Universidad de Huelva

Departamento de Matemáticas

Análisis Numérico I

Ingeniería Técnica Informática.

Curso académico 2008/09

Tema 2

Ecuaciones y sistemas de ecuaciones

RESUMEN TEÓRICO

En este tema estudiaremos uno de los problemas básicos de la aproximación numérica: el problema de la búsqueda de las soluciones de una ecuación o de un sistema de ecuaciones. Los métodos numéricos que se dan en este capítulo sirven para obtener aproximaciones a las soluciones cuando no sea posible obtener soluciones exactas con métodos algebraicos.

2.1 Resolución de ecuaciones no lineales

Algoritmo de bisección
Sea $f : [a, b] \longmapsto \mathbb{R}$ continua y $f(a)f(b) < 0$, entonces:
(1) Sea $I_1 = [a, b]$.
(2) Para $n \geq 1$ (Construcción de I_{n+1} a partir de I_n).
Sea $I_n = [a_n, b_n]$, $p_n = \frac{a_n+b_n}{2}$.
Si $f(p_n) = 0$ o bien $\frac{|b_n-a_n|}{2} < TOL$ entonces $p = p_n$ y PARAR.
Si $f(p_n)f(a_n) < 0$ entonces $a_{n+1} = a_n$ y $b_{n+1} = p_n$, en caso contrario $a_{n+1} = p_n$ y $b_{n+1} = b_n$.

Teorema
Sea $f : [a, b] \longmapsto \mathbb{R}$ *continua y* $f(a)f(b) < 0$. *Entonces el algoritmo de bisección genera una sucesión* $\{p_n\}_{n\geq 1}$ *que tiende a* p *con* $f(p) = 0$ *y además* $|p_n - p| \leq \frac{b-a}{2^n}$, *para todo* $n \geq 1$.

Algoritmo de la regula falsi
Sea $f : [a, b] \longmapsto \mathbb{R}$ continua y $f(a)f(b) < 0$, entonces:
(1) Sea $I_1 = [a, b]$.
(2) Para $n \geq 1$.
Sea $I_n = [a_n, b_n]$, $p_n = \frac{a_n f(b_n)-b_n f(a_n)}{f(b_n)-f(a_n)}$.
Si $f(p_n) = 0$ o bien $|p_n - p_{n-1}| < TOL$ entonces $p = p_n$ y PARAR.
Si $f(p_n)f(a_n) < 0$ entonces $a_{n+1} = a_n$ y $b_{n+1} = p_n$, en caso contrario $a_{n+1} = p_n$ y $b_{n+1} = b_n$.

Los dos algoritmos anteriores se utilizan para encontrar buenas aproximaciones para métodos de convergencia más rápida como Newton-Raphson y Steffensen. Ambos métodos utilizan la técnica conocida como iteración de punto fijo, que está basada en el teorema del punto fijo.

Teorema (del punto fijo)
Sea $g : [a,b] \longmapsto [a,b]$ continua y derivable, tal que existe una constante $K \in (0,1)$ con $|g'(x)| \leq K$, para cualquier $x \in (a,b)$. Entonces g tiene un único punto fijo $p \in [a,b]$ tal que el algoritmo:

$$(PF) \qquad \begin{cases} p_0 \in [a,b], \\ p_n = g(p_{n-1}), \ n \geq 1, \end{cases}$$

converge a p y además se cumple:

$$|p_n - p| \leq \frac{K^n}{1-K}|p_1 - p_0|, \qquad n \geq 1.$$

Definición (orden de una raíz)
Supongamos que $f \in C^m[p-\delta, p+\delta]$, $m \geq 1$. Diremos que $f(x) = 0$ tiene una raíz de es de orden m en $x = p$ si $f(p) = 0, f'(p) = 0, \cdots, f^{(m-1)}(p) = 0$, y $f^{(m)}(p) \neq 0$. Si $m = 1$ la raíz se llama raíz simple, mientras que si $m > 1$ la raíz se denomina raíz múltiple.

Definición (orden de convergencia)
Supongamos que $\{p_n\}$ es una sucesión que converge a p. Si existen dos constantes positivas λ y α tales que

$$lim_{n \to +\infty} \frac{|p_{n+1} - p|}{|p_n - p|^\alpha} = \lambda,$$

diremos que la sucesión $\{p_n\}$ converge a p con orden de convergencia α y constante asintótica de error λ.

Teorema (convergencia del algoritmo del punto fijo)
Sea p un punto fijo de g, $p = g(p)$. Supongamos que g es de clase m en un entorno de p, tal que $g'(p) = g''(p) = \cdots = g^{(m-1)}(p) = 0$ y $g^{(m)}(p) \neq 0$. Entonces existe un $\delta > 0$ tal que , para $p_0 \in [p-\delta, p+\delta]$, el algoritmo de punto fijo para g converge a p a orden m y con error asintótico $\frac{|g^{m)}(p)|}{m!}$.

Algoritmo de Newton-Raphson

$$(NR) \qquad \begin{cases} p_0 \approx p, \\ p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, \ n \geq 1, \end{cases}$$

Teorema (convergencia local de Newton-Raphson)
Sea $f \in C^2[a,b]$. Si $p \in [a,b]$ es tal que $f(p) = 0$ y $f'(p) \neq 0$, entonces existe $\delta > 0$ tal que el método de Newton-Raphson (NR) converge cuadráticamente a p para cualquier $p_0 \in [p-\delta, p+\delta]$. Si p es una raíz múltiple de orden $m > 1$, entonces la convergencia es lineal y $\lambda = \frac{m-1}{m}$ es la constante asintótica de error .

Teorema (convergencia global de Newton-Raphson)
Sea $f \in C^2[a,b]$, cumpliendo:
(i) $f(a)f(b) < 0$,
(ii) $f'(x) \neq 0$, para todo $x \in [a,b]$,
(iii) $f''(x)$ es o bien ≥ 0 o bien ≤ 0 para todo $x \in [a,b]$,
(iv) Si c denota el extremo de $[a,b]$ en el que $|f'(x)|$ es más pequeño, entonces $\frac{|f(c)|}{|f'(c)|} \leq b - a$.
Entonces el método de Newton-Raphson converge a la única solución p de $f(x) = 0$ en $[a,b]$, para cualquiera que sea $p_0 \in [a,b]$.

El problema, a veces grave, de la necesidad de evaluar $f'(x)$ en cada iteración se puede soslayar mediante el algoritmo de la secante, que tiene orden de convergencia $\alpha = \frac{1+\sqrt{5}}{2} \approx 1.618$ y constante asintótica de error $\lambda = (|\frac{f''(p)}{2f'(p)}|)^{0.618}$.

Algoritmo de la secante

$$(Sec) \qquad \begin{cases} p_0, p_1 \ \ datos, \\ p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1}-p_{n-2})}{f(p_{n-1})-f(p_{n-2})}, \ n \geq 2. \end{cases}$$

En el teorema de convergencia local de Newton-Raphson vimos que este converge lentamente si la raíz es múltiple, ya que la sucesión de aproximaciones presenta convergencia lineal. El siguiente algoritmo proporciona un método para acelerar la convergencia de cualquier sucesión que sea linealmente convergente.

Algoritmo Δ^2 de Aitken
Sea $\{p_n\}$ una sucesión que converge linealmente a p. Entonces la sucesión $p'_n = p_n - \frac{(p_{n+1}-p_n)^2}{p_{n+2}-2p_{n+1}+p_n}$, converge superlinealmente a p ($lim_{n\to+\infty}\frac{p'_n-p}{p_n-p} = 0$).

Cuando el método de Aitken se combina con una iteración de punto fijo, el resultado se conoce como algoritmo de Steffensen.

Algoritmo de Steffensen
Sea g una función que define un método del punto fijo de orden 1.

$$(Stff) \qquad \begin{cases} p_0 \approx p, \\ x_n = g(p_n), \ \ y_n = g(x_n), \\ p_{n+1} = p_n - \frac{(x_n-p_n)^2}{y_n-2x_n+p_n}, \ \ n \geq 0, \end{cases}$$

o equivalentemente:

$$(Stff) \qquad \begin{cases} p_0 \approx p, \\ p_{n+1} = G(p_n) = \frac{p_n g(g(p_n))-g(p_n)^2}{g(g(p_n))-2g(p_n)+p_n}, \ \ n \geq 0. \end{cases}$$

Teorema (convergencia local del método de Steffensen)
Sea g una función de iteración de clase $C^3[a,b]$ que tiene un punto fijo p. Si $g'(p) \neq 1$, entonces la función G del algoritmo de Steffensen determina localmente un método de orden 2 para calcular p.

Teorema (convergencia global del método de Steffensen)
Sea $I = (a, \infty)$ y $g \in C^2(I)$, cumpliendo:
(i) $g(x) > a$, para todo $x \in I$,
(ii) $g'(x) < 0$, para todo $x \in I$,
(iii) $g''(x) > 0$, para todo $x \in I$.
El método de Steffensen converge al único punto fijo de g para cualquier condición inicial $p_0 \in I$.

Criterios de paro
Una vez fijada una tolerancia ϵ. Parar cuando se cumpla uno o varios de los siguientes criterios:
(i) $|p_n - p_{n-1}| < \epsilon$,
(ii) $\dfrac{|p_n - p_{n-1}|}{|p_n|} < \epsilon$,
(iii) $|f(p_n)| < \epsilon$,
(iv) cuando agotemos el número máximo de iteraciones que hayamos prefijado.

El caso en que la ecuación cuyas raíces queremos hallar es un polinomio es, sin duda, el caso de mayor interés. Los métodos anteriores son eficientes si lo que se desea es hallar una raíz particular. A la hora de implementarlos, y con objeto de ahorrar en el número de operaciones, es necesario tener presente el esquema de Ruffini-Horner para evaluar polinomios. Si lo que queremos son todas las raíces del polinomio, lo más eficaz es construir una matriz cuyo polinomio característico sea el que tenemos y aplicar el algoritmo QR. De hecho, así es como funciona la instrucción **roots** de *MATLAB* que es la que sirve para calcular las raíces de un polinomio (hacemos notar que esta instrucción proporciona aproximaciones a todas las raíces, sean éstas reales o complejas).

2.2 Resolución de sistemas de ecuaciones lineales

Consideremos el sistema de n-ecuaciones con n-incógnitas:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n. \end{cases}$$

Representamos el sistema como $\mathbf{Ax} = \mathbf{b}$. Los métodos iterativos que estudiaremos transforman el sistema dado en otro equivalente de la forma $\mathbf{x} = T\mathbf{x} + \mathbf{c}$ para alguna matriz fija T y un vector $\mathbf{c}$.

Definición (radio espectral de una matriz)
El radio espectral de una matriz se define como

$$\rho(A) = \max_{1 \le i \le n} \{ \ |\lambda_i| \ / \ \lambda_i \ es \ un \ autovalor \ de \ A \ \}.$$

Teorema (convergencia de los métodos iterativos)
Para cualquier $\mathbf{x}^0 \in \mathbb{R}^n$, la sucesión $\{\mathbf{x}^k\}_{k \ge 0}$ definida por

$$\mathbf{x}^k = T\mathbf{x}^{k-1} + \mathbf{c}, \quad para \ cada \ k \ge 1,$$

converge a la solución única de $\mathbf{x} = T\mathbf{x} + \mathbf{c}$ si y sólo si $\rho(T) < 1$.

Algoritmo de Jacobi
(1) Partimos de $\mathbf{x}^0 = (x_1^0, \cdots, x_j^0, \cdots, x_n^0)$.
(2) Calculamos $\mathbf{x}^{(k+1)}$, para $k \ge 0$, iterando con la fórmula

$$x_j^{(k+1)} = \frac{b_j - a_{j1}x_1^{(k)} - \cdots - a_{jj-1}x_{j-1}^{(k)} - a_{jj+1}x_{j+1}^{(k)} - \cdots - a_{jn}x_n^{(k)}}{a_{jj}},$$

para $j = 1, 2, \cdots, n$.

Algoritmo de Gauss-Seidel
(1) Partimos de $\mathbf{x}^0 = (x_1^0, \cdots, x_j^0, \cdots, x_n^0)$.
(2) Calculamos $\mathbf{x}^{(k+1)}$, para $k \ge 0$, iterando con la fórmula

$$x_j^{(k+1)} = \frac{b_j - a_{j1}x_1^{(k+1)} - \cdots - a_{jj-1}x_{j-1}^{(k+1)} - a_{jj+1}x_{j+1}^{(k)} - \cdots - a_{jn}x_n^{(k)}}{a_{jj}},$$

para $j = 1, 2, \cdots, n$.

Notemos que en el método iterativo de Jacobi se utilizan las coordenadas del punto anterior para obtener las del nuevo punto, mientras que en el método iterativo de Gauss-Seidel para obtener el nuevo punto se emplean las coordenadas nuevas ya generadas y las del punto anterior aún no generadas.

Definición (matriz diagonal estrictamente dominante)
Una matriz A, de orden nxn, se dice de diagonal estrictamente dominante cuando

$$|a_{jj}| > |a_{j1}| + \cdots + |a_{jj-1}| + |a_{jj+1}| + \cdots + |a_{jn}|,$$

para $j = 1, 2, \cdots, n$.

Teorema (convergencia de los métodos de Jacobi y Gauss-Seidel)
Si una matriz A es de diagonal estrictamente dominante, entonces para cualquier $\mathbf{x}^0$, los métodos de Jacobi y Gauss-Seidel convergen a la única solución de $\mathbf{Ax} = \mathbf{b}$.

Algoritmos de relajación
(1) Partimos de $\mathbf{x}^0 = (x_1^0, \cdots, x_j^0, \cdots, x_n^0)$.
(2) Calculamos $\mathbf{x}^{(k+1)}$, para $k \geq 0$, iterando con la fórmula

$$x_j^{(k+1)} = (1-\omega)x_j^{(k)} + \frac{\omega\left(b_j - a_{j1}x_1^{(k+1)} - \cdots - a_{jj-1}x_{j-1}^{(k+1)} - a_{jj+1}x_{j+1}^{(k)} - \cdots - a_{jn}x_n^{(k)}\right)}{a_{jj}},$$

para $j = 1, 2, \cdots, n$ y ω una constante positiva.

Teorema (convergencia de los métodos de relajación)
Los métodos de relajación pueden converger sólo si $0 < \omega < 2$.

Los métodos que surgen al tomar $\omega \in (0, 1)$ se denominan métodos de subrelajación y pueden servir para obtener la convergencia de algunos sistemas que no son convergentes con el método de Gauss-Seidel ($\omega = 1$). Las elecciones de $\omega \in (1, 2)$ dan lugar a los métodos denominados de sobrerrelajación y sirven para acelerar la convergencia se sistemas que son convergentes con el método de Gauss-Seidel.

Criterios de paro
Las normas más usuales en los criterios de paro son:
(1)

$$||(x_1, \cdots, x_n)||_1 = |x_1| + \cdots + |x_n|.$$

(2)

$$||(x_1, \cdots, x_n)||_2 = \sqrt{x_1^2 + \cdots + x_n^2}.$$

(3)

$$||(x_1, \cdots, x_n)||_\infty = \max_{1 \leq i \leq n}\{|x_i|\}.$$

Una vez fijada una tolerancia ϵ, parar cuando se cumpla uno o varios de los siguientes criterios:

(i) $||\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}|| < \epsilon$,

(ii) $\dfrac{||\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}||}{||\mathbf{x}^{(k+1)}||} < \epsilon$,

(iii) $||A\mathbf{x}^{(k)} - \mathbf{b}|| < \epsilon$,

(iv) cuando agotemos el número máximo de iteraciones que hayamos prefijado.

2.3 Resolución de sistemas de ecuaciones no lineales

Sea el sistema de n-ecuaciones no lineales con n-incógnitas:

$$\left\{\begin{array}{l} f_1(x_1,x_2,\cdots,x_n)=0,\\ f_2(x_1,x_2,\cdots,x_n)=0,\\ \quad\vdots\\ f_n(x_1,x_2,\cdots,x_n)=0,\end{array}\right. \equiv \left\{\begin{array}{l} x_1=g_1(x_1,x_2,\cdots,x_n),\\ x_2=g_2(x_1,x_2,\cdots,x_n),\\ \quad\vdots\\ x_n=g_n(x_1,x_2,\cdots,x_n),\end{array}\right.$$

donde $f_i, g_i : \mathbb{R}^n \longmapsto \mathbb{R}$ para $i = 1, 2, \cdots, n$. Alternativamente podemos representar nuestro sistema como $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ o equivalentemente $\mathbf{x} = \mathbf{g}(\mathbf{x})$, con $\mathbf{f}, \mathbf{g} : \mathbb{R}^n \longmapsto \mathbb{R}^n$, donde $\mathbf{f} = (f_1, f_2, \cdots, f_n)$ y $\mathbf{g} = (g_1, g_2, \cdots, g_n)$.

Un primer método para resolver los sistemas no lineales, que también sirve para determinar un mínimo local para una función de la forma $\mathbf{f} : \mathbb{R}^n \longmapsto \mathbb{R}$ es el algoritmo del descenso más rápido. Más concretamente un sistema de la forma $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ tiene una solución en el punto $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$ cuando la función $\mathbf{g} : \mathbb{R}^n \longmapsto \mathbb{R}$ definida por

$$g(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n} [f_i(x_1, x_2, \cdots, x_n)]^2,$$

tiene el valor mínimo en $\mathbf{x} = \mathbf{0}$.

Algoritmo del descenso más rápido
Sea $\mathbf{f} : \mathbb{R}^n \longmapsto \mathbb{R}^n$ continua y con derivadas parciales continuas entonces:
(1) Partimos de $\mathbf{p_0} \in \mathbb{R}^n$.
(2) Para $n \geq 1$ calcular $\mathbf{p_{n+1}} = \mathbf{p_n} - \alpha \nabla g(\mathbf{p_n})$, para alguna constante $\alpha > 0$.
Si $g(\mathbf{p_{n+1}}) < \mathbf{g}(\mathbf{p_n})$, CONTINUAR.
Si $g(\mathbf{p_{n+1}}) \geq \mathbf{g}(\mathbf{p_n})$, disminuir el valor de α hasta que se cumpla lo anterior y CONTINUAR.

Nota: Para calcular el gradiente de la función g se puede aplicar la fórmula $\nabla g(\mathbf{p}) = 2\ (D\mathbf{f}(\mathbf{p}))^t\ \mathbf{f}(\mathbf{p})$, donde la matriz $D\mathbf{f}(\mathbf{p})$ es la matriz cuadrada con las derivadas parciales de la función $\mathbf{f}$.

El algoritmo anterior se utiliza para encontrar buenas aproximaciones para métodos de convergencia más rápida como Newton-Raphson y Broyden. Ambos métodos utilizan la técnica conocida como iteración de punto fijo, que está basada en el teorema del punto fijo correspondiente al caso vectorial.

Teorema(del punto fijo) *Sea A un conjunto cerrado de $\mathbb{R}^n$ y $g : A \longmapsto A$ continua y con derivadas parciales continuas, tal que existe $K \in (0, 1)$ con*

$$\left|\frac{\partial g_i(\mathbf{x})}{\partial x_j}\right| \leq \frac{K}{n}, \quad \text{para todo} \quad x \in A, \quad i, j = 1, \cdots, n.$$

Entonces g tiene un único punto fijo $p \in A$, de forma que el algoritmo:

$$(PF) \qquad \left\{\begin{array}{l} p_0 \in A,\\ p_n = g(p_{n-1}),\ n \geq 1,\end{array}\right.$$

converge a p y además se cumple:

$$||p_n - p|| \leq \frac{K^n}{1-K}||p_1 - p_0||, \qquad n \geq 1.$$

Una forma de acelerar, generalmente, la convergencia del método de iteración de punto fijo consiste en utilizar la misma idea del método iterativo de Gauss-Seidel para los sistemas lineales.

Algoritmo de Seidel
(1) Partimos de $\mathbf{p^0} = (p_1^0, p_2^0, \cdots, p_n^0)$.
(2) Para $k \geq 1$, calcular $\mathbf{p^k}$ iterando con la fórmula:

$$(S) \qquad \begin{cases} p_1^k = g_1(p_1^{k-1}, p_2^{k-1}, p_3^{k-1}, \cdots, p_{n-1}^{k-1}, p_n^{k-1}), \\ p_2^k = g_2(p_1^k, p_2^{k-1}, p_3^{k-1}, \cdots, p_{n-1}^{k-1}, p_n^{k-1}), \\ p_3^k = g_3(p_1^k, p_2^k, p_3^{k-1}, \cdots, p_{n-1}^{k-1}, p_n^{k-1}), \\ \vdots \\ p_n^k = g_n(p_1^k, p_2^k, p_3^k, \cdots, p_{n-1}^k, p_n^{k-1}). \end{cases}$$

Algoritmo de Newton-Raphson para sistemas no lineales

$$(NRvv) \qquad \begin{cases} p_0 \approx p, \\ Df(p_n) z_n = -f(p_n), \\ p_{n+1} = p_n + z_n, \ n \geq 0. \end{cases}$$

Algoritmo de Broyden (cuasi-Newton)
(1) Tomar $p_0 \approx p$ y definir $A_0 = Df(p_0)$ y $p_1 = p_0 - A_0^{-1} f(p_0)$.
(2) Para $n \geq 1$.
(2-1)

$$A_n = A_{n-1} + \frac{[f(p_n) - f(p_{n-1}) - A_{n-1}(p_n - p_{n-1})]\,[p_n - p_{n-1}]^t}{||p_n - p_{n-1}||_2^2},$$

(2-2) Resolver: $A_n z_n = -f(p_n)$,
(2-3) $p_{n+1} = p_n + z_n$.

2.4 Ejercicios Resueltos Tema 2

PROBLEMA 2-1: Se trata de encontrar la única solución en $[0,1]$ de la ecuación $f(x) = x^7 + 5x^5 + 2x^2 + x - 1 = 0$, aplicando algunos de los métodos estudiados para tal fín.
(1) Construir un gráfico de $y = f(x)$ en el intervalo $[-10, 10]$. (Con ello sacar una idea de las raices de $f(x) = 0$). Demostrar que la ecuación $f(x) = 0$ sólo tiene una raíz en $\mathbb{R}$.
(2) Construir un gráfico de $y = f(x)$ en el intervalo $[0, 1]$. (Con ello obtener una primera aproximación de la raíz de $f(x) = 0$ en dicho intervalo).
(3) Calcular la raíz por el método de bisección con $TOL = 10^{-10}$, y construye un gráfico que muestre la convergencia de las iteraciones.
(4) ¿Es posible aplicar el método de punto fijo con la función $g_1(x) = \frac{1-x-5x^5-x^7}{2x}$? ¿y con la función $g_2(x) = \frac{1}{x^6+5x^4+2x+1}$? En caso afirmativo, calcular la solución con las siguientes especificaciones: $p_0 = 0.4$ (condición inicial), $N_{max} = 200$ (número máximo de iteraciones) y como criterio de paro tomar: $|p_n - p_{n-1}| < TOL$ y además $|f(p_n)| < TOL$ con una tolerancia TOL dada.
(5) Aplicar Newton-Raphson con las especificaciones del apartado anterior.
(6) Aplicar Steffensen con la función $g_2(x)$ y las especificaciones del apartado (4).
(7) Escribir en una tabla las iteraciones 3,6,...,30 de los métodos: punto fijo para $g_2(x)$, sucesión acelerada de Aitken, Newton-Raphson y Steffensen.

SOLUCIÓN PROBLEMA 2-1:

En principio definimos las funciones que vamos a utilizar:

• Fichero f.m que define la función $f(x) = x^7 + 5x^5 + 2x^2 + x - 1$.

```
function y=f(x)
y=x.^7+5*x.^5+2*x.^2+x-1;
```

• Fichero fprima.m que define la función $fprima(x) = 7x^6 + 25x^4 + 4x + 1$.

```
function y=fprima(x)
y=7*x.^6+25*x.^4+4*x+1;
```

• Fichero gprima1.m que define la función $gprima1(x) = (-2 - 40x^5 - 12x^7)/(4x^2)$.

```
function y=gprima1(x)
y=(-2-40*x.^5-12*x.^7)./(4*x.^2);
```

• Fichero gprima2.m que define la función $gprima2(x) = -(6x^5 + 20x^3 + 2)/(x^6 + 5x^4 + 2x + 1)^2$.

```
function y=gprima2(x)
y=-(6*x.^5+20*x.^3+2)./(x.^6+5*x.^4+2*x+1).^2;
```

• Fichero g2.m que define la función $g2(x) = 1/(x^6 + 5x^4 + 2x + 1)$.

```
function y=g2(x)
y=1./(x.^6+5*x.^4+2*x+1);
```

Apartado (1)

```
% Grafico de y=f(x) en [-10,10] (la funcion f(x) esta en fichero f.m)
% Construye un vector desde -10 hasta 10 con incremento de .01 en .01,
x=-10:.01:10;
% Construye el vector y=[f(-10) f(-9.99) f(-9.98) ...f(9.98) f(9.99) f(10)],
y=f(x);
% dibuja x frente a y
plot(x,y)
```

RESULTADO

▷ El gráfico aparece representado en la figura 2.1. ◁

Apartado (2)

```
% Grafico de y=f(x) en [0,1]
x=0:.01:1;
y=f(x);
plot(x,y)
% hace que en el dibujo salga una malla.
grid
```

RESULTADO

▷ El gráfico aparece representado en la figura 2.2. ◁

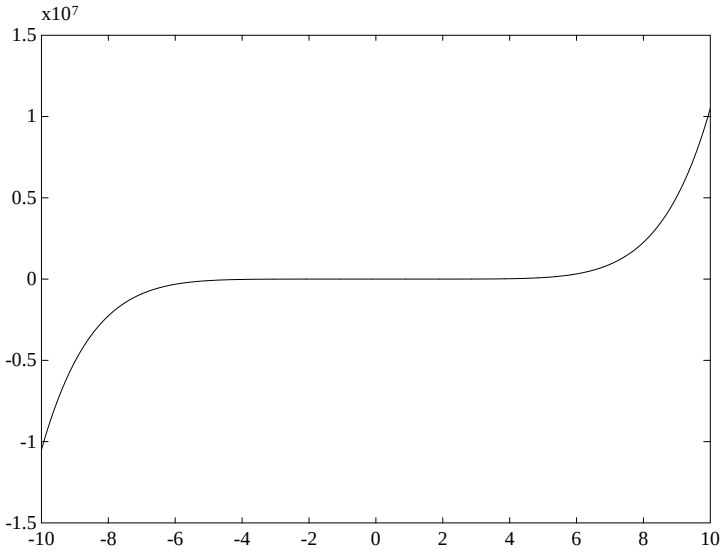

Figura 2.1: Representación de $y = x^7 + 5x^5 + 2x^2 + x - 1$ en $[-10, 10]$.

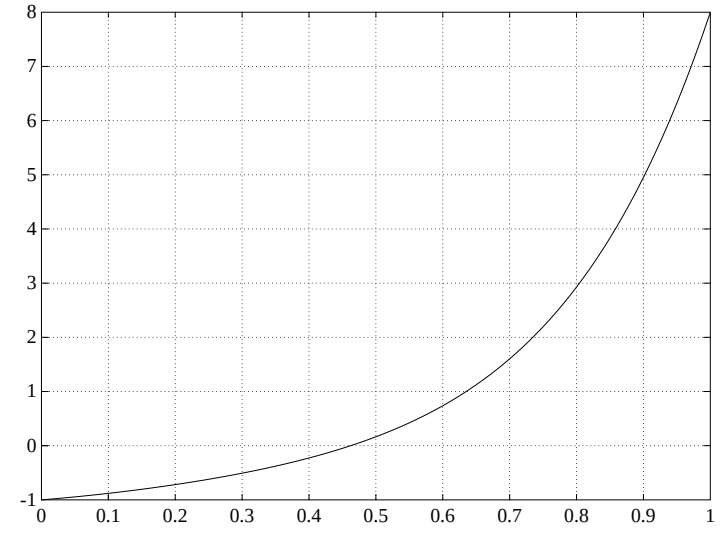

Figura 2.2: Representación de $y = x^7 + 5x^5 + 2x^2 + x - 1$ en $[0, 1]$.

Apartado (3)
Para resolver este apartado generamos el siguiente archivo, que proporciona la aproximación cuando se alcanza la tolerancia prefijada.

```
function problema1a3tema2(f,a,b,tol)
%             Datos de entrada para el metodo de biseccion
% f es la funcion introducida como una cadena de caracteres 'f' que toma
% desde el fichero f.m
% a y b son los extremos del intervalo
% tol es la tolerancia
%
format long
ya=feval(f,a);
yb=feval(f,b);
if ya*yb>0
    disp('el intervalo podria no contener solucion')
    return
end
if ya==0
    disp('la solucion es el extremo inferior'), p=a
    return
end
if yb==0
    disp('la solucion es el extremo superior'), p=b
    return
end
i=0;
while b-a >= tol
    i=i+1;
    c(i)=(a+b)/2;
    yc=feval(f,c(i));
    if yc==0
        p=c(i);
       break
    end
    if ya*yc<0
        b=c(i);
    else
        a=c(i);
    end
end
%           Salidas del programa
% la aproximacion a la raiz de la ecuacion f(x)=0
p=(a+b)/2
% el numero de iteraciones realizadas
disp('el numero de iteraciones realizadas es'), i pause
% grafica de las iteraciones frente a las aproximaciones a la solucion
plot(c)
grid
```

RESULTADO

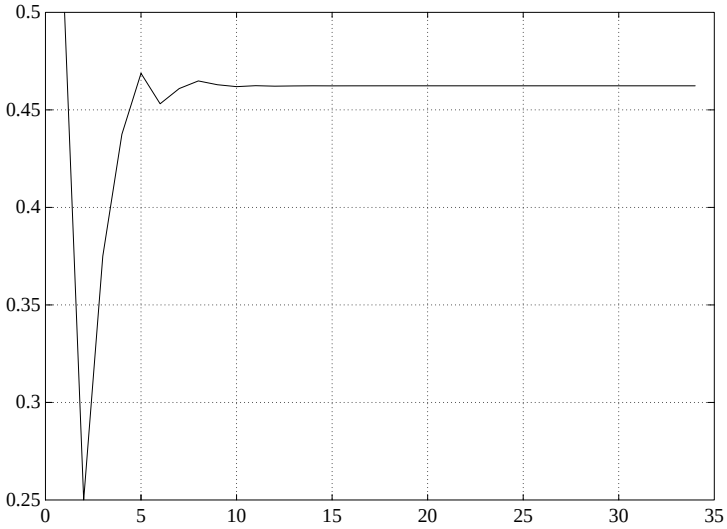

Figura 2.3: Evolución de las 34 primeras iteraciones por el método de bisección para la ecuación $x^7 + 5x^5 + 2x^2 + x - 1 = 0$.

▷ extremo inferior del intervalo a= 0
extremo superior del intervalo b= 1
Tolerancia para parar $TOL = 10^{-10}$
>> problema1a3tema2('f',0,1,10^{-10})
solucion aproximada= 0.46234002718120
el numero de iteraciones es: 34. ◁

Apartado (4)

```
%Dibujo de la funcion gprima2 ( derivada de la funcion
% g que me dan para aplicar punto fijo ) con ello poder observar
% si el metodo del punto fijo se puede aplicar localmente. La funcion
% gprima2 esta definida en el fichero gprima2.m
x=.4:.001:.5;
y=gprima2(x);
plot(x,y)
```

RESULTADO

```
%Dibujo de la funcion gprima1 ( derivada de la funcion
% g que me dan para aplicar punto fijo ) con ello poder observar
% si el metodo del punto fijo se puede aplicar localmente. La funcion
% gprima1 esta definida en el fichero gprima1.m
x=.4:.001:.5;
y=gprima1(x);
plot(x,y)
```

RESULTADO

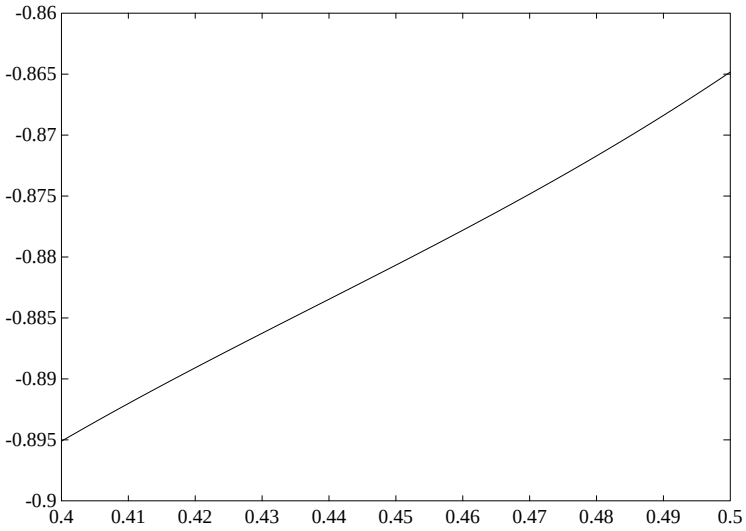

Figura 2.4: Representación de $g_2'(x)$ (converge localmente g_2).

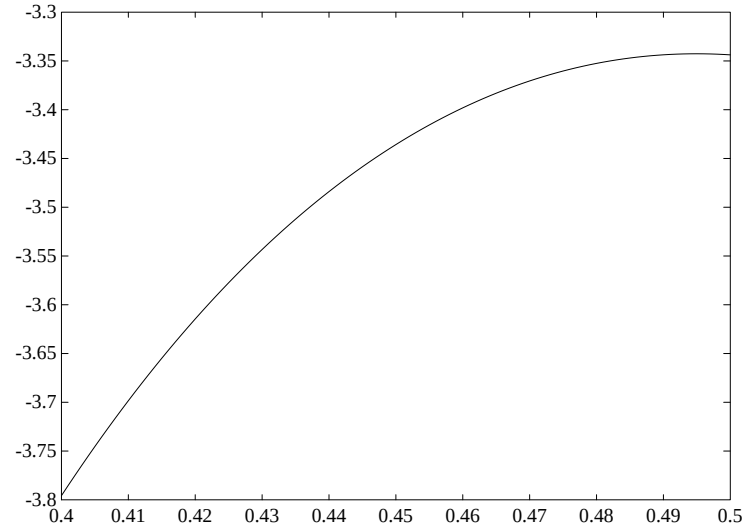

Figura 2.5: Representación de $g_1'(x)$ (no converge g_1).

```
% Aplicacion del metodo de punto fijo a la funcion
%g2(x)=1/(x^6+5x^4+2x+1)(la funcion g2(x) esta en fichero g2.m)
%  para resolver la ecuacion
%f(x)=x^7+5x^5+2x^2+x-1=0 (la funcion f(x) esta en fichero f.m)
% Adoptamos como criterio de paro que la diferencia entre dos iteraciones
% consecutivas sea menor que TOL y ademas que abs(f(pn))<TOL.
format long
N=200; %numero maximo de iteraciones
TOL=10^(-10);
p0=.4;%condicion inicial
c=0;
for k=1:N
p1=g2(p0);
if abs(p1-p0)<TOL & abs(f(p1))<TOL
'la solucion aproximada es', p1
'alcanzada en ',k,'iteraciones'
c=1;
break
end
p0=p1;
end
if c==0
'el metodo no converge para', N, 'iteraciones con la condicion inicial y TOL dada'
end
```

RESULTADO

▷ la solucion aproximada es 0.46234002717677
alcanzada en 166 iteraciones. ◁

Apartado (5)

```
%Metodo de Newton-Raphson, fprima (x)=f'(x).
format long
N=200; %numero maximo de iteraciones
TOL=10^(-10);
p0=.4;%condicion inicial
c=0;
for k=1:N
p1=p0-f(p0)/fprima(p0);
if abs(p1-p0)<TOL & abs(f(p1))<TOL
'la solucion aproximada es', p1
'alcanzada en ',k,'iteraciones'
c=1;
break
end
p0=p1;
end
if c==0
'el metodo no converge para', N, 'iteraciones con la condicion inicial y TOL dada'
end
```

RESULTADO

▷ la solucion aproximada es 0.46234002719882 alcanzada en 5 iteraciones. ◁

Apartado (6)

```
%Metodo de Steffensen, para g_2(x)=1/(x^6+5x^4+2x+1).
format long
N=200; %numero maximo de iteraciones
TOL=10^(-15);
p0=.4;%condicion inicial
c=0;
for k=1:N
p1=g2(p0);
p2=g2(p1);
p0p=p0-(p1-p0)^2/(p2-2*p1+p0);
if abs(p0p-p0)<TOL & abs(f(p0p))<TOL
'la solucion aproximada es', p0p
'alcanzada en ',k,'iteraciones'
c=1;
break
end
p0=p0p;
end
if c==0
'el metodo no converge para', N, 'iteraciones con la condicion inicial y TOL dada'
end
```

RESULTADO

▷ la solucion aproximada es 0.46234002719882 alcanzada en 4 iteraciones. ◁

Apartado (7)

```
%Iteraciones 3, 6,..., 30 del metodo de punto fijo, para p0=0.4 y
%g2(x)=1/(x^6+5x^4+2x+1)(la funcion g2(x) esta en fichero g2.m)
format long
p0=.4;%condicion inicial
for k=1:30
p1=g2(p0);
if fix(k/3)==k/3
PuntoFijo(fix(k/3))=p1;%iteraciones multiplo de 10 del punto fijo.
end
p0=p1;
end
%Iteraciones 1, 4,..., 28 de la sucesion anterior
% acelerada por el metodo de Aitken.
p0=.4;%condicion inicial
for k=1:29
p1=g2(p0);
p2=g2(p1);
if fix((k+1)/3)==(k+1)/3
Aitken(fix((k+1)/3))=p0-(p1-p0)^2/(p2-2*p1+p0);%iteraciones multiplo de 10 del punto fijo.
end
```

```
p0=p1;
end
%Iteraciones 3, 6,..., 30 del metodo de Newton-Raphson, para p0=0.4.
p0=.4;%condicion inicial
for k=1:30
p1=p0-f(p0)/fprima(p0);
if fix(k/3)==k/3
Newton(fix(k/3))=p1;%iteraciones multiplo de 10 del punto fijo.
end
p0=p1;
end
%Iteraciones 3, 6,..., 30 del metodo de Steffensen, para p0=0.4 y
%g2(x)=1/(x^6+5x^4+2x+1).
p0=.4;%condicion inicial
for k=1:30
p1=g2(p0);
p2=g2(p1);
p0p=p0-(p1-p0)^2/(p2-2*p1+p0);
if fix(k/3)==k/3
Steffensen(fix(k/3))=p0p;%iteraciones multiplo de 10 del punto fijo.
end
p0=p0p;
end
[PuntoFijo' Aitken' Newton' Steffensen']
```

RESULTADO

▷ Warning: Divide by zero
Warning: Divide by zero
Warning: Divide by zero
Warning: Divide by zero

k	Punto Fijo	Aitken	Newton	Steffensen
3	0.50472176008353	0.46255651365537	0.46234004291216	0.46234002719882
6	0.43395454400580	0.46243482993704	0.46234002719882	NaN
9	0.48156760658663	0.46238324452366	0.46234002719882	NaN
12	0.44940530680112	0.46235946124672	0.46234002719882	NaN
15	0.47108449142613	0.46234890458723	0.46234002719882	NaN
18	0.45644717540983	0.46234405120277	0.46234002719882	NaN
21	0.46631998145212	0.46234186320516	0.46234002719882	NaN
24	0.45965594778309	0.46234086171329	0.46234002719882	NaN
27	0.46415198013856	0.46234040759105	0.46234002719882	NaN
30	0.46111764217292	0.46234020027946	0.46234002719882	NaN

◁

PROBLEMA 2-2: Una persona espera impacientemente a otra que llega con retraso, de manera que camina nerviosa a lo largo de una calle, se mueve hacia la izquierda con frecuencia triple que a la derecha. Supongamos posiciones numeradas en la calle de 0 a 20, de manera que, partiendo de $x_0 = 1$ y $x_{20} = 0$, las posiciones a lo largo del tiempo se obtendrán resolviendo el sistema tridiagonal:

$$x_k = \frac{3}{4}x_{k-1} + \frac{1}{4}x_{k+1}.$$

(1) Resolver el sistema planteado utilizando el operador \ de Matlab.
(2) Aplicar el método iterativo de Jacobi para obtener la solución, analizando previamente la convergencia del método.
(3) Aplicar el método iterativo de Gauss-Seidel para obtener la solución, analizando previamente la convergencia del método.
(4) Experimenta con distintos valores de ω en el método S.O.R. para acelerar la convergencia. Calcular el radio espectral de la matriz del método.
(5) El problema descrito es un problema de frontera para una ecuación en diferencias, cuya solución exacta es:

$$x_k = 1 - \frac{3^k - 1}{3^{20} - 1}.$$

Calcula estos valores para $k = 0, 1, 2, \cdots, 20$ y compara los resultados con los obtenidos en los apartados anteriores.

SOLUCIÓN PROBLEMA 2-2:

El sistema que se obtiene es

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0.75 & -1 & 0.25 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0.75 & -1 & 0.25 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0.75 & -1 & 0.25 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{19} \\ x_{20} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}.$$

Apartado (1)
En primer lugar generamos el sistema en Matlab

```
function [A,B]=sistemaproblema2tema2(n)
A(1,1)=1;
A(n,n)=1;
for i=2:n-1
    for j=1:n
        if i==j
            A(i,j)=-1;
        elseif i==j-1
            A(i,j)=0.25;
        elseif i==j+1
            A(i,j)=0.75;
        end
```

```
    end
end
B=zeros(n,1);
B(1)=1;
```

RESULTADO

>> sistemaproblema2tema2(21)
>> A \B
El vector solución resultante es

$$\begin{pmatrix} 1.00000000000000 \\ 0.99999999942641 \\ 0.99999999770562 \\ 0.99999999254327 \\ 0.99999997705622 \\ 0.99999993059508 \\ 0.99999979121164 \\ 0.99999937306132 \\ 0.99999811861037 \\ 0.99999435525753 \\ 0.99998306519898 \\ 0.99994919502336 \\ 0.99984758449648 \\ 0.99954275291584 \\ 0.99862825817392 \\ 0.99588477394817 \\ 0.98765432127091 \\ 0.96296296323914 \\ 0.88888888914382 \\ 0.66666666685787 \\ 0 \end{pmatrix}$$

Apartado (2)
Para analizar la convergencia del método, elaboramos un fichero que calcula el radio espectral de la matriz de Jacobi

```
function re=radioespectralJacobi
%
%           Datos de salida
% re es el radio espectral de la matriz de Jacobi
%
format long
% lee la matriz de los coeficientes del sistema de ecuaciones Ax=B
A=sistemaproblema2tema2(21);
% calculo de las matrices D,L,U
D=diag(diag(A)); L=tril(A)-D; U=triu(A)-D;
% calculo del radio espectral
re=max(abs(eig(-inv(D)*(L+U))));
```

A continuación generamos el siguiente archivo, que proporciona la aproximación cuando se alcanza la tolerancia prefijada.

```
function X=Jacobi(P,tol,max)
%
%         Datos de entrada
% P es el dato inicial, un vector de orden nx1
% tol es la tolerancia
% max es el numero maximo de iteraciones
%
%         Datos de salida
% X es una matriz de orden nx1 con la aproximacion a la
% solucion de AX=B obtenida por el metodo de Jacobi
%
format long
% lee la matriz A, matriz cuadrada invertible de orden n,
% y el vector B, de orden nx1, al ejecutar el fichero sistemaproblema2tema2
[A,B]=sistemaproblema2tema2(21);
n=length(B);
c=0;
for k=1:max
   % construccion de la siguiente iteracion por el metodo de Jacobi
   for j=1:n
      X(j)=(B(j)-A(j,[1:j-1,j+1:n])*P([1:j-1,j+1:n]))/A(j,j);
   end
   % condicion de paro
      ea=abs(norm(X'-P));
      er=ea/(norm(X)+eps);
   if(ea<tol)|(er<tol)
      disp('Se alcanzo la tolerancia con exito en la iteracion')
      k
      disp('y la solucion aproximada es')
      X=X';
      c=1;
      break
      else
      P=X';
   end
end
% no cumplimiento de la condicion de paro
if c==0
      disp('El metodo no cumple el criterio de paro propuesto para')
      max
      disp('iteraciones.')
end
```

RESULTADO

>> radioespectralJacobi
ans = 0.85536319397709
Tomando como aproximación inicial el vector nulo
>> P=zeros(21,1);
>> Jacobi(P,10^{-10},200)

Se alcanzo la tolerancia con exito en la iteración
k = 170
y la solución aproximada es

$$\begin{pmatrix} 1.00000000000000 \\ 0.99999999942637 \\ 0.99999999770546 \\ 0.99999999254292 \\ 0.99999997705531 \\ 0.99999993059345 \\ 0.99999979120786 \\ 0.99999937305516 \\ 0.99999811859707 \\ 0.99999435523707 \\ 0.99998306515708 \\ 0.99994919496210 \\ 0.99984758437714 \\ 0.99954275275031 \\ 0.99862825786984 \\ 0.99588477355462 \\ 0.98765432060895 \\ 0.96296296248186 \\ 0.88888888810060 \\ 0.66666666607545 \\ 0 \end{pmatrix}$$

Apartado (3)
Para analizar la convergencia del método, elaboramos un fichero que calcula el radio espectral de la matriz de GaussSeidel

```
function re=radioespectralGaussSeidel
%
%           Datos de salida
% re es el radio espectral de la matriz de GaussSeidel
%
format long
% lee la matriz de los coeficientes del sistema de ecuaciones Ax=B
A=sistemaproblema2tema2(21);
% calculo de las matrices D,L,U
D=diag(diag(A)); L=tril(A)-D; U=triu(A)-D;
% calculo del radio espectral
re=max(abs(eig(-inv(D+L)*U)));
```

A continuación generamos el siguiente archivo, que proporciona la aproximación cuando se alcanza la tolerancia prefijada.

```
function X=GaussSeidel(P,tol,max)
%
%         Datos de entrada
% P es el dato inicial, un vector de orden nx1
% tol es la tolerancia
% max es el numero maximo de iteraciones
```

```
%
%         Datos de salida
% X es una matriz de orden nx1 con la aproximacion a la
% solucion de AX=B obtenida por el metodo de GaussSeidel
%
format long
% lee la matriz A, matriz cuadrada invertible de orden n,
% y el vector B, de orden nx1, al ejecutar el fichero sistemaproblema2tema2
[A,B]=sistemaproblema2tema2(21);
n=length(B);
c=0;
for k=1:max
   % construccion de la siguiente iteracion por el metodo de GaussSeidel
   for j=1:n
      if j==1
         X(1)=(B(1)-A(1,2:n)*P(2:n))/A(1,1);
      elseif j==n
         X(n)=(B(n)-A(n,1:n-1)*(X(1:n-1))')/A(n,n);
      else
         X(j)=(B(j)-A(j,1:j-1)*(X(1:j-1))'-A(j,j+1:n)*P(j+1:n))/A(j,j);
      end
   end
   % condicion de paro
      ea=abs(norm(X'-P));
      er=ea/(norm(X)+eps);
   if(ea<tol)|(er<tol)
      disp('Se alcanzo la tolerancia con exito en la iteracion')
      k
      disp('y la solucion aproximada es')
      X=X';
      c=1;
      break
      else
      P=X';
   end
end
% no cumplimiento de la condicion de paro
if c==0
      disp('El metodo no cumple el criterio de paro propuesto para')
      max
      disp('iteraciones.')
end
```

RESULTADO

>> radioespectralGaussSeidel
ans = 0.73164619361068
Tomando como aproximación inicial el vector nulo
>> P=zeros(21,1);
>> GaussSeidel(P,10^{-10},100)

Se alcanzo la tolerancia con exito en la iteración
k = 78
y la solución aproximada es

$$\begin{pmatrix} 1.00000000000000 \\ 0.99999999942605 \\ 0.99999999770457 \\ 0.99999999254099 \\ 0.99999997705184 \\ 0.99999993058728 \\ 0.99999979119844 \\ 0.99999937303981 \\ 0.99999811857639 \\ 0.99999435520529 \\ 0.99998306512071 \\ 0.99994919490893 \\ 0.99984758433339 \\ 0.99954275268964 \\ 0.99862825786984 \\ 0.99588477355462 \\ 0.98765432078643 \\ 0.96296296268489 \\ 0.88888888858496 \\ 0.66666666643872 \\ 0 \end{pmatrix}$$

Apartado (4)
Para analizar la convergencia del método, elaboramos un fichero que calcula el radio espectral de la matriz de S.O.R.

```
function re=radioespectralSOR(w)
%
%           Datos de entrada
% w es el valor de omega
%           Datos de salida
% re es el radio espectral de la matriz de Jacobi
%
format long
% lee la matriz de los coeficientes del sistema de ecuaciones Ax=B
A=sistemaproblema2tema2(21);
% calculo de las matrices D,L,U
D=diag(diag(A)); L=tril(A)-D; U=triu(A)-D;
% calculo del radio espectral
re=max(abs(eig(-inv(w*L+D)*((w-1)*D+w*U))));
```

A continuación generamos el siguiente archivo, que proporciona la aproximación cuando se alcanza la tolerancia prefijada.

```
function X=SOR(P,w,tol,max)
%
%         Datos de entrada
% P es el dato inicial, un vector de orden nx1
```

```
% w es el valor de omega
% tol es la tolerancia
% max es el numero maximo de iteraciones
%
%         Datos de salida
% X es una matriz de orden nX1 con la ultima iteracion en la aproximacion
% a la solucion de AX=B obtenida por el metodo SOR
%
format long
% lee la matriz A, matriz cuadrada invertible de orden n,
% y el vector B, de orden nx1, al ejecutar el fichero sistemaproblema2tema2
[A,B]=sistemaproblema2tema2(21);
n=length(B);
c=0;
for k=1:max
%construccion de la siguiente iteracion por el metodo de SOR
   for j=1:n
      if j==1
         X(1)=(1-w)*P(j)+ w*(B(1)-A(1,2:n)*P(2:n))/A(1,1);
      elseif j==n
         X(n)=(1-w)*P(j)+ w*(B(n)-A(n,1:n-1)*(X(1:n-1))')/A(n,n);
      else
         X(j)=(1-w)*P(j)+ w*(B(j)-A(j,1:j-1)*(X(1:j-1))' ...
         -A(j,j+1:n)*P(j+1:n))/A(j,j);
      end
   end
%condicion de paro
   ea=abs(norm(X'-P));
   er=ea/(norm(X)+eps);
   if(ea<tol)|(er<tol)
       disp('Se alcanzo la tolerancia con exito en la iteracion')
       k
       disp('y la solucion aproximada es')
       X=X';
       c=1;
    break
    else
       P=X';
    end
end
if c==0
      disp('El metodo no cumple el criterio de paro propuesto para')
      max
      disp('iteraciones.')
end
```

RESULTADO

>> radioespectralSOR(1.3)
ans = 0.42443592739282
Tomando como aproximación inicial el vector nulo

>> P=zeros(21,1);
>> SOR(P,1.3,10^{-10},100)

Se alcanzo la tolerancia con exito en la iteración
k = 27
y la solución aproximada es

$$\begin{pmatrix} 1.00000000000001 \\ 0.99999999942450 \\ 0.99999999770134 \\ 0.99999999253618 \\ 0.99999997704584 \\ 0.99999993058100 \\ 0.99999979119344 \\ 0.99999937303872 \\ 0.99999811858313 \\ 0.99999435522561 \\ 0.99998306516251 \\ 0.99994919498272 \\ 0.99984758445230 \\ 0.99954275286916 \\ 0.99862825812607 \\ 0.99588477390099 \\ 0.98765432122665 \\ 0.96296296320057 \\ 0.88888888911419 \\ 0.66666666684095 \\ 0 \end{pmatrix}$$

Apartado (5)
Para resolver este apartado generamos un archivo que nos proporciona la solución exacta y el valor absoluto del error cometido en cada componente de la solución.

```
function error=problema2a5tema2
% calculo de la solucion exacta X
X(1)=1;
for k=2:21
    X(k)=1-(3^(k-1)-1)/(3^20-1);
end
% calculo de las aproximaciones
P=zeros(21,1);
S2=Jacobi(P,10^(-10),200);
S3=GaussSeidel(P,10^(-10),200);
S4=SOR(P,1.3,10^(-10),100);
[A,B]=sistemaproblema2tema2(21);
S1=A\B;
format long e
error=[abs(X'-S2),abs(X'-S3),abs(X'-S4),abs(X'-S1)];
```

RESULTADO

$$\begin{pmatrix}
0 & 0 & 7.549516567451065e-015 \\
4.019007349143067e-014 & 3.592681707687007e-013 & 1.901478974275506e-012 \\
1.606492716632602e-013 & 1.050715070505248e-012 & 4.280353849139829e-012 \\
3.496092304544618e-013 & 2.285172051585960e-012 & 7.088440945324237e-012 \\
9.162670622231417e-013 & 4.379274720633930e-012 & 1.037936403491813e-011 \\
1.631805801594055e-012 & 7.796985279640012e-012 & 1.408162475513564e-011 \\
3.778533042009258e-012 & 1.320166198581774e-011 & 1.819766559663094e-011 \\
6.160294496737606e-012 & 2.151656630644538e-011 & 2.260647224971990e-011 \\
1.330557886092265e-011 & 3.398759051975731e-011 & 2.724176439983239e-011 \\
2.045408287187911e-011 & 5.223577126400869e-011 & 3.191380493205998e-011 \\
4.189926183784110e-011 & 7.827150039219077e-011 & 3.647138147044871e-011 \\
6.125955298585950e-011 & 1.144228045646401e-010 & 4.064137915094079e-011 \\
1.193405374522172e-010 & 1.630904300498060e-010 & 4.417266552536603e-011 \\
1.655253711874138e-010 & 2.261937304126604e-010 & 4.668310182864843e-011 \\
3.040797613707014e-010 & 3.040797613707014e-010 & 4.784606044694328e-011 \\
3.935509695907058e-010 & 3.935509695907058e-010 & 4.717493062855738e-011 \\
6.619647052730215e-010 & 4.844827872219071e-010 & 4.426303767957052e-011 \\
7.572783511378134e-010 & 5.542527548030307e-010 & 3.856548413949668e-011 \\
1.043219510776794e-009 & 5.588575158199660e-010 & 2.963107537112819e-011 \\
7.824146885937466e-010 & 4.191431646205501e-010 & 1.691535800318889e-011 \\
0 & 0 & 0
\end{pmatrix}$$

$$\begin{pmatrix}
0 \\
0 \\
1.110223024625157e-016 \\
1.110223024625157e-016 \\
2.220446049250313e-016 \\
1.110223024625157e-016 \\
3.330669073875470e-016 \\
4.440892098500626e-016 \\
4.440892098500626e-016 \\
4.440892098500626e-016 \\
4.440892098500626e-016 \\
6.661338147750939e-016 \\
6.661338147750939e-016 \\
5.551115123125783e-016 \\
5.551115123125783e-016 \\
5.551115123125783e-016 \\
5.551115123125783e-016 \\
5.551115123125783e-016 \\
5.551115123125783e-016 \\
4.440892098500626e-016 \\
0
\end{pmatrix}$$

PROBLEMA 3-2: Se trata de resolver el sistema:

$$\begin{cases} 3x - cos(yz) - \frac{1}{2} = 0, \\ x^2 - 81(y+0.1)^2 + sen(z) + 1.06 = 0, \\ e^{-xy} + 20z + \frac{10\pi - 3}{3} = 0. \end{cases}$$

(1) Escribir el sistema en la forma [1]. Estudiar la convergencia del algoritmo del punto fijo con las especificaciones: vector inicial $p_0 = (0.1, 0.1, -0.1)$ y como criterio de paro $||p_n - p_{n-1}||_\infty < 10^{-10}$ y número máximo de iteraciones 1000. Imprimir las 10 primeras iteraciones, asi como la norma infinito de la diferencia de dos iteraciones consecutivas. Idem para las iteraciones $10, 20, \cdots, 100$.
(2) Repetir las preguntas del apartado (1) para el método de Seidel.
(3) Análogo al apartado (1) pero utilizando el método de Newton-Raphson.
(4) Idem al apartado (1) pero usando el método de Broyden.

SOLUCIÓN PROBLEMA 3-2:

Método del punto fijo.
La función g del punto fijo la llamaremos gtrid, y está en el fichero gtrid.m que listamos a continuacion:

```
%define la funcion tridimensional (llamada gtrid). p(x,y,z)----->g(g1,g2,g3).
function g=gtrid(p)
x=p(1);y=p(2);z=p(3);
g(1)=1/3*cos(y*z)+1/6;
g(2)=1/9*sqrt(x^2+sin(z)+1.06)-0.1;
g(3)=-1/20*exp(-x*y)-(10*pi-3)/60;
```

A continuación, calculamos las iteraciones del punto fijo hasta que la norma infinito de dos iteraciones consecutivas es menor que TOL (partiendo de la condicion inicial [.1 .1 −.1]).
El programa devuelve el número de iteraciones necesarias para alcanzar dicha tolerancia así como la aproximación alcanzada (número máximo de iteraciones 1000).

```
TOL=10^(-10);
x=[.1 .1 -.1];
y=gtrid(x);
c=1;
while norm(x-y,inf)>=TOL
  x=y;
  y=gtrid(x);
  c=c+1;
  if c>1000  %si en 1000 iteraciones no converge mensaje de error
     error('no se alcanzo dicha tolerancia')
  end
end
'el numero de iteraciones en alcanzar la TOL',
TOL,
'es',
c
'la solucion aproximada es'
format long
y
```

[1] $x = g_1(x,y,z) = \frac{cos(yz)+0.5}{3}$, $y = g_2(x,y,z) = \frac{1}{9}\sqrt{x^2 + sen(z) + 1.06} - 0.1$ y $z = g_3(x,y,z) = -\frac{1}{20}e^{-xy} - \frac{10\pi-3}{60}$. Demostrar que $g = (g_1, g_2, g_3)$ satisface el teorema del punto fijo con $A = [-1, 1]^3$ y $K = 0.85$.

RESULTADO

▷ el numero de iteraciones en alcanzar la TOL 1.000000000000000e-10 es 8
la solucion aproximada es
0.50000000000000 0.00000000000006 -0.52359877559775. ◁

El siguiente código, imprime las iteraciones 1,2,..., 10, así como la norma infinito de la diferencia de dos iteraciones consecutivas.

```
%imprime las primeras 10 iteraciones  y la norma infinito
%entre  dos iteraciones consecutivas
format long
'condicion inicial',
x=[.1 .1 -.1];
for k=1:10
    y=gtrid(x);
    A(k,:)=[y norm(x-y,inf)];
    x=y;
end
format long
A
```

RESULTADO

k	x_k	y_k	z_k	$\|\|p^{(k)}-p^{(k-1)}\|\|_\infty$
0	0.10000000000000	0.10000000000000	-0.10000000000000	–
1	0.49998333347222	0.00944114960371	-0.52310126728576	0.42310126728576
2	0.49999593491931	0.00002556774677	-0.52336331090880	0.00941558185695
3	0.49999999997016	0.00001233672036	-0.52359813641391	0.00023482550511
4	0.49999999999305	0.00000003416791	-0.52359846718124	0.00001230255246
5	0.50000000000000	0.00000001648704	-0.52359877474410	0.00000030756286
6	0.50000000000000	0.00000000004566	-0.52359877518612	0.00000001644138
7	0.50000000000000	0.00000000002203	-0.52359877559716	0.00000000041103
8	0.50000000000000	0.00000000000006	-0.52359877559775	0.00000000002197
9	0.50000000000000	0.00000000000003	-0.52359877559830	0.00000000000055
10	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000003

El programa que sigue, imprime las iteraciones 10,20,...,100, así como la norma infinito de la diferencia de dos iteraciones consecutivas.

```
%imprime las primeras 100 iteraciones de 10 en 10 y la norma infinito
%entre  dos iteraciones consecutivas.
'condicion inicial',
x=[.1 .1 -.1];
for k=1:100
    y=gtrid(x);
    if fix(k/10)==k/10
       A(fix(k/10),:)=[y norm(x-y,inf)];
    end
    x=y;
end
format long
A
```

RESULTADO

k	x_k	y_k	z_k	$\|\|p^{(k)} - p^{(k-1)}\|\|_\infty$
0	0.10000000000000	0.10000000000000	-0.10000000000000	–
10	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000003
20	0.50000000000000	-0.00000000000000	-0.52359877559830	0
30	0.50000000000000	-0.00000000000000	-0.52359877559830	0
40	0.50000000000000	-0.00000000000000	-0.52359877559830	0
50	0.50000000000000	-0.00000000000000	-0.52359877559830	0
60	0.50000000000000	-0.00000000000000	-0.52359877559830	0
70	0.50000000000000	-0.00000000000000	-0.52359877559830	0
80	0.50000000000000	-0.00000000000000	-0.52359877559830	0
90	0.50000000000000	-0.00000000000000	-0.52359877559830	0
100	0.50000000000000	-0.00000000000000	-0.52359877559830	0

Método de Seidel
Cálculo de una iteración para el método de Seidel. Basta llamar al fichero gtrids.m, definido de la siguiente manera:

```
function g=gtrids(p) %nos devuelve el valor de la iteracion que
%sigue a p por el metodo de Seidel
x=p(1);y=p(2);z=p(3); g=zeros(p);
g(1)=1/3*cos(y*z)+1/6;
x=g(1); %cambia la antigua x por la nueva.
g(2)=1/9*sqrt(x^2+sin(z)+1.06)-0.1;
y=g(2); %cambia la antigua y por la nueva.
g(3)=-1/20*exp(-x*y)-(10*pi-3)/60;
```

A continuación, calculamos las iteraciones del método de Seidel hasta que la norma infinito de dos iteraciones consecutivas es menor que TOL (partiendo de la condicion inicial $[.1\ .1\ -.1]$. Devuelve el número de iteraciones necesarios así como la aproximación alcanzada (número máximo de iteraciones 1000).

```
TOL=10^(-10);
x=[.1 .1 -.1];
y=gtrids(x);
c=1;
while norm(x-y,inf)>=TOL
  x=y;
  y=gtrids(x); %calcula la iteracion que sigue al vector x.
  c=c+1;
  if c>1000 %si en 1000 iteraciones no converge mensaje de error.
     error('no se alcanzo dicha tolerancia')
  end
end
'el numero de iteraciones en alcanzar la TOL', TOL,
'es',
c
'la solucion aproximada es'
format long
y
```

RESULTADO

▷ el numero de iteraciones en alcanzar la TOL = 1.0000e-10, es 5
la solucion aproximada es
0.50000000000000 0.00000000000007 -0.52359877559830.
Otra llamada al fichero con TOL=10^ (-15):
el numero de iteraciones en alcanzar la TOL = 1.0000e-15, es 7
la solucion aproximada es
0.50000000000000 -0.00000000000000 -0.52359877559830 ◁

El código que sigue, imprime las iteraciones 1,2,...,10, así como la norma infinito de la diferencia de dos iteraciones consecutivas

```
x=[.1 .1 -.1];
for k=1:10
    y=gtrids(x);
    A(k,:)=[y norm(x-y,inf)];
    x=y;
end
format long
A
```

RESULTADO

k	x_k	y_k	z_k	$\|\|p^{(k)} - p^{(k-1)}\|\|_\infty$
0	0.10000000000000	0.10000000000000	-0.10000000000000	–
1	0.49998333347222	0.02222979355858	-0.52304612619137	0.42304612619137
2	0.49997746826183	0.00002815366194	-0.52359807179342	0.02220163989664
3	0.49999999996378	0.00000003762202	-0.52359877465775	0.00002811603992
4	0.50000000000000	0.00000000005028	-0.52359877559704	0.00000003757174
5	0.50000000000000	0.00000000000007	-0.52359877559830	0.00000000005021
6	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000007
7	0.50000000000000	-0.00000000000000	-0.52359877559830	0.00000000000000
8	0.50000000000000	-0.00000000000000	-0.52359877559830	0
9	0.50000000000000	-0.00000000000000	-0.52359877559830	0
10	0.50000000000000	-0.00000000000000	-0.52359877559830	0

A continuación, imprimimos las iteraciones 10,20,...,100, así como la norma infinito de la diferencia de dos iteraciones consecutivas.

```
%imprime las primeras 100 iteraciones de 10 en 10 y la norma infinito
%entre  dos iteraciones consecutivas
'condicion inicial',
x=[.1 .1 -.1];
for k=1:100
    y=gtrids(x);
    if fix(k/10)==k/10
       A(fix(k/10),:)=[y norm(x-y,inf)];
    end
    x=y;
end
format long
A
```

RESULTADO

k	x_k	y_k	z_k	$\|\|p^{(k)} - p^{(k-1)}\|\|_\infty$
0	0.10000000000000	0.10000000000000	-0.10000000000000	–
10	0.50000000000000	-0.00000000000000	-0.52359877559830	0
20	0.50000000000000	-0.00000000000000	-0.52359877559830	0
30	0.50000000000000	-0.00000000000000	-0.52359877559830	0
40	0.50000000000000	-0.00000000000000	-0.52359877559830	0
50	0.50000000000000	-0.00000000000000	-0.52359877559830	0
60	0.50000000000000	-0.00000000000000	-0.52359877559830	0
70	0.50000000000000	-0.00000000000000	-0.52359877559830	0
80	0.50000000000000	-0.00000000000000	-0.52359877559830	0
90	0.50000000000000	-0.00000000000000	-0.52359877559830	0
100	0.50000000000000	-0.00000000000000	-0.52359877559830	0

Método de Newton-Raphson
Definición de la función que nos define el sistema, le llamaremos ftrid y está en el fichero ftrid.m que a continuación listamos:

```
%define la funcion tridimensional (llamada ftrid) p(x,y,z)----->f(f1,f2,f3)
function f=ftrid(p)
x=p(1);y=p(2);z=p(3);
f(1)=3*x-cos(y*z)-1/2;
f(2)=x^2-81*(y+0.1)^2+sin(z)+1.06;
f(3)=exp(-x*y)+20*z+(10*pi-3)/3;
```

Definición del Jacobiano de la funcion ftrid, le llamamos jacftrid y está en el fichero jacftrid.m conteniendo:

```
%este fichero jacftrid.m, nos define el jacobiano de ftrid
function JA=jacftrid(p)
x=p(1);y=p(2);z=p(3);
JA(1,:)=[3 z*sin(y*z) y*sin(y*z)];
JA(2,:)=[2*x -162*(y+0.1) cos(z)];
JA(3,:)=[-y*exp(-x*y) -x*exp(-x*y) 20];
```

Calcula las iteraciones de Newton hasta que la norma infinito de dos iteraciones consecutivas es menor que TOL (tanto TOL como la condición inicial se dan por el teclado).
Devuelve el número de iteraciones necesarios así como la aproximación alcanzada (número máximo de iteraciones 1000).

```
%Solucion por el metodo de Newton-Raphson, el sistema esta definido en
%ftrid y el jacobiano de f en jacftrid
'Introducir por teclado la tolerancia'
TOL=input('TOL=');
'Condicion inicial'
x=input('condicion inicial x=');
c=1;
z=-jacftrid(x)\ftrid(x)';
```

```
while norm(z,inf)>=TOL
  x=x+z';
  z=-jacftrid(x)\ftrid(x)';
  c=c+1;
  if c>1000 %si en 1000 iteraciones no converge mensaje de error
     error('no se alcanzo dicha tolerancia')
  end
end
'se alcanzo la tolerancia ', TOL
'en la iteracion,'
c
'solucion aproximada,'
x+z'
```

RESULTADO

▷ Introducir por teclado la tolerancia TOL=10^ (-15)
Condicion inicial x=[.1 .1 -.1]
se alcanzo la tolerancia TOL = 1.000000000000000e-15 en la iteracion, 6
solucion aproximada,
0.50000000000000 0.00000000000000 -0.52359877559830.
Otra llamada al fichero:
Introducir por teclado la tolerancia TOL=10^ (-15)
Condicion inicial x=[1 1 1]
se alcanzo la tolerancia TOL= 1.000000000000000e-15 en la iteracion, 9
solucion aproximada,
0.50000000000000 -0.00000000000000 -0.52359877559830.
Otra llamada:
Introducir por teclado la tolerancia TOL=10^ (-15)
Condicion inicial x=[10 10 10]
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 5.762925e-054.
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 3.847577e-051.
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 1.297916e-050.

.....

Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 4.775966e-028.
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 6.389268e-027.
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 1.616856e-027.
se alcanzo la tolerancia TOL = 1.000000000000000e-15 en la iteracion 48
solucion aproximada,
0.50000000000000 -0.00000000000000 -0.52359877559830. ◁

A continuación, imprimimos las 10 primeras iteraciones, así como la norma infinito de la diferencia de dos iteraciones consecutivas.

```
%Solucion por el metodo de Newton-Raphson, el sistema esta definido en
%ftrid y el jacobiano de f en jacftrid
%imprime las primeras 10 iteraciones de la norma infinito
%entre  dos iteraciones consecutivas
x=[.1 .1 -.1];
for k=1:10
    y=x-(jacftrid(x)\ftrid(x)')';
    A(k,:)=[y norm(x-y,inf)];
    x=y;
end
format long
A
```

RESULTADO

$$p^{(k)} = [x_k, y_k, z_k] :$$

k	x_k	y_k	z_k	$\|\|p^{(k)} - p^{(k-1)}\|\|_\infty$
0	0.10000000000000	0.10000000000000	-0.10000000000000	–
1	0.49986967292643	0.01946684853742	-0.52152047193583	0.42152047193583
2	0.50001424016422	0.00158859137029	-0.52355696434764	0.01787825716712
3	0.50000011346783	0.00001244478332	-0.52359845007289	0.00157614658697
4	0.50000000000708	0.00000000077579	-0.52359877557801	0.00001244400754
5	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000077579
6	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000
7	0.50000000000000	-0.00000000000000	-0.52359877559830	0.00000000000000
8	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000
9	0.50000000000000	-0.00000000000000	-0.52359877559830	0.00000000000000
10	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000

El código que sigue, imprime las iteraciones 10,20,...,100, así como la norma infinito de la diferencia de dos iteraciones consecutivas.

```
%Solucion por el metodo de Newton-Raphson, el sistema esta definido en
%ftrid y el jacobiano de f en jacftrid
%imprime las primeras 100 iteraciones de 10 en 10 y la norma infinito
%entre  dos iteraciones consecutivas
x=[.1 .1 -.1];
for k=1:100
    y=x-(jacftrid(x)\ftrid(x)')';
    if fix(k/10)==k/10
       A(fix(k/10),:)=[y norm(x-y,inf)];
    end
    x=y;
end
format long
A
```

RESULTADO

k	x_k	y_k	z_k	$\|\|p^{(k)} - p^{(k-1)}\|\|_\infty$
0	0.10000000000000	0.10000000000000	-0.10000000000000	–
10	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000
20	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000
30	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000
40	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000
50	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000
60	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000
70	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000
80	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000
90	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000
100	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000

Método de Broyden
En lo que sigue, calculamos las iteraciones del método de Broyden hasta que la norma infinito de dos iteraciones consecutivas es menor que TOL (tanto TOL como la condición inicial se dan por el teclado).
Devuelve el número de iteraciones necesarios así como la aproximación alcanzada (número máximo de iteraciones 1000).

```
%Solucion por el metodo de Broyden, el sistema esta definido en ftrid.
'Introducir por teclado la tolerancia' TOL=input('TOL=');
'Condicion inicial' x=input('condicion inicial x=');
c=1;
B=jacftrid(x);
z=-B\ftrid(x)';
while norm(z,inf)>=TOL
  xa=x;
  x=x+z';
  B=B+1/norm(x-xa)^2*(ftrid(x)'-ftrid(xa)'-B*(x-xa)')*(x-xa);
  z=-B\frtid(x)';
  c=c+1;
  if c>1000 %si en 1000 iteraciones no converge mensaje de error.
     error('no se alcanzo dicha tolerancia')
  end
end
'se alcanzo la tolerancia,'
TOL
'en la iteracion,'
c
'la solucion aproximada es,'
format long
x+z'
```

RESULTADO

▷ Introducir por teclado la tolerancia TOL=10^ (-15) Condicion inicial x=[.1 .1 -.1]
se alcanzo la tolerancia TOL = 1.000000000000000e-15 en la iteracion, 10 la solucion aproximada es, 0.50000000000000 0.00000000000000 -0.52359877559830.

Otra llamada al fichero:
Introducir por teclado la tolerancia TOL=10^ (-15) Condicion inicial x=[1 1 1]
se alcanzo la tolerancia TOL=1.000000000000000e-15 en la iteracion, 15 la solucion aproximada es,
0.50000000000000 0.00000000000000 -0.52359877559830.
Otra llamada:
Introducir por teclado la tolerancia TOL=10^ (-15) Condicion inicial [10 10 10]
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 7.453521e-051.
¿ In problema3a4tema2 at 12
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 7.163188e-051.
¿ In problema3a4tema2 at 12
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 1.696540e-051.

.....

¿ In problema3a4tema2 at 12
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 3.509951e-051.
¿ In problema3a4tema2 at 12
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 3.509417e-051.

¿ In problema3a4tema2 at 12 se alcanzo la tolerancia,
TOL = 1.000000000000000e-015 en la iteracion,
c = 119 la solucion aproximada es,
0.13814060609138 -0.03349091869652 49.46055157157920. ◁

A continuación, imprimimos las iteraciones 1,2,...,10, así como la norma infinito de la diferencia de dos iteraciones consecutivas.

```
x=[.1 .1 -.1];
B=jacftrid(x);
z=-B\ftrid(x)';
xa=x;
x=x+z';
A(1,:)=[x norm(x-xa,inf)];
for k=2:10
    B=B+1/norm(x-xa)^2*(ftrid(x)'-ftrid(xa)'-B*(x-xa)')*(x-xa);
    z=-B\ftrid(x)';
    xa=x;
    x=x+z';
    A(k,:)=[x norm(x-xa,inf)];
end
format long
A
```

RESULTADO

k	x_k	y_k	z_k	$\|\|p^{(k)} - p^{(k-1)}\|\|_\infty$
0	0.10000000000000	0.10000000000000	-0.10000000000000	–
1	0.49986967292643	0.01946684853742	-0.52152047193583	0.42152047193583
2	0.49998637545691	0.00873783929926	-0.52317457439975	0.01072900923816
3	0.50000659705997	0.00086727355579	-0.52357234148640	0.00787056574347
4	0.50000032871755	0.00003952827531	-0.52359768537883	0.00082774528048
5	0.50000000156688	0.00000019354398	-0.52359877005998	0.00003933473133
6	0.50000000000033	0.00000000000053	-0.52359877559910	0.00000019354344
7	0.50000000000000	0.00000000000017	-0.52359877559829	0.00000000000081
8	0.50000000000000	-0.00000000000001	-0.52359877559830	0.00000000000018
9	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000001
10	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000

Por último, imprimimos las iteraciones 10,20,...,100, así como la norma infinito de la diferencia de dos iteraciones consecutivas.

```
x=[.1 .1 -.1];
B=JFnewton(x);
z=-B\Fnewton(x)';
xa=x;
x=x+z';
A(1,:)=[x norm(x-xa,inf)];
for k=2:100
    B=B+1/norm(x-xa)^2*(Fnewton(x)'-Fnewton(xa)'-B*(x-xa)')*(x-xa);
    z=-B\Fnewton(x)';
    xa=x;
    x=x+z';
    if fix(k/10)==k/10
       A(fix(k/10),:)=[x norm(x-xa,inf)];
    end
end
format long
A
```

RESULTADO

k	x_k	y_k	z_k	$\|\|p^{(k)} - p^{(k-1)}\|\|_\infty$
0	0.10000000000000	0.10000000000000	-0.10000000000000	–
10	0.50000000000000	0.00000000000000	-0.52359877559830	0.00000000000000
20	0.50000000000000	0.00000000000000	-0.52359877559830	0.00179924699531
30	NaN	NaN	NaN	NaN
40	NaN	NaN	NaN	NaN
50	NaN	NaN	NaN	NaN
60	NaN	NaN	NaN	NaN
70	NaN	NaN	NaN	NaN
80	NaN	NaN	NaN	NaN
90	NaN	NaN	NaN	NaN
100	NaN	NaN	NaN	NaN

2.5 Ejercicios Propuestos Tema 2

1. Programar la función:

$$f(x, A, X) = a_0 + a_1(x - x_1) + a_2(x - x_1)(x - x_2) + ... + a_n(x - x_1)(x - x_2)...(x - x_n),$$

donde $A = [a_0, a_1, ..., a_n]$, $X = [x_1, x_2, ..., x_n]$, con $x \in \mathbb{R}$.

Calcular todas las raíces de dicha función para $x \in [0, 3]$, en el caso particular de ser $A = [1, 2, 3, 4, 5]$, $X = [0, 1, 2, 3]$.

2. Los problemas relacionados con la cantidad de dinero requerida para pagar una hipoteca en un periodo fijo de tiempo involucran la fórmula

$$A = \frac{P}{i}[1 - (1 + i)^{-n}].$$

En esta ecuación, A es el importe de la hipoteca, P es el importe de cada pago e i es la tasa de interés por periodo, para los n periodos de pago. Supongamos que se necesita una hipoteca de 135000 euros para una casa, con un periodo de 30 años, y que los pagos máximos que puede realizar el cliente son de 1000 euros mensuales. Cuál será el interés más alto que podrá pagar?. Utilizar el algoritmo de punto fijo y obtener las aproximaciones correspondientes a las iteraciones $5, 10, 15, ..., k$ donde k corresponde a la última iteración.

Indicación: El interés x viene dado por $x = 1200i$.

3. Se pide:

   a) Hallar todas las raíces de la ecuación $x^3 - 7x + 2 = 0$. Realizar un análisis gráfico para calcular las aproximaciones iniciales necesarias en el método de la secante, tomando como criterio de paro $|p_n - p_{n-1}| < 10^{-7}$ y 100 como número máximo de iteraciones. Construye un gráfico que muestre la convergencia de las iteraciones a una de las raíces existentes.

   b) Siendo $F(p)$ la única raíz de la ecuación $x^3 - x^2 - p = 0$. Realizar un programa que, utilizando el método de Newton, calcule el vector $[F(1), F(1.4), F(1.8), F(2)]$ y las iteraciones necesarias para obtener cada una de ellas. Tomar $x_0 = 2$ y como criterio de paro $|p_k - p_{k-1}| < 10^{-5}$.

   c) Repetir el apartado anterior utilizando el algoritmo de Steffesen con las mismas especificaciones y la función $g(x) = x^3 - x^2 + x - p$.

4. Calcular el mínimo y el máximo discreto de la función

$$h(x, y) = (x + y - 1)^2 + (2x - y - 2)^2,$$

así como los puntos (x, y) donde se alcanzan dicho mínimo y máximo, cuando $x, y \in [0, 1]$. Usar una malla de 10000 puntos.

5. Resolver el sistema $Ax = b$, cuadrado de orden 100, donde $A = (a_{ij})$, $b = (b_i)$ con $a_{ij} = \dfrac{1}{i+j}$, $b_i = i^2 - 1$, e $i, j = 1, ..., 100$. Utilizar el método de Jacobi siendo el vector inicial el origen, tolerancia 10^{-15}, medida en la norma euclidea, y máximo número de iteraciones 100. Estudiar previamente la convergencia del método.

6. Consideremos el sistema de ecuaciones $Ax = b$, dado por

$$\begin{pmatrix} 10 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 4 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 4 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 4 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 4 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 4 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 4 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} = \begin{pmatrix} 9 \\ 5 \\ -2 \\ 2 \\ 1 \\ 6 \\ 2 \\ -10 \end{pmatrix}.$$

   a) Resolverlo utilizando tanto el método de Jacobi como el de Gauss-Seidel con las siguientes especificaciones para ambos casos: el vector inicial es el origen, la tolerancia es 10^{-10}, medida en la norma euclidea, y el número máximo de iteraciones que se permite es 100. Convergen ambos métodos para las especificaciones dadas?.

   b) Estudiar la eficacia de ambos métodos obteniendo una gráfica donde se represente la tolerancia 10^{-i}, con $i = 1, 2, ..., 14$, frente al número de iteraciones necesarias para alcanzar esa tolerancia.

   c) Encontrar el ω_{optimo} del método SOR para el sistema dado, en el sentido de que ω_{optimo} se corresponda con el mínimo valor de k para el que $||x^k - x^{k-1}||_2 <$ tolerancia. Tomar como vector inicial el origen, la tolerancia $= 10^{-10}$ y $k \leq 1000$. Dibujar k frente a $\omega \in (0, 2)$, cuando $k > 1000$ o no converja el método para un cierto ω, tomar $k = 1000$. Hallar el mínimo discreto de $\rho(T_\omega)$ y comparar la gráfica anterior con la obtenida al representar $\rho(T_\omega)$ frente a ω.

   d) Resolver el sistema dado mediante el método SOR para el ω_{optimo} obtenido en el apartado anterior con las especificaciones del apartado (a). Calcular las iteraciones 5, 10, ...

7. Sea el sistema de ecuaciones no lineal

$$\begin{cases} F(x, y) = xy - x - y + 1 = 0, \\ G(x, y) = 3x^2y - x^2 - 12xy + 4x + 12y - 4 = 0, \end{cases}$$

   se pide:

   a) Aplicar el método de Seidel con las siguientes especificaciones: el vector inicial es el origen, la tolerancia es 10^{-10}, medida en la norma infinito, y el número máximo de iteraciones que se permite es 100.

   b) Aplicar el método de máximo descenso, partiendo de $P = [0, 0]$, hasta obtener una aproximación que cumpla $h(x, y) = F^2(x, y) + G^2(x, y) < 10^{-1}$.

   c) Con la aproximación obtenida en el apartado (b) aplicar el método de Newton con criterio de paro $||P_n - P_{n-1}||_2 < 10^{-10}$. Qué respuesta obtienes?.

8. Dado el sistema de ecuaciones no lineal:

$$\begin{cases} f(x,y) = 3x + xy + y^3 = 0, \\ g(x,y) = 9x^2 + xy^3 + y^4 + y - 1 = 0. \end{cases}$$

a) Aplicar el algoritmo de punto fijo partiendo de $P = [0,0]$ con tolerancia 10^{-5}, medida en la norma euclidea, y el número máximo de iteraciones igual a 100, siendo la función de iteración $G(x,y) = \left(g_1(x,y) = \frac{-xy-y^3}{3}, \quad g_2(x,y) = -9x^2 - xy^3 - y^4 + 1\right)$.

b) Hallar el punto (x_0, y_0), con $x_0 = -10 + i, \quad y_0 = -10 + j$, e $i, j = 1, 2, ..., 20$, donde se alcanza el valor mínimo de la función $h(x,y) = f^2(x,y) + g^2(x,y)$.

c) Tomar el punto obtenido en el apartado (b) como punto inicial para aplicar el método de Newton con criterio de paro $h(x,y) \leq 10^{-10}$ ó número máximo de iteraciones igual a 50. Converge?. En caso afirmativo decir en cuantas iteraciones lo hace, y hallar las tres últimas aproximaciones calculadas, encontrando el valor de h en cada una de ellas.

d) Tomar el punto obtenido en el apartado (b) como punto inicial para aplicar el método de Broyden con tolerancia 10^{-10}, en la norma infinito, y número máximo de iteraciones igual a 50.

Soluciones

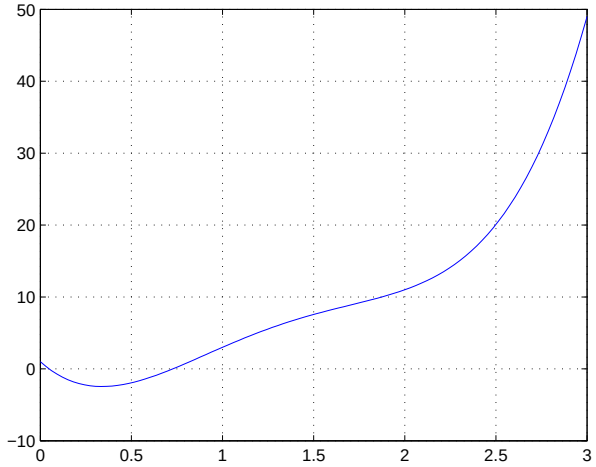

Figura 2.6: Gráfica de la función $y = f(x)$ del problema 1.

1. En la figura 2.6 se observa que hay dos raíces en el intervalo $[0, 3]$. Utilizando el algoritmo de bisección en los intervalos $[0, 0.5]$ y $[0.5, 1]$, y tolerancia 10^{-10} para la longitud del último intervalo se obtiene : $r_1 \approx 0.04795390772051$ y $r_2 \approx 0.72721331214416$.

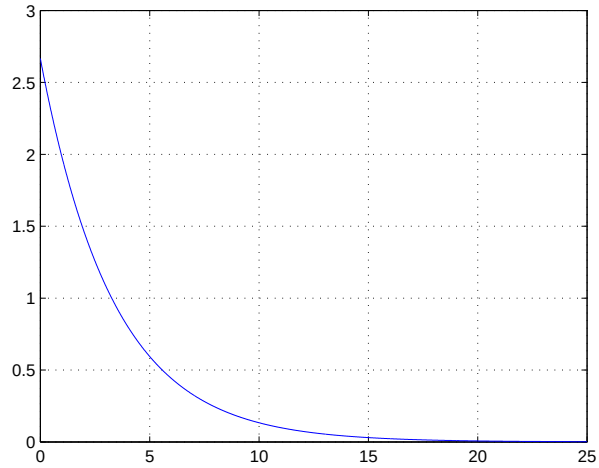

Figura 2.7: Gráfica de la función $y = |g'(x)|$ del problema 2.

2. En la figura 2.7 se observa que la gráfica de la función valor absoluto de la derivada de la función $g(x) = \frac{1200\times1000}{135000}[1-(1+\frac{x}{1200})^{-30\times12}]$ es menor que 1 en el intervalo $[0, 25]$. Utilizando el algoritmo de punto fijo con la función g, condición inicial $p_0 = 1$, tolerancia 10^{-7} y condición de parada $f(p_k) = 135000 - \frac{1200\times1000}{p_k}[1-(1+\frac{p_k}{1200})^{-30\times12}] < tolerancia$ y $|p_k - p_{k-1}| < tolerancia$, el interés máximo que puede pagar es $x \approx 8.0999\%$ obtenido en la iteración $k = 22$. La matriz con las aproximaciones para $k = 5, 10, 15, 20, 22$ en la primera columna, el valor de $f(p_k)$ en la segunda, así como la diferencia $p_k - p_{k-1}$ en la tercera es:

$$\begin{pmatrix} 7.98108249399 & -1528.79407496 & 0.35184526669 \\ 8.09981321825 & -1.11386278889 & 0.00028423198 \\ 8.09990052812 & -0.00080017894 & 0.00000020420 \\ 8.09990059084 & -0.00000057530 & 0.00000000015 \\ 8.09990059088 & -0.00000003132 & 0.00000000000 \end{pmatrix}.$$

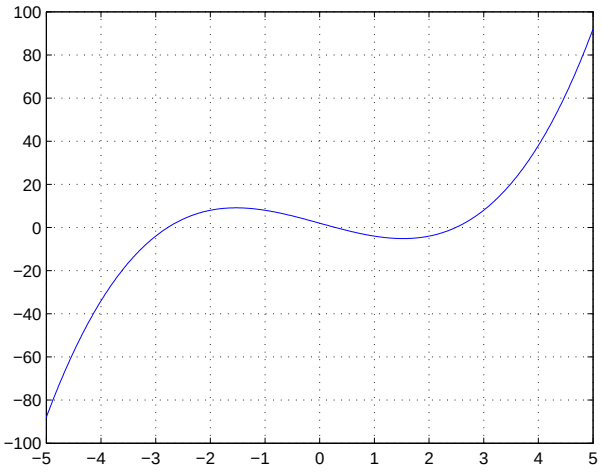

Figura 2.8: Gráfica de la función $y = f(x)$ del problema 3a.

3. a) Observando la figura 2.8 tomamos $p_0 = -4, p_1 = -3$ obteniéndose, en la iteración 8, la raíz $r_1 \approx -2.77845711825839$. Para la elección $p_0 = 0, p_1 = 1$ se obtiene, en la iteración 7, $r_2 \approx 0.28916854644787$ y finalmente si $p_0 = 3, p_1 = 4$ obtenemos, en la iteración 9, $r_3 \approx 2.48928857181016$.

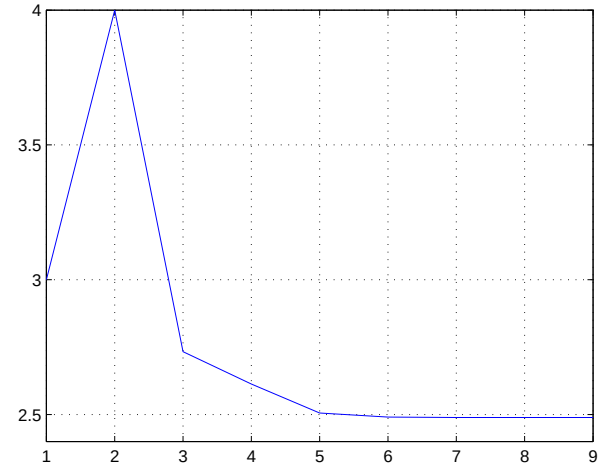

Figura 2.9: Evolución de las 9 primeras iteraciones para la raíz r_3 del problema3a.

b) El vector con las aproximaciones de las raíces es:
[1.46557123187679, 1.56882527131585, 1.65620903538834, 1.69562076957059], y las iteraciones necesarias para obtener cada una de ellas son: $[5, 5, 5, 4]$.

c) El vector con las aproximaciones de las raíces es:
[1.46557123191001, 1.56882527154556, 1.65620903538834, 1.69562076958491], y las iteraciones necesarias para obtener cada una de ellas son: $[9, 8, 8, 7]$.

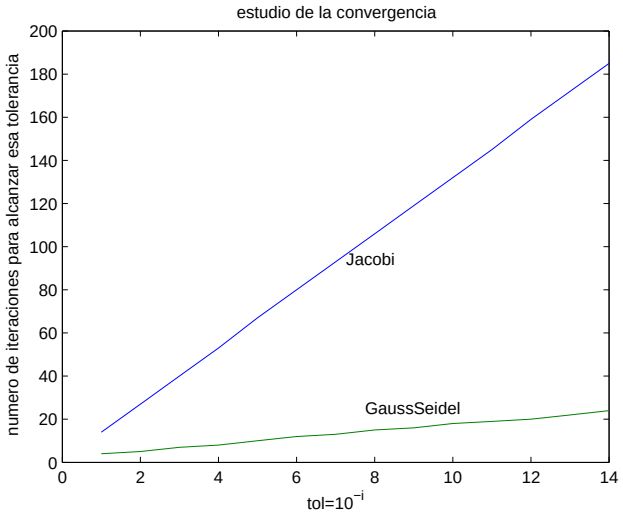

Figura 2.10: Gráfica del problema 6 apartado (b).

4. El mínimo discreto de la función $h(x, y)$, se obtiene en el punto $(1, 0)$ y su valor es 0. El máximo dicreto lo alcanza en el punto $(0, 1)$ y su valor es 9.

5. La sucesión generada por el método de Jacobi no converge a la solución, ya que el radio espectral de la matriz del método es 86.75777558.

   Si bajamos el orden a 2, es decir $i, j = 1, 2$, entonces el método de Jacobi converge para este nuevo sistema, ya que el radio espectral es 0.94280904, obteniendose como solución: $[-72, \quad 108]$.

6. a) Con las especificaciones dadas, el método de Jacobi no converge. Con el método de Gauss-Seidel si hay convergencia, obteniendose en la iteración 17:

$$\begin{pmatrix} 0.70499911675086 \\ 1.67234092056811 \\ -0.98825080461501 \\ 0.98177517546556 \\ -0.23773701965394 \\ 1.67028578075603 \\ 0.25770677421194 \\ -1.40611199434835 \end{pmatrix}.$$

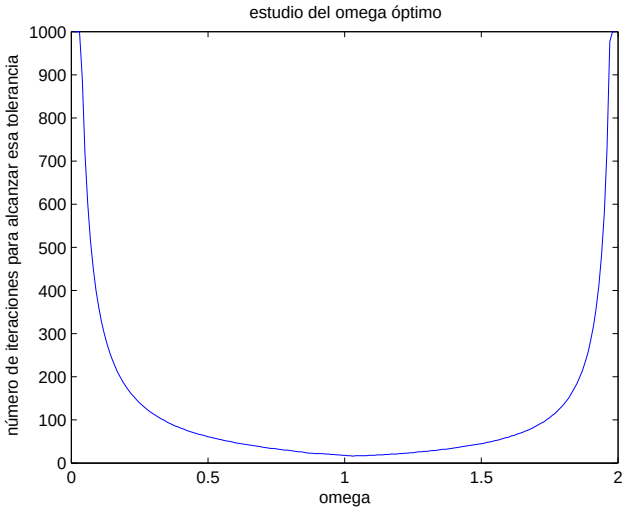

Figura 2.11: Primera gráfica del problema 6 apartado (c).

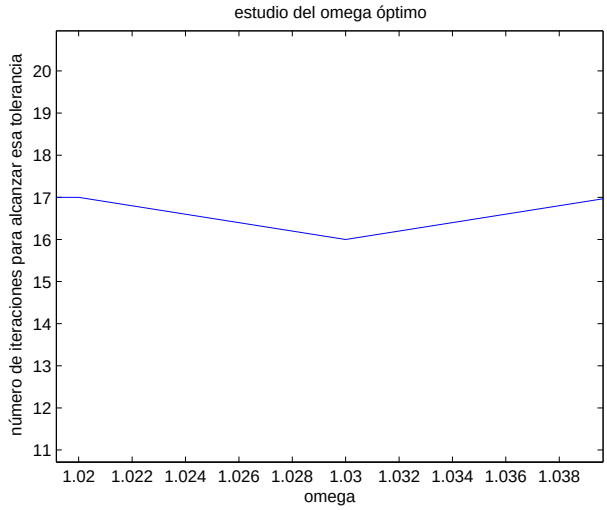

Figura 2.12: Zoom de la primera gráfica del problema 6 apartado (c).

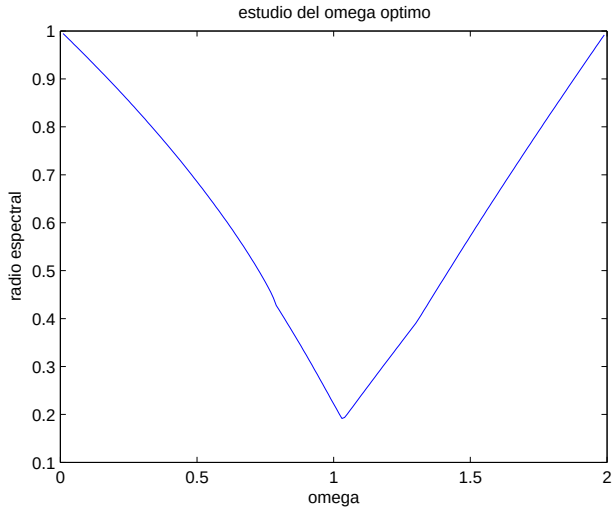

Figura 2.13: Segunda gráfica del problema 6 apartado (c).

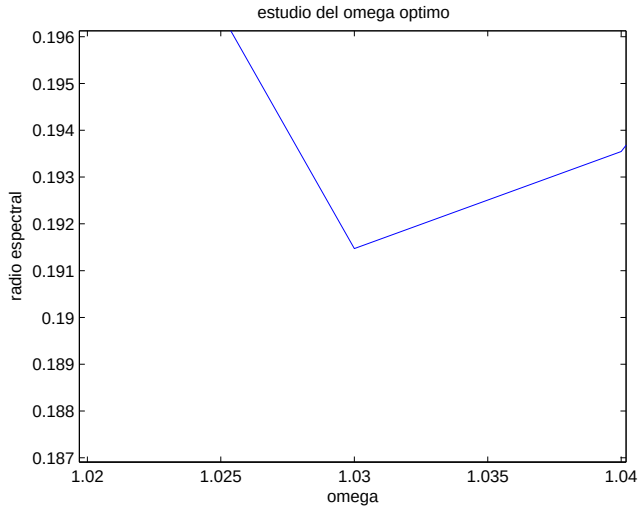

Figura 2.14: Zoom de la segunda gráfica del problema 6 apartado (c).

El $\omega_{optimo} \approx 1.03$, se corresponde con $k = 16$ iteraciones.

El mínimo discreto de $\rho(T_\omega) \approx 0.1914729277$ se obtiene para $\omega_{optimo} \approx 1.03$.

d) Para $\omega = 1.03$, el método SOR da como aproximación a la solución en la iteración 16:

$$\begin{pmatrix} 0.70499911676823 \\ 1.67234092053369 \\ -0.98825080460557 \\ 0.98177517545630 \\ -0.23773701965450 \\ 1.67028578075220 \\ 0.25770677420430 \\ -1.40611199434565 \end{pmatrix}.$$

La matriz con las iteraciones 5,10 y 15 es:

$$\begin{pmatrix} 5.00000000000000 & 10.00000000000000 & 15.00000000000000 \\ 0.70520057890389 & 0.70499891042941 & 0.70499911672859 \\ 1.67104095096625 & 1.67234102587039 & 1.67234092046753 \\ -0.98828747605606 & -0.98825103602589 & -0.98825080455592 \\ 0.98249314791357 & 0.98177540496771 & 0.98177517545394 \\ -0.23838746894696 & -0.23773711890423 & -0.23773701964823 \\ 1.67009603271334 & 1.67028572470848 & 1.67028578073209 \\ 0.25757572587607 & 0.25770684435928 & 0.25770677421918 \\ -1.40598190080799 & -1.40611197279274 & -1.40611199433937 \end{pmatrix}.$$

7. a) El método de Seidel no converge con las especificaciones dadas.

   b) La aproximación obtenida por el método de máximo descenso en la iteración 14 es:

$$\begin{pmatrix} x \\ y \end{pmatrix} \approx \begin{pmatrix} 0.96501683696697 \\ 0.40415353559737 \end{pmatrix}.$$

   c) Con $P = [0.9, 0.4]$, mediante el método de Newton se consigue la tolerancia exigida en la iteración 5:

$$\begin{pmatrix} x \\ y \end{pmatrix} \approx \begin{pmatrix} 1.00000000000000 \\ 0.33333333333333 \end{pmatrix}.$$

8. a) El método de punto fijo no converge con las especificaciones dadas.

   b) El punto donde se alcanza el mínimo es $(x_0, y_0) = (0, 0)$, siendo $h(0, 0) = 1$.

   c) Si converge con las especificaciones dadas en la iteración 5 y las tres últimas aproximaciones y el valor de h en cada una de ellas son:

$$\begin{pmatrix} iteracion \\ x \\ y \\ h(x,y) \end{pmatrix} \approx \begin{pmatrix} 3.00000000000000 & 4.00000000000000 & 5.00000000000000 \\ -0.09473975920591 & -0.09463792103603 & -0.09465788363288 \\ 0.72113892298610 & 0.70553809724951 & 0.70521838437809 \\ 0.00186212320267 & 0.00000075696059 & 0.00000000000014 \end{pmatrix}.$$

   d) Utilizando el método de Broyden converge, con las especificaciones dadas, en la iteración 12 y la última aproximación es:

$$\begin{pmatrix} x \\ y \end{pmatrix} \approx \begin{pmatrix} -0.09465789215212 \\ 0.70521824759221 \end{pmatrix}.$$