FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA

..:: Mecánica e Industrial

CURSOS ABIERTOS

DIPLOMADO DE MATEMÁTICAS APLICADAS A LA INGENIERÍA

MÓDULO II CA-484
"MÉTODOS NUMÉRICOS"
TEMA: APUNTES GENERALES

EXPOSITOR: M. EN C. ABEL VALDÉS RAMÍREZ
12 DE NOVIEMBRE AL 3 DE DICIEMBRE DE 2005
PALACIO DE MINERIA

1. Solución de ecuaciones no lineales

Se trata de encontrar un valor $\underline{x}$ que cumple la igualdad $f(\underline{x}) = 0$. Este valor se conoce como un *cero* o una *raíz* de f(x).

1.1 Método iterativo de punto fijo

Dada una función g(x) , un valor x tal que x = g(x) es llamado un *punto fijo* de g.
Una ecuación no lineal f(x) = 0 puede ser transformada en una función x = g(x) de tal manera que un punto fijo de g(x) sea también una raíz de f(x). Por ejemplo, considere a $f(x) = x^2 - x - 2 = 0$; cualquiera de las posibilidades

$$g(x) = x^2 - 2$$

$$g(x) = \sqrt{x+2}$$

$$g(x) = 1 + \frac{2}{x}$$

$$g(x) = \frac{x^2 + 2}{2x - 1}$$

es una función cuyo punto fijo es una solución de f(x) = 0.
El primer paso de un procedimiento que puede encontrar un punto fijo de g(x) consiste en proponer una estimación inicial $x^{(0)}$, evaluar g(x) en $x^{(0)}$ para obtener una nueva aproximación $x^{(1)}$, es decir, $x^{(1)} = g(x^{(0)})$. La siguiente aproximación se calcula mediante $x^{(2)} = g(x^{(1)})$ y de esta forma se van generando aproximaciones $x^{(3)}$, $x^{(4)}$, $x^{(5)}$, $x^{(6)}$, ... El proceso se da por terminado cuando se cumple alguno de los siguientes criterios de convergencia para las dos últimas iteraciones:

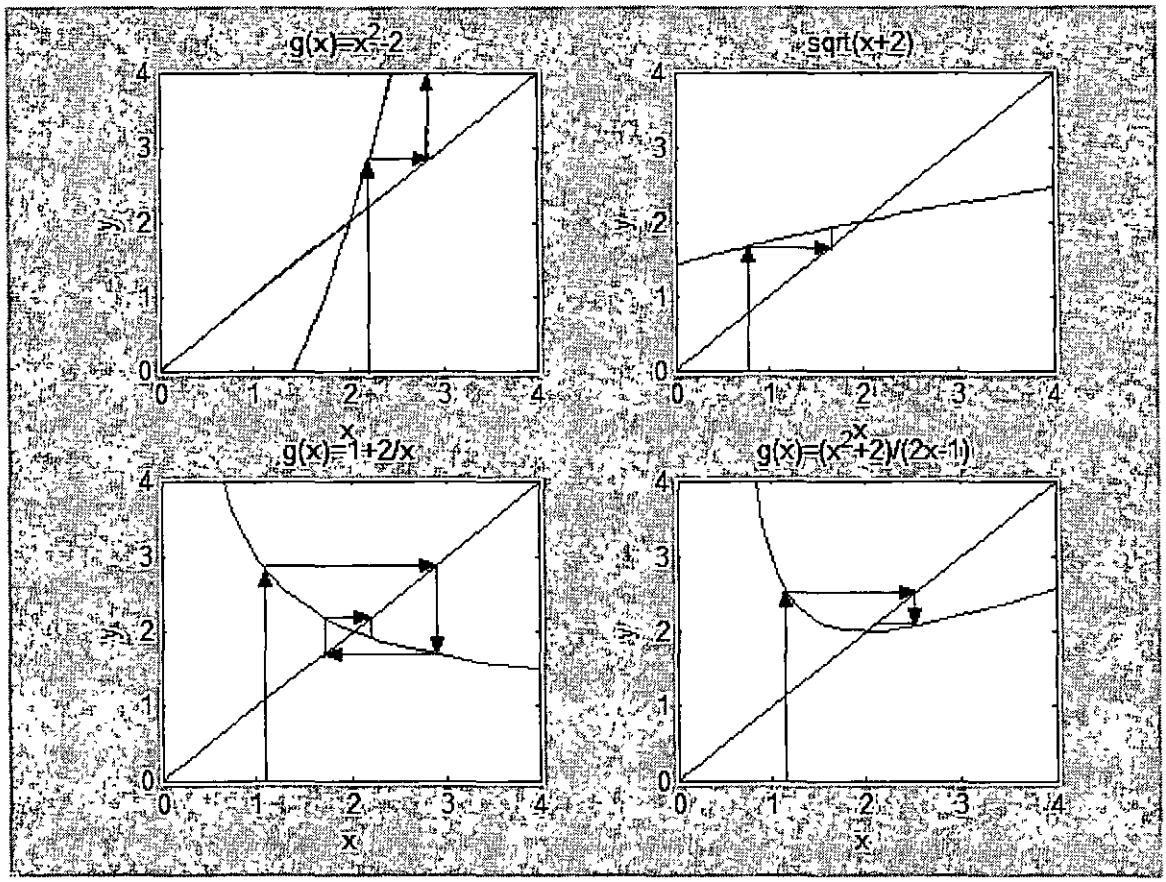
1.- $\left|x^{(k+1)} - x^{(k)}\right| < \epsilon$

2.- $\left|x^{(k+1)} - x^{(k)}\right| / \left|x^{(k+1)}\right| < \epsilon$

3.- $f(\left|x^{(k+1)}\right|) < \epsilon$

y entonces se considera que $x^{(k+1)}$ es un punto fijo aproximado de g(x) y se dice que el proceso alcanzó la convergencia. La tolerancia ϵ es fijada antes de iniciar el cálculo de las iteraciones. Este procedimiento es conocido como el *método de punto fijo* o de *iteración funcional*.
Geométricamente un punto fijo de g(x) es la intersección de la recta y = x con la curva

y = g(x). La siguiente figura muestra las gráficas de las cuatro funciones g(x) indicadas arriba y el comportamiento del método de punto fijo para cada una de ellas.



Ejemplo

Encuentre una raíz aproximada de $f(x) = x^2 - x - 2 = 0$ con el método de punto fijo. Use las funciones g(x) descritas anteriormente. En todos los casos, inicie con $x^{(0)} = 1.5$ y considere un criterio de convergencia $|x^{(k+1)} - x^{(k)}| < 1X10^{-4}$ con un máximo de 20 iteraciones.

$g(x) = x^2 - 2$

Iteración k	$x^{(k)}$	$\|x^{(k)} - x^{(k-1)}\|$
0	1.50000	
1	0.25000	1.25000

2	-1.93750	2.18750
3	1.75391	3.69141
4	1.07619	0.67772
5	-0.84182	1.91801
6	-1.29134	0.44952
7	-0.33245	0.95889
8	-1.88948	1.55703
9	1.57013	3.45960
10	.46530	1.10483
11	-1.78350	2.24880
12	1.18087	2.96437
13	-0.60555	1.78642
14	-1.63331	1.02776
15	.66770	2.30101
16	-1.55417	2.22188
17	0.41545	1.96962
18	-1.82740	2.24585
19	1.33939	3.16679
20	-0.20603	1.54542

$g(x) = \sqrt{(x + 2)}$

Iteración k	$x^{(k)}$	$\lvert x^{(k)} - x^{(k-1)} \rvert$
0	1.50000	
1	1.87083	0.37083
2	1.96744	0.09661

3	1.99184	0.02440
4	1.99796	0.00612
5	1.99949	0.00153
6	1.99987	0.00038
7	1.99997	0.00010

$g(x) = 1 + 2/x$

Iteración k	$x^{(k)}$	$\lvert x^{(k)} - x^{(k-1)} \rvert$
0	1.50000	
1	2.33333	0.83333
2	1.85714	0.47619
3	2.07692	0.21978
4	1.96296	0.11396
5	2.01887	0.05590
6	1.99065	0.02821
7	2.00469	0.01404
8	1.99766	0.00704
9	2.00117	0.00351
10	1.99941	0.00176
11	2.00029	0.00088
12	1.99985	0.00044
13	2.00007	0.00022
14	1.99996	0.00011
15	2.00002	0.00005

$g(x) = (x^2 + 2)/(2x - 1)$

Iteración k	$x^{(k)}$	$\lvert x^{(k)} - x^{(k-1)} \rvert$
0	1.50000	
1	2.12500	0.62500
2	2.00481	0.12019
3	2.00001	0.00480
4	2.00000	$7.68X10^{-6}$

La condición suficiente pero no necesaria que garantiza la convergencia del método de punto fijo está dada por el teorema:

Si $g(x)$ y $g'(x)$ son continuas en un intervalo alrededor de un punto fijo $\underline{x}$ de la ecuación $x = g(x)$, y si $|g'(x)| < 1$ para toda x en el intervalo, entonces $x^{(k+1)} = g(x^{(k)})$, $k = 0, 1, 2, \ldots,$ converge en el punto fijo $x = \underline{x}$ en el supuesto de que $x^{(1)}$ se escoja en el intervalo.

A continuación, un pseudocódigo de un procedimiento que encuentra un punto fijo de $x = g(x)$[1]

proc punto_fijo(real func g, real &xinicial, real eps, entero maxiter, real &x, lógico &convergencia)
!
! Este procedimiento encuentra una aproximación a la solución del problema x = g(x)
! con el método iterativo de punto fijo.
! Descripción de parámetros:
! g Función externa que evalúa a la función g(x) .
! xinicial Estimación inicial.
! eps Máximo error permisible. Cuando el valor absoluto de la diferencia de las
! dos últimas iteraciones es menor que eps, se considera haber llegado a la
! solución.
! maxiter Número máximo de iteraciones.
! x Última iteración calculada. Si en convergencia se devuelve el valor
! VERDADERO, entonces x también es el punto fijo buscado.
! convergencia Bandera cuyo valor indica si hubo convergencia:
! convergencia=VERDADERO se alcanzó la convergencia

[1]El lenguaje de pseudocódigo empleado está basado el propuesto por Guillermo Levine G., en su libro "Computación y Programación Moderna", Pearson Educación, México2001.

```
!                    convergencia=FALSO          no se alcanzó la convergencia.
!
entero iter
convergencia=VERDADERO
ejecuta  iter = 1, maxiter
{
        x = g(xinicial)   ! cálculo de una nueva iteración
        si ABS(x - xinicial) < eps entonces  regresa    ! se checa la convergencia
        xinicial = x      ! preparación de la siguiente iteración
}
convergencia = FALSO
regresa
fin
```

1.2 Aceleración de la convergencia

El método de punto fijo cuando converge, generalmente lo hace en forma muy lenta. Para acelerar su convergencia se pueden aplicar varias técnicas, una de las más usadas es la *aceleración de Aitken*: las dos primeras iteraciones $x^{(1)}$ y $x^{(2)}$ se calculan con el método de punto fijo mientras que la tercera se obtiene aplicando la fórmula:

$$x^{(3)} = x^{(0)} - \frac{\left(x^{(0)} - x^{(1)}\right)^2}{x^{(2)} - 2x^{(1)} + x^{(0)}}$$

Las iteraciones $x^{(4)}$ y $x^{(5)}$ se calculan nuevamente con punto fijo y $x^{(6)}$ se determina con la fórmula de Aitken pero aplicada a $x^{(3)}$, $x^{(4)}$ y $x^{(5)}$. Es de esperarse que al aplicar la aceleración de Aitken cada tres iteraciones, el problema converja en un menor número de iteraciones.
Este técnica de Aitken puede aplicarse a cualquier proceso iterativo donde los errores decrezcan proporcionalmente, no sólo a la iteración de punto fijo.

Ejemplo
Obtenga un punto fijo aproximado de $g(x) = 1 + 2/x$. Inicie con $x^{(0)} = 1.5$ y termine cuando el valor absoluto de las últimas dos iteraciones sea menor que $1X10^{-4}$. Considere un número máximo de iteraciones igual a 20.

Iteración k	$x^{(k)}$	$\lvert x^{(k)} - x^{(k-1)} \rvert$
0	1.50000	

1	2.33333	0.83333
2	1.85714	0.47619
3	2.03030	0.17316
4	1.98507	0.04523
5	2.00752	0.02244
6	2.00000	0.00744
7	1.99996	0.00011
8	2.00002	0.00006

Un pseudocódigo de un procedimiento que encuentra un punto fijo de g(x) con punto fijo y aceleración de Aitken es:

```
proc  Aitken(real func g, real &xinicial, real eps, entero maxiter, real &x, lógico &convergencia)
!
!  Este procedimiento encuentra una aproximación a la solución del  problema x = g(x)
!  empleando el método iterativo de punto fijo con aceleración de Aitken.
!  Descripción de parámetros:
!  g                Función externa que evalúa a la función g(x) .
!  xinicial         Estimación inicial.
!  eps              Máximo error permisible. Cuando el valor absoluto de la diferencia de las
!                   dos últimas iteraciones es menor que eps, se considera haber llegado a la
!                   solución.
!  maxiter          Número máximo de iteraciones.
!  x                Última iteración calculada. Si en convergencia se devuelve el valor
!                   VERDADERO, entonces x también es el punto fijo buscado.
!  convergencia     Bandera cuyo valor indica si hubo convergencia:
!                   convergencia=VERDADERO      se alcanzó la convergencia
!                   convergencia=FALSO          no se alcanzó la convergencia.
!
entero iter
real x0, x1, x2
convergencia=VERDADERO
x1 = g(xinicial)        ! cálculo de las dos primeras iteraciones x1 y x2 con punto fijo
si ABS(x1 - xinicial) < eps entonces  regresa
x2 = g(x1)
si ABS(x2 - x1) < eps entonces  regresa
ejecuta  iter = 3, maxiter
```

```
{
        !
        ! se aplica la aceleración de Aitken cada tres iteraciones
        !
        si MOD(iter, 3) = 0 entonces
                x = xinicial- (xinicial - x1)↑2 / (x2 - 2*x1+ xinicial)          ! Aitken
        otro
                x= g(x2)                                                        ! punto fijo
        si ABS(x - x2) < eps entonces  regresa    ! se checa la convergencia
        xinicial = x1      ! preparación de la siguiente iteración
        x1 = x2
        x2 = x
}
convergencia = FALSO
regresa
fin
```

1.3 Método de Newton-Raphson

Si f(x) es un función continua y diferenciable n veces en un intervalo y si $f^{(n+1)}$ existe en ese mismo intervalo, entonces el valor de la función f(x) alrededor de un punto x_0 está dado por

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + f''(x_0)(x - x_0)^2/2! + f'''(x_0)(x - x_0)^3/3! + \ldots\ldots + f^{(n)}(x_0)(x - x_0)^n/n! +$$

$$R_n$$

donde

$$R_n = f^{(n+1)}(\zeta(x))(x - x_0)^{n+1}/(n+1)!$$

Aplicando este resultado, conocido como teorema de Taylor, a un punto base $x_0 = x^{(k)}$ y a $x = x^{(k+1)}$

$$f(x^{(k+1)}) = f(x^{(k)}) + f'(x^{(k)})(x^{(k+1)} - x^{(k)}) + f''(x^{(k)})(x^{(k+1)} - x^{(k)})^2/2! + f'''(x^{(k)})(x^{(k+1)} - x^{(k)})^3/3! + \ldots\ldots +$$

$$f^{(n)}(x^{(k)})(x^{(k+1)} - x^{(k)})^n/n! + R_n$$

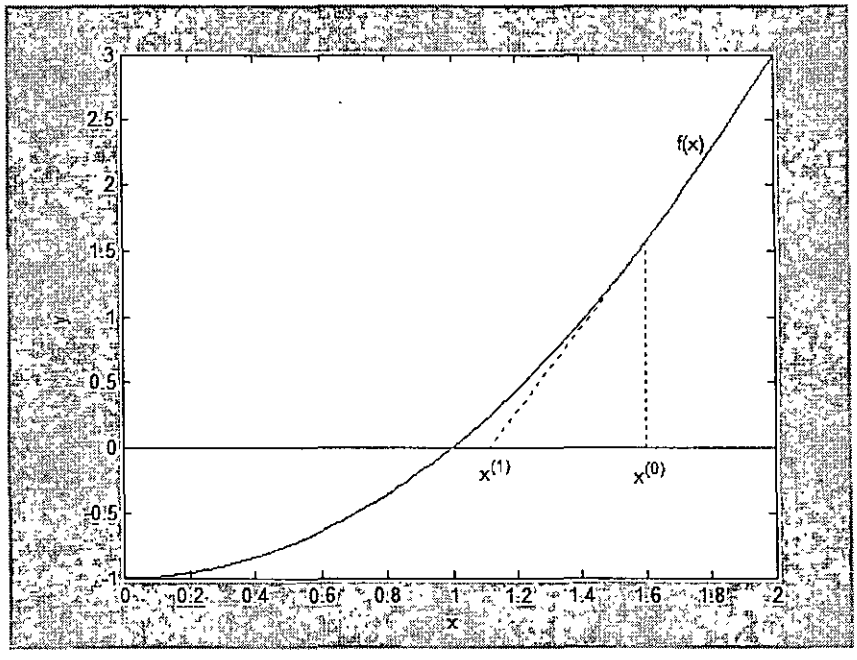
A continuación se harán dos suposiciones:
1.- $x^{(k+1)} \approx \underline{x}$, por lo que es razonable pensar que $f(x^{(k+1)}) \approx 0$
2.- $x^{(k+1)}$ y $x^{(k)}$ son valores cercanos, por lo tanto $(x^{(k+1)} - x^{(k)})^j \approx 0$ para $j = 2, 3, 4, \ldots$
las cuales simplifican a la última ecuación:

$$0 \approx f(x^{(k)}) + f'(x^{(k)})(x^{(k+1)} - x^{(k)})$$

y entonces se despeja a $x^{(k+1)}$:

$$x^{(k+1)} = x^{(k)} - f(x^{(k)})/f'(x^{(k)})$$

Esta última ecuación es el método de Newton-Raphson.
Desde el punto de vista geométrico; $x^{(k+1)}$ es la intersección del eje horizontal x con la recta tangente a f(x) en $(x^{(k)}, f(x^{(k)}))$.



Ejemplo
Encuentre una raíz aproximada de $f(x) = 3x + \sin(x) - e^{-x} = 0$. Tome como estimación inicial $x^{(0)} = 0$ y un criterio de convergencia $|x^{(k+1)} - x^{(k)}| < 1X10^{-4}$.

$$f(x) = 3x + \text{sen}(x) - e^{x}$$

$$f'(x) = 3 + \cos(x) - e^{x}$$

Iteración k	$x^{(k)}$	$\lvert x^{(k)} - x^{(k-1)} \rvert$
0	0.00000	
1	0.33333	0.33333
2	0.36017	0.02684
3	0.36042	0.00025

4	.36042	$2.24843X10^{-8}$

Newton Raphson casi siempre converge rápidamente, sin embargo para que funcione en forma adecuada se requiere de una estimación inicial cercana a la raíz.
Una propuesta del pseudocódigo de un procedimiento que encuentra una raíz aproximada de una función f(x) = 0 con Newton Raphson es:

```
proc newton(real func f, real func f_prima, real &xinicial, real eps, entero maxiter, real &x,
lógico &convergencia)
!
!  Este procedimiento encuentra una aproximación a una raíz de f(x)=0 con el método iterativo
!  de Newton-Raphson.
!  Descripción de parámetros:
!  f                Función externa que evalúa a la función f(x) .
!  f_prima          Función externa que evalúa a la derivada de f(x).
!  xinicial         Estimación inicial.
!  eps              Máximo error permisible. Cuando el valor absoluto de la diferencia de las
!                   dos últimas iteraciones es menor que eps, se considera haber llegado a la
!                   solución.
!  maxiter          Número máximo de iteraciones.
!  x                Última iteración calculada. Si en convergencia se devuelve el valor
!                   VERDADERO, entonces x también es una raíz de f(x).
!  convergencia     Bandera cuyo valor indica si hubo convergencia:
!                   convergencia=VERDADERO      se alcanzó la convergencia
!                   convergencia=FALSO          no se alcanzó la convergencia.
!
entero iter
convergencia=VERDADERO
ejecuta  iter = 1, maxiter
{
        x = xinicial - f(xinicial)/f_prima(xinicial)   ! cálculo de una nueva iteración
        si ABS(x - xinicial) < eps entonces  regresa    ! se checa la convergencia
        xinicial = x      ! preparación de la siguiente iteración
}
convergencia = FALSO
regresa
fin
```

1.4 Método de la secante

Newton Raphson es un método muy empleado por su simplicidad y rapidez, pero tiene el

inconveniente de requerir la derivada de f(x). Esta situación puede ser muy problemática si la función es difícil de derivar. Una forma de evitar la derivación es sustituir en la ecuación de Newton Raphson

$x^{(k+1)} = x^{(k)} - f(x^{(k)})/f'(x^{(k)})$

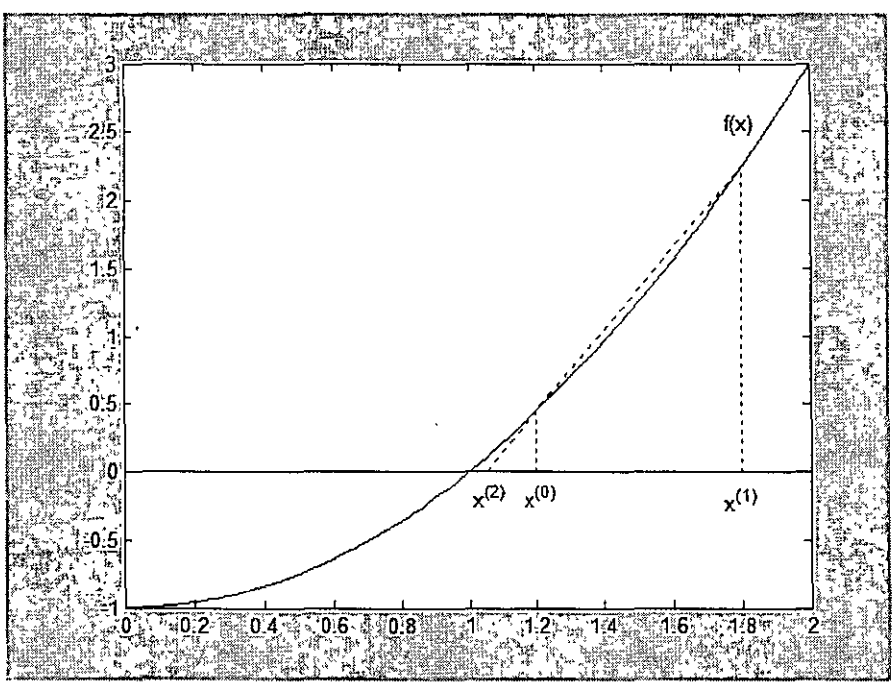
$f'(x^{(k)})$ por la *fórmula de diferencias finitas*

$$f'(x^{(k)}) = \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$$

quedando entonces

$x^{(k+1)} = x^{(k)} - f(x^{(k)}) [f(x^{(k)}) - f(x^{(k-1)})] / [x^{(k)} - x^{(k-1)}]$

que define el método de la secante. El nombre de este algoritmo se debe a que la intersección de la recta secante que pasa por los puntos $(x^{(k-1)}, f(x^{(k-1)}))$, $(x^{(k)}, f(x^{(k)}))$ con el eje horizontal produce la nueva aproximación $x^{(k+1)}$.



Ejemplo
Repita el ejemplo anterior empleando el método de la secante con $x^{(0)} = 1.5$ y $x^{(1)} = 0.8$.

$f(x) = 3x + \text{sen}(x) - e^x$

Iteración k	$x^{(k)}$	$\lvert x^{(k)} - x^{(k-1)} \rvert$
0	1.50000	
1	0.80000	0.70000
2	-4.23482	5.03482
3	0.44708	4.68189
4	0.36549	0.08159
5	0.36025	0.00523
6	0.36042	0.00017
7	0.36042	$3.06666X10^{-7}$

Este método es un miembro importante de una familia conocida como *métodos de dos puntos*, llamada así porque incluye algoritmos que requieren de dos estimaciones iniciales.
El precio que se paga por evitar la derivación de f(x) es una convergencia más lenta. Esto es, secante consumirá más iteraciones que Newton Raphson para un mismo problema.
A continuación, el pseudocódigo de un procedimiento que encuentra un cero aproximado de f(x) = 0 con el método de la secante:

```
proc secante(real func f, real &x0, real &x1, real eps, entero maxiter, real &x, lógico
&convergencia)
!
! Este procedimiento encuentra una aproximación a una raíz de f(x)=0 con el método iterativo
! de la secante.
! Descripción de parámetros:
! f          Función externa que evalúa a la función f(x) .
! x0         Primera estimación inicial.
! x1         Segunda estimación inicial.
! eps        Máximo error permisible. Cuando el valor absoluto de la diferencia de las
!            dos últimas iteraciones es menor que eps, se considera haber llegado a la
!            solución.
! maxiter    Número máximo de iteraciones.
! x          Última iteración calculada. Si en convergencia se devuelve el valor
```

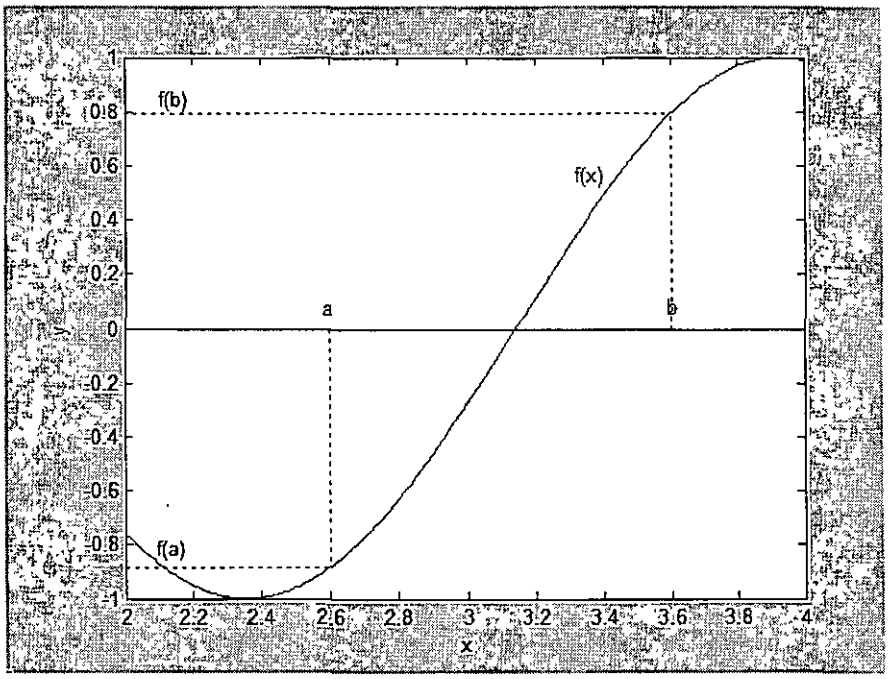
```
!                       VERDADERO, entonces x también es una raíz de f(x).
!  convergencia         Bandera cuyo valor indica si hubo convergencia:
!                       convergencia=VERDADERO      se alcanzó la convergencia
!                       convergencia=FALSO          no se alcanzó la convergencia.
!
entero iter
real f0, f1
convergencia=VERDADERO
f0 = f(x0)
ejecuta  iter = 2, maxiter
{
        f1 = f(x1)
        x = x1 - f1*(x1 - x0)/(f1 - f0)  ! cálculo de una nueva iteración
        si ABS(x - x1) < eps entonces  regresa    ! se checa la convergencia
        x0 = x1      ! preparación de la siguiente iteración
        x1 = x
        f0 = f1
}
convergencia = FALSO
regresa
fin
```

1.5 Método de la bisección

Mediante bisecciones sucesivas busca una raíz contenida en un intervalo [a, b]. Desde luego, para que esta búsqueda funcione, el intervalo original deberá contener un número impar de raíces. La condición que garantiza tal situación es:

$f(a) * f(b) \leq 0$

Si lo anterior no se cumple, entonces o no existe raíz alguna o existen un número par de raíces y en ambos caso no es factible emplear el método de la bisección. La gráfica mostrada a continuación puede ser de utilidad para apreciar mejor esta situación.



Una vez aceptado un intervalo [a, b] tal que f(a) * f(b) ≤ 0, entonces se calcula la primera iteración $x^{(1)}$ con

$$x^{(1)} = (a + b) / 2$$

generándose dos nuevos intervalos: [a, $x^{(1)}$] y [$x^{(1)}$, b]. Entonces se desecha el intervalo que no contiene la raíz y el intervalo aceptado nuevamente es dividido en dos con la fórmula

$$x^{(2)} = (a + b) / 2$$

suponiendo que los valores de a y b son actualizados en cada iteración de tal forma que siempre sean los extremos del intervalo que contiene a la raíz. El proceso se puede generalizar fácilmente para cualquier iteración:

$$x^{(k)} = (a + b) / 2$$

y termina cuando se cumpla alguno de los criterios de convergencia usuales. La última fórmula es método de la bisección.

Ejemplo
Calcule una raíz aproximada de $f(x) = x^3 + 4x^2 - 10 = 0$ con el método de la bisección en el intervalo [1, 2]. Considere un criterio de convergencia $|f(x^{(k)})| < 1X10^{-5}$ y un número máximo de iteraciones igual a 20.

Primeramente, se checa que en [1, 2] exista un número impar de raíces:

$f(1) = f(a) = 1^3 + 4(1)^2 - 10 = -5$

$f(2) = f(b) = 2^3 + 4(2)^2 - 10 = 14$

como $f(a)*f(b) = -5(14) < 0$, es aplicable el método de la bisección en este intervalo.
La primera iteración es:

$x^{(1)} = (1 + 2) / 2 = 1.50000$

y para averiguar en cual de los intervalos [1, 1.5] y [1.5, 2] está la raíz, se evalúa f(x) en $x^{(1)}$

$f(x^{(1)}) = 1.50000^3 + 4(1.50000)^2 - 10 = 2.37500$

Como $f(a)*f(x^{(1)}) = f(1)*f(1.50000) = -5(2.37500) < 0$, la raíz se encuentra en [1, 1.5] y por lo tanto a = 1 y b = 1.5.

Iteración k	$x^{(k)}$	$\lvert f(x^{(k)}) \rvert$
1	1.50000	2.37500
2	1.25000	1.79688
3	1.37500	0.16211
4	1.31250	0.84839
5	1.34375	0.35098
6	1.35938	0.09641
7	1.36719	0.03236
8	1.36328	0.03215
9	1.36523	0.00007

La convergencia del método de bisección es lenta pero casi nunca falla. Cuando todos los demás métodos fracasan, la opción obligada es emplear bisección.
El siguiente es el pseudocódigo de un procedimiento que encuentra una raíz aproximada de f(x) = 0 con el método de bisección.

```
proc biseccion(real func f, real &a, real &b, real eps, entero maxiter, real &x, entero &error)
!
!  Este procedimiento encuentra una aproximación a una raíz de f(x)=0 con el método iterativo
!  de la bisección.
!  Descripción de parámetros:
!  f              Función externa que evalúa a la función f(x) .
!  a              Límite inferior de un intervalo que contiene un número impar de raíces de
!                 f(x).
!  b              Límite superior de un intervalo que contiene un número impar de raíces de
!                 f(x).
!  eps            Máxima magnitud posible de f(x). Cuando el valor absoluto de f evaluada
!                 en la última iteración x es menor que eps, se considera haber llegado a la
!                 solución.
!  maxiter        Número máximo de iteraciones.
!  x              Última iteración calculada. Si en error se devuelve el valor 0, entonces x
!                 también es una raíz de f(x).
!  error          Bandera cuyo valor indica el tipo de situación detectada:
!                 error = 0      sin error, se alcanzó la convergencia
!                 error = 1      no se alcanzó la convergencia
!                 error = 2      f(a)*f(b)>0 y entonces en el intervalo [a,b] no existe un
!                                número impar de raíces.
!
entero iter
real fa, fb, fx
error = 2
fa = f(a)
fb = f(b)
si fa*fb > 0.0 entonces regresa
error = 0
ejecuta  iter = 1, maxiter
{
        x = (a + b)/2  ! cálculo de una nueva iteración
        fx = f(x)
        si ABS(fx) < eps entonces  regresa    ! se checa la convergencia
        si fa*fx ≤ 0.0 entonces      ! preparación de la siguiente iteración
                b = x
        otro
                a = x
                fa = fx
}
error = 1
regresa
fin
```

1.6 Orden de convergencia

El orden de convergencia de un método iterativo se define como sigue:

Si $|e_{n+1}|$ tiende a $K*|e_n|^p$ cuando n tiende a infinito , se dice que el método es de orden p.

Punto fijo exhibe convergencia lineal (orden de convergencia p = 1):

$$|e_{n+1}| = |g'(\zeta)|\,|e_n|$$

Newton Raphson exhibe convergencia cuadrática (orden de convergencia p = 2) para raíces simples:

$$|e_{n+1}| = g''(\zeta)|\,e_n|^2 / 2 \quad \text{donde } g(x) = x^{(n)} - f(x^{(n)}) / f'(x^{(n)})$$

Secante, orden de convergencia p:

$$p = (1 + \sqrt{5}) / 2 \approx 1.62$$

Bisección, convergencia lineal (orden de convergencia p = 1) :

$$|e_{n+1}| = 0.5|\,e_n|$$

2. Solución de sistemas de ecuaciones lineales

El sistema de n ecuaciones con n incógnitas

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots\ldots\ldots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots\ldots\ldots + a_{2n}x_n = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \ldots\ldots\ldots + a_{3n}x_n = b_3$$
$$\vdots \qquad\qquad \vdots \quad \vdots$$
$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots\ldots\ldots + a_{nn}x_n = b_n$$

se puede escribir en forma matricial como $\mathbf{Ax} = \mathbf{b}$, donde $\mathbf{A}$ es la matriz de coeficientes,

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots\ldots\ldots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots\ldots\ldots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \ldots\ldots\ldots & a_{3n} \\ \vdots & & & & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \ldots\ldots\ldots & a_{nn} \end{bmatrix}$$

$\mathbf{x}$ es el vector de incógnitas y $\mathbf{b}$ el vector de términos del lado derecho

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}$$

Existen dos tipos de métodos para resolver el problema $\mathbf{Ax} = \mathbf{b}$; los exactos y los iterativos. Los métodos exactos son llamados así porque, en ausencia de errores de redondeo, obtienen la solución analítica o exacta y están basados en la eliminación de Gauss o en la factorización LU. Los métodos del segundo tipo encuentran una solución aproximada mediante procesos iterativos. Para sistemas con matrices de coeficientes densas y relativamente pequeñas, se prefieren los

métodos exactos. En el caso de sistemas con matrices de coeficientes grandes y poco densas, son recomendados los métodos iterativos.

2.1 Eliminación de Gauss

Es un proceso que convierte a la matriz de coeficientes **A** de nxn en una matriz triangular superior mediante la aplicación sistemática de *transformaciones elementales de renglón*. Una vez obtenida la matriz triangular superior se aplica un procedimiento conocido como *sustitución hacia atrás* para obtener el vector solución **x**.
Las transformaciones elementales de renglón son:
1.- La fila i de una matriz puede ser multiplicada por un constante $\lambda \neq 0$

$\lambda R_i \rightarrow R_i$

2.- A la fila i de un matriz le puede ser sumada otra fila j de la misma matriz multiplicada por una constante λ

$\lambda R_j + R_i \rightarrow R_i$

3.- Las filas i y j de una matriz pueden ser intercambiadas

$R_i \leftrightarrow R_j$

La eliminación de Gauss consta en n-1 pasos de eliminación y cada uno de ellos consiste en:
1.- Seleccionar un pivote: en el i-ésimo paso de eliminación se escoge como pivote al i-ésimo elemento de la diagonal principal de **A**, estrategia conocida *como pivoteo natural.*
2.- Colocar ceros debajo del pivote en la misma columna.

Ejemplo
Resolver el sistema

$$x_1 - x_2 + x_3 = -4$$

$$5x_1 - 4x_2 + 3x_3 = -12$$

$$2x_1 + x_2 + x_3 = 11$$

con eliminación de Gauss y sustitución hacia atrás. Use una estrategia de pivoteo natural.

En primer lugar, se forma una matriz agregando una columna a la matriz de coeficientes. La columna añadida es el vector de términos del lado derecho y la matriz así formada se conoce como *matriz aumentada* del sistema:

1.00000	-1.00000	1.00000	- 4.00000
5.00000	-4.00000	3.00000	-12.00000
2.00000	1.00000	1.00000	11.00000

Primer paso de eliminación:
El pivote es el primer elemento de la diagonal principal de la matriz de coeficientes, a_{11} = 1.00000. Ahora se trata de colocar ceros debajo del pivote en la primera columna; las transformaciones de renglón requeridas para tal efecto son:

$-5R_1 + R_2 \rightarrow R_2$

$-2R_1 + R_3 \rightarrow R_3$

La fila del pivote es multiplicada por el negativo de una cantidad conocida como *multiplicador* y es el elemento que se quiere convertir en cero dividido entre el pivote.
Aplicando esta transformaciones a la matriz aumentada, el resultado es:

1.00000	-1.00000	1.00000	-4.00000
0.00000	1.00000	-2.00000	8.00000
0.00000	3.00000	-1.00000	19.00000

Segundo paso de eliminación:
El pivote es ahora a_{22} = 1.00000 y el multiplicador es m =3.00000/1.00000 = 3.00000. Por lo tanto la transformación elemental requerida es:

$-3R_2 + R_3 = R_3$

y la matriz aumentada se convierte en:

1.00000	-1.00000	1.00000	-4.00000
0.00000	1.00000	-2.00000	8.00000
0.00000	0.00000	5.00000	-5.00000

Con estos dos pasos termina la eliminación de Gauss y la parte correspondiente a la matriz de coeficientes es una matriz triangular superior.
La última matriz aumentada define un sistema cuya solución es equivalente a la del sistema original:

$$1.00000x_1 - 1.00000x_2 + 1.00000x_3 = -4.00000$$

$$1.00000x_2 - 2.00000x_3 = 8.00000$$

$\det(\mathbf{A}) = (-1)^0\ [(1.00000)(1.00000)(5.00000)] = 5.00000$

Es posible contar el número de operaciones necesarias para resolver un sistema lineal con eliminación de Gauss y sustitución hacia atrás:

Multiplicaciones o divisiones: $n^3/3 + n^2 - n/3$

Sumas o restas: $n^3/3 + n^2/2 - 5n/6$

En análisis numérico se emplea la notación $O(n^3)$ la cual indica que el número de operaciones es "de orden n^3 ". La conclusión final es: la eliminación gaussiana con sustitución hacia atrás es un proceso muy costoso en términos de operaciones aritméticas.
A continuación, un pseudocódigo de un procedimiento que resuelve el sistema **Ax** = **b** con eliminación gaussiana y sustitución hacia atrás. Pivoteo natural es empleado.

```
proc eliminacion_gauss(entero n, real &A[n, n+1], real &x[n], lógico &solucion_unica)
!
! Este procedimiento resuelve un sistema lineal de ecuaciones con eliminación gaussiana y
! sustitución hacia atrás. Se emplea una estrategia de pivoteo natural.
! Descripción de parámetros:
! n                Número de ecuaciones y de incógnitas.
! A                Matriz aumentada del sistema, de n filas y n+1 columnas.
! x                Vector solución, de n elementos. Si en solucion_unica se devuelve
!                  el valor VERDADERO, entonces x es la solución única.
! solucion_unica   Bandera que indica la existencia de un solución única:
!                  solucion_unica=VERDADERO     el sistema tiene solución única.
!                  solucion_unica= FALSO        la matriz de coeficientes es singular
!                                               y no hay solución única.
!
entero i, p, k, j
real auxiliar, multiplicador, suma
solucion_unica = FALSO
! Eliminación de gauss con pivoteo natural
ejecuta i = 1, n-1
{
    ejecuta p = i, n     ! selección del pivote
        si ABS(A[p, i]) ≥ 1X10^-10 entonces salida
    si p > n entonces regresa
    si p ≠ i entonces   ! intercambio de las filas p e i, en caso necesario
        ejecuta k = 1, n+1
            auxiliar = A[p, k]
            A[p, k] = A[i, k]
            A[i, k] = auxiliar
    ejecuta j = i+1, n   ! colocación de ceros debajo del pivote
        multiplicador = A[j, i]/A[i, i]
        ejecuta k = i, n+1
            A[j, k] = A[j, k] - multiplicador*A[i, k]
```

$$5.00000x_3 = -5.00000$$

Esta solución **x** se puede encontrar fácilmente:

$x_3 = -5.00000 / 5.00000 = -1.00000$

$x_2 = (8.00000 + 2.00000x_3) / 1.00000 = (8.00000+2.00000(-1.00000)) / 1.00000 = 6.00000$

$x_1 = (-4.00000 + 1.00000x_2 - 1.00000x_3) / 1.00000$
$x_1 = (-4.0000 + 1.00000(6.00000) - 1.00000(-1.0000)) / 1.00000 = 3.00000$

Este proceso es la sustitución hacia atrás, conocido así porque se obtienen los valores de las incógnitas comenzando con la última x_n y terminando con la primera x_1. La siguientes fórmulas describen la sustitución hacia atrás y en ellas la letra a denota a los elementos de la matriz aumentada:

$$x_n = \frac{a_{n,n+1}}{a_{n,n}}$$

$$x_i = \frac{a_{i,n+1} - \sum_{j=i+1}^{n} x_j a_{i,j}}{a_{i,i}} \qquad \text{para } i = n-1, n-2, \ldots, 1$$

El último ejercicio muestra que para poder calcular los multiplicadores en cada paso de eliminación, el pivote necesariamente debe ser distinto de cero. Por lo tanto, si en el i-ésimo paso de eliminación a_{ii} es cero, se debe intercambiar la fila i con alguna otra colocada debajo de ella de tal forma que se lleve a la posición del pivote un número distinto de cero. Si no es posible colocar un pivote diferente de cero, entonces el sistema lineal no tiene solución única.
Otra posible situación en la cual no existe solución única es aquella en la cual después del último paso de eliminación el elemento a_{nn} es igual a cero.
Con eliminación de Gauss también es posible calcular el determinante de la matriz de coeficientes:

$$\det(\mathbf{A}) = (-1)^{ncf} \prod_{i=1}^{n} s_{ii}$$

donde ncf es el número de cambios de filas efectuados en la eliminación y s_{ii} representa los elementos de la diagonal principal de la matriz triangular superior resultado de la eliminación. Por lo tanto, el determinante de la matriz de coeficientes del último ejemplo es:

```
}
si ABS(A[n, n]) < 1X10^-10 entonces regresa
solucion_unica = VERDADERO
! sustitución hacia atrás
x[n] = A[n, n+1]/A[n, n]
ejecuta i = n-1, 1, -1
{
        suma = 0.0
        ejecuta j = i+1, n
                suma = suma + A[i, j]*x[j]
        x[i] = (A[i, n+1] - suma)/A[i, i]
}
regresa
fin
```

2.2 Pivoteo máximo de columna

Si los cálculo involucrados en la eliminación gaussiana se efectuaran en una computadora o en una calculadora con longitud de palabra infinita, los resultados obtenidos serían exactos. Pero como lo anterior no es posible y las computadoras solo pueden representar y operar un número finito de dígitos, necesariamente los resultados serán aproximados debido a los inevitables errores de redondeo.
Una forma de minimizar los errores de redondeo en la eliminación de Gauss es emplear precisiones más altas, es decir, usar un mayor número de dígitos en los cálculos. Otra manera es emplear alguna *técnica de pivoteo*. De todas las estrategias de pivoteo la más simple y frecuentemente muy efectiva, es el *pivoteo parcial o pivoteo máximo de columna:* en cada paso de eliminación siempre se escoge como pivote al elemento de mayor valor absoluto entre los colocados en y abajo de la diagonal principal:

Pivoteo máximo de columna
En el i-ésimo paso de eliminación, se determina un valor $p \geq i$ tal que:

$$|a_{pi}| = \max_{i \leq k \leq n} |a_{ki}| \qquad \text{para } i = 1,2,....,n-1 \text{ pasos de eliminación}$$

Ejemplo
Resuelva el sistema

$$x_1 + 3x_2 - 2x_3 = 7$$

$$4x_1 - x_2 + 3x_3 = 10$$

$$-5x_1 + 2x_2 + 3x_3 = 7$$

empleando eliminación de Gauss con pivoteo máximo de columna.

La matriz aumentada del sistema es:

1.00000	3.00000	-2.00000	7.00000
4.00000	-1.00000	3.00000	10.00000
-5.00000	2.00000	3.00000	7.00000

Primer paso de eliminación:
El pivote se escoge como el elemento con el mayor valor absoluto entre los elementos de la primera columna desde la fila 1 hasta la 3, por lo tanto el pivote es $a_{31} = -5.00000$. Como el pivote está en la fila 3, es necesario intercambiar la fila 1 con la 3 para colocar en el primer renglón al -5.00000.

$R_1 \leftrightarrow R_3$

-5.00000	2.00000	3.00000	7.00000
4.00000	-1.00000	3.00000	10.00000
1.00000	3.00000	-2.00000	7.00000

Los multiplicadores son ahora 4.00000/-5.00000 y 1.00000/-5.00000 y las transformaciones elementales que colocan ceros debajo de a_{11} son:

$-(4.00000/-5.00000)R_1 + R_2 \rightarrow R_2$

$-(1.00000/-5.00000)R_1 + R_3 \rightarrow R_3$

La matriz aumentada queda al final de este primer paso de eliminación de esta manera:

-5.00000	2.00000	3.00000	7.00000
0.00000	0.60000	5.40000	15.60000
0.00000	3.40000	-1.40000	8.40000

Segundo paso de eliminación:
Aquí los posibles pivotes son $a_{22} = 0.60000$ y $a_{23} = 3.40000$. Como $|a_{23}| > |a_{22}|$, entonces el pivote es $a_{23} = 3.40000$ y por lo tanto se requiere intercambiar las filas 2 y 3:

$R_2 \leftrightarrow R_3$

-5.00000	2.00000	3.00000	7.00000
0.00000	3.40000	- 1.40000	8.40000
0.00000	0.60000	5.40000	15.60000

El multiplicador es 0.60000/3.40000 y la transformación que coloca un cero debajo del pivote es:

$-(0.60000/3.40000)\ R_2 + R_3 \rightarrow R_3$

Entonces la matriz aumentada al final queda así:

-5.00000	2.00000	3.00000	7.00000
0.00000	3.40000	- 1.40000	8.40000

0.00000 0.00000 5.64706 14.11765

La sustitución hacia atrás genera el vector solución $\mathbf{x} = [1.50000, 3.50000, 2.50000]^T$.

Pseudocódigo de un procedimiento que resuelve un sistema lineal, con eliminación de Gauss y pivoteo máximo de columna:

```
proc eliminacion_gauss_pm(entero n, real &A[n, n+1], real &x[n], lógico &solucion_unica)
!
!  Este procedimiento resuelve un sistema lineal de ecuaciones con eliminación gaussiana y
!  sustitución hacia atrás. Se emplea una estrategia de pivoteo máximo de columna.
!  Descripción de parámetros:
!  n                Número de ecuaciones y de incógnitas.
!  A                Matriz aumentada del sistema, de n filas y n+1 columnas.
!  x                Vector solución, de n elementos. Si en solucion_unica se devuelve
!                   el valor VERDADERO, entonces x es la solución única.
!  solucion_unica   Bandera que indica la existencia de un solución única:
!                   solucion_unica=VERDADERO     el sistema tiene solución única.
!                   solucion_unica= FALSO        la matriz de coeficientes es singular
!                                                y no hay solución única.
!
entero i, p, k, j
real auxiliar, multiplicador, suma
solucion_unica = FALSO
! Eliminación de gauss con pivoteo máximo de columna
ejecuta i = 1, n-1
{
        p = i
        ejecuta j = i+1, n      ! selección del pivote
                si ABS(A[p, i]) ≥ ABS(A[j, i]) entonces p = j
        si ABS(A[p, i]) > 1X10^-10 entonces regresa
        si p ≠ i entonces   ! intercambio de las filas p e i, en caso necesario
                ejecuta k = 1, n+1
                        auxiliar = A[p, k]
                        A[p, k] = A[i, k]
                        A[i, k] = auxiliar
        ejecuta j = i+1, n    ! colocación de ceros debajo del pivote
                multiplicador = A[j, i]/A[i, i]
                ejecuta k = i, n+1
                        A[j, k] = A[j, k] - multiplicador*A[i, k]
}
si ABS(A[n, n]) < 1X10^-10 entonces regresa
solucion_unica = VERDADERO
! sustitución hacia atrás
x[n] = A[n, n+1]/A[n, n]
ejecuta i = n-1, 1, -1
{
```

```
        suma = 0.0
        ejecuta j = i+1, n
                suma = suma + A[i, j]*x[j]
        x[i] = (A[i, n+1] - suma)/A[i, i]
}
regresa
fin
```

2.3 Descomposición LU

Dada una matriz **A** de nxn , es posible encontrar las matrices **L** y **U** tales que:

$$\mathbf{A = LU}$$

Las matrices **L** y **U** son también de nxn. **L** es triangular inferior y **U** es triangular superior. Para mostrar los detalles de la descomposición LU, considere una matriz **A** de 4x4:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A = L\ U}$$

Para poder realizar la descomposición es necesario asignar valores arbitrarios a una de los diagonales, ya sea la de **L** o la de **U**. Si los elementos de la diagonal principal de **U** se hacen iguales a uno, se trata de la *descomposición de Crout*, como es el caso mostrado arriba. Si los elementos de la diagonal principal de **L** se toman iguales a uno, entonces es la *descomposición de Doolittle.*
Multiplicando los renglones de **L** por la primera columna de **U**:

$l_{11}\,(1) = a_{11}$

$l_{21}\,(1) = a_{21}$

$l_{31}\,(1) = a_{31}$

$l_{41}\,(1) = a_{41}$

Multiplicando el primer renglón de **L** por las columnas de **U**:

$l_{11}\,(1) = a_{11}$

$l_{11}\,u_{12} = a_{12}$ y por lo tanto $u_{12} = a_{12} / l_{11}$

FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA

CURSOS ABIERTOS

DIPLOMADO DE MATEMÁTICAS APLICADAS A LA INGENIERÍA

"MÉTODOS NUMÉRICOS"

MÓDULO II

TEMA: APUNTES GENERALES

CA-484

EXPOSITOR: M. EN C. ABEL VALDÉS RAMÍREZ

12 DE NOVIEMBRE AL 3 DE DICIEMBRE DE 2005-11-18 PALACIO DE MINERÍA

$l_{11}\, u_{13} = a_{13}$ y por lo tanto $u_{13} = a_{13} / l_{11}$

$l_{11}\, u_{14} = a_{14}$ y por lo tanto $u_{14} = a_{14} / l_{11}$

Multiplicando los renglones de **L** por la segunda columna de **U**:

$l_{11}\, u_{12} = a_{12}$

$l_{21}\, u_{12} + l_{22}\, (1) = a_{22}$ y por lo tanto $l_{22} = a_{22} - l_{21}\, u_{12}$

$l_{31}\, u_{12} + l_{32}\, (1) = a_{32}$ y por lo tanto $l_{32} = a_{32} - l_{31}\, u_{12}$

$l_{41}\, u_{12} + l_{42}\, (1) = a_{42}$ y por lo tanto $l_{42} = a_{42} - l_{41}\, u_{12}$

Multiplicando el segundo renglón de **L** por las columnas de **U**:

$l_{21}\, (1) = a_{21}$

$l_{21}\, u_{12} + l_{22}\, (1) = a_{22}$

$l_{21}\, u_{13} + l_{22}\, u_{23} = a_{23}$ y por lo tanto $u_{23} = (a_{23} - l_{21}\, u_{13}) / l_{22}$

$l_{21}\, u_{14} + l_{22}\, u_{24} = a_{24}$ y por lo tanto $u_{24} = (a_{24} - l_{21}\, u_{14}) / l_{22}$

Multiplicando los renglones de **L** por la tercera columna de **U**:

$l_{11}\, u_{13} = a_{13}$

$l_{21}\, u_{13} + l_{22}\, u_{23} = a_{23}$

$l_{31}\, u_{13} + l_{32}\, u_{23} + l_{33}\, (1) = a_{33}$ y por lo tanto $l_{33} = a_{33} - l_{31}\, u_{13} - l_{32}\, u_{23}$

$l_{41}\, u_{13} + l_{42}\, u_{23} + l_{43}\, (1) = a_{43}$ y por lo tanto $l_{43} = a_{43} - l_{41}\, u_{13} - l_{42}\, u_{23}$

Multiplicando el tercer renglón de **L** por las columnas de **U**:

$l_{31}\, (1) = a_{31}$

$l_{31}\, u_{12} + l_{32}\, (1) = a_{32}$

$l_{31}\, u_{13} + l_{32}\, u_{23} + l_{33}\, (1) = a_{33}$

$l_{31}\, u_{14} + l_{32}\, u_{24} + l_{33}\, u_{34} = a_{34}$ y por lo tanto $u_{34} = (a_{34} - l_{31}\, u_{14} - l_{32}\, u_{24}) / l_{33}$

Multiplicando los renglones de **L** por la cuarta columna de **U**:

$l_{11}\, u_{14} = a_{14}$

$l_{21}\, u_{14} + l_{22}\, u_{24} = a_{24}$

$l_{31}\, u_{14} + l_{32}\, u_{24} + l_{33}\, u_{34} = a_{34}$

$l_{41}\, u_{14} + l_{42}\, u_{24} + l_{43}\, u_{34} + l_{44}\,(1) = a_{44}$ y por lo tanto $l_{44} = a_{44} - l_{41}\, u_{14} - l_{42}\, u_{24} - l_{43}\, u_{34}$

Entonces, este procedimiento basado en las propiedades del producto de matrices, permite encontrar a los elementos desconocidos de **L** y **U**. Las siguientes ecuaciones representan a dicho procedimiento:

$$l_{i1} = a_{i1} \qquad ,i = 1,2,\ldots,n$$

$$u_{1j} = \frac{a_{1j}}{l_{11}} \qquad ,j = 2,3,\ldots,n$$

Para $k = 2,3,\ldots,n-1$

$$l_{ik} = a_{ik} - \sum_{j=1}^{k-1} l_{ij} u_{jk} \qquad ,i = k,k+1,\ldots,n$$

$$u_{ki} = \frac{a_{ki} - \sum_{j=1}^{k-1} l_{kj} u_{ji}}{l_{kk}} \qquad ,i = k+1,k+2,\ldots,n$$

$$l_{nn} = a_{nn} - \sum_{j=1}^{n-1} l_{nj} u_{jn}$$

Una vez obtenida la descomposición LU de la matriz de coeficientes **A**, es posible resolver el sistema **Ax** = **b**: como **A** = **LU**, el sistema original se puede escribir como (**LU**)**x** = **b** y si se define un vector **z** como **z** = **Ux**, entonces ahora queda el sistema **Lz** = **b** el cual se puede resolver mediante *sustitución hacia adelante*. Conocido **z**, entonces se resuelve **Ux** = **z** con sustitución hacia atrás encontrándose así al vector **x**.

Ejemplo
Resolver el sistema

$3x_1 - x_2 + 2x_3 = 12$

$x_1 + 2x_2 + 3x_3 = 11$

$2x_1 - 2x_2 - x_3 = 2$

con descomposición LU.

$l_{11} = a_{11} = 3.00000$

$l_{21} = a_{21} = 1.00000$ $\qquad u_{12} = a_{12} / l_{11} = -1.00000/3.00000 = -0.33333$

$l_{31} = a_{31} = 2.00000$ $\qquad u_{13} = a_{13} / l_{11} = 2.00000/3.00000 = 0.66667$

3	-1	2		3.00000	0	0	1	-0.33333	0.66667
1	2	3	=	1.00000	l_{22}	0 *	0	1	u_{23}
2	-2	-1		2.00000	l_{32}	l_{33}	0	0	1

$l_{22} = a_{22} - l_{21} u_{12} = 2.00000 - (1.00000)(-0.33333) = 2.33333$

$l_{32} = a_{32} - l_{31} u_{12} = -2.00000 - (2.00000)(-0.33333) = -1.33333$

$u_{23} = (a_{23} - l_{21} u_{13}) / l_{22} = (3.00000 -(1.00000)(0.66667)) / 2.33333 = 1.00000$

3	-1	2							
1	2	3	=						
2	-2	-1							

3.00000	0	0		1	-0.33333	0.66667			
1.00000	2.33333	0	*	0	1	1.00000			
2.00000	-1.33333	l_{33}		0	0	1			

$l_{33} = a_{33} - l_{31} u_{13} - l_{32} u_{23} = -1.00000 - (2.00000)(0.66667) - (-1.33333)(1.00000) = -1.00000$

Entonces, la descomposición LU es:

3	-1	2							
1	2	3	=						
2	-2	-1							

3.00000	0	0		1	-0.33333	0.66667			
1.00000	2.33333	0	*	0	1	1.00000			
2.00000	-1.33333	-1.00000		0	0	1			

Lz = **b** es el sistema:

$3.00000z_1$ $\qquad 12.00000$

$1.00000z_1 + 2.33333z_2$ $\qquad 11.00000$

$2.00000z_1 - 1.33333z_1 - 1.00000z_3 = \quad 2.00000$

el cual se resuelve con sustitución hacia adelante. Este algoritmo es llamado así porque la primera incógnita que se determina es z_1 y la última z_n.

$z_1 = 12.00000 / 3.00000 = 4.00000$

$z_2 = (11.00000 - z_1) / 2.33333 = (11.00000 - 4.00000) / 2.33333 = 3.00000$

$z_3 = (2.00000 - 2z_1 + 1.33333z_2) / (-1.00000)$

$z_3 = (2.00000 - 2.00000(4.00000) + 1.33333(3.00000)) / (-1.00000) = 2.000000$

Por último, es necesario resolver el sistema $\mathbf{Ux} = \mathbf{z}$ mediante sustitución hacia atrás:

$1.00000x_1 - 0.33333x_2 + 0.66667x_3 = 4.00000$

$1.00000x_2 + 1.00000x_3 = 3.00000$

$1.00000x_3 = 2.00000$

$x_3 = 2.00000 / 1.00000 = 2.00000$

$x_2 = (3.00000 - x_3) / 10.00000 = (3.00000 - 2.00000) / 1.00000 = 1.00000$

$x_1 =(4.00000 + 0.33333x_2 - 0.66667x_3) / 1.00000$

$x_1 = (4.00000 + 0.33333(1.00000) - 0.66667(2.00000)) / 1.00000 = 3.00000$

Si se conoce la descomposición LU de **A**, la solución de $\mathbf{Ax} = \mathbf{b}$ implica solamente resolver $\mathbf{Lz} = \mathbf{b}$ y $\mathbf{Ux} = \mathbf{z}$. El trabajo computacional asociado con cualquiera de estas soluciones es de $O(n^2/2)$ multiplicaciones o divisiones. Esta es la razón por la que se prefiere la descomposición LU para resolver varios sistemas lineales con la misma matriz de coeficientes. La descomposición LU junto con la solución de $\mathbf{Lz} = \mathbf{b}$ y de $\mathbf{Ux} = \mathbf{z}$ requieren un total de $O(n^3/3)$ multiplicaciones o divisiones.

En el algoritmo de la reducción de Crout, los elementos de la diagonal principal de L deben ser distintos de cero. Si en algún momento, se detecta que $l_{ii} = 0$ entonces es obligado un intercambio de filas. Más específicamente; si $l_{ii} = 0$, se debe intercambiar la fila i con alguna de las filas colocadas debajo de ella, cuidando que el intercambio solo afecte a los elementos de la parte triangular inferior. El mismo intercambio de filas deberá realizarse también en **A** y en **b**.

Si no es posible encontrar un $l_{ii} \neq 0$, entonces la matriz **A** es singular y el sistema $\mathbf{Ax} = \mathbf{b}$ no tiene solución única.

Después de los cambios de renglones, al terminar la descomposición LU la igualdad $\mathbf{A} = \mathbf{LU}$ sigue siendo válida, **L** sigue siendo triangular inferior y **U** triangular superior pero **A** ahora es la matriz original afectada por los cambios de filas.

Es factible implementar un pivoteo máximo de columna en la descomposición de Crout. En este

caso, se busca colocar en la posición l_{ii} al elemento de mayor valor absoluto entre los colocados en y debajo de la diagonal principal de **L**. El intercambio de filas deberá afectar solamente a la parte triangular inferior de **L** y también deberá ser aplicado a la matriz **A** y al vector **b**.
A continuación, el pseudocódigo de procedimientos que resuelven sistemas lineales con descomposición LU:

```
proc LU(entero n, real &A[n, n], real &L[n, n], real &U[n, n],entero &info_cambiosf[n], lógico
&exito)
!
! Factorización  A = LU mediante reducción de Crout.
! Descripción de parámetros:
! n                 Orden de las matrices cuadradas A, L y U.
! A                 Matriz por factorizar. Como resultado de la factorización, en el regreso
!                   A puede tener algunas filas intercambiadas.
! L                 Matriz triangular inferior.
! U                 Matriz triangular superior.
! info_cambiosf     Vector que contiene información sobre los cambios de filas efectuados
!                   durante la factorización.
! exito             Bandera cuyo valor indica el tipo de situación detectada durante el
!                   proceso:
!                   exito = VERDADERO     factorización exitosa, A = LU
!                   exito = FALSO         no fue posible la factorización o A es
!                                         singular.
!
entero i, j, k, p
real auxiliar, suma
ejecuta i = 1,n
        info_cambiosf[i] = i    ! cuando info_cambiosf[i] = i, no se ha intercambiado la fila i
ejecuta i = 1,n
        U[i, i] = 1.0    ! se asigna el valor 1.0 a los elementos de la diagonal principal de U
exito = FALSO
ejecuta i = 1,n          ! búsqueda de un L[1, 1] diferente de cero
        si ABS(A[i,1]) ≥ 1X10^-10 entonces salir
si i > n entonces regresa
si i ≠ 1 entonces
{
        ejecuta j = 1,n    ! intercambio de las filas 1 e i
                auxiliar = A[1,j]
                A[1, j] = A[i, j]
                A[i, j] = auxiliar
        ejecuta j = 1, n
                auxiliar = info_cambiosf[1]
                info_cambiosf[1] = info_cambios_f[i]
                info_cambiosf[i] = auxiliar
}
ejecuta i = 1,n          ! cálculo de la primera columna de L
        L[i,1] = A[i,1]
```

```
ejecuta j = 2,n          ! cálculo de los elementos restantes de la primera fila de U
        U[1, j] = A[1, j] / L[1,1]
ejecuta k = 2,n
{
        ejecuta p = k,n          ! búsqueda de un L[i, i] distinto de cero
                suma = 0.0
                ejecuta j = 1,k-1
                        suma = suma + L[p, j]*U[j, k]
                auxiliar = A[p, k] - suma
                si ABS(A[p, k]) ≥ 1X10^-10 entonces  salir
        si p > n entonces  regresar
        si p≠ k entonces
        {
                ejecuta j = 1,n
                        auxiliar = A[k, j]       ! intercambio de las filas k y p en A
                        A[k, j] = A[p, j]
                        A[p, j] = auxiliar
                ejecuta j = 1,k-1                ! intercambio de las filas k y p en L
                        auxiliar = L[k, j]
                        L[k, j] = L[p, j]
                        L[p, j] =auxiliar
                ejecuta j = 1,n
                        auxiliar = info_cambiosf[k]                    ! intercambio de los
                        info_cambiosf[k] = info_cambiosf[p]            ! elementos con índices k y p
                        info_cambiosf[p] = auxiliar                    ! en info_cambiosf
        {
        ejecuta i = k,n
                suma = 0.0
                ejecuta j = 1,k-1                ! cálculo de columna k de L
                        suma = suma + L[i, j]*U[j, k]
                L[i, k ] = A[i,k] - suma
        ejecuta i = k+1,n                ! cálculo de los elementos restantes de la fila k de U
                suma = 0.0
                ejecuta j = 1,k-1
                        suma = suma + L[k, j]*U[j, i]
                U[k, i] = (A[k, i] - suma)/L[k, k]
}
suma = 0.0
ejecuta j = 1,n-1        ! cálculo del último elemento de la diagonal principal de L
        suma = suma + L[n, j]*U[j, n]
L[n, n] = A[n, n] - suma
si ABS(L[n, n]) < 1X10^-10 entonces  regresa
exito = VERDADERO
regresa
fin

proc solve_lu(entero n, real L[n, n], real U[n, n], entero info_cambiosf[n], real b[n], real
```

```
                &x[n], lógico &solucion_unica)
!
! Dada la factorización A = LU, este procedimiento determina el vector solución x de Ax = b.
! Descripción de parámetros:
! n                Orden de las matrices cuadradas L y U. También es el número de
!                  incógnitas.
! L                Matriz triangular inferior.
! U                Matriz triangular superior.
! info_cambiosf    Vector que contiene información sobre los cambios de filas efectuados
!                  durante la factorización A = LU.
! b                Vector de términos del lado derecho del sistema lineal.
! x                Vector solución.
! solucion_unica   Bandera cuyo valor indica la existencia de la solución única:
!                  solucion_unica = VERDADERO    x es la solución única
!                  solucion_unica = FALSO        el sistema no tiene solución única.
!
entero i, j, auxiliar
real z[n], suma
!
! Solución de Lz = b
!
solucion_unica = FALSO
si ABS(L[1, 1]) < 1X10^-10 entonces regresa
z[1] = b[info_cambiosf[1]]/L[1, 1]          ! sustitución hacia adelante
ejecuta i = 2, n
        si ABS(L[i, i]) < 1X10^-10 entonces regresa
        suma = 0.0
        ejecuta j = 1, i-1
                suma = suma + L[i, j]*z[j]
        z[i] = (b[info_cambiosf[i]] - suma)/L[i, i]
!
! Solución de Ux = z
!
si ABS(U[n, n]) < 1X10^-10 entonces regresa
x[n] = z[n]/U[n, n]                         ! sustitución hacia atrás
ejecuta i = n-1, 1, -1
        si ABS(U[i, i]) < 1X10^-10 entonces regresa
        suma = 0.0
        ejecuta j = i+1, n
                suma = suma + U[i, j]*x[j]
        x[i] = (z[i] - suma)/U[i, i]
 solucion_unica = VERDADERO
regresa
fin
```

2.4 Métodos iterativos: Jacobi y Gauss Seidel

Dado el sistema lineal

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots\ldots\ldots\ldots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots\ldots\ldots\ldots + a_{2n}x_n = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \ldots\ldots\ldots\ldots + a_{3n}x_n = b_3$$

$$\vdots \qquad\qquad\qquad\qquad \vdots \quad \vdots$$

$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots\ldots\ldots\ldots + a_{nn}x_n = b_n$$

es factible despejar a x_1 de la primera ecuación, a x_2 de la segunda ecuación, a x_3 de la tercera y así sucesivamente:

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3 - \ldots\ldots\ldots\ldots - a_{1n}x_n) / a_{11}$$

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3 - \ldots\ldots\ldots\ldots - a_{2n}x_n) / a_{22}$$

$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2 - \ldots\ldots\ldots\ldots - a_{3n}x_n) / a_{33}$$

$$\vdots \quad \vdots \qquad\qquad\qquad\qquad \vdots \quad \vdots$$

$$x_n = (b_n - a_{n1}x_1 - a_{n2}x_2 - \ldots\ldots\ldots\ldots - a_{n,n-1}x_{n-1}) / a_{nn}$$

Entonces se parte de una estimación inicial de la solución, $\mathbf{x}^{(0)}$, la cual se sustituye en las últimas ecuaciones para producir un nueva estimación, $\mathbf{x}^{(1)}$:

$$x_1^{(1)} = (b_1 - a_{12}x_2^{(0)} - a_{13}x_3^{(0)} - \ldots\ldots\ldots\ldots - a_{1n}x_n^{(0)}) / a_{11}$$

$$x_2^{(1)} = (b_2 - a_{21}x_1^{(0)} - a_{23}x_3^{(0)} - \ldots\ldots\ldots\ldots - a_{2n}x_n^{(0)}) / a_{22}$$

$$x_3^{(1)} = (b_3 - a_{31}x_1^{(0)} - a_{32}x_2^{(0)} - \ldots\ldots\ldots\ldots - a_{3n}x_n^{(0)}) / a_{33}$$

$$\vdots \quad \vdots \qquad\qquad\qquad\qquad \vdots \quad \vdots$$

$$x_n^{(0)} = (b_n - a_{n1}x_1^{(0)} - a_{n2}x_2^{(0)} - \ldots\ldots\ldots\ldots - a_{n,n-1}x_{n-1}^{(0)}) / a_{nn}$$

El vector $\mathbf{x}^{(1)}$ se sustituye en esas mismas ecuaciones para obtener ahora a $\mathbf{x}^{(2)}$. Este procedimiento se repite entonces para calcular las estimaciones $\mathbf{x}^{(3)}$, $\mathbf{x}^{(4)}$, $\mathbf{x}^{(5)}$, ... Lo anterior se puede generalizar mediante las ecuaciones:

$$x_1^{(k+1)} = (b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \ldots\ldots\ldots\ldots - a_{1n}x_n^{(k)}) / a_{11}$$

$$x_2^{(k+1)} = (b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - \ldots\ldots\ldots\ldots - a_{2n}x_n^{(k)}) / a_{22}$$

$$x_3^{(k+1)} = (b_3 - a_{31}x_1^{(k)} - a_{32}x_2^{(k)} - \ldots\ldots\ldots - a_{3n}x_n^{(k)}) / a_{33}$$

. . . .

. . . .

. . . .

. . . .

$$x_n^{(k+1)} = (b_n - a_{n1}x_1^{(k)} - a_{n2}x_2^{(k)} - \ldots\ldots\ldots - a_{n,n-1}x_{n-1}^{(k)}) / a_{nn}$$

las cuales se pueden escribir en una forma más compacta:

$$x_i^{(k+1)} = \frac{b_i - \sum_{\substack{j=1 \\ i \neq j}}^{n} a_{ij}x_j^{(k)}}{a_{ii}} \qquad , i = 1,2,\ldots,n$$

Estas ecuaciones definen al *método de Jacobi o de los desplazamientos simultáneos.*
El proceso termina cuando se cumple alguno de estos criterios de convergencia:

1.- $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| < \epsilon$
2.- $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| / \|\mathbf{x}^{(k+1)}\| < \epsilon$

La tolerancia ϵ es fijada antes de iniciar los cálculos. En ambos criterios, $\|\mathbf{v}\|$, define la norma de un vector **v**. La norma de un vector se calcula mediante cualquiera de esta dos definiciones:

Las normas $\|\mathbf{v}\|_2$ y $\|\mathbf{v}\|_\infty$ del vector $\mathbf{v} = [v_1, v_2, v_3, \ldots, v_n]^T$ están definidas por

$$\|\mathbf{v}\|_2 = \left(\sum_{i=1}^{n} v_i^2 \right)^{1/2} \qquad \text{y} \qquad \|\mathbf{v}\|_\infty = \max_{1 \le i \le n} |v_i|$$

A $\|v\|_2$ se le conoce como *norma euclidiana* del vector **v**.

Ejemplo
Encuentre una solución aproximada del sistema

$$10x_1 - x_2 = 9$$

$$-x_1 + 10x_2 - 2x_3 = 7$$

$$-2x_2 + 10x_3 = 6$$

con el método de Jacobi. Inicie con $\mathbf{x}^{(0)} = [0, 0, 0]^T$ y considere a $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_2 < 1X10^{-4}$ como criterio de convergencia.

Al aplicar el método de Jacobi se generan las ecuaciones:

$x_1^{(k+1)} = (9.00000 + 1.00000x_2^{(k)}) / 10.00000$

$x_2^{(k+1)} = (7.00000 + 1.00000x_1^{(k)} + 2.00000x_3^{(k)}) / 10.00000$

$x_3^{(k+1)} = (6.00000 + 2.00000x_2^{(k)}) / 10.00000$

que aplicadas a la estimación inicial $\mathbf{x}^{(0)}$ permiten calcular la nueva iteración $\mathbf{x}^{(1)}$

$x_1^{(1)} = (9.00000 + 1.00000x_2^{(0)}) / 10.00000$

$x_1^{(1)} = (9.00000 + 1.00000(0.00000)) / 10.00000 = 0.900000$

$x_2^{(1)} = (7.00000 + 1.00000x_1^{(0)} + 2.00000x_3^{(0)}) / 10.00000$

$x_2^{(1)} = (7.00000 + 1.00000(0.00000) + 2.00000(0.00000)) / 0.70000$

$x_3^{(1)} = (6.00000 + 2.00000x_2^{(0)}) / 10.00000$

$x_3^{(1)} = (6.00000 + 2.00000(0.00000)) / 10.00000 = 0.60000$

Con $\mathbf{x}^{(1)} = [0.90000, 0.70000, 0.60000]^T$ se checa la convergencia del método:

$\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| = \sqrt{((0.90000 - 0.00000)^2 + (0.70000 - 0.00000)^2 + (0.60000 - 0.00000)^2)}$

$\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| = 1.28841$

Como la distancia entre las dos últimas iteraciones, $\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|$, no es menor que $1X10^{-4}$ será necesario calcular al menos una iteración más.

$x_1^{(2)} = (9.00000 + 1.00000x_2^{(1)}) / 10.00000$

$x_1^{(2)} = (9.00000 + 1.00000(0.70000)) / 10.00000 = 0.970000$

$x_2^{(2)} = (7.00000 + 1.00000x_1^{(1)} + 2.00000x_3^{(1)}) / 10.00000$

$x_2^{(2)} = (7.00000 + 1.00000(0.90000) + 2.00000(0.60000)) / 0.91000$

$x_3^{(2)} = (6.00000 + 2.00000x_2^{(1)}) / 10.00000$

$x_3^{(2)} = (6.00000 + 2.00000(0.70000)) / 10.00000 = 0.74000$

$\|\mathbf{x}^{(2)} - \mathbf{x}^{(1)}\| = \sqrt{((0.97000 - 0.90000)^2 + (0.91000 - 0.70000)^2 + (0.74000 - 0.60000)^2)}$

$\|\mathbf{x}^{(2)} - \mathbf{x}^{(1)}\| = 0.26192$

Iteración k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$\|x^{(k)} - x^{(k-1)}\|$
0	0.00000	0.00000	0.00000	
1	0.90000	0.70000	0.60000	1.28841
2	0.97000	0.91000	0.74000	0.26192
3	0.99100	0.94500	0.78200	0.05857
4	0.99450	0.95550	0.78900	0.01310
5	0.99555	0.95725	0.79110	0.00293
6	0.99573	0.95778	0.79145	0.00065
7	0.99578	0.95786	0.79156	0.00015
8	0.99579	0.95789	0.79157	0.00003

El siguiente es el pseudocódigo de un procedimiento que resuelve al sistema **Ax** = **b** con el método de Jacobi:

```
proc jacobi(entero n, real A[n, n], real b[n], real xinicial[n], real eps, entero maxiter,
            real &x[n], lógico &convergencia)
!
!  Este procedimiento encuentra una solución aproximada x del sistema lineal
!  Ax = b con el método iterativo de Jacobi.
!  Descripción de parámetros:
!  n              Número de ecuaciones y de incógnitas.
!  A              Matriz de coeficientes del sistema.
!  b              Vector de términos del lado derecho del sistema.
!  xinicial       Vector de estimaciones iniciales.
!  eps            Máximo error permisible. Cuando la norma euclidiana de la diferencia
!                  entre las dos últimas aproximaciones es menor que eps, se considera
!                  haber llegado a solución.
!  maxiter        Número máximo de iteraciones.
!  x              Última iteración calculada. Si en convergencia se devuelve el valor
!                 VERDADERO, entonces x también es el vector solución de Ax=b.
!  convergencia   Bandera cuyo valor indica si hubo convergencia:
!                 convergencia=VERDADERO      se alcanzó la convergencia
!                 convergencia=FALSO          no se alcanzó la convergencia.
!
entero iter, i, j
real suma, norma2
convergencia=VERDADERO
ejecuta  iter = 1, maxiter
{
```

```
        ejecuta i = 1, n                    ! cálculo de la nueva aproximación x
        {
                suma = 0.0
                ejecuta j = 1, n
                        si i≠j entonces suma = suma + A[i, j]*xinicial[j]
                x[i] = (b[i] - suma) / A[n, n]
        }
        suma = 0.0
        ejecuta i =1, n
                suma = suma + (x[i] - xinicial[i])↑2
        norma2 = SQRT(suma)
        si norma2 < eps entonces regresa      ! se checa la convergencia
        ejecuta i = 1, n                    ! preparación de la siguiente iteración
                xinicial[i] = x[i]
}
convergencia = FALSO
regresa
fin
```

Una forma de acelerar la convergencia del método de Jacobi pudiera ser utilizar una estrategia de *desplazamientos sucesivos*: una vez calculada $x_i^{(k+1)}$, se sustituye inmediatamente en la expresión para calcular $x_{i+1}^{(k+1)}$. Las ecuaciones por emplear ahora son:

$$x_1^{(k+1)} = (b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \ldots\ldots\ldots\ldots - a_{1n}x_n^{(k)}) / a_{11}$$

$$x_2^{(k+1)} = (b_2 - a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} - \ldots\ldots\ldots\ldots - a_{2n}x_n^{(k)}) / a_{22}$$

$$x_3^{(k+1)} = (b_3 - a_{31}x_1^{(k+1)} - a_{32}x_2^{(k+1)} - \ldots\ldots\ldots\ldots - a_{3n}x_n^{(k)}) / a_{33}$$

$$\vdots \qquad\qquad \vdots$$

$$x_n^{(k+1)} = (b_n - a_{n1}x_1^{(k+1)} - a_{n2}x_2^{(k+1)} - \ldots\ldots\ldots\ldots - a_{n,n-1}x_{n-1}^{(k+1)}) / a_{nn}$$

las cuales se pueden representar en una forma más condensada:

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}}{a_{nn}} \qquad , i = 1,2,\ldots,n$$

Esta estrategia define al método de *Gauss Seidel o de los desplazamientos sucesivos.*

Ejemplo
Resolver el sistema lineal del ejemplo anterior empleando Gauss Seidel. Considere el mismo

vector inicial y también idéntico criterio de convergencia.

Las ecuaciones del método de Gauss Seidel son:

$x_1^{(k+1)} = (9.00000 + 1.00000x_2^{(k)}) / 10.00000$

$x_2^{(k+1)} = (7.00000 + 1.00000x_1^{(k+1)} + 2.00000x_3^{(k)}) / 10.00000$

$x_3^{(k+1)} = (6.00000 + 2.00000x_2^{(k+1)}) / 10.00000$

Comenzando con $\mathbf{x}^{(0)} = [0, 0, 0]^T$, entonces

$x_1^{(1)} = (9.00000 + 1.00000x_2^{(0)}) / 10.00000$

$x_2^{(1)} = (7.00000 + 1.00000x_1^{(1)} + 2.00000x_3^{(0)}) / 10.00000$

$x_3^{(1)} = (6.00000 + 2.00000x_2^{(1)}) / 10.00000$

$x_1^{(1)} = (9.00000 + 1.00000(0.00000)) / 10.00000 = 0.90000$

$x_2^{(1)} = (7.00000 + 1.00000(0.90000) + 2.00000(0.00000)) / 10.00000 = 0.79000$

$x_3^{(1)} = (6.00000 + 2.00000(0.79000)) / 10.00000 = 0.75800$

Con $\mathbf{x}^{(1)} = [0.90000, 0.79000, 0.75800]^T$ se checa la convergencia del método:

$\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| = \sqrt{((0.90000 - 0.00000)^2 + (0.79000 - 0.00000)^2 + (0.75800 - 0.00000)^2)}$

$\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\| = 1.41727$

Como $\epsilon = 1X10^{-4}$, se requiere de al menos una iteración adicional:

$x_1^{(2)} = (9.00000 + 1.00000x_2^{(1)}) / 10.00000$

$x_2^{(2)} = (7.00000 + 1.00000x_1^{(2)} + 2.00000x_3^{(1)}) / 10.00000$

$x_3^{(2)} = (6.00000 + 2.00000x_2^{(2)}) / 10.00000$

$x_1^{(2)} = (9.00000 + 1.00000(0.79000)) / 10.00000 = 0.97900$

$x_2^{(2)} = (7.00000 + 1.00000(0.97900) + 2.00000(0.75800)) / 10.00000 = 0.94950$

$x_3^{(2)} = (6.00000 + 2.00000(0.94950)) / 10.00000 = 0.78990$

$\|\mathbf{x}^{(2)} - \mathbf{x}^{(1)}\| = \sqrt{((0.97900 - 0.90000)^2 + (0.94950 - 0.79000)^2 + (0.78990 - 0.75800)^2)}$

$\|\mathbf{x}^{(2)} - \mathbf{x}^{(1)}\| = 0.18083$

Iteración k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$\|x^{(k)} - x^{(k-1)}\|$
0	0.00000	0.00000	0.00000	
1	0.90000	0.79000	0.75800	1.41727
2	0.97900	0.94950	0.78990	0.18083
3	0.99495	0.95748	0.79150	0.01790
4	0.99575	0.95787	0.79157	0.00090
5	0.99579	0.95789	0.79158	0.00004

En general, Gauss Seidel es más rápido que Jacobi.
Ahora, un procedimiento para resolver un sistema **Ax** = b con el método de Gauss Seidel:

```
proc gauss_seidel(entero n, real A[n, n], real b[n], real xinicial[n], real eps, entero maxiter,
                  real &x[n], lógico &convergencia)
!
! Este procedimiento encuentra una solución aproximada x del sistema lineal
! Ax = b con el método iterativo de Gauss Seidel.
! Descripción de parámetros:
! n              Número de ecuaciones y de incógnitas.
! A              Matriz de coeficientes del sistema.
! b              Vector de términos del lado derecho del sistema.
! xinicial       Vector de estimaciones iniciales.
! eps            Máximo error permisible. Cuando la norma euclidiana de la diferencia
!                entre las dos últimas aproximaciones es menor que eps, se considera
!                haber llegado a solución.
! maxiter        Número máximo de iteraciones.
! x              Última iteración calculada. Si en convergencia se devuelve el valor
!                VERDADERO, entonces x también es el vector solución de Ax=b.
! convergencia   Bandera cuyo valor indica si hubo convergencia:
!                convergencia=VERDADERO      se alcanzó la convergencia
!                convergencia=FALSO          no se alcanzó la convergencia.
!
entero iter, i, j
real suma, norma2
convergencia=VERDADERO
ejecuta iter = 1, maxiter
{
        ejecuta i = 1, n                ! cálculo de la nueva aproximación x
        {
                suma = 0.0
```

```
            ejecuta j = 1, i - 1
                    suma = suma + A[i, j]*x[j]
            ejecuta j = i + 1, n
                    suma = suma + A[i, j]*xinicial[j]
            x[i] = (b[i] - suma) / A[n, n]
        }
        suma = 0.0
        ejecuta i =1, n
                suma = suma + (x[i] - xinicial[i])↑2
        norma2 = SQRT(suma)
        si norma2 < eps entonces regresa      ! se checa la convergencia
        ejecuta i = 1, n                      ! preparación de la siguiente iteración
                xinicial[i] = x[i]
}
convergencia = FALSO
regresa
fin
```

La condición suficiente pero no necesaria que garantiza la convergencia tanto de Jacobi como de Gauss Seidel para cualquier vector inicial $\mathbf{x}^{(0)}$ es que la matriz de coeficientes del sistema sea *estrictamente diagonal dominante*. A continuación, la definición de una matriz estrictamente diagonal dominante.

Una matriz A de nxn es estrictamente diagonal dominante si y sólo si

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \qquad ,i = 1, 2, \ldots, n$$

En base a esta definición, la matriz de coeficientes de los dos últimos ejemplos es estrictamente diagonal dominante ya que

$|10| > |-1| + |0|$

$|10| > |-1| + |-2|$

$|10| > |0| + |-2|$

Esto asegura la convergencia de ambos métodos para cualquier estimación inicial $\mathbf{x}^{(0)}$.

3. Solución de sistemas de ecuaciones no lineales

Se trata de encontrar un vector $\underline{\mathbf{x}} = [\underline{x}_1, \underline{x}_2, \underline{x}_3, ..., \underline{x}_n]^T = 0$ que satisfaga las igualdades

$f_1(x_1, x_2, x_3, ..., x_n) = 0$

$f_2(x_1, x_2, x_3, ..., x_n) = 0$

$f_3(x_1, x_2, x_3, ..., x_n) = 0$

. . .

. . .

. . .

$f_n(x_1, x_2, x_3, ..., x_n) = 0$

Estas ecuaciones también se pueden representar en una forma más compacta:

$f_i(\mathbf{x}) = 0$, para i = 1, 2, 3, ..., n

3.1 Método de punto fijo multivariable

El sistema $f_i(\mathbf{x}) = 0$, i = 1,2,3, ..., n es transformado en el conjunto de ecuaciones

$x_1 = g_1(\mathbf{x})$

$x_2 = g_2(\mathbf{x})$

$x_3 = g_3(\mathbf{x})$

. .

. .

. .

$x_n = g_n(\mathbf{x})$

mediante la aplicación de operaciones algebraicamente válidas. A cada una de esta ecuaciones se les aplica el método iterativo de punto fijo:

$x_1^{(k+1)} = g_1(\mathbf{x}^{(k)})$

$x_2^{(k+1)} = g_2(\mathbf{x}^{(k)})$

$x_3^{(k+1)} = g_3(\mathbf{x}^{(k)})$

. .

. .

. .

$x_n^{(k+1)} = g_n(\mathbf{x}^{(k)})$

FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA

CURSOS ABIERTOS

DIPLOMADO DE MATEMÁTICAS APLICADAS A LA INGENIERÍA

"MÉTODOS NUMÉRICOS"

MÓDULO II

TEMA: APUNTES GENERALES

CA-484

EXPOSITOR: M. EN C. ABEL VALDÉS RAMÍREZ

12 DE NOVIEMBRE AL 3 DE DICIEMBRE DE 2005-11-18 PALACIO DE MINERÍA

Se comienza con una estimación inicial $\mathbf{x}^{(0)}$, la cual es sustituida en las ecuaciones $g_1, g_2, g_3, ..., g_n$ resultando una nueva aproximación $\mathbf{x}^{(1)}$. Estas funciones son evaluadas en $\mathbf{x}^{(1)}$ para generar $\mathbf{x}^{(2)}$. Este procedimiento es repetido para calcular las aproximaciones $x^{(3)}, x^{(4)}, x^{(5)}, ...$. En el momento en que se cumpla alguno de los criterios de convergencia usuales, se termina el proceso iterativo.

Ejemplo
Encuentre una solución aproximada del sistema

$$f_1(x_1, x_2) = x_1^2 - 10x_1 + x_2^2 + 8 = 0$$

$$f_2(x_1, x_2) = x_1 x_2^2 + x_1 - 10x_2 + 8 = 0$$

con el método de punto fijo. Inicie con $\mathbf{x}^{(0)} = [0, 0]^T$ y considere el criterio de convergencia $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_2 < 1X10^{-3}$.

Una posible manera de aislar las variables x_1 y x_2 sería:

$$x_1 = (8 + x_1^2 + x_2^2) / 10$$

$$x_2 = (8 + x_1 x_2^2 + x_1) / 10$$

Aplicando la iteración de punto fijo a estos despejes:

$$x_1^{(k+1)} = (8 + [x_1^{(k)}]^2 + [x_2^{(k)}]^{(2)}) / 10$$

$$x_2^{(k+1)} = (8 + x_1^{(k)} [x_2^{(k)}]^2 + x_1^{(k)}) / 10$$

Con $\mathbf{x}^{(0)} = [0, 0]^T$ y las ecuaciones anteriores , entonces $\mathbf{x}^{(1)}$ es:

$$x_1^{(1)} = (8 + [x_1^{(0)}]^2 + [x_2^{(0)}]^2) / 10 = (8.00000 + 0.00000^2 + 0.00000^2) / 10.00000 = 0.80000$$

$$x_2^{(1)} = (8 + x_1^{(0)} [x_2^{(k)}]^2 + x_1^{(0)}) / 10 = (8.00000 + (0.00000)(0.00000^2) + 0.00000) / 10.00000$$

$$x_2^{(1)} = 0.80000$$

$$\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|_2 = \sqrt{((0.80000 - 0.00000)^2 + (0.80000 - 0.00000)^2)} = 1.13137$$

Como la distancia entre $\mathbf{x}^{(1)}$ y $\mathbf{x}^{(0)}$ es mayor que $1X10^{-3}$, se requiere por lo menos de una iteración más:

$$x_1^{(2)} = (8 + [x_1^{(1)}]^2 + [x_2^{(1)}]^2) / 10 = (8.00000 + 0.80000^2 + 0.80000^2) / 10.00000 = 0.92800$$

$$x_2^{(2)} = (8 + x_1^{(1)} [x_2^{(1)}]^2 + x_1^{(1)}) / 10 = (8.00000 + (0.80000)(0.80000^2) + 0.80000) / 10.00000$$

$$x_2^{(2)} = 0.93120$$

$\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|_2 = \sqrt{((0.80000 - 0.00000)^2 + (0.80000 - 0.00000)^2)} = 1.13137$

Iteración k	$x_1^{(k)}$	$x_2^{(k)}$	$\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_2$
0	0.00000	0.00000	
1	0.80000	0.80000	1.13137
2	0.92800	0.93120	0.18330
3	0.97283	0.97327	0.06148
4	0.98937	0.98944	0.02312
5	0.99578	0.99579	0.00903
6	0.99832	0.99832	0.00358
7	0.99933	0.99933	0.00143
8	0.99973	0.99973	0.00057

El método iterativo de punto fijo multivariable converge, cuando lo hace, de una forma lenta. Para tratar de acelerar su convergencia, frecuentemente se intenta aplicar un estrategia de desplazamientos sucesivos, similar a la aplicada en el método de Gauss Seidel.

Ejemplo
Aplicar desplazamientos sucesivos al ejemplo anterior.

En este caso, las ecuaciones del método iterativo de punto fijo quedarían:

$x_1^{(k+1)} = (8 + [x_1^{(k)}]^2 + [x_2^{(k)}]^2) / 10$

$x_2^{(k+1)} = (8 + x_1^{(k+1)} [x_2^{(k)}]^2 + x_1^{(k+1)}) / 10$

En el cálculo de $x_2^{(k+1)}$ se sustituye inmediatamente el valor recién calculado $x_1^{(k+1)}$. Con $\mathbf{x}^{(0)} = [0, 0]^T$, entonces $\mathbf{x}^{(1)}$ es:

$x_1^{(1)} = (8 + [x_1^{(0)}]^2 + [x_2^{(k)}]^2) / 10 = (8.00000 + 0.00000^2 + 0.00000^2) / 10.00000 = 0.80000$

$x_2^{(1)} = (8 + x_1^{(1)} [x_2^{(0)}]^2 + x_1^{(1)}) / 10 = (8.00000 + (0.80000)(0.00000^2) + 0.80000) / 10.00000$

$x_2^{(1)} = 0.88000$

$\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|_2 = \sqrt{((0.80000 - 0.00000)^2 + (0.88000 - 0.00000)^2)} = 1.18929$

Ya que la distancia entre $\mathbf{x}^{(1)}$ y $\mathbf{x}^{(0)}$ es mayor que $1X10^{-3}$, se requiere al menos de una iteración

más:

$x_1^{(2)} = (8 + [x_1^{(1)}]^2 + [x_2^{(1)}]^2) / 10 = (8.00000 + 0.80000^2 + 0.88000^2) / 10.00000 = 0.94144$

$x_2^{(2)} = (8 + x_1^{(2)} [x_2^{(1)}]^2 + x_1^{(2)}) / 10 = (8.00000 + (0.94144)(0.88000^2) + 0.94144) / 10.00000$

$x_2^{(2)} = 0.96705$

$\|\mathbf{x}^{(2)} - \mathbf{x}^{(1)}\|_2 = \sqrt{((0.94144 - 0.80000)^2 + (0.96705 - 0.88000)^2)} = 0.16608$

Iteración k	$x_1^{(k)}$	$x_2^{(k)}$	$\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_2$
0	0.00000	0.00000	
1	0.80000	0.88000	1.18929
2	0.94144	0.96705	0.16608
3	0.98215	0.99006	0.04676
4	0.99448	0.99693	0.01412
5	0.99829	0.99905	0.00435
6	0.99947	0.99970	0.00135
7	0.99983	0.99991	0.00042

En general; el método de punto fijo multivariable con desplazamientos sucesivos, conocido también como *Gauss Seidel no lineal*, converge más rápido que el mismo método pero con desplazamientos simultáneos (*Jacobi no lineal*).
Ahora, el pseudocódigo de un procedimiento que resuelve en forma aproximada un sistema de ecuaciones no lineales descrito como **x** = **g**(**x**):

```
proc punto_fijo_n(entero n, proc g, xinicial[n], real eps, entero maxiter, real &x[n],
                  lógico &convergencia)
!
! Este procedimiento encuentra una solución aproximada x del sistema no lineal
! descrito como x = g(x), con el método iterativo de punto fijo multivariable.
! Descripción de parámetros:
! n              Número de ecuaciones y de incógnitas.
! g              Procedimiento que calcula una nueva aproximación x = g(xinicial, x)
!                proc g(entero n, real xinicial[n], real &x[n])
!                n          Número de ecuaciones y de incógnitas.
!                xinicial   Vector de incógnitas en la iteración inicial.
!                x          Vector de incógnitas en la nueva iteración.
! xinicial       Vector de estimaciones iniciales.
```

```
!  eps              Máximo error permisible. Cuando la norma euclidiana de la diferencia
!                    entre las dos últimas aproximaciones es menor que eps, se considera
!                    haber llegado a solución.
!  maxiter          Número máximo de iteraciones.
!  x                Última iteración calculada. Si en convergencia se devuelve el valor
!                    VERDADERO, entonces x también es el vector solución de Ax=b.
!  convergencia     Bandera cuyo valor indica si hubo convergencia:
!                    convergencia=VERDADERO      se alcanzó la convergencia
!                    convergencia=FALSO          no se alcanzó la convergencia.
!
entero iter, i
real suma, norma2
convergencia=VERDADERO
ejecuta  iter = 1, maxiter
{
        llama g(n, xinicial, x)          ! cálculo de la nueva aproximación x
        suma = 0.0
        ejecuta i =1, n
                suma = suma + (x[i] - xinicial[i])↑2
        norma2 = SQRT(suma)
        si norma2 < eps entonces regresa      ! se checa la convergencia
        ejecuta i = 1, n                  ! preparación de la siguiente iteración
                xinicial[i] = x[i]
}
convergencia = FALSO
regresa
fin
```

3.2 Método de Newton Raphson multivariable

Para deducir las ecuaciones de Newton Raphson multivariable, supongáse el caso particular de dos variables y dos ecuaciones (n=2):

$f_1(x_1, x_2) = 0$

$f_2(x_1, x_2) = 0$

Si todas las n-ésimas derivadas parciales de $f(x_1, x_2)$ son continuas en una región cerrada y si las (n+1)-ésimas derivadas parciales existen en la región abierta, se tiene:

$$f(x_1 + h, x_2 + k) = f(x_1, x_2) + \left(h\frac{\partial}{\partial x_1} + k\frac{\partial}{\partial x_2}\right) f(x_1, x_2) + \frac{1}{2!}\left(h\frac{\partial}{\partial x_1} + k\frac{\partial}{\partial x_2}\right)^2 f(x_1, x_2) + \ldots.$$

$$\ldots. + \frac{1}{n!}\left(h\frac{\partial}{\partial x_1} + k\frac{\partial}{\partial x_2}\right)^n f(x_1, x_2) + R_n$$

donde R_n es:

$$R_n = \frac{1}{(n+1)!}\left(h\frac{\partial}{\partial x_1} + k\frac{\partial}{\partial x_2}\right)^{n+1} f(x_1 + \theta h, x_2 + \theta k) \quad , 0 < \theta < 1$$

aplicando este resultado tanto a f_1 como a f_2 en el punto base $(x_1^{(k)}, x_2^{(k)})$ y con incrementos $h_1 = x_1^{(k+1)} - x_1^{(k)}$ en la dirección x_1 y $h_2 = x_2^{(k+1)} - x_2^{(k)}$ en la dirección x_2:

$$f_1(x_1^{(k+1)}, x_2^{(k+1)}) = f_1(x_1^{(k)}, x_2^{(k)}) + (x_1^{(k+1)} - x_1^{(k)})\frac{\partial f_1}{\partial x_1} + (x_2^{(k+1)} - x_2^{(k)})\frac{\partial f_1}{\partial x_2} +$$

$$+\frac{1}{2!}\left[\begin{array}{l}(x_1^{(k+1)}, x_2^{(k+1)})^2 \dfrac{\partial^2 f_1}{\partial x_1^2} + 2(x_1^{(k+1)}, x_2^{(k+1)})(x_2^{(k+1)} - x_2^{(k)})\dfrac{\partial^2 f_1}{\partial x_2 x_1} + \\ (x_2^{(k+1)} - x_2^{(k)})^2 \dfrac{\partial^2 f_1}{\partial x_2^2}\end{array}\right] + \ldots.$$

$$f_2(x_1^{(k+1)}, x_2^{(k+1)}) = f_2(x_1^{(k)}, x_2^{(k)}) + (x_1^{(k+1)} - x_1^{(k)})\frac{\partial f_2}{\partial x_1} + (x_2^{(k+1)} - x_2^{(k)})\frac{\partial f_2}{\partial x_2} +$$

$$+\frac{1}{2!}\left[\begin{array}{l}(x_1^{(k+1)}, x_2^{(k+1)})^2 \dfrac{\partial^2 f_2}{\partial x_1^2} + 2(x_1^{(k+1)}, x_2^{(k+1)})(x_2^{(k+1)} - x_2^{(k)})\dfrac{\partial^2 f_2}{\partial x_2 x_1} + \\ (x_2^{(k+1)} - x_2^{(k)})^2 \dfrac{\partial^2 f_2}{\partial x_2^2}\end{array}\right] + \ldots.$$

donde las derivadas parciales son evaluadas es el punto base $\mathbf{x}^{(k)}$.
Si suponemos que $\mathbf{x}^{(k+1)} \approx \underline{\mathbf{x}}$ es razonable también suponer que $f_1(x_1^{(k+1)}, x_2^{(k)}) \approx 0$ y $f_2(x_1^{(k+1)}, x_2^{(k)}) \approx 0$. Si además $\mathbf{x}^{(k+1)} \approx \mathbf{x}^{(k)}$, entonces $(x_1^{(k+1)} - x_1^{(k)})^j \approx 0$ y $(x_2^{(k+1)} - x_2^{(k)})^j \approx 0$ para j = 2,3,4, ...
Después de estas suposiciones las ecuaciones anteriores se simplifican considerablemente:

$$0 \approx f_1(x_1^{(k)}, x_2^{(k)}) + (x_1^{(k+1)} - x_1^{(k)})\frac{\partial f_1}{\partial x_1} + (x_2^{(k+1)} - x_2^{(k)})\frac{\partial f_1}{\partial x_2}$$

$$0 \approx f_2(x_1^{(k)}, x_2^{(k)}) + (x_1^{(k+1)} - x_1^{(k)})\frac{\partial f_2}{\partial x_1} + (x_2^{(k+1)} - x_2^{(k)})\frac{\partial f_2}{\partial x_2}$$

Como $x_1^{(k+1)} - x_1^{(k)} = h_1$ y $x_2^{(k+1)} - x_2^{(k)} = h_2$, entonces

$$h_1 \frac{\partial f_1}{\partial x_1} + h_2 \frac{\partial f_1}{\partial x_2} = -f_1(x_1^{(k)}, x_2^{(k)})$$

$$h_1 \frac{\partial f_2}{\partial x_1} + h_2 \frac{\partial f_2}{\partial x_2} = -f_2(x_1^{(k)}, x_2^{(k)})$$

Esta últimas ecuaciones definen un sistema de dos ecuaciones lineales con dos incógnitas (h_1 y h_2). Resuelto este, la nueva aproximación $\mathbf{x}^{(k+1)}$ se calcula mediante:

$$x_1^{(k+1)} = x_1^{(k)} + h_1$$

$$x_2^{(k+1)} = x_2^{(k)} + h_2$$

Las fórmulas obtenidas arriba se pueden generalizar fácilmente para el caso de n variables:

$$\mathbf{J}(\mathbf{x}^{(k)})\ \mathbf{h} = -\ \mathbf{f}(\mathbf{x}^{(k)})$$

Esta última ecuación es la del método de Newton Raphson multivariable. En ella, **J** es la *matriz Jacobiana del sistema de ecuaciones* y se define por:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \cdots\cdots\cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \cdots\cdots\cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \frac{\partial f_n}{\partial x_3} & \cdots\cdots\cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

mientras que $\mathbf{h} = [h_1, h_2, h_3, ..., h_n]^T$ y $\mathbf{f} = [f_1, f_2, f_3, ..., f_n]^T$.
Por lo tanto, en cada iteración de Newton Raphson multivariable será necesario resolver un sistema lineal de n ecuaciones y n incógnitas. Si la matriz jacobiana es singular, entonces el sistema $\mathbf{J}(\mathbf{x}^{(k)})\ \mathbf{h} = -\ \mathbf{f}(\mathbf{x}^{(k)})$ no tiene solución única y por lo tanto Newton Raphson falla en la búsqueda de una raíz $\underline{\mathbf{x}}$.

Ejemplo
Encuentre una raíz aproximada del sistema

$$f_1(x_1, x_2) = x_1^2 + x_2^2 - 4 = 0$$

$$f_2(x_1, x_2) = x_1\, x_2 - 1 = 0$$

mediante la aplicación de Newton Raphson multivariable con $\mathbf{x}^{(0)} = [2, 0]^T$. Considere el criterio

de convergencia $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_2 < 1X10^{-4}$.

$\mathbf{f}(\mathbf{x}^{(0)}) = [f_1(\mathbf{x}^{(0)}), f_2(\mathbf{x}^{(0)})]^T = [[x_1^{(0)}]^2 + [x_2^{(0)}]^2 - 4, x_1^{(0)} x_2^{(0)} - 1]^T$

$\mathbf{f}(\mathbf{x}^{(0)}) = [2.00000^2 + 0.00000^2 - 4.00000, 2.00000(0.00000) - 1.00000]^T = [0.00000, -1.00000]^T$

$$\mathbf{J}(\mathbf{x}^{(0)}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}^{(0)}) & \frac{\partial f_1}{\partial x_2}(\mathbf{x}^{(0)}) \\ \frac{\partial f_2}{\partial x_1}(\mathbf{x}^{(0)}) & \frac{\partial f_2}{\partial x_2}(\mathbf{x}^{(0)}) \end{bmatrix} = \begin{bmatrix} 2x_1^{(0)} & 2x_2^{(0)} \\ x_2^{(0)} & x_1^{(0)} \end{bmatrix} = \begin{bmatrix} 4.00000 & 0.00000 \\ 0.00000 & 2.00000 \end{bmatrix}$$

Por lo tanto, el sistema lineal $\mathbf{J}(\mathbf{x}^{(0)})\,\mathbf{h} = -\mathbf{f}(\mathbf{x}^{(0)})$ por resolver en esta iteración es:

$4.00000h_1 + 0.00000h_2 = 0.00000$

$0.00000h_1 + 2.00000h_2 = 1.00000$

cuya solución es $\mathbf{h} = [0.00000, 0.50000]^T$. La nueva aproximación $\mathbf{x}^{(1)}$ se obtiene con:

$x_1^{(1)} = x_1^{(0)} + h_1 = 2.00000 + 0.00000 = 2.00000$

$x_2^{(1)} = x_2^{(0)} + h_2 = 0.00000 + 0.50000 = 0.50000$

$\|\mathbf{x}^{(1)} - \mathbf{x}^{(0)}\|_2 = \|\mathbf{h}\|_2 = \sqrt{(0.00000^2 + 0.50000^2)} = 0.50000$

Dado que todavía no se cumple con el criterio de convergencia, será necesario calcular al menos una iteración adicional.

Iteración k	$x_1^{(k)}$	$x_2^{k)}$	$\|x^{(k)} - x^{(k-1)}\|_2$
0	2.00000	0.00000	
1	2.00000	0.50000	0.50000
2	1.93333	0.51667	0.06872
3	1.93185	0.51764	0.00177
4	1.93185	0.51764	$1.50229X10^{-6}$

En el caso multivariable, Newton Raphson también presenta convergencia cuadrática, es decir, su orden de convergencia es 2. Sin embargo, se requiere partir de una estimación inicial cercana a la raíz, para que el método funcione adecuadamente.
El siguiente es el pseudocódigo de un procedimiento que resuelve en forma aproximada un

sistema de n ecuaciones no lineales con Newton Raphson multivariable:

```
proc newton_n(entero n, proc funs, proc jacobiana, xinicial[n], real eps, entero maxiter, real
&x[n],
                    lógico &convergencia)
!
!  Este procedimiento encuentra una solución aproximada x del sistema no lineal
!  f (x) = 0, con el método de Newton Raphson  multivariable.
!  Descripción de parámetros:
!  n              Número de ecuaciones y de incógnitas.
!  funs           Procedimiento que calcula las funciones f (x) = 0
!                 proc funs(entero n, real x, real &f[n])
!                 n          Número de ecuaciones y de incógnitas.
!                 x          Vector de variables x.
!                 f          Vector de funciones f.
!  jacobiana      Procedimiento que calcula los elementos de la matriz jacobiana del
!                 sistema de ecuaciones
!                 proc jacobiana(entero n, real x, real &J[n, n])
!                 n          Número de ecuaciones y de incógnitas.
!                 x          Vector de variables x.
!                 J          Matriz jacobiana del sistema. J[i, j] = df[i] / dx[j] evaluada
!                            en x.
!  xinicial       Vector de estimaciones iniciales.
!  eps            Máximo error permisible. Cuando la norma euclidiana de la diferencia
!                  entre las dos últimas aproximaciones es menor que eps, se considera
!                  haber llegado a solución.
!  maxiter        Número máximo de iteraciones.
!  x              Última iteración calculada. Si en convergencia se devuelve el valor
!                 VERDADERO, entonces x también es el vector solución de Ax=b.
!  convergencia   Bandera cuyo valor indica si hubo convergencia:
!                 convergencia=VERDADERO      se alcanzó la convergencia
!                 convergencia=FALSO          no se alcanzó la convergencia.
!
entero iter, i, j
real f[n], J[n, n], A[n, n+1], h[n], suma, norma2
lógico solucion_unica
convergencia=VERDADERO
ejecuta  iter = 1, maxiter
{
        llama funs(n, xinicial, f)        ! cálculo de las funciones f evaluadas en xinicial
        llama jacobiana(n, xinicial, J)   ! cálculo de la matriz jacobiana J evaluada en xinicial
        ejecuta i = 1, n                  ! formación de la matriz aumentada A del sistema lineal
        {
                ejecuta j = 1, n
                        A[i, j] = jacobiana[i, j]
                A[i, n+1] = -f[i]
        }
```

```
        llama eliminacion_gauss(n, A, h, solucion_unica)    ! solución del sistema lineal
        si no solucion_unica entonces
        {
                escribe 'El sistema lineal en la iteración', iter, ' no tiene solución única'
                convergencia = FALSO
                regresa
        }
        ejecuta i = 1, n
                x[i] = xinicial[i] + h[i]          ! cálculo de la nueva aproximación x
        suma = 0.0
        ejecuta i =1, n
                suma = suma + h[i]↑2
        norma2 = SQRT(suma)
        si norma2 < eps entonces regresa      ! se checa la convergencia
        ejecuta i = 1, n                      ! preparación de la siguiente iteración
                xinicial[i] = x[i]
}
convergencia = FALSO
regresa
fin
```

El esfuerzo requerido para determinar analíticamente las n^2 derivadas parciales de la matriz jacobiana puede llegar a ser considerable y frustrante, particularmente para n grande y/o ecuaciones complejas. En ese caso, se aconseja calcular las derivadas en forma aproximada empleando *fórmulas de diferencias finitas*. Una de la más usadas es:

$$\frac{\partial f_i}{\partial x_j}(\mathbf{x}^{(k)}) \approx \frac{f_i(\mathbf{x}^{(k)} + \mathbf{e}_j h) - f_i(\mathbf{x}^{(k)})}{h}$$

donde h es una cantidad pequeña en valor absoluto y $\mathbf{e}_j$ es un vector cuyo único elemento distinto de cero es un 1 de la j-ésima coordenada.

4. Aplicaciones

1.- Si se compra una pieza de un equipo que cuesta $20,000 al contado y en pagos de $4,000 al año durante 6 años, ¿qué tasa de interés i se está pagando ? La fórmula que relaciona el valor presente P, los pagos anuales A, el números de años n y la tasa de interés es

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

2.- La fórmula que define la fuerza por unidad de área, P/A, que causa un máximo esfuerzo σ_m en una columna que tiene una relación de esbeltez L_e/r es:

$$\frac{P}{A} = \frac{\sigma_m}{1 + (ec/r^2)\sec[\sqrt{P/(EA)}(L_e/r)]}$$

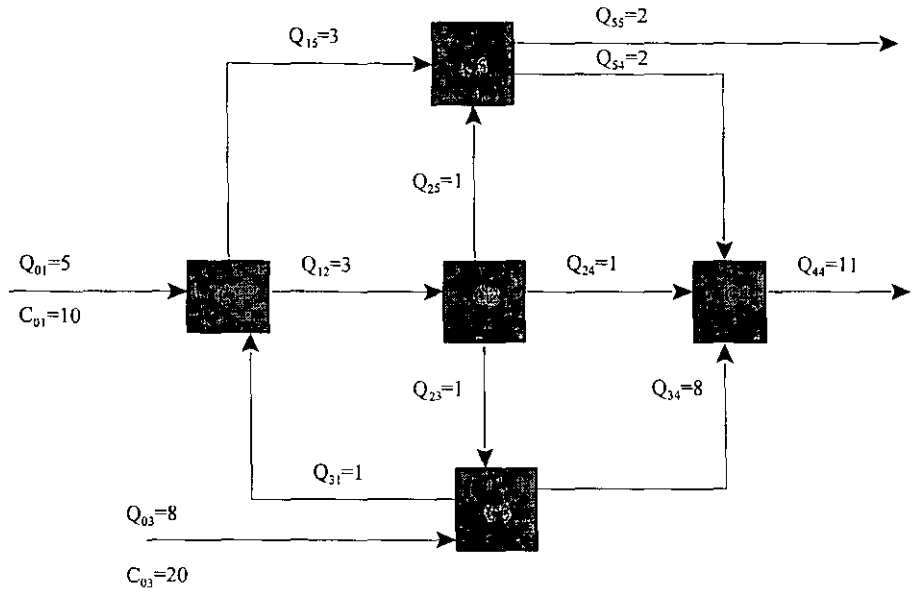
Si E = 200000 kPa, $ec/r^2 = 0.2$ y $\sigma_m = 250$ kPa, calcule P/A para $L_e/r = 100$.

3.- La siguiente ecuación sirve para calcular el nivel de oxígeno en un río aguas abajo desde una descarga de agua residuales:

$$c = 10 - 20(e^{-0.2x} - e^{-0.75x})$$

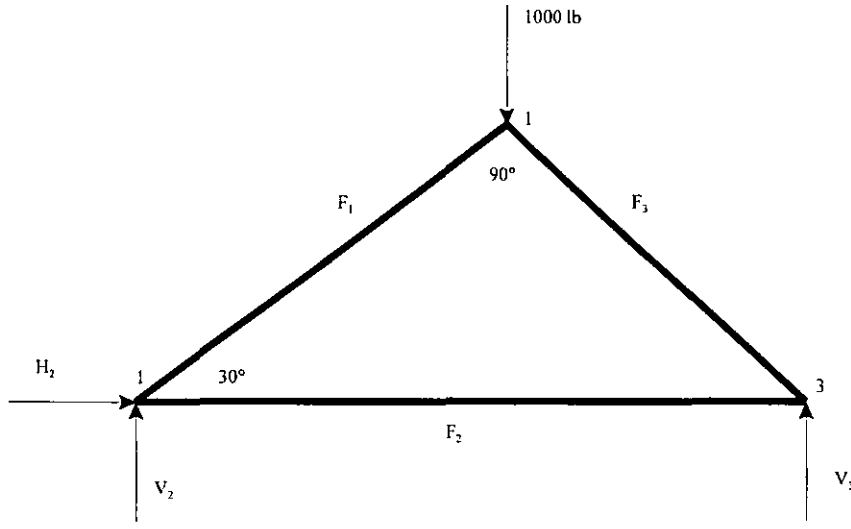
donde x es la distancia aguas abajo en kilómetros. Determine la distancia aguas abajo donde la lectura del nivel de oxígeno es 5.

4.- Considere 5 reactores interconectados. Los reactores son del tipo tanque agitado y en ellos la concentración C es uniforme. En el diagrama adjunto; los flujos volumétricos Q de cada corriente están dados en m^3/min y la concentración C de cierta sustancia es reportada en mg/m^3. Defina y resuelva el conjunto de ecuaciones que resume el balance de materia en estado estacionario para esa sustancia.



5.- La siguiente figura muestra una estructura estáticamente determinada. Las fuerzas (F_1, F_2 y F_3) representan ya sea la tensión o la compresión sobre los componentes de la armadura. Las reacciones externas (H_2, V_2 y V_3) son fuerzas que caracterizan cómo interactúa dicha estructura con la superficie del soporte. El apoyo fijo en el nodo 2 puede transmitir fuerzas horizontales y verticales a la superficie, mientras que el apoyo móvil en el nodo 3 transmite sólo fuerzas verticales. Se observa que el efecto de la carga externa de 1000 lb se distribuye entre los componentes de la armadura.

Plantee un sistema de ecuaciones cuya solución permita encontrar los valores de F_1, F_2, F_3, H_2, V_2 y V_3.



6.- Considere las reacciones simultáneas

$CH_4 + H_2O \equiv\equiv\equiv CO + 3H_2$ $\quad K_1 = 0.574$ @ 600 °C

$CO + H_2O \equiv\equiv\equiv CO_2 + H_2$ $\quad K_2 = 2.21$ @ 600 °C

donde K_1 y K_2 son las constantes de equilibrio químico. Las reacciones se llevan a cabo a 600 °C y la presión P= 1 atm. La carga inicial es 5 moles de H_2O y 1 mol de CH_4. Si se llega al equilibrio, entonces el balance de materia está dado por el sistema de ecuaciones:

$$0.574 = \frac{(\varepsilon_1 - \varepsilon_2)(3\varepsilon_1 + \varepsilon_2)^3}{(1-\varepsilon_1)(5-\varepsilon_1-\varepsilon_2)} \frac{P}{(6+2\varepsilon_1)^2}$$

$$2.21 = \frac{\varepsilon_2(3\varepsilon_1 + \varepsilon_2)}{(\varepsilon_1 - \varepsilon_2)(5-\varepsilon_1-\varepsilon_2)}$$

donde ϵ_1 es la coordenada de la primera reacción y ϵ_2 la coordenada de la segunda reacción. Determine los valores de ambas coordenadas.

7.- En un termo, el compartimiento interior está separado del compartimiento intermedio por vacío. Alrededor del termo hay una última capa, que está separado de la capa intermedio por una delgada capa de aire. La parte exterior de la última capa está en contacto con el medio ambiente. La transferencia de calor del compartimiento interior a la siguiente capa, q_1, es sólo por radiación (ya que este espacio está vacío). La transferencia de calor entre la capa intermedia y la capa final, q_2, es por convección en un espacio reducido. La transferencia de calor de la última capa al medio ambiente, q_3, es por convección natural. El flujo de calor desde cada reagión del termo debe ser igual, es decir, $q_1 = q_2 = q_3$.Encuentre las temperaturas T_1 y T_2 en estado estacionario. Si T_0 es 500 °C y T_3, es 25 °C.

$q_1 = 10^{-9} [(T_0 + 273)^4 - (T_1 + 273)^4]$

$q_2 = 4(T_1 - T_2)$

$q_3 = 1.3(T_2 - T_3)^{4/3}$