RESOLUCION NUMÉRICA DE ECUACIONES Y SISTEMAS DE ECUACIONES

1. Resolución de ecuaciones no lineales

Sea la ecuación no lineal $f(x) = 0$. A partir de un valor inicial p_0, cercano a una solución p de la ecuación dada, el siguiente algoritmo proporciona, bajo ciertas condiciones, una aproximación a la solución exacta p.

Algoritmo de Newton-Raphson

$$(NR) \qquad \begin{cases} p_0 \approx p, \\ p_{n+1} = p_n - \dfrac{f(p_n)}{f'(p_n)}, \ n \geq 0, \end{cases}$$

2. Resolución de sistemas de ecuaciones no lineales

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

$$\begin{cases} f_1(x_1, x_2, \cdots, x_n) = 0, \\ f_2(x_1, x_2, \cdots, x_n) = 0, \\ \qquad\qquad \vdots \\ f_n(x_1, x_2, \cdots, x_n) = 0, \end{cases}$$

donde $f_i : \mathbb{R}^n \longmapsto \mathbb{R}$ para $i = 1, 2, \cdots, n$. Alternativamente podemos representar nuestro sistema como $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, con $\mathbf{f} : \mathbb{R}^n \longmapsto \mathbb{R}^n$, donde $\mathbf{f} = (f_1, f_2, \cdots, f_n)$. A partir de un vector inicial $\mathbf{p_0}$, cercano a una solución $\mathbf{p}$ de la ecuación dada, el siguiente algoritmo proporciona, bajo ciertas condiciones, una aproximación a la solución exacta $\mathbf{p}$.

Algoritmo de Newton-Raphson para sistemas no lineales

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

3. Resolución de sistemas de ecuaciones lineales

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

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

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

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

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

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

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

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

Algoritmo de Jacobi $\quad (\mathbf{T} = \mathbf{D^{-1}}(\mathbf{L} + \mathbf{U}))$

(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

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

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

Algoritmo de Gauss-Seidel $\quad (\mathbf{T} = (\mathbf{D} - \mathbf{L})^{-\mathbf{1}}\mathbf{U}))$

(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

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

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.

Algoritmos de relajación $\quad (\mathbf{T} = (\omega\mathbf{L} + \mathbf{D})^{-\mathbf{1}}((\omega - 1)\mathbf{D} + \omega\mathbf{U}))$

(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

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

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

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

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 de los criterios:

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

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

Referencias básicas: Burden y Faires [2002]; Mathews y Fink [2000].

4. Ejercicios Resueltos

PROBLEMA 1:

Se trata de encontrar la única solución en $[0,1]$ de la ecuación $f(x) = x^7 + 5x^5 + 2x^2 + x - 1 = 0$.

(1) Construir un gráfico de $y = f(x)$ en el intervalo $[-10, 10]$. (Con ello sacar una idea de las raices de $f(x) = 0$).

(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) Aplicar Newton-Raphson con $p_0 = 0{,}4$ (condición inicial), $N_{max} = 200$ (número máximo de iteraciones) y como criterio de paro tomar: $|p_n - p_{n-1}| < 10^{-10}$ y $|f(p_n)| < 10^{-10}$.

SOLUCIÓN PROBLEMA 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;
```

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 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. ◁

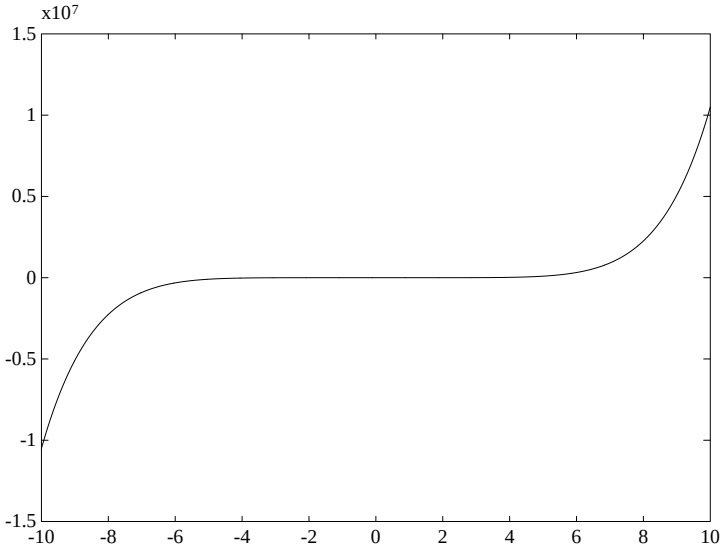

Figura 1: Representación de $y = x^7 + 5x^5 + 2x^2 + x - 1$ en $[-10, 10]$.

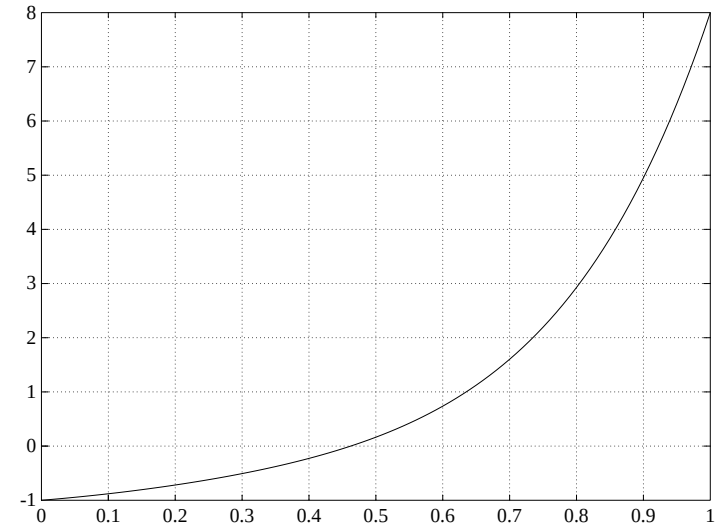

Figura 2: Representación de $y = x^7 + 5x^5 + 2x^2 + x - 1$ en $[0, 1]$.

Apartado (3)

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

```
format long
N=200; %numero maximo de iteraciones
TOL=10^(-10);
p0=0.4;%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. ◁

PROBLEMA 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:

El sistema que se obtiene es

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0{,}75 & -1 & 0{,}25 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0{,}75 & -1 & 0{,}25 & \dots & 0 & 0 & 0 \\ \vdots & & & & & & & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0{,}75 & -1 & 0{,}25 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{19} \\ x_{20} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}.$$

Apartado (1)

En primer lugar generamos el sistema en Matlab

```
function [A,B]=sistemaproblema2(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

>> sistemaproblema2(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=sistemaproblema2(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 sistemaproblema2
[A,B]=sistemaproblema2(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
  % condición de paro
     ea=abs(norm(X'-P));
  if ea<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=sistemaproblema2(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 sistemaproblema2
[A,B]=sistemaproblema2(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));
  if ea<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=sistemaproblema2(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 sistemaproblema2
[A,B]=sistemaproblema2(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));
   if ea<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=problema2a5
% 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]=sistemaproblema2(21);
S1=A\B;
format long e
error=[abs(X'-S2),abs(X'-S3),abs(X'-S4),abs(X'-S1)];
```

RESULTADO

$$\begin{pmatrix}
0 & 0 & 7{,}54951656e-015 & 0 \\
4{,}01900734e-014 & 3{,}59268170e-013 & 1{,}90147897e-012 & 0 \\
1{,}60649271e-013 & 1{,}05071507e-012 & 4{,}28035384e-012 & 1{,}11022302e-016 \\
3{,}49609230e-013 & 2{,}28517205e-012 & 7{,}08844094e-012 & 1{,}11022302e-016 \\
9{,}16267062e-013 & 4{,}37927472e-012 & 1{,}03793640e-011 & 2{,}22044604e-016 \\
1{,}63180580e-012 & 7{,}79698527e-012 & 1{,}40816247e-011 & 1{,}11022302e-016 \\
3{,}77853304e-012 & 1{,}32016619e-011 & 1{,}81976655e-011 & 3{,}33066907e-016 \\
6{,}16029449e-012 & 2{,}15165663e-011 & 2{,}26064722e-011 & 4{,}44089209e-016 \\
1{,}33055788e-011 & 3{,}39875905e-011 & 2{,}72417643e-011 & 4{,}44089209e-016 \\
2{,}04540828e-011 & 5{,}22357712e-011 & 3{,}19138049e-011 & 4{,}44089209e-016 \\
4{,}18992618e-011 & 7{,}82715003e-011 & 3{,}64713814e-011 & 4{,}44089209e-016 \\
6{,}12595529e-011 & 1{,}14422804e-010 & 4{,}06413791e-011 & 6{,}66133814e-016 \\
1{,}19340537e-010 & 1{,}63090430e-010 & 4{,}41726655e-011 & 6{,}66133814e-016 \\
1{,}65525371e-010 & 2{,}26193730e-010 & 4{,}66831018e-011 & 5{,}55111512e-016 \\
3{,}04079761e-010 & 3{,}04079761e-010 & 4{,}78460604e-011 & 5{,}55111512e-016 \\
3{,}93550969e-010 & 3{,}93550969e-010 & 4{,}71749306e-011 & 5{,}55111512e-016 \\
6{,}61964705e-010 & 4{,}84482787e-010 & 4{,}42630376e-011 & 5{,}55111512e-016 \\
7{,}57278351e-010 & 5{,}54252754e-010 & 3{,}85654841e-011 & 5{,}55111512e-016 \\
1{,}04321951e-009 & 5{,}58857515e-010 & 2{,}96310753e-011 & 5{,}55111512e-016 \\
7{,}82414688e-010 & 4{,}19143164e-010 & 1{,}69153580e-011 & 4{,}44089209e-016 \\
0 & 0 & 0 & 0
\end{pmatrix}$$

PROBLEMA 3: Se trata de resolver el sistema:

$$\begin{cases}
3x - cos(yz) - \frac{1}{2} = 0, \\
x^2 - 81(y + 0{,}1)^2 + sen(z) + 1{,}06 = 0, \\
e^{-xy} + 20z + \frac{10\pi - 3}{3} = 0.
\end{cases}$$

Estudiar la convergencia del método de Newton-Raphson 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.

SOLUCIÓN PROBLEMA 3:

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.
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

5. Ejercicios Propuestos

1. 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 Newton, tomando como criterio de paro $|p_n - p_{n-1}| < 10^{-7}$ y 50 como número máximo de iteraciones. Construye un gráfico que muestre la convergencia de las iteraciones a una de las raíces existentes.

2. Consideremos el sistema de ecuaciones $\mathbf{Ax} = \mathbf{b}$, 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) Resolver el sistema dado mediante el método SOR para $\omega = 1.03$ obtenido en el apartado anterior con las especificaciones del apartado (a).

3. Sea el sistema de ecuaciones no lineal

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

   Aplicar el método de Newton con criterio de paro $||\mathbf{P_n} - \mathbf{P_{n-1}}||_2 < 10^{-10}$ y las condiciones iniciales $\mathbf{P} = [0.9, 0.4]$, $\mathbf{P} = [1.8, 0.9]$.

6. Soluciones Ejercicios Propuestos

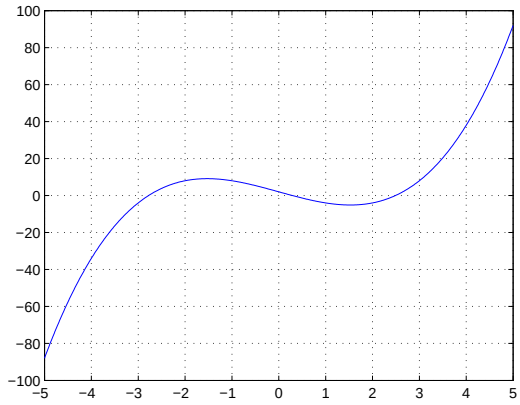

Figura 3: Gráfica de la función $y = f(x)$ del problema 1.

1. Observando la figura 3 tomamos $p_0 = -3$, obteniéndose en la iteración 4 la raíz $r_1 \approx -2{,}77845711825839$. Para la elección $p_0 = 0$ se obtiene, en la iteración 4, $r_2 \approx 0{,}28916854644831$ y finalmente si $p_0 = 3$ obtenemos, en la iteración 5, $r_3 \approx 2{,}48928857181008$.

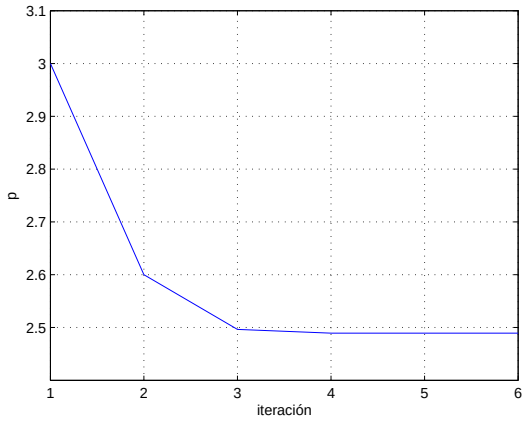

Figura 4: Evolución de las 6 primeras iteraciones para la raíz r_3 del problema1.

Para realizar el gráfico necesitamos guardar todas las iteraciones en un vector. El siguiente programa guarda todas las aproximaciones en el vector P y dibuja el mismo.

```
function p=newton(f,derf,p0,tol,max)
%
%         Datos de entrada para el metodo de newton
% f es la función introducida como una cadena de caracteres 'f' que toma
% desde el fichero f.m
% derf es la derivada de f introducida como una cadena de caracteres
% 'derf' que toma desde el fichero derf.m
% p0 es el valor inicial
% tol es la tolerancia para la solución p
% max es el número máximo de iteraciones
%
%         Datos de salida
% Dibuja las iteraciones realizadas frente a los valores de las
% aproximaciones que figuran en el vector P
%
format long
P(1)=p0;
for k=2:max
    P(k)=P(k-1)-feval(f,P(k-1))/feval(derf,P(k-1));
    err=abs(P(k)-P(k-1));
    if err < tol
       break
    end
end
plot(P)
grid
```

La gráfica de la figura 4 se obtiene escribiendo en la ventana de comandos

```
>>newton('f','derf',3,10^(-7),50)
```

2. 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:

$$\begin{pmatrix} 0{,}70499911675086 \\ 1{,}67234092056811 \\ -0{,}98825080461501 \\ 0{,}98177517546556 \\ -0{,}23773701965394 \\ 1{,}67028578075603 \\ 0{,}25770677421194 \\ -1{,}40611199434835 \end{pmatrix}.$$

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

$$\begin{pmatrix} 0{,}70499911676823 \\ 1{,}67234092053369 \\ -0{,}98825080460557 \\ 0{,}98177517545630 \\ -0{,}23773701965450 \\ 1{,}67028578075220 \\ 0{,}25770677420430 \\ -1{,}40611199434565 \end{pmatrix}.$$

3. Con $\mathbf{P} = [0.9, 0.4]$, mediante el método de Newton se consigue la tolerancia exigida en la iteración 5 y la aproximación a la solución es:

$$\begin{pmatrix} x \\ y \end{pmatrix} \approx \begin{pmatrix} 1{,}00000000000000 \\ 0{,}33333333333333 \end{pmatrix}.$$

Con $\mathbf{P} = [1.8, 0.9]$, mediante el método de Newton se consigue la tolerancia exigida en la iteración 39 y la aproximación a la solución es:

$$\begin{pmatrix} x \\ y \end{pmatrix} \approx \begin{pmatrix} 2{,}00000000125992 \\ 1{,}00000000000000 \end{pmatrix}.$$