Departamento de Matemática Aplicada

Facultade de Matemáticas
Campus Vida
Rúa Lope Gómez de Marzoa s/n
15782 Santiago de Compostela

Análise Numérica Matricial– Curso 2018-19

Práctica 2: Sistemas triangulares. Método de eliminación de Gauss. Método de eliminación de Householder.

1 Sistemas triangulares superiores: remonte.

1. Escribe un subprograma `sistu(a,b,u)` que resolva polo método de **remonte** un **sistema triangular superior**, $Au = b$, de orden `n`, con matriz cuadrada A, almacenada en `a`, segundo membro almacenado en `b` e a solución en `u`. Pódese utilizar o bucle seguinte:

```
u(n)=b(n)/a(n,n)
do i=n-1,1,-1
u(i)=b(i)
do j=i+1,n
u(i)=u(i)-a(i,j)*u(j)
enddo
u(i)=u(i)/a(i,i)
enddo
```

2. Escribe un programa principal `sistu_ppal` que permita comprobar o bo funcionamiento do anterior subprograma. Debes incluir no programa, na parte dedicada a especificacións, un bloque `interface` de presentación da subrutina `sistu(a,b)`:

```
interface
  subroutine sistu(a,b,u)
  implicit none
  real,dimension(:,:),intent(in) :: a !matriz del S.E.L.
  real,dimension(:),intent(in)   :: b !termino independente  do S.E.L.
  real,dimension(:),intent(out)  :: u ! solución do S.E.L.
  end subroutine sistu
end interface
```

   Crea ficheiros de datos adecuados para validar a tarefa proposta. Na entrada de datos debes introducir só a parte superior da matriz triangular, por filas, e poñer a cero a parte inferior (que, en realidade, non se utiliza no programa).

   Para comprobar que a o vector calculado aproxima á solución do sistema debes comprobar que $nr = \|r\|_2$ é próximo a cero, sendo $r = Au - b$ o vector residuo. Polo tanto, incluiremos despois da chamada á subrutina `sistu` unha chamada á subrutina `residuo(a,b,u,r)` que calcule o residuo $r = Au - b$. Non esquecerse de incluir no bloque `interface` a presentación desta subrutina.

   Fai as comprobacións pertinentes con sistemas de solución conocida para asegurarte que funciona ben a nova subrutina .

3. Modifica os programas do apartado anterior para crear unha subrutina `sistub(a,b,u)` e un programa principal `sistub_ppal` que resolvan o mesmo sistema triangular superior, $Au = b$, polo método de **remonte**, aproveitando as posibilidades de operacións en bloque que ofrece o cálculo con arreglos do Fortran 90. Debes utilizar o seguinte bucle:

```
u=b
do i=n,1,-1
u(i)=u(i)/a(i,i)
u(1:i-1)=u(1:i-1)-a(1:i-1,i)*u(i)
end do
```

Cos mesmos sistemas triangulares dos casos anteriores comproba o bo funcionamento destes novos programas.

4. Compara os tempos de cálculo entre as dúas subrutinas `gauss` e `gaussb`, usando a función intrínseca do Fortran 90 `dtime`. ¿Son diferentes os tempos de cálculo? Fai unha táboa cos tempos de cálculo para cada un dos programas cando se resolve o sistema triangular superior con $a_{ij} = j - i + 1, j = i, ..., n, j \geq i$ y $b_i = (n - i + 1)(n - i + 2)/2, i = 1, ...n$, para $n = 1000, 2000, ...15000$.

2 Sistemas triangulares inferiores: descenso.

Repite o exercicio anterior (excepto o apartado 4) cambiando `sistu(a,b,u)` por `sistl(a,b,u)`, **superior** por **inferior** e `remonte` por `descenso`.

3 Método de eliminación de Gauss (sen pivote)

3.1 Programa básico

Agora debes implementar, en Fortran 90, a resolución dun S.E.L. $Au = b$, mediante o **método de eliminación de Gauss**. O programa aproveitarase para calcular tamén o **determinante** da matriz A. Tes que:

a) Escribir as **subrutinas seguintes:**

- `datsis(a,b)`: **lectura e escritura dos datos de entrada** (matriz de coeficientes `a` e termo independente `b`).
- `gauss(a,b,deter)`: **proceso de 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 de coeficientes. 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)
```

b) Escribir o **programa principal** `gauss_ppal` que lea a **orde do sistema**, n, reserve memoria para todos os arreglos que interveñen e chame as subrutinas `datsis(a,b)`, `gauss(a,b,deter)` e `sistu(a,b,u)`–elaborada no exercicio 1– para concluir resolvendo o sistema lineal e comprobar o resultado mediante o cálculo da norma euclídea nr do residuo $r = Au - b$ e escribilo. A estructura xeral do programa será a seguinte:

```
program gauss_ppal
interface das subrutinas datsis, gauss e sistu
call datsis(a,b)
call gauss(a,b,deter)
call sistu(a,b,u)
call residuo(a,b,u,r)
Escritura de resultados: deter, u,r,nr.
end program gauss_ppal
```

c) Comprobar o bo funcionamento dos programas escritos con distintos exemplos de sistemas lineais, algúns con solución conocida.

3.2 Programa con operacións vectoriais.

Repetir o apartado anterior adaptando os programas de eliminación e remonte ás posibilidades de operacións vectoriais nos arreglos que nos da o Fortran 90. Polo tanto, no programa principal –que chamaremos `gaussb_ppal`– no lugar de `sistu` uitilizaremos a subrutina `sistub(a,b,u)`– feita no exercicio 1–, que utiliza as operacións vectoriais no remonte, e adaptaremos a eliminación escribindo unha subrutina `gaussb(a,b,deter)` na que o doble bucle `do` da eliminación se substitúe 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)
```

Comprobar que o funcionamento destes novos programas `gaussb, sistub, gaussb_ppal` é igualmente correcto (basta resolver os mesmos sistemas que no apartado anterior).

4 Sistemas transformables a triangulares superiores por permutación de filas.

Escribe unha subrutina `sistupf(a,b,u,l)` para resolver un sistema lineal $Au = b$ sendo A unha matriz, almacenada en `a`, que mediante unha permutación `l` das súas filas se reduce a unha matriz triangular superior (`l(i)` é a fila que pasaría a ser a fila `i` na transformación a matriz triangular superior, ou sexa, $a_{l_i 1} = a_{l_i 2} = \ldots a_{l_i i-1} = 0$). Podes utilizar o código:

```
u(n)=b(l(n))/a(l(n),n)
do i=n-1,1,-1
u(i)=b(l(i))
do j=i+1,n
u(i)=u(i)-a(l(i),j)*u(j)
enddo
u(i)=u(i)/a(l(i),i)
enddo
```

5 Método de eliminación de Gauss con estratexia de pivote parcial.

Trátase de implementar en Fortran 90 o método de eliminación de Gauss con pivote parcial para resolver un sistema lineal. Debes seguir os seguintes pasos:

1. Escribir a subrutina `gausspp(a,b,ip,deter)` que efectúa o proceso de eliminación de Gauss con pivote parcial e devolve o sistema equivalente a $Au = b$ que é transfomable en triangular superior pola permutación de filas `ip` dada pola situación dos pivotes. Podes utilizar un código de cálculo coma o seguinte:

```
deter=1.
!inicializacion da permutacion de filas
l=(/(i,i=1,n)/)

!etapa k-esima da eliminacion
do k=1,n-1
  !busqueda do pivote e
  !da fila na que se atopa
  piv=a(l(k),k)
  p=k
  do i=k+1,n
    if(abs(piv)<abs(a(l(i),k)))then
      piv=a(l(i),k)
      p=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  e singular!'
    stop
  end if
  !posta ao dia da permutacion
  deter=deter*piv
  if (p /= k) then
    deter = -deter

    m=l(k)
    l(k)=l(p)
    l(p)=m
  endif
  piv=a(l(k),k)
  lk=l(k)

  !Eliminación
  do i=k+1,n
    li=l(i)
z=a(li,k)/piv
    do j=k+1,n
      a(li,j)=a(li,j)-z*a(lk,j)
    end do
    b(li)=b(li)-z*b(lk)
  end do
end do
!Remate do calculo do determinante.
piv=a(l(n),n)
deter=deter*piv
if(abs(piv)<1.e-12) then
  print*,'pivote nulo na etapa: ',n
  print*,'A matriz e singular!'
  stop
end if
```

2. Rematar a programación do método nos pasos seguintes

   a) Escribe o **programa principal** `gauspp_ppal` que lea a **orde do sistema,** reserve memoria para todos os arreglos que interveñen e despois:
      - Chame á subrutina de lectura e escritura dos datos do sistema `datsis(a,b)`
      - Chame á subrutina que efectúa a eliminación con pivoter parcial: `gausspp(a,b,ip,deter)`

- Chame á subrutina `sistupf(a,b,u,ip)` que calcula a solución do sistema triangular superior.
- Chame á subrutina `residuo(a,b,u,r)` para calcular o residuo $r = Au - b$ e comprobar o funcionamento correcto do método.
- Escriba os resultados: solución e determinante.
- Non esquecerse de incluir os `interface` para as subrutinas `datsis, gausspp, sistupf` e `residuo`.

c) Comproba o bo funcionamento dos programas escritos con distintos exemplos de sistemas de solución conocida.

6 Método de eliminación de Householder

Seguindo o mesmo esquema que para o método de eliminación de Gauss debes implementar agora o **método de eliminación de Householder** para S.E.L. $Au = b$. O programa aproveitarase para calcular tamén o **determinante** da matriz A. Tes que escribir a subrutina `househ(a,b,deter)` do **proceso de eliminación de Householder** 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 de coeficientes. Utiliza o código que proporcionamos na notas de teoría. Utiliza as subrutinas `datsis, sistu` e `residuo` para lectura de datos, resolver o sistema triangular equivalente e comprobar o resultado calculando o residuo. Verifica o bo funcionamento dos programas escritos con distintos exemplos de sistemas lineais de solución conocida.