CONTEO OPERACIONES

Factorización LU: Esun analisis importante dentro de la complejidad computacional de los metodos numéricos.—

1) Factorización LU

Dado $\underline{\underline{A}}\,\underline{x} = \underline{b}$ donde A: matriz $n \times n$.

la factorización descompone A en $A = \underline{\underline{L}}\,\underline{\underline{U}}$.

$\underline{\underline{L}}$: matriz triangular inferior con 1 en la diagonal. y en el resto de resdementos. no mas los valores de los multiplicadores "m" del proceso de eliminación Gaussiana..

$\underline{\underline{U}}$: matriz triangular superior, que surge del proceso de eliminación.—

El algoritmo de eliminación Gaussiano consta de dos tipos de operaciones fundamentales

1) Calculo de los multiplicadores que podemos llamar "m" o bien l_{ij} asociados con la matriz L. $\Rightarrow$ $\boxed{l_{ij} = a_{ij}/a_{jj}}$ → pivot

Como descartamos la primera fila tenemos desde $j = 1$ hasta $(n-1)$ divisiones $\Rightarrow$ Es decir contamos los multiplicadores que calculamos debajo del pivot: $\Rightarrow$.

$$\textcircled{*} \quad \boxed{\sum_{j=1}^{(n-1)} (n-j) = \frac{n(n-1)}{2}} \text{ operaciones}$$

2) Actualizamos La matriz: Cada vez que calculamos un multiplicador "m" o l_{ij} actualizamos la submatriz restante con la siguiente operación: $\boxed{a_{ik} = a_{ik} - l_{ij}\, a_{jk}. \quad \forall k = j+1, j+2, \ldots, n.}$

$\Rightarrow \quad a_{ik} = a_{ik} - l_{ij} a_{jk} \quad \forall k = j+1, j+2, \dots n.$

es decir cada pivot j, tenemos que actualizar los valores me afectan los filas i, tal que.

$$i = j+1, j+2, \dots n.$$

y los columnas $k = j+1, j+2 \dots n.$

$\Rightarrow$ el conteo de operaciones será un conjunto de multiplicaciones y sumas.:

$$\sum_{j=1}^{n-1} \sum_{i=j+1}^{n} \sum_{k=j+1}^{n} 2 \simeq \frac{2}{3} n^3.$$

sumando ambos conteos tenemos:

$$\frac{2}{3} n^3 + \frac{n(n-1)}{2} \simeq O(n^3)$$

$\Rightarrow$ la factorización LU tiene como complejidad computacional del $O(n^3)$

———o———

REGLAS BÁSICAS DEL CONTEO DE OPERACIONES

Las reglas matemáticas básicas para el conteo de operaciones se basan en propiedades de SUMATORIAS y SERIES. Algunas de las fundamentales son:

1) SUMATORIA DE UNA CONSTANTE:

(I) $\sum_{j=1}^{n} C = C \cdot n$. Ejemplo: $\sum_{j=1}^{n} 1 = n$.

2) SUMATORIA DE UNA VARIABLE LINEAL

(II) $\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$ Ejemplo: $\sum_{j=1}^{5} j = 1+2+3+4+5 = 15$.

Deducción: vamos a calcular (II) y la aplicamos a $\sum_{i=1}^{n-1}(n-i)$ que se usa en el conteo.—

Pero demostrado $S = \sum_{j=1}^{n} j = \frac{n(n+1)}{2}$ (III)

usamos inducción matemática $\Rightarrow$.

reponemos $n = 1$. $\Rightarrow$.

$\sum_{j=1}^{n=1} j = 1$ (Revisa la formula III $S = \frac{1(2)}{2} = 1$ se cumple.)

Para inductivo:

si $\sum_{j=1}^{n}(j) = \frac{n(n+1)}{2}$ probamos para una cantidad $n+1$.

es decir $\sum_{j=1}^{n+1} j = \left(\sum_{j=1}^{n} j\right) + (n+1) = \frac{n(n+1)}{2} + (n+1)$

factorizamos $(n+1) \Rightarrow \frac{n(n+1)}{2} + \frac{2(n+1)}{2} = \frac{(n+1)(n+2)}{2}$

y si aplicamos la formula para $n+1$ $\Rightarrow$.

$S = \frac{(n+1)(n+2)}{2}$ y coincide. y la demostración por inducción está COMPLETA.

En nuestro conteo tenemos:

$\sum_{j=1}^{(n-1)} (n-i)$ => lo reescribimos como:

$\sum_{j=1}^{n-1} (n-i) = \sum_{j=1}^{n-1} n - \sum_{j=1}^{n-1} i$, esto es descomponemos en 2 sumatorias =>

$n \underbrace{\sum_{j=1}^{n-1} 1}_{(n-1)} - \underbrace{\sum_{j=1}^{n-1} i}_{\frac{n(n-1)}{2}} = n.(n-1) - \frac{n(n-1)}{2}$

Factorizamos => $= \frac{2n(n-1) - n(n-1)}{2}$

$= n(n-1)\left(1 - \frac{1}{2}\right) = \frac{n(n-1)}{2}$ =>.

$$\boxed{\sum_{i=1}^{n-1} (n-i) = \frac{(n-1)n}{2}}$$

RESUMEN.

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$$

$$\sum_{j=1}^{n-1} (n-j) = \frac{(n-1)n}{2}$$

} Ambas son de $O(n^2)$

— x —

FORMULAS FUNDAMENTALES EN EL CONTEO

1) $\sum_{j=1}^{n} c = c.n$

2) $\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$

3) $\sum_{j=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}$.

Ejemplo $\sum_{j=1}^{3} j^2 = 1^2 + 2^2 + 3^2 = 14$

$\boxed{\text{si } n=3 \quad S = \frac{3(4)(7)}{6} = 14.}$

4) $\sum_{j=1}^{n} j^3 = \left(\frac{n(n+1)}{2}\right)^2$

Ejemplo $\sum_{j=1}^{n=3} j^3 = 1 + 8 + 27 = 36.$

5) $\sum_{j=1}^{n} r^j = \frac{r^{n+1} - 1}{r-1}$; si $r \neq 1$.

Ejemplo $\sum_{j=0}^{3} 2^j = 1 + 2 + 4 + 8 = 15.$

MEMORIA CONSUMIDA

Operación: Eliminación Gaussiana con sustitución hacia atrás $\underline{A}\underline{x} = \underline{b}$

Sistema: $n \times n$.

Consideramos: Almacenamiento de A
" de b
Cálculos Intermedios

1) Memoria Requerida (Almacenamiento)

Depende en que se programó (Ej. Octave o MATLAB)

Números de punto flotante de doble precisión ocupan 8 bytes por número $\Rightarrow$ $n = 100$

Matriz $A (n \times n)$

Almacenamiento $= n^2$

Memoria Requerida $_A$: $n^2 \times 8$ bytes

Vector $b (n \times 1)$

Almacenamiento $= n$

Memoria $_b$: $n \times 8$ bytes

Si $n = 100$ Matriz $= 100^2 \times 8 = 80\,000$ bytes $= 0.08$ MB

Vector $= 100 \times 8 = 800$ bytes $= 0.0008$ MB

2) MEMORIA ADICIONAL (proceso de eliminación Gaussiana)

En este proceso la matriz se transforma en una matriz triangular superior y se almacena en el mismo espacio. Consideramos: USAMOS PIVOTEO PARCIAL

- Vector de pivoteo (parcial) $= n \times 8$ bytes.
- Vector de solución x: $n \times 8$ bytes

si $n = 100 \Rightarrow$ el total nos dá: 0.0016 MB (extra)

3) CONSUMO DE MEMORIA (por operaciones)

Cada operación tiene un costo en memoria y CPU

Para este proceso de eliminación con sustitución hacia atrás tenemos.

1) Multiplicaciones y divisiones (× /)
Cada operación de punto flotante usa 8 bytes para los operandos y el resultado

2) Sumas y Restas (+ −)
Igual al anterior 8 bytes (/operación)

3) Acceso a Memoria
Leer y escribir matrices/vectores constante memoria

No tenemos en cuenta el acceso a memoria

⇒ Eliminación Gaussiana: $\frac{2}{3}n^3$ FLOPs.
Sustitución hacia atrás: $\frac{n^2}{2}$ FLOPs.

si $n=100$ ⇒ Elim. Gaussiana = $\frac{2}{3}100^3$
Sust hacia atrás = $\frac{1}{2}n^2 = \frac{100^2}{2}$

⇒ $\frac{2}{3}100^3 + \frac{1}{2}100^2 = 666.666.67 + 5000 = 671.666.67$ operaciones.

Cada operación necesita ≈ 24 bytes (2 operandos + resultado)

Memoria Temporal: $671666,67 \times 24 \approx$ 16 MB.

⇒ Reservar:

	Componente	Memoria (MB)
$Ax=b$	Matriz A	0.08
	Vector b	0.0008.
	Vector x	0.0008.
	Vector pivoteo	0.0008.
	Memoria oper. extras	≈ 16.

Total estimado: ≈ 16.08 MB.

Ahora calculamos el tiempo:

TIEMPO CONSUMIDO POR CADA OPERACION.

Tiempo en nanosegundos (ns)
depende de la:
- ARQUITECTURA DEL PROCESADOR
- MEMORIA CACHE
- OPTIMIZACION DEL COMPILADOR

Hagamos un calculo con los procesadores o ticoles =>.

TIEMPO DE REFERENCIA.

Procesador (Intel Core i7/i9, AMD Ryzen, Xeon)
Operacion de doble precisión (64 bits, 8 byte)

Operación	Tiempo estimado
Suma/Resta (+ -)	~0.5 - 1.5 ns.
Multi/División (× ÷)	~ 1 - 3 ns.
División (/)	~ 5 - 30 ns.
Acceso a RAM.	~ 100 - 200 ns.
" a Cache L1.	~ 0.5 - 1 ns.
" a Cache L2	~ 3 - 10 ns.

Vamos a nuestro caso de Eliminación Gaussiana

$\frac{2}{3}n^3$ FLOPs. (operaciones) Elim.
$\frac{1}{2}n^2$ FLOPs. (") Sustitución

Si $n=100$ => Como vimos: 671 666,67 FLOPs.
Suponemos que c/ operación tarda en promedio 2ns
=> 671666,67 x 2 ns = 1.34 ms.

Si hay operaciones costosas en un proceso como. división el tiempo es mayor. ~ 5-10 ns.
Con la opción TIC y TOC lo podemos determinar en OCTAVE o MATLAB _

Ejercicio N°TP02 (2)

A matriz cuadrada de orden n

Simétrica definida positiva (S.d.p)

$\Rightarrow A = A^T$

Definida Positiva $\forall x \neq 0 \Rightarrow x^T A x > 0$

es decir la forma cuadrática asociada a A siempre es positiva para cualquier $x \neq 0$

otra forma equivalente:

si sus valores propios son positivos $\Rightarrow$.

$\lambda_i > 0 \quad \forall i = 1, 2, \ldots n.$

$\Rightarrow$ A es S.d.p.

Como demostramos que es no singular

Dado A S.d.p. $\Rightarrow$ es no singular y $\det(A) \neq 0$

Contradicción Supongamos A SINGULAR

$\Rightarrow Ax = 0$ con $x \neq 0$

multiplicamos por $x^T A x = \underset{\text{prod. escalar}}{x^T \cdot 0} = 0$

$\Rightarrow x^T A x = 0$ contradice la definición de def. positivo $x^T A x > 0$

Supusimos que A era singular y $x \neq 0$ como se contradice $\Rightarrow$ A es NO SINGULAR y $Ax = 0$ tiene como única solución $x = 0$

$\Rightarrow \boxed{\det(A) \neq 0}$ y se puede invertir