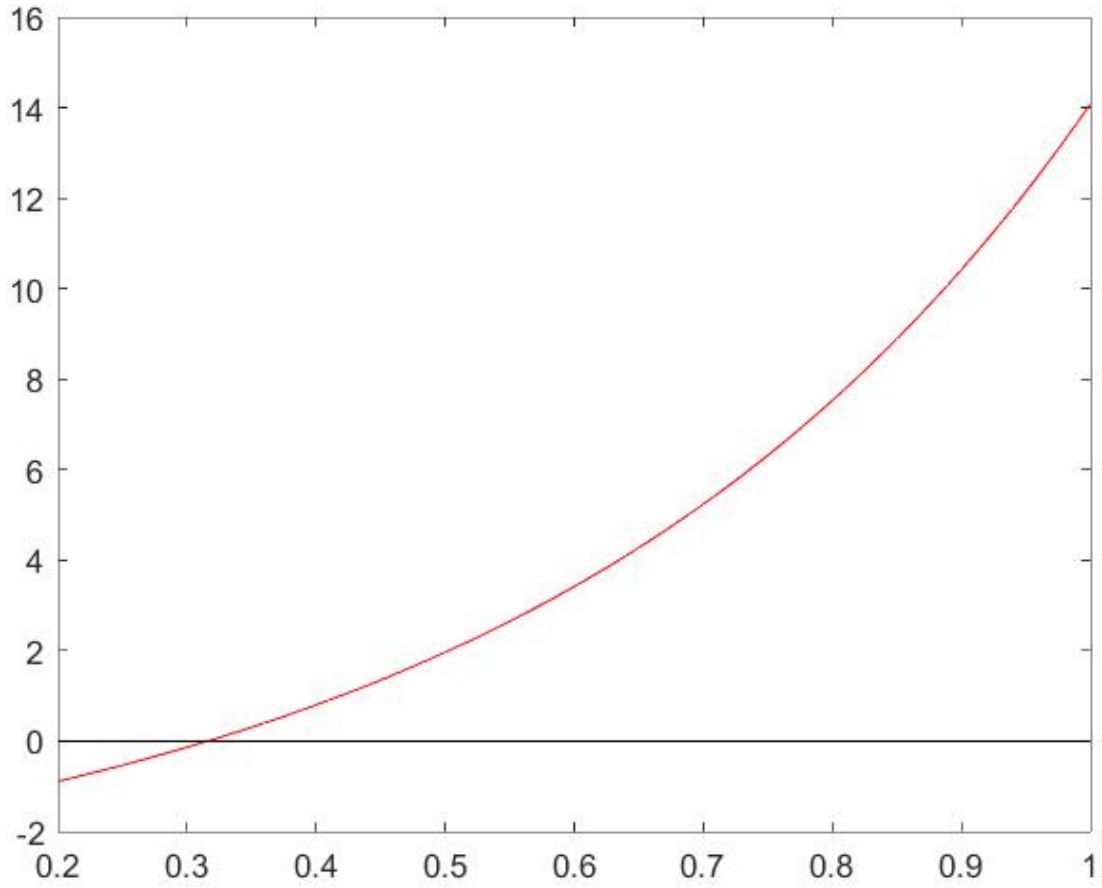
PRÁCTICA DE ALGORÍTMICA NUMÉRICA (18/19)
APELLIDOS, NOMBRE: Parra García, Alejandro C.
APELLIDOS, NOMBRE: Revuelta Martínez,Álvaro

Parte I. Cálculo solución de f(x)= (exp(3x)-1)/x-5=0 en intervalo [0.2,1].

```
x=[0.2:0.01:1];
fx=(exp(3.*x)-1)./x-5;
```

1. – Representación grafica de y=f(x) (en rojo), y=0 (en negro) en el intervalo [0.2,1] .

```
y=0*x;
plot(x,fx,'r',x,y,'k');
```



r (f(r)=0) existe pues ambas curvas se intersectan. Valor aproximado: 0.315.

- Justificar formalmente que existe una raíz de f(x) en el intervalo señalado.

```
f0_2=(exp(3*0.2)-1)/0.2-5;
f1 =(exp(3*1)-1)/1-5;
f0_2*f1

%resultados:
```

ans =

-12.5278

Como el resultado es menor. Según el teorema de Bolzano confirmamos que tiene almenos una raíz en ese intervalo.

2. **Método iterativo 1**: $x_{k+1}=g_1(x_k)$ con $x_0=0.4$ y $g_1(x)=(\exp(3x)-1)/5$.

- Código:

```
error=100.0; % Control
iteracion=0;
xn=0.4; % Valor inicial
while (iteracion<20) %Algoritmo
iteracion=iteracion+1;
g1x=(exp(3*xn)-1)/5;
error=abs(g1x-xn); %Estimación error
fprintf('Iter %d Sol %.16f Error %.2e\n', iteracion,g1x,error)
%Imprimir soluciones aprox.
xn=g1x;
end
```

- Resultados:

Iter 1 Sol 0.4640233845473096 Error 6.40e-02
Iter 2 Sol 0.6046340036570275 Error 1.41e-01
Iter 3 Sol 1.0268674087201834 Error 4.22e-01
Iter 4 Sol 4.1543019643505987 Error 3.13e+00
Iter 5 Sol 51712.9810502508917125 Error 5.17e+04
Iter 6 Sol Inf Error Inf
Iter 7 Sol Inf Error NaN
Iter 8 Sol Inf Error NaN
Iter 9 Sol Inf Error NaN
Iter 10 Sol Inf Error NaN
Iter 11 Sol Inf Error NaN
Iter 12 Sol Inf Error NaN
Iter 13 Sol Inf Error NaN
Iter 14 Sol Inf Error NaN
Iter 15 Sol Inf Error NaN
Iter 16 Sol Inf Error NaN
Iter 17 Sol Inf Error NaN
Iter 18 Sol Inf Error NaN
Iter 19 Sol Inf Error NaN
Iter 20 Sol Inf Error NaN

- ¿Converge el método iterativo aplicado en este caso?

No converge pues el resutado tiende a infinito

3. **Método iterativo 2.** $x_{k+1}=g_2(x_k)$ con $x_0=0.4$ y $g_2(x)=\log(1+5x)/3$.

- Código:

```
error=100.0; % Control
iteracion=0;
xn=0.4; % Valor inicial
while (iteracion<20) %Algoritmo
```

```
iteracion=iteracion+1;
g2x=log(1+5*xn)/3;
error=abs(g2x-xn); %Estimación error
e(iteracion)=error;
fprintf('Iter %d Sol %.16f Error %.2e\n', iteracion,g2x,error)
%Imprimir soluciones aprox.
xn=g2x;
end
```

- Resultados en Tabla 1.

Iter 1 Sol 0.3662040962227032 Error 3.38e-02
Iter 2 Sol 0.3468790802299291 Error 1.93e-02
Iter 3 Sol 0.3353034505092931 Error 1.16e-02
Iter 4 Sol 0.3281721389220204 Error 7.13e-03
Iter 5 Sol 0.3237016282368912 Error 4.47e-03
Iter 6 Sol 0.3208682479449680 Error 2.83e-03
Iter 7 Sol 0.3190599200673110 Error 1.81e-03
Iter 8 Sol 0.3179006542924013 Error 1.16e-03
Iter 9 Sol 0.3171553567458821 Error 7.45e-04
Iter 10 Sol 0.3166753198091506 Error 4.80e-04
Iter 11 Sol 0.3163657675248701 Error 3.10e-04
Iter 12 Sol 0.3161659998697136 Error 2.00e-04
Iter 13 Sol 0.3160370174571252 Error 1.29e-04
Iter 14 Sol 0.3159537118727944 Error 8.33e-05
Iter 15 Sol 0.3158998964122841 Error 5.38e-05
Iter 16 Sol 0.3158651269707490 Error 3.48e-05
Iter 17 Sol 0.3158426609759019 Error 2.25e-05
Iter 18 Sol 0.3158281439488178 Error 1.45e-05
Iter 19 Sol 0.3158187630319493 Error 9.38e-06
Iter 20 Sol 0.3158127009340193 Error 6.06e-06

- Convergencia. ¿$g_2(x)$ es contrativa en el intervalo [0.2,1]?

```
x=[0.2:0.01:1];
g2dx=abs(diff(log(1+5.*x)/3));
g2d=g2dx<1;
all(g2d)

%resultados:
```

ans =

logical

1

Por tanto todos los elementos son menores estrictos que 1.

- Velocidad convergencia/orden método: Sucesión e_{k+1} / e_k , nº (no nulo) al que converge. Orden del método:

```
round(e(2:end)./e(1:end-1),1)

%resultados:
```

ans =

Columns 1 through 11

0.6000 0.6000 0.6000 0.6000 0.6000 0.6000 0.6000 0.6000 0.6000 0.6000 0.6000

Columns 12 through 19

0.6000 0.6000 0.6000 0.6000 0.6000 0.6000 0.6000 0.6000

Si converge a un numero no nulo, concretamente a 0.6.
El método es de orden 1 (al menos)o tiene convergencia lineal

4. **Método terativo 3** (Método de Newton-Raphson) $z_{k+1} = z_k - \frac{f(z_k)}{f'(z_k)}$ =... con z_0=0.4.

- Código:

```
error=100.0; % Control
iteracion=0;
x=0.4; % Valor inicial
ez=ones(1,20);
while (iteracion<20)
iteracion=iteracion+1;
g3x=x-((exp(3*x)-1)/x-5)/((exp(3*x)*(3*x-1)+1)/x^2);
error=abs(g3x-x);
ez(iteracion)=error;
fprintf('Iter %d Sol %.16f Error %0.2e\n', iteracion,g3x,error)
x=g3x;
end
```

- Resultados en Tabla 1.

Iter 1 Sol 0.3230498980460820 Error 7.70e-02
Iter 2 Sol 0.3158577970597116 Error 7.19e-03
Iter 3 Sol 0.3158016305923851 Error 5.62e-05
Iter 4 Sol 0.3158016272050506 Error 3.39e-09
Iter 5 Sol 0.3158016272050506 Error 0.00e+00
Iter 6 Sol 0.3158016272050506 Error 0.00e+00
Iter 7 Sol 0.3158016272050506 Error 0.00e+00
Iter 8 Sol 0.3158016272050506 Error 0.00e+00
Iter 9 Sol 0.3158016272050506 Error 0.00e+00
Iter 10 Sol 0.3158016272050506 Error 0.00e+00
Iter 11 Sol 0.3158016272050506 Error 0.00e+00
Iter 12 Sol 0.3158016272050506 Error 0.00e+00
Iter 13 Sol 0.3158016272050506 Error 0.00e+00
Iter 14 Sol 0.3158016272050506 Error 0.00e+00
Iter 15 Sol 0.3158016272050506 Error 0.00e+00
Iter 16 Sol 0.3158016272050506 Error 0.00e+00
Iter 17 Sol 0.3158016272050506 Error 0.00e+00
Iter 18 Sol 0.3158016272050506 Error 0.00e+00
Iter 19 Sol 0.3158016272050506 Error 0.00e+00
Iter 20 Sol 0.3158016272050506 Error 0.00e+00

- ¿Converge el método iterativo? ¿Cuántas iteraciones garantizan que el error es menor que 10^-16?

Si converge, concretamente a 0.3158016272050506. En la iteración 5 se consigue un error menor que 10^-16

- Velocidad convergencia/orden método: Sucesión de términos $ez_{k+1} / (ez_k)^2$ ¿se aproxima a un nº no nulo? ¿Cuál es el orden de convergencia del método?

```
ez(2:end)./(ez(1:end-1)).^2

%resultados:
```

ans =

Columns 1 through 11

1.2146 1.0858 1.0738 0 NaN NaN NaN NaN NaN NaN NaN

Columns 12 through 19

NaN NaN NaN NaN NaN NaN NaN NaN

No, se aproxima a un numero nulo

PREGUNTAR POR EL ORDEN DE CONVERGENCIA A LA PROFESORA IMPORTANTE QUE NO SE NOS OLVIDE

Tabla 1

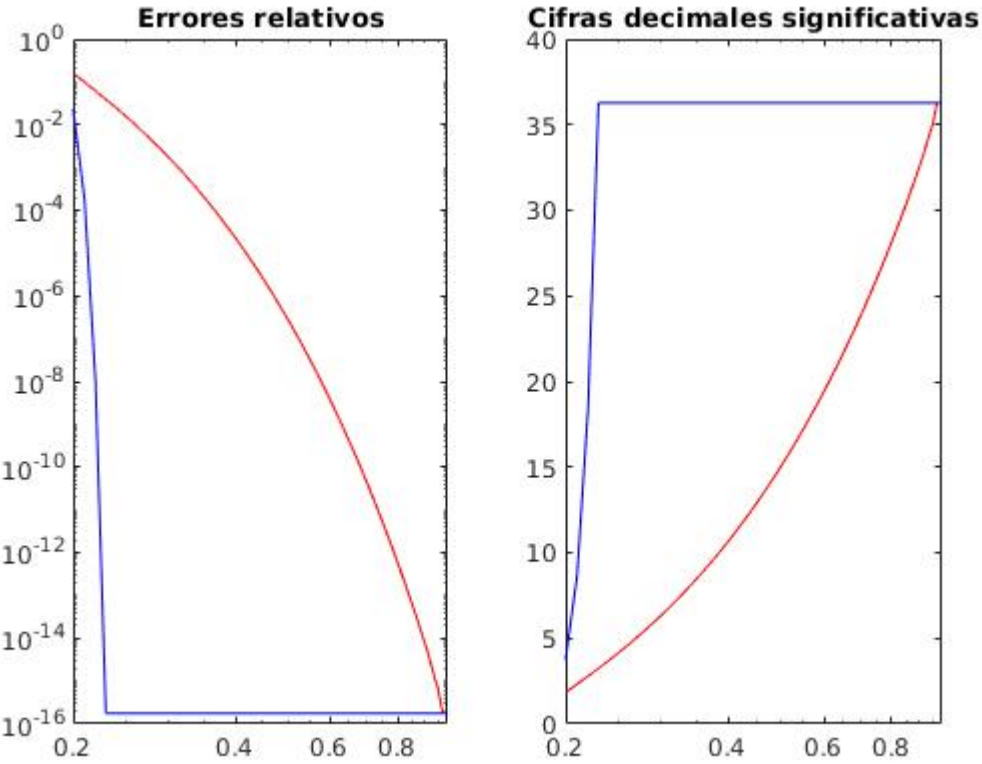
Método 2		Método 3	
x_k	e_k	z_k	ez_k
0.3662040962227032	3.38e-02	0.3230498980460820	7.70e-02
0.346879080229929	1.93e-02	0.3158577970597116	7.19e-03
0.3353034505092931	1.16e-02	0.3158016305923851	5.62e-05
0.3281721389220204	7.13e-03	0.3158016272050506	3.39e-09
0.3237016282368912	4.47e-03	0.3158016272050506	0.00e+00
0.3208682479449680	2.83e-03	0.3158016272050506	0.00e+00
0.3190599200673110	1.81e-03	0.3158016272050506	0.00e+00
0.3179006542924013	1.16e-03	0.3158016272050506	0.00e+00
0.3171553567458821	7.45e-04	0.3158016272050506	0.00e+00
0.3166753198091506	4.80e-04	0.3158016272050506	0.00e+00
0.3163657675248701	3.10e-04	0.3158016272050506	0.00e+00
0.3161659998697136	2.00e-04	0.3158016272050506	0.00e+00
0.3160370174571252	1.29e-04	0.3158016272050506	0.00e+00
0.3159537118727944	8.33e-05	0.3158016272050506	0.00e+00
0.3158998964122841	5.38e-05	0.3158016272050506	0.00e+00
0.3158651269707490	3.48e-05	0.3158016272050506	0.00e+00
0.3158426609759019	2.25e-05	0.3158016272050506	0.00e+00
0.3158281439488178	1.45e-05	0.3158016272050506	0.00e+00
0.3158187630319493	9.38e-06	0.3158016272050506	0.00e+00
0.3158127009340193	6.06e-06	0.3158016272050506	0.00e+00

5. Gráficas errores relativos y cifras decimales significativas. Comentar resultados. ¿Cuántas iteraciones son necesarias en cada caso para calcular la solución con una precisión de 15 cifras decimales?

```
%subaoartado 5.
%errores relativos
%clear

x=[0.2:0.01:1];
fx=0.3158016272050506
iteracion=0;
%metodo 2
erelx=100.0; % Control
xn=0.4; % Valor inicial
erelxk=ones(1,80);
%-----------------
%metodo 3
```

```
erelz=100.0; % Control
xz=0.4;
erelzk=ones(1,80);
%-----------------
while (iteracion<80) %Algoritmo
iteracion=iteracion+1;
%metodo 2
g2x=log(1+5*xn)/3;
erelx=abs(fx-g2x)/fx; %EstimaciÃ³n erel
erelxk(iteracion)=erelx;
xn=g2x;
%------------------------
%metodo 3
g3x=xz-((exp(3*xz)-1)/xz-5)/((exp(3*xz)*(3*xz-1)+1)/xz^2);
erelz=abs(fx-g3x)/fx;
erelzk(iteracion)=erelz;
xz=g3x;
%-------------------
end
x=[0.2:0.01:0.99];
subplot(1,2,1);loglog(x,erelxk,'r',x,erelzk,'b');title('Errores relativos')
ncifxk=-log(erelxk);
ncifzk=-log(erelzk);
subplot(1,2,2);semilogx(x,ncifxk,'r',x,ncifzk,'b');title('Cifras decimales significativas')
```




```
i=1;
finzk=true;
finxk=true;
while(i<80&&(finzk||finxk))
   if(ncifzk(i)>=15&&finzk)
      fprintf('Las 15 cifras significativas de zk se obtienen en la iter: %d\n',i)
      finzk=false;
   end
   if(ncifxk(i)>=15&&finxk)
```

```
        fprintf('Las 15 cifras significativas de xk se obtienen en la iter: %d\n',i)
        finxk=false;
    end
   i=i+1;
end
```

%resultados:
Las 15 cifras significativas de zk se obtienen en la iter: 3
Las 15 cifras significativas de xk se obtienen en la iter: 31

6. - *function[x,e,iter]=fun2(x0,tol,nmax)*

```
function [x,ex,iter] = fun2(x0,tol,nmax)

ex=100; %control
iter=0;
x=x0; %valor inicial
while (iter<nmax && ex >= tol) %Algoritmo
iter=iter+1;
g2x=log(1+5*x)/3;
ex=abs(g2x-x); %EstimaciÃ³n error
x=g2x;
end %while

end %function
```

- *function[z,ez,iterz]=fun3(x0,tol,nmax)*

```
function [z,ez,iterz] = fun3(x0,tol,nmax)
ez=100.0; % Control
iter=0;
z=x0; % Valor inicial
while (iter<nmax && ez>=tol) %algoritmo
iter=iter+1;
g3x=z-((exp(3*z)-1)/z-5)/((exp(3*z)*(3*z-1)+1)/z^2);
ez=abs(g3x-z);
z=g3x;
end

end
```

- Tabla 2

nmax=100 tol=e-16	Método 2			Método 3		
	iter	x	e	iterz	z	ez
x0=0.2	80	0.3158	0	6	0.3159	0
x0=0.4	78	0.3158	5.5511e-17	5	0.3158	0
x0=0.6	80	0.3158	5.5511e-17	6	0.3158	0

Se puede observar que el método 2 necesita de muchas más iteraciones para llegar a un resultado con un error que no sea inferior a la tolerancia dada.
Al ser una tolerancia muy pequeña en todos los casos de los dos métodos se alcanza una x o una z muy parecidas al valor final.
Se observa que cuando el valor inicial es 0.4 se necesita de una o dos iteraciones menos que cuando dicho valor inicial es 0.2 o 0.6 y en ningún casp se llega al máximo número de iteraciones(100).

Parte II. Cálculo PageRank

1. - Construir y mostrar matriz P:

```
A=[0 1 0 0 0 0 0;1 0 0 0 0 0 0;0 1 0 1 0 0 0;0 0 0 0 0 0 0;0 0
1 0 0 0 1;0 0 0 0 1 0 0;1 0 0 1 1 1 0];
P=ones(7,7);
k=1
while(k<=7)
    P(:,k)=A(:,k)/sum(A(:,k))
    k=k+1;
end
P
%resultados:
```

P =

0	0.5000	0	0	0	0	0
0.5000	0	0	0	0	0	0
0	0.5000	0	0.5000	0	0	0
0	0	0	0	0	0	0
0	0	1.0000	0	0	0	1.0000
0	0	0	0	0.5000	0	0
0.5000	0	0	0.5000	0.5000	1.0000	0

2. Método iterativo

v^0 inicial (por ejemplo $v^0 = (1/7) * ones(7,1)$)

$v^{k+1} = Pv^k$ %Iteración

$v^{k+1} = v^{k+1} / norm(v^{k+1},1)$ %Normalización (sum(v^{k+1})=1))

- Código

```
V0=1/7*ones(7,1);
k=1;
e=100; %control
errores=ones(1,100);
while(k<=100 && e>=1e-10)
    V=P*V0;
    V=V/norm(V,1);
    e=norm((V-V0),1);
    errores(k)=e;
    fprintf('Iter %d Error %0.4e\n', k,e)
    k=k+1;
    V0=V;
end
k=k-1;
fprintf('El bucle realiza %d iteraciones\n',k)
V
iter=[1:k];
semilogy(iter,errores(1:k));
```

- ¿Cuántas iteraciones realiza el método?

```
%resultados
El bucle realiza 68 iteraciones
```

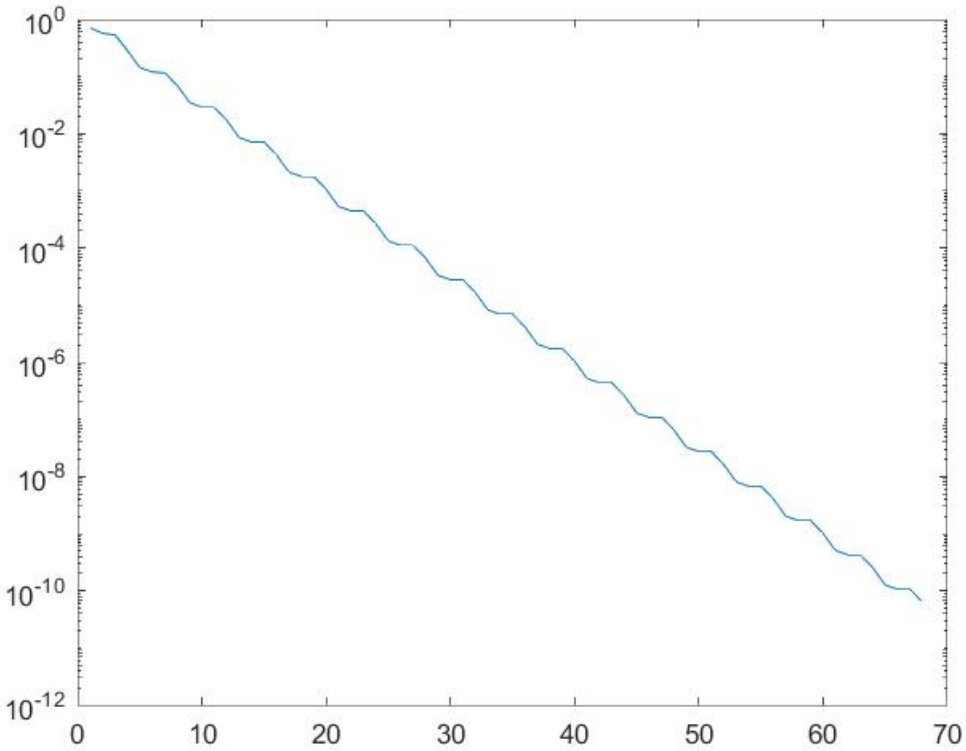
- Mostrar vector v obtenido. Ordenación de las páginas.

```
V =

    0.0000
    0.0000
    0.0000
         0
    0.4000
    0.2000
    0.4000
```

La primera que se muestra es la 5ª, luego la 7ª, luego la 6ª, luego la 1ª, la 2ª, la 3ª y por útlimo la 4ª

- Representar gráficamente, en la escala adecuada, los errores resultantes en las iteraciones realizadas.



3. Vector v proporcionado por comando eig.

```
[V,E]=eig(P)
v=V(:,1)/norm(V(:,1),1)

%resultados:
v =

    0.0000
    0.0000
    0.0000
         0
   -0.4000
   -0.2000
   -0.4000
```