399

400

401 # 12. APPENDIX I.: LISTING FOR SELECTION OF THE BEST QUALITY

402 ## SOUNDINGS IN FORTRAN LANGUAGE

```
403     !{************************************************}
404     SUBROUTINE SortBathy(NameFicSort,LoMin,LoMax,LaMin,LaMax &
405          ,NbFicBat,TabFic,TabQual,TabId,iMax,jMax,SaveAll)
406     !************************************************************
407     !
408     ! Fortran code for publication "Automatic calculation of bathymetry for coastal hydrodynamic
409 models"
410     ! Object    : Sort bathymetric soundings according quality factor and model resolution
411     !           read several raw bathymetric files, result is one sorted file with the resolution selected
412     ! File access are generic, they must be adapted to user rules
413     !
414     ! Delete soundings with the quality number higher than theses located in the same cell
415     ! (ResMeters gives values of NCol and Nlig for array dimensions
416     !
```

```
!*************************************************************
!*************************************************************************
!------- Historic:
! 18/09/2003 - P. Bailly du Bois : Creation
! 01/09/2010 - P. Bailly du Bois : Translation for publication
!*************************************************************************
!
 implicit none

! Variable declaration
 character * (*) NameFicSort ! name of output file
 integer NbFicBat ! number of bathymetry files
 character TabFic(NbFicBat) ! names of bathymetry files
 Integer TabQual(NbFicBat) ! quality of each bathymetry file
 Integer TabId(NbFicBat) ! indentification number of each bathymetry file
 Integer contFic ! bathymetry files counter
 real Lon,Lat,Depth ! Location and depth of each sounding
 integer imin,imax,jmin,jmax,i,j ! dimentions and references of the tables
 parameter (imin=1,jmin=1)
!  jmax=int( ((LaMax-LaMin)*60.*1852.)/ResolutionMeters )
!  imax=int( ((LoMax-LoMin)*60.*1852.*cos(DegRad*(LaMax+LaMin)*0.5) )/ResolutionMeters )
 real TabDepth(imin:imax,jmin:jmax) ! array of sounding values
 integer TabQF(imin:imax,jmin:jmax) ! array of quality factors
 integer TabNVal(imin:imax,jmin:jmax)! array of number of soundings
 real LoMin,LoMax,LaMin,LaMax ! limits of the area
```

```
442       real dLo,dLa ! mesh size in degrees
443       logical SaveAll ! For save in a file of all selected records
444
445      ! implementation
446       dLo=(LoMax-LoMin)/real(imax)
447       dLa=(LaMax-LaMin)/real(jmax)
448
449       ! Initialisation
450       do i=imin,imax
451         do j=jmin,jmax
452           TabDepth(i,j)=0.
453           TabNVal(i,j)=0
454           TabQF(i,j)=0
455         enddo
456       enddo
457
458       if (SaveAll) open(13,file=NameFicSort//'AllRecords') ! Save all data selected
459       do contFic=1,NbFicBat ! read each bathymetry files
460         open(12,file=TabFic(contFic))
461         do while (not(eof(12)))
462           read(12,*,end=200)Lon,Lat,Depth
463           if ( (Lat.ge.LaMin).and.(Lat.le.LaMax).and. &
464                (Lon.ge.LoMin).and.(Lon.le.LoMax) )  then ! in the area
465             i= nint( (Lon-LoMin)/dLo )
466             j= nint( (Lat-LaMin)/dLa )
```

```
467             if (TabNVal(i,j).eq.0) then ! cell empty
468               TabNVal(i,j)=1
469               TabDepth(i,j)=Depth
470               TabQF(i,j)=TabQual(contFic)
471             else ! mesh with sounding
472               if (TabQual(contFic).eq.TabQF(i,j)) then ! quality is the same
473                 TabDepth(i,j)=(TabDepth(i,j)+Depth)
474                 TabNVal(i,j)=TabNVal(i,j)+1
475                 if (SaveAll) write(13,*) Lon,Lat,Depth,TabId(contFic)
476               endif
477             endif
478           endif ! in area
479         enddo ! FicBat
480       200 close(12)
481         if (SaveAll) close (13)
482         enddo ! contFic
483
484         open(14,file=NameFicSort) ! write result of sorting
485           do i=imin,imax
486             do j=jmin,jmax
487               if (TabNVal(i,j).gt.0) then
488                 Lon=(real(i)*dLo)+LoMin
489                 Lat=(real(j)*dLa)+LaMin
490                 write(14,*) Lon,Lat,TabDepth(i,j)/TabNVal(i,j)
491               endif
```

```
492             enddo
493           enddo
494         close(14)
495       end SUBROUTINE SortBathy
496       !{**************************************************}
```

13. APPENDIX II. : LISTING FOR ELIMINATION OF ON LAND RECORDS IN FORTRAN LANGUAGE

```
499       !************************************************************
500       SUBROUTINE ShorelineMask(NameShoreline,NameBathy,NameBathyOut, &
501                       LoMin,LoMax,LaMin,LaMax,NCol,Nlig,LoInit,LaInit)
502       !************************************************************
503       !
504       ! Fortran code for publication "Automatic calculation of bathymetry for coastal hydrodynamic
505   models"
506       ! Object    : Sort bathymetric soundings located on land
507       ! Using a mask with shoreline file
508       ! File access are generic, they must be adapted to user rules
509       !
510       ! Parameters transmitted:
511       ! NameShoreline: shoreline (or other) limit file for the area to mask
512       ! NameBathy: bathymetry file to sort with the mask
513       ! NameBathyOut: result bathymetry file
514       ! LoMin,LoMax,LaMin,LaMax: Area covered by the mask array
515       ! NCol,Nlig: dimensions of the mask array, to parameter with resolution wanted
```

```
516     ! NLig=int( ((LaMax-LaMin)*60.*1852.)/ResMeters )
517     ! NCol=int( ((LoMax-LoMin)*60.*1852.*cos(DegRad*(LaMax+LaMin)*0.5) )/ResMeters )
518     ! LoInit,LaInit: point located in the area to fill (at sea), for filling initialisation
519     !
520     ! call : Hole, LineEqu
521     !
522     !**************************************************************
523     !******************************************************************************
524     !------- Historic:
525     ! 06/10/2003 - P. Bailly du Bois : Creation
526     ! 01/09/2010 - P. Bailly du Bois : Translation for publication
527     !******************************************************************************
528     !
529       implicit none
530
531     ! Variable declaration
532       character * (*) NameShoreline,NameBathy,NameBathyOut
533       integer NbMaxHoles
534       parameter (NbMaxHoles = 40000000) ! could be defined dynamicaly with Ncol*Nlig
535       real LoMin,LoMax,LaMin,LaMax,dLo,dLa,LoInit,LaInit
536       integer*4 NCol,Nlig
537       integer*4 Count,NbPoints
538       Integer i,j,k
539       real PosXr,PosYr,PosXrPrec,PosYrPrec,dx,dy,da,db,x,y
540       Integer PosXi,PosYi,PosXiPrec,PosYiPrec,xi,yi,cx,cy,pas,xiPrec,yiPrec
```

```
541         integer Pos, PosPrec,NbHoles,xii,yii
542         integer PtsHoles(1:2,1:NbMaxHoles) ! PtsHoles(1,n)=x, PtsHoles(1,n)=y in the array
543         logical IsHole(0:3),output,CulSac,CulSacPrec,Hole
544         real Long,Lat,Prof
545         Integer*1 Tabl(NCol,Nlig),ValShoreLine,ValSea,ValNul
546         parameter (ValNul=0,ValShoreLine=1,ValSea = 2)
547         integer CountFill
548         logical Filling
549
550         ! Implementation
551         dLo=(LoMax-LoMin)/real(NCol)
552         dLa=(LaMax-LaMin)/real(Nlig)
553         do i=1,NCol ! array initialisation
554           do j=1,Nlig
555             Tabl(i,j)=ValNul
556           enddo
557         enddo
558
559         Count=0
560         open(13,file=NameShoreline)
561           do while (not(eof(13)))
562           read(13,*,end=100)NbPoints ! number of points in a closed section (island, continent)
563           do Count=1,NbPoints
564             read(13,*)Long,Lat
565             PosXi=nint((Long-LoMin)/dLo)
```

```
566          PosYi=nint((Lat-LaMin)/dLa)
567          if ((Count.ne.1).and. ((PosXi.ne.PosXiPrec).or.(PosYi.ne.PosYiPrec) ) ) then
568           call LineEqu(da,db,PosXr,PosYr,PosXrPrec,PosYrPrec) ! find parameters of the line between
569 two points
570           if(PosXi.gt.PosXiPrec) then
571            pas=1
572           else
573            pas=-1
574           endif
575           do cx=PosXiPrec+1,PosXi,pas ! fill the line between two points
576            x=cx
577            xi=cx
578            if( PosYi.eq.PosYiPrec) then
579             yi=PosYi
580            else
581             yi=int(da*x+db)
582             if (yi.lt.min(PosYi,PosYiPrec)) yi=min(PosYi,PosYiPrec)
583             if (yi.gt.max(PosYi,PosYiPrec)) yi=max(PosYi,PosYiPrec)
584            endif
585            if ( (xi.ge.1).and.(xi.le.NCol).and.(yi.ge.1) &
586                  .and.(yi.le.NLig) ) Tabl(Xi,Yi)=ValShoreLine
587            xi=cx-1
588            if ( (xi.ge.1).and.(xi.le.NCol).and.(yi.ge.1) &
589                  .and.(yi.le.NLig) ) Tabl(Xi,Yi)=ValShoreLine
590           enddo
```

```
591             if(PosYi.gt.PosYiPrec) then
592               pas=1
593             else
594               pas=-1
595           endif
596           do cy=PosYiPrec+1,PosYi, pas
597             y=cy
598             yi=cy
599             if( PosXi.eq.PosXiPrec) then
600               xi=PosXi
601             else
602               xi=int((y-db)/da)
603               if (xi.lt.min(PosXi,PosXiPrec)) xi=min(PosXi,PosXiPrec)
604               if (xi.gt.max(PosXi,PosXiPrec)) xi=max(PosXi,PosXiPrec)
605             endif
606               if ( (xi.ge.1).and.(xi.le.NCol).and.(yi.ge.1) &
607                     .and.(yi.le.NLig) ) Tabl(Xi,Yi)=ValShoreLine
608             yi=cy-1
609               if ( (xi.ge.1).and.(xi.le.NCol).and.(yi.ge.1) &
610                     .and.(yi.le.NLig) ) Tabl(Xi,Yi)=ValShoreLine
611           enddo
612         endif
613         PosXiPrec=PosXi
614         PosYiPrec=PosYi
615         PosXrPrec=PosXr
```

```
616             PosYrPrec=PosYr
617            enddo
618           enddo
619          100  close(13)
620          ! End of shoreline
621           xi=nint((LoInit-LoMin)/dLo) ! Position of the initial point at sea for filling
622           yi=nint((LaInit-LaMin)/dLa)
623           xiPrec=xi
624           yiPrec=yi
625           NbHoles=1
626           PtsHoles(1,NbHoles)=xi
627           PtsHoles(2,NbHoles)=yi
628           PosPrec=1
629           CountFill=0
630           Filling=.true.
631           do while (Filling) ! Fill the area at sea
632            output=.false.
633            CulSac=.true.
634            PosPrec=imod(PosPrec+2,4)
635            do k=PosPrec+1,PosPrec+3 ! look in the cells around the current cell except the preceding
636             Pos=imod(k,4)
637             IsHole(Pos)=Hole(xi,yi,Pos,i,j,1,NCol,1,Nlig,Tabl)
638             if (IsHole(Pos).and.output) CulSac=.false.
639             if (IsHole(Pos).and.CulSac) then
640              output=.True.
```

```
641             PosPrec=Pos
642             xii=i
643             yii=j
644           endif
645         enddo
646         if (not(CulSac)) then
647           PtsHoles(1,NbHoles)=xi
648           PtsHoles(2,NbHoles)=yi
649         endif
650         if (CulSac.and.not(CulSacPrec)) then
651           NbHoles=NbHoles+1
652           if (NbHoles.gt.NbMaxHoles) then
653             write(*,*)'Too much Holes to fill, change parameter NbMaxHoles'
654             Filling=.false.
655           endif
656         endif
657         if (output) then
658           xi=xii
659           yi=yii
660           Tabl(Xi,Yi)=ValSea
661           CountFill=CountFill+1
662         else
663           if (NbHoles.gt.1) then
664             NbHoles=NbHoles-1
665             xi=PtsHoles(1,NbHoles)
```

```
666             yi=PtsHoles(2,NbHoles)
667            else
668             Filling=.false.
669            endif
670           endif
671           CulSacPrec=CulSac
672          enddo ! end filling
673
674          open(13,file=NameBathy)
675          open(14,file=NameBathyOut)
676          do while (not(eof(13))) ! Select soundings at sea
677           read(13,*,end=200)Long,Lat,Prof
678           PosXi=nint((Long-LoMin)/dLo)
679           PosYi=nint((Lat-LaMin)/dLa)
680           if ( (PosXi.ge.1).and.(PosXi.le.NCol).and.(PosYi.ge.1) &
681                    .and.(PosYi.le.NLig) ) Then
682            if (Tabl(PosXi,PosYi).eq.ValSea) then
683             write(14) Long,Lat,prof
684            endif
685           else
686            write(14) Long,Lat,prof
687           endif
688          enddo
689         200  close(13)
690          close(14)
```

```
691         return
692       end SUBROUTINE ShorelineMask
693       !{************************************************}
694
695       !{************************************************}
696       SUBROUTINE LineEqu(a,b,x1,y1,x2,y2)
697       !************************************************************
698                 real a,b,x1,y1,x2,y2
699
700          if ((x1-x2).ne.0.) then
701                   a=(y1-y2)/(x1-x2)
702                   b=y1-a*x1
703                 else
704                   a=1.7014100E+038
705                   b=0.
706                 endif
707                 return
708       end SUBROUTINE LineEqu
709       !{************************************************}
710
711       !{************************************************}
712       logical FUNCTION Hole(xi,yi,Pos,i,j,iMin,NCol,jMin,Nlig,Tabl)
713       !************************************************************
714          implicit none
715          integer iMin,NCol,jMin,Nlig
```

```
716       integer*1 Tabl(iMin:NCol,jMin:Nlig),ValNul
717       parameter (ValNul=0)
718       integer i,j,xi,yi,Pos,a
719
720       if (pos.eq.0) then
721         i=xi+1
722         j=yi
723       endif
724       if (pos.eq.1) then
725         i=xi
726         j=yi-1
727       endif
728       if (pos.eq.2) then
729         i=xi-1
730         j=yi
731       endif
732       if (pos.eq.3) then
733         i=xi
734         j=yi+1
735       endif
736       if ( (i.ge.iMin).and.(i.le.NCol).and.(j.ge.jMin).and.(j.le.NLig) ) then
737         if (Tabl(i,j).eq.ValNul) then
738           Hole=.true.
739         else
740           Hole=.false.
```

```
741           endif
742          else ! outside the array
743           Hole=.false.
744          endif
745          return
746        end FUNCTION Hole
747        !{**************************************************}
```