

Sistemas de ecuaciones no lineales

José Luis de la Fuente O'Connor
jldelafuente@etsii.upm.es
joseluis.delafuente@upm.es

Índice

- El problema
- Métodos de numéricos de resolución
 - Método de Newton-Raphson
 - Variantes del método de Newton
 - Métodos cuasi Newton

El problema

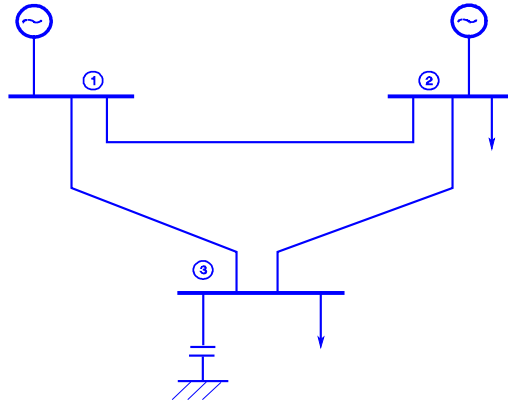
- En términos matemáticos,

dada $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, hallar un \bar{x} tal que $f(\bar{x}) = \mathbf{0}$.

La función vectorial f se supone **continua y derivable** en algún conjunto abierto de \mathbb{R}^n , con derivadas parciales continuas en ese abierto.

- Los algoritmos que estudiamos son **iterativos**. En **en cada etapa se resolverá un sistema de ecuaciones lineales** resultante de una simplificación del problema original en el entorno de cada punto de un proceso.

Estudios de cargas en sistemas de generación y transporte de energía eléctrica



- A partir de un patrón de demanda y generación de potencia en cada uno de los nudos que configuran un sistema, como el esquematizado en la figura:

Los estudios de circulación de cargas determinan las tensiones en los nudos de la red, en módulo y argumento, los flujos de potencia activa y reactiva por todos los elementos del sistema, la intensidad por las líneas, las pérdidas en éstas, etc.

- Las leyes esenciales que gobiernan los flujos de electricidad en redes de este tipo las formuló Gustav Robert Kirchhoff, Alemania (Prusia, Königsberg) 1824-1887.



- Si se supone que los **parámetros físicos de un sistema eléctrico permanecen constantes**, existen cuatro variables asociadas a cada nudo i de ese sistema:
 - la **tensión**, en módulo, V_i , y argumento, θ_i ;
 - la **potencia activa** inyectada, P_i , y
 - la **potencia reactiva** inyectada, Q_i .

- Las potencias inyectadas en el nudo i dependen de su tensión y de las de los a él unidos. Las expresiones que las relacionan, **si no hay transformadores**, son

$$P_i = |V_i|^2 \sum_{\substack{j=1 \\ j \neq i}}^n (G_{p_{ij}} + G_{s_{ij}}) - |V_i| \sum_{\substack{j=1 \\ j \neq i}}^n |V_j| [G_{s_{ij}} \cos(\theta_i - \theta_j) + B_{s_{ij}} \text{sen}(\theta_i - \theta_j)]$$

$$Q_i = -|V_i|^2 \sum_{\substack{j=1 \\ j \neq i}}^n (B_{p_{ij}} + B_{s_{ij}}) - |V_i| \sum_{\substack{j=1 \\ j \neq i}}^n |V_j| [G_{s_{ij}} \text{sen}(\theta_i - \theta_j) - B_{s_{ij}} \cos(\theta_i - \theta_j)]$$

donde: V_i es el módulo de la tensión en el nudo i ;

θ_i el argumento de la tensión en el nudo i ;

$G_{s_{ij}}$ la conductancia serie (constante) de la línea que une el nudo i con el nudo j ;

$G_{p_{ij}}$ la conductancia a tierra (constante) de la línea que une el nudo i con el j ;

$B_{s_{ij}}$ la susceptancia serie (constante) de la línea que une el nudo i con el nudo j ; y

$B_{p_{ij}}$ la susceptancia a tierra (constante) de la línea que une el nudo i con el j .

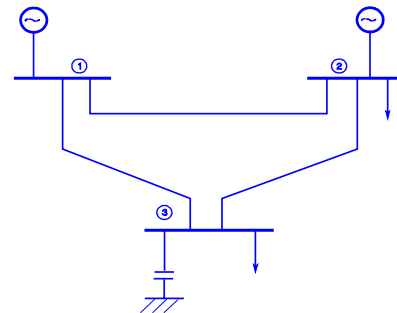
- Es norma que las **tensiones se midan respecto a una referencia**, que se elige en un nudo cualquiera que sea factible, **nudo holgura**, al que se asigna una **tensión de 1 para el módulo y 0 para el argumento**.
- Los tipos de nudos de un sistema y qué variables e incógnitas definirían cada uno de ellos se recogen en la siguiente tabla.

Tipo de nudo	Variables dadas	Incógnitas
Carga o PQ	P, Q	V, θ
Generación o PV	P, V	Q, θ
Holgura	V, θ	P, Q

- Si, por ejemplo, de un nudo se conocen el módulo de la tensión y la potencia activa inyectada, para caracterizarlo totalmente habrá que calcular la potencia reactiva inyectada en él y el argumento de su tensión.

- Al suponer $V_1 = 1$ y $\theta_1 = 0$, para caracterizar un sistema eléctrico de n nudos se necesitarán conocer $2n - 2$ variables.
- Para caracterizar totalmente un sistema general habrá que resolver un sistema de $2n - 2$ ecuaciones no lineales de la forma

$$\begin{aligned} f_1(x_1, x_2, \dots, x_{2n-2}) &= b_1 \\ f_2(x_1, x_2, \dots, x_{2n-2}) &= b_2 \\ &\vdots \\ f_{2n-2}(x_1, x_2, \dots, x_{2n-2}) &= b_{2n-2}. \end{aligned} \tag{1}$$



- Consideremos el pequeño sistema eléctrico de antes.

Si se elige como nudo de holgura el 1, el 2 es PV y el 3 PQ. La función vectorial $f(\mathbf{x})$ que definiría el sistema no lineal de ecuaciones con el que determinar el estado de funcionamiento de ese sistema es la siguiente:

$$f(\mathbf{x}) = \begin{bmatrix} V_2 \\ V_2^2 \sum_{j=1,3} (G_{p2j} + G_{s2j}) - V_2 \sum_{j=1,3} V_j (G_{2j} \cos(\theta_2 - \theta_j) + B_{2j} \sin(\theta_2 - \theta_j)) \\ V_3^2 \sum_{j=1,2} (G_{p3j} + G_{s3j}) - V_3 \sum_{j=1,2} V_j (G_{3j} \cos(\theta_3 - \theta_j) + B_{3j} \sin(\theta_3 - \theta_j)) \\ -V_3^2 \sum_{j=1,2} (B_C + B_{p3j} + B_{s3j}) - V_3 \sum_{j=1,2} V_j (G_{3j} \sin(\theta_3 - \theta_j) - B_{3j} \cos(\theta_3 - \theta_j)) \end{bmatrix} \cdot$$

La susceptancia del condensador conectado al nudo 3 es B_C .

- El \mathbf{b} de (1), lo constituirán los valores de V_2 , P_2 , P_3 y Q_3 , datos del problema.

Índice

- El problema
- Métodos de numéricos de resolución
 - Método de Newton-Raphson
 - Variantes del método de Newton
 - Métodos cuasi Newton

Método de Newton-Raphson

- Estudiaremos funciones vectoriales¹ $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ continuas, con derivadas parciales de primer orden continuas.
- En un punto \mathbf{x}_k del proceso iterativo se aproxima la función mediante **desarrollo en serie de Taylor** truncándolo a partir de los términos de segundo orden,

$$M_k(\mathbf{x}_k) = f(\mathbf{x}_k) + \mathbf{J}(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k),$$

donde $\mathbf{J}(\mathbf{x}_k)$ es la **matriz Jacobiana del sistema** en \mathbf{x}_k :

$$\mathbf{J}(\mathbf{x}_k) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \dots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix}_{\mathbf{x}=\mathbf{x}_k}.$$

¹De momento con $n = m$.

- Si se resuelve el sistema lineal $\mathbf{f}(\mathbf{x}_k) + \mathbf{J}(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) = \mathbf{0}$, su solución dará un avance a un nuevo punto del proceso iterativo.
La **relación de recurrencia** del método es, en general:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{J}(\mathbf{x}_k)^{-1} \mathbf{f}(\mathbf{x}_k)$$

- En condiciones adecuadas, la **convergencia del método es cuadrática**.

- Algoritmo de **Newton-Raphson** para sistemas de ecuaciones no lineales:

I – Definir un $\mathbf{x}_0 \in \mathbb{R}^n$; hacer $k = 1$ y $\mathbf{x}_k \leftarrow \mathbf{x}_0$.

II – Resolver el sistema lineal $\mathbf{J}(\mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k) = -\mathbf{f}(\mathbf{x}_k)$

III – Si $\|\mathbf{f}(\mathbf{x}_{k+1})\|_2 < Tol$, parar: el problema está resuelto

Si no, hacer $k = k + 1$, $\mathbf{x}_k = \mathbf{x}_{k+1}$ e ir al paso II

- Ejemplo** Resolvamos mediante Newton-Raphson, partiendo del punto $[1, 1, 1]^T$, el sistema de ecuaciones no lineales

$$3x_1 - \cos(x_2x_3) - \frac{1}{2} = 0$$

$$x_1^2 - 81 \left(x_2 + \frac{1}{10} \right)^2 + \sin(x_3) + 1,06 = 0$$

$$e^{-x_1x_2} + 20x_3 + \frac{10\pi - 3}{3} = 0.$$

- Un código de **Matlab** para resolverlo:

```
function [x,dnor,iter]=Newt_Rap(funcion,x0,tol,maxiter)
% Newton-Raphson para cualquier sistema no lineal
if nargin<4, maxiter=99999; end
if nargin<3, tol=sqrt(eps); end
dnor=1.0; iter=0; A=[]; Dnor=[];
while dnor>tol && iter<maxiter
    iter=iter+1;
    [f,J]=funcion(x0);
    p=J\f';
    x=x0-p; [f]=funcion(x);
    dnor=norm(f,inf); A=[A;x']; x0=x; Dnor=[Dnor;dnor];
end
s=' %15.10e'; for i=1:size(x0)-1 s=[s ' %15.10e']; end
for i=1:iter
    fprintf([s ' %15.10e\n'], A(i,:),Dnor(i));
end
end
```

```
function [f J]=NRP_1(x)
f(1) = 3*x(1)-cos(x(2)*x(3))-0.5;
f(2) = x(1)^2-81*(x(2)+0.1)^2+sin(x(3))+1.06;
f(3) = exp((-x(1)*x(2)))+20*x(3)+(10*pi-3)/3;
if nargout<2, return, end
J(1,1) = 3.0; J(1,2) = sin(x(2)*x(3))*x(3); J(1,3) = sin(x(2)*x(3))*x(2);
J(2,1) = 2.0*x(1); J(2,2) = -162.0*(x(2)+0.1); J(2,3) = cos(x(3));
J(3,1) = -exp((-x(1)*x(2)))*x(2); J(3,2) = -exp((-x(1)*x(2)))*x(1);
J(3,3) = 20.0;
end
```

- Si se utiliza para resolver el problema:

```
>> [x,dnor,iter]=Newt_Rap(@NRP_1,[1;1;1])
9.1968721308e-001 4.6082245570e-001 -5.0338763550e-001 2.4087256490e+001
5.0100048532e-001 1.8743347767e-001 -5.2086923301e-001 5.8788006806e+000
5.0054293549e-001 6.1153453678e-002 -5.2200096420e-001 1.2916807111e+000
5.0010443627e-001 1.1617105749e-002 -5.2329514612e-001 1.9876169457e-001
5.0000551037e-001 6.0561572295e-004 -5.2358293632e-001 9.8214794394e-003
5.0000001666e-001 1.8263674473e-006 -5.2359872783e-001 2.9529468423e-005
5.0000000000e-001 1.6710515026e-011 -5.2359877560e-001 2.7018031860e-010
x =
    0.5000
    0.0000
   -0.5236
dnor =
    2.7018e-010
iter =
    7
```

Modificaciones del Método de Newton

Newton-Raphson por diferencias finitas

- El cálculo de la **la matriz Jacobiana** del sistema se lleva a cabo mediante **su aproximación por diferencias finitas**.
- Cada columna \mathbf{a}_j de la matriz Jacobiana se aproxima mediante la expresión

$$\mathbf{a}_j = \frac{\mathbf{f}(\mathbf{x} + h_j \mathbf{e}_j) - \mathbf{f}(\mathbf{x})}{h_j}.$$

En **condiciones favorables** el parámetro h hacerse $\sqrt{\epsilon}$ para todos los x_j .

- Si el valor de los coeficientes del vector \mathbf{x} difieren mucho unos de otros, hay que verificar los resultados de esta aproximación a menudo.

- **Ejemplo** Partiendo de $\mathbf{x} = [1, 1, 1]^T$, Newton-Raphson por diferencias finitas para resolver el ejemplo anterior:

```
function Newtrp_df_1(fx,x)
% Newton-Raphson por diferencias finitas para sistemas
global h
tol=sqrt(eps); dnor=1.0; h=tol;
while dnor >tol
    [f J]=fx(x);
    p=J\f';
    x=x-p;
    dnor=norm(fx(x));
    fprintf('%15.10e %15.10e %15.10e %15.10e\n', x,dnor);
end
end
```

```
function [f,J]=NRP_1_dif(x)
global h
f(1) = 3*x(1)-cos(x(2)*x(3))-0.5;
f(2) = x(1)^2-81*(x(2)+0.1)^2+sin(x(3))+1.06;
f(3) = exp((-x(1)*x(2)))+20*x(3)+(10*pi-3)/3;
if nargin<2, return, end
for i=1:3
    x(i)=x(i)+h;
    f1=NRP_1_dif(x);
    J(1:3,i)=(f1-f)/h;
    x(i)=x(i)-h;
end
end
```


- El proceso de convergencia que resulta de la ejecución de este código, con

```
>> Newtrp_df_1(@NRP_1_dif,x)
```

es el que describe la tabla.

k	x_1	x_2	x_3	$\ f(x_k)\ _2$
1	9,1968721314e-1	4,6082245826e-01	-5,0338763389e-1	2,4087256721e+01
2	5,0100048524e-1	1,8743348339e-01	-5,2086923236e-1	5,8788009464e+00
3	5,0054293556e-1	6,1153459243e-02	-5,2200096436e-1	1,2916808565e+00
4	5,0010443628e-1	1,1617109794e-02	-5,2329514576e-1	1,9876176740e-01
5	5,0000551039e-1	6,0561685407e-04	-5,2358293631e-1	9,8214978438e-03
6	5,0000001666e-1	1,8264191607e-06	-5,2359872782e-1	2,9530304459e-05
7	5,0000000000e-1	-1,6847869395e-11	-5,2359877560e-1	2,7240041680e-10

Según se ve por la ejecución:

```
>> x=[1;1;1];  
>> Newtrp_df_1(@NRP_1_dif,x)  
9.1968721314e-01 4.6082245826e-01 -5.0338763389e-01 2.4087256721e+01  
5.0100048524e-01 1.8743348339e-01 -5.2086923236e-01 5.8788009464e+00  
5.0054293556e-01 6.1153459243e-02 -5.2200096436e-01 1.2916808565e+00  
5.0010443628e-01 1.1617109794e-02 -5.2329514576e-01 1.9876176740e-01  
5.0000551039e-01 6.0561685407e-04 -5.2358293631e-01 9.8214978438e-03  
5.0000001666e-01 1.8264191607e-06 -5.2359872782e-01 2.9530304459e-05  
5.0000000000e-01 1.6847869395e-11 -5.2359877560e-01 2.7240041680e-10
```

- Podemos usar diferencias finitas centradas:

```
function [f J]=NRP_1_dif_1(x)
global h
f(1) = 3*x(1)-cos(x(2)*x(3))-0.5;
f(2) = x(1)^2-81*(x(2)+0.1)^2+sin(x(3))+1.06;
f(3) = exp((-x(1)*x(2)))+20*x(3)+(10*pi-3)/3;
if nargin<2, return, end
for i=1:3
    x(i)=x(i)+h;    f1=NRP_1_dif_1(x);
    x(i)=x(i)-2*h; f2=NRP_1_dif_1(x);
    J(1:3,i)=(f1-f2)/(2*h);
    x(i)=x(i)+h;
end
end
```

Resultando:

```
>> Newtrp_df_1_c(@NRP_1_dif_1,[1;1;1])
9.1968721308e-001 4.6082245570e-001 -5.0338763551e-001 2.4087256490e+001
5.0100048532e-001 1.8743347768e-001 -5.2086923301e-001 5.8788006807e+000
5.0054293549e-001 6.1153453680e-002 -5.2200096420e-001 1.2916807112e+000
5.0010443627e-001 1.1617105749e-002 -5.2329514612e-001 1.9876169458e-001
5.0000551037e-001 6.0561572299e-004 -5.2358293632e-001 9.8214794401e-003
5.0000001666e-001 1.8263674477e-006 -5.2359872783e-001 2.9529468429e-005
5.0000000000e-001 1.6710502161e-011 -5.2359877560e-001 2.7018009656e-010
```

Newton modificado

- Resulta de considerar la misma matriz Jacobiana, $\mathbf{J}(\mathbf{x}_0)$, durante todo el proceso, o en un número fijo de iteraciones.

Jacobi

- La matriz Jacobiana se aproxima mediante los coeficientes de su diagonal principal. La relación de recurrencia es:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{D}_k^{-1} \mathbf{f}(\mathbf{x}_k),$$

donde $d_{ii}^k = J_{ii}^k$.

- Si la matriz \mathbf{J} es diagonal dominante, esta estrategia puede ser suficiente.

Gauss-Seidel

- La matriz del sistema es la parte triangular inferior de la Jacobiana. La relación de recurrencia es:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{L}_k^{-1} \mathbf{f}(\mathbf{x}_k)$$

donde $L_{ij}^k = J_{ij}^k, i \geq j$.

Relajación SOR

- El esquema iterativo en forma matricial que se utiliza es

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\rho \mathbf{D}_k + \mathbf{L}_k)^{-1} \mathbf{f}(\mathbf{x}_k)$$

El parámetro de relajación es $\omega = 1/(\rho + 1)$.



Charles George Broyden, Reino Unido, 1933-2011.

Métodos cuasi Newton

- Extienden la idea del método de la secante a n dimensiones. La primera idea se publicó en 1965 por Broyden.
- Su centran en escoger $\mathbf{J}(\mathbf{x}_k)$ de tal forma que se minimice el valor de la función que se obtendría en un mismo punto mediante sus dos aproximaciones \mathbf{A}_k y \mathbf{A}_{k-1} y que además se cumpla que $\mathbf{A}_k(\mathbf{x}_k - \mathbf{x}_{k-1}) = \mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}_{k-1})$.
- Si $\mathbf{s}_{k-1} = \mathbf{x}_k - \mathbf{x}_{k-1}$ y $\mathbf{y}_{k-1} = \mathbf{f}(\mathbf{x}_k) - \mathbf{f}(\mathbf{x}_{k-1})$, lo cumple la aproximación de $\mathbf{J}(\mathbf{x}_k)$ que sigue esta recurrencia:

$$\mathbf{A}_k = \mathbf{A}_{k-1} + \frac{(\mathbf{y}_{k-1} - \mathbf{A}_{k-1}\mathbf{s}_{k-1})\mathbf{s}_{k-1}^T}{\mathbf{s}_{k-1}^T\mathbf{s}_{k-1}}$$

- El algoritmo con esta fórmula es el que sigue.

I – Definir $x_0 \in \mathbb{R}^n$ y $A_0 \in \mathbb{R}^{n \times n}$; hacer $k = 1$ y $x_k \leftarrow x_0$.

II – Determinar la solución de $A_k s_k = -f(x_k)$.

III – Si $\|f(x_k)\|_2 < Tol$, PARAR: el problema está resuelto.
Si $> Tol$, hacer

$$x_{k+1} \leftarrow x_k + s_k$$
$$y_k \leftarrow f(x_{k+1}) - f(x_k)$$
$$A_{k+1} \leftarrow A_k + \frac{(y_k - A_k s_k) s_k^T}{s_k^T s_k}$$
$$k \leftarrow k + 1$$

y volver al paso II.

- La A_0 se puede obtener por una aproximación parcial o total: por diferencias finitas, por ejemplo.
- La **convergencia** del método a que da lugar es **superlineal**.

- **Ejemplo** Desde $[1, 1, 1]^T$, resolvamos este sistema con Newton y fórmula de Broyden.

$$\begin{aligned}
 3x_1 - \cos(x_2x_3) - \frac{1}{2} &= 0 \\
 x_1^2 - 81\left(x_2 + \frac{1}{10}\right)^2 + \sin(x_3) + 1,06 &= 0 \\
 e^{-x_1x_2} + 20x_3 + \frac{10\pi-3}{3} &= 0.
 \end{aligned}$$

- En **Matlab**:

```

function Broyden_3(fx,x)
% Método cuasi Newton con la fórmula de Broyden
tol=sqrt(eps); n=length(x); dnor=1.0; J=zeros(n,n); f=fx(x);
J(1,1)=3; J(2,2)=-178.2; J(3,3)=20; % Trampa

while dnor >tol
    p=J\f';
    x1=x-p;
    f=fx(x1);
    dnor=norm(f);
    fprintf(' %15.10e %15.10e %15.10e %15.10e\n',x,dnor);
    J=broy(J,f,p);
    x=x1;
end
end

function J=broy(J,f,p)
    prod=p'*p;
    J=J-(1/prod)*f'*p'; % igual que J=J-(1/prod)*(y'+J*p)*p' ver Quarteroni p.289
end

```

- Como matriz A_0 se utiliza la diagonal de la Jacobiana en el punto de partida.

- El proceso de convergencia que desencadena la instrucción

```
>> Broyden_3(@NRP_1, [1 1 1]')
```

es el de la tabla.

k	x_1	x_2	x_3	$\ f(x_k)\ _2$
1	3,467674352893799E-1	4,662821024806326E-01	-4,919927476568708E-1	25,275175252053120
2	4,921232306763561E-1	3,236527849976335E-01	-5,162769886149683E-1	13,729480399369230
3	4,993486752210201E-1	2,131483731754155E-01	-5,166714279059975E-1	7,127754268893964
4	5,011649166344201E-1	9,690341763001632E-02	-5,210585843043458E-1	2,327087146316625
5	5,003080441638231E-1	4,279330810076126E-02	-5,224749127576461E-1	8,403043972608411E-01
6	5,001066711564305E-1	1,172102964534057E-02	-5,232913081815899E-1	2,006362866054586E-01
7	5,000183047478762E-1	2,032314047074978E-03	-5,235457794542374E-1	3,319399780484372E-02
8	5,000009846717887E-1	1,115674463108231E-04	-5,235958520108367E-1	1,804970096278442E-03
9	5,000000108711760E-1	1,033227841870006E-06	-5,235987485509558E-1	1,678549255880026E-05
10	5,000000000415024E-1	6,118437770431628E-10	-5,235987755897009E-1	9,122344458875651E-08
11	4,99999999986228E-1	-5,059011531347285E-09	-5,235987757245316E-1	4,849895176081806E-10

```
>> Broyden_3(@NRP_1, [1 1 1]')
1.0000000000e+00 1.0000000000e+00 1.0000000000e+00 2.5275175224e+01
3.4676743529e-01 4.6628210429e-01 -4.9199274766e-01 1.3729480376e+01
4.9212323059e-01 3.2365278698e-01 -5.1627698859e-01 7.1277542490e+00
4.9934867516e-01 2.1314837540e-01 -5.1667142786e-01 2.3270871287e+00
5.0116491660e-01 9.6903420362e-02 -5.2105858424e-01 8.4030438329e-01
5.0030804414e-01 4.2793311463e-02 -5.2247491268e-01 2.0063627763e-01
5.0010667115e-01 1.1721033805e-02 -5.2329130808e-01 3.3193994651e-02
5.0001830475e-01 2.0323188142e-03 -5.2354577934e-01 1.8049697418e-03
5.0000098467e-01 1.1157244942e-04 -5.2359585189e-01 1.6785486153e-05
5.0000001087e-01 1.0382563731e-06 -5.2359874843e-01 9.1223423183e-08
5.0000000004e-01 5.6408065297e-09 -5.2359877546e-01 4.8499505118e-10
```

Se aprecia aquí

Implementación práctica del método de Broyden

Lema Fórmula de Sherman-Morrison-Woodbury 1949-1950

(a) Si A es una matriz regular $n \times n$ y u y v dos vectores cualesquiera de \mathbb{R}^n , $A + uv^T$ es regular si y sólo si $w = 1 + v^T A^{-1} u \neq 0$.

(b) En este caso, además, $(A + uv^T)^{-1} = A^{-1} - \left(\frac{1}{w}\right) A^{-1} uv^T A^{-1}$.

- La **idea** es partir de una A_0^{-1} y adaptar A^{-1} con la **nueva fórmula de Broyden**:

$$A_{k+1}^{-1} = A_k^{-1} + \frac{(s_k - A_k^{-1} y_k) s_k^T A_k^{-1}}{s_k^T A_k^{-1} y_k}$$