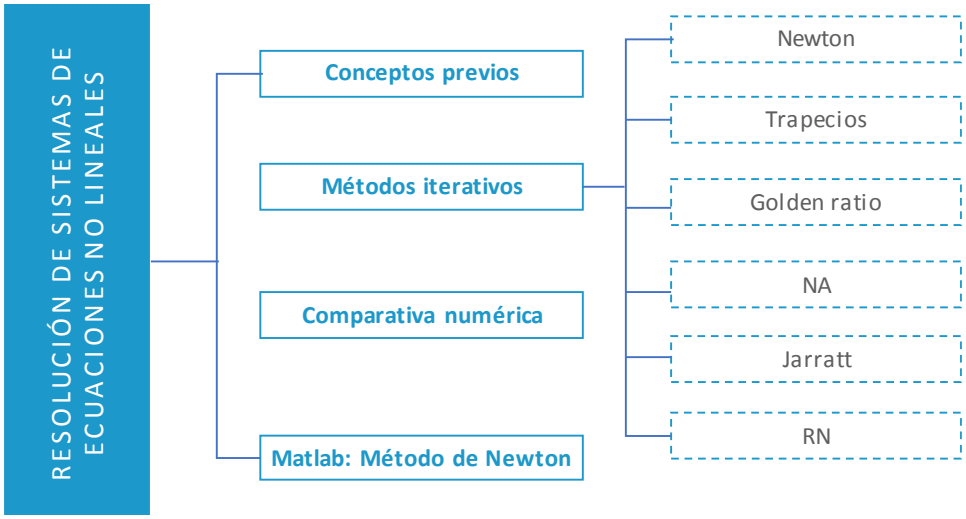
Métodos Numéricos Avanzados en Ingeniería

Sistemas de ecuaciones no lineales

Índice

Esquema



Ideas clave

12.1. ¿Cómo estudiar este tema?

Para estudiar este tema lee las Ideas Clave que encontrarás a continuación

La resolución de ecuaciones no lineales supone una pequeña aproximación a la resolución de problemas no lineales. Sin embargo, la resolución de sistemas no lineales permite resolver problemas de mayores dimensiones. Existen numerosas aplicaciones en ingeniería que se describen a partir de sistemas de ecuaciones diferenciales acopladas, de ecuaciones en derivadas parciales o de problemas de frontera. En cualquier caso, tanto de forma inmediata como a partir de una discretización o transformación del problema, se puede utilizar el cálculo numérico para obtener las soluciones aproximadas como ocurría en el caso de las ecuaciones no lineales.

Los términos y la mayoría de conceptos se reutilizan, aunque hay algunos que deben ser adaptados a su versión multidimensional, como son los cocientes, las distancias, u otros que iremos viendo a lo largo de este tema.

Los apartados de los que consta este tema son:

- Sistemas de ecuaciones no lineales:
  - Conceptos previos.
  - Sistemas de ecuaciones no lineales.
  - Métodos iterativos para sistemas no lineales.
  - Comparativa numérica.
  - Implementación en Matlab del método de Newton.

12.2. Conceptos previos

El problema que tenemos que resolver consiste en obtener una solución $\alpha \in \mathbb{R}^n$ del sistema de ecuaciones $F(X) = 0$, donde $F: D \subseteq \mathbb{R}^n \to \mathbb{R}^n$. En ocasiones el vector X también se denota por x. En ambos casos, nos estaremos refiriendo a un vector de n componentes. En este sentido, podemos expresar el sistema como:

$$F(X) = 0 \leftrightarrow \begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{cases}$$

donde $f_i: \mathbb{R}^n \to \mathbb{R}$ son las funciones coordenadas.

Denominamos matriz Jacobiana de F, denotada por $J_F(X)$, a la matriz:

$$J_F(X) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Ejemplo 1. Sea el sistema de ecuaciones:

$$\begin{cases} xy - x^2 = z \\ x + y - z = 8 \\ y^2 + z^2 = 1 \end{cases}$$

Obtén un sistema de la forma $F(X) = 0$ y calcula su matriz jacobiana $J_F(X)$.

El sistema de ecuaciones quedaría como:

$$F(X) = 0 \leftrightarrow \begin{cases} f_1(x, y, z) = xy - x^2 - z = 0 \\ f_2(x, y, z) = x + y - z - 8 = 0 \\ f_3(x, y, z) = y^2 + z^2 - 1 = 0 \end{cases}$$

La matriz Jacobiana es:

$$J_F(X) = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} \end{bmatrix} = \begin{bmatrix} y - 2x & x & -1 \\ 1 & 1 & -1 \\ 0 & 2y & 2z \end{bmatrix}$$

12.3. Sistemas de ecuaciones lineales

Extensión a sistemas de ecuaciones

Lo primero que debemos tener en cuenta es que vamos a trabajar con **vectores** en lugar de con escalares. Así, el término **«distancia»** deja de ser obtenido a partir del valor absoluto para ser **calculado a partir de la norma.**

Relacionado con los métodos iterativos para la resolución de sistemas de ecuaciones no lineales, volvemos a tener una serie de parámetros que caracterizan a los métodos.

Dada la secuencia $\{x^{(k)}\}_{k\geq 0}$ generada por un método iterativo, se dice que tiene **orden de convergencia p** si:

$$\exists M > 0 \text{ y } \exists k_0 \text{ tal que } \|x^{(k+1)} - x^{(k)}\| \leq M\|x^{(k)} - \alpha\|^p, \forall k \geq k_0$$

Hay diferentes alternativas para la demostración del orden de convergencia de un método iterativo para resolver sistemas de ecuaciones no lineales. Por un lado, podemos utilizar derivadas parciales en la solución, a raíz del Teorema 1 (Traub, 1982).

Teorema 1. Sea $G(X)$ una función de punto fijo con derivadas parciales continuas hasta orden p. El esquema iterativo $x^{(k+1)} = G(x^{(k)})$ tiene orden de convergencia p si:

- $G(\alpha) = \alpha$
- $\frac{\partial^k g_i(\alpha)}{\partial x_{j_1}\partial x_{j_2}\cdots\partial x_{j_k}} = 0, 1 \leq k \leq p-1, 1 \leq i, j_1, j_2, \ldots, j_k \leq n$
- $\frac{\partial^p g_i(\alpha)}{\partial x_{j_1}\partial x_{j_2}\cdots\partial x_{j_k}} \neq 0$, para algún $i, j_1, j_2, \ldots, j_p$

Por otro lado, podemos utilizar los desarrollos de Taylor (Cordero, Hueso, Martínez, & Torregrosa, A modified Newton-Jarratt's composition, 2010).

De forma análoga a lo que ocurría para ecuaciones, tenemos el orden de convergencia computacional aproximado ACOC (Cordero y Torregrosa, 2007) cuya expresión se redefine como:

$$ACOC = \frac{\ln(\|x_{k+1} - x_k\|/\|x_k - x_{k-1}\|)}{\ln(\|x_k - x_{k-1}\|/\|x_{k-1} - x_{k-2}\|)}$$

Con la misma definición que en el caso de sistemas no lineales, aparecen el **índice de eficiencia** I (Ostrowski, 1960), cuya expresión es:

$$I = p^{1/d}$$

donde d es el número de evaluaciones de la función f en cada iteración, y el **índice de eficiencia computacional** IC como:

$$IC = p^{1/(d+op)}$$

donde op representa el número de productos y cocientes por iteración.

Conjetura 1. Dado un método iterativo para resolver sistemas de ecuaciones no lineales, que requiere $d = k_1 + k_2$ evaluaciones funcionales por iteración tal que k_1 corresponden a evaluaciones funcionales de la matriz Jacobiana y k_2 a evaluaciones de la función F. Conjeturamos que el orden de este método será, como máximo, $2^{k_1+k_2-1}$ si $k_1 \leq k_2$.

Implementación en Matlab

Para implementar los métodos iterativos de resolución de sistemas de ecuaciones no lineales en Matlab, el esqueleto de los programas seguirá las mismas pautas. No obstante, hay algunas particularidades que debemos tener en cuenta. Por ejemplo, tenemos que redefinir las **condiciones de parada**, y de nuevo pueden ser alguna de las siguientes o una combinación entre ellas.

- Los dos últimos iterados están muy próximos:

$$\|x_k - x_{k-1}\| < \epsilon$$

- Se ha alcanzado un valor muy próximo a la solución:

$$\|F(x_k)\| < \epsilon$$

- Hemos iterado muchas veces y no hemos conseguido ninguna solución:

$$k > M$$

De nuevo, a la hora de implementar el método iterativo en Matlab, vamos a relacionar algunos conceptos previos con el nombre que le daremos en el código. La Tabla 1. Relación entre parámetros y código. establece dicha relación.

Parámetro	k	ϵ	M
Código	iter	tol	maxiter

Tabla 1. Relación entre parámetros y código.

La estructura de un método iterativo para la resolución de sistemas de ecuaciones no lineales se muestra en el pseudocódigo siguiente:

```
1   function [sol,ACOC]=metodoIterativo(x0,F,tol,maxiter)
2   % Inicialización de las variables
3   iter=1;
4   condición_de_parada=1;
5   x0=x0(:);
6   while condición_de_parada
7         x=Aplicar_método_iterativo(x0);
8         actualizar_condición_de_parada;
9         actualizar_valores;
10        iter=iter+1;
11  end
12  % Preparar datos de salida
13  sol=x;
14  ACOC=evaluar_ACOC(x);
```

Destaca, a diferencia del caso de ecuaciones no lineales, la línea 5. En este caso, para asegurar que el vector de entrada se introduce como vector columna, se utiliza dicha

notación. Es necesario mantener esa estructura de vectores para que no haya errores debido a las dimensiones.

Ejemplo 2. Implementa en Matlab un método iterativo cuyo criterio de parada sea:

$$\begin{cases} \|x_k - x_{k-1}\| < 10^{-3} \\ k \geq 40 \end{cases}$$

Basándonos en el **¡Error! No se encuentra el origen de la referencia.**anterior, desde la consola de Matlab, ejecutaríamos:

```
>> [sol,ACOC]=metodoIterativo(x0,F,1e-3,40)
```

Dentro de la implementación del p**¡Error! No se encuentra el origen de la referencia.** modificaríamos la línea:

```
6          while          norm(x(iter)-x(iter-
))>=tol&&iter<maxiter
```

Un elemento relevante es la **introducción del sistema de ecuaciones no lineales F** en Matlab. Para ello, deberemos conocer si el método iterativo que se va a aplicar requiere del valor de la función en un punto y / o de la evaluación de su matriz Jacobiana. Por tanto, generaremos un archivo .m con el nombre de la función, dándole como parámetro de entrada un vector con el punto sobre el que se quiere evaluar, y como parámetros de salida los valores que requiera el método iterativo. La estructura general se recoge en este pseudocódigo:

```
1  function [F,dF]=sistema_F(X)
2  x=X(1); y=X(2); % Incluir tantas variables como tenga el
   sistema
3  F=expresión_sistema_F_en_columnas;
4  dF=expresión_Jacobiana_F;
```

Ejemplo 3. Escribe un archivo .m para el sistema no lineal:

$$\begin{cases} xy = y^2 \\ \cos(x) - \sin(y) = 0 \end{cases}$$

El método iterativo que lo resuelve requiere del valor del sistema y de su matriz Jacobiana.

En primer lugar, escribimos el sistema con el formato $F(X) = 0$:

$$F(X) = 0 \leftrightarrow \begin{bmatrix} f_1(X) \\ f_2(X) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \leftrightarrow \begin{bmatrix} xy - y^2 \\ \cos(x) - \sin(y) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

A continuación, obtenemos su matriz Jacobiana $J_F(X)$:

$$J_F(X) = \begin{bmatrix} \partial f_1/\partial x & \partial f_1/\partial y \\ \partial f_2/\partial x & \partial f_2/\partial y \end{bmatrix} = \begin{bmatrix} y & x - 2y \\ -\sin(x) & -\cos(y) \end{bmatrix}$$

Por último, utilizamos el segundo **¡Error! No se encuentra el origen de la referencia.**para implementar el sistema en Matlab:

```
1  function [F,dF]=ejemplo5_F(X)
2  x=X(1); y=X(2);
3  F=[x.*y-y.^2; cos(x)-sin(y)];
4  dF=[y, x-2*y; -sin(x), -cos(y)];
```

12.4. Métodos iterativos para sistemas no lineales

Adaptación de métodos utilizados en ecuaciones no lineales

En este apartado vamos a comprobar si podemos reutilizar alguno de los métodos del tema anterior. Comenzaremos por el método por excelencia, que es el **método de Newton**. La expresión iterativa para ecuaciones no lineales era:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Teniendo en cuenta que matricialmente no podemos evaluar cocientes, deberemos reemplazarlos por sus matrices inversas. Además, la notación que utilizamos para los escalares x_k la sustituimos por $x^{(k)}$. Por ello, el **método de Newton para sistemas de ecuaciones no lineales** se escribe como:

$$x^{(k+1)} = x^{(k)} - \left[F'\left(x^{(k)}\right)\right]^{-1} F\left(x^{(k)}\right)$$

donde $F'\left(x^{(k)}\right)$ representa la matriz Jacobiana $J_F(X)$.

Ejemplo 4. El método de Trapecios para ecuaciones no lineales tiene la expresión:

$$\begin{aligned} y_k &= x_k - \frac{f(x_k)}{f'(x_k)} \\ x_{k+1} &= x_k - \frac{2f(x_k)}{f'(y_k) + f'(x_k)} \end{aligned}$$

¿Es posible su implementación para resolver sistemas no lineales?

Realizando una extensión como en el caso del método de Newton, podemos expresar el método de trapecios para sistemas no lineales como:

$$\begin{aligned} y^{(k)} &= x^{(k)} - \left[F'\left(x^{(k)}\right)\right]^{-1} F\left(x^{(k)}\right) \\ x^{(k+1)} &= x^{(k)} - 2\left[F'\left(y^{(k)}\right) + F'\left(x^{(k)}\right)\right]^{-1} F\left(x^{(k)}\right) \end{aligned}$$

Diseño de métodos para sistemas no lineales

Existen diferentes técnicas para la generación de métodos iterativos de resolución de sistemas de ecuaciones no lineales. A continuación, se listan una serie de técnicas:

- Fórmulas de cuadratura.
- Composición de métodos iterativos.
- Polinomios de Adomain.
- Pseudocomposición.
- Funciones peso matriciales.
- Diferencias divididas multidimensionales.

En lo sucesivo, vamos a diseñar una serie de métodos numéricos a partir de la técnica de la composición con el método de Newton.

Sea $z^{(k)} = \Phi\left(x^{(k)}, y^{(k)}\right)$ un método iterativo de orden p, en el que $x^{(k)}$ es el iterado actual e $y^{(k)}$ es un paso intermedio de dicha iteración. Entonces, si se hace la composición de dicho método iterativo con el método de Newton, obtendremos un método de orden $2p$, teniendo como expresión iterativa:

$$x^{(k+1)} = z^{(k)} - \left[F'\left(z^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right)$$

El mayor problema que presenta esta composición es la evaluación de una matriz Jacobiana nueva. Es por ello que como alternativa se presenta congelar la derivada (Cordero y Torregrosa, 2010), dando lugar al siguiente método iterativo:

$$x^{(k+1)} = z^{(k)} - \left[F'\left(x^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right)$$

Teorema 2. Sea $F: D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ una función suficientemente diferenciable en un entorno abierto D de la solución α del sistema no lineal $F(x) = 0$. Supongamos que también $F'(x)$ es continua y no singular en α. Entonces la sucesión de iterados obtenida en el esquema iterativo anterior converge a α, con orden de convergencia $p + 1$.

Con la idea de congelar la derivada y la técnica de la composición, se plantea un método de un número de pasos indeterminado, partiendo de un método de dos

pasos (Cordero, Hueso, Martínez y Torregrosa, 2011). Al método de dos pasos se le denomina **método Golden Ratio**, y tiene la expresión:

$$\begin{aligned} y^{(k)} &= x^{(k)} - a\left[F'\left(x^{(k)}\right)\right]^{-1} F\left(x^{(k)}\right) \\ z^{(k)} &= x^{(k)} - b\left[F'\left(x^{(k)}\right)\right]^{-1} F\left(y^{(k)}\right) \end{aligned}$$

donde $a = \frac{-1\pm\sqrt{5}}{2}$ y b= $\frac{3\pm\sqrt{5}}{2}$. Se trata de un método de orden 3. Si realizamos la composición de este método con Newton, obtenemos un método de orden 4, cuya expresión iterativa es:

$$x^{k+1} = z^{(k)} - \left[F\left(x^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right)$$

Nótese que para el método de orden 4, denotado por **método NA**, solo hay una evaluación de la matriz Jacobiana.

Si comparamos en términos de índice de eficiencia computacional a los métodos de Newton, Golden Ratio y NA, obtenemos:

$$\begin{aligned} IC_{Newton} &= 2^{\frac{1}{\frac{n^3}{3}+2n^2+\frac{2n}{3}}} \\ IC_{Golden\ Ratio} &= 3^{\frac{1}{\frac{n^3}{3}+3n^2+\frac{5n}{3}}} \\ IC_{NA} &= 4^{\frac{1}{\frac{n^3}{3}+4n^2+\frac{8n}{3}}} \end{aligned}$$

donde n representa el número de productos / cocientes por iteración. La Figura 1 representa dicho índice:

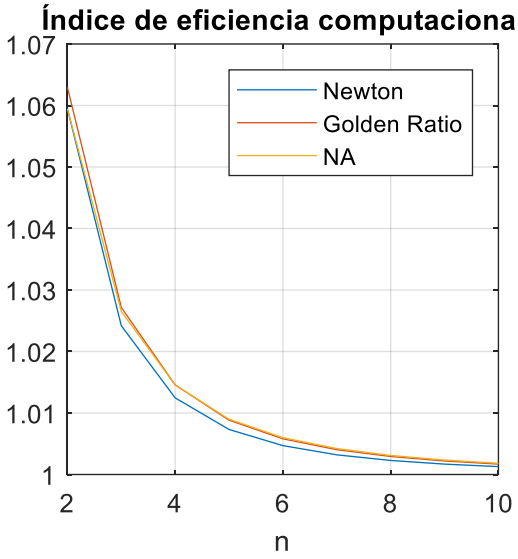


Figura 1. Índice de eficiencia computacional de los métodos de Newton, Golden Ratio y NA.

Un método iterativo no solo tiene que tener un buen valor de orden de convergencia, sino que también es necesario que sea computacionalmente eficiente. Podemos comprobar como esto ocurre con el método NA en la figura 1.

Vamos a mostrar un último método, partiendo del **método de Jarratt** para ecuaciones. Recordemos que su expresión iterativa era:

$$\begin{aligned} y_k &= x_k - \frac{2}{3}\frac{f(x_k)}{f'(x_k)} \\ x_{k+1} &= y_k - \frac{3f'(y_k)+f'(x_k)}{6f'(y_k)-2f'(x_k)}\frac{f(y_k)}{f'(x_k)} \end{aligned}$$

Al realizar la extensión a la resolución de sistemas no lineales, tenemos la expresión:

$$\begin{aligned} y^{(k)} &= x^{(k)} - \frac{2}{3}\left[F'\left(x^{(k)}\right)\right]^{-1}F\left(x^{(k)}\right) \\ z^{(k)} &= y_k - \left[6F'\left(y^{(k)}\right)-2F'\left(x^{(k)}\right)\right]^{-1}\left[3F'\left(y^{(k)}\right)+F'\left(x^{(k)}\right)\right]\left[F'\left(x^{(k)}\right)\right]^{-1}F\left(x^{(k)}\right) \end{aligned}$$

El método de Jarratt, aunque tenga orden 4, implica la evaluación de dos matrices Jacobianas. Al realizar la composición con una variante de Newton, obtenemos el método:

$$x^{(k+1)} = z^{(k)} - \left[aF'\left(x^{(k)}\right) + bF'\left(z^{(k)}\right)\right]^{-1} F\left(z^{(k)}\right)$$

denotado por **método RN**, cuyo orden de convergencia se establece en el Teorema 3.

Teorema 3. Sea $F: D \subseteq \mathbb{R}^n \to \mathbb{R}^n$ una función suficientemente diferenciable en un entorno abierto D de la solución α del sistema no lineal $F(x) = 0$. Supongamos que también $F'(x)$ es continua y no singular en α. Entonces la sucesión de iterados obtenida en el esquema iterativo anterior converge a α, con orden de convergencia 5 en los esquemas que verifican $a + b = 1$. Además, el método con $a = -\frac{1}{2}$ y $b = \frac{3}{2}$ tiene orden de convergencia 6.

12.5. Comparativa numérica

Para poder establecer una comparativa entre diferentes métodos iterativos para resolver sistemas de ecuaciones no lineales, se plantea la comparativa numérica como se ha hecho en temas anteriores. En este caso, vamos a realizar la comparativa de los métodos de Newton (NE), Trapecios (TP), Golden Ratio (GR), NA (NA), Jarratt (JA) y RN (RN) con $a = -\frac{1}{2}$ y $b = \frac{3}{2}$.

Los parámetros que vamos a evaluar son:

- El valor de la solución: $x^{(k)}$
- El número de iteraciones necesario para converger: k o `iter`
- La norma de la función evaluada en el último iterado: $\left\|F\left(x^{(k)}\right)\right\|$
- La distancia entre los dos últimos iterados: $\left\|x^{(k)} - x^{(k-1)}\right\|$
- El orden de convergencia computacional aproximado: $ACOC$

Hay que destacar que, en ocasiones, la precisión que por defecto que aporta Matlab es insuficiente. Es por ello que se trabaja con aritmética de precisión variable. Para ello, basta con incluir en la segunda línea del primer **¡Error! No se encuentra el origen de la referencia.**1 el comando:

```
>> digits(200);
```

de forma que trabajaremos con 200 dígitos. Asimismo, deberemos explicitar que el vector que introducimos como estimación inicial tiene que estar en aritmética de precisión variable. Si queremos introducir el vector $x^{(0)} = (1,2)$, en Matlab deberíamos indicarlo como `vpa([1,2])`.

Los sistemas no lineales que vamos a resolver son:

- $F_1(x,y) = (e^x e^y + x\cos(y), x + y - 1)$
- $F_2(x_1,x_2) = (x_1^2 + x_2^2 - 1, x_1^2 - x_2^2 + 0.5)$
- $F_3(x,y,z,t) = (yz + t(y+z), xz + t(x+z), xy + t(x+y), xy + xz + yz - 1)$
- $F_4(x_1,x_2,x_3) = (x_1x_2 - 1, x_2x_3 - 1, x_3x_1 - 1)$

En la Tabla 2 se muestran los resultados de la aplicación de los diferentes métodos para resolver el sistema no lineal $F_1(x,y)$, tomando como estimación inicial $(x,y)^{(0)} = (0.5,-0.5)$ y como criterio de parada $\|x^{(k)} - x^{(k-1)}\| < 10^{-9}$.

Método	k	$\|F(x^{(k)})\|$	$\|x^{(k)} - x^{(k-1)}\|$	$ACOC$
NE	7	$1.8086e-15$	$9.4013e-31$	1.9997
TP	7	$4.5339e-11$	$1.8900e-32$	3.0911
GR	6	$2.1829e-22$	$3.0718e-37$	2.9914
NA	5	$3.3758e-10$	$3.1358e-35$	4.2886
JA	5	$7.6929e-12$	$1.2135e-46$	2.9202
RN	6	$2.2025e-28$	$5.3674e-141$	4.8594

Tabla 2. Comparativa numérica para $F_1(x,y)$ con $x^{(0)} = (0.5,-0.5)$.

Gracias al uso de la aritmética de precisión variable, podemos obtener resultados precisos con los 200 dígitos que hemos utilizado. Podemos observar en la Tabla 2 cómo los métodos NA y Jarratt son los que menos iteraciones han necesitado para converger a la solución. El método RN es el que mayor orden de convergencia computacional tiene; sin embargo, en cuanto al número de iteraciones es superior a NA y Jarratt.

Pasemos a obtener la comparativa numérica correspondiente a $F_2(x_1, x_2)$, tomando como estimación inicial $(x_1, x_2)^{(0)} = (0.5, -0.5)$ y como criterio de parada $\left\|x^{(k)} - x^{(k-1)}\right\| < 10^{-9}$. Los resultados se muestran en la Tabla 3.

Método	k	$\left\|F(x^{(k)})\right\|$	$\left\|x^{(k)} - x^{(k-1)}\right\|$	$ACOC$
NE	7	$8.6345e-19$	$1.0544e-36$	1.9999
TP	5	$6.2514e-16$	$1.9947e-46$	3.0073
GR	6	$2.4914e-25$	$7.4127e-41$	2.7159
NA	5	$4.4576e-20$	$6.8264e-55$	3.9694
JA	5	$4.3044e-37$	$1.6183e-146$	3.9986
RN	4	$1.2426e-15$	$8.0628e-76$	5.1436

Tabla 3. Comparativa numérica para $F_2(x_1, x_2)$ con $x^{(0)} = (0.5, -0.5)$.

En este caso, el método que mejores prestaciones obtiene es el método RN. De hecho, es un método que tiene como orden de convergencia teórico el valor 6, por lo que debería ser el que mejores prestaciones obtuviera en cualquier caso.

La Tabla 4 muestra la comparativa numérica cuando aplicamos los métodos sobre el sistema no lineal $F_3(x, y, z, t)$, tomando como estimación inicial $x^{(0)} = (1,1,1,1)$ y como criterio de parada $\left\|x^{(k)} - x^{(k-1)}\right\| < 10^{-9}$.

Método	k	$\|F(x^{(k)})\|$	$\|x^{(k)} - x^{(k-1)}\|$	$ACOC$
NE	7	$4.7580e-17$	$9.5436e-35$	2.0718
TP	5	$2.9035e-14$	$2.2851e-44$	3.3125
GR	5	$1.2474e-11$	$3.0541e-27$	3.3692
NA	5	$1.9626e-26$	$2.2491e-69$	4.3854
JA	5	$4.7574e-35$	$5.8590e-144$	4.2916
RN	4	$1.6568e-11$	$1.9715e-61$	6.3664

Tabla 4. Comparativa numérica para $F_3(x, y, z, t)$ con $x^{(0)} = (1,1,1,1)$.

De nuevo, se obtiene un comportamiento similar a lo que ocurría en el sistema F_2. El método de Newton es el que más iteraciones necesita para converger, y conforme aumenta el orden de convergencia computacional de los métodos, menor es el número de iteraciones requerido para alcanzar la solución.

Por último, vemos en la Tabla 5 la aplicación de los métodos sobre el sistema $F_4(x_1, x_2, x_3)$. Mantenemos el criterio de parada en $\|x^{(k)} - x^{(k-1)}\| < 10^{-9}$ y utilizamos el vector $x^{(0)} = (0.5,0.5,0.5)$ como estimación inicial para establecer la comparativa numérica.

Método	k	$\|F(x^{(k)})\|$	$\|x^{(k)} - x^{(k-1)}\|$	$ACOC$
NE	7	$1.9230e-15$	0	1.9968
TP	5	$4.5420e-13$	0	3.0173
GR	6	$1.9313e-14$	$3.3799e-30$	3.0116
NA	5	$8.7247e-14$	$1.0676e-42$	3.8949
JA	5	$1.0089e-30$	$4.9840e-122$	3.9962
RN	4	$4.6623e-13$	$3.0595e-64$	5.2111

Tabla 5. Comparativa numérica para $F_4(x_1, x_2, x_3)$ con $x^{(0)} = (0.5,0.5,0.5)$.

De nuevo se cumple el comportamiento esperado. Además, en este caso observamos cómo hay valores de $\|x^{(k)} - x^{(k-1)}\|$ que son iguales a cero. Lo que se debe interpretar es que Matlab no ha podido obtener un valor exacto con la precisión que le permiten los 200 dígitos, de forma que ha aplicado redondeo y el resultado es cero.

12.6. Implementación en Matlab: el método de Newton

En este apartado veremos la implementación del método de Newton para la resolución de sistemas de ecuaciones no lineales, basándonos en el **¡Error! No se encuentra el origen de la referencia.**. Recordemos que el método de Newton tiene la expresión iterativa:

$$x^{(k+1)} = x^{(k)} - \left[F'\left(x^{(k)}\right)\right]^{-1} F\left(x^{(k)}\right)$$

El archivo que contenga el sistema $F(x) = 0$ tiene que devolvernos el valor de la función y su matriz Jacobiana.

El código del método de Newton para sistemas no lineales se presenta en el siguiente código:

```
function [sol,iter,i1,i2,ACOC]=NewtonSys(x0,F,tol,maxiter)
% Inicialización de las variables
digits(200)
iter=1;
i1=1;
x0=x0(:);
[Fx0,dFx0]=feval(F,x0);
I=[]; ACOC=[];
while and(iter<maxiter,i1>tol)
      x=x0-dFx0\Fx0;
      i1=norm(x-x0);
      I=[I,i1];
      [Fx,dFx]=feval(F,x);
      i2=norm(fx0);
      iter=iter+1;
```

```
16        x0=x; Fx0=Fx; dFx0=dFx;
17  end
18  % Preparar datos de salida
19  sol=x;
20  if length(I)>2
21      ACOC=log(I(3:end)./I(2:end-1))./…
22            log(I(2:end-1)./I(1:end-2));
23  end
```

En primer lugar, se inicializan las variables con las que se va a trabajar. En este sentido, la variable `i1` va a representar la diferencia entre iterados, y la variable `i2` será el valor de la función en el iterado actual. Asimismo, la variable `I` va a almacenar los diferentes valores de `i1`, de modo que podamos calcular al final el valor de ACOC. En cuanto al método iterativo de Newton, su expresión tan solo ocupa la línea 10, de modo que este código se puede reutilizar para diferentes métodos iterativos.

Entremos en detalle con lo que ocurre en el bucle `while`. Destacar que, previo al acceso al bucle `while`, tenemos disponibles los valores de `Fx` y `dFx`, de forma que la primera línea del bucle es el método de Newton y, por tanto, el nuevo iterado. A continuación se van actualizando valores:

- Línea 11: la distancia al iterado anterior.
- Línea 12: el vector que registra todos los valores de distancias.
- Líneas 13 y 14 la evaluación del nuevo iterado y, por tanto, el valor de `i2.`
- Línea 15: el incremento en el número de iteraciones.
- Línea 16: la actualización de los valores para volver a ejecutar el bucle.

Por último, se preparan los datos de salida que no estaban definidos.

Ejemplo 5. Resuelve el sistema no lineal:

$$\begin{cases} x^2 + y^2 = 4 \\ (x-1)^2 + \left(y + \frac{1}{2}\right)^2 = 2 \end{cases}$$

utilizando el método de Newton. El criterio de parada se establece como:

$$\|x_k - x_{k-1}\| < 10^{-9}$$

Toma como aproximación inicial $x_0 = (2,1)$. Indica qué solución se obtiene, el número de iteraciones, la distancia entre los dos últimos iterados, el valor de la función en el último iterado y el ACOC.

En primer lugar, debemos implementar en Matlab la función que queremos resolver. Para ello, ponemos la expresión en forma $F(x) = 0$ y también calculamos su matriz Jacobiana.

$$F(X) = 0 \leftrightarrow \begin{cases} f_1(x,y) = x^2 + y^2 - 4 \\ f_2(x,y) = (x-1)^2 + \left(y + \frac{1}{2}\right)^2 - 2 \end{cases}$$

$$J_F(X) = \begin{bmatrix} 2x & 2y \\ 2(x-1) & 2\left(y + \frac{1}{2}\right) \end{bmatrix}$$

A continuación escribimos el archivo .m correspondiente:

```
1 function [F,dF]=ejemplo5(X)
2 x=X(1); y=X(2);
3 F=[x.^2+y.^2-4; (x-1).^2+(y+1/2).^2-2];
4 dF=[2*x, 2*y; 2*(x-1), 2*(y+1/2)];
```

Verificamos que la condición de parada es la que está establecida en el método de Newton que tenemos programado. Como el criterio de parada coincide con el del **¡Error! No se encuentra el origen de la referencia.**, no hacemos ninguna modificación en el archivo NewtonSys.m. Así pues, ejecutamos en la consola de Matlab:

```
>>
[sol,iter,i1,i2,ACOC]=NewtonSys(vpa([2,1]),'ejemplo5',1e-9,40)
```

La solución es:

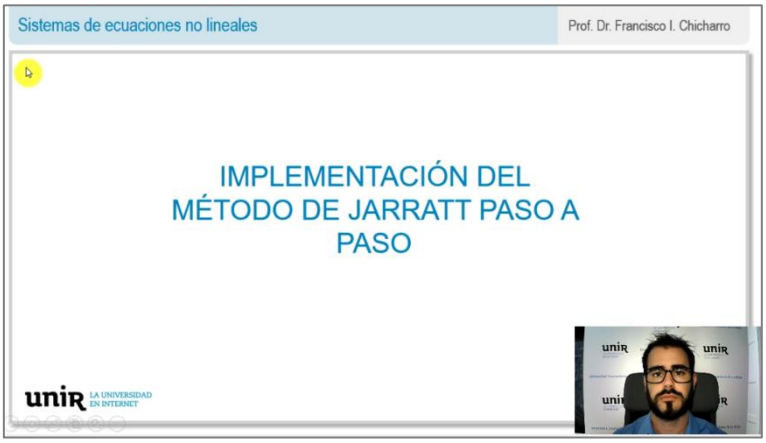
- $x^{(k)} = (1.9144, 0.5788)$
- $k = 6$
- $\|x_k - x_{k-1}\| = 3.68e - 14$
- $\|F(x^k)\| = 1.91e - 27$
- $ACOC = 1.9999$

Lo + recomendado

Lecciones magistrales

Implementación del método de Jarratt paso a paso

En esta lección magistral vamos a mostrar la implementación en Matlab del método de Jarratt paso a paso.



Accede a la lección magistral a través del aula virtual

No dejes de leer

Soluciones numéricas para sistemas de ecuaciones no lineales

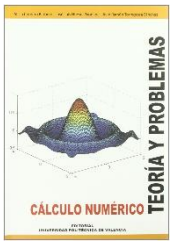
Chapra, S. C. y Canale, R. P. (2007). «Soluciones numéricas para sistemas de ecuaciones no lineales». En Autores, *Métodos numéricos para ingenieros* (5ª Ed.). McGraw-Hill.

En la parte final del Capítulo 6 de este libro se resuelven sistemas de ecuaciones no lineales a partir de diferentes métodos, entre ellos el de Newton. Se pueden observar diferentes ejemplos resueltos paso a paso.

Accede al libro a través de la Biblioteca Virtual de UNIR

Soluciones numéricas a sistemas de ecuaciones no lineales

Cordero, A. Hueso, J. L., y Torregrosa, J. R. (2004). *Cálculo numérico*. Universidad Politécnica de Valencia.



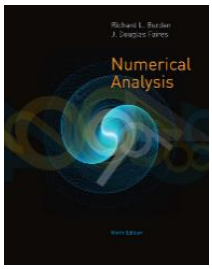
En el primer capítulo de este libro se presentan los sistemas de ecuaciones no lineales y diferentes métodos de resolución. Sobretodo, se centran en el método de Newton para resolver sistemas de ecuaciones no lineales, y presentan una serie de ejemplos.

+ Información

A fondo

Soluciones numéricas a sistemas de ecuaciones no lineales

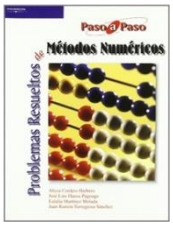
Burden, R. L. y Faires, J. D. (2011). *Numerical analysis* (9ª ed). Boston: Brooks/Cole CENGAGE learning.



El libro *Numerical Analysis* es el libro de referencia de esta asignatura. En el capítulo 10 se centran en resolver de forma numérica los sistemas de ecuaciones no lineales, a partir de diferentes métodos, entre los que se encuentra el método de Newton. Asimismo, se presentan una serie de ejemplos resueltos paso a paso.

Sistemas de ecuaciones no lineales

Cordero, A., Hueso, J. L., Martínez, E., Torregrosa, J. R. (2006). «Sistemas de ecuaciones lineales». En Autores, *Problemas resueltos de métodos numéricos*. Madrid: Thomson.



En el capítulo 8 se plantea la resolución de sistemas de ecuaciones no lineales. Para ello, se plantean los métodos de Newton, variantes del método de Newton y el método de Broyden. Asimismo, se plantean una serie de problemas resueltos y se proponen otros tantos.

Bibliografía

Cordero, A. y Torregrosa, J. (2010). On interpolation variants of Newton's method for functions of several variables. *Journal of Computational and Applied Mathematics, 234*, 34--43.

Cordero, A. y Torregrosa, J. R. (2007). Variants of Newton's method using fifth-order quadrature formulas. *Applied Mathematics and Computation, 190*, 686-698.

Cordero, A., Hueso, J., Martínez, E. y Torregrosa, J. (2010). A modified Newton-Jarratt's composition. *Numerical Algorithms, 55*, 87--99.

Cordero, A., Hueso, J., Martínez, E. y Torregrosa, J. (2011). Efficient high-order methods based on golden ratio for nonlinear systems. *Applied Mathematics and Computation, 217*, 4548--4556.

Kung, H. T. y Traub, J. F. (1974). Optimal order of one-point and multi-point iteration. *Applied Mathematics and Computation*, 643-651.

Ostrowski, A. (1960). *Solutions of equations and systems of equations.* New York: Academic Press.

Traub, J. (1982). *Iterative methods for the solution of equations.* New York: Chelsea Publishing Company.

Test

1. En un sistema de n ecuaciones y m incógnitas, la matriz Jacobiana tiene dimensiones:

A. $n \times m$.

B. $nm \times 1$.

C. Ninguna de las anteriores es correcta.

2. La distancia entre dos vectores se obtiene a partir de:

A. El valor absoluto de la diferencia entre ambos.

B. La norma de la diferencia entre ambos.

C. La raíz cuadrada de la diferencia entre ambos.

3. Para resolver el sistema $Ax = b$, en Matlab se utiliza el operador:

A. Barra (/)

B. Contrabarra (\).

C. Producto elemento a elemento (.*).

4. ¿Cuál de las siguientes técnicas no se utiliza para diseñar métodos iterativos?

A. Pseudocomposición.

B. Composición.

C. Inversión.

5. Si tenemos un método iterativo de orden p y lo componemos con el método de Newton, ¿qué orden tendrá el nuevo método?

A. 2.

B. $2p$.

C. $p + 2$.

6. ¿Qué efecto tiene congelar la derivada?

A. Reduce el orden del método.

B. Permite tener menor número de operaciones en cada iteración.

C. Las dos respuestas anteriores son correctas.

7. ¿Qué permite la aritmética de precisión variable?

A. Aumentar el número de bits con los que trabaja Matlab por defecto.

B. Poder utilizar en determinadas operaciones la barra (/) o la contrabarra (\).

C. Reducir matrices dispersas.

8. ¿Cuántas componentes de salida debe tener como mínimo una función .m que se utiliza para evaluar un sistema, si vamos a utilizar el método de Newton?

A. 1.

B. 2.

C. 3.

9. ¿Cuál de los siguientes métodos tiene menor índice de eficiencia computacional?

A. Newton.

B. Golden Ratio.

C. NA.

10. ¿Cuál es la matriz Jacobiana de $F(x,y) = (x+y, x^2+y^2)$?

A. $J_F(x,y) = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$

B. $J_F(x,y) = \begin{bmatrix} 1 & 1 \\ 2y & 2x \end{bmatrix}$

C. $J_F(x,y) = \begin{bmatrix} 1 & 1 \\ 2x & 2y \end{bmatrix}$