TEMA 3: RESOLUCIÓN NUMÉRICA DE ECUACIONES Y SISTEMAS DE ECUACIONES ALGEBRAÍCAS NO LINEALES

1. INTRODUCCIÓN

Objetivos del tema: resolver numéricamente sistemas algebráicos no lineales de la forma:

$$\left.\begin{aligned} f_1(x_1, x_2, \dots, x_n) &= 0 \\ f_2(x_1, x_2, \dots, x_n) &= 0 \\ \dots\dots\dots\dots\dots\dots & \\ f_n(x_1, x_2, \dots, x_n) &= 0 \end{aligned}\right\}$$

De forma más compacta, si $f: \mathbb{R}^n \to \mathbb{R}^n$, buscamos $x^* \in \mathbb{R}^n$ tal que $f(x^*) = 0$.

Si $n=1$, tenemos una ecuación escalar

$$f(x) = 0 \; ; \quad f: \mathbb{R} \to \mathbb{R}.$$

Si $n>1$, tenemos el caso vectorial o de sistemas

$$f(x) = 0 \; ; \quad f = (f_1, \dots, f_n): \mathbb{R}^n \to \mathbb{R}^n.$$

Buscaremos "aproximaciones numéricas" a la solución exacta x^*, y lo haremos de forma iterativa:

$$x^0 \longrightarrow x^1 \longrightarrow x^2 \longrightarrow \dots \longrightarrow x^n \simeq x^*$$

x^0 = inital guess

La forma en que calculamos x^{j+1} a partir de x^j vendrá determinada por el método numérico elegido.
Estudiaremos dos métodos: punto fijo y Newton.

¿Por qué hemos de estudiar estos métodos en GITI?
¿Dónde aparecen este tipo de problemas NO LINEALES?

En realidad, la OPTIMIZACION es la fuente más importante de problemas no lineales en Ingeniería.
Hamilton, y su famoso principio de Mínima Acción, es el culpable de todo esto.
Veamos dos ejemplos más concretos:

(1)* Programación No Lineal. Por ejemplo, para problemas con restricciones de igualdad, tipo

$$\begin{cases} \text{Minimizar} & f(x_1, \dots, x_n) \\ \text{sujeto a} & h_1(x_1, \dots, x_n) = 0 \\ & \dots \\ & h_d(x_1, \dots, x_n) = 0 \end{cases}$$

la solución $x^* = (x_1^*, \ldots, x_n^*)$ es, a su vez, solución del sistema de optimalidad (ecuaciones de Karush-Kuhn-Tucker)

$$\left.\begin{aligned} \nabla f(x^*) + \lambda^* \cdot \nabla h(x^*) = 0 \\ h(x^*) = 0 \end{aligned}\right\}$$

con $\lambda^* = (\lambda_1^*, \ldots, \lambda_d^*)$, $h = (h_1, \ldots, h_d)$,

que, en general, es un sistema no lineal algebraico en las incógnitas (x^*, λ^*).

(2)* Discretización de ecuaciones diferenciales no lineales.

Por ejemplo, las ecuaciones de Navier-Stokes para un fluido estacionario e incompresible

$$\left\{\begin{aligned} (\vec{u} \cdot \nabla)\vec{u} - \mu \Delta \vec{u} + \nabla p = 0 \quad &\text{en } \Omega \\ \nabla \cdot \vec{u} = 0 \quad &\text{en } \Omega \\ + \text{ cond. contorno} \end{aligned}\right.$$

$\vec{u} = (u_1, u_2, u_3) \equiv$ velocidad, $p =$ presión

El término convectivo $(\vec{u} \cdot \nabla)\vec{u}$ es NO LINEAL.

Si discretizamos dicho término cambiando la función $\vec{u}(x) = (u_1(x), u_2(x), u_3(x))$ por el vector

$$\bar{u}_h = (u_1^h, u_2^h, u_3^h)$$

con

$$u_i^h = (u_i^{h,1}, u_i^{h,2}, \ldots, u_i^{h,N}), \quad i=1,2,3$$

de modo que $u_i^{h,j} \simeq u_i(x_j)$

↑ punto de discretización o nodo de Ω,

obtendremos una ecuación algebraica NO LINEAL, donde aparecerán términos como

$$u_i^{h,j} \cdot \underbrace{\frac{u_i^{h,j+1} - u_i^{h,j}}{h}}_{\text{aproximación}}$$

$$\text{ss} \qquad \text{ss}$$

$$(\vec{u} \cdot \nabla)\cdot\vec{u}$$

NOTACIÓN:

Cuando la ecuación no lineal a resolver proviene de un problema finito-dimensional es habitual escribir

$$f(x) = 0, \qquad x = (x_1, \ldots, x_n).$$

Cuando el origen es un problema infinito-dimensional, es decir, una ecuación diferencial, es más común

$$g(u) = 0, \qquad u = (u_1, \ldots, u_n).$$

Veamos ahora los dos métodos numéricos fundamentales para resolver ecuaciones y sistemas de ecuaciones algebraícos no lineales.

2. MÉTODO DE PUNTO FIJO

Para simplificar, supongamos que $x \in \mathbb{R}$, y $f: \mathbb{R} \to \mathbb{R}$.

Es evidente que si x^* es solución de la ecuación $f(x)=0$, es decir, $f(x^*)=0$, entonces x^* es un punto fijo de la función $g(x) = x + f(x)$, es decir,

$$x^* = g(x^*) = x^* + f(x^*)$$

Por tanto, calcular las soluciones de $f(x)=0$ es equivalente a encontrar los puntos fijos de $g(x) = x + f(x)$.

En lo que sigue, nos centramos en resolver

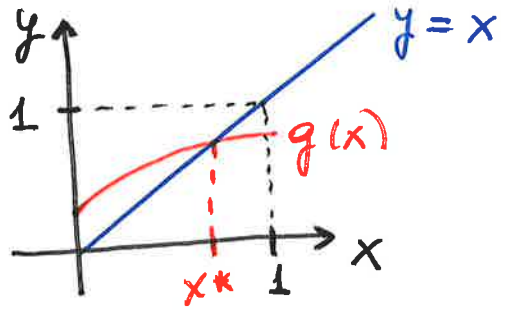
$$x = g(x).$$

Supongamos que:

(1) $g: [0,1] \longrightarrow [0,1]$

(2) g es continua.

Geométricamente, es evidente que entonces g tiene, al menos, un punto fijo.



$$x^* = g(x^*)$$

Una vez que estamos convencidos de la existencia de ese tal punto fijo x^*, el siguiente y principal objetivo es calcularlo, al menos aproximarlo numéricamente con la precisión que deseemos.

Probemos con el siguiente algoritmo numérico:

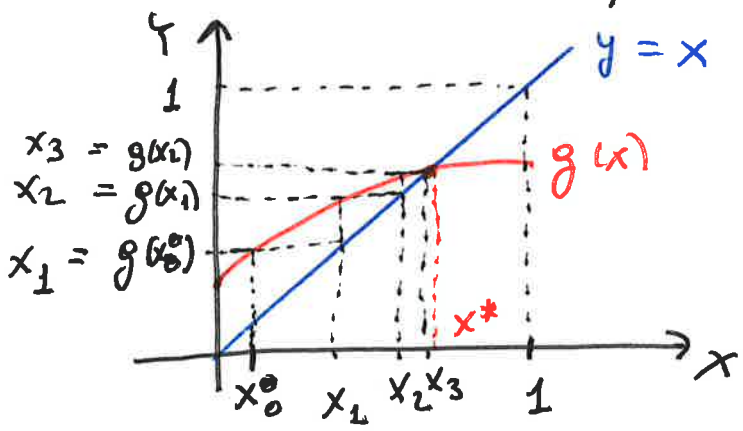
1º) Inicialización: tomar $x_0^{\bullet} \in [0,1]$ arbitrario.

2º) Iteración: para $n \geq 1$,

$$x_n = g(x_{n-1})$$

3º) Criterio de parada: $|x_n - x_{n-1}| \leq$ tolerancia.

Veamos geométricamente qué estamos haciendo:



Parece que todo va bien en el sentido de que a medida que iteramos, según $x_n = g(x_{n-1})$, los puntos x_n están cada vez más cerca de x^*. Es decir, todo parece indicar que este algoritmo es convergente. Ahora sólo nos falta probarlo. Como de costumbre, necesitaremos alguna hipótesis adicional sobre g. Esta nueva hipótesis es la famosa (en Cálculo Numérico) condición de Lipschitz.

<u>Definición</u>. Sea $g : \mathbb{R}^n \to \mathbb{R}^n$. Se dice que g es lipschitziana si existe $0 < L < 1$ tal que

$$\|g(x) - g(y)\| \leq L \|x - y\|, \quad \forall\, x, y \in \mathbb{R}^n.$$

En general, no es fácil chequear esta condición. Pero sí lo es cuando g es derivable y crece (o decrece) lentamente, es decir, cuando su gradiente cumple:

$$\|\nabla g(c)\| \leq L < 1 \quad \forall\, c \in \mathbb{R}^n.$$

En efecto, esto es consecuencia inmediata del teorema del valor medio:

$$g(x) - g(y) = \nabla g(c)\,(x - y)$$

de donde

$$\|g(x) - g(y)\| \leq \|\nabla g(c)\|\,\|x - y\|.$$

Volvamos al caso 1D donde habíamos supuesto:

(1) $g:[0,1] \to [0,1]$

(2) g es continua

y ahora añadimos la condición de Lipschitz

(3) $|g(x)-g(y)| \leq L\,|x-y|$, $\forall x,y \in [0,1]$, $0<L<1$.

Recordemos que las condiciones (1) y (2) garantizan la existencia del punto fijo x^*, tal que $x^* = g(x^*)$.

Además, la condición (1) nos asegura que el algoritmo $x_n = g(x_{n-1})$ está bien definido pues $g(x_{n-1}) \in [0,1]$.

Veamos ahora la convergencia del algoritmo:

$$\begin{aligned} |x_n - x^*| &= |g(x_{n-1}) - g(x^*)| \\ &\leq L\,|x_{n-1} - x^*| \\ &= L\,|g(x_{n-2}) - g(x^*)| \\ &\leq L \cdot L\,|x_{n-2} - x^*| \\ &= L^2\,|x_{n-2} - x^*| \\ &\leq \cdots\cdots \\ &\leq L^n\,|x_0 - x^*| \end{aligned}$$

Y como $0 < L < 1$, entonces $L^n \underset{n\to\infty}{\longrightarrow} 0$,

lo que garantiza que $|x_n - x^*| \longrightarrow 0$, cuando $n \to \infty$,

es decir $x_n \longrightarrow x^*$ cuando $n \to \infty$,

lo que prueba la convergencia del algoritmo.

Todo lo anterior se puede generalizar al caso multi-dimensional, es decir, cuando

$x = (x_1, \dots, x_n) \in \mathbb{R}^n$ y $g = (g_1, \dots, g_n) : \mathbb{R}^n \to \mathbb{R}^n$.

En concreto:

TEOREMA (MÉTODO DE PUNTO FIJO).

Sea $g : K \subset \mathbb{R}^n \to K$, siendo K cerrado y acotado, o bien $K = \mathbb{R}^n$. Supongamos que:

(1) g es continua

(2) g es lipschitziana.

Entonces el algoritmo de punto fijo

$$x_n = g(x_{n-1}), \qquad x_0 \text{ arbitrario}$$

converge a un punto fijo de g, es decir,

$x_n \underset{n\to\infty}{\longrightarrow} x^*$, con $x^* = g(x^*)$.

En la práctica, no es fácil comprobar las hipótesis del teorema anterior, sobre todo, el hecho de que g aplique un conjunto K en sí mismo y que g sea Lipschitziana.

Sin embargo, recordemos que la Simulación Numérica es la "rama experimental" de las Matemáticas.

Por tanto, en un problema concreto, siempre podemos implementar el algoritmo de punto fijo en un ordenador, ejecutar el código y ver qué pasa.

Aún falta un poco más de trabajo: la convergencia de un algoritmo iterativo (como el de punto fijo) es una propiedad esencial. Pero no es la única importante. El coste computacional, la velocidad de convergencia, del algoritmo, también es muy importante.

Interesa pués responder a la siguiente pregunta: cuando converje, ¿a qué velocidad converje el algoritmo de punto fijo?

La forma de averiguarlo es estimando la manera en que decrece el error de iteración en iteración.

Recordemos:

$e_n = |x_n - x^*|$ error en la iteración n-ésima

$e_{n-1} = |x_{n-1} - x^*|$ " " " " anterior

$$e_n = |x_n - x^*| \leq L\,|x_{n-1} - x^*| = L\,e_{n-1}, \quad \forall n.$$

Esta estimación nos afirma que el error en una iteración disminuye de manera proporcional (con constante de proporcionalidad L) al error en la iteración anterior.

Cuando esto sucede, en Cálculo Numérico se dice que la convergencia es lineal o de orden 1.

Ejercicio Octave:

Tomar $L = 0.5$, $e0 = 0.5$, $e1 = L * e0$,

~~Definir los vectores~~

~~n = [0 1 2 3 4 5]~~

~~error_lineal = [e0, L*e~~

$e2 = L * e1$,

$e3 = L * e2$

$e4 = L * e3$

$e5 = L * e4$.

$E1 = L * e0^2$, $E2 = L * E1^2$, $E3 = L * E2^2$

$E4 = L * E3^2$, $E5 = L * E4^2$.

Definir los vectores

n = [0 1 2 3 4 5]

error_lineal = [e0 e1 e2 e3 e4 e5]

error_cuadratico = [e0 E1 E2 E3 E4 E5].

DS

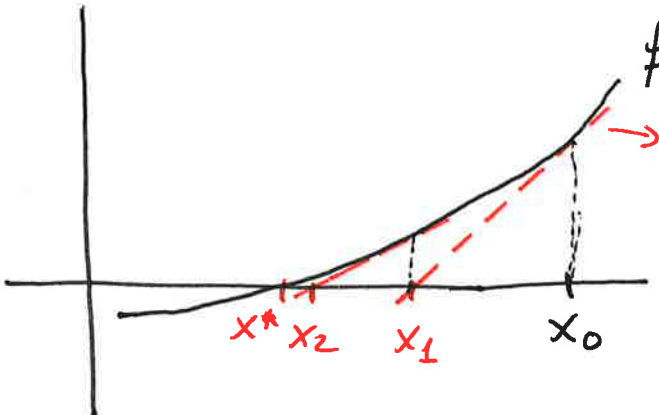
Dibujar en una misma gráfica n frente a error_lineal y n frente a error_cuadratico.

Uno de los inconvenientes del método de punto fijo es que puede ser lento, es decir, que se requieran muchísimas iteraciones para obtener la precisión deseada. Sería, pues, conveniente disponer de algún otro algoritmo que nos proporcionase una convergencia, al menos, cuadrática.

Esta labor la desempeña, de manera excepcional, el llamado algoritmo de Newton (o Newton-Raphson).

3. MÉTODO DE NEWTON .... Y VARIANTES

Idea geométrica del método de Newton:



$\rightarrow y = f(x_0) + f'(x_0)\cdot(x - x_0)$

recta que pasa por $(x_0, f(x_0))$ y es tangente a $f(x)$ en dicho punto.

$$0 = f(x) \simeq f(x_0) + f'(x_0)\cdot(x - x_0)$$

$$0 = f(x_0) + f'(x_0)\cdot x_1 - f'(x_0)\cdot x_0 ;$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} .$$

La idea consiste en elegir x_0 "cerca" de la solución x^* de la ecuación $f(x)=0$.

Localmente, $f(x)$ es similar a la recta que pasa por $(x_0, f(x_0))$ y es tangente a f en dicho punto. Por tanto, el punto donde se anula f debe estar próximo al punto donde se anula la recta

$$y = f(x_0) + f'(x_0)\cdot(x-x_0).$$

Igualando a cero esta ecuación y despejando x se obtiene el nuevo punto

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

y el proceso se repite.

Algoritmo de Newton para ecuaciones escalares

1º) Inicialización: tomar x^0, próximo a x^*.

2º) Iteración: para $n \geq 1$

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

3º) Criterio de parada: $|f(x_n)| \leq$ tolerancia.

¿Cuándo es convergente el método de Newton?

* Cuando x^0 está próximo a x^*
* Cuando $f'(x)$ no se anula en el intervalo donde estemos trabajando.

Nótese que el método de Newton es más caro, desde el punto de vista computacional, que el método de punto fijo pues en cada iteración hemos de evaluar $f(x_n)$ (como en el método de punto fijo) y además la inversa de $f'(x_n)$.
Piénsese en un sistema! Estamos hablando de calcular en cada iteración la matriz Jacobiana, evaluarla en x_n, y calcular su inversa (es decir, resolver un sistema lineal!)
Sin embargo, el método de Newton tiene una gran virtud: es muy rápido. En concreto, proporciona una convergencia cuadrática. En efecto:

TEOREMA

Sea $f:[a,b] \to \mathbb{R}$ de clase C^2 (existen las derivadas de f hasta orden 2 y son continuas), y supongamos que f' no se anula en $[a,b]$. Supongamos también que la ecuación $f(x)=0$ tiene solución x^* en dicho intervalo. Sea x_n la sucesión obtenida por el método de Newton.
Si $|x_n - x^*| \leq h$ entonces $|x_{n+1} - x^*| \leq \frac{M}{2m} h^2$

donde $M = \max_{a \leq x \leq b} |f''(x)|$ y $m = \min_{a \leq x \leq b} |f'(x)|$.

- Demostración -

Desarrollando por Taylor hasta orden 2:

$$0 = f(x^*) = f(x_n) + f'(x_n)(x^* - x_n) + \frac{1}{2} f''(c)(x^* - x_n)^2$$

con $c \in [x_n, x^*]$ o $[x^*, x_n]$.

Dividiendo por $f'(x_n)$:

$$-\frac{f(x_n)}{f'(x_n)} = x^* - x_n + \frac{1}{2}\frac{f''(c)}{f'(x_n)}(x^* - x_n)^2;$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x^* + \frac{1}{2}\frac{f''(c)}{f'(x_n)}(x^* - x_n)^2;$$

$$|x_{n+1} - x^*| \leq \frac{1}{2}\frac{M}{m}|x_n - x^*|^2 = \frac{1}{2}\frac{M}{m}h^2. \quad \blacksquare$$

Algoritmo de Newton para sistemas

Supongamos ahora que $\vec{f} = (f_1, \dots, f_m) : \mathbb{R}^m \to \mathbb{R}^m$

y buscamos una solución del sistema

$$\vec{f}(x) = 0; \qquad x = (x_1, \dots, x_m).$$

El algoritmo es el mismo que en el caso escalar, pero cambiando derivadas por gradientes (o matrices Jacobianas).

Recordemos que el gradiente de una función vectorial $\vec{f}$, (también más conocido en matemáticas como matriz Jacobiana), es la matriz con entradas

$$(\nabla \vec{f})_{ij} \equiv J(\vec{f})_{ij} = \left(\frac{\partial f_i}{\partial x_j}\right).$$

Repitiendo los mismos cálculos que en el caso escalar:

~~$0 = \vec{f}(x^*) \simeq f(x_{n-1}) + f'(x_{n-1}) \cdot (x_n - x_{n-1})$~~

se tiene:

$$0 = \vec{f}(x^*) \simeq \vec{f}(x_{n-1}) + (J\vec{f})(x_{n-1}) \underbrace{(x_n - x_{n-1})}_{\Delta x_n}.$$

El algoritmo queda:

1º) Inicialización: tomar x_0 cerca de x^*.

2º) Iteración: para $n \geq 1$:

2.1) Resolver el sistema lineal

$$(J\vec{f})(x_{n-1})\, \Delta x_n = -\vec{f}(x_{n-1}).$$

2.2) $x_n = x_{n-1} + \Delta x_n$

3º) Criterio de parada: $\|f(x_n)\| \leq$ tolerancia.

Nota.- Nótese que hemos evitado calcular $[J\vec{f}(x_{n-1})]^{-1}$.
Es más conveniente resolver el sistema

$$(J\vec{f})(x_{n-1})\ \Delta x_n = -\vec{f}(x_{n-1}),$$

para lo cual podemos hacer uso de toda la artillería de métodos que vimos en el tema anterior.

Conclusión: resolver un sistema no lineal mediante el método de Newton "equivale" a resolver varios sistemas lineales.

Variantes del Método de Newton

Hay varios métodos que modifican al de Newton. La mayoría de ellos van encaminados a aliviar el alto coste computacional que supone calcular la matriz Jacobiana y su "inversa" en cada iteración. Estos métodos superan el alcance de este curso. Una breve descripción de estos métodos puede encontrarse en el libro de Strang., p. 174.