Métodos iterativos para sistemas no lineales

Matemáticas de la Especialidad
Manuel Colera Rico
Febrero de 2026

---

1. Introducción

En este capítulo se estudiará la resolución de sistemas no lineales; véase la sección 6.6 a modo de ejemplo introductorio.

En general, un sistema no lineal de N ecuaciones y N incógnitas se puede escribir de la forma

$$f_i(x_1, \dots, x_N) = 0, \quad i = 1, \dots, N, \tag{1}$$

donde $x_1, \dots, x_N$ son las *incógnitas*, y $f_1, \dots, f_N$ son los *residuos*. En notación vectorial, (1) se escribe como

$$\mathbf{f}(\mathbf{x}) = \mathbf{0},$$

donde $\mathbf{x} = [x_1, \dots, x_N]^{\mathrm{T}}$ es el *vector de incógnitas* y $\mathbf{f} = [f_1, \dots, f_N]^{\mathrm{T}}$ es el *vector residuo*. Notemos que $\mathbf{x}$ y $\mathbf{f}$ son vectores columna, lo que hará más cómodos de escribir algunos de los desarrollos que siguen.

Salvo para casos muy particulares, no es posible resolver (1) mediante un número finito de pasos, a diferencia de lo estudiado en el tema 1 para sistemas lineales. En su lugar, hay que resolver (1) de forma iterativa; es decir, partiendo de una *aproximación inicial* $\mathbf{x}^0$ supuesta conocida y suficientemente cercana a la solución exacta $\mathbf{x}^{\mathrm{ex}}$, se genera una secuencia de vectores $\mathbf{x}^1, \mathbf{x}^2, \dots$ que converge a $\mathbf{x}^{\mathrm{ex}}$.

Nota 1: *No existe ningún criterio para determinar a priori si la aproximación inicial* $\mathbf{x}^0$ *es "suficientemente cercana a la solución exacta* $\mathbf{x}^{\mathrm{ex}}$*". Esto sólo se puede saber mediante la experiencia: si el método utilizado produce una solución satisfactoria en un tiempo razonable, entonces* $\mathbf{x}^0$ *es una buena aproximación inicial; si no lo hace, hay que probar con un nuevo valor para* $\mathbf{x}^0$ *o con otro método.*

Nota 2: *Si existe, la solución al sistema no lineal* (1) *no tiene por qué ser única; puede haber múltiples soluciones. Para encontrar distintas soluciones, es necesario establecer diferentes aproximaciones iniciales* $\mathbf{x}^0$.

2. El método de Newton–Raphson

El método de Newton–Raphson es el más popular para resolver sistemas no lineales, aunque no necesariamente el más eficiente.

Sea $\mathbf{x}^k$ la aproximación a la solución exacta $\mathbf{x}^{\mathrm{ex}}$ en la k-ésima iteración, y sea $\mathbf{f}^k := \mathbf{f}(\mathbf{x}^k)$ el correspondiente valor del residuo. La idea del método de Newton–Raphson consiste en linealizar $\mathbf{f}(\mathbf{x})$ cerca del punto $\mathbf{x}^k$. Esto nos proporciona

$$f_i(\mathbf{x}) \approx f_i^k + \sum_{j=1}^{N} \frac{\partial f_i}{\partial x_j}(\mathbf{x}^k)\left(x_j - x_j^k\right), \quad i = 1, \dots, N,$$

o, matricialmente,

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}^k + \mathbb{J}(\mathbf{x}^k)\left(\mathbf{x} - \mathbf{x}^k\right),$$

donde

$$\mathbb{J}(\mathbf{x}) := \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1}(\mathbf{x}) & \cdots & \dfrac{\partial f_1}{\partial x_N}(\mathbf{x}) \\ \vdots & & \vdots \\ \dfrac{\partial f_N}{\partial x_1}(\mathbf{x}) & \cdots & \dfrac{\partial f_N}{\partial x_N}(\mathbf{x}) \end{bmatrix} \tag{2}$$

es la *matriz jacobiana* de la función $\mathbf{f}(\mathbf{x})$.

El valor de $\mathbf{x}$ en la siguiente iteración, $\mathbf{x}^{k+1}$ se elige de tal forma que $\mathbf{f}(\mathbf{x}^{k+1}) \approx \mathbf{0}$, es decir,

$$\mathbf{0} = \mathbf{f}^k + \mathbb{J}(\mathbf{x}^k)\left(\mathbf{x}^{k+1} - \mathbf{x}^k\right).$$

Esto proporciona la iteración

$$\mathbf{p}^k := -\mathbb{J}^{-1}(\mathbf{x}^k)\,\mathbf{f}^k, \tag{3}$$

$$\mathbf{x}^{k+1} := \mathbf{x}^k + \mathbf{p}^k, \tag{4}$$

Por supuesto, en la expresión (3) no se calcula explícitamente la matriz inversa de $\mathbb{J}(\mathbf{x}^k)$; en su lugar, se resuelve el sistema $\mathbb{J}(\mathbf{x}^k)\mathbf{p}^k = -\mathbf{f}^k$ de la forma más adecuada, tal y como se explicó en el tema 1.

La iteración (3)-(4) se inicia con una aproximación inicial $\mathbf{x}^0$ dada. Para finalizar el algoritmo, existen multitud de criterios. En este curso, tomaremos como criterio de parada que la media cuadrática de los elementos del vector incremento $\mathbf{p}^k = [p_1^k, \ldots, p_N^k]^{\mathrm{T}}$ sea menor o igual que una cierta tolerancia prescrita Tol, es decir,

$$\frac{1}{N}\sqrt{\left(p_1^k\right)^2 + \ldots + \left(p_N^k\right)^2} \leq \mathrm{Tol},$$

o, equivalentemente, cuando

$$\left\|\mathbf{p}^k\right\|_2 \leq \sqrt{N}\,\mathrm{Tol}. \tag{5}$$

Notemos que la expresión anterior, junto con (3), es equivalente a

$$\left\|\mathbf{f}^k\right\|_2 = \left\|\mathbb{J}(\mathbf{x}^k)\mathbf{p}^k\right\|_2 \leq \left\|\mathbb{J}(\mathbf{x}^k)\right\|_2 \left\|\mathbf{p}^k\right\|_2 \leq \sqrt{N}\left\|\mathbb{J}(\mathbf{x}^k)\right\|_2 \mathrm{Tol},$$

lo que garantiza que la solución hallada satisface (1) bajo una cierta tolerancia.

Nota 3: *Si la matriz jacobiana $\mathbb{J}$ se calcula de forma exacta y la aproximación inicial $\mathbf{x}^0$ está suficientemente cerca de la solución exacta $\mathbf{x}^{\mathrm{ex}}$, es posible demostrar que el método de Newton–Raphson converge de forma cuadrática, es decir, existe una constante $C > 0$ tal que*

$$\left\|\mathbf{x}^{k+1} - \mathbf{x}^{\mathrm{ex}}\right\| \leq C\left\|\mathbf{x}^k - \mathbf{x}^{\mathrm{ex}}\right\|^2.$$

Esto hace que el método de Newton–Raphson converja en pocas iteraciones.

Nota 4: *En algunos casos prácticos, es difícil calcular la matriz jacobiana $\mathbb{J}$ de forma exacta. No obstante, si se calcula una buena aproximación a la misma, la iteración (3)-(4) también proporciona la solución al sistema no lineal (1). Esto se debe a que el vector incremento $\mathbf{p}^k$ únicamente determina, de forma aproximada, cuánto debe valer $\mathbf{x}^{k+1}$ para estar más cerca de la solución exacta que $\mathbf{x}^k$. Por tanto, no es fundamental calcular $\mathbf{p}^k$ con total precisión, de ahí que se pueda utilizar una matriz jacobiana*

aproximada, en lugar de la exacta. Ahora bien, en esos casos, la convergencia es lineal, es decir, existe una constante $C > 0$ tal que

$$\left\| \mathbf{x}^{k+1} - \mathbf{x}^{\text{ex}} \right\| \leq C \left\| \mathbf{x}^{k} - \mathbf{x}^{\text{ex}} \right\|,$$

y por tanto son necesarias más iteraciones.

Nota 5: *El principal inconveniente del método de Newton–Raphson es que requiere calcular la matriz jacobiana $\mathbb{J}$ en cada iteración y resolver el correspondiente sistema (3), lo cual puede ser costoso para sistemas grandes.*

Véase el ejercicio 1 para la implementación del método de Newton–Raphson.

3. Método de Anderson

El método de Anderson, perteneciente a la familia de los métodos *multisecante*, es un método iterativo que no requiere calcular la matriz jacobiana ni resolver los sistemas lineales asociados. A pesar de que normalmente requiere un número mayor de iteraciones que el método de Newton–Raphson, cada una de las iteraciones es menos costosa y, por tanto, suele resultar más eficiente.

Para entender este método, es conveniente introducir primero los conceptos de *sistema precondicionado* y las *condiciones multisecante*.

3.1. Precondicionamiento

En primer lugar, consideremos, en lugar del sistema original $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, el *sistema precondicionado*

$$\mathbf{g}(\mathbf{x}) = \mathbf{0}, \tag{6}$$

donde $\mathbf{g} : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ se elige de tal forma que se satisfacen las siguientes condiciones:

(C1) Resolver $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ es equivalente a resolver $\mathbf{f}(\mathbf{x}) = \mathbf{0}$.
(C2) $\mathbf{g}(\mathbf{x})$ no es mucho más costoso de calcular que $\mathbf{f}(\mathbf{x})$.
(C3) La matriz jacobiana de $\mathbf{g}(\mathbf{x})$ es aproximadamente igual a la identidad, $\partial \mathbf{g}(\mathbf{x})/\partial \mathbf{x} \approx \mathbb{I}$.

Por ejemplo, dada la aproximación inicial $\mathbf{x}^0$ y suponiendo que la matriz jacobiana $\mathbb{J}(\mathbf{x}^0)$ no es singular, se puede definir

$$\mathbf{g}(\mathbf{x}) := \left(\mathbb{J}(\mathbf{x}^0)\right)^{-1} \mathbf{f}(\mathbf{x}), \tag{7}$$

Si se calcula $\mathbb{J}(\mathbf{x}^0)$ y su factorización, $\mathbf{g}(\mathbf{x})$ se puede evaluar rápidamente. Además, si $\mathbf{x}$ está "suficientemente cerca" de $\mathbf{x}^0$, entonces $\partial \mathbf{g}(\mathbf{x})/\partial \mathbf{x} = \left(\mathbb{J}(\mathbf{x}^0)\right)^{-1} \mathbb{J}(\mathbf{x}) \approx \mathbb{I}$.

Otra opción es considerar

$$\mathbf{g}(\mathbf{x}) := \operatorname{diag}\left(\mathbb{J}(\mathbf{x})\right)^{-1} \mathbf{f}(\mathbf{x}), \tag{8}$$

donde $\operatorname{diag}(\mathbb{A})$ denota la matriz diagonal formada por la diagonal de una matriz cualquiera $\mathbb{A}$. Dicho de otra forma,

$$g_i(\mathbf{x}) := \left(\frac{\partial f_i}{\partial x_i}(\mathbf{x}) \right)^{-1} f_i(\mathbf{x}), \quad i = 1, \ldots, N.$$

En ese caso, sólo sería necesario calcular los elementos de la diagonal de la matriz jacobiana $\mathbb{J}(\mathbf{x})$. Además, $\partial \mathbf{g}(\mathbf{x})/\partial \mathbf{x} = \operatorname{diag}\left(\mathbb{J}(\mathbf{x})\right)^{-1} \mathbb{J}(\mathbf{x}) \approx \mathbb{I}$.

Nota 6: *Aquí, los términos "aproximadamente", "suficientemente cerca", etc., se interpretan en un sentido heurístico, y no riguroso. Las aproximaciones hechas anteriormente serán más o menos aceptables según el problema en cuestión.*

3.2. Condiciones multisecante

Si se aplicara el método de Newton–Raphson al sistema precondicionado (6), las iteraciones (3)-(4) se podrían escribir de la forma

$$\mathbf{p}^k := -\left(\frac{\partial \mathbf{g}}{\partial \mathbf{x}}(\mathbf{x}^k)\right)^{-1} \mathbf{g}^k, \tag{9}$$

$$\mathbf{x}^{k+1} := \mathbf{x}^k + \mathbf{p}^k, \tag{10}$$

con $\mathbf{g}^k := \mathbf{g}(\mathbf{x}^k)$. Puesto que calcular la matriz jacobiana $\partial \mathbf{g}(\mathbf{x}^k)/\partial \mathbf{x}$ y resolver el correspondiente sistema lineal (9) es muy costoso, los *métodos multisecante* tratan, en su lugar, de hallar un operador $\mathbb{H}^k$ que se comporte de forma similar a la matriz jacobiana inversa $\left(\partial \mathbf{g}(\mathbf{x}^k)/\partial \mathbf{x}\right)^{-1}$, es decir,

$$\mathbb{H}^k \approx \left(\frac{\partial \mathbf{g}(\mathbf{x}^k)}{\partial \mathbf{x}}\right)^{-1}. \tag{11}$$

De esta forma, se tiene simplemente

$$\mathbf{p}^k := -\mathbb{H}^k \mathbf{g}^k, \tag{12}$$

$$\mathbf{x}^{k+1} := \mathbf{x}^k + \mathbf{p}^k, \tag{13}$$

Notemos que, si se satisface (11), un simple desarrollo de Taylor proporciona

$$\mathbf{x} - \mathbf{x}^k \approx \mathbb{H}^k \left(\mathbf{g}(\mathbf{x}) - \mathbf{g}^k\right). \tag{14}$$

El operador $\mathbb{H}^k$ se calcula a partir de los valores de $\mathbf{x}$ y $\mathbf{g}$ en las últimas $m := \min\{k, M\}$ iteraciones, siendo $1 \leq M \leq N$ un dato conocido. Los métodos multisecante difieren entre sí según cómo construyen el operador $\mathbb{H}^k$.

En el caso particular del método de Anderson (o método de Broyden generalizado II), perteneciente a la familia de métodos multisecante, el operador $\mathbb{H}^k$ se construye como sigue. Sean

$$\mathbb{X}^k = \left[\frac{\mathbf{x}^k - \mathbf{x}^{k-1}}{\left\|\mathbf{g}^k - \mathbf{g}^{k-1}\right\|_2 + \varepsilon} \,\middle|\, \cdots \,\middle|\, \frac{\mathbf{x}^{k-m+1} - \mathbf{x}^{k-m}}{\left\|\mathbf{g}^{k-m+1} - \mathbf{g}^{k-m}\right\|_2 + \varepsilon} \right] \in \mathbb{R}^{N \times m} \tag{15}$$

$$\mathbb{G}^k = \left[\frac{\mathbf{g}^k - \mathbf{g}^{k-1}}{\left\|\mathbf{g}^k - \mathbf{g}^{k-1}\right\|_2 + \varepsilon} \,\middle|\, \cdots \,\middle|\, \frac{\mathbf{g}^{k-m+1} - \mathbf{g}^{k-m}}{\left\|\mathbf{g}^{k-m+1} - \mathbf{g}^{k-m}\right\|_2 + \varepsilon} \right] \in \mathbb{R}^{N \times m} \tag{16}$$

dos matrices que almacenan las variaciones de $\mathbf{x}$ y $\mathbf{g}$ en las últimas m iteraciones (normalizadas para mejorar la estabilidad numérica y con $m \leq N$), con $\varepsilon = 10^{-12}$ un valor pequeño para evitar posibles divisiones por cero. Puesto que el residuo precondicionado $\mathbf{g}$ se define de tal forma que $\partial \mathbf{g}/\partial \mathbf{x} \approx \mathbb{I}$, en virtud de (11) se supondrá que $\mathbb{H}^k$ es la matriz identidad más una corrección de rango m, es decir, es de la forma

$$\mathbb{H}^k = \mathbb{I} + \mathbb{A}\mathbb{B}, \quad \mathbb{A} \in \mathbb{R}^{N \times m}, \quad \mathbb{B} \in \mathbb{R}^{m \times N}. \tag{17}$$

Las matrices rectangulares $\mathbb{A}$ y $\mathbb{B}$ se eligen de tal modo que se satisfacen condiciones análogas a (14):

(i) Las llamadas *condiciones multisecante*, es decir, $\mathbb{H}^k$ tiene que "predecir" los cambios en $\mathbf{x}$ a partir de los cambios en $\mathbf{g}$ para las m últimas iteraciones,

$$\begin{aligned} \mathbb{H}^k(\mathbf{g}^k - \mathbf{g}^{k-1}) &= \mathbf{x}^k - \mathbf{x}^{k-1}, \\ \vdots \quad &= \quad \vdots \\ \mathbb{H}^k(\mathbf{g}^{k-m+1} - \mathbf{g}^{k-m}) &= \mathbf{x}^{k-m+1} - \mathbf{x}^{k-m}. \end{aligned}$$

(ii) $\mathbb{H}^k$ tiene que "predecir" los mismos cambios en $\mathbf{x}$ que la matriz identidad en direcciones ortogonales al espacio

$$\mathcal{S} := \operatorname{span}\left\{\mathbf{g}^k - \mathbf{g}^{k-1}, \ldots, \mathbf{g}^{k-m+1} - \mathbf{g}^{k-m}\right\},$$

es decir

$$\mathbb{H}^k\mathbf{q} = \mathbb{I}\mathbf{q} \quad \forall \mathbf{q} \perp \mathcal{S} = 0.$$

La condición (ii) es equivalente a

$$\mathbb{H}^k\mathbf{q} = \mathbf{q} \quad \forall \mathbf{q} : \left(\mathbb{G}^k\right)^{\mathrm{T}}\mathbf{q} = \mathbf{0},$$

y se satisface automáticamente si tomamos $\mathbb{B} = \left(\mathbb{G}^k\right)^{\mathrm{T}}$ en (17). Por otra parte, la condición (i) se puede escribir en forma matricial como

$$\mathbb{H}^k\mathbb{G}^k = \mathbb{X}^k.$$

Sustituyendo (17) en la ecuación anterior, se tiene

$$\left(\mathbb{I} + \mathbb{A}(\mathbb{G}^k)^{\mathrm{T}}\right)\mathbb{G}^k = \mathbb{X}^k,$$

de donde

$$\mathbb{A} = \left(\mathbb{X}^k - \mathbb{I}\mathbb{G}^k\right)\left((\mathbb{G}^k)^{\mathrm{T}}\mathbb{G}^k\right)^{-1}$$

y, por tanto,

$$\mathbb{H}^k = \mathbb{I} + \left(\mathbb{X}^k - \mathbb{G}^k\right)\left((\mathbb{G}^k)^{\mathrm{T}}\mathbb{G}^k\right)^{-1}(\mathbb{G}^k)^{\mathrm{T}}. \tag{18}$$

En la práctica, no obstante, no se calcula la matriz $\mathbb{H}^k$ explícitamente. En su lugar, se sustituye (18) en (12), lo que proporciona

$$\boldsymbol{\gamma}^k := ((\mathbb{G}^k)^{\mathrm{T}}\mathbb{G}^k)^{-1}(\mathbb{G}^k)^{\mathrm{T}}\mathbf{g}^k, \tag{19}$$

$$\mathbf{p}^k := -\mathbf{g}^k - \mathbb{X}^k\boldsymbol{\gamma}^k + \mathbb{G}^k\boldsymbol{\gamma}^k, \tag{20}$$

$$\mathbf{x}^{k+1} := \mathbf{x}^k + \mathbf{p}^k. \tag{21}$$

3.3. Cálculo de $\boldsymbol{\gamma}^k$

La mayor dificultad en la implementación de (19)-(21) está en el cálculo de $\boldsymbol{\gamma}^k$. En realidad, $\boldsymbol{\gamma}^k$ se calcula como la solución al sistema

$$\left((\mathbb{G}^k)^{\mathrm{T}}\mathbb{G}^k\right)\boldsymbol{\gamma}^k = (\mathbb{G}^k)^{\mathrm{T}}\boldsymbol{\gamma}^k.$$

Sin embargo, la matriz $\mathbb{G}^k$ puede ser de rango deficiente, lo que provocaría que la matriz $(\mathbb{G}^k)^{\mathrm{T}}\mathbb{G}^k$ estuviese mal condicionada o fuese singular. Por tanto, para resolver este sistema, se puede recurrir a la descomposición en valores singulares de $\mathbb{G}^k$ (orden `svd` de Matlab). Es decir,

$$\mathbb{G}^k = \mathbb{U}^k\boldsymbol{\Sigma}^{\boldsymbol{k}}(\mathbb{V}^k)^{\mathrm{T}},$$

donde $\mathbb{U}^k \in \mathbb{R}^{N\times m}$ y $\mathbb{V}^k \in \mathbb{R}^{m\times m}$ son matrices ortogonales, y

$$\Sigma^{\boldsymbol{k}} = \begin{bmatrix} \sigma_{11} & & \\ & \ddots & \\ & & \sigma_{mm} \end{bmatrix}$$

con $\sigma_{11} \geq \ldots \geq \sigma_{mm} \geq 0$. Recordemos que $\mathbb{G}^k$ tiene dimensiones $N \times m$. Es fácil demostrar que, si $\Sigma^{\boldsymbol{k}}$ es invertible, entonces

$$\gamma^k = \mathbb{V}^k(\Sigma^{\boldsymbol{k}})^{-1}(\mathbb{U}^k)^T \mathbf{g}^k.$$

En la práctica, es posible que $\Sigma^{\boldsymbol{k}}$ esté mal condicionada, así que se sustituye $(\Sigma^{\boldsymbol{k}})^{-1}$ por su pseudo-inversa

$$\left(\Sigma^{\boldsymbol{k}}\right)^+ := \begin{bmatrix} \sigma_{11}^+ & & \\ & \ddots & \\ & & \sigma_{mm}^+ \end{bmatrix}, \quad \sigma_{ii}^+ := \begin{cases} 1/\sigma_{ii} & \text{si } \sigma_{ii} \geq 10^{-10} \\ 0 & \text{si } \sigma_{ii} < 10^{-10} \end{cases}, \tag{22}$$

de tal forma que

$$\boldsymbol{\gamma}^k = \mathbb{V}^k \left(\Sigma^{\boldsymbol{k}}\right)^+ \left(\mathbb{U}^k\right)^T \mathbf{g}^k.$$

3.4. Comentarios generales sobre el método de Anderson

Como criterio de parada, se tomará el mismo que el del método de Newton–Raphson (expresión (5)), es decir,

$$\left\|\mathbf{p}^k\right\|_2 \leq \sqrt{N}\,\text{Tol}. \tag{23}$$

Nota 7: *La convergencia del método de Anderson es superlineal, es decir, existen $C > 0$ y $q \in (1, 2)$ tales que*

$$\left\|\mathbf{x}^{k+1} - \mathbf{x}^{\text{ex}}\right\| \leq C \left\|\mathbf{x}^k - \mathbf{x}^{\text{ex}}\right\|^q.$$

Nota 8: *El método de Anderson converge más lento que el de Newton–Raphson y, por tanto, necesita un mayor número de iteraciones. Sin embargo, cada iteración es menos costosa, ya que no requiere calcular la matriz jacobiana ni resolver ningún sistema lineal asociado a la misma. Es por ello que este método resulta ser más eficiente en muchos casos.*

Para la implementación del método de Anderson, véase el ejercicio 2.

4. Resolución iterativa de sistemas lineales

En el tema 1, se estudiaron métodos directos para la resolución de sistemas lineales, es decir, métodos que proporcionan la solución exacta (ignorando errores de redondeo) después de un número finito de pasos.

Ahora bien, el método de Anderson visto anteriormente también se puede aplicar a la resolución de un sistema lineal de la forma $\mathbb{A}\mathbf{x} = \mathbf{b}$. Para ello, únicamente hay que notar que dicho sistema se puede escribir de la forma

$$\mathbf{f}(\mathbf{x}) := \mathbb{A}\mathbf{x} - \mathbf{b} = \mathbf{0}. \tag{24}$$

A partir de ahí, se puede construir el residuo precondicionado

$$\mathbf{g}(\mathbf{x}) := \widetilde{\mathbb{A}}^{-1}\,\mathbf{f}(\mathbf{x}) = \widetilde{\mathbb{A}}^{-1}\,(\mathbb{A}\mathbf{x} - \mathbf{b})\,, \tag{25}$$

donde $\widetilde{\mathbb{A}}$ es una matriz tal que

(i) El sistema $\widetilde{\mathbb{A}}\mathbf{g} = \mathbb{A}\mathbf{x} - \mathbf{b}$ es fácil de resolver.
(ii) $\widetilde{\mathbb{A}} \approx \mathbb{A}$, de tal modo que se satisface la condición (C3) $\partial\mathbf{g}(\mathbf{x})/\partial\mathbf{x} \approx \mathbb{I}$.

Por ejemplo, puede tomarse $\widetilde{\mathbb{A}}$ igual a la parte triangular inferior de $\mathbb{A}$, tril(A), o igual a la diagonal, diag($\mathbb{A}$), aunque existen otras opciones.

Nota 9: *Si la matriz $\mathbb{A}$ es grande y dispersa, y se dispone de una buena aproximación inicial $\mathbf{x}^0$, suele ser más rápido hallar la solución del sistema lineal mediante un método iterativo, como podría ser el caso del método de Anderson aplicado a (25), que mediante un método directo. Esto se debe a que no resulta necesario calcular la costosa factorización de la matriz. Véase el ejercicio 24.*

En la práctica habitual, el método de Anderson se aplica para sistemas no lineales. Para sistemas lineales, existe un método específico, que es equivalente al método de Anderson, conocido como GMRES (*Generalized Minimal Residual*). También existen otros métodos, como por ejemplo el del gradiente conjugado, de aplicación si la matriz del sistema es simétrica y definida positiva.

5. Cálculo numérico de la matriz jacobiana

En ocasiones, calcular analíticamente la matriz jacobiana $\mathbb{J} \in \mathbb{R}^{N\times N}$ e implementarla puede ser tedioso. Incluso si esto se ha conseguido, también es conveniente disponer de alguna herramienta alternativa para comprobar los resultados.

Notemos que la matriz jacobiana (2) también puede escribirse como

$$\mathbb{J}(\mathbf{x}) = \left[\; \frac{\partial \mathbf{f}}{\partial x_1}(\mathbf{x}) \;\middle|\; \cdots \;\middle|\; \frac{\partial \mathbf{f}}{\partial x_N}(\mathbf{x}) \;\right].$$

Recordando que la derivada parcial se define como

$$\frac{\partial \mathbf{f}}{\partial x_j}(\mathbf{x}) = \lim_{\delta\to 0} \frac{\mathbf{f}(\mathbf{x} + h\mathbf{e}_j) - \mathbf{f}(\mathbf{x})}{h},$$

siendo $\mathbf{e}_j$ un vector unitario en la dirección del eje cartesiano x_j, es posible aproximar la misma mediante la expresión

$$\frac{\partial \mathbf{f}}{\partial x_j}(\mathbf{x}) \approx \frac{\mathbf{f}(\mathbf{x} + \delta\mathbf{e}_j) - \mathbf{f}(\mathbf{x})}{\delta}, \tag{26}$$

siendo δ un número pequeño para que la aproximación sea precisa, pero no tan pequeño como para causar errores numéricos debidos a la división por cero. Típicamente, se suele tomar $\delta = 10^{-6}$. En el tema siguiente se estudiará cómo de precisa es la aproximación anterior.

Así pues, la matriz jacobiana puede aproximarse numéricamente aplicando (26) para $j = 1, \ldots, N$. No obstante, es importante señalar que este procedimiento es útil para matrices llenas y no demasiado grandes. Para matrices dispersas y grandes, el procedimiento sería prohibitivamente costoso, y habría que utilizar algoritmos especializados que tienen en cuenta el patrón de elementos no nulos de la matriz.

Véase el ejercicio 26.

6. Problemas

6.1. Implementación del método de Newton–Raphson

El método de Newton–Raphson se puede implementar de acuerdo al algoritmo 1.

Algoritmo 1: Método de Newton–Raphson

Datos: Función $\mathbf{f}(\mathbf{x})$, aproximación inicial $\mathbf{x}^0 \in \mathbb{R}^N$, tolerancia Tol, máximo iteraciones $k^{\text{máx}}$.
Resultado: Solución $\mathbf{x}^k$, número de iteraciones k, indicador de convergencia `flag`.
comprobar que $\mathbf{x}^0$ es un vector columna; si no lo es, **lanzar** un error,
inicializar $k = 0$, $\mathbf{x}^k = \mathbf{x}^0$,
while *true* **do**
    evaluar $\mathbf{f}^k := \mathbf{f}(\mathbf{x}^k)$, $\mathbb{J}^k = \mathbb{J}(\mathbf{x}^k)$,
    calcular $\mathbf{p}^k := -(\mathbb{J}^k)^{-1}\mathbf{f}^k$,
    if $\left\|\mathbf{p}^k\right\|_2 \leq \sqrt{N}\text{Tol}$ **then**
        flag = 1
        salir del bucle (se ha encontrado la solución)
    else if $k = k^{\text{máx}}$ **then**
        flag = −1
        salir del bucle (se ha alcanzado el máximo de iteraciones)
    end
    actualizar variables $k \longleftarrow k + 1$, $\mathbf{x}^k \longleftarrow \mathbf{x}^k + \mathbf{p}^k$,
end

En lo que sigue, supondremos que se dispone de una función de la forma `[f,J] = fun(x, CalcJ)` para calcular el residuo $\mathbf{f}(\mathbf{x})$ y la matriz jacobiana $\mathbb{J}(\mathbf{x})$. En particular, si `CalcJ==true`, se calculan ambas variables, mientras que, si `CalcJ==false`, se calcula sólo el residuo $\mathbf{f}(\mathbf{x})$, mientras que la salida `J` puede ser simplemente `NaN`. De este modo, ahorramos el costoso cálculo de la matriz jacobiana en las ocasiones en los que ésta no sea necesaria.

Ejercicio 1: *Crear una función de la forma `[x_k, nIters, flag] = NewtonRaphson1(fun, x0, Tol, MaxIter)` que implemente el algoritmo 1. Para ello, puede utilizarse el archivo `Newton-Raphson1_incompleto.m`.*

6.2. Implementación del método de Anderson

El método de Anderson se puede implementar de acuerdo al algoritmo 2.

En lo que sigue, se supone que el residuo precondicionado $\mathbf{g}(\mathbf{x})$ se calcula a través de una función de la forma `g = gfun(x)`.

Ejercicio 2: *Crear una función de la forma `[x_k, nIters, flag] = Anderson1(gfun, x0, Tol, MaxIter, M)` que implemente el algoritmo 2. Para ello, puede utilizarse el archivo `Anderson1_incompleto.m`.*

Algoritmo 2: Método de Anderson

Datos: *Función* $\mathbf{g}(\mathbf{x})$, *condición inicial* $\mathbf{x}^0 \in \mathbb{R}^N$, *tolerancia* Tol, *máximo iteraciones* $k^{\text{máx}}$, *número de pasos almacenados* M.
Resultado: *Solución* $\mathbf{x}^k$, *número de iteraciones* k, *indicador de convergencia* flag.
comprobar que $\mathbf{x}^0$ es un vector columna; si no lo es, **lanzar** un error,
comprobar que $M \leq N$; si no lo es, tomar $M = N$,
inicializar $k = 0$, $\mathbf{x}^k = \mathbf{x}^0$, $\mathbf{g}^k = \mathbf{g}(\mathbf{x}^k)$, $\mathbf{p}^k = -\mathbf{g}^k$, $\mathbb{X}^k = \mathbb{O}_{N\times M}$, $\mathbb{G}^k = \mathbb{O}_{N\times M}$,
while *true* **do**
    if $\left\|\mathbf{p}^k\right\|_2 \leq \sqrt{N}\text{Tol}$ **then**
        flag $= 1$
        salir del bucle (se ha encontrado la solución)
    else if $k = k^{\text{máx}}$ **then**
        flag $= -1$
        salir del bucle (se ha alcanzado el máximo de iteraciones)
    end
    actualizar variables $\mathbf{x}^{k-1} \longleftarrow \mathbf{x}^k$, $\mathbf{g}^{k-1} \longleftarrow \mathbf{g}^k$,
    actualizar variables $k \longleftarrow k+1$, , $\mathbf{x}^k \longleftarrow \mathbf{x}^k + \mathbf{p}^k$, $\mathbf{g}^k \longleftarrow \mathbf{g}(\mathbf{x}^k)$,
    actualizar matrices $\mathbb{X}^k$, $\mathbb{G}^k$,
    calcular factorización en valores singulares $\mathbb{G}^k = \mathbb{U}^k\boldsymbol{\Sigma}^{\boldsymbol{k}}(\mathbb{V}^k)^{\mathrm{T}}$ (orden `svd` de Matlab),
    calcular pseudoinversa $\left(\boldsymbol{\Sigma}^{\boldsymbol{k}}\right)^+$ de acuerdo a (22),
    calcular $\boldsymbol{\gamma}^k := \mathbb{V}^k\left(\boldsymbol{\Sigma}^{\boldsymbol{k}}\right)^+(\mathbb{U}^k)^{\mathrm{T}}\mathbf{g}^k$
    calcular $\mathbf{p}^k := -\mathbf{g}^k - \mathbb{X}^k\boldsymbol{\gamma}^k + \mathbb{G}^k\boldsymbol{\gamma}^k$
end

6.3. Sistema polinomial

Consideremos el sistema de ecuaciones dado por

$$f_i(\mathbf{x}) := \left(\sum_{j=1}^{N} x_j^2 + i\right)\left(x_i - \cos\left(\frac{2\pi i}{N}\right)\right) = 0, \quad i = 1,\dots,N, \tag{27}$$

y la aproximación inicial $\mathbf{x}^0$ dada por $x_i^0 = 1$, $i = 1,\dots,N$.

Ejercicio 3: *¿Cuál es la solución exacta,* x_i^{ex}*, al sistema* (27)*?*

Solución: Notemos que el primer término entre paréntesis es siempre positivo, por lo que la solución exacta es $x_i^{\text{ex}} = \cos(2\pi i/N)$. Aunque éste sea un sistema muy fácil de resolver, por motivos didácticos se estudiará de todos modos. □

Ejercicio 4: *Demostrar que la matriz jacobiana asociada a* (27) *es*

$$J_{ij}(\mathbf{x}) := \frac{\partial f_i}{\partial x_j}(\mathbf{x}) = 2x_j\left(x_i - \cos\left(\frac{2\pi i}{N}\right)\right) + \left(\sum_{k=1}^{N} x_k^2 + i\right)\delta_{ij}, \quad i,j = 1,\dots,N, \tag{28}$$

donde

$$\delta_{ij} := \begin{cases} 1, & i = j, \\ 0, & i \neq j, \end{cases} \tag{29}$$

es la delta de Kronecker.

Solución: En primer lugar, si vamos a derivar respecto de x_j, para evitar complicaciones es conveniente comprobar que el índice j no aparece en la expresión a derivar. Puesto que no es así, reescribimos f_i como

$$f_i(\mathbf{x}) := \left(\sum_{k=1}^{N} x_k^2 + i\right)\left(x_i - \cos\left(\frac{2\pi i}{N}\right)\right), \quad i = 1, \dots, N.$$

Aplicando ahora la regla del producto y la regla de la cadena, se tiene

$$\frac{\partial f_i}{\partial x_j}(\mathbf{x}) = \left(\sum_{k=1}^{N} 2x_k \frac{\partial x_k}{\partial x_j}\right)\left(x_i - \cos\left(\frac{2\pi i}{N}\right)\right) + \left(\sum_{k=1}^{N} x_k^2 + i\right)\frac{\partial x_i}{\partial x_j}.$$

Puesto que $x_1, \dots, x_N$, son variables independientes entre sí, se tiene $\partial x_i / \partial x_j = \delta_{ij}$. Por otra parte, por la definición de δ_{ij},

$$\sum_{k=1}^{N} 2x_k \delta_{kj} = 2x_j,$$

puesto que el único término no nulo en el sumatorio es aquél que corresponde a $k = j$. De esta forma, llegamos a la expresión (28). □

Ejercicio 5: *Implementar una función* `[f,J] = Polin_residuo(x,CalcJ)` *que calcule el residuo* $\mathbf{f}$ *dado por* (27). *Además, si* `CalcJ=true`, *debe devolver la matriz jacobiana determinada por* (28), *de lo contrario, debe ignorar el cálculo de la misma y devolver simplemente* `J=NaN`. *Puede utilizarse el archivo* `Polin_residuo_incompleto`.

Ejercicio 6: *Implementar una función* `Polin_Newton(N)` *que, recibido* N, *ejecute las siguientes tareas:*

- *Hallar la solución del sistema* (27) *utilizando el método de Newton–Raphson programado en el ejercicio 1. Utilizar una tolerancia* $\mathrm{Tol} = 10^{-6}$ *y un máximo de 100 iteraciones.*
- *Dibujar una gráfica con los valores de la solución numérica,* x_i^*, *frente a* i, *con línea continua y cruces azules (opción* `'+-b'` *en* `plot`). *Sobre esa misma gráfica, hacer lo mismo con la solución exacta* x_i^{ex}.
- *Mostrar por pantalla el valor de* N, *el tiempo empleado en la resolución por el método de Newton–Raphson, el número de iteraciones empleadas, y el error* $\|\mathbf{x}^* - \mathbf{x}^{\mathrm{ex}}\|_\infty$.

Puede utilizarse el archivo `Polin_Newton_incompleto`.

Se desea ahora resolver el sistema anterior con el método de Anderson. Para ello, se considera la ecuación precondicionada

$$\mathbf{g}(\mathbf{x}) := \mathbb{J}^{-1}(\mathbf{x}^0)\mathbf{f}(\mathbf{x}) = \mathbf{0}. \tag{30}$$

Por supuesto, como ya se explicó en el tema anterior, $\mathbb{J}^{-1}(\mathbf{x}^0)$ no es la matriz inversa de $\mathbb{J}(\mathbf{x}^0)$, sino un operador cuyo efecto es el mismo. Es decir, en la práctica, $\mathbf{g}(\mathbf{x})$ se obtiene resolviendo el sistema $\mathbb{J}(\mathbf{x}^0)\mathbf{g}(\mathbf{x}) = \mathbf{f}(\mathbf{x})$ *de la forma más adecuada.* Notemos, además, que la matriz de dicho sistema, $\mathbb{J}(\mathbf{x}^0)$, es independiente de $\mathbf{x}$, y por tanto es siempre la misma.

Ejercicio 7: *¿Cómo se ha de implementar el cálculo del residuo condicionado* $\mathbf{g}(\mathbf{x})$ *para que éste sea lo más eficiente posible?*

Solución: Puesto que la matriz del sistema, $\mathbb{J}(\mathbf{x}^0)$, es siempre la misma, podemos calcular primero una factorización de la misma ($\mathcal{O}\left(N^3\right)$ operaciones), y reutilizar ésta cada vez que haya que calcular $\mathbf{g}(\mathbf{x})$ ($\mathcal{O}\left(N^2\right)$ operaciones).

En particular, notemos primero que $\mathbb{J}(\mathbf{x}^0)$ es una matriz llena y no simétrica. Por tanto, la factorización más adecuada de la misma es la LU con pivotaje parcial, es decir,

$$\mathbb{P}\mathbb{J}(\mathbf{x}^0) = \mathbb{L}\mathbb{U}.$$

En ese caso, el sistema $\mathbb{J}(\mathbf{x}^0)\mathbf{g}(\mathbf{x}) = \mathbf{f}(\mathbf{x})$ se puede reescribir como $\mathbb{L}\mathbb{U}\mathbf{g}(\mathbf{x}) = \mathbb{P}\mathbf{f}(\mathbf{x})$ y, por tanto, el residuo precondicionado se reescribe como

$$\mathbf{g}(\mathbf{x}) = \mathbb{U}^{-1}\mathbb{L}^{-1}\mathbb{P}\mathbf{f}(\mathbf{x}). \tag{31}$$

Véase la sección 2.3 del tema 1 para la implementación numérica de (31). □

Ejercicio 8: *Implementar una función* `Polin_Anderson(N)` *que, recibido* N*, ejecute las siguientes tareas:*

- *Definir una función que calcule el residuo* $\mathbf{g}(\mathbf{x})$ *de la forma más adecuada.*
- *Hallar la solución del sistema* $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ *utilizando el método de Anderson programado en el ejercicio 2. Utilizar una tolerancia* $\mathrm{Tol} = 10^{-6}$*, un máximo de 100 iteraciones y* $M = 20$ *condiciones secantes.*
- *Dibujar una gráfica con los valores de la solución numérica,* x_i^**, frente a* i*, con línea continua y cruces azules (opción* `'+-b'` *en* `plot`*). Sobre esa misma gráfica, hacer lo mismo con la solución exacta* x_i^{ex}*.*
- *Mostrar por pantalla el valor de* N*, el tiempo empleado en la resolución por el método de Anderson, el número de iteraciones empleadas, y el error* $\|\mathbf{x}^* - \mathbf{x}^{\mathrm{ex}}\|_\infty$*.*

Puede utilizarse el archivo `Polin_Anderson_incompleto`*.*

Ejercicio 9: *Ejecutar* `Polin_Newton(N)` *y* `Polin_Anderson(N)` *para varios valores de* N*. Comenzar con* $N = 100$ *y duplicar su valor hasta que ambos necesiten más de dos segundos de ejecución. ¿Qué método necesita menos iteraciones? ¿Qué método es más eficiente? Justificar los resultados.*

6.4. Ecuación-H de Chandrasekhar

La función $H : [0,1] \longrightarrow \mathbb{R}$ de Chandrasekhar aparece con frecuencia en el estudio de problemas de absorción y reflexión de luz en estrellas. Una forma de determinar $H(x)$ de forma numérica consiste en dar un mallado para la variable $x \in [0,1]$, por ejemplo,

$$x_i := \frac{i - 1/2}{N}, \quad i = 1, \ldots, N,$$

y resolver el sistema de ecuaciones

$$f_i(H_1, \ldots, H_N) := H_i - \left(1 - \frac{c}{2N}\sum_{j=1}^{N}\frac{x_i H_j}{x_i + x_j}\right)^{-1} = 0, \quad i = 1, \ldots, N, \tag{32}$$

donde $c = 0.9$, N se supone conocido y $H_i \approx H(x_i)$. (La aproximación es más precisa cuanto mayor es N.) Notemos que ahora $x_1, \ldots, x_N$ son valores conocidos, mientras que $H_1, \ldots, H_N$ son incógnitas. Como aproximación inicial para los resolvedores iterativos, considérese $H_1^0 = \ldots = H_N^0 = 1$.

Ejercicio 10: *Demostrar que la matriz jacobiana asociada a (32) viene dada por*

$$J_{ij} := \frac{\partial f_i}{\partial H_j} = \delta_{ij} - \left(1 - \frac{c}{2N}\sum_{k=1}^{N}\frac{x_i H_k}{x_i + x_k}\right)^{-2} \frac{c}{2N}\frac{x_i}{x_i + x_j}. \tag{33}$$

Ejercicio 11: *Implementar una función* `[f,J] = Chandrasekhar_residuo(H, CalcJ)` *que calcule el residuo* $\mathbf{f}$ *dado por (32). Además, si* `CalcJ=true`*, debe devolver la matriz jacobiana determinada por (33), de lo contrario, debe ignorar el cálculo de la misma y devolver simplemente* `J=NaN`*.*

Ejercicio 12: *Implementar una función* `Chandrasekhar_Newton(N)` *que, recibido* N*, ejecute las siguientes tareas:*

- *Hallar la solución del sistema (32) utilizando el método de Newton–Raphson programado en el ejercicio 1. Utilícese una tolerancia de* 10^{-8} *y un máximo de 100 iteraciones.*
- *Representar la aproximación obtenida para* $H(x)$ *en una gráfica. Es decir, representar* H_i *frente a* x_i*.*
- *Mostrar por pantalla el valor de* N*, el tiempo empleado en la resolución por el método de Newton–Raphson, y el número de iteraciones empleadas.*

Ejercicio 13: *Implementar una función* `Chandrasekhar_Anderson(N)` *que, recibido* N*, ejecute las siguientes tareas:*

- *Definir una función que calcule el residuo* $\mathbf{g}(\mathbf{x})$*, definido de la misma forma que en (7), de la forma más adecuada.*
- *Hallar la solución del sistema* $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ *utilizando el método de Anderson programado en el ejercicio 2. Utilizar una tolerancia de* 10^{-8}*, un máximo de 100 iteraciones y 20 condiciones secantes.*
- *Representar la aproximación obtenida para* $H(x)$ *en una gráfica. Es decir, representar* H_i *frente a* x_i*.*
- *Mostrar por pantalla el valor de* N*, el tiempo empleado en la resolución por el método de Anderson, y el número de iteraciones empleadas.*

Ejercicio 14: *Comparar el rendimiento de los métodos de Newton–Raphson y Anderson, de forma similar al ejercicio 9.*

6.5. Sistema de Broyden con bandas

El *sistema de Broyden con bandas* viene dado por las ecuaciones

$$f_i(\mathbf{x}) := (2 + 5x_i^2)x_i + 1 + \sum_{k=k_{i,1}}^{k_{i,2}} x_k(1 + x_k) = 0, \quad i = 1, \dots, N, \tag{34}$$

siendo

$$k_{i,1} := \max\{1, i - 5\}, \qquad k_{i,2} := \min\{N, i + 1\}.$$

Como aproximación inicial para la resolución por métodos iterativos, tómese $x_i^0 = -1/2$, $i = 1, \dots, N$.

Ejercicio 15: *Demostrar que la matriz jacobiana del sistema viene dada por*

$$J_{ij} := \frac{\partial f_i}{\partial x_j} = \begin{cases} \left(2 + 15x_i^2\right)\delta_{ij} + 1 + 2x_j, & k_{i1} \le j \le k_{i2}, \\ 0, & \text{en caso contrario.} \end{cases} \tag{35}$$

Recordemos que δ_{ij} *es la delta de Kronecker, definida por la expresión (29).*

Ejercicio 16: *Implementar una función* `[f,J] = BroydenBandas_residuo(x, CalcJ)` *que calcule el residuo* $\mathbf{f}$ *dado por* (34)*. Además, si* `CalcJ=true`*, debe devolver la matriz jacobiana determinada por* (35)*, de lo contrario, debe ignorar el cálculo de la misma y devolver simplemente* `J=NaN`*.*

Ejercicio 17: *Implementar una función* `BroydenBandas_Newton(N)` *que, recibido* N*, ejecute las siguientes tareas:*

- *Hallar la solución del sistema* (34) *utilizando el método de Newton–Raphson programado en el ejercicio 1. Utilícese una tolerancia de* 10^{-8} *y un máximo de 100 iteraciones.*
- *Representar la solución numérica* x_i^* *frente a* i*.*
- *Mostrar por pantalla el valor de* N*, el tiempo empleado en la resolución por el método de Newton–Raphson, y el número de iteraciones empleadas.*

Ejercicio 18: *Implementar una función* `BroydenBandas_Anderson(N)` *que, recibido* N*, ejecute las siguientes tareas:*

- *Definir una función que calcule el residuo* $\mathbf{g}(\mathbf{x})$*, definido de la misma forma que en* (7)*, de la forma más adecuada.*
- *Hallar la solución del sistema* $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ *utilizando el método de Anderson programado en el ejercicio 2. Utilizar una tolerancia de* 10^{-8}*, un máximo de 100 iteraciones y 20 condiciones secantes.*
- *Representar la solución numérica* x_i^* *frente a* i*.*
- *Mostrar por pantalla el valor de* N*, el tiempo empleado en la resolución por el método de Anderson, y el número de iteraciones empleadas.*

Ejercicio 19: *Implementar una función* `g = BroydenBandas_residuo2(x)` *que calcule el residuo* $\mathbf{g}(\mathbf{x})$ *dado por las expresiones* (8) *y* (34)*.*

Ejercicio 20: *Implementar una función* `BroydenBandas_Anderson2(N)` *que, recibido* N*, ejecute las siguientes tareas:*

- *Hallar la solución del sistema* $\mathbf{g}(\mathbf{x}) = \mathbf{0}$*, donde* $\mathbf{g}(\mathbf{x})$ *está definido por las expresiones* (8) *y* (34)*, utilizando el método de Anderson programado en el ejercicio 2. Utilizar una tolerancia de* 10^{-8}*, un máximo de 100 iteraciones y 20 condiciones secantes.*
- *Representar la solución numérica* x_i^* *frente a* i*.*
- *Mostrar por pantalla el valor de* N*, el tiempo empleado en la resolución por el método de Anderson, y el número de iteraciones empleadas.*

Ejercicio 21: *Comparar, para varios valores de* N*, los tiempos de cálculo empleados por los métodos implementados en los ejercicios 17, 18 y 20. Justificar los resultados.*

6.6. Fuerza sobre dos muelles

La figura 1 muestra un sistema elástico formado por dos muelles de longitud natural L y rigideces k_1 y k_2. Los dos muelles tienen un extremo en común, mientras que los otros dos están anclados y separados una distancia L. Inicialmente, la longitud de cada muelles coincide con su longitud natural. Entonces, se aplica una fuerza vertical F hacia arriba que hace que el sistema se deforme.

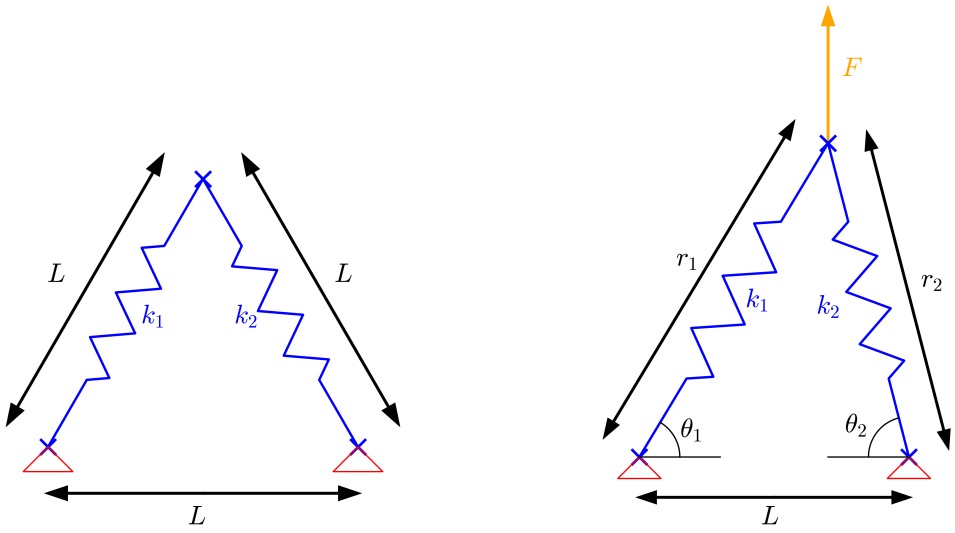

Figura 1: Sistema elástico formado por dos muelles sobre los que actúa una fuerza. (izq-da.) Configuración inicial. (dcha.) Configuración deformada.

Las ecuaciones que gobiernan la configuración deformada son

$$\begin{aligned} f_1(\mathbf{x}) &:= k_1(r_1 - L)\cos\theta_1 - k_2(r_2 - L)\cos\theta_2 = 0, \\ f_2(\mathbf{x}) &:= k_1(r_1 - L)\sin\theta_1 + k_2(r_2 - L)\sin\theta_2 - F = 0, \\ f_3(\mathbf{x}) &:= r_1\sin\theta_1 - r_2\sin\theta_2 = 0, \\ f_4(\mathbf{x}) &:= r_1\cos\theta_1 + r_2\cos\theta_2 - L = 0, \end{aligned}$$

donde $\mathbf{x} = [r_1, r_2, \theta_1, \theta_2]^{\mathrm{T}}$ representa el vector de incógnitas.

Ejercicio 22: *Implementar una función* `g=Muelles_residuo(x, L, k1, k2, F)` *que reciba el vector* $\mathbf{x}$*, así como los parámetros* L*,* k_1*,* k_2 *y* F*, y que devuelva el residuo precondicionado* $\mathbf{g}(\mathbf{x})$ *definido por la expresión* (8)*.*

Ejercicio 23: *Implementar una función* `Muelles_Anderson(L, k1, k2, F)` *que halle la configuración deformada haciendo uso del método de Anderson. Tómese una tolerancia* $\mathrm{Tol} = 10^{-8}$*,* $M = 4$ *condiciones secantes y la aproximación inicial* $\mathbf{x}^0$ *que se estime oportuna. La función debe representar las configuraciones inicial y deformada en una gráfica de forma esquemática (por ejemplo, los muelles pueden representarse por segmentos rectos). Probar esta función para* $L = 1$*,* $k_1 = 1.2$*,* $k_2 = 2.4$ *y* $F = 1.5$*.*

6.7. Resolución iterativa de un sistema lineal

Sea $\mathbb{A} = \{A_{ij}\}$ la matriz de la sección 6.6 del tema 1, es decir, una matriz de dimensiones $N \times N$ cuyos elementos no nulos son $A_{11} = A_{NN} = 1$ y

$$A_{i,i-\Delta_i} = -3, \quad A_{ii} = 7, \quad A_{i,i+\Delta_i} = -4, \quad i = 2, \ldots, N-1,$$

donde $\Delta_i := \lceil (N-i)(i-1)/(N-1) \rceil$.

Ejercicio 24: *Implementar una función* `SistemaLineal_Anderson(N)` *que, recibido* N*, realice las siguientes tareas:*

- *Crear la matriz* $\mathbb{A}$.
- *Crear un vector* $\mathbf{x}^{\text{ex}}$ *aleatorio y definir* $\mathbf{b} := \mathbb{A}\mathbf{x}^{\text{ex}}$.
- *Resolver el sistema* $\mathbb{A}\mathbf{x} = \mathbf{b}$ *mediante un método directo (comando* \ *de Matlab). Mostrar por pantalla el tiempo de cálculo y la norma del error* $\|\mathbf{x}^* - \mathbf{x}^{\text{ex}}\|_\infty$, *siendo* $\mathbf{x}^*$ *la solución numérica.*
- *Considerar el residuo precondicionado* $\mathbf{g}(\mathbf{x}) = \mathbb{D}^{-1}(\mathbb{A}\mathbf{x} - \mathbf{b})$, *siendo* $\mathbb{D} = \text{diag}(A)$ *la diagonal de* $\mathbb{A}$ *(puede conseguirse mediante la orden* `diag(diag(A))`*). Resolver mediante el método de' Anderson con* $\text{Tol} = 10^{-8}$, *un máximo de 100 iteraciones y* $M = 10$ *condiciones secantes. Como aproximación inicial, tómese el vector nulo* $\mathbf{x}^0 = \mathbf{0}$. *Mostrar por pantalla el número de iteraciones, el tiempo de cálculo y la norma del error* $\|\mathbf{x}^* - \mathbf{x}^{\text{ex}}\|_\infty$.
- *Repetir el paso anterior considerando esta vez* $\mathbf{g}(\mathbf{x}) = \mathbb{A}_L^{-1}(\mathbb{A}\mathbf{x}^* - \mathbf{b})$, *donde* $\mathbb{A}_L$ *es el bloque triangular inferior de* $\mathbb{A}$ *(utilizar la orden* `tril(A)`*).*

Puede utilizarse el archivo `SistemaLineal_Anderson_incompleto.m`

Ejercicio 25: *Probar la función con valores cada vez mayores de* N *muy grandes, hasta que el tiempo requerido por el método directo sea del orden de 10 segundos. ¿Cuánto tardan los métodos iterativos?*

6.8. Cálculo numérico de la matriz jacobiana

Supongamos que se dispone de una función de la forma `f = fun(x)` que para calcular $\mathbf{f}(\mathbf{x})$.

Ejercicio 26: *Implementar una función* `J=JacobianEst(fun, x, delta)` *que, recibida la función* `fun`, *el valor de* $\mathbf{x}$ *y el valor de* δ, *calcule la matriz jacobiana utilizando la aproximación* (26).

Bibliografía

[1] Walker, H.F.; Ni, P., "Anderson Acceleration for Fixed-Point Iterations," *SIAM Journal on Numerical Analysis*, vol. 49(4):pp. 1715–1735, 2011, URL http://dx.doi.org/10.1137/10078356X.

[2] Fang, H.r.; Saad, Y., "Two classes of multisecant methods for nonlinear acceleration," *Numerical Linear Algebra with Applications*, vol. 16(3):pp. 197–221, 2009, URL http://dx.doi.org/10.1002/nla.617.

[3] Lukšan, L., "Inexact trust region method for large sparse systems of nonlinear equations," *Journal of Optimization Theory and Applications*, vol. 81(3):pp. 569–590, 1994, URL http://dx.doi.org/10.1007/BF02193101.

[4] Leong, W.J.; Hassan, M.A.; Yusuf, M.W., "A matrix-free quasi-Newton method for solving large-scale nonlinear systems," *Computers & Mathematics with Applications*, vol. 62(5):pp. 2354–2363, 2011, URL http://dx.doi.org/10.1016/j.camwa.2011.07.023.

[5] Kelley, C.T., *Iterative Methods for Linear and Nonlinear Equations*, Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, 1995, URL http://dx.doi.org/10.1137/1.9781611970944.