Informática II

Clase 4: Sistemas de Ecuaciones no lineales

Mario Merino Martínez
mario.merino@upm.es

Escuela de Ingeniería Aeronáutica y del Espacio
Universidad Politécnica de Madrid

2 de abril de 2013

Método de Newton para Sistemas

- En múltiples ocasiones debemos resolver varias ecuaciones acopladas, con varias variables, pero no es posible hacerlo analíticamente. Recurrimos a **métodos iterativos**
- El **método de Newton para sistemas** está basado en la misma idea que el de una ecuación: **linealizar** todas las ecuaciones en torno a una aproximación de la raíz, y **resolver** la versión linealizada de las mismas.
- Repitiendo el proceso, **el método converge o no** hacia la raíz deseada dependiendo de la condición inicial.
- Salvo raíces múltiples/etc., **la convergencia es cuadrática**

Ecuaciones a resolver

- Ponemos todas las ecuaciones en la forma

$$\begin{cases} f_1(x_1, x_2) = 0 \\ f_2(x_1, x_2) = 0 \end{cases}$$

- En general podemos escribirlo **vectorialmente**: $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$, con $\boldsymbol{f}$, $\boldsymbol{x}$ vectores de dimensión n

Un ejemplo sencillo (ver apuntes):

$$\begin{cases} x_1^2 + x_2^2 = 1 \\ x_1^3 = x_2 \end{cases} \Rightarrow \begin{cases} x_1^2 + x_2^2 - 1 = 0 \\ x_1^3 - x_2 = 0 \end{cases}$$

Linealización

$$\begin{cases} f_1(x_1, x_2) = 0 \\ f_2(x_1, x_2) = 0 \end{cases}$$

- Tomamos una **aproximación inicial**, $\boldsymbol{x}^0 = (x_1^0, x_2^0)$
- **Aproximamos** la función vectorial $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ con su serie de Taylor hasta orden 1:

$$\begin{cases} f_1^0 + (x_1 - x_1^0) \left.\dfrac{\partial f_1}{\partial x_1}\right|_{\boldsymbol{x}^0} + (x_2 - x_2^0) \left.\dfrac{\partial f_1}{\partial x_2}\right|_{\boldsymbol{x}^0} = 0 \\ f_2^0 + (x_1 - x_1^0) \left.\dfrac{\partial f_2}{\partial x_1}\right|_{\boldsymbol{x}^0} + (x_2 - x_2^0) \left.\dfrac{\partial f_2}{\partial x_2}\right|_{\boldsymbol{x}^0} = 0 \end{cases}$$

Formulación vectorial

El sistema anterior puede escribirse de forma vectorial como sigue:

$$J^0 \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = - \begin{bmatrix} f_1^0 \\ f_2^0 \end{bmatrix}$$

donde $\Delta \boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}^0$, y J^0 es la **matriz Jacobiana** de $\boldsymbol{f}$ en $\boldsymbol{x}^0$, que recoge todas las derivadas de las distintas f_i respecto a las distintas x_j ($J_{ij} = \partial f_i / \partial x_j$):

$$J^0 = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} \end{bmatrix}_{\boldsymbol{x}^0}$$

Nótese que esta notación es más apropiada para cualquier dimensión n

Resolución

- El sistema resultante es **lineal**, por tanto lo podemos resolver fácilmente (e.g. con factorización LU):

$$\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = -\left(J^0\right)^{-1} \begin{bmatrix} f_1^0 \\ f_2^0 \end{bmatrix}$$

- La **solución** de este sistema proporciona $\boldsymbol{x}^1 = \boldsymbol{x}^0 + \Delta \boldsymbol{x}$, que se utiliza como **nueva aproximación inicial** para repetir el proceso hasta la convergencia

Criterio de parada

Al igual que en una única variable, hay que **decidir cuándo terminar la iteración**. Utilizaremos el mismo criterio doble, pero usando **la norma de las variables vectoriales**. Los dos criterios que han de cumplirse simultáneamente para terminar en la iteración n son:

- Norma del **error en $\boldsymbol{f}^n$** menor que la tolerancia ε:

$$\|\boldsymbol{f}^n\| < \varepsilon$$

- **Distancia relativa de $\boldsymbol{x}^n$ a la raíz $\boldsymbol{\alpha}$** menor que la tolerancia ε:

$$\frac{\|\boldsymbol{\alpha} - \boldsymbol{x}^n\|}{\|\boldsymbol{\alpha}\|} \simeq \frac{\|\boldsymbol{x}^{n+1} - \boldsymbol{x}^n\|}{\|\boldsymbol{x}^n\|} < \varepsilon$$

Como siempre, también limitamos las iteraciones máximas a realizar (10 ó 20 es más que suficiente)

Implementación

- La **subrutina** `newton_sistemas(x,fun,Jfun,n,itermax,tol)`:
  - Declara los vectores `x(n)`, `f_(n)`, y la matriz `J_(n,n)`
  - Recibe la **condición inicial** a traves de la **variable x**
  - Recibe las **subrutinas `fun` y `Jfun`**, con la función a resolver y su jacobiana
  - **llama** a `Jfun(x,J_)`, **factoriza** `J_`, y **resuelve** el sistema $J \cdot \Delta \boldsymbol{x} = -\boldsymbol{f}$ en cada iteración
  - Comprueba el **criterio de parada** en cada iteración (utiliza función `norma_2`)
  - Devuelve la **solución** a través de la **variable `x`**
- La función $\boldsymbol{f}(\boldsymbol{x})$ y su Jacobiana $J(\boldsymbol{x})$ se implementan en **dos subrutinas (¡no "function"!)**, `fun(x,f)`, `Jfun(x,J)`. Las subrutinas toman $\boldsymbol{x}$ (vector) y devuelven $\boldsymbol{f}$ (vector) y J (matriz)

Notas finales

- La subrutina `newton_sistemas` también es aplicable cuando `n=1`
- El método tiene una **clara interpretación geométrica en dos variables** (x, y): cada componente de $f_i = z$ se aproxima con su plano tangente en el punto $\boldsymbol{x}^0$. El nuevo punto $\boldsymbol{x}^1$ es la intersección común de los dos planos con el plano $z = 0$
- En sistemas es especialmente importante la **elección de la condición inicial**, ya que hacerlo mal implicará la no convergencia del método
- La **jacobina** ha de ser una **matriz regular** (i.e., existe inversa) en todos los puntos $\boldsymbol{x}^k$ de la iteración