Práctica 3: Método de eliminación de Gauss e factorización A=LU. Estratexia de pivote parcial: factorización PA=LU

Nesta práctica debes implementar en **FORTRAN 90** a **resolución dun sistema lineal:**

$$Ax = b,\ A \text{ matriz non singular de orde } n,\ b \in \mathbb{R}^n,$$

utilizando sucesivamente os **métodos de: eliminación de Gauss, factorización A=LU e factorización PA=LU.**

1. Comezamos co **método de eliminación Gauss.**

   1.1. Escribe as **subrutinas:**

   - `datsis(n,a,b)`: **lectura e escritura dos datos** matriz do sistema `a` e termo independente `b`.
   - `gauss(n,a,b,deter)`: **eliminación de Gauss** que, en $n-1$ etapas, permite transformar o sistema de partida noutro equivalente con matriz triangular superior; a subrutina debe calcular o **determinante da matriz do sistema.** Podes utilizar o código:

```
!inicializacion do determinante
deter=1.
!etapa k-esima da eliminacion
do k=1,n-1
  piv=a(k,k)
  !comprobacion de que o
  !k-esimo pivote non e nulo
  if(abs(piv)<1.e-12) then
    print*,'pivote nulo na etapa: ',k
    stop
  end if
  !actualizacion do determinante
  deter=deter*piv
  !eliminacion
  do i=k+1,n
    factor=a(i,k)/piv
      do j=k+1,n
        a(i,j)=a(i,j)-factor*a(k,j)
      end do
      b(i)=b(i)-factor*b(k)
    end do
end do
!comprobacion de que o
!ultimo pivote non e nulo
if(abs(a(n,n))<1.e-12) then
  print*,'pivote nulo na etapa: ',n
  stop
end if
!remate do calculo do determinante
deter=deter*a(n,n)
```

   Nota: O FORTRAN 90 permite sustituir o doble bucle `do` da eliminación polo seguinte código, menos custoso en tempo de cálculo:

```
a(k+1:n,k)=a(k+1:n,k)/piv
  do j=k+1,n
    a(k+1:n,j)=a(k+1:n,j)-a(k+1:n,k)*a(k,j)
  end do
b(k+1:n)=b(k+1:n)-a(k+1:n,k)*b(k)
```

   1.2. Escribe o **programa principal** que **lea a orde do sistema,** reserve memoria para tódolos arreglos que interveñen e, despois:

   - Chame á subrutina de lectura e escritura dos datos do sistema.
   - Garde a matriz `a` e o termo independente `b` en novas variables `aa` e `bb`, co fin de calcular posteriormente o residuo do sistema.
   - Chame á subrutina que efectúa a eliminación e calcula o determinante da matriz.
   - Chame á subrutina que calcula a solución do sistema triangular superior.
   - Chame á subrutina que calcula o residuo da solución.

   1.3. Comproba o bo funcionamento dos programas escritos con distintos exemplos.

2. Neste novo exercicio trátase de utilizar o **método de factorización** $A = LU$ tendo en conta que:

$$Ax = b \Longleftrightarrow LUx = b \Longleftrightarrow \begin{cases} Ly = b, \\ Ux = y, \end{cases}$$

2.1. Escribe as novas **subrutinas:**

- `lu(n,a,deter)`: obtención da **factorización A=LU,** a partir da eliminación de Gauss, e do determinante da matriz do sistema. U, matriz triangular superior, debe almacenarse na parte correspondente da matriz A e L, matriz triangular inferior, tamén debe almacenarse en A excluíndo a diagonal. As fórmulas da factorización son análogas ás da eliminación coidando de gardar o factor que fai nulo ao coeficiente (i, k) na matriz na posición correspondente.
- `sistl1(n,a,b)`: resolución dun sistema triangular inferior con diagonal de uns.

2.2. Escribe o **programa principal** que **lea a orde do sistema,** reserve memoria para tódolos arreglos que interveñen e, despois:

- Chame á subrutina de lectura e escritura dos datos do sistema.
- Garde a matriz `a` e o termo independente `b` en novas variables `aa` e `bb`, co fin de calcular posteriormente o residuo do sistema.
- Chame á subrutina que calcula a factorización e o determinante da matriz.
- Chame á subrutina que calcula a solución do sistema triangular inferior con diagonal de uns.
- Chame á subrutina que calcula a solución do sistema triangular superior.
- Chame á subrutina que calcula o residuo da solución.

2.3. Comproba o bo funcionamento dos programas escritos con distintos exemplos.

3. Finalmente, utiliza o **método de factorización PA = LU** tendo en conta que:

$$Ax = b \Longleftrightarrow PAx = Pb \Longleftrightarrow LUx = Pb \Longleftrightarrow \begin{cases} Ly = Pb, \\ Ux = y, \end{cases}$$

3.1. Escribe as novas **subrutinas:**

- `lupp(n,a,ip,deter)`: obtención da **factorización PA=LU,** a partir da eliminación de Gauss con estratexia de pivote parcial, e do determinante da matriz do sistema. Podes utilizar o código:

```
!inicializacion do determinante
deter=1.
!inicializacion da permutacion de filas
ip=(/(i,i=1,n)/)
!inicializacion do contador de cambios de filas
cont=0
!etapa k-esima da eliminacion
do k=1,n-1
  !busqueda do pivote e
  !da fila na que se encontra
  piv=a(ip(k),k)
  ipiv=k
  do i=k+1,n
    if(abs(piv)<abs(a(ip(i),k)))then
      piv=a(ip(i),k)
      ipiv=i
    end if
  end do
  !comprobacion de que o
  !k-esimo pivote non e nulo
  if(abs(piv)<1.e-12) then
    print*,'pivote nulo na etapa: ',k
    print*,'A matriz do sistema e singular!'
    stop
  end if
  !posta ao dia da permutacion
  !e do contador de cambios de filas,
  !se o pivote non esta na fila k
  if(ipiv/=k) then
    ipk=ip(ipiv)
    ip(ipiv)=ip(k)
    ip(k)=ipk
    cont=cont+1
  else
    ipk=ip(k)
  end if
  !actualizacion do determinante
  deter=deter*piv
  !eliminacion
  do i=k+1,n
    ipi=ip(i)
    a(ipi,k)=a(ipi,k)/piv
    do j=k+1,n
      a(ipi,j)=a(ipi,j)-a(ipi,k)*a(ipk,j)
    end do
  end do
end do
```

```
!comprobacion de que o
!ultimo pivote non e nulo
piv=a(ip(n),n)
if(abs(piv)<1.e-12) then
  print*,'pivote nulo na etapa: ',n
  print*,'A matriz do sistema e singular!'
  stop
end if
!remate do calculo do determinante
deter=deter*piv*(-1)**cont
```

O resultado é que tanto os elementos da matriz U: $u_{ij}, i \leq j$, como os elementos da matriz L: $l_{ij}, i > j$, gárdanse nas posicións `a(ip(i),j)`.

- `sistlpf1(n,a,b,u,ip)`: **resolución do sistema triangular inferior salvo a permutacion de filas** `ip` **e con uns na diagonal**. Neste caso, tanto a matriz do sistema como o termo independente teñen as filas permutadas, é preferible usar outro vector para gardar a solución do sistema triangular. Podes utilizar o código:

```
u(1)=b(ip(1))
do i=2,n
  aux=0.
  do j=1,i-1
    aux=aux+a(ip(i),j)*u(j)
  end do
  u(i)=b(ip(i))-aux
end do
```

- `sistupf2(n,a,b,ip)`: **resolución do sistema triangular superior salvo a permutacion de filas** `ip`**.** Neste caso soamente a matriz ten as filas permutadas e podes reutilizar o termo independiente para gardar a solución do sistema triangular. Podes utilizar o código:

```
do i=n,1,-1
  aux=0.
  do j=i+1,n
    aux=aux+a(ip(i),j)*b(j)
  end do
  b(i)=(b(i)-aux)/a(ip(i),i)
end do
```

3.2. Escribe o **programa principal** que **lea a orde do sistema,** reserve memoria para tódolos arreglos que interveñen e, despois:

- Chame á subrutina de lectura e escritura dos datos do sistema.
- Garde a matriz `a` na nova variable `aa`, co fin de calcular posteriormente o residuo do sistema.
- Chame á subrutina que efectúa a factorización $PA = LU$.
- Chame á subrutina que calcula a solución do sistema triangular inferior salvo a permutacion de filas `ip` e con uns na diagonal, e co termo independente afectado pola mesma permutacion de filas `ip`.
- Chame á subrutina que calcula a solución do sistema triangular superior salvo a permutacion de filas `ip`.
- Chame á subrutina que calcula o residuo da solución.

3.3. Comproba o bo funcionamento dos programas escritos con distintos exemplos.