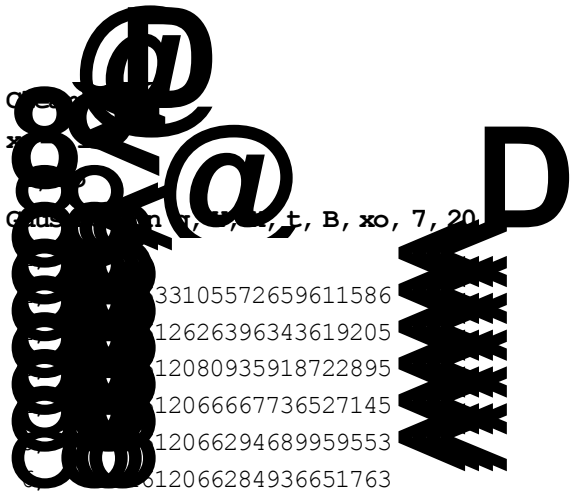
Se exponen a continuación las distintas programaciones que se realizaros en Mathematica 4.0 en donde previamente se definieron dos nuevas funciones. Los programas se presentan a continuación:

Inicializaciones

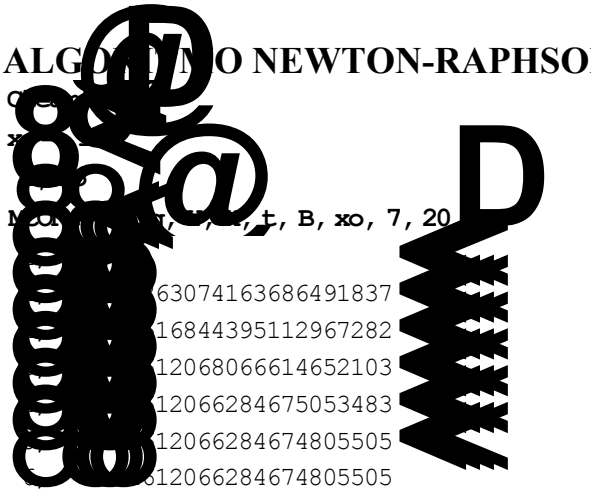
MCO NO LINEALES



ALGORITMO GAUSS-NEWTON



ALGORITMO NEWTON-RAPHSON



Por cuestiones de espacio solo se muestran las programaciones y los resultados para un solo punto inicial ya que los resultados son idénticos a los obtenidos por Gauss Light solo que con mas precisión.

MAPLE 6

Se exponen a continuación las programaciones efectuadas en este programa

MÉTODO DE NEWTON

JORGE M. OVIEDO

Agosto 2002

INICIALIZACIONES

```
> restart;
with(linalg):
```

```
Warning, new definition for norm
Warning, new definition for trace
```

OPTIMIZACIÓN NO LINEAL

Resultados Exactos del Algoritmo

Sintaxis:

Procedimiento

```
> Newton := proc(f,x,xo,n)
#f=funcion a optimizar
#x=listado de variables
#xo=listado de valores iniciales de las variables
#n= numero de iteraciones
local i,h,t,Sol,xx,xxo;
xx := [seq(x[i]=xo[i], i=1..nops(x) )];
xxo := xo;
for t from 1 to n do
Sol:=evalm(xxo-eval(multiply(grad(f, x),inverse(hessian(f,x))),xx)):
xxo := Sol:
xx:=[seq(x[h]=Sol[h],h=1..nops(x))]:
print(Sol);
od;
end;
```

```
Newton := proc(f, x, xo, n)
local i, h, t, Sol, xx, xxo;
    xx := [seq(x[i] = xo[i], i = 1 .. nops(x))];
    xxo := xo;
    for t to n do
        Sol := evalm(xxo - eval(multiply(grad(f, x), inverse(hessian(f, x))), xx));
        xxo := Sol;
        xx := [seq(x[h] = Sol[h], h = 1 .. nops(x))];
        print(Sol)
    od
end
```

Ejem plo

Resultados Numericos

MCO NO LINEALES

Algoritmo Newton-Raphson

Sintaxis

MCO_Newton (*f unción, l istado de observaciones, m atriz de datos variable explicativa, l istado de variables explicativa, li stado de parametros a estimar, c ondiciones iniciales de los parámetros, n úmero de iteraciones, d ígitos de precisión* **)**

Procedimiento

```
> MCO_Newton := proc(g,y,d,dd,x,xo,n,m)
#Argumentos para MCO_Newton
#g=funcion que relaciona parametros y observaciones [f(X,b)]
#y=lista de observaciones variable dependiente
#d=matriz de observaciones variable-s dependiente-s
#dd=listado de variable dependientes
#x=Listado de parametros a estimar
#xo=condicioens iniciales para los parametros
local i,h,t,Sol,xx,xxo,f;
f:=sum((y[i]-eval(g,[seq(dd[j]=d[i][j],j=1..nops(dd))]))^2, i=1..nops(y));
xx := [seq(x[i]=xo[i], i=1..nops(x) )];
xxo := xo;
for t from 1 to n do
Sol:=evalf(evalm(xxo-eval(multiply(grad(f, x),inverse(hessian(f,x))),xx)),m):
xxo := Sol:
xx:=[seq(x[h]=Sol[h],h=1..nops(x))]:
print(Sol);
od;
end;
```

```
MCO_Newton := proc(g, y, d, dd, x, xo, n, m)
local i, h, t, Sol, xx, xxo, f,
    f := sum((y[i] - eval(g, [seq(dd[j] = d[i][j], j = 1 .. nops(dd))]))^2, i = 1 .. nops(y));

    xx := [seq(x[i] = xo[i], i = 1 .. nops(x))];
    xxo := xo;
    for t to n do
        Sol := evalf(evalm(xxo - eval(multiply(grad(f, x), inverse(hessian(f, x))), xx)), m);

        xxo := Sol;
        xx := [seq(x[h] = Sol[h], h = 1 .. nops(x))];
        print(Sol)
```

```
    od
end
```

Ejemplo

```
> Da:=[[3284, 0286, 0645],[
3149, 0973, 0585],[
2877, 0384, 0310],[
-0467, 0276, 0058],[
1211, 0973, 0455],[
1389, 0543, 0779],[
1145, 0957, 0259],[
2321, 0948, 0202],[
0998, 0543, 0028],[
0379, 0797, 0099],[
1106, 0936, 0142],[
0428, 0889, 0296],[
0011, 0006, 0175],[
1179, 0828, 0180],[
1858, 0399, 0842],[
0388, 0617, 0039],[
0651, 0939, 0103],[
0593, 0784, 0620],[
0046, 0072, 0158],[
1152, 0889, 0704]]:

> y1:=[seq(Da[i,1],i=1..nops(Da))]:
d1:=[seq([Da[i,2],Da[i,3]],i=1..nops(Da))]:

> y:=1/1000*y1;d:=1/1000*d1:
```

$$y := \left[\frac{821}{250}, \frac{3149}{1000}, \frac{2877}{1000}, \frac{-467}{1000}, \frac{1211}{1000}, \frac{1389}{1000}, \frac{229}{200}, \frac{2321}{1000}, \frac{499}{500}, \frac{379}{1000}, \frac{553}{500}, \frac{107}{250}, \frac{11}{1000}, \frac{1179}{1000}, \frac{929}{500}, \frac{97}{250}, \frac{651}{1000}, \frac{593}{1000}, \frac{23}{500}, \frac{144}{125}\right]$$

```
> g:=b*x1+b^2*x2:
dd:=[x1,x2]:

> MCO_Newton(b*x1+b^2*x2,y,d,[x1,x2],[b],[1.],10,30);
```

$$[1.18630741636864918373321324885]$$

$$[1.16168443951129672822759606544]$$

$$[1.16120680666146521029316253250]$$

$$[1.16120662846750534831590017163]$$

$$[1.161206628467480550534318276 19]$$

$$[1.161206628467480550534318275 71]$$

$$[1.161206628467480550534318275 71]$$

$$[1.161206628467480550534318275 71]$$

$$[1.161206628467480550534318275 71]$$

$$[1.161206628467480550534318275 71]$$

>

Algoritmo Gauss-Newton

Sintaxis

Procedimiento

```
> GaussNewton := proc(g,y,d,dd,x,xo,n,m)
#Argumentos para GaussNewton
#g=funcion que relaciona parametros y observaciones [f(X,b)]
#y=lista de observaciones variable dependiente
#d=matriz de observaciones variable-s dependiente-s
#dd=listado de variable dependientes
#x=Listado de parametros a estimar
#xo=condicioens iniciales para los parametros
local i,h,t,Sol,xx,xxo,f,Z;
f:=[seq(eval(g,[seq(dd[j]=d[i][j],j=1..nops(dd))]), i=1..nops(y))];
xx := [seq(x[i]=xo[i], i=1..nops(x) )];
xxo := xo;
for t from 1 to n do
Z:=eval(jacobian(f,x),xx) :
Sol:= evalf (evalm( xxo + inverse(transpose(Z)&*Z)&*transpose(Z)&*(y-
eval(f,xx))),m): xxo := Sol:
xx:=[seq(x[h]=Sol[h],h=1..nops(x))]:
print(Sol);
od;
end;
```

```
GaussNewton := proc(g, y, d, dd, x, xo, n, m)
local i, h, t, Sol, xx, xxo, f, Z;
    f := [seq(eval(g, [seq(dd[j] = d[i][j], j = 1 .. nops(dd))]), i = 1 .. nops(y))];
    xx := [seq(x[i] = xo[i], i = 1 .. nops(x))];
```

```
    xxo := xo;
    for t to n do
        Z := eval(jacobian(f, x), xx);
        Sol := evalf(
            evalm(xxo + ((inverse(transpose(Z) `&*` Z) `&*` transpose(Z)) `&*` (y - eval(f, xx)))), m)

        xxo := Sol;
        xx := [seq(x[h] = Sol[h], h = 1 .. nops(x))];
        print(Sol)
    od
end
```

Ejemplo

```
> y:=[2,4];d:=[[1,3],[5,7]];dd:=[P,M];
```

$$y := [2, 4]$$

$$d := [[1, 3], [5, 7]]$$

$$dd := [P, M]$$

```
> GaussNewton(b*x1+b^2*x2,y,d,[x1,x2],[b],[1.],10,30);
```

$$[1.163310557265961158630554723 20]$$

$$[1.161262639634361920481314594 90]$$

$$[1.161208093591872289493067850 92]$$

$$[1.161206666773652714482170212 01]$$

$$[1.161206629468995955270113758 96]$$

$$[1.161206628493665176258740380 66]$$

$$[1.161206628468165147711036558 02]$$

$$[1.161206628467498449331085485 27]$$

$$[1.161206628467481018498453400 75]$$

$$[1.161206628467480562769243036 94]$$

MATLAB 5.3

A raíz de que este software no cuenta con comandos predeterminados para calcular gradientes ni hessianos se debió proceder a crearlos usando derivadas numéricas.

En este software solo se expondrán las programaciones y no las salidas a los efectos de no ser redundante:

GAUSS NEWTON

```
digits(30);
f=inline('b*x1+b.^2*x2','x1','x2','b')
h=0.01
bo=5;
y=[1,9];
x1=[1,3];
x2=[4,9];
  for j=1:15
      for i = 1:2
        F(i)=f(x1(i),x2(i),bo);
        z1(i)=f(x1(i),x2(i),bo+h);
        z2(i)=f(x1(i),x2(i),bo-h);
      end
    Z=(z1-z2)*(1/(2*h));
    bn=bo+inv(Z*Z')*Z*(y-F)';
    bo=bn;
    vpa([j,bo])
  end
```

NEWTON-RAPSHON

```
digits(30);
z=inline('b*x1+b.^2*x2','x1','x2','b')
h=0.01
bo=5;
y=[1,9];              %CARGAR DATOS%
x1=[1,3];
x2=[4,9];
  for j=1:15
      for i = 1:2
        s1(i)=(y(i)-z(x1(i),x2(i),bo)).^2;
        s2(i)=(y(i)-z(x1(i),x2(i),bo+h)).^2;
        s3(i)=(y(i)-z(x1(i),x2(i),bo-h)).^2;
      end
    so=sum(s1);
    somash=sum(s2);
    somenosh=sum(s3);
```

```
    der=(somash-somenosh)/(2*h);
    hes=(somash-2*so+somenosh)/(h.^2);
    bn=bo-(der/hes);
    bo=bn;
    vpa([j,bo])
  end
```

EXCEL 2002

Por medio de una Macro se creó un programa capaz de estimar n parámetros de un modelo no lineal con k variables independientes y t observaciones. Los comandos de Visual Basic se muestran a continuación. Para mayor detalle explorar el archivo .xls que se adjunta en este trabajo.

```
Sub Volver()
'
'
' Macro escrita el 20/08/2002 por JORGE MAURICIO OVIEDO
'

'

  Cells(1, 4).FormulaR1C1 = "b1"
  Cells(1, 5).FormulaR1C1 = "b2"
  Cells(1, 6).FormulaR1C1 = "b3"
  Cells(1, 7).FormulaR1C1 = "b4"
  Cells(1, 8).FormulaR1C1 = "b5"
  Cells(1, 9).FormulaR1C1 = "b6"
  Cells(1, 10).FormulaR1C1 = "b7"
  Cells(1, 11).FormulaR1C1 = "b8"
  Cells(1, 12).FormulaR1C1 = "b9"
  Cells(1, 13).FormulaR1C1 = "b10"
  Cells(1, 14).FormulaR1C1 = "b11"
  Cells(1, 15).FormulaR1C1 = "b12"
  Cells(1, 16).FormulaR1C1 = "b13"
  Cells(1, 17).FormulaR1C1 = "b14"
  Cells(1, 18).FormulaR1C1 = "b15"
  Cells(1, 19).FormulaR1C1 = "b16"



  Cells(8, 5).FormulaR1C1 = "x1"
  Cells(8, 6).FormulaR1C1 = "x2"
  Cells(8, 7).FormulaR1C1 = "x3"
  Cells(8, 8).FormulaR1C1 = "x4"
  Cells(8, 9).FormulaR1C1 = "x5"
  Cells(8, 10).FormulaR1C1 = "x6"
  Cells(8, 11).FormulaR1C1 = "x7"
  Cells(8, 12).FormulaR1C1 = "x8"
  Cells(8, 13).FormulaR1C1 = "x9"
  Cells(8, 14).FormulaR1C1 = "x10"
  Cells(8, 15).FormulaR1C1 = "x11"
```

```
  Cells(8, 16).FormulaR1C1 = "x12"
  Cells(8, 17).FormulaR1C1 = "x13"
  Cells(8, 18).FormulaR1C1 = "x14"
  Cells(8, 19).FormulaR1C1 = "x15"
  Cells(8, 20).FormulaR1C1 = "x16"
  Cells(8, 21).FormulaR1C1 = "x17"
  Cells(8, 22).FormulaR1C1 = "x18"
  Cells(8, 23).FormulaR1C1 = "x19"
  Cells(8, 24).FormulaR1C1 = "x20"
  Cells(8, 25).FormulaR1C1 = "x21"
  Cells(8, 26).FormulaR1C1 = "x22"
  Cells(8, 27).FormulaR1C1 = "x23"
  Cells(8, 28).FormulaR1C1 = "x24"
  Cells(8, 29).FormulaR1C1 = "x25"
  Cells(8, 30).FormulaR1C1 = "x26"
  Cells(8, 31).FormulaR1C1 = "x27"
  Cells(8, 32).FormulaR1C1 = "x28"
  Cells(8, 33).FormulaR1C1 = "x29"
  Cells(8, 34).FormulaR1C1 = "x30"
  Cells(8, 35).FormulaR1C1 = "x31"
  Cells(8, 36).FormulaR1C1 = "x32"
  Cells(8, 37).FormulaR1C1 = "x33"
  Cells(8, 38).FormulaR1C1 = "x34"
  Cells(8, 39).FormulaR1C1 = "x35"
  Cells(8, 40).FormulaR1C1 = "x36"
  Cells(8, 41).FormulaR1C1 = "x37"
  Cells(8, 42).FormulaR1C1 = "x38"
  Cells(8, 43).FormulaR1C1 = "x39"
  Cells(8, 44).FormulaR1C1 = "x40"
  Cells(8, 45).FormulaR1C1 = "x41"
  Cells(8, 46).FormulaR1C1 = "x42"
  Cells(8, 47).FormulaR1C1 = "x43"
  Cells(8, 48).FormulaR1C1 = "x44"
  Cells(8, 49).FormulaR1C1 = "x45"
  Cells(8, 50).FormulaR1C1 = "x46"
  Cells(8, 51).FormulaR1C1 = "x47"
  Cells(8, 52).FormulaR1C1 = "x48"
  Cells(8, 53).FormulaR1C1 = "x49"
  Cells(8, 54).FormulaR1C1 = "x50"
  Cells(8, 55).FormulaR1C1 = "x51"
  Cells(8, 56).FormulaR1C1 = "x52"
  Cells(8, 57).FormulaR1C1 = "x53"
  Cells(8, 58).FormulaR1C1 = "x54"
  Cells(8, 59).FormulaR1C1 = "x55"
  Cells(8, 60).FormulaR1C1 = "x56"
  Cells(8, 61).FormulaR1C1 = "x57"
  Cells(8, 62).FormulaR1C1 = "x58"
  Cells(8, 63).FormulaR1C1 = "x59"
  Cells(8, 64).FormulaR1C1 = "x60"
  Cells(8, 65).FormulaR1C1 = "x61"
  Cells(8, 66).FormulaR1C1 = "x62"
  Cells(8, 67).FormulaR1C1 = "x63"
  Cells(8, 68).FormulaR1C1 = "x64"
  Cells(8, 69).FormulaR1C1 = "x65"
  Cells(8, 70).FormulaR1C1 = "x66"
  Cells(8, 71).FormulaR1C1 = "x67"
  Cells(8, 72).FormulaR1C1 = "x68"
  Cells(8, 73).FormulaR1C1 = "x69"
  Cells(8, 74).FormulaR1C1 = "x70"
  Cells(8, 75).FormulaR1C1 = "x71"
  Cells(8, 76).FormulaR1C1 = "x72"
```

```
  Cells(8, 77).FormulaR1C1 = "x73"
  Cells(8, 78).FormulaR1C1 = "x74"
  Cells(8, 79).FormulaR1C1 = "x75"
  Cells(8, 80).FormulaR1C1 = "x76"
  Cells(8, 81).FormulaR1C1 = "x77"
  Cells(8, 82).FormulaR1C1 = "x78"
  Cells(8, 83).FormulaR1C1 = "x79"
  Cells(8, 84).FormulaR1C1 = "x80"
  Cells(8, 85).FormulaR1C1 = "x81"
  Cells(8, 86).FormulaR1C1 = "x82"
  Cells(8, 87).FormulaR1C1 = "x83"
  Cells(8, 88).FormulaR1C1 = "x84"
  Cells(8, 89).FormulaR1C1 = "x85"
  Cells(8, 90).FormulaR1C1 = "x86"
  Cells(8, 91).FormulaR1C1 = "x87"
  Cells(8, 92).FormulaR1C1 = "x88"
  Cells(8, 93).FormulaR1C1 = "x89"
  Cells(8, 94).FormulaR1C1 = "x90"
  Cells(8, 95).FormulaR1C1 = "x91"
  Cells(8, 96).FormulaR1C1 = "x92"
  Cells(8, 97).FormulaR1C1 = "x93"
  Cells(8, 98).FormulaR1C1 = "x94"
  Cells(8, 99).FormulaR1C1 = "x95"
  Cells(8, 100).FormulaR1C1 = "x96"
  Cells(8, 101).FormulaR1C1 = "x97"
  Cells(8, 102).FormulaR1C1 = "x98"
  Cells(8, 103).FormulaR1C1 = "x99"
  Cells(8, 104).FormulaR1C1 = "x100"


 Sheets("Hoja1").Select

End Sub
Sub defx()
'
'
' Macro escrita el 20/08/2002 por JORGE MAURICIO OVIEDO
'

'
   ActiveWorkbook.Names.Add Name:="x_1", RefersToR1C1:="=Hoja2!RC5"
   ActiveWorkbook.Names.Add Name:="x_2", RefersToR1C1:="=Hoja2!RC6"
   ActiveWorkbook.Names.Add Name:="x_3", RefersToR1C1:="=Hoja2!RC7"
   ActiveWorkbook.Names.Add Name:="x_4", RefersToR1C1:="=Hoja2!RC8"
   ActiveWorkbook.Names.Add Name:="x_5", RefersToR1C1:="=Hoja2!RC9"
   ActiveWorkbook.Names.Add Name:="x_6", RefersToR1C1:="=Hoja2!RC10"
   ActiveWorkbook.Names.Add Name:="x_7", RefersToR1C1:="=Hoja2!RC11"
   ActiveWorkbook.Names.Add Name:="x_8", RefersToR1C1:="=Hoja2!RC12"
   ActiveWorkbook.Names.Add Name:="x_9", RefersToR1C1:="=Hoja2!RC13"
   ActiveWorkbook.Names.Add Name:="x_10", RefersToR1C1:="=Hoja2!R14"
   ActiveWorkbook.Names.Add Name:="x_11", RefersToR1C1:="=Hoja2!RC15"
   ActiveWorkbook.Names.Add Name:="x_12", RefersToR1C1:="=Hoja2!RC16"
   ActiveWorkbook.Names.Add Name:="x_13", RefersToR1C1:="=Hoja2!RC17"
   ActiveWorkbook.Names.Add Name:="x_14", RefersToR1C1:="=Hoja2!RC18"
   ActiveWorkbook.Names.Add Name:="x_15", RefersToR1C1:="=Hoja2!RC19"
   ActiveWorkbook.Names.Add Name:="x_16", RefersToR1C1:="=Hoja2!RC20"
   ActiveWorkbook.Names.Add Name:="x_17", RefersToR1C1:="=Hoja2!RC21"
   ActiveWorkbook.Names.Add Name:="x_18", RefersToR1C1:="=Hoja2!RC22"
   ActiveWorkbook.Names.Add Name:="x_19", RefersToR1C1:="=Hoja2!RC23"
   ActiveWorkbook.Names.Add Name:="x_20", RefersToR1C1:="=Hoja2!RC24"
   ActiveWorkbook.Names.Add Name:="x_21", RefersToR1C1:="=Hoja2!RC25"
```

```
ActiveWorkbook.Names.Add Name:="x_22", RefersToR1C1:="=Hoja2!RC26"
ActiveWorkbook.Names.Add Name:="x_23", RefersToR1C1:="=Hoja2!RC27"
ActiveWorkbook.Names.Add Name:="x_24", RefersToR1C1:="=Hoja2!RC28"
ActiveWorkbook.Names.Add Name:="x_25", RefersToR1C1:="=Hoja2!RC29"
ActiveWorkbook.Names.Add Name:="x_26", RefersToR1C1:="=Hoja2!RC30"
ActiveWorkbook.Names.Add Name:="x_27", RefersToR1C1:="=Hoja2!RC31"
ActiveWorkbook.Names.Add Name:="x_28", RefersToR1C1:="=Hoja2!RC32"
ActiveWorkbook.Names.Add Name:="x_29", RefersToR1C1:="=Hoja2!RC33"
ActiveWorkbook.Names.Add Name:="x_30", RefersToR1C1:="=Hoja2!RC34"
ActiveWorkbook.Names.Add Name:="x_31", RefersToR1C1:="=Hoja2!RC35"
ActiveWorkbook.Names.Add Name:="x_32", RefersToR1C1:="=Hoja2!RC36"
ActiveWorkbook.Names.Add Name:="x_33", RefersToR1C1:="=Hoja2!RC37"
ActiveWorkbook.Names.Add Name:="x_34", RefersToR1C1:="=Hoja2!RC38"
ActiveWorkbook.Names.Add Name:="x_35", RefersToR1C1:="=Hoja2!RC39"
ActiveWorkbook.Names.Add Name:="x_36", RefersToR1C1:="=Hoja2!RC40"
ActiveWorkbook.Names.Add Name:="x_37", RefersToR1C1:="=Hoja2!RC41"
ActiveWorkbook.Names.Add Name:="x_38", RefersToR1C1:="=Hoja2!RC42"
ActiveWorkbook.Names.Add Name:="x_39", RefersToR1C1:="=Hoja2!RC43"
ActiveWorkbook.Names.Add Name:="x_40", RefersToR1C1:="=Hoja2!RC44"
ActiveWorkbook.Names.Add Name:="x_41", RefersToR1C1:="=Hoja2!RC45"
ActiveWorkbook.Names.Add Name:="x_42", RefersToR1C1:="=Hoja2!RC46"
ActiveWorkbook.Names.Add Name:="x_43", RefersToR1C1:="=Hoja2!RC47"
ActiveWorkbook.Names.Add Name:="x_44", RefersToR1C1:="=Hoja2!RC48"
ActiveWorkbook.Names.Add Name:="x_45", RefersToR1C1:="=Hoja2!RC49"
ActiveWorkbook.Names.Add Name:="x_46", RefersToR1C1:="=Hoja2!RC50"
ActiveWorkbook.Names.Add Name:="x_47", RefersToR1C1:="=Hoja2!RC51"
ActiveWorkbook.Names.Add Name:="x_48", RefersToR1C1:="=Hoja2!RC52"
ActiveWorkbook.Names.Add Name:="x_49", RefersToR1C1:="=Hoja2!RC53"
ActiveWorkbook.Names.Add Name:="x_50", RefersToR1C1:="=Hoja2!RC54"
ActiveWorkbook.Names.Add Name:="x_51", RefersToR1C1:="=Hoja2!RC55"
ActiveWorkbook.Names.Add Name:="x_52", RefersToR1C1:="=Hoja2!RC56"
ActiveWorkbook.Names.Add Name:="x_53", RefersToR1C1:="=Hoja2!RC57"
ActiveWorkbook.Names.Add Name:="x_54", RefersToR1C1:="=Hoja2!RC58"
ActiveWorkbook.Names.Add Name:="x_55", RefersToR1C1:="=Hoja2!RC59"
ActiveWorkbook.Names.Add Name:="x_56", RefersToR1C1:="=Hoja2!RC60"
ActiveWorkbook.Names.Add Name:="x_57", RefersToR1C1:="=Hoja2!RC61"
ActiveWorkbook.Names.Add Name:="x_58", RefersToR1C1:="=Hoja2!RC62"
ActiveWorkbook.Names.Add Name:="x_59", RefersToR1C1:="=Hoja2!RC63"
ActiveWorkbook.Names.Add Name:="x_60", RefersToR1C1:="=Hoja2!RC64"
ActiveWorkbook.Names.Add Name:="x_61", RefersToR1C1:="=Hoja2!RC65"
ActiveWorkbook.Names.Add Name:="x_62", RefersToR1C1:="=Hoja2!RC66"
ActiveWorkbook.Names.Add Name:="x_63", RefersToR1C1:="=Hoja2!RC67"
ActiveWorkbook.Names.Add Name:="x_64", RefersToR1C1:="=Hoja2!RC68"
ActiveWorkbook.Names.Add Name:="x_65", RefersToR1C1:="=Hoja2!RC69"
ActiveWorkbook.Names.Add Name:="x_66", RefersToR1C1:="=Hoja2!RC70"
ActiveWorkbook.Names.Add Name:="x_67", RefersToR1C1:="=Hoja2!RC71"
ActiveWorkbook.Names.Add Name:="x_68", RefersToR1C1:="=Hoja2!RC72"
ActiveWorkbook.Names.Add Name:="x_69", RefersToR1C1:="=Hoja2!RC73"
ActiveWorkbook.Names.Add Name:="x_70", RefersToR1C1:="=Hoja2!RC74"
ActiveWorkbook.Names.Add Name:="x_71", RefersToR1C1:="=Hoja2!RC75"
ActiveWorkbook.Names.Add Name:="x_72", RefersToR1C1:="=Hoja2!RC76"
ActiveWorkbook.Names.Add Name:="x_73", RefersToR1C1:="=Hoja2!RC77"
ActiveWorkbook.Names.Add Name:="x_74", RefersToR1C1:="=Hoja2!RC78"
ActiveWorkbook.Names.Add Name:="x_75", RefersToR1C1:="=Hoja2!RC79"
ActiveWorkbook.Names.Add Name:="x_76", RefersToR1C1:="=Hoja2!RC80"
ActiveWorkbook.Names.Add Name:="x_77", RefersToR1C1:="=Hoja2!RC81"
ActiveWorkbook.Names.Add Name:="x_78", RefersToR1C1:="=Hoja2!RC82"
ActiveWorkbook.Names.Add Name:="x_79", RefersToR1C1:="=Hoja2!RC83"
ActiveWorkbook.Names.Add Name:="x_80", RefersToR1C1:="=Hoja2!RC84"
ActiveWorkbook.Names.Add Name:="x_81", RefersToR1C1:="=Hoja2!RC85"
ActiveWorkbook.Names.Add Name:="x_82", RefersToR1C1:="=Hoja2!RC86"
```

```
   ActiveWorkbook.Names.Add Name:="x_83", RefersToR1C1:="=Hoja2!RC87"
   ActiveWorkbook.Names.Add Name:="x_84", RefersToR1C1:="=Hoja2!RC88"
   ActiveWorkbook.Names.Add Name:="x_85", RefersToR1C1:="=Hoja2!RC89"
   ActiveWorkbook.Names.Add Name:="x_86", RefersToR1C1:="=Hoja2!RC90"
   ActiveWorkbook.Names.Add Name:="x_87", RefersToR1C1:="=Hoja2!RC91"
   ActiveWorkbook.Names.Add Name:="x_88", RefersToR1C1:="=Hoja2!RC92"
   ActiveWorkbook.Names.Add Name:="x_89", RefersToR1C1:="=Hoja2!RC93"
   ActiveWorkbook.Names.Add Name:="x_90", RefersToR1C1:="=Hoja2!RC94"
   ActiveWorkbook.Names.Add Name:="x_91", RefersToR1C1:="=Hoja2!RC95"
   ActiveWorkbook.Names.Add Name:="x_92", RefersToR1C1:="=Hoja2!RC96"
   ActiveWorkbook.Names.Add Name:="x_93", RefersToR1C1:="=Hoja2!RC97"
   ActiveWorkbook.Names.Add Name:="x_94", RefersToR1C1:="=Hoja2!RC98"
   ActiveWorkbook.Names.Add Name:="x_95", RefersToR1C1:="=Hoja2!RC99"
   ActiveWorkbook.Names.Add Name:="x_96", RefersToR1C1:="=Hoja2!RC100"
   ActiveWorkbook.Names.Add Name:="x_97", RefersToR1C1:="=Hoja2!RC101"
   ActiveWorkbook.Names.Add Name:="x_98", RefersToR1C1:="=Hoja2!RC102"
   ActiveWorkbook.Names.Add Name:="x_99", RefersToR1C1:="=Hoja2!RC103"
   ActiveWorkbook.Names.Add Name:="x_100",
RefersToR1C1:="=Hoja2!RC104"


End Sub
Sub defb()
'
' Macro5 Macro
' Macro escrita el 20/08/2002 por JORGE MAURICIO OVIEDO
'

'
    ActiveWorkbook.Names.Add Name:="b_1", RefersToR1C1:="=Hoja2!R2C4"
    ActiveWorkbook.Names.Add Name:="b_2", RefersToR1C1:="=Hoja2!R2C5"
    ActiveWorkbook.Names.Add Name:="b_3", RefersToR1C1:="=Hoja2!R2C6"
    ActiveWorkbook.Names.Add Name:="b_4", RefersToR1C1:="=Hoja2!R2C7"
    ActiveWorkbook.Names.Add Name:="b_5", RefersToR1C1:="=Hoja2!R2C8"
    ActiveWorkbook.Names.Add Name:="b_6", RefersToR1C1:="=Hoja2!R2C9"
    ActiveWorkbook.Names.Add Name:="b_7", RefersToR1C1:="=Hoja2!R2C10"
    ActiveWorkbook.Names.Add Name:="b_8", RefersToR1C1:="=Hoja2!R2C11"
    ActiveWorkbook.Names.Add Name:="b_9", RefersToR1C1:="=Hoja2!R2C12"
    ActiveWorkbook.Names.Add Name:="b_10",
RefersToR1C1:="=Hoja2!R2C13"
    ActiveWorkbook.Names.Add Name:="b_11",
RefersToR1C1:="=Hoja2!R2C14"
    ActiveWorkbook.Names.Add Name:="b_12",
RefersToR1C1:="=Hoja2!R2C15"
    ActiveWorkbook.Names.Add Name:="b_13",
RefersToR1C1:="=Hoja2!R2C16"
    ActiveWorkbook.Names.Add Name:="b_14",
RefersToR1C1:="=Hoja2!R2C17"
    ActiveWorkbook.Names.Add Name:="b_15",
RefersToR1C1:="=Hoja2!R2C18"
    ActiveWorkbook.Names.Add Name:="b_16",
RefersToR1C1:="=Hoja2!R2C19"

End Sub
Sub MCO_n_VAR()
'
' MCO_n_VAR Macro
' Macro escrita el 20/08/2002 por JORGE MAURICIO OVIEDO
'

'
```

```
    n = Cells(4, 4).Value
    k = Cells(5, 4).Value
    m = Cells(6, 4).Value

    Cells(10, 3).Select
    Selection.AutoFill Destination:=Range(Cells(10, 3), Cells(10 + m -
1, 3)), Type:=xlFillDefault
    Range(Cells(11, 1), Cells(11, 2)).Select
    Selection.AutoFill Destination:=Range(Cells(11, 1), Cells(11 + m -
2, 2)), Type:=xlFillDefault

    Cells(1, 1).Select

    SolverReset
    SolverOk SetCell:=Cells(10 + m - 1, 1), MaxMinVal:=2,
ValueOf:="0", ByChange:=Range(Cells(2, 4), Cells(2, 4 + n - 1))
    SolverSolve


End Sub

Sub ir()
'
' ir Macro
' Macro escrita el 20/08/2002 por JORGE MAURICIO OVIEDO
'

'
    Sheets("Hoja2").Select
    t = Cells(4, 4).Value
    q = Cells(5, 4).Value
    Range(Cells(1, 4 + t), Cells(2, 21)).Select
  Selection.ClearContents
  Range(Cells(8, 5 + q), Cells(65000, 105)).Select
  Selection.ClearContents
  Cells(1, 3).Select
End Sub
```

EVIEW-S

A continuación vamos a utilizar el programa E-views para estimar los parámetros de un modelo utilizando MCNL. Éste permite aplicar MCNL a cualquier ecuación (un tipo de objeto que reconoce E-views, junto con series, muestras, gráficos, grupos, etc.) que sea no lineal con respecto a sus parámetros, de forma automática simplemente al pedirle al programa que utilice mínimos cuadrados ordinarios..

Los valores iniciales de β que E-views utiliza para comenzar el proceso iterativo están dados por los valores de los parámetros que se encuentran en el vector de coeficientes (otro tipo de objeto que reconoce el programa) en el momento de comenzar la estimación. Por lo tanto, si se utiliza el vector de coeficientes por defecto (C) que E-views crea siempre que un nuevo workfile es generado, los valores iniciales para todos los parámetros es "0".

En nuestro caso, una vez creada la ecuación con las especificaciones del modelo, E-views lleva a cabo una estimación por MCNL, donde el valor inicial del parámetro es "0". El output obtenido es:

Dependent Variable: Y
Method: Least Squares
Date: 08/11/02 Time: 21:53
Sample: 1 20
Included observations: 20
Convergence achieved after 5 iterations
Y=C(1)*X1+(C(1)^2)*X2

	Coefficient	Std. Error	t-Statistic	Prob.
C(1)	1.161211	0.130655	8.887632	0.0000

R-squared	0.217987	Mean dependent var	1.184900
Adjusted R-squared	0.217987	S.D. dependent var	1.047650
S.E. of regression	0.926452	Akaike info criterion	2.733799
Sum squared resid	16.30797	Schwarz criterion	2.783585
Log likelihood	-26.33799	Durbin-Watson stat	1.005021

Como vemos, no hemos llegado al mínimo absoluto de nuestra función, sino a un mínimo local. Al aplicar Gauss-Newton, llegábamos a dos valores mínimos, donde 1,1612066 era un mínimo local, pero no absoluto (dado por –2,2.0294944). Se debe ser cuidadoso en este aspecto al estimar un modelo no lineal, ya que dependiendo del valor de partida será el mínimo (local o absoluto) al que llegaremos.

Utilizando los siguientes comandos, llegamos a los mismos resultados ya obtenidos con GAUSS, al partir de los mismos valores iniciales. El comando param, nos permite establecer cual es el valor inicial elegido para el parámetro β.

```
' MCNL para distintos valores iniciales
open c:\misdoc~1\datos2.wf1
param C(1) 4
equation eqmoda.ls Y=C(1)*X1+(C(1)^2)*X2
param C(1) -0.9
equation eqmodb.ls Y=C(1)*X1+(C(1)^2)*X2
param C(1) -3
equation eqmodc.ls Y=C(1)*X1+(C(1)^2)*X2
param C(1) -1.05
equation eqmodd.ls Y=C(1)*X1+(C(1)^2)*X2
```

Los outputs obtenidos son:

$\beta_0=1$

Dependent Variable: Y
Method: Least Squares
Date: 12/11/02 Time: 23:12
Sample: 1 20
Included observations: 20

Convergence achieved after 6 iterations
Y=C(1)*X1+(C(1)^2)*X2

	Coefficient	Std. Error	t-Statistic	Prob.
C(1)	1.161208	0.130662	8.887105	0.0000

R-squared	0.217987	Mean dependent var	1.184900
Adjusted R-squared	0.217987	S.D. dependent var	1.047650
S.E. of regression	0.926452	Akaike info criterion	2.733799
Sum squared resid	16.30797	Schwarz criterion	2.783585
Log likelihood	-26.33799	Durbin-Watson stat	1.005021

β_0= -2

Dependent Variable: Y
Method: Least Squares
Date: 12/11/02 Time: 23:17
Sample: 1 20
Included observations: 20
Convergence achieved after 8 iterations
Y=C(1)*X1+(C(1)^2)*X2

	Coefficient	Std. Error	t-Statistic	Prob.
C(1)	-2.029462	0.302417	-6.710797	0.0000

R-squared	-1.806364	Mean dependent var	1.184900
Adjusted R-squared	-1.806364	S.D. dependent var	1.047650
S.E. of regression	1.755044	Akaike info criterion	4.011572
Sum squared resid	58.52343	Schwarz criterion	4.061359
Log likelihood	-39.11572	Durbin-Watson stat	1.039450

β_0= 0.1

Dependent Variable: Y
Method: Least Squares
Date: 12/11/02 Time: 23:20
Sample: 1 20
Included observations: 20
Convergence achieved after 8 iterations
Y=C(1)*X1+(C(1)^2*X2)

	Coefficient	Std. Error	t-Statistic	Prob.
C(1)	1.161208	0.130662	8.887105	0.0000

R-squared	0.217987	Mean dependent var	1.184900
Adjusted R-squared	0.217987	S.D. dependent var	1.047650
S.E. of regression	0.926452	Akaike info criterion	2.733799
Sum squared resid	16.30797	Schwarz criterion	2.783585
Log likelihood	-26.33799	Durbin-Watson stat	1.005021

β_0= -0.9

Dependent Variable: Y
Method: Least Squares
Date: 0
Included observations: 20
Convergence achieved after 7 iterations
Y=C(1)*X1+(C(1)^2)*X2

	Coefficient	Std. Error	t-Statistic	Prob.
C(1)	1.161209	0.130660	8.887264	0.0000

R-squared	0.217987	Mean dependent var	1.184900
Adjusted R-squared	0.217987	S.D. dependent var	1.047650
S.E. of regression	0.926452	Akaike info criterion	2.733799
Sum squared resid	16.30797	Schwarz criterion	2.783585
Log likelihood	-26.33799	Durbin-Watson stat	1.005021

Comparando los valores alcanzados, el número de iteraciones es parecido al Gauss-Newton, al igual que los coeficientes del estimador del parámetro y su desviación estándar.

El algoritmo no lineal que utiliza E-views por defecto es el de Marcuardt (con derivadas numéricas), pero cuenta con la opción de utilizar de forma alternativa con el algoritmo de Gauss-Newton.

El algoritmo de Maquardt modifica el algoritmo de Gauss-Newton agregándole una matriz de corrección. El efecto de esta modificación es que la matriz de corrección empuja al estimador del parámetro en la dirección del vector gradiente. La idea es que cuando nos encontramos lejos del máximo, la aproximación lineal de la función adoptada por el algoritmo puede ser una guía pobre.

Con la corrección, este algoritmo mejora la performance del estimador cuando más lejos se encuentre el valor inicial del mínimo.

E-views incluye una variable de longitud de paso (que mejora la función objetivo), a través de una búsqueda por prueba y error. Esta variable no es optimizada en cada iteración (ya que prioriza un correcto valor del vector de dirección).

Como vemos, la utilización del programa E-views permite facilitar en gran medida el trabajo de programación que se requiere para aplicar MCNL y los resultados obtenidos son similares a los que se obtuvieron en las secciones anteriores, pero con un esfuerzo significativamente menos en términos de programación.

PROBLEMAS DEL MÉTODO DE NEWTON Y FRACTALES[3]

Cuando tratamos el algoritmo Newton-Raphson, aclaramos que presentaba problemas de convergencia. Nos podía llevar tanto a un mínimo como a un máximo (dependiendo de si se encontraba muy cerca de un máximo). Sin embargo existen otro tipo de puntos notables de una función a los cuales el algoritmo nos puede conducir.

Consideremos el caso unidimencional. Como vimos en la sección 2.2, el método Newton-Raphson consiste en aproximar la ecuación suma de cuadrados de los errores (la cual será no lineal) a través de un polinomio de Taylor de segundo orden y minimizarla. Para ello se plantea la condición de primer orden y se la iguala a cero. Despejando para β obtenemos el algoritmo.

[3] El contenido de esta sección surge de trabajos conjuntos realizados con Paulo José Regis en agosto-setiembre del presente año en el tema de caos y fractales.

En símbolos:

$$S(\beta) \cong S(\beta_0) + S'(\beta)\big|_{\beta_0}(\beta - \beta_0) + \frac{1}{2}S''(\beta)\big|_{\beta_0}(\beta - \beta_0)^2$$

$$C.P.O.: S'(\beta) = S'(\beta)\big|_{\beta_0} + S''(\beta)\big|_{\beta_0}(\beta - \beta_0) = 0$$

Despejando e iteratuando n veces:

$$\beta_{n+1} = \beta_n - \frac{S'(\beta)\big|_{\beta_0}}{S''(\beta)\big|_{\beta_0}}$$

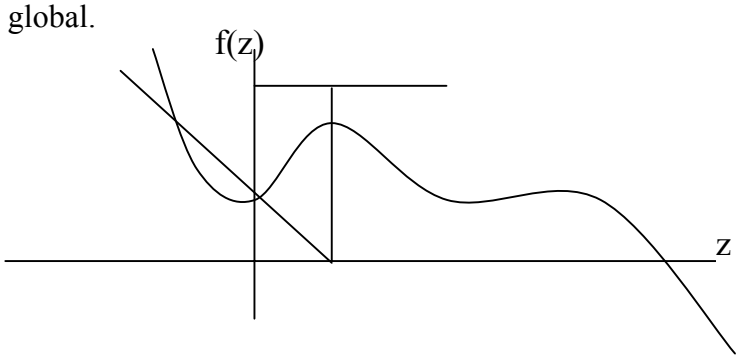
El problema de encontrar el mínimo a una ecuación es equivalente a encontrar las raíces a la condición de primer orden de dicha ecuación. Denominemos a la condición de primer orden como y=f(z). El método de Newton-Raphson para encontrar las raíces de una ecuación consiste en aproximar la ecuación por un polinomio de Taylor de primer orden, a continuación se resuelve para z. En símbolos:

$$f(z) \cong f(z_0) + f'(z)\big|_{z_0}(z - z_0) = 0$$

Resolviendo para z y aplicando el algoritmo n veces:

$$z_{n+1} = z_n + \frac{f(z_n)}{f'(z)\big|_{z_n}}$$

Dado que establecimos que S'(β)=f(z) claramente los dos algoritmos son los mismos. Esta segundo enfoque del problema nos sirve para ganar intuición a la hora de plantear las dificultades que posee Newton-Rahpson en cuanto a sus propiedades de convergencia global.



Lo que hace el algoritmo gráficamente consiste en partir de un valor inicial z_0 que es una apuesta inicial de cual es la raíz, y valuarla en y=f(z). Una vez que me encuentro en un punto de la curva de f(z), aproximo su pendiente por la tangente y extiendo la línea de dicha tangente hasta que cruce el eje de z=0, obteniendo un nuevo valor estimado de la raíz z_1. Nuevamente valúo el nuevo valor de z en y=f(z), busco un nuevo corte de la tangente de la función en dicho punto con el eje de ordenadas. Repetimos el proceso hasta que dos estimaciones sucesivas de la raíz sean iguales.

El problema surge cuando esta función tiene puntos notables como los máximos y mínimos del gráfico anterior y el valor inicial z_0 es tal que se encuentra muy lejos de la raíz. En el gráfico mostramos dicho caso y como el algoritmo no nos lleva a la raíz, sino a uno de dichos puntos notables (en este caso, un máximo).

Al encontrarse en un máximo, la tangente es una línea paralela al eje z=0 y por lo tanto no se cortarán nunca. El algoritmo interpreta esto como si hubiera alcanzado

una raíz (de echo ha alcanzado una raíz ¡¡pero es imaginaria!!, ver siguiente sección), la raíz que estamos buscando, pero solo ha alcanzado un máximo. En el problema de minimizar una función, el algoritmo nos llevaría a un punto de inflexión y no hacia un mínimo. Vemos que este problema no es el que ya habíamos visto. Ya habíamos tratado el caso en que Newton-Raphson nos llevara a un máximo en vez de un mínimo (después de todo una raíz para y=f(z) equivale tanto a una CPO par un máximo como un mínimo), pero éste es distinto.

En la siguiente sección mostramos una función que presenta estos problemas. Para ciertos valores iniciales, el algoritmo nos llevaría a un mínimo, para otros valores iniciales, nos llevaría a un máximo y para otros valores iniciales nos llevaría a otro tipo de punto notable.

2.5.1. NEWTON-RAPHSON Y FRACTALES

Una forma interesante de analizar las propiedades de convergencia global impredecible del algoritmo de Newton-Raphson es investigar para ciertas ecuaciones especiales los valores iniciales para los cuales el método converge y no converge a sus diferentes raíces. Consideremos la ecuación

$$z^3 - 1 = 0$$

La cual posee una sola raíz real z=1, pero que también posee otras dos raíces imaginarias z=(-0,5+0,833i) y z=(-0,5-0,833i). Dichas raíces imaginarias también son puntos notables, por lo que pueden ser candidatos a raíces. Las raíces corresponden a máximos.

Cabe aclarar que este problema es equivalente a encontrar el mínimo de la función $S(z)=z^4/4+z$ (para el caso de MCNL, digamos que esta función es la suma de cuadrados de los residuos de un modelo intrínsecamente no lineal).

Aplicando el algoritmo de Newton-Raphson para encontrar las raíces en nuestra ecuación particular, tenemos:

$$z_{t+1} = z_t - \frac{z_t^3 - 1}{3z_t^2} \qquad (2.5.1)$$

Hasta ahora, solo hemos aplicado dicho iterador solo para valores iniciales (z_0) reales, pero ello es demasiado restrictivo, Newton-Raphson se puede aplicar a casos más generales. En realidad, se puede aplicar el algoritmo tanto para valores iniciales reales como imaginarios y puede converger, como demostraremos a continuación, tanto a valores reales como imaginarios.

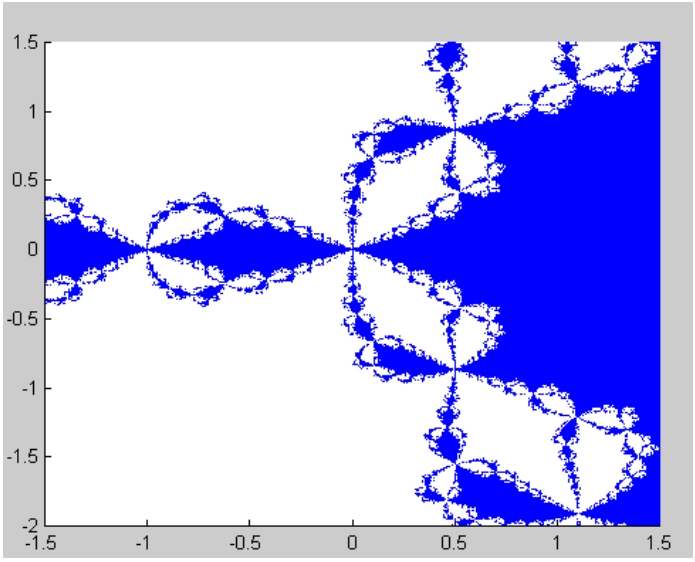
Lo que haremos a continuación es centrar nuestra atención en el plano imaginario. Utilizaremos 2.5.1 para valores iniciales que pertenezcan al plano complejo y veremos que ocurre. Sin duda que el algoritmo nos llevará a la convergencia de alguna de las tres raíces.

Lo esperable sería que hubiera zonas de influencia de cada raíz bien delimitadas, con fronteras entre dichas zonas suaves.

El siguiente programa en Matlab 5.3 examina los valores iniciales en el plano imaginario con componentes reales en el rango [-3/2,3/2] e imaginarios en el rango [-2,3/2]. Nos grafica los puntos de dicho plano que corresponden a valores iniciales de z (z_0)que nos llevan, a través de un proceso iterativo, a la no convergencia de z_{t+1} hacia z=1 (o dicho de otra forma, que convergen a las otras dos raíces).

```
clear all
figure
axis([-1.5  1.5  -2  1.5])
n=100
hold on
s=0;
for j=1:n
   t = -1.5 + (3/n)*j;
   for k = 1:n
   z = t + (-2+(3.5/n)*k)*i;
   y=z;
   for h=1:45
      w=(y.^3-1)/(2*y.^2);
      x=y-w;
      y=x;
   end
   if abs(imag(y))<0.0000001
         plot(real(z),imag(z));
     end
   end
end
```

El gráfico obtenido por el programa es:

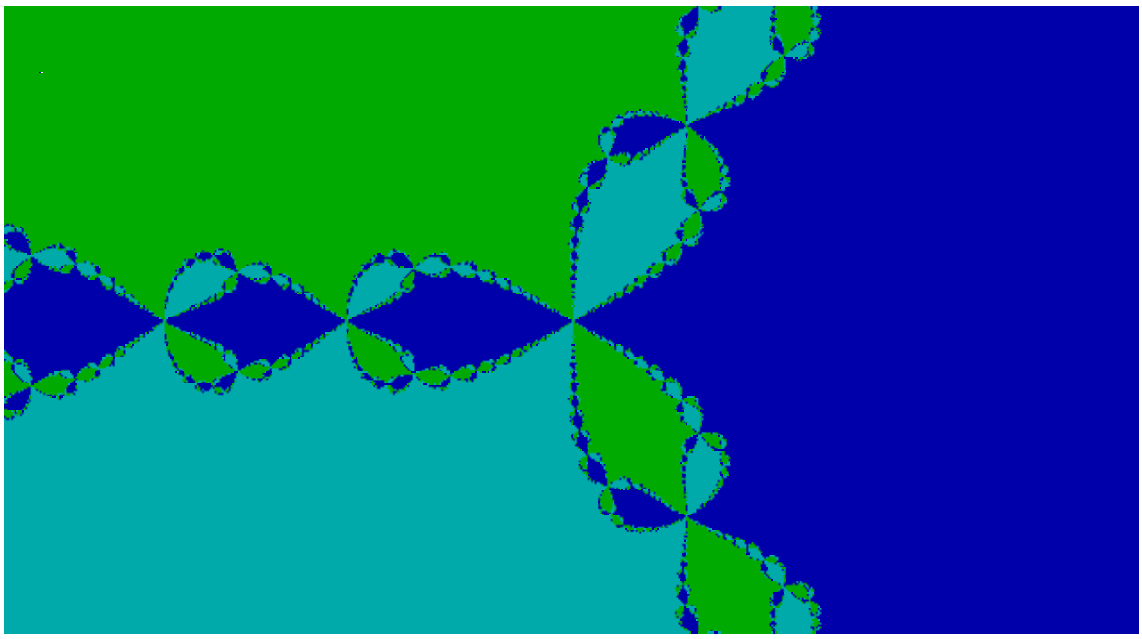


Las fronteras entre las zonas de convergencia a la raíz real y de convergencia a las raíces imaginarias es muy interesante, pero no suave, sino muy irregular. Es, de echo, un fractal

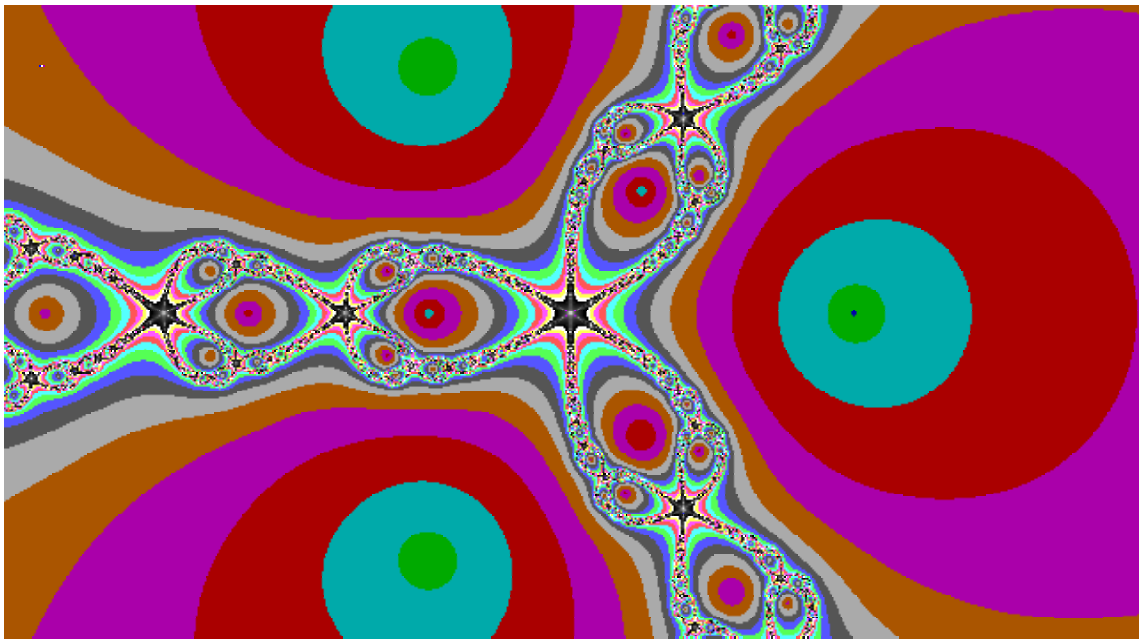
Si nos restringimos a continuación al campo de los reales nuevamente, en este gráfico puede apreciarse los problemas que surgen en cuanto a la existencia de puntos notables sobre la recta de los reales, donde el algoritmo no nos lleva a la raíz real. Ello quiere decir que cumplen con la condición TANTO PARA MÍNIMO COMO PARA MÁXIMO (es decir que no es el caso presentado en la sección 2.4.2 donde para el valor inicial (-0,9) el algoritmo nos llevara a un máximo) y aún así el valor obtenido por el algoritmo NO ES NI UN MÁXIMO NI UN MÍNIMO.

El algoritmo de Newton-Raphson es un sistema dinámico discreto, en el conjunto de los reales, y para la mayoría de los valores iniciales, la secuencia z,, ...z convergerá a la raíz de la ecuación. Pero como ya dijimos, para ciertos valores iniciales no se producirá dicha convergencia hacia una raíz. La estructura del conjunto de valores iniciales para los cuales el método falla es muy interesante y lleva hacia el CAOS.

Por último, modificando en algunos aspectos el programa de Matlab, sería posible construir un nuevo gráfico para que nos muestre en diferentes colores las regiones de atracción de cada una de las 3 raíces. El origen de coordenadas está situado en el centro de la imagen.



Dentro de cada una de las regiones de atracción podemos representar los puntos con diferentes colores según el número de iteraciones necesarias para alcanzar un determinado entorno de las raíces



VEMOS AQUÍ UN FRACTAL EN TODO SU ESPLENDOR. Las singularidades del algoritmo se observan con detenimiento.

BIBLIOGRAFÍA:

- **APÓSTOL, Tom:** *Calculus.* Ed. Reverté. 1976. Segunda Edición
- **CHIANG, Alpha:** *Economía Matemática.* Ed. McGraw-Hill. 1998
- **GUJARATI, Damodar :** *Econometría Básica.* Ed. McGraw-Hill. 1991
- **JOHNSTON, J. :** *Métodos Econométricos.* Ed. Vivens Vives. 1987. Tercera Edición.
- **LUEMBERGER, M.:** *Programcicion lineal y no lineal*
- **MADDALA, G. S. :** *Econometría.* Ed. McGraw-Hill. 1985.
- **STEWARD, James:** *Cálculo.* Ed. Iberoamerica.1981
- **STEIN, S.:** *Cálculo con geometría analítica. Ed Mc. Graw-Hill. 1995*