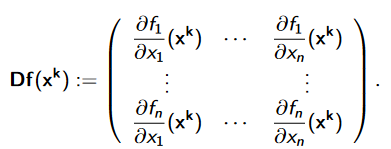
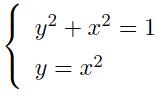
**Resolución de sistemas de varias ecuaciones**

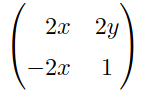
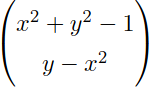
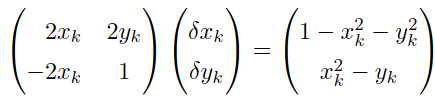
**Método de Newton**

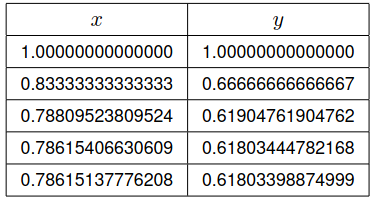
* Generalización do método de Newton-Raphson (ver tema 1) para sistemas de n ecuaciones.
* **Caso n=1 (Newton-Raphson):**
  + Consideramos un itinerante inicial x0
  + Construímos xk+1 tal que xk+1 = xk  -
* **Caso n=2 (Newton):**
  + Aplicamos o desenvolvemento de Taylor para funcións de varias variables a f, nunha contorna do punto xk. Obtemos:
    - 0 = f(a) = f(xk) + Df(xk)(a-xk) + O(||a-xk||2), sendo **a** unha raíz de f.
  + Onde **Df(xk)** é a **matriz Jacobiana[[1]](#footnote-0):** e O(||a-xk||2) pódese desprezar.
  + **0 ≃ f(x(k)) + Df(x(k))(a-x(k))** é a ecuación vectorial resultante. Despexamos a raíz **a**.
  + O **método de Newton** consiste en, a partir da aproximación da solución x(k), calcular unha nova aproximación x(k+1):



* + Para evitar ter que invertir a matriz Df(x(k)), realizase o seguinte algoritmo:
    - Para k=0,1,2..., sexa **δx(k)**:= x(k+1)-x(k)
    - Resolver Df(x(k))δx(k) = -f(x(k))
    - x(k+1):=x(k)+δx(k)
  + O algoritmo pode repetirse ata que ||xk+1 - xk|| sexa menor que o erro ε que permitimos.

**Ejemplo** método de Newton

* Resolver el sistema siguiente con error menor que tol = 10-5
* f(x,y) = , Df(x,y) =
* Sexan os datos iniciais x0 e y0 (1,1)
* Empregamos o algoritmo descrito previamente. Para k=0,1,2…:
  + Resolvemos o sistema 
    - Datos iniciais: x0=1, y0=1.
      * 2δx(k) + 2δy(k) = -1, -2δx(k) + δy(k) = 0
      * δx(k) = -⅙, δy(k) = -⅓
      * Calculamos x(k+1):=x(k)+δx(k) = ⅚
      * Calculamos y(k+1):=y(k)+δy(k) = ⅔
    - Ahora, x1=⅚ e y2=⅔. Calculamos os novos valores de δx(k) e δy(k).
  + Repetimos ata que 



**Método do descenso rápido**

* Sexa o problema de atopar o punto a onde f(a)=0, sendo f: R2→R2.
* Podemos reformulalo como atopar os puntos onde a función G acada un mínimo absoluto (que será 0)[[2]](#footnote-1), sendo G o cadrado da lonxitude do vector f(x1,x2):

[[3]](#footnote-2)

* Para atopar o mínimo de G, debemos seguir a dirección **-▽G(x1,x2)**.
  + 
  + 
* Denominamos **X0** o noso itinerante inicial, e **Z** a dirección de máximo decrecemento.
  + Consideramos a recta **X1 := X0 -a\*z**. Debemos calcular cal é o valor de **a**, que indica ‘canto’ nos temos que desplazar para atopar o mínimo.
  + A función que debemos minimizar será **h(a)**:



* + - Partimos de a1=0. O valor g1 obtido será G(X0).
    - Logo, definimos a3=1. Obtemos un valor distinto de g3.
      * Se g3>g1, indica que nos pasamos do mínimo. Definimos un novo a2=a3/2 ata que g2<g1
      * Se g3<g1, este valor de a3 é máis próximo que o previo e consideramolo unha aproximación correcta.
      * Repetímolo ata que o valor de a3 sexa máis pequeno que a metade da tolerancia do código.
    - Unha vez posuímos a1, a2 e a3, definimos o **polinomio de interpolación** de grao 2 da función h, que pasa por a1,a2 e a3 e nos que dita función acada os valores g1, g2, g3



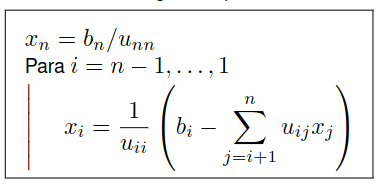
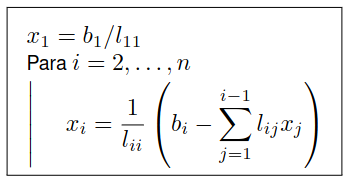
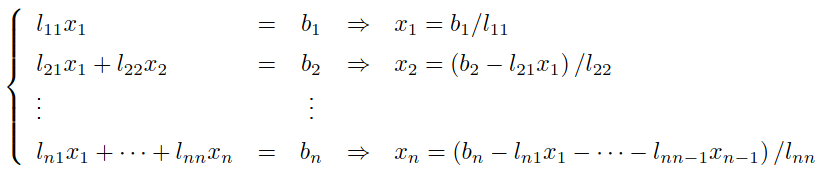
* + - * Calculamos un novo punto **a0** onde este polinomio acada o mínimo, sendo
  + En conclusión, a **sucesión do método do descenso rápido** ven dada por:



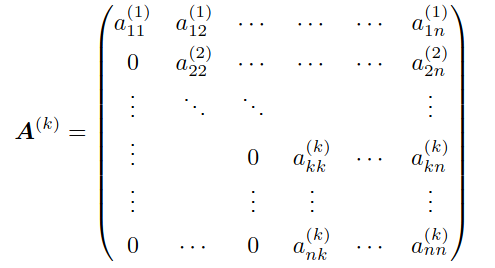
* + Onde, en cada iteración, existirán 4 valores posibles de a (a0,a1,a2,a3) que se obteñen co método previo. Deberemos escoller o que proporcione un valor mínimo de G.

**Resolución de sistemas mediante matrices LU**

* Emprégase **L** para denotar unha matriz triangular inferior e **U** para denotar unha triangular superior.
* Tras aplicar transformación lineais, podemos transformar un sistema para que sexa do tipo **Lx = b**, e atopar a súa solución mediante o algoritmo (**sustitución progresiva[[4]](#footnote-3))**:

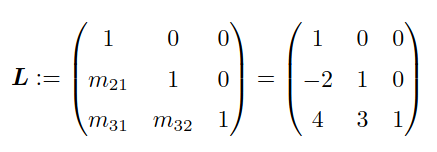
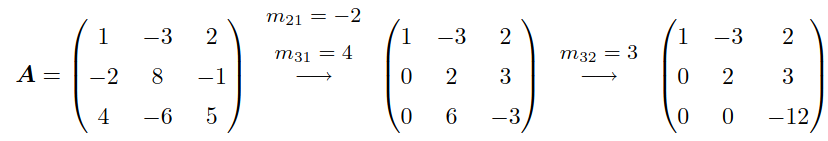


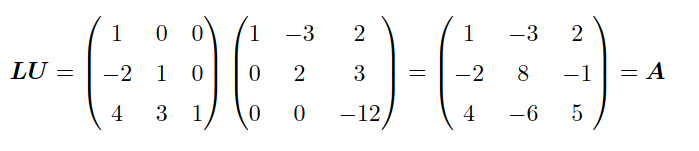
* Análogamente, para un sistema **Ux = b** (**sustitución regresiva):**

**Método de Eliminación Gaussiana**

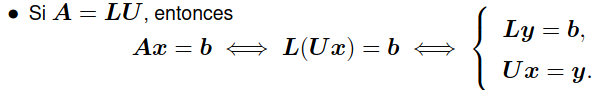
* Método polo cal se transforma un sistema Ax=b en outro **equivalente**, do tipo  esto é o escalonar matrices de toda a vida
* Se denotan con un superíndice a11**(1)** los elementos de la matriz antes de cambiarlos. El número del superíndice indica el número de transformaciones elementales que han recibido. La matriz resultante será A(k)x=b(k)
* Se asume que **aii(i)**!=0, para i∊[1,k-1]
* Custo operacional aproximado de 

**Factorización LU**

* Mediante Gauss, escalonamos una matriz para obtener una matriz triangular superior **U**.
* Con los factores utilizados, definimos una matriz L.
* Si los pivotes (mij) calculados son distintos de 0, e cumple siempre que **LU = A**.

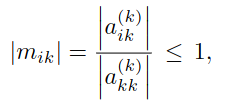


**Solución de sistemas mediante factorización LU**



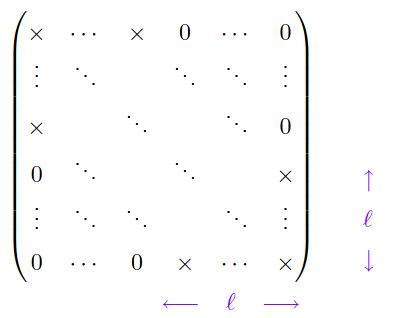
* Para resolver un sistema **Ax=b**, primero se resuelve **Ly=b** y luego **Ux=y**.
* Dado que la parte más costosa de este proceso es la factorización LU, el coste operacional total de resolución de un sistema es 

**Necesidad del pivoteo**

* Si existe un 0 en una posición de pivote, se debe intercambiar la ecuación con otra para evitarlo. Además, si los multiplicadores mij utilizados son muy grandes, se puede aumentar el error cometido.
* Para solucionar esto se utiliza el **pivoteo parcial**: en lugar de escalonar la matriz en el orden que aparezcan las filas, se leen primero los pivotes más grandes. 
  + De esta forma se reduce el error cometido y se evita tomar pivotes 0.
  + Tras esto, los multiplicadores no pasarán de 1 en módulo:

**Matrices de permutación**

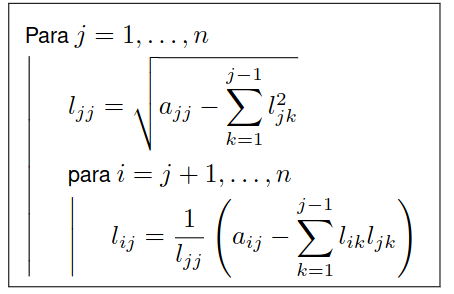
* Son matrices elementales.
* Intercambios de filas multiplicando pola esquerda.
* **Teorema:** Se A é non singular, existen matrices **LU = PA**, sendo P matriz de permutación. Para resolver o sistema, **Ly=Pb** e **Ux=y**

**Matrices banda**

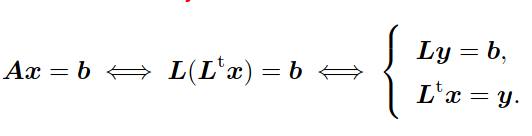
* Una matriz **A** es una matriz banda si **aij=0** cuando |i-j|>=**ℓ**, para **ℓ**<<n, siendo ℓ el **ancho de banda** de la matriz.
* Si una matriz banda se puede factorizar LU sin necesidad de pivoteo, entonces sus matrices **L** y **U** serán también banda y tendrán el mismo ancho que A.
* **Matriz tridiagonal:** Matriz banda donde **ℓ=2**. (algoritmo de tomas non entra)

**Matrices definidas positivas**

* Una matriz A nxn se dice **definida positiva** si 
* Se A é simétrica[[5]](#footnote-4), será tamén definida positiva se e solo se se cumplen **algunha** das seguintes:
  + Todos os autovalores de A son positivos
  + Todos os determinantes dos menores principais de A son positivos
  + Existe unha matriz L triangular inferior tal que A=LL**t**
    - Entonces, se A é simétrica e DP, pódese factorar A en matrices triangulares sen pivoteo, e a matriz triangular superior será simplemente a trasposta da inferior.



**Método de Cholesky**

* Aplícase a matrices simétricas e definidas positivas.
* Consiste en encontrar **L** tal que **A = LLt**.
* Custo operacional ≈ 1/3n3. Aprox a metade do do MEG, e máis estable. 
* Unha vez atopada a matriz **L**, realízase:

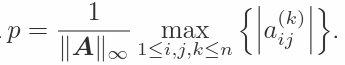
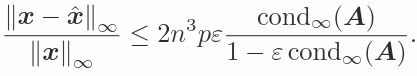
**Propagación de error, número de condición**

* Supongamos que se quiere resolver el sistema **Ax = b**, pero que del segundo miembro **b** se conoce un valor aproximado **b’**.
* Supongamos que b’ tiene un error δb, por lo que b’ = b + δb
* Sea δx=x’-x el error cometido al calcular la solución aproximada x’. Entonces



* Definimos **(||A|| ||A-1||)** como el **número de condición cond(A)**.
  + El número de condición cond(A) es una cota del factor del propagación entre el error del dato b’ y el error que se cometerá en la solución x’.
  + Para toda matriz no singular, **cond(A)>=1**.
    - Si cond(A)≈1, el sistema está bien condicionado. Si es mucho mayor, está mal condicionado.
  + Depende esencialmente de la matriz A, no del segundo miembro b.

**Propagación de error en computación**

* Sea x la solución de un sistema Ax=b. Sea **x̂** ∈ Rn la solución calculada mediante MEG con pivoeo parcial, en un computador con cte. de precisión ε.
* Sea
* Si ε (cond A)<1, se acota el error relativo de la siguiente forma:
* Entonces, si el error es muy grande, será debido a la dimensión del sistema **n**, la cte. de precisión **ε** o el número de condición **cond(A)**.

**Estimación del error de una solución calculada**

* **r**:= b - A**x̂** es el **residuo** de la solución calculada. Podemos obtener este valor una vez tenemos la solución calculada **x̂**.
* **e**:= x - **x̂** es el **error** de la solución calculada. Para calcularlo, resolvemos **Ae = r.** 
  + Obtendremos una estimación del tamaño del error, **‖ê‖**

1. Na imaxe, a matriz jacobiana para unha función de n variables. Para n=2 sería unha matriz 2x2 [↑](#footnote-ref-0)
2. G é unha función R2→R1. A función só toma valores positivos. Cando vale 0, tanto f1 como f2 valen 0, polo que os mínimos desta función son os puntos onde se anula f. [↑](#footnote-ref-1)
3. ||f(x1,x2)|| denota o **módulo** do vector f, é dicir, a raíz da suma dos cadrados de cada termo da ecuación. Logo, vólvese a elevar ao cadrado para anular a raíz. [↑](#footnote-ref-2)
4. Custo operacional: n2 [↑](#footnote-ref-3)
5. Non ten por que selo para ser mdp [↑](#footnote-ref-4)