Sample Fortran Programs

•#1 Sine Computation

In computer $\sin(x)$ is computed by a so-called Taylor series expansion:

$$sin(x) \;=\; x - x^3/(3!)+x^5/(5!)-x^7/(7!)-----+(-1)^{2n+1}x^{2n+1}/(2n+1)!... \tag{1}$$

We cannot proceed indefinitely. We have to stop at N terms. The error at stoping at N-th term is less than the absolute value of the $N+1$-th term. So, given ϵ, we stop if

$$\|x^{(2**n+1)}/(2n+1)!\| \;<\; \epsilon \tag{2}$$

The following is the program and outputs.

```
      implicit real*8(A-H, O-Z)
      real*8 mysine

!  This program computes the sin function by Taylor series
!  expansion. It calls the function mysine and compare the
!  computed value with the library value.
!
!  Epsilon is the error control parameter
!

      print*, ' ** Enter x, epsilon **'
      read(*,*) X, epsilon

      y = mysine(x, epsilon)
!  y_lib is the library value
      y_lib = dsin(x)

      print*, X, ' My value = ', y, ' Library value = ', y_lib

      stop
      end

      real*8 function mysine(X, epsilon)

      implicit real*8(A-H, O-Z)

      kk = 1
      term = x
      i = 0
      mysine = 0.
10    continue
      mysine = mysine + term*kk
      i = i + 1
```

```
         kk = -kk
         term = term * x*x/dfloat(2*i)/dfloat(2*i+1)
         print*,' i = ', i, term, mysine
!  if the absolute value of the next term is less than epsilon,
!    stop
         if(abs(term) .ge. epsilon) goto 10

         return
         end
```

```
 gfortran -o bbm mysine.f

bbm
  ** Enter x, epsilon **
0.7 1.e-4
  i =  1  0.0571666667  0.7
  i =  2  0.00140058333  0.642833333
  i =  3  1.63401389E-05  0.644233917
  0.7 My value =   0.644233917 Library value =   0.644217687

bbm
  ** Enter x, epsilon **
0.7 1.e-16
  i =  1  0.0571666667  0.7
  i =  2  0.00140058333  0.642833333
  i =  3  1.63401389E-05  0.644233917
  i =  4  1.11203723E-07  0.644217577
  i =  5  4.95362039E-10  0.644217688
  i =  6  1.55594487E-12  0.644217687
  i =  7  3.63053802E-15  0.644217687
  i =  8  6.54030746E-18  0.644217687
  0.7 My value =   0.644217687 Library value =   0.644217687
```

- #2 Matrix Multiplication

Let A be a m by n matrix, B a n by p matrix. Then, C = A B is a m by p matrix. In other words,

$$C(i,j) \;=\; AB \;=\; \sum_{k=1}^{n} A_{i,k}B_{k,j} \;,\; i \;=\; 1,\ldots,m,\; j \;=\; 1\;,\ldots,p \quad (3)$$

The following program computes C = AB

```
        implicit real*8(A-H, O-Z)

        dimension A(100, 25), B(25, 300), C(100, 300)

        do i = 1, 100
        do j = 1, 25
          a(i,j) = 1./float(i+j-1)
        enddo
        enddo

        do i = 1, 25
        do j = 1, 300
          b(i,j) = i + j - 0.2
        enddo
        enddo

M1 = 100 ; M2 = 25 ; M3 = 300
call mymatmul(M1, M2, M3,  A, B, C)

write(*, *) C

stop
end

subroutine mymatmul(M1, M2, M3,  A, B, C)
implicit real*8(a-h, o-z)

dimension A(M1, *), B(M2, *), C(M1, *)


        do i = 1 , M1
        do j = 1 , M3
          C(i,j) = 0.
          do k = 1 , M2
```

```
   C(i,j) = C(i,j) + A(i,k)*B(k,j)
  enddo
enddo
enddo

return
end
```

```
!
!  Main Routine for Serial Precondtioner for Large Sparse
!  Nonsymmetric Linear Systems  using Flexible GMRES(m)
!   Matrix in Matrix Market Format
!   ILU(0), ILU(k), ILUT, Point-SSOR, AGMG
!     MC64 + RCMK
!
implicit real*8(a-h, o-z)
parameter(np=332, np2=np*np, np3=np2*10)
parameter(Maxlfil=2*40, ndiag=10, Mdl=Ndiag+Maxlfil)
        parameter(Maxnz=ndiag*np2)
        parameter(liw=maxnz, ldw=maxnz)
dimension x(np2), sol(np2), rhs(np2), vv(np2, 41), a(np3),  &
  ja(np3), ia(np2), w(np2, 40), wk1(np2), wk2(np2)
dimension alu(Mdl*np2), jlu(Mdl*np2), ju(np2)
        dimension wu(np2), wl(np2), jr(np2), jwu(3*np2), jwl(np2)
dimension levs(Mdl*np2)
dimension a2(np3), ja2(np3), ia2(np2)
dimension iperm (np2)
integer icntl(10), info(10), iw(liw), colperm(np2),  &
       rowperm(np2), lenc(np2)
        real*8 dw(ldw)
integer lfront(np2), iord(np2), riord(np2), il(np2)
        character*1 line(80)
        character*15 filename
        character*60 filename2
common/Debug1/Idebug1
common/Debug2/iters_fgmres

open(unit=10, file='out_precon53', access='append')
write(10,*) ' -------------------'
write(10,*) fdate()
```

```
write(*,*) ' ** Enter M,, Preconditioner **'
read(*,*) m, Iprecon

if(Iprecon .ge. 1 .and. Iprecon .le. 3) then
  read(*,*) Lfil, droptol
else if(Iprecon .eq. 4) then
  read(*,*) Omega, Iter_ssor
else if(Iprecon .eq. 6) then
  read(*,*) iter_dagmg, tol_dagmg, nrest
endif

if(Iprecon .eq. 1) then
  write(10,*)' ** ILU(0) Preconditioning **'
else if(Iprecon .eq. 2) then
  write(10,*)' ** ILU(', Lfil, droptol,  ') Preconditioning **'
else if(Iprecon .eq. 3) then
  write(10,*)' ** ILUT(,', Lfil,') Preconditioning **'
else if(Iprecon .eq. 4) then
  write(10,*) ' ** Point SSOR Preconditioning with omega = ', omeg
else if(Iprecon .eq. 6) then
  write(10,*) ' ** Aggregate Multi-Grid Preconditioning = ', &
              Iter_dagmg, tol_dagmg
endif

write(*,*) ' ** Enter Tol, maxits, Idebug1'
read(*,*) eps, maxits, Idebug1

write(*,*)' ** Enter IMC64, Ifirst of RCMK **'
read(*,*) IMC64, ifirst
write(*,*) ' ** IMC64 = ', Imc64

if(Imc64 == 1) then
```

```
  write(10,*) ' -- MC64 preordering --'
else
  write(10,*) ' -- Natural ordering --'
endif

! if(Iprob .eq. 1) then
!   call mat1d(Ny, Nx, a, ja, ia)
!   N = NX*NY
! else if(Iprob .eq. 4) then
!   call mat4d(Ny, nx, a, ja, ia)
!   N = NX*NY
! endif


close(10)

Ifail = 0
1234 continue

open(unit=10, file='out_precon53', access='append')

ifail2 = 0
icase = 0

        do i = 1, np2
          ia(i) = 0
          ia2(i) = 0
        enddo

        do i = 1, Maxnz
          a(i) = 0.
          a2(i) = 0.
          ja(i) = 0
          ja2(i) = 0
```

```
        enddo

        write(*,*)'  -- Enter FileName --'
        read(*,*,end=9876) filename
        open(unit=99,file='temp99')
        write(99,2) filename
2       format('../MATRICES/',a15)
        rewind 99
        read(99,3) filename2
3       format(a30)

        open(unit=8,file=filename2)
write(*,*) ' filename = ', filename


        nlines = 0
        ncomment = 0
100     continue
        read(8,1,end=200) line
1       format(80a1)
        nlines = nlines + 1
        if(line(1) .eq. '%') then
          ncomment = ncomment + 1
          goto 100
        else
          goto 200
        endif
200     continue
        print*,' nlines, ncomment', nlines, ncomment

        rewind 8
300     continue
        do i = 1, ncomment
          read(8,1) line
```

```
        enddo

        read(8,*) N1, N, nnz
        print*, ' N, NNZ = ', N, NNz
        write(10,*)  ' N, NNZ = ', N, NNz
        k = 1
        l = 0
        ia2(1) = 1
        sumk = 0.
400     continue
        read(8,*,end=1000) int1, int2, val
        l = l + 1
        if(int2 .eq. k) then
          a2(l) = val
          ja2(l) = int1
          sumk = sumk + dabs(val)
        else
          if(sumk .eq. 0.) print*,' -- row ',k,' is zero'
          a2(l) = val
          ja2(l) = int1
          k = k + 1
          ia2(k) = l
          sumk = dabs(val)
        endif

        goto 400
1000    continue
        ia2(k+1) = l + 1
        print*,' -- # of Rows = ', k

        print*,' N, nnz, l = ', N, nnz, l

        call csrcsc(N, 1, 1, a2, ja2, ia2, a, ja, ia)
         call csrcsc(N, 1, 1, a, ja, ia, a2, ja2, ia2)
```

```
1100 continue

icase = icase + 1



!       if(isym .eq. 1) then
!         call apmbt (n, n, 1, a, ja, ia, a, ja, ia, a2, ja2, ia2,
!     1     maxnz, iw, ierr)
!         print*,' ierr from apmbt = ', ierr
!  do i = 1, N
!    ia(i) = ia2(i)
!    do j = ia2(i), ia2(i+1)-1
!      ja(j) = ja2(j)
!      a(j) = a2(j)
!      if(ja(j) .eq. i) a(j) = a(j) / 2.0
!    enddo
!   enddo
!
! endif

!
!   do i = 1 , N
!     sol(i) = mod(i,3)
!            x(i) = 0.
!          enddo
!
!   call amux(n,sol,rhs,a,ja,ia)

!c errk = 0.
!c do 1077 i = 1 , N
!c   errk = errk + rhs(i)**2
!c1077 continue
```

```
!c print*,' ** Errk = ',sqrt(errk)
!c write(10,*)' ** Errk = ',sqrt(errk)
!c
!c if(idebug1 .gt. 1) then
!c write(12,*) ' ** Matrix Dump **'
!c  do 1080 i = 1 , N
!c    write(12,*)i,ia(i),ia(i+1)-1
!c    write(12,*)(ja(k),a(k),k=ia(i),ia(i+1)-1)
!cc7717   format(9(i2,1x,f5.2,1x))
!c1080 continue
!c endif
!c
!c call compact(n,a,ja,ia,alu,jlu,ju)
!c
!c time0 = second()
!cC  Convert a matrix in CSR format into JAD(Jagged Diagonal) Form
!c call csrjad(N, a, ja, ia, idiag, iperm, a2, ja2, ia2)

  time0 = second()
do 1090 i = 1 , (Ndiag+Maxlfil)*N
  alu(i) = 0.
  jlu(i) = 0
1090 continue
do 1095 i = 1 , N
  ju(i) = 0
1095 continue

tt1 = second()

! print*,' ** Icm64, Imc64', icm64, imc64
! if(icm64 == 0) goto 3456
! if(imc64 == 1) goto 1430

print*, '** Point 1 **'
```

```
call mc64id(icntl)

!  do job = 1, 5
job = 3
write(10, *) ' ** Icase, job = ', Icase, job

call mc64ad (job, n, nnz, ia2, ja2, a2, num, colperm, liw,  &
            iw, ldw, dw, icntl, info)

print*,' info(1), num = ', info(1), num
if(info(1) .lt. 0) write(*,*) ' -- Abnormal return from &
                 mc64ad --'
if(info(1) .eq. 1) then
  write(*,*) ' -- Structually Singular **'
  ifail2 = ifail2 + 1
  goto 4500
endif
       !  enddo

if(info(1) .eq. 1) then
  write(10,*) ' -- Structually Singular **'
endif

       if(info(1) <   0 ) then
         ifail2 = ifail2 + 1
 goto 4500
       endif

!
!  Do the scaling
!
do j=1, N
  do k = ia2(j), ia2(j+1)-1
```

```
    i = ja2(k)
    a2(k) = a2(k)*exp(dw(i) + dw(N+j))
  enddo
enddo

do i = 1, N
  rhs(i) = rhs(i)*exp(dw(i))
enddo

do j = 1, N
  rowperm(j) = j
enddo

do j = 1, N
  lenc(j) = ia2(j+1) - ia2(j)
enddo

write(*,*) ' -- Before mc22ad --'
call mc22ad(n, ja2, a2, nnz, lenc, colperm, rowperm, iw(1), &
           iw(2*n+1))
write(*,*) ' -- After mc22ad --'

ia2(1) = 1
do j = 1, N
  ia2(j+1) = ia2(j) + lenc(j)
enddo

call csrcsc(n, 1, 1, a2, ja2, ia2, a, ja, ia)

print*,' -- before RCMK = ', Ndiag, maxlfil

NDD = 0
do i = 1, N
  Resti = 0.
```

```
  do j = ia(i), ia(i+1)-1
    if(ja(j) == i) then
      diag = abs(a(j))
    else
      Resti = Resti + abs(a(j))
    endif
  enddo

  if(Diag >= Resti) NDD = NDD + 1

enddo

print*,' ** Out of ', N, ' rows', NDD,' rows are DD'
write(10,*) ' ** Out of ', N, ' rows', NDD,' rows are DD'


!     ifirst = 1

1200 continue
time3 = second()
! call lredbl(n,ja,ia,ifirst,lfront,nfr,iord,riord,iL)

1400 continue

! if(Imc64 == 2) goto 1450

1430 continue

print*, '** Point 2 **'

call rcmk(n,ja,ia,ifirst,lfront,nfr,iord,riord,il)
print*,' -- nfr = ', nfr
trcmk = second() - time3
```

```
call dperm(n,a,ja,ia,a2,ja2,ia2,iord,iord,3)


  call csrcsc(n,1,1,a2,ja2,ia2,a,ja,ia)
  call csrcsc(n,1,1,a,ja,ia,a2,ja2,ia2)
call dvperm(n,rhs,iord)
print*,' -- after matrix permutation --'



!  do i = 1, N
!    ia(i) = ia2(i)
!    do j = ia2(i), ia2(i+1)-1
!      a(j) = a2(j)
!      ja(j) = ja2(j)
!    enddo
!   enddo
!   ia(N+1) = ia2(N+1)


! call csrcsc(n, 1, 1, a, ja, ia, a2, ja2, ia2)
! call csrcsc(n, 1, 1, a2, ja2, ia2, a, ja, ia)

3456 continue

print*, '** Point 3 **'

 do i = 1, N
   ia(i) = ia2(i)
   do j = ia2(i), ia2(i+1)-1
     a(j) = a2(j)
     if(Icase == 2) a(j) = -a(j)
     ja(j) = ja2(j)
   enddo
```

```
  enddo
  ia(N+1) = ia2(N+1)



1450 continue


if(Iprecon .eq. 4) then
  do i = 1, N
    do j = ia(i), ia(i+1)-1
      if(ja(j) .eq. i) ju(i) = j
    enddo
  enddo
endif



9333 continue
close(99)
close(8)

write(10,*) ' ** Icase = ', Icase, Ifail
if(icase == 2 .and. ifail2 == 2) write(*,*) filename, ' ** Failure
if(icase == 2 .and. ifail2 == 2) write(10,*) filename, ' ** Failur
close(10)
goto 1234

9876 continue
write(*,*) Ifail, ' failed cases --'
write(10,*) Ifail, ' failed cases --'

stop
end
```