```
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
! Compilation with IFORT:
!  ifort -xP -O3 -ipo -vms -c m_ctes.f90
!  ifort -xP -O3 -ipo -vms -c m_temps.f90
!  ifort -xP -O3 -ipo -vms -c m_tools.f90
!  ifort -xP -O3 -ipo -vms -c m_gauss0.f90
!  ifort -xP -O3 -ipo -vms -c m_krylov1.f90
!  ifort -xP -O3 -ipo -vms -c dfom.f90
!  ifort -xP -O3 -ipo -vms -o dfom_s.exe *.o
!
! Compilation with G95:
!  g95 -c m_ctes.f90
!  g95 -c m_temps.f90
!  g95 -c m_tools.f90
!  g95 -c m_gauss0.f90
!  g95 -c m_krylov1.f90
!  g95 -c dfom.f90
!  g95 -o dfom.exe *.o
!
! Compilation with (ADAPTOR + MPICH):
!  adaptor -settings
!  adaptor -hpf -dm -free -v -c m_ctes.f90
!  adaptor -hpf -dm -free -v -c m_temps.f90
!  adaptor -hpf -dm -free -v -c m_tools.f90
!  adaptor -hpf -dm -free -v -c m_gauss0.f90
!  adaptor -hpf -dm -free -v -c m_krylov1.f90
!  adaptor -hpf -dm -free -v -c dfom.f90
!  adaptor -hpf -dm -free -v -o dfom_p.exe *.o
!
! Running with MPICH:
!  mpdboot -n 21 -f ~/mpd.hosts
!  mpdtrace
!  mpdringtest
!  mpiexec -machinefile ~/machi.dat -np 2 dfom_p.exe
!  mpiexec -machinefile ~/machi.dat -1 -np 4 dfom_p.exe
!  mpdallexit
!  pdsh -a hostname
!
! The same running but with MPI (old):
!  mpirun -machinefile ~/machi.dat -np 2 dfom_p.exe
!  mpirun -machinefile ~/machi.dat -nolocal -np 4 dfom_p.exe
!
! Some running details:
!  si n = 8 000 entonces m = n^2, RAM = m * 8 /1e6 =  512 [Mbytes]
!  pero como aqui hay una copia de A se tiene ram = 1,024 [Gbytes]
!
! output example:
!  dfom
!  number of unknowns ; n = ?
!  max diff =  5.83895598538220E-10
!  time   =  576.0575  seconds
!  mflops =  374.9626  on  8  processors
!  mflops =  46.87032  for one processor
! ---------------------------------------------------------------------
program  dfom
  use m_ctes
  use m_temps
  use m_tools1
  use m_krylov1
  implicit none
  integer (iin) :: j
  real (idp), dimension (:,:), allocatable ::  a
  real (idp), dimension (:)  , allocatable ::  b, x
  !hpf$ distribute (block,*) :: a
  !hpf$ align (i)   with a (i,*) :: b
  integer (iin) :: n, m
  integer (iin) :: i, k, falla = 1
  !
  integer (iin) :: n_unknowns, m_subspace
```



```
  namelist /krylov_basic_data/ n_unknowns, m_subspace
  !
  ! lee del disco
  open  (1, file = "krylov_basic.dat", status = "old", err = 100)
  read  (1,nml = krylov_basic_data)
  write (*,nml = krylov_basic_data)
  close (1, status = "keep")
  n = n_unknowns
  m = m_subspace
  if (m > n .or. m < 1) m = n
  !
  !carteles
  write (*,*)
  write (*,*) "solution of a SEL : FOM "
  write (*,*) "number of unknows           ; n = ", n
  write (*,*) "size of the Krylov subspace ; m = ", m
  !
  !aloca sistema
  ier = 0
  allocate ( a (1:n,1:n), stat = ier (1) )
  allocate ( b (1:n)    , stat = ier (2) )
  allocate ( x (1:n)    , stat = ier (3) )
  if ( any (ier .ne. 0) ) call errata (ier(1:3), "aloca" )
  !
  !define un SEAL A X = B donde A es la matriz del sistema,
  !X, B matrices solucion y carga, respectivamente.
  do  j = 1, n
    call  random_number (x)
    a (1:n,j) = x (1:n)
  end do
  call  random_number (x)
  b = x
  !
  !hpf$ independent
  forall (i=1:n)  a (i,i) = a (i,i) + dble (n)
  !
  !solucion fom
  call  dsolve_fom (a, b, m)
  !
  !eventual impresion
  ! print *
  ! print *, "matriz A "
  ! do  i = 1, n
  !   print *, real ( a (i,:) )
  ! end do ! i
  ! print *
  ! print *, "carga  b "
  ! do  i = 1, n
  !   print *, real ( x (i,:) )
  ! end do ! i
  ! print *
  ! print *, "soluc       x "
  ! do  i = 1, n
  !   print *, real ( b (i) )
  ! end do ! i
  !
  !dloca
  ier = 0
  deallocate ( x, stat = ier (1) )
  deallocate ( b, stat = ier (2) )
  deallocate ( a, stat = ier (3) )
  if (any (ier .ne. 0))  call errata (ier(1:3), "dloca" )
  !
  falla = 0
  100 if (falla .ne. 0) stop "error: namelist file was not found ..."
  !
  print *
  print *, "fine ... "
  !
```

```
  ! formatos
  110 format (a)
  120 format (1x, i4, 1x, i20, 1x, e24.12)
end program
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
```

```
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
! Compilation with IFORT:
!  ifort -xP -O3 -ipo -vms -c m_ctes.f90
!  ifort -xP -O3 -ipo -vms -c m_temps.f90
!  ifort -xP -O3 -ipo -vms -c m_tools1.f90
!  ifort -xP -O3 -ipo -vms -c m_gauss0.f90
!  ifort -xP -O3 -ipo -vms -c dgauss1v.f90
!  ifort -xP -O3 -ipo -vms -o dgauss1v_s.exe *.o
!
! Compilation with G95:
!  g95 -c m_ctes.f90
!  g95 -c m_temps.f90
!  g95 -c m_tools1.f90
!  g95 -c m_gauss1.f90
!  g95 -c gauss1.f90
!  g95 -o gauss1.exe *.o
!
! Compilation with (ADAPTOR + MPICH):
!  adaptor -settings
!  adaptor -hpf -dm -free -v -c m_ctes.f90
!  adaptor -hpf -dm -free -v -c m_temps.f90
!  adaptor -hpf -dm -free -v -c m_tools1.f90
!  adaptor -hpf -dm -free -v -c m_gauss1.f90
!  adaptor -hpf -dm -free -v -c dgauss1v.f90
!  adaptor -hpf -dm -free -v -o dgauss1v_p.exe *.o
!
! Running with MPICH:
!  mpdboot -n 21 -f ~/mpd.hosts
!  mpdtrace
!  mpdringtest
!  mpiexec -machinefile ~/machi.dat -np 2 dgauss1v_p.exe
!  mpiexec -machinefile ~/machi.dat -1 -np 4 dgauss1v_p.exe
!  mpdallexit
!  pdsh -a hostname
!
! The same running but with MPI (old):
!  mpirun -machinefile ~/machi.dat -np 2 dgauss1v_p.exe
!  mpirun -machinefile ~/machi.dat -nolocal -np 4 dgauss1v_p.exe
!
! Some running details:
!  si n = 8 000 entonces m = n^2, RAM = m * 8 /1e6 =  512 [Mbytes]
!  pero como aqui hay una copia de A se tiene ram = 1,024 [Gbytes]
!
! output example:
!  number of unknowns ; n = ?
!  max diff =  5.83895598538220E-10
!  time   =  576.0575  seconds
!  mflops =  374.9626  on  8  processors
!  mflops =  46.87032  for one processor
! -------------------------------------------------------------------
program  dgauss1v
  use m_ctes
  use m_temps
  use m_tools1
  use m_gauss1
  implicit none
  integer (iin), parameter :: ncut = 1000
  integer (iin) :: i, j
  real    (idp), dimension (:,:), allocatable ::  a
  real    (idp), dimension (:)  , allocatable ::  b, r
  real    (idp), dimension (:,:), allocatable :: a0
  real    (idp), dimension (:)  , allocatable :: b0
  real    (idp), dimension (:)  , allocatable ::  x
  real    (idp) :: s
  !
  !hpf$ distribute (*,block) :: a
  !hpf$ align (i,*) with a (i,*) :: a0
  !hpf$ align (i)   with a (i,*) :: b0, b,  r
  !
```



```
  integer (iin) :: n
  integer (iin), parameter :: m = 1
  integer (iin) :: k, falla = 1
  !
  integer (iin) :: n_unknowns, n_loads
  namelist /gauss_basic_data/ n_unknowns, n_loads
  !
  ! lee del disco
  open  (1, file = "gauss_basic.dat", status = "old", err = 100)
  read  (1,nml = gauss_basic_data)
  write (*,nml = gauss_basic_data)
  close (1, status = "keep")
  n = n_unknowns
  !
  !carteles
  write (*,*)
  write (*,*) "Gauss solution of a SEL with dynamic RAM "
  write (*,*) "number of unknows ; n = ", n
  write (*,*) "number of loads   ; m = ", m
  !
  !aloca sistema
  ier = 0
  allocate ( a (1:n,1:n), stat = ier (1) )
  allocate ( b (1:n)    , stat = ier (2) )
  allocate (a0 (1:n,1:n), stat = ier (3) )
  allocate (b0 (1:n)    , stat = ier (4) )
  allocate ( r (1:n)    , stat = ier (5) )
  allocate ( x (1:n)    , stat = ier (6) )
  if ( any (ier .ne. 0) ) call errata (ier(1:6), "aloca")
  !
  !define un SEAL A X = B donde A es la matriz del sistema,
  !X, B matrices solucion y carga, respectivamente.
  do  j = 1, n
    call  random_number (x)
    a (1:n,j) = x (1:n)
  end do
  call  random_number (x)
  b (1:n) = x (1:n)
  !hpf$ independent
  forall (i=1:n)  a (i,i) = a (i,i) + dble (n)
  !
  !copia para posterior verificacion de los resultados
  if (n < ncut) then
    a0 = a
    b0 = b
  end if
  !
  !solucion gaussiana
  call  dsolve_b1_adp (a, b, imprime = .true.)
  !
  !verifica matriz solucion X con R = B - A X
  if (n < ncut) then
    print *
    print *, "verification; r = A x - b "
    r = matmul (a0,b) - b0
    s = sqrt (sum (abs (r * r)))
    print *, "|| r ||_2 = ", s
  end if
  !
  !eventual impresion
  ! print *
  ! print *, "matriz dato A "
  ! do  i = 1, n
  !   print *,  a0 (i,:)
  ! end do ! i
  ! print *
  ! print *, "carga B "
  ! do  i = 1, n
  !   print *,  b0 (i)
```

```
  ! end do ! i
  ! print *
  ! print *, "matriz factorizada A "
  ! do  i = 1, n
  !   print *, a (i,:)
  ! end do ! i
  ! print *
  ! print *, "solucion X "
  ! do  i = 1, n
  !   print *, b (i)
  ! end do ! i
  ! print *
  ! print *, "residuo  R "
  ! do  i = 1, n
  !   print *, r (i)
  ! end do ! j
  !
  !dloca
  ier = 0
  deallocate ( x, stat = ier (6) )
  deallocate ( r, stat = ier (5) )
  deallocate (b0, stat = ier (4) )
  deallocate (a0, stat = ier (3) )
  deallocate ( b, stat = ier (2) )
  deallocate ( a, stat = ier (1) )
  if (any (ier .ne. 0))  call errata (ier(1:7), "dloca")
  !
  falla = 0
  100 if (falla .ne. 0) stop "error: namelist file was not found ..."
  !
  print *
  print *, "fine ... "
  !
  ! formatos
  110 format (a)
  120 format (1x, i4, 1x, i20, 1x, e24.12)
end program
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
```

```
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
! Problema del Conjunto de Mandelbrot
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
! Compilation with INTEL:
!  ifort -vms -c mandel.f90
!  ifort -vms -o mandel.exe1 *.o
!
! Compilation with GNU:
!  g95 -c circulo.f90
!  g95 -o mandel.exe2 *.o
!
! Compilation with (ADAPTOR + MPICH):
!  adaptor -hpf -dm -free -c mandel.f90
!  adaptor -hpf -dm -free -o mandel.exe3 *.o
!
! Running with MPICH:
!  mpdboot -n 4 -f ~/mpd.hosts
!  mpdtrace
!  mpdringtest
!  mpiexec -machinefile ~/machi.dat -np 4 mandel.exe3
!  mpiexec -machinefile ~/machi.dat -1 -np 4 mandel.exe3
!  mpdallexit
!
! Para ver Salida Grafica Color con
!    xv -expand 0.5 mandel_co.pgm &
!
! Para ver Salida Grafica Blanco y Negro con
!    xv -expand 0.5 mandel_bw.pgm &
!
! ------------------------------------------------------------------
! Programa HPF para el conjunto de Mandelbrot.
! Ref. Liverpool HPF course, Marshall A.C., 1997.
! ------------------------------------------------------------------
program mandel
  implicit none
  integer, parameter :: n = 2048
  integer, parameter :: resolucion = 516
  integer, dimension (n,n) :: color
  real   , dimension (n,n) :: zr, zi, cr, ci, ur, ui
  !hpf$ distribute (block,block) :: zr, zi, cr, ci, ur, ui, color
  real    :: s10, s11, s15, s20, s25, s30, s40, s50, s60
  integer :: i, j, k, ini, fin
  integer :: c, rojo, verde, azul
  !
  write (*,*)
  write (*,*)" Conjunto de Mandelbrot ... "
  write (*,*)" n = ", n
  write (*,*)
  !
  ! inicializa
  forall (j=1:n, i=1:n)
    zr (i,j) = real (i-1) / real (n-1)
    zi (i,j) = real (j-1) / real (n-1)
  end forall
  cr = zr       ; ci = zi
  ur = zr * zr  ; ui = zi * zi
  color = 0
  call system_clock (ini)
  !
  ! lazo
  do   k = 0, resolucion
    where ( (ur + ui) .le. 4.0 )
      ur = zr * zr
      ui = zi * zi
      zi = 2.0 * zr * zi + ci
      zr = (ur - ui)     + cr
      color = k
    end where
  end do
```



```
  call system_clock (fin)
  !
  ! archivo gv monocromatico. Usar gv -expand 0.5 mandel_bw.pgm
  open  (10, file = 'mandel_bw.pgm', status = 'unknown')
  write (10,100)  n, n, resolucion
  write (10,110)  color
  close (10, status = 'keep')
  !
  ! archivo gv color. Usar gv -expand 0.5 mandel2_co.pgm
  s10 = 0.039216 * resolucion ! 10
  s11 = 0.043137 * resolucion ! 11
  s15 = 0.058824 * resolucion ! 15
  s20 = 0.078431 * resolucion ! 20
  s25 = 0.098039 * resolucion ! 25
  s30 = 0.117650 * resolucion ! 30
  s40 = 0.156860 * resolucion ! 40
  s50 = 0.196080 * resolucion ! 50
  s60 = 0.235290 * resolucion ! 60
  !
  open  (11, file = 'mandel_co.pgm', status = 'unknown')
  write (11,120)  n, n, resolucion
  do  i = 1, n
  do  j = 1, n
    c = color (i,j)
    if     ( c .eq. resolucion ) then
      rojo = 0       ; verde = 0       ; azul = 0
    elseif ( c .ge.   0 .and. c .le. s10 ) then
      rojo = c * s25 ; verde = c * s20 ; azul = c * s10
    elseif ( c .ge. s11 .and. c .le. s15 ) then
      rojo = c * s60 ; verde = c * s15 ; azul = c * s30
    else
      rojo  = resolucion / 1
      azul  = resolucion / 2
      verde = resolucion / 4
    end if
    write (11,130) rojo, verde, azul
  end do
  end do
  close (11, status = 'keep')
  !
  ! formatos
  100 format ('P2' / i5, 2x, i5 / i5)
  110 format ( 11 (1x, i5) )
  120 format ('P3' / i5, 2x, i5 / i5)
  130 format ( 11 (1x, i5) )
end program
! ====================================================================
```

```
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
! Compilation with IFORT:
!  ifort -xP -O3 -ipo -vms -c m_ctes.f90
!  ifort -xP -O3 -ipo -vms -c m_temps.f90
!  ifort -xP -O3 -ipo -vms -c m_tools1.f90
!  ifort -xP -O3 -ipo -vms -c m_gauss1.f90
!  ifort -xP -O3 -ipo -vms -c zgauss1v.f90
!  ifort -xP -O3 -ipo -vms -o zgauss1v_s.exe *.o
!
! Compilation with G95:
!  g95 -c m_ctes.f90
!  g95 -c m_temps.f90
!  g95 -c m_tools1.f90
!  g95 -c m_gauss1.f90
!  g95 -c zgauss1v.f90
!  g95 -o zgauss1v.exe *.o
!
! Compilation with (ADAPTOR + MPICH): there is an adaptor BUG
!  adaptor -settings
!  adaptor -hpf -dm -free -v -c m_ctes.f90
!  adaptor -hpf -dm -free -v -c m_temps.f90
!  adaptor -hpf -dm -free -v -c m_tools1.f90
!  adaptor -hpf -dm -free -v -c m_gauss1.f90
!  adaptor -hpf -dm -free -v -c zgauss1v.f90
!  adaptor -hpf -dm -free -v -o zgauss1v_p.exe *.o
!
! Running with MPICH:
!  mpdboot -n 21 -f ~/mpd.hosts
!  mpdtrace
!  mpdringtest
!  mpiexec -machinefile ~/machi.dat -np 2 zgauss1v_p.exe
!  mpiexec -machinefile ~/machi.dat -1 -np 4 zgauss1v_p.exe
!  mpdallexit
!  pdsh -a hostname
!
! The same running but with MPI (old):
!  mpirun -machinefile ~/machi.dat -np 2 zgauss1v_p.exe
!  mpirun -machinefile ~/machi.dat -nolocal -np 4 zgauss1v_p.exe
!
! Some running details:
!  si n = 8 000 entonces m = n^2, RAM = m * 8 /1e6 =  512 [Mbytes]
!  pero como aqui hay una copia de A se tiene ram = 1,024 [Gbytes]
!
! output example:
!  number of unknowns ; n = ?
!  max diff =  5.83895598538220E-10
!  time   =  576.0575  seconds
!  mflops =  374.9626  on  8  processors
!  mflops =  46.87032  for one processor
! -------------------------------------------------------------------
program  zgauss1v
  use m_ctes
  use m_temps
  use m_tools1
  use m_gauss1
  implicit none
  integer (iin), parameter :: ncut = 1000
  integer (iin) :: j
  complex (idp), dimension (:,:), allocatable ::  a
  complex (idp), dimension (:)  , allocatable ::  b, r
  complex (idp), dimension (:,:), allocatable :: a0
  complex (idp), dimension (:)  , allocatable :: b0
  real    (idp), dimension (:)  , allocatable ::  x, y
  real    (idp) :: s
  !
  !hpf$ distribute (*,block) :: a
  !hpf$ align (i,*) with a (i,*) :: a0
  !hpf$ align (i)   with a (i,*) :: b0, b, r
  !
```



```
  integer (iin) :: n, m
  integer (iin) :: i, k, falla = 1
  !
  integer (iin) :: n_unknowns, n_loads
  namelist /gauss_basic_data/ n_unknowns, n_loads
  !
  ! lee del disco
  open  (1, file = "gauss_basic.dat", status = "old", err = 100)
  read  (1,nml = gauss_basic_data)
  write (*,nml = gauss_basic_data)
  close (1, status = "keep")
  n = n_unknowns
  m = n_loads
  !
  !carteles
  write (*,*)
  write (*,*) "Gauss solution of a SEL with dynamic RAM "
  write (*,*) "number of unknows ; n = ", n
  write (*,*) "number of loads   ; m = ", m
  !
  !aloca sistema
  ier = 0
  allocate ( a (1:n,1:n), stat = ier (1) )
  allocate ( b (1:n)    , stat = ier (2) )
  allocate (a0 (1:n,1:n), stat = ier (3) )
  allocate (b0 (1:n)    , stat = ier (4) )
  allocate ( r (1:n)    , stat = ier (5) )
  allocate ( x (1:n)    , stat = ier (6) )
  allocate ( y (1:n)    , stat = ier (7) )
  if ( any (ier .ne. 0) ) call errata (ier(1:7), "aloca" )
  !
  !define un SEAL A X = B donde A es la matriz del sistema,
  !X, B matrices solucion y carga, respectivamente.
  do  j = 1, n
    call  random_number (x)
    call  random_number (y)
    a (1:n,j) = cmplx ( x (1:n) , y (1:n) )
  end do
  call  random_number (x)
  call  random_number (y)
  b (1:n) = cmplx ( x (1:n) , y (1:n) )
  !hpf$ independent
  forall (i=1:n)  a (i,i) = a (i,i) + cmplx (dble(n), 0.0_idp)
  !
  !copia para posterior verificacion de los resultados
  if (n < ncut) then
    a0 = a
    b0 = b
  end if
  !
  !solucion gaussiana
  call  zsolve_b1_adp (a, b, imprime = .true.)
  !
  !verifica matriz solucion X con R = B - A X
  if (n < ncut) then
    print *
    print *, "verification; r = A x - b "
    r = matmul (a0,b) - b0
    s = sqrt (sum (abs (r * r)))
    print *, "|| r ||_2 = ", s
  end if
  !
  !eventual impresion
    print *
    print *, "matriz dato A "
    do  i = 1, n
      print *,  a0 (i,:)
    end do ! i
    print *
```

```
    print *, "carga B "
    do  i = 1, n
      print *,  b0 (i)
    end do ! i
    print *
    print *, "matriz factorizada A "
    do  i = 1, n
      print *, a (i,:)
    end do ! i
    print *
    print *, "solucion X "
    do  i = 1, n
      print *, b (i)
    end do ! i
    print *
    print *, "residuo  R "
    do  i = 1, n
      print *, r (i)
    end do ! j
  !
  !dloca
  ier = 0
  deallocate ( y, stat = ier (7) )
  deallocate ( x, stat = ier (6) )
  deallocate ( r, stat = ier (5) )
  deallocate (b0, stat = ier (4) )
  deallocate (a0, stat = ier (3) )
  deallocate ( b, stat = ier (2) )
  deallocate ( a, stat = ier (1) )
  if (any (ier .ne. 0))  call errata (ier(1:7), "dloca" )
  !
  falla = 0
  100 if (falla .ne. 0) stop "error: namelist file was not found ..."
  !
  print *
  print *, "fine ... "
  !
  ! formatos
  110 format (a)
  120 format (1x, i4, 1x, i20, 1x, e24.12)
end program
! -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----
```