Métodos Directos

de resolución de SEAL

Victorio E. Sonzogni

Sistema de ecuaciones lineales

- Un acertijo:
  La edad de un hombre y la de su hijo suman 40 años. Y hace diez años su edad doblaba a la de su hijo
  Todos saben como resolverlo. Tenemos dos incógnitas: la edad del hombre H, y la de su hijo h. Y dos condiciones (que proporcionan dos ecuaciones): su suma es 40, y restando 10 su cociente es 2.

$$\begin{aligned} H + h &= 40 \\ (H - 10) &= 2\,(h - 10) \end{aligned}$$

- Este es un sistema de ecuaciones con dos incognitas. También se obtiene un sistema similar si se busca las coordenadas de intersección de dos rectas ($y = ax + b$ y $y = cx + d$).

Sistema de ecuaciones lineales

- En muchos casos nos encontramos con la necesidad de resolver sistemas de ecuaciones algebraicas lineales (SEAL), que pueden escribirse:

$$\begin{aligned}
E_1: &\quad a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n &&= b_1 \\
E_2: &\quad a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2n}x_n &&= b_2 \\
&\qquad\qquad \ldots \\
E_n: &\quad a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \ldots + a_{nn}x_n &&= b_n
\end{aligned}$$

y representarse matricialmente:

$$\mathbf{Ax} = \mathbf{b}$$

donde

- $\mathbf{x}$ es un vector conteniendo n incógnitas;
- $\mathbf{A}$ una matriz cuadrada de $n \times n$ con los coeficientes del sistema;
- $\mathbf{b}$ un vector con los términos independientes,

Sistema de ecuaciones lineales

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

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

Sistema de ecuaciones lineales

- A lo largo del curso encontraremos varios casos, pero algunos ejemplos de problemas que llevan a resolver un SEAL pueden ser:
  - Construir un polinomio

$$P_n(x) = a_n\ x^n + a_{n-1}\ x^{n-1} + \ldots + a_1\ x + a_0$$

    de grado n que pase por $n+1$ puntos. Esta condición (que pase por esos puntos) nos proporciona las $n+1$ ecuaciones, con las cuales calcularemos las $n+1$ incognitas (los a_i).
  - En una estructura reticulada (un puente de ferrocarril) los esfuerzos en cada barra conforman el vector $\mathbf{x}$ de incognitas. Y las ecuaciones surgen de condiciones de *equilibrio* de esas fuerzas (y eventualmente de *compatibilidad* de deformaciones).
  - En el último capítulo se verá como utilizar un método numérico (diferencias finitas) para hallar la solución aproximada a un problema planteado en un medio continuo. En ese caso se llega tambien al planteo de un SEAL.

Sistemas de ecuaciones lineales

- Las características de la matriz determinan si el sistema de ecuaciones tiene solución; si ésta única; si es poco sensible a la variación en los datos; si es fácil o dificil de resolver; etc.
- El sistema de ecuaciones anterior tiene solución única si se da una de las siguientes (equivalentes) condiciones:
  - La matriz $\mathbf{A}$ es invertible (no singular)
  - El rango de $\mathbf{A}$ es n (su determinante no es nulo)
  - El sistema $\mathbf{Ax} = \mathbf{0}$ admite solamente la solución nula.

Tipos de matrices

En cuanto a la estructura de las matrices, éstas se denominan:

- **Matriz general**: el caso más general que no corresponde a alguno de los listados abajo
- **Matriz simétrica**: cuando $a_{ij} = a_{ji}$
- **Matriz diagonal**: cuando sus elementos son todos nulos (0), salvo en la diagonal ($a_{ii} \neq 0$)
- **Matriz triangular**: cuando los elementos estrictamente por arriba de la diagonal son nulos (matriz triangular inferior), o los elementos estrictamente por debajo de la diagonal son nulos (matriz triangular superior)
- **Matriz banda**: cuando hay una banda, alrededor de la diagonal principal, donde pueden estar los elementos no nulos. Fuera de esa banda son todos nulos.
- **Matriz rala** (*sparse*): matriz poblada de ceros con pocos elementos no nulos.

Tipos de matrices

- Definición: **Matriz diagonalmente dominante**:
  Una matriz se dice *estrictamente diagonal dominante* si

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

- Una matriz estrictamente diagonal dominante es no singular.
- Definición: **Matriz simétrica definida positiva**:
  Una matriz $\mathbf{A}$ simétrica se dice *definida positiva* si

$$\mathbf{x^T A x} > 0 \qquad \forall \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq \mathbf{0}$$

  Obs: Una matriz $\mathbf{B}$ no simétrica es positiva definida si y solo si su parte simétrica $\frac{1}{2}(\mathbf{B} + \mathbf{B}^T)$ lo es

Tipos de matrices

- También una matriz es definida positiva si y sólo si todos sus autovalores son positivos.
- Si una matriz simétrica $\mathbf{A}$ es definida positiva, entonces:

  a) $\mathbf{A}$ es no singular

  b) $a_{ii} > 0 \qquad \forall i$

  c) $\max\limits_{\substack{1\leq i\leq n\\ 1\leq j\leq n}} |a_{ij}| \leq \max\limits_{1\leq i\leq n} |a_{ii}|$

  d) $a_{ij}^2 < a_{ii}a_{jj} \qquad \forall i \neq j$

Sistemas de ecuaciones lineales

- La resolución de un SEAL puede realizarse de diferentes métodos que suelen clasificarse en:.
  - Métodos *directos*
    En ellos a través un número finito de pasos se obtiene la solución del problema.
  - Métodos *iterativos*
    En estos se contruye una secuencia de soluciones que se aproxima a la solución.

Sistemas de ecuaciones lineales

- Sistemas equivalentes:
  Dos sistemas $\mathbf{Ax} = \mathbf{b}$ y $\mathbf{Bx} = \mathbf{d}$ se dicen *equivalentes* si tienen la misma solución $\mathbf{x}$
- Se puede demostrar que realizando una serie de *operaciones elementales* sobre un SEAL se obtiene otro equivalente.
- Operaciones elementales:
  - Intercambio de 2 ecuaciones: $E_i \leftrightarrow E_j$
  - Multiplicación de una ecuación por un escalar $(\neq 0)$: $\lambda E_i \rightarrow E_i$
  - Sumar a una ecuación un multiplo de otra: $E_i + \lambda E_j \rightarrow E_i$

Métodos directos

- Hay métodos que se basan en transformar un SEAL mediante operaciones elementales de modo de obtener un sistema equivalente, más fácil de resolver.
- ¿Que sistemas pueden ser fáciles para resolver? Por ejemplo:
  - Sistemas con matrices diagonales
  - Sistemas con matrices triangulares

Sistemas con matrices diagonales

$$\mathbf{A} = \begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ 0 & a_{22} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix}$$

- Su resolución es trivial. La solución es:

$$\mathbf{x} = \begin{bmatrix} b_1/a_{11} \\ b_2/a_{22} \\ \vdots \\ b_n/a_{nn} \end{bmatrix}$$

Sistemas con matrices triangulares

- Para un sistema con matriz triangular superior:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix}$$

- La resolución comienza con el término x_n y progresa hacia arriba

$$x_n = b_n/a_{nn}$$
$$x_{n-1} = (b_{n-1} - a_{n-1,n}x_n)/a_{n-1,n-1}$$
$$\dots$$
$$x_i = (b_i - \sum_{j=i+1}^{n} a_{ij}x_j)/a_{ii} \qquad \text{para i} = \text{n} - 1, \dots 1$$

- Análogamente se resuelve un sistema con matriz triangular inferior.

Eliminación de Gauss

- Un método basado en esta transformación del sistema de ecuaciones es el método de Eliminación de Gauss
- Se basa en realizar una serie de transformaciones sobre el sistema de modo de obtener un sistema equivalente, con matriz triangular.
- En un sistema de n ecuaciones se efectuan $n-1$ pasos de eliminación.
- La matriz $\mathbf{A}$ del sistema, va siendo transformada en matrices $\mathbf{A^{(k)}}$ en cada paso k de la eliminación.
- El vector de términos independientes va transformándose también en sucesivos vectores $\mathbf{b^{(k)}}$
- Se introducirá el método a través de un ejemplo sencillo.

Eliminación de Gauss

Ejemplo:

$$\begin{matrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{matrix} \begin{bmatrix} 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \\ 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 34 \\ 27 \\ -38 \end{bmatrix}$$

Se propone transformar el sistema en varios pasos

En el primer paso se propone un sistema:

$$\begin{matrix} E_1 \\ E_2 - 2E_1 \\ E_3 - \frac{1}{2}E_1 \\ E_4 - (-1)E_1 \end{matrix} \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & -12 & 8 & 1 \\ 0 & 2 & 3 & -14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ 21 \\ -26 \end{bmatrix}$$

Eliminación de Gauss

En el segundo paso se propone un sistema:

$$\begin{matrix} E_1 \\ E_2 \\ E_3 - 3E_2 \\ E_4 - (-\frac{1}{2})E_2 \end{matrix} \quad \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 4 & -13 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ -9 \\ -21 \end{bmatrix}$$

En el tercer paso:

$$\begin{matrix} E_1 \\ E_2 \\ E_3 \\ E_4 - 2E_3 \end{matrix} \quad \begin{bmatrix} 6 & -2 & 2 & 4 \\ 0 & -4 & 2 & 2 \\ 0 & 0 & 2 & -5 \\ 0 & 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 10 \\ -9 \\ -3 \end{bmatrix}$$

Eliminación de Gauss

- El sistema ha quedado transformado en uno equivalente con matriz triangular superior, que es más sencillo para resolver.
- La solución del sistema triangular se realiza por retrosustitución, dando:

$$\mathbf{x} = \begin{bmatrix} 1 \\ -3 \\ -2 \\ 1 \end{bmatrix}$$

Eliminación de Gauss

- Se forma una sucesión de sistemas con matrices $\mathbf{A}^{(k)}$
- En el paso k la $k-esima$ fila queda inalterada, igual que las filas superiores a ella. Se modifican las filas inferiores.
- En el paso k el elemento a_{kk} se denomina *pivote*. La fila k se llama *fila pivote* y la columna k *columna pivote*

Eliminación de Gauss

$$\mathbf{A}^{(k)} = \begin{array}{c} \\ \\ fila\ k \\ \\ fila\ i \\ \\ \end{array} \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \\ \\ \\ col.k \\ \\ \\ \end{array} \begin{array}{c} \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \\ \\ \\ col.j \\ \\ \\ \end{array} \begin{bmatrix} a_{11}^{(k)} & \dots & a_{1,k-1}^{(k)} & a_{1k}^{(k)} & \dots & a_{1j}^{(k)} & \dots & a_{1n}^{(k)} \\ \vdots & \vdots & \vdots & & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & a_{kk}^{(k)} & \dots & a_{kj}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & \vdots & & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & a_{ik}^{(k)} & \dots & a_{ij}^{(k)} & \dots & a_{in}^{(k)} \\ \vdots & \vdots & & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & a_{nk}^{(k)} & \dots & a_{nj}^{(k)} & \dots & a_{nn}^{(k)} \end{bmatrix}$$

Eliminación de Gauss

Los elementos de la matriz $\mathbf{A}^{(k+1)}$ se calculan:

$$a_{ij}^{(k+1)} = \begin{cases} a_{ij}^{(k)} & si \quad i \leq k \\ a_{ij}^{(k)} - \left(\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \right) a_{kj}^{(k)} & si \quad i \geq k+1 \quad y \quad j \geq k+1 \\ 0 & si \quad i \geq k+1 \quad y \quad j \leq k \end{cases}$$

Algoritmo para eliminación de Gauss

Input: n, $\tilde{\mathbf{A}} = [\mathbf{A}\ \mathbf{b}]$
Output: $\mathbf{x}$, o mensaje de error

Eliminacion

for $k = 1, 2, \ldots n-1$ do	for k=1:n-1
for $i = k+1, k+2, \ldots n$ do	for i=k+1:n
$m \leftarrow a_{ik}/a_{kk}$	m = A(i,k)/A(k,k);
$a_{ik} \leftarrow 0$	A(i,k)=0;
for $j = k+1, k+2, \ldots n+1$ do	for j=k+1:n+1
$a_{ij} \leftarrow a_{ij} - m\, a_{kj}$	A(i,j)=A(i,j)-m*A(k,j);
end	endfor
end	endfor
end	endfor

Algoritmo para eliminación de Gauss

Input: n, $\tilde{\mathbf{A}} = [\mathbf{A}\ \mathbf{b}]$
Output: $\mathbf{x}$, o mensaje de error

function [x] = gauss1(A,b)
n=length(b);
A=[A b];

Eliminacion

for $k = 1, 2, \ldots n-1$ do	for k=1:n-1
for $i = k+1, k+2, \ldots n$ do	for i=k+1:n
$m \leftarrow a_{ik}/a_{kk}$	m = A(i,k)/A(k,k);
$a_{ik} \leftarrow 0$	A(i,k)=0;
for $j = k+1, k+2, \ldots n+1$ do	for j=k+1:n+1
$a_{ij} \leftarrow a_{ij} - m\, a_{kj}$	A(i,j)=A(i,j)-m*A(k,j);
end	endfor
end	endfor
end	endfor

Algoritmo para eliminación de Gauss

Input: n, $\tilde{\mathbf{A}} = [\mathbf{A}\ \mathbf{b}]$
Output: $\mathbf{x}$, o mensaje de error

function [x] = gauss1(A,b)
n=length(b);
A=[A b];

Eliminacion

for $k = 1, 2, \ldots n-1$ do — for k=1:n-1
  for $i = k+1, k+2, \ldots n$ do — for i=k+1:n
    $m \leftarrow a_{ik}/a_{kk}$ — m = A(i,k)/A(k,k);
    $a_{ik} \leftarrow 0$ — A(i,k)=0;
    for $j = k+1, k+2, \ldots n+1$ do — for j=k+1:n+1
      $a_{ij} \leftarrow a_{ij} - m\, a_{kj}$ — A(i,j)=A(i,j)-m*A(k,j);
    end — endfor
  end — endfor
end — endfor

Algoritmo para eliminación de Gauss

Input: n, $\tilde{\mathbf{A}} = [\mathbf{A}\ \mathbf{b}]$
Output: $\mathbf{x}$, o mensaje de error

function [x] = gauss1(A,b)
n=length(b);
A=[A b];

Eliminacion

for $k = 1, 2, \ldots n-1$ do	for k=1:n-1
for $i = k+1, k+2, \ldots n$ do	for i=k+1:n
$m \leftarrow a_{ik}/a_{kk}$	m = A(i,k)/A(k,k);
$a_{ik} \leftarrow 0$	A(i,k)=0;
for $j = k+1, k+2, \ldots n+1$ do	for j=k+1:n+1
$a_{ij} \leftarrow a_{ij} - m\, a_{kj}$	A(i,j)=A(i,j)-m*A(k,j);
end	endfor
end	endfor
end	endfor

Algoritmo para eliminación de Gauss

Input: n, $\tilde{\mathbf{A}} = [\mathbf{A}\ \mathbf{b}]$
Output: $\mathbf{x}$, o mensaje de error

function [x] = gauss1(A,b)
n=length(b);
A=[A b];

Eliminacion

for $k = 1, 2, \ldots n-1$ do	for k=1:n-1
for $i = k+1, k+2, \ldots n$ do	for i=k+1:n
$m \leftarrow a_{ik}/a_{kk}$	m = A(i,k)/A(k,k);
$a_{ik} \leftarrow 0$	A(i,k)=0;
for $j = k+1, k+2, \ldots n+1$ do	for j=k+1:n+1
$a_{ij} \leftarrow a_{ij} - m\, a_{kj}$	A(i,j)=A(i,j)-m*A(k,j);
end	endfor
end	endfor
end	endfor

Algoritmo para eliminación de Gauss

Input: n, $\tilde{\mathbf{A}} = [\mathbf{A}\ \mathbf{b}]$
Output: $\mathbf{x}$, o mensaje de error

function [x] = gauss1(A,b)
n=length(b);
A=[A b];

Eliminacion

for $k = 1, 2, \ldots n-1$ do	for k=1:n-1
for $i = k+1, k+2, \ldots n$ do	for i=k+1:n
$m \leftarrow a_{ik}/a_{kk}$	m = A(i,k)/A(k,k);
$a_{ik} \leftarrow 0$	A(i,k)=0;
for $j = k+1, k+2, \ldots n+1$ do	for j=k+1:n+1
$a_{ij} \leftarrow a_{ij} - m\, a_{kj}$	A(i,j)=A(i,j)-m*A(k,j);
end	endfor
end	endfor
end	endfor

Algoritmo para eliminación de Gauss

Input: n, $\tilde{\mathbf{A}} = [\mathbf{A}\ \mathbf{b}]$
Output: $\mathbf{x}$, o mensaje de error

function [x] = gauss1(A,b)
n=length(b);
A=[A b];

Eliminacion

for $k = 1, 2, \ldots n-1$ do	for k=1:n-1
for $i = k+1, k+2, \ldots n$ do	for i=k+1:n
$m \leftarrow a_{ik}/a_{kk}$	m = A(i,k)/A(k,k);
$a_{ik} \leftarrow 0$	A(i,k)=0;
for $j = k+1, k+2, \ldots n+1$ do	for j=k+1:n+1
$a_{ij} \leftarrow a_{ij} - m\, a_{kj}$	A(i,j)=A(i,j)-m*A(k,j);
end	endfor
end	endfor
end	endfor

Algoritmo para eliminación de Gauss (cont.)

if $a_{nn} = 0 \quad \rightarrow$ mens. error: 'no hay sol. unica'

```
if (A(n,n)==0) disp('no hay sol. unica'), endif
```

Retrosustitucion

```
function x=sust_atras1(A)
```

$x_n = a_{n,n+1}/a_{nn}$

```
x(n)=A(n,n+1)/A(n,n);
```

for $i = n-1, n-2, \ldots 1$ do

```
for i=n-1:-1:1
```

$s \leftarrow a_{i,n+1}$

```
s=A(i,n+1);
```

for $j = i+1, i+2, \ldots n$ do

```
for j=i+1:n
```

$s \leftarrow s - a_{ij}x_j$

```
s=s-A(i,j)*x(j);
```

end

```
endfor
```

$x_i \leftarrow s/a_{ii}$

```
x(i)=s/A(i,i);
```

end

```
endfor
```

Algoritmo para eliminación de Gauss (cont.)

if $a_{nn} = 0 \quad \rightarrow$ mens. error: 'no hay sol. unica'

```
if (A(n,n)==0) disp('no hay sol. unica'), endif
```

Retrosustitucion

```
function x=sust_atras1(A)
```

$x_n = a_{n,n+1}/a_{nn}$

```
x(n)=A(n,n+1)/A(n,n);
```

for $i = n-1, n-2, \ldots 1$ do

```
for i=n-1:-1:1
```

$s \leftarrow a_{i,n+1}$

```
s=A(i,n+1);
```

for $j = i+1, i+2, \ldots n$ do

```
for j=i+1:n
```

$s \leftarrow s - a_{ij}x_j$

```
s=s-A(i,j)*x(j);
```

end

```
endfor
```

$x_i \leftarrow s/a_{ii}$

```
x(i)=s/A(i,i);
```

end

```
endfor
```

Algoritmo para eliminación de Gauss (cont.)

if $a_{nn} = 0 \quad \rightarrow$ mens. error: 'no hay sol. unica'

```
if (A(n,n)==0) disp('no hay sol. unica'), endif
```

Retrosustitucion

```
function x=sust_atras1(A)
```

$x_n = a_{n,n+1}/a_{nn}$

```
x(n)=A(n,n+1)/A(n,n);
```

for $i = n-1, n-2, \ldots 1$ do

```
for i=n-1:-1:1
```

$s \leftarrow a_{i,n+1}$

```
s=A(i,n+1);
```

for $j = i+1, i+2, \ldots n$ do

```
for j=i+1:n
```

$s \leftarrow s - a_{ij}x_j$

```
s=s-A(i,j)*x(j);
```

end

```
endfor
```

$x_i \leftarrow s/a_{ii}$

```
x(i)=s/A(i,i);
```

end

```
endfor
```

Algoritmo para eliminación de Gauss (cont.)

if $a_{nn} = 0 \quad \rightarrow$ mens. error: 'no hay sol. unica'

```
if (A(n,n)==0) disp('no hay sol. unica'), endif
```

Retrosustitucion

```
function x=sust_atras1(A)
```

$x_n = a_{n,n+1}/a_{nn}$

```
x(n)=A(n,n+1)/A(n,n);
```

for $i = n-1, n-2, \ldots 1$ do

```
for i=n-1:-1:1
```

$s \leftarrow a_{i,n+1}$

```
s=A(i,n+1);
```

for $j = i+1, i+2, \ldots n$ do

```
for j=i+1:n
```

$s \leftarrow s - a_{ij}x_j$

```
s=s-A(i,j)*x(j);
```

end

```
endfor
```

$x_i \leftarrow s/a_{ii}$

```
x(i)=s/A(i,i);
```

end

```
endfor
```

Algoritmo para eliminación de Gauss (cont.)

if $a_{nn} = 0 \quad \rightarrow$ mens. error: 'no hay sol. unica'

```
if (A(n,n)==0) disp('no hay sol. unica'), endif
```

Retrosustitucion

```
function x=sust_atras1(A)
```

$x_n = a_{n,n+1}/a_{nn}$

```
x(n)=A(n,n+1)/A(n,n);
```

for $i = n-1, n-2, \ldots 1$ do

```
for i=n-1:-1:1
```

$s \leftarrow a_{i,n+1}$

```
s=A(i,n+1);
```

for $j = i+1, i+2, \ldots n$ do

```
for j=i+1:n
```

$s \leftarrow s - a_{ij}x_j$

```
s=s-A(i,j)*x(j);
```

end

```
endfor
```

$x_i \leftarrow s/a_{ii}$

```
x(i)=s/A(i,i);
```

end

```
endfor
```

Algoritmo para eliminación de Gauss (cont.)

if $a_{nn} = 0 \quad \rightarrow$ mens. error: 'no hay sol. unica'

```
if (A(n,n)==0) disp('no hay sol. unica'), endif
```

Retrosustitucion

```
function x=sust_atras1(A)
```

$x_n = a_{n,n+1}/a_{nn}$

```
x(n)=A(n,n+1)/A(n,n);
```

for $i = n-1, n-2, \ldots 1$ do

```
for i=n-1:-1:1
```

$s \leftarrow a_{i,n+1}$

```
s=A(i,n+1);
```

for $j = i+1, i+2, \ldots n$ do

```
for j=i+1:n
```

$s \leftarrow s - a_{ij}x_j$

```
s=s-A(i,j)*x(j);
```

end

```
endfor
```

$x_i \leftarrow s/a_{ii}$

```
x(i)=s/A(i,i);
```

end

```
endfor
```

```
function [x] = gauss1(A,b)
n=length(b);
A=[A b];
% Eliminacion
for k=1:n-1
  for i=k+1:n
    m = A(i,k)/A(k,k);
    A(i,k)=0;
    for j=k+1:n+1
      A(i,j) = A(i,j)-m*A(k,j);
    endfor
  endfor
endfor
if (A(n,n)==0)
  disp('no hay sol. unica')
endif
x=sust_atras1(A); %retrosustitucion
```

```
function x=sust_atras1(A)
x=A(:,end); %necesario para que x sea columna
n=length(x); %definimos n por ser una variable local
x(n)=A(n,n+1)/A(n,n);
for i = n-1:-1:1
  s = A(i,n+1);
  for j = i+1:n
    s = s - A(i,j)*x(j);
  endfor
  x(i) = s/A(i,i);
endfor
```

Factorización LU

Una forma de abordar la solución de $\mathbf{Ax} = \mathbf{b}$ es *factorizar* la matriz. Es decir buscar

$$\mathbf{A} = \mathbf{LU}$$

donde $\mathbf{L}$ y $\mathbf{U}$ son matrices triangulares inferior y superior, respectivamente, tales que multiplicadas dan la matriz $\mathbf{A}$
Si se tiene esa factorización, el sistema se puede escribir:

$$\mathbf{Ax} = \mathbf{LUx} = \mathbf{b}$$

LLamando

$$\mathbf{Ux} = \mathbf{y}$$

El sistema queda

$$\mathbf{Ly} = \mathbf{b}$$

De esta última ecuación se obtiene el vector $\mathbf{y}$ y de la anterior, el vector $\mathbf{x}$

Factorización LU

La solución se hace entonces en tres etapas:

1) Factorización de la matriz

$$\mathbf{A} = \mathbf{LU}$$

2) Solución del sistema

$$\mathbf{Ly} = \mathbf{b}$$

por sustitución hacia adelante

3) Solución del sistema

$$\mathbf{Ux} = \mathbf{y}$$

por sustitución hacia atrás

Factorización LU

- La factorización $\mathbf{LU}$ no es única
- Factorización de Doolittle: los términos de la diagonal de $\mathbf{L}$ son unitarios
- Factorización de Crout: los términos de la diagonal de $\mathbf{U}$ son unitarios

Factorización LU

- Si se observa el ejemplo de eliminación de Gauss, puede verse que en la primera etapa el sistema se construyó:

$$
\begin{aligned}
&E_1 \\
&E_2 - \frac{a_{21}}{a_{11}} E_1 \\
&E_3 - \frac{a_{31}}{a_{11}} E_1 \\
&E_4 - \frac{a_{41}}{a_{11}} E_1
\end{aligned}
$$

- En la segunda:

$$
\begin{aligned}
&E_1 \\
&E_2 \\
&E_3 - \frac{a_{32}}{a_{22}} E_2 \\
&E_4 - \frac{a_{42}}{a_{22}} E_2
\end{aligned}
$$

Factorización LU

- En la tercera etapa:

$$
\begin{aligned}
&E_1 \\
&E_2 \\
&E_3 \\
&E_4 - \frac{a_{43}}{a_{33}} E_3
\end{aligned}
$$

- Los multiplicadores usados pueden colocarse para formar una matriz triangular inferior:

$$
\mathbf{L} = \begin{bmatrix}
1 & 0 & 0 & 0 \\
\frac{a_{21}}{a_{11}} & 1 & 0 & 0 \\
\frac{a_{31}}{a_{11}} & \frac{a_{32}}{a_{22}} & 1 & 0 \\
\frac{a_{41}}{a_{11}} & \frac{a_{42}}{a_{22}} & \frac{a_{43}}{a_{33}} & 1
\end{bmatrix}
$$

Factorización LU

- Si se define

$$\mathbf{U} = \mathbf{A}^{(n-1)}$$

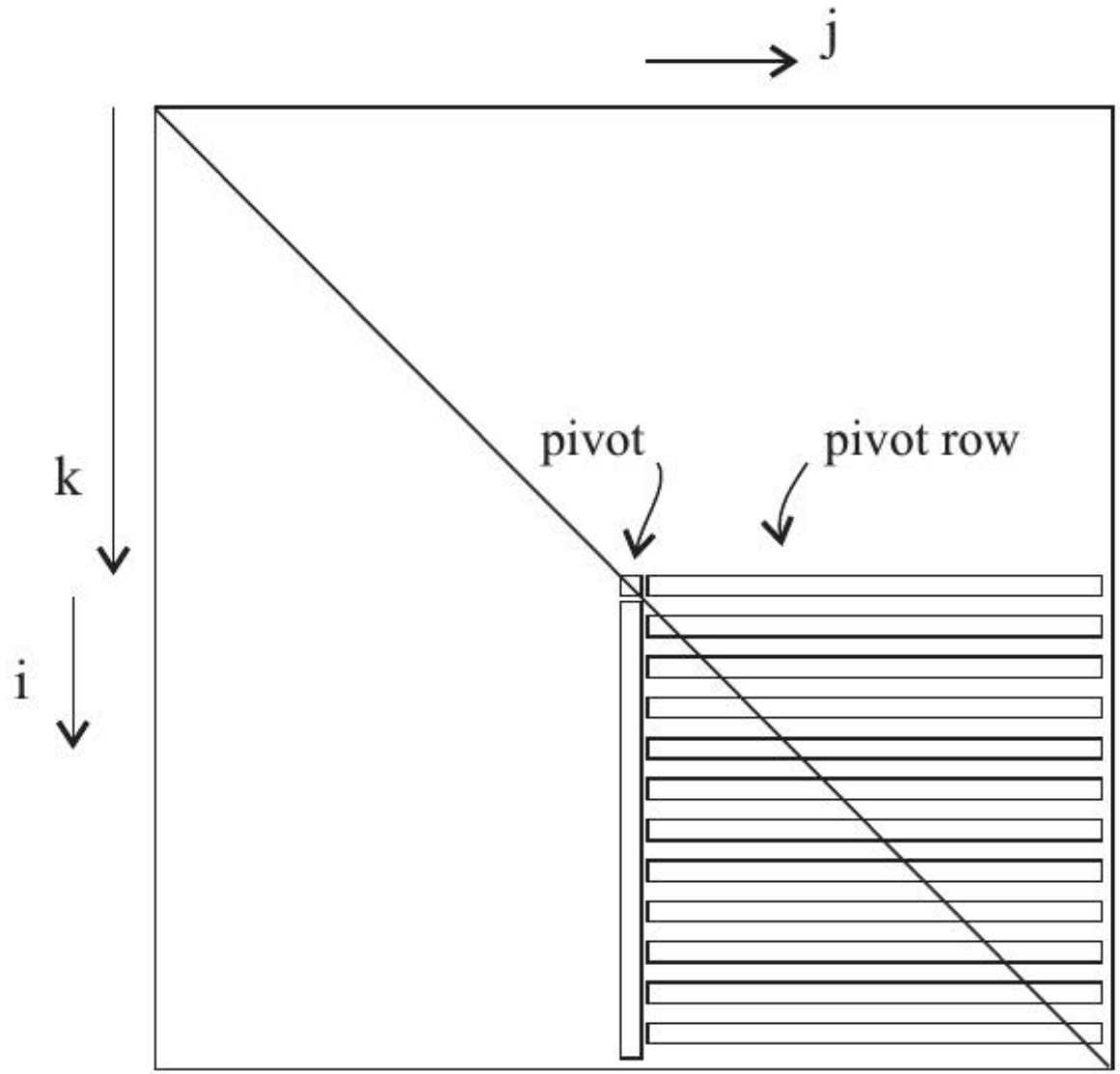
siendo esta la última matriz obtenida en el proceso de eliminación de Gauss. Ésta matriz, junto con la matriz triangular inferior $\mathbf{L}$ de la transparencia anterior, son los factores de la matriz $\mathbf{A}$.

- Teorema: Si todos los elementos $a_{kk}^{(k)}$ de la descomposicion de Gauss son distintos de cero, entonces $\mathbf{A} = \mathbf{LU}$, donde $\mathbf{L}$ y $\mathbf{U}$, son las matrices triangulares recién definidas.
- Siguiendo las operaciones como en el algoritmo de eliminación de Gauss, el algoritmo para descomposición $\mathbf{LU}$ se muestra a continuación.
- Las matrices $\mathbf{L}$ y $\mathbf{U}$ se almacenan sobre la matriz original $\mathbf{A}$. (La diagonal de $\mathbf{L}$ no precisa almacenarse pues son todos 1).

Factorización LU kij

```
for k = 1, 2, ... n − 1 do                          fila pivotal
        for i = k + 1, ... n do
                s ← a_ik / a_kk
                a_ik ← s                                    l_ik
                for j = k + 1, ... n do
                        a_ij ← a_ij − s a_kj                u_ij
                end
        end
end
```

Factorización LU kij



Factorización LU

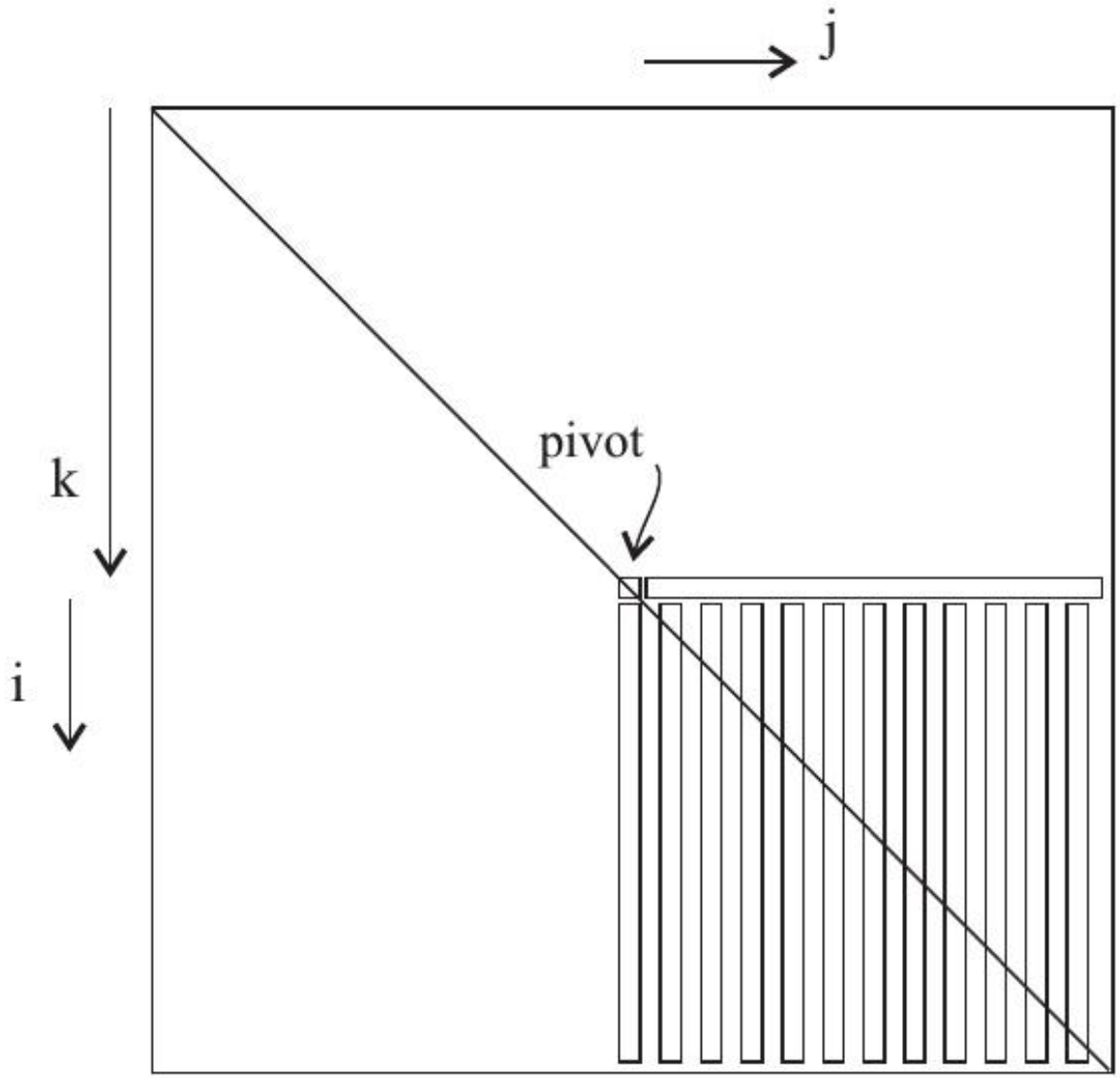
- Para cada etapa, parándose en el pivote (índice k), se inicia un ciclo sobre las filas (índice i), y otro sobre las columnas (índice j).
- Por debajo del pivote (a_{kk}) queda formada una columna de la matriz $\mathbf{L}$.
- A la derecha de la diagonal va quedando formada la matriz $\mathbf{U}$.
- La parte más cara del procedimiento es la generación de la matriz $\mathbf{U}$, que se hace dentro de tres ciclos anidados.
- Esa operación puede hacerse de distintas formas, cambiando el orden en que se recorren esos tres ciclos. La indicada se denomina kij por el orden en que se realizaron las cuentas.
- A continuación se muestran las factorizaciones kji y jki

Factorización LU kji

```
for k = 1, ... n − 1 do
        for i = k + 1, ... n do
                s ← a_ik / a_kk
                a_ik ← s                                  l_ik
        end

        for j = k + 1, ... n do
                for i = k + 1, ... n do
                        a_ij ← a_ij − a_ik a_kj            u_ij
                end
        end
end
```

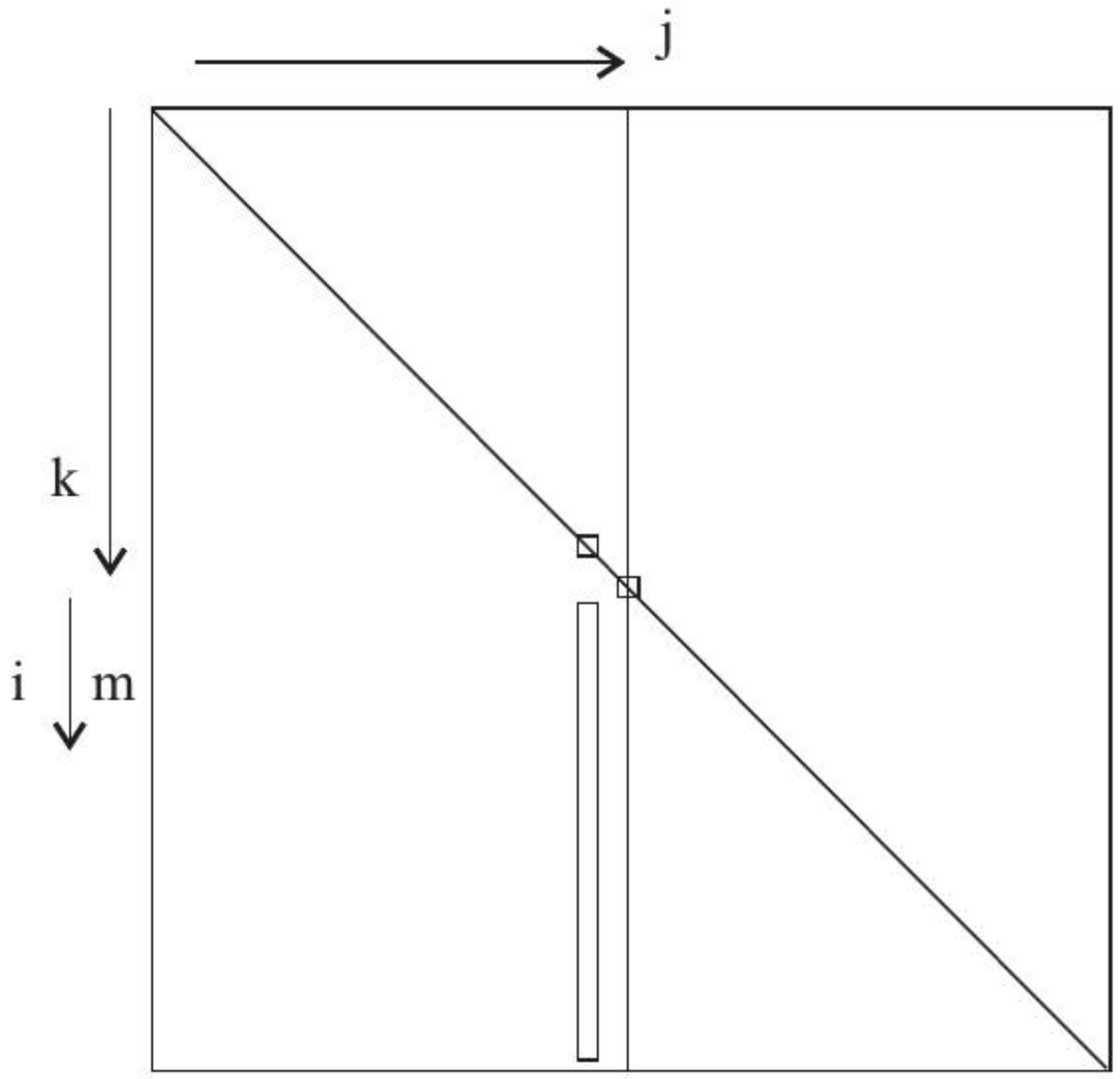
Factorización LU kji



Factorización LU jki

for $j = 2, \ldots n$ do
  for $m = j, \ldots n$ do
    $s \leftarrow a_{m,j-1}/a_{j-1,j-1}$
    $a_{m,j-1} \leftarrow s$ $\qquad l_{mk}$
  end
  for $k = 1, \ldots j-1$ do
    for $i = k+1, \ldots n$ do
      $a_{ij} \leftarrow a_{ij} - a_{ik}\, a_{kj}$ $\qquad u_{ij}$
    end
  end
end

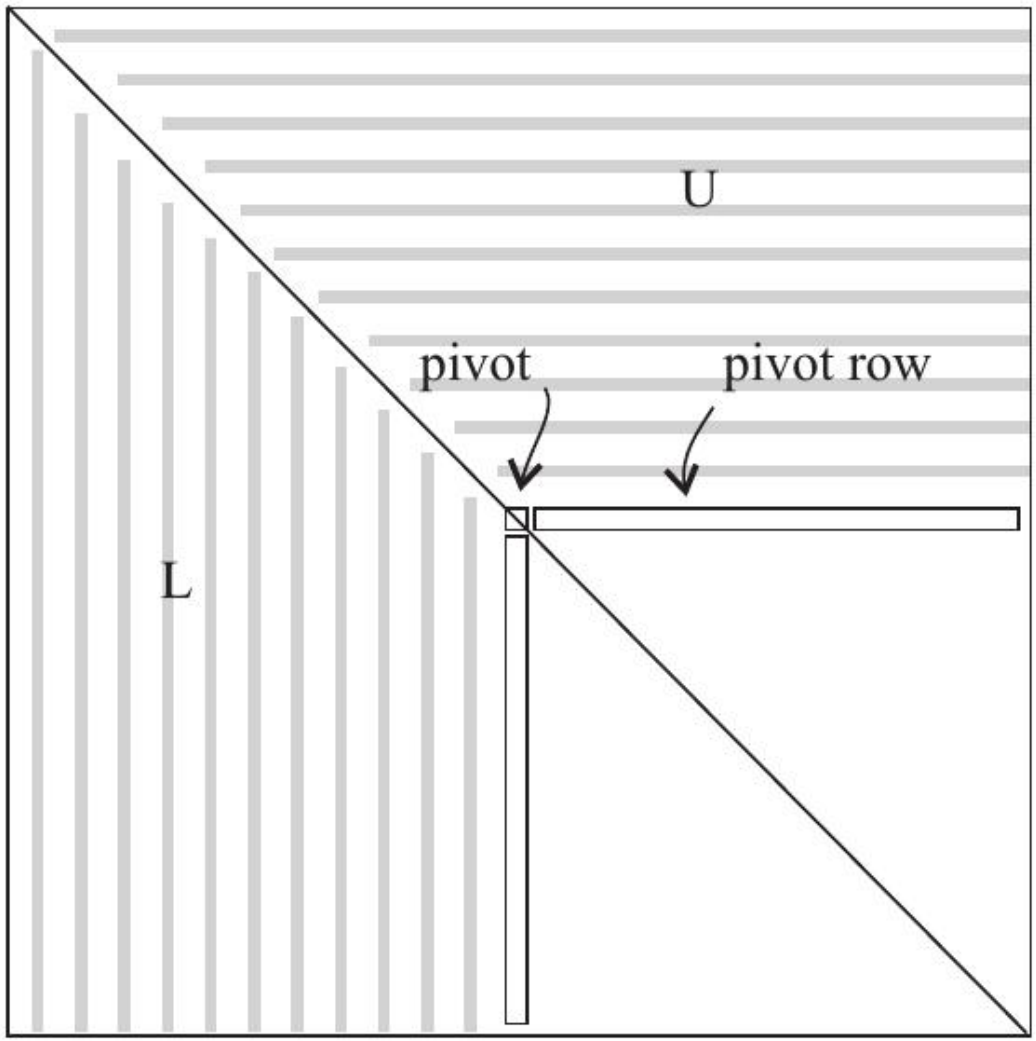
Factorización LU jki



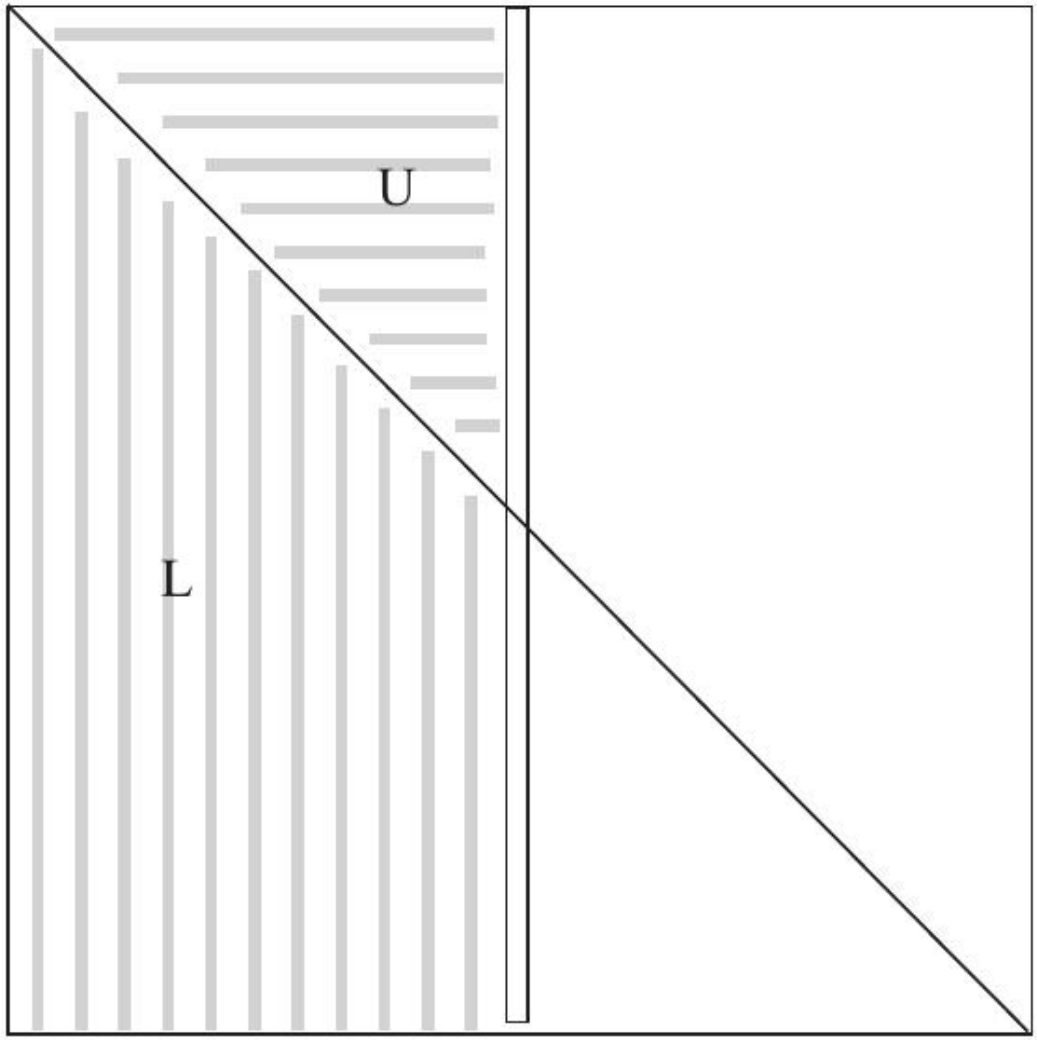
Factorización LU

- En realidad los tres ciclos anidados pueden recorrerse de 6, maneras diferentes:
  - kij
  - kji
  - ijk
  - ikj
  - jik
  - jki

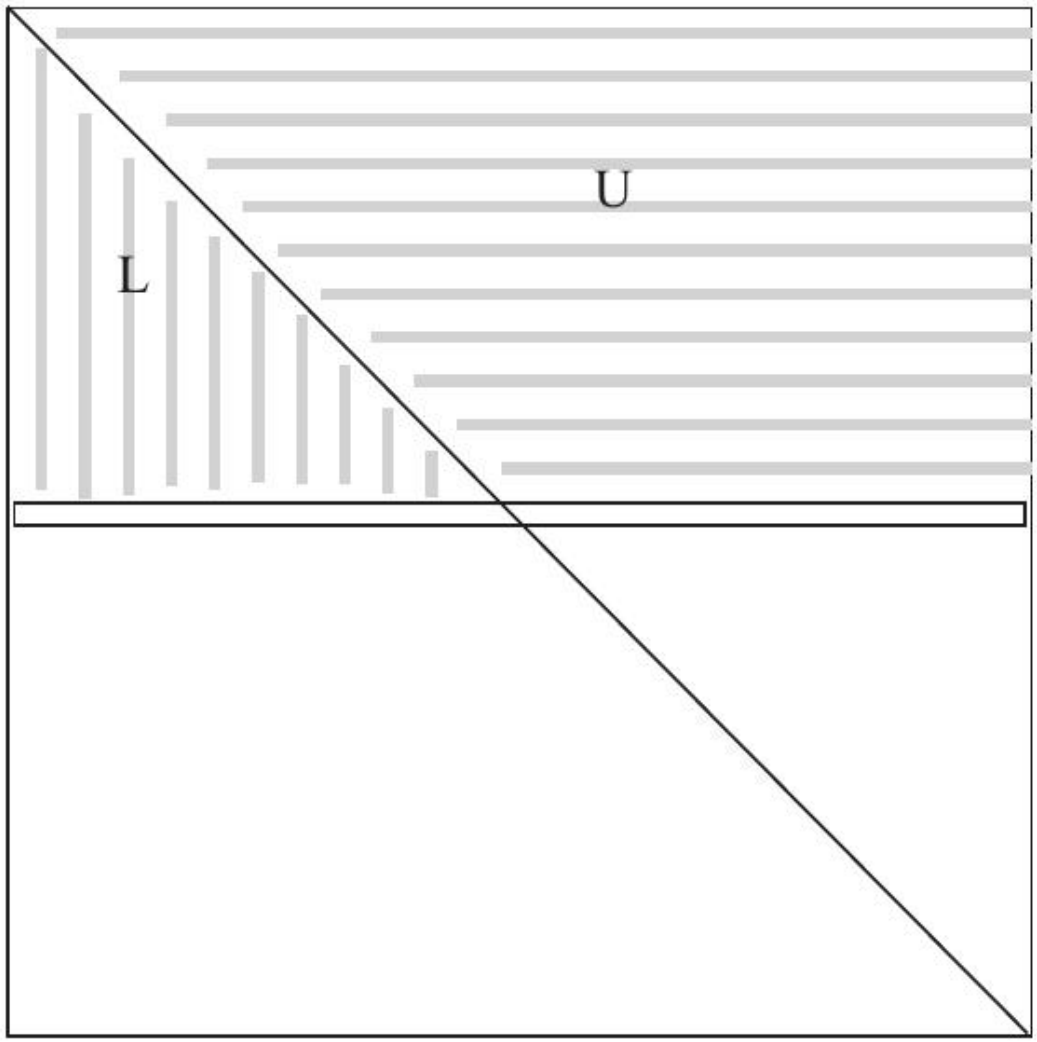
Factorización LU kij y kji



Factorización LU jik y jki

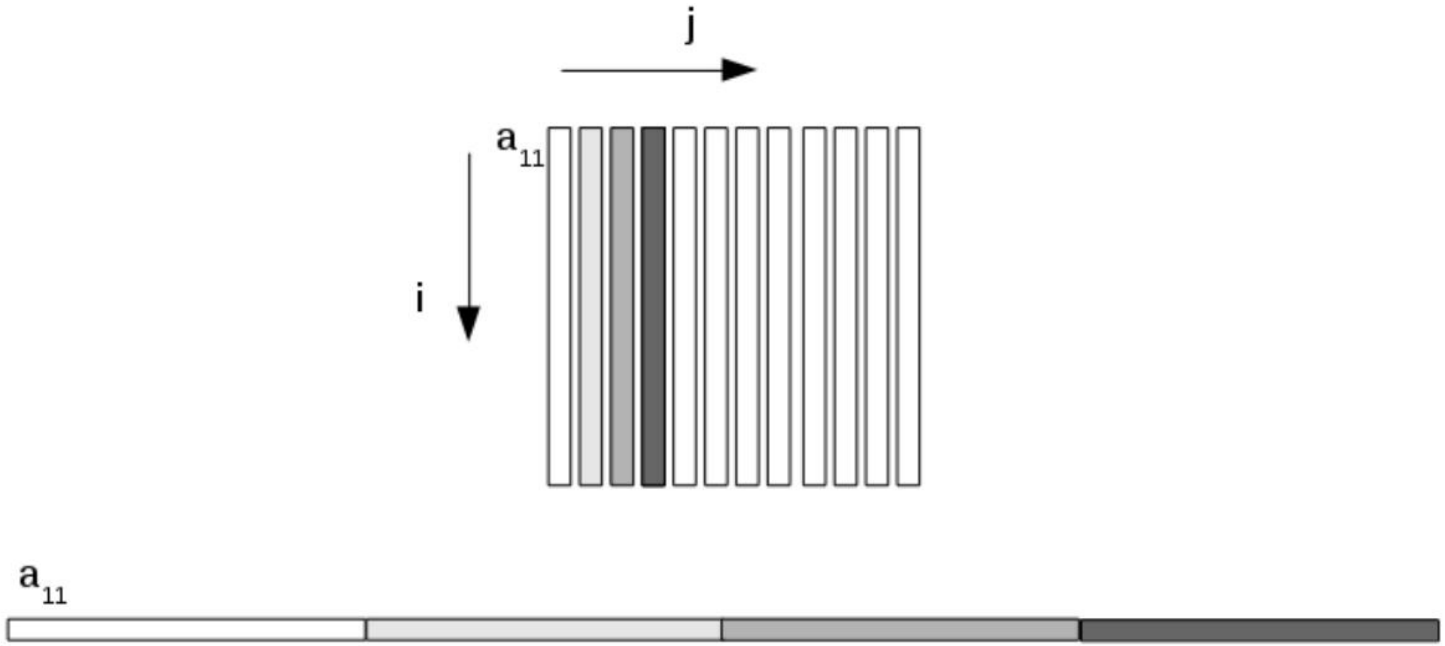


Factorización LU ikj y ijk



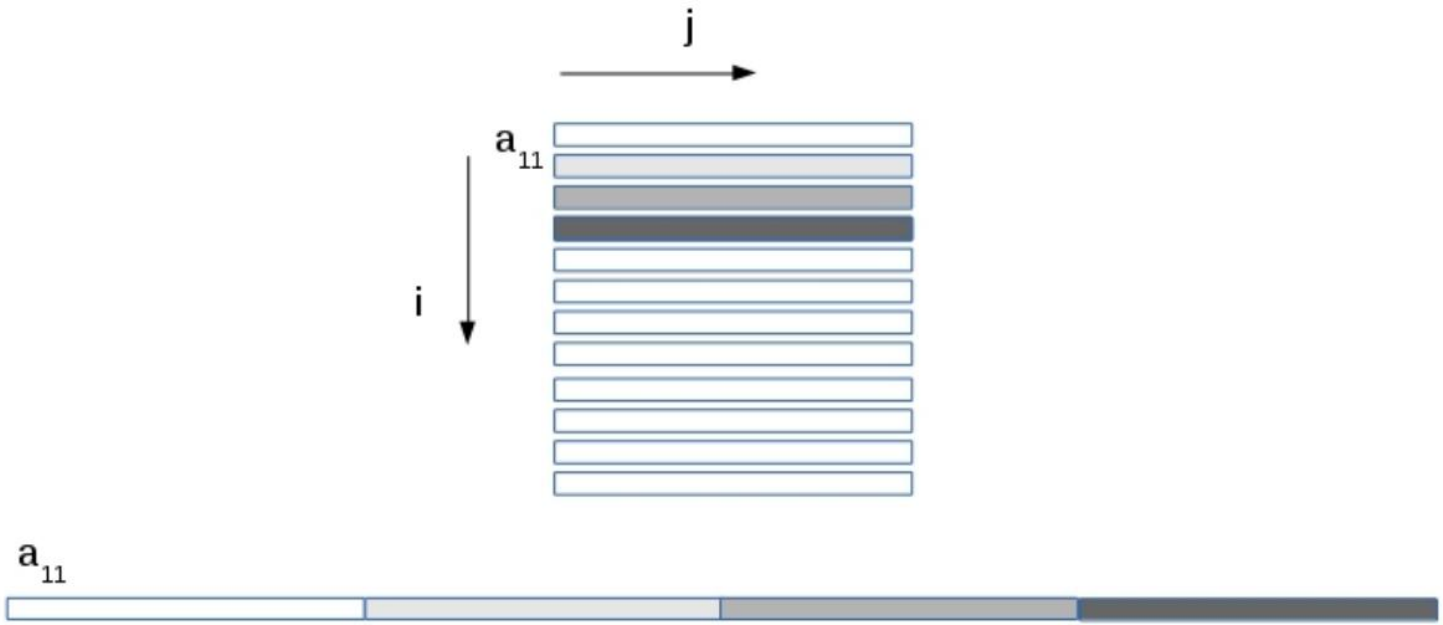
Factorización LU

- ¿Cuál es el interés de usar una u otra forma de recorrer los ciclos? Todo depende de la forma en que se almacenan los datos.
- En lenguaje FORTRAN las matrices están almacenadas por columnas. Esto significa que el arreglo que guarda la matriz contiene su primera columna, luego su segunda, y así sucesivamente.



Factorización LU

- En lenguaje C las matrices están almacenadas por filas. Esto significa que el arreglo que guarda la matriz contiene su primera fila, luego su segunda, y así sucesivamente.



- En Octave, están almacenadas por columnas.

Pivoteo

- La factorización descrita puede fallar si alguno de los pivotes (a_{kk}) es cero, o un número muy pequeño.
- Ejemplo:

  Sea el sistema:

$$\begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \qquad \text{con } \epsilon << 1$$

  aplicando el método de Gauss

$$\begin{bmatrix} \epsilon & 1 \\ 0 & 1 - \epsilon^{-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 - \epsilon^{-1} \end{bmatrix}$$

Pivoteo

y de allí puede obtenerse

$$x_2 = \frac{2 - \epsilon^{-1}}{1 - \epsilon^{-1}} \simeq 1$$

$$x_1 = (1 - x_2)\ \epsilon^{-1} \simeq 0$$

La solución correcta debería ser

$$x_1 = \frac{1}{1 - \epsilon} \simeq 1$$

$$x_2 = \frac{1 - 2\epsilon}{1 - \epsilon} \simeq 1$$

La solución numérica es exacta para x_2, pero no para x_1.
El problema se da cuando a_{kk}, de la diagonal, es pequeño frente a los otros coeficientes de la columna.
Si se intercambian filas, no hay error numérico y los resultados son correctos.

Pivoteo

- Este problema puede remediarse intercambiando las filas de la matriz para evitar que ese término (nulo o muy pequeño) quede en la diagonal.
- Hay distintas maneras de realizar esos cambios en la matriz. Se puede hacer *pivoteo total* o *parcial*
- El más sencillo de hacer es el pivoteo parcial, que se describe a continuación.
- En cada etapa se busca en la columna, por debajo del pivot, el elemento de mayor valor absoluto. Esa fila se intercambia con la actual.
- En realidad las filas no se intercambian físicamente. Se mantiene un vector $\mathbf{r}$ que indica el orden en que se ha realizado la factorización.
- En el pivoteo total (que no sera descripto aquí) la busqueda se hace no sólo sobre las filas, sino también sobre las columnas de la submatriz debajo del pivote.

Factorización kij con pivoteo parcial

for $i = 1, \ldots n$ do
$r_i = i$
end
for $k = 1, \ldots n-1$ do *fila pivotal*
buscar $p \in \{k, k+1, \ldots n\}$
tal que $|a_{r_p k}| = \max_{k \le i \le n} |a_{r_i k}|$
if $a_{r_p k} = 0 \rightarrow$ mensaje de error
if $r_p \neq r_k$
$z \leftarrow r_p$
$r_p \leftarrow r_k$
$r_k \leftarrow z$

for $i = k+1, \ldots n$ do
$s \leftarrow a_{r_i k} / a_{r_k k}$
$a_{r_i k} \leftarrow s$
for $j = k+1, \ldots n$ do
$a_{r_i j} \leftarrow a_{r_i j} - s\, a_{r_k j}$
end
end
end

Solución con pivoteo parcial

- El intercambio de filas equivale a afectar a la matriz con una matriz de permutación $\mathbf{P}$, que es el producto de las matrices de permutación de cada paso k.
- De modo que la factorización se ha hecho para

$$\mathbf{PA} = \mathbf{LU}$$

- El sistema a resolver es

$$\mathbf{PAx} = \mathbf{Pb}$$

$$\mathbf{LUx} = \mathbf{Pb}$$

- y puede escribirse:

$$\mathbf{Ly} = \mathbf{Pb}$$

$$\mathbf{Ux} = \mathbf{y}$$

Pivoteo parcial escalado

- A veces el pivoteo parcial no alcanza. Si el término de la diagonal es pequeño frente a los de su fila, no debería ser pivote.
- En el pivoteo parcial escalado se elige el mayor valor absoluto de los a_{ij} en cada fila

$$s_i = \max_{j=1,n} |a_{ij}|$$

la fila pivotal se elige:

se busca $p \in \{k, k+1, \ldots n\}$ tal que $\dfrac{|a_{r_p k}|}{s_{r_p}} = \max\limits_{k \leq i \leq n} \dfrac{|a_{r_i k}|}{s_{r_i}}$

Algoritmo fact. LU con pivoteo parcial escalado

Input: n, $\mathbf{A}$
Output: $\mathbf{L}$ y $\mathbf{U}$ (sobre $\mathbf{A}$)
for $i = 1, 2, \ldots n$ do
$s_i = \max_{i \le j \le n} |a_{ij}|$
$r_i = i$
end
for $k = 1, 2, \ldots n-1$ do *fila pivotal*
se busca $p \in \{k, k+1, \ldots n\}$ tal que $\frac{|a_{r_p k}|}{s_{r_p}} = \max_{k \le i \le n} \frac{|a_{r_i k}|}{s_{r_i}}$
si $a_{r_p k} = 0 \rightarrow$ mensaje de error y termina.
si $r_p \neq r_k$
$x \leftarrow r_p$
$r_p \leftarrow r_k$
$r_k \leftarrow x$
for $i = k+1, k+2, \ldots n$ do
$m \leftarrow a_{r_i k} / a_{r_k k}$
$a_{r_i k} \leftarrow m$
for $j = k+1, k+2, \ldots n$ do
$a_{r_i j} \leftarrow a_{r_i j} - m\, a_{r_k j}$
end
end
end

Factorización de Cholesky

- Hay muchas aplicaciones en que las matrices del sistema de ecuaciones son reales, simétricas y positiva definidas. Por ejemplo en problemas de mecánica de sólidos, en problemas de térmodinámica, o en problemas estructurales.
- En estos casos la descomposición LU puede adquirir una forma particular: la descomposición de Cholesky.
- Su justificación esta dada por el siguiente teorema:

  Teorema: Si $\mathbf{A}$ es matriz real, simétrica y definida positiva, entonces tiene una factorización *única* $\mathbf{A} = \mathbf{C}\mathbf{C}^{\mathbf{T}}$, donde $\mathbf{C}$ es matriz triangular inferior con diagonal positiva.

Factorización de Cholesky

Demostración

Como $\mathbf{A}$ es simétrica (o sea $\mathbf{A} = \mathbf{A^T}$):

$$\mathbf{A} = \mathbf{LU} = \mathbf{A^T} = \mathbf{U^T L^T}$$

$$\mathbf{L^{-1} LU} = \mathbf{L^{-1} U^T L^T}$$

$$\mathbf{U} = \mathbf{L^{-1} U^T L^T}$$

$$\mathbf{U L^{T^{-1}}} = \mathbf{L^{-1} U^T L^T L^{T^{-1}}}$$

$$\mathbf{U L^{T^{-1}}} = \mathbf{L^{-1} U^T}$$

El miembro izquierdo es una matriz triangular superior y el derecho una triangular inferior. Esto es así dado que se puede demostrar que la inversa de una matriz triangular es también triangular, y si la matriz tiene 1 en su diagonal, también los tiene su inversa.

Factorización de Cholesky

Demostración

Además el producto de dos matrices triangulares superiores es también una matriz triangular superior.
Así, deben ser ambos miembros matrices diagonales.

$$\mathbf{U}\mathbf{L}^{\mathbf{T}\,-\mathbf{1}} = \mathbf{L}^{-\mathbf{1}}\mathbf{U}^{\mathbf{T}} = \mathbf{D}$$

luego:

$$\mathbf{U} = \mathbf{D}\mathbf{L}^{\mathbf{T}}$$

y entonces

$$\mathbf{A} = \mathbf{L}\mathbf{U} = \mathbf{L}\mathbf{D}\mathbf{L}^{\mathbf{T}}$$

Puede demostrarse que $\mathbf{M}$ es una matriz definida positiva y $\mathbf{N}$ no es singular, si y solo is $\mathbf{N}\mathbf{M}\mathbf{N}^{\mathbf{T}}$ es definida positiva.

Factorización de Cholesky

Demostración

Como $\mathbf{A}$ es definida positiva también lo es $\mathbf{D}$, y por ser ésta diagonal sus términos son positivos.
Haciendo:

$$\mathbf{D} = \mathbf{D}^{\frac{1}{2}}\mathbf{D}^{\frac{1}{2}}$$

se puede escribir:

$$\boxed{\mathbf{A} = \mathbf{C}\mathbf{C}^{\mathbf{T}}}$$

donde $\mathbf{C} = \mathbf{L}\mathbf{D}^{\frac{1}{2}}$
Lo cual completa la demostración.

- La factorización de Cholesky es un caso particular de la factorización LU. En lugar de tener 1 en la diagonal, tiene $\sqrt{d_{ii}}$.

Factorización de Cholesky

Input: n, $\mathbf{A}$ (simétrica, definida positiva)
Output: $\mathbf{C}$ (sobreescrita en el triángulo inferior de $\mathbf{A}$)

$c_{11} \leftarrow \sqrt{a_{11}}$
for $i = 2, \ldots n$ do
$\quad c_{i1} \leftarrow a_{i1}/c_{11}$
end
for $i = 2, \ldots n-1$ do
$$c_{ii} \leftarrow \left(a_{ii} - \sum_{s=1}^{i-1} c_{is}^2 \right)^{\frac{1}{2}}$$
$\quad$ for $j = i+1, i+2, \ldots n$ do
$$c_{ji} \leftarrow \left(a_{ji} - \sum_{s=1}^{i-1} c_{js} c_{is} \right) / c_{ii}$$
$\quad$ end
end
$$c_{nn} \leftarrow \left(a_{nn} - \sum_{s=1}^{n-1} c_{ns}^2 \right)^{\frac{1}{2}}$$

Conteo de operaciones solución con descomposición LU

La factorización

- Para la primera fila pivotal se requiere una división, $(n-1)$ multiplicaciones y $(n-1)$ sumas, para una fila. Tradicionalmente se computan como flops. (flop = FLoating point OPerations) la multiplicación o división, ya que tardan más que la suma o resta. Así en esa primera fila se tienen n flops y eso se repite para $(n-1)$ filas o sea: $n(n-1)$. Es decir que en la primera fase de descomposición se realizan del orden de n^2 flops.
- A medida que avanza el proceso (la fila pivotal) se hacen las mismas cuentas con matrices cada vez menores. En total:

$$n^2+(n-1)^2+(n-2)^2+\ldots+2^2 = \frac{1}{3}n^3+\frac{1}{2}n^2+\frac{1}{6}n \simeq \frac{1}{3}n^3+\frac{1}{2}n^2 \text{ flops.}$$

(Aquí se usó el hecho de que: $\sum_{k=1}^{n} k^2 = \frac{1}{6}n(n+1)(2n+1)$ *)*

- Si n es grande, n^3 es dominante, y entonces la factorización LU requiere $\sim \frac{1}{3}n^3$ flops.

Conteo de operaciones solución con descomposición LU

La actualización del vector **b**

- Son $(n-1)$. En el primero hay $(n-1)$ flops; en el segundo $(n-2)$; en el tercero $(n-3)$; y así.
  Luego

$$(n-1)+(n-2)+\ldots+1 \;=\; \frac{1}{2}n^2-\frac{1}{2}n$$

(Aquí se usó el hecho de que: $\displaystyle\sum_{k=1}^{n} k=\frac{1}{2}n(n+1)$ *)*

La retrosustitución

- Hay 1 flop para calcular la incógnita x_n; 2 flops para calcular x_{n-1}; etc. Luego son:

$$1+2+3+\ldots+n \;=\; \frac{1}{2}n^2-\frac{1}{2}n$$

Conteo de operaciones solución con descomposición LU

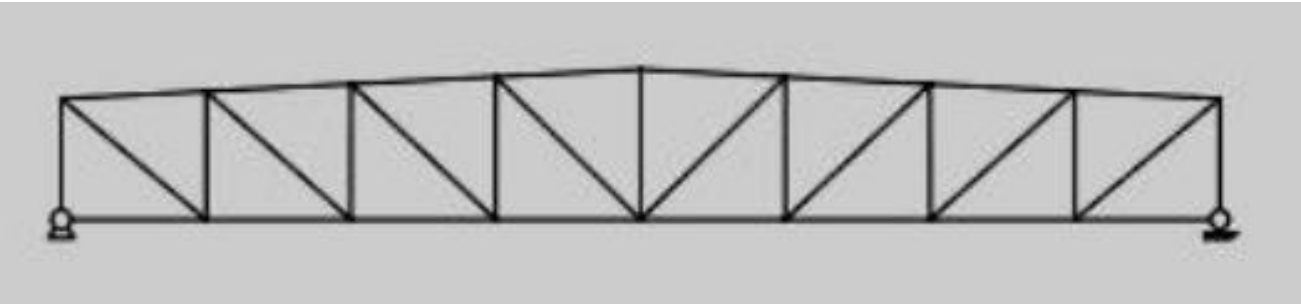
En resumen:

- para factorizar: $\frac{1}{3}n^3$
- para las 2 sustituciones: n^2

Por eso, si hay varios vectores **b**, conviene hacer la factorización de la matriz, una sola vez ($\frac{1}{3}n^3$ flops) y luego hacer varias veces las sustituciones (n^2 flops).

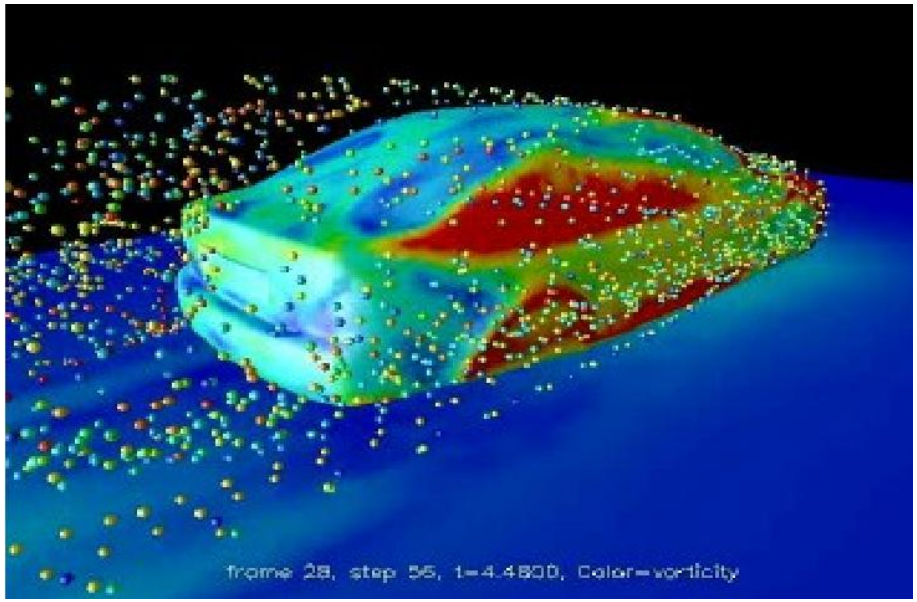
¿Qué métodos usar?

- Los sistemas de ecuaciones que se precisan resolver en la práctica son de muy variados tamaños.
- Ya hemos visto en los ejemplos sencillos mencionados puede haber sistemas tan pequeñós como aquellos de dos incógnitas.
- Una estructura reticulada plana posee una incógnita por barra (el esfuerzo de compresión o tracción en la barra). Por tanto, una como la de la figura conduce a un sistema de algunas decenas de incógnitas.



¿Qué métodos usar?

- Pero problemas como los de mecánica de fluidos tridimensional tiene una cantidad de incógnitas que puede ser de millones.



¿Qué métodos usar?

- Ahora bien, un método directo requiere del orden de $\frac{n^3}{3} + 2n^2$ operaciones de punto flotante (*flop*) para su resolución.
- O sea, un sistema de 100 incognitas precisa unas 353.000 flop; uno de 1000 incógnitas, unos 335*10^6; uno de 10.000 incógnitas, del orden de 3,33*10^{11}, flop.
- Si suponemos que un procesador tiene una velocidad sostenida de resolución del orden de 1000 MFLOPS (millones de operaciones de punto flotante por segundo), para resolver esos sistemas de ecuaciones con matrices llenas (generales) con 100, 1000 o 10.000 incognitas, tardará –respectivamente– 3,53*10^{-5} segundos; 3,35*10^{-2} segundos; y 33,35 segundos.
- Ya un sistema lleno de 100.000 incógnitas requeriría, en ese procesador, unas 100 horas de cálculo.

¿Qué métodos usar?

- Esta constatación hace que los métodos directos sean inviables para grandes sistemas de ecuaciones.
- En el capítulo siguiente veremos otros métodos, denominados *iterativos*, que se pueden usar para grandes sistemas.
- Los métodos directos son buenos para sistemas que no sean muy grandes.
- La factorizacion LU es interesante en problemas donde hay que resolver varias veces la ecuación cambiando el vector de términos independientes (por ejemplo, en la estructura reticulada, para diferentes estados de carga). En ese caso, la tarea más cara (la factorización) se hace una sola vez, y para cada estado de carga se hacen "solamente" las sustituciones hacia atrás y adelante, que tiene un costo del orden de n^2.

¿Qué métodos usar?

- En realidad, en los ejemplos que presentamos hicimos una pequeña trampa. El conteo de operaciones, que hicimos, es para resolver un sistema con una matriz general (llena).
- En muchos casos prácticos (como el del reticulado o el de fluidos) las matrices no son llenas, sino que tienen una estructura de *banda*. Es decir hay una banda alrededor de la diagonal principal donde estan los términos no nulos, y fuera de ella los terminos son todos nulos. Así, si b es el ancho de banda, la cantidad de operaciones, en lugar de ser, por ejemplo n^3 es del orden de $b\ n^2$.
- En problemas como el del reticulado, ademas las matrices son simétricas.

¿Qué métodos usar?

- Hay otros problemas donde las matrices son *ralas*, es decir poblada de ceros con pocos elementos no nulos. O bien otros con matrices tridiagonales o pentadiagonales. En todos estos casos hay métodos apropiados que resultan más convenientes.
- Pero hay algunos problemas en que las matrices sí son llenas, y allí cabe el análisis realizado.

Normas, radio espectral y número de condición

Normas, radio espectral y número de condición

- Veremos aquí algunas medidas que se pueden hacer sobre una matriz, tales como **norma, radio espectral** y **número de condición**.
- Ellas permiten evaluar el comportamiento de la matriz en la solución de un SEAL, cuan fácil o rápido será la solución, o cuan estables serán los resultados frente a errores en los datos.

Normas Vectoriales

- Una *norma* para un espacio vectorial V es una función que, aplicada a un vector, da un escalar no nulo, tal que:
  1) $\|\mathbf{x}\| > 0 \qquad$ si $\mathbf{x} \neq 0, \mathbf{x} \in V$
  2) $\|\lambda\mathbf{x}\| = |\lambda|\|\mathbf{x}\| \qquad$ para $\lambda \in \mathbb{R}, \mathbf{x} \in V$
  3) $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| \qquad$ si $\mathbf{x}, \mathbf{y} \in V$

  (Con doble raya se designa la norma. Así $\|\mathbf{x}\|$ se lee *norma de* $\mathbf{x}$)

- Algunas normas para vectores:
  - Norma Euclídea

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}}$$

  - Norma Infinito

$$\|\mathbf{x}\|_\infty = \max_{i=1} |x_i|$$

  - Norma L_1

$$\|\mathbf{x}\|_1 = \sum_{i=1}^{n} |x_i|$$

Normas Vectoriales

- Para un vector en $\mathbb{R}^2$:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

  la norma euclidea es el módulo del vector.

- Por ejemplo para un vector $\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, será:

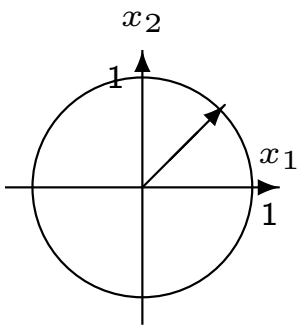
$$\|\mathbf{x}\|_2 = \sqrt{2}$$

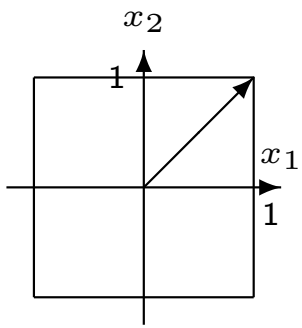
$$\|\mathbf{x}\|_\infty = 1$$

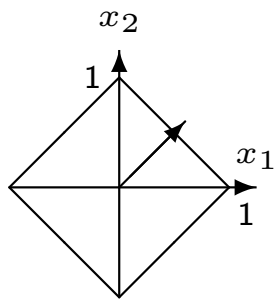
$$\|\mathbf{x}\|_1 = 2$$

Normas Vectoriales

- Todos los vectores en $\mathbb{R}^2$ de $\|\mathbf{x}\|_2 = 1$ tienen su extremo en un círculo (figura de la izq.).
- Todos los vectores en $\mathbb{R}^2$ de $\|\mathbf{x}\|_\infty = 1$ tienen su extremo en un cuadrado de lado 2 (figura central).
- Todos los vectores en $\mathbb{R}^2$ de $\|\mathbf{x}\|_1 = 1$ tienen su extremo en un cuadrado de lado $\sqrt{2}$ (figura de la derecha).







Normas Matriciales

- Una norma para matrices, debe cumplir las tres condiciones mencionadas.
- Si bien puede definirse de distintas formas, se la suele definir asociada a una definición de norma vectorial; Estas normas se denominan **naturales**, o **inducidas**, o **subordinadas**, o **asociadas** a una norma vectorial. Y se define como:

$$\|\mathbf{A}\| = sup\{ \|\mathbf{Au}\| \ : \mathbf{u} \in \mathbb{R}^n, \|\mathbf{u}\| = 1\}$$

- Puede verificarse que esta definición de norma natural cumple las tres condiciones pedidas.

Normas Matriciales

- Además, de la definición surge una importante consecuencia:

$$\|\mathbf{Ax}\| \leq \|\mathbf{A}\|\|\mathbf{x}\|$$

Demostración:
Para $\mathbf{x} = \mathbf{0}$, *se verifica.*

Para $\mathbf{x} \neq \mathbf{0}$, $\mathbf{v} = \mathbf{x}/\|\mathbf{x}\|, \text{es tal que} \|\mathbf{v}\| = 1$
De la definición de norma matricial:

$$\|\mathbf{A}\| \geq \|\mathbf{Av}\| = \|\frac{1}{\|\mathbf{x}\|}\mathbf{Ax}\| = \frac{1}{\|\mathbf{x}\|}\|\mathbf{Ax}\|$$

Luego

$$\|\mathbf{Ax}\| \leq \|\mathbf{A}\|\|\mathbf{x}\| \qquad \mathbf{x} \in \mathbb{R}^n$$

Lo que completa la demostración.

Normas Matriciales

- Además de las condiciones (1) a (3), la norma matricial subordinada verifica:
  - $\|\mathbf{I}\| = 1$ ($\mathbf{I}$ matriz identidad)
  - $\|\mathbf{AB}\| \leq \|\mathbf{A}\|\|\mathbf{B}\|$
- Ejemplo:
  Para la norma vectorial $\|.\|_\infty$:

$$\|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

la norma matricial subordinada es:

$$\|\mathbf{A}\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}|$$

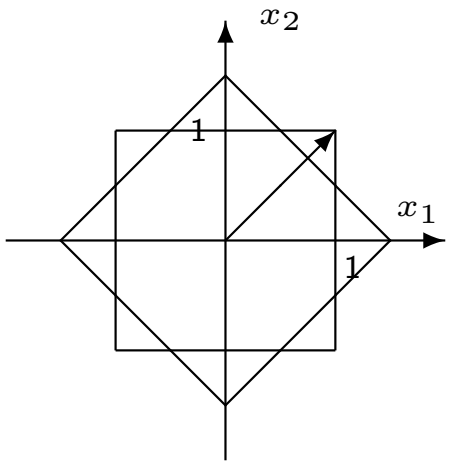
Normas Matriciales

Otros ejemplos:

- Una transformación de un vector puede mirarse como una matriz.
- La matriz identidad $\mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ mantiene inalterado al vector.
- Una matriz $\mathbf{A} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$ por ejemplo cuando multiplicada por un vector, duplica las componentes según x_1. O sea lo *estira* al doble en el sentido de ese eje.
- Una matriz $\mathbf{A} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$ cuando multiplicada por un vector, lo hace rotar un ángulo α.

Normas Matriciales

- Supóngase una matriz de rotación con $\alpha = 45^o$:
  $\mathbf{A} = \begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$
- La norma infinito es: $\|\mathbf{A}\|_\infty = \max\limits_{1 \le i \le n} \sum\limits_{j=1}^{n} |a_{ij}| = \sqrt{2}$
- Todos los vectores de norma inf. unitaria estan sobre un cuadrado. Al rotar 45^o el de mayor norma inf. es $\sqrt{2}$



Radio espectral

- Se define como *radio espectral* de una matriz $\mathbf{A}$:

$$\rho(\mathbf{A}) = \max\{|\lambda| \quad | \quad det(\mathbf{A} - \lambda\mathbf{I}) = 0\}$$

  - λ son los autovalores de $\mathbf{A}$, soluciones de la ecuación:

$$det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

    Una matriz de $n \times n$ tiene n autovalores (algunos pueden ser repetidos; los autovalores pueden ser reales o complejos)

- $\rho(\mathbf{A})$ es el módulo del mayor autovalor; o sea el radio del menor círculo en el plano complejo, que contiene a todos los autovalores.

Radio Espectral

- Se puede demostrar que:
  - $[\rho(\mathbf{A}^T\mathbf{A})]^{1/2} = \|\mathbf{A}\|_2$
  - $\rho(\mathbf{A}) \leq \|\mathbf{A}\|$ para toda norma de $\mathbf{A}$
- En particular, si $\mathbf{A}$ es simétrica:

$$\rho(\mathbf{A}) = \|\mathbf{A}\|_2$$

- Si el $\rho(\mathbf{A}) < 1$ la matriz se llama **convergente** y ello equivale a

$$\lim_{k\to\infty} (a_{ij})^k = 0$$

para cada elemento a_{ij} de la matriz. Esto será útil para estudiar la convergencia de procedimientos iterativos, en el próximo capítulo.

Número de condición

- Sea $\mathbf{Ax} = \mathbf{b}$. Supóngase que existe $\mathbf{A^{-1}}$. Luego

$$\mathbf{x} = \mathbf{A^{-1}b}$$

- Considérese el caso en que se perturba el vector $\mathbf{b}$. En su lugar se tiene $\tilde{\mathbf{b}}$. La solución será:

$$\tilde{\mathbf{x}} = \mathbf{A^{-1}}\tilde{\mathbf{b}}$$

El error absoluto es: $\mathbf{e} = \mathbf{x} - \tilde{\mathbf{x}}$
La norma del error absoluto:

$$\|\mathbf{e}\| = \|\mathbf{x}-\tilde{\mathbf{x}}\| = \|\mathbf{A^{-1}b}-\mathbf{A^{-1}}\tilde{\mathbf{b}}\| = \|\mathbf{A^{-1}}(\mathbf{b}-\tilde{\mathbf{b}})\| \leq \|\mathbf{A^{-1}}\|\|\mathbf{b}-\tilde{\mathbf{b}}\|$$

El error relativo

$$\frac{\|\mathbf{x}-\tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A^{-1}}\|\|(\mathbf{b}-\tilde{\mathbf{b}})\|\frac{\|\mathbf{Ax}\|}{\|\mathbf{b}\|}\frac{\mathbf{1}}{\|\mathbf{x}\|} \leq \|\mathbf{A^{-1}}\|\|\mathbf{A}\|\frac{\|(\mathbf{b}-\tilde{\mathbf{b}})\|}{\|\mathbf{b}\|}\frac{\|\mathbf{x}\|}{\|\mathbf{x}\|}$$

Número de condición

- Esta expresión puede escribirse:

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \leq \kappa(\mathbf{A}) \frac{\|(\mathbf{b} - \tilde{\mathbf{b}})\|}{\|\mathbf{b}\|}$$

- Donde se ha definido el **número de condición**:

$$\kappa(\mathbf{A}) = \|\mathbf{A}^{-1}\| \|\mathbf{A}\|$$

- Si el número de condición es pequeño, una pequeña perturbación en los datos $(\mathbf{b})$ produce errores relativos pequeños en la solución.
- Si el número de condición es grande el error en la solución es grande.

Número de condición

- Ejemplo:

$$\textit{Sea } \mathbf{A} = \begin{bmatrix} 1 & 1+\epsilon \\ 1-\epsilon & 1 \end{bmatrix} \textit{ con } \epsilon > 0.$$

$$\mathbf{A}^{-1} = \epsilon^{-2} \begin{bmatrix} 1 & -1-\epsilon \\ -1+\epsilon & 1 \end{bmatrix}$$

Usando $\|.\|_\infty$*:*

$$\|\mathbf{A}\|_\infty = 2+\epsilon \qquad \text{y} \qquad \|\mathbf{A}^{-1}\|_\infty = (2+\epsilon)\epsilon^{-2}$$

$$\kappa(\mathbf{A}) = \frac{(2+\epsilon)^2}{\epsilon^2} > \frac{4}{\epsilon^2}$$

Si $\epsilon = 0.01 \rightarrow \kappa(\mathbf{A}) > 4 \times 10^4$

Un error en los datos produce errores 40.000 veces mayor en la solución.

Número de condición

- Una solución aproximada tiene un error: $\mathbf{e} = \mathbf{x} - \tilde{\mathbf{x}}$
- De la ecuación se obtiene $\mathbf{Ax} - \mathbf{b} = \mathbf{0}$, pero si se reemplaza $\mathbf{x}$ por $\tilde{\mathbf{x}}$ se tiene $\mathbf{A}\tilde{\mathbf{x}} - \mathbf{b} = \mathbf{r}$
- Allí $\mathbf{r}$ es el residuo, definido también como: $\mathbf{r} = \mathbf{b} - \tilde{\mathbf{b}}$
- Restando $\mathbf{Ax} - \mathbf{b} = \mathbf{0}$ y $\mathbf{A}\tilde{\mathbf{x}} - \tilde{\mathbf{b}} = \mathbf{0}$ se obtiene:

$$\mathbf{Ae} = \mathbf{r}$$

- Los errores relativos de los datos y la solución están asociados por la misma relación funcional que los datos y la solución.
- Puede verse que

$$\|\mathbf{r}\|\|\mathbf{x}\| = \|\mathbf{Ae}\|\|\mathbf{A^{-1}b}\| \leq \|\mathbf{A}\|\|\mathbf{A^{-1}}\|\|\mathbf{e}\|\|\mathbf{b}\|$$

De donde

$$\frac{1}{\kappa(\mathbf{A})}\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} \leq \frac{\|\mathbf{e}\|}{\|\mathbf{x}\|}$$

Número de condición

- Juntando esto con la expresión obtenida en transparencias anteriores se obtienen cotas superiores e inferiores para el error en la solución:

$$\frac{1}{\kappa(\mathbf{A})}\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} \leq \frac{\|\mathbf{e}\|}{\|\mathbf{x}\|} \leq \kappa(\mathbf{A})\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}$$

- Si $\kappa(\mathbf{A})$ pequeño (o moderado) $\rightarrow$ *matriz bien condicionada*
- Si $\kappa(\mathbf{A})$ grande $\rightarrow$ *matriz mal condicionada*

Resumen

En este capítulo hemos visto:

- Los métodos *directos* de resolución de SEAL
- En particular:
  - el método de eliminación de Gauss
  - factorización (o descomposición) $\mathbf{LU}$
  - una particularización de $\mathbf{LU}$ para matrices simétricas: el método de Cholesky
- Hicimos un conteo de operaciones de estos métodos observando que el número de operaciones requeridas son de $O(n^3)$ lo que los hace inviables para sistemas muy grandes.
- Vimos luego como calcular *normas* de vectores y matrices, *radio espectral* y *número de condición* de una matriz y la implicancia de este último en la solución del sistema.